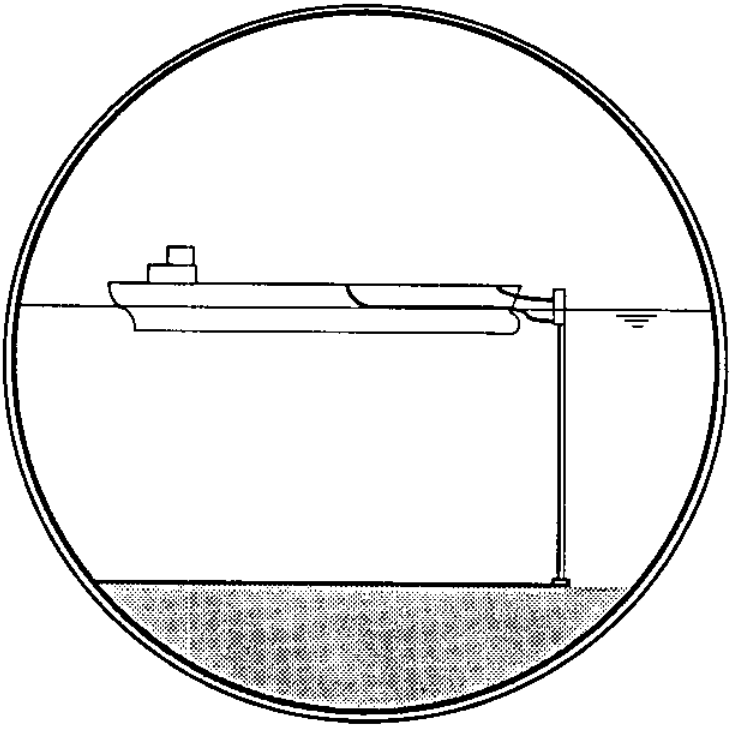
An Analytical Approach to Cable Dynamics

Theory and User Manual

SEA GRANT PROJECT R/OE-26



by :
Wei Ma

and :
Professor William C. Webster

Contents

Chapter 1 **Introduction** 1

Chapter 2 **Statement of the Problem** 3

2.1 Basic Assumptions 3

2.2 Governing Equations 3

2.3 Formulation of External and Internal Forces 5

2.4 Numerical Implementation 9

2.5 The Static Problem 13

2.6 The Dynamic Problem 14

2.7 The Ocean Bottom Boundary Condition 16

2.8 Energy Dissipation 17

Chapter 3 **Results and Discussions** 18

3.1 Validation 18

3.2 Analysis of a Catenary Anchor Leg Mooring System 21

Chapter 4 **User Manual** 24

Chapter 5 **Conclusions** 40

Reference 41

Appendix I. **The Geometrical Relationships** 42

Appendix II. **Working Examples** 43

Appendix III. **The Source Code** 44

Chapter 1

Introduction

Most California business and industrial centers are located on the coast. Every year, California imports large volumes of hydrocarbons and chemicals (most of them crude oils) to meet the demands of these centers. A principal means for the importation is waterborne tanker transport. Yearly, approximately 1 billion barrels of crude oil and hydrocarbon products are transported into California harbors. Along with the surging of the population and the recovery of economy, California harbors are required to handle more and more incoming cargoes.

In order to transport this huge amount of cargo, some 2,500 large tankers are used. One third of these tankers are foreign flagged. Because of the limiting water depth of California ports, tankers are often too large to enter the ports, which results in using a large number of smaller tankers to unload the oil from the large tankers and then bring the oil into the ports. This operation is called lightering. By 1990, the volume of oil handled by lightering was 1.75 million barrels per day, which was more than twice the volume four years ago.

The use of a number of smaller tankers increases the possibility of collisions, grounding, operational casualties, fire/explosion, and overflows in the relatively shallow, narrow, congested and environmentally sensitive waterways. Oil spills are the direct result of these accidents. A report from the US Coast Guard *Deepwater Ports Study* (1993) indicates that lightering at sea and discharging in port. This is because lightering possesses all the possible conditions to generate the above mentioned accidents.

A Single Point Mooring System (SPMS) is an alternative way to lighter. A SPMS generally consists of a buoy, a mooring system, and a riser. It is usually located in water deep enough for Very Large Crude Carriers (VLCC) and Ultra Large Crude Carriers (ULCC). Lightering is no longer necessary with using of the SPMS. Thus, SPMS reduces the number of smaller tankers used. Also, the location of the SPMS is some distance from the coast, which provides some degree of environmental protection through the space between coast and the SPMS.

The research project *Offshore Single Point Mooring System for Import of Hazardous Liquids* was proposed with a view to lessen the effects of accidents. The objectives of the project are stated to be:

" A. To evaluate the feasibility and reliability of offshore Single Point Mooring System (SPMS) for two locations offshore California: EL Segundo and Morro Bay.

B. To develop and verify the analytical procedures required to characterize the forces, loadings, and motions developed by the components of SPMS when subjected to extreme storm (survival) conditions and major seismic events.

C. To use these procedures to engineer and evaluate the performance and reliability characteristics of alternative SPMS concepts."

This research project has been carried out by two approaches: an engineering approach and an analytical approach, which resulted in two final reports. The engineering approach focuses on the feasibility and reliability of the use of the SPMS in California offshore. The analytical approach provides the engineering approach with suitable analytical tools for determining the forces and motions of the components of the SPMS. This report is the analytical final report, in which one particular analytical problem, cable dynamics, is studied.

The research project was performed by two graduate students Mr. Wei Ma and Mr. Aaron Salancy under the direction of Prof. William C. Webster and Prof. Robert G. Bea. Mr. Ma was responsible for the work presented in this report while Mr. Salancy was responsible for the engineering approach report (Salancy, 1994).

This paper is funded in part by a grant from the National Sea Grant College Program, National Oceanic and Atmospheric Administration, U. S. Department of Commerce, under grant number NA36RG0537, project number R/OE-26 through California Sea Grant College, and in part by the California State Resources Agency. The views expressed herein are those of the authors and do not necessarily reflect the views of NOAA or any of its sub-agencies. The U. S. Government is authorized to reproduce and distribute for governmental purposes.

Chapter 2

Statement of the Problem

Consider an ocean cable or rod (being used alternately in this report) immersed fully in the sea. The cable is exposed to loadings from many sources. The primary ones are the effect of gravity on the mass of the cable, the hydrostatic forces on the cable (both due to the water outside the cable and to any fluids which may be inside the cable), and the hydrodynamic forces due to the cable motion and wave motion.

The complexity of the loadings, combined with the nonlinear geometrical characters of the rod, requires for a theory which is able to predict the hydrodynamic and the hydrostatic force on the rod and to evaluate the motion and the nonlinear restoring tension of the rod.

With this in mind, we have selected a theory for the dynamics of slender rods. The theory was first introduced by Garrett (1982) and then expanded by Paulling & Webster (1986). The theory is based on the Bernoulli beam theory and has a great flexibility in handling the rod without any geometric limitations.

2.1 Basic Assumptions

In dealing with the motions of a rod in the sea, we make the following assumptions:

- The rod is an elastic rod with equal inertia about its principal axes.
- The rod is fully immersed in the sea.
- The motion of the rod follows Bernoulli-Euler beam theory.
- There is no rotational inertia, no torque, and no shear deformation.

2.2 Governing Equations

Let a 3-D inertia Cartesian coordinate system be located at some place in the sea. Let the instantaneous configuration of a rod be expressed as a vector distance[1], $\mathbf{r}(s,t)$, from the origin of the coordinate system as a function of s, the arc length along the rod, and time t (see Figure 2.1).

[1] The bold face letters are used for vector and matrix quantities.

The model, which describes external forces and internal forces acting on a segment of a typical ocean rod, is given in figure 2.2. On the basis of conservation of linear momentum and moment of momentum, we have equation 2.1 and equation 2.2:

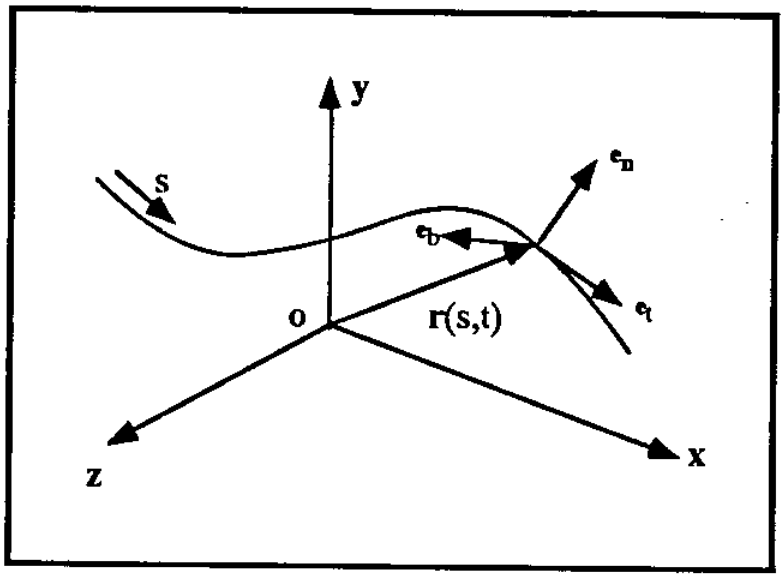


Figure 2.1 Schematic sketch of the inertia coordinate system

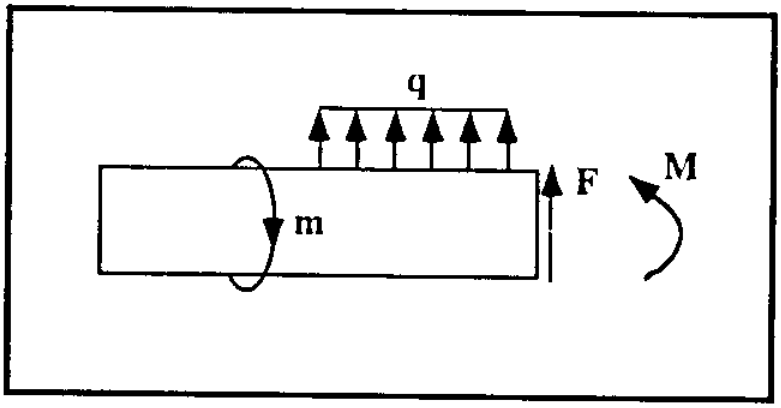


Figure 2.2 Theoretical model of a ocean rod

$$F' + q = \rho \ddot{r}(s,t) \tag{2.1}$$

$$M' + r' \times F + m = 0 \tag{2.2}$$

where q is the distributed external force per unit length, F is the shear force, M is the bending moment, and m is the twisting moment per unit length.

By Bernoulli-Euler beam theory, the bending moment M and its derivative[2] with respective to arc length s can be expressed as:

$$M = r' \times (Br'') + Hr' \tag{2.3}$$

$$M' = r' \times (Br'')' + H'r' + Hr'' \tag{2.4}$$

where H is the torque and B is the bending rigidity. Assume H = 0 and $m = 0$, and plug M' into equation 2.2, this gives the expression of F:

$$F = \lambda r' - (Br'')' \tag{2.5}$$

where $\lambda = T - B\kappa^2$, is a scalar variable, κ is the local curvature of the rod, and T(s,t), $= r' \cdot F$, is the local tension.

The equation of motion is finally obtained by plugging F into equation 2.1:

$$-(Br'')'' + (\lambda r')' + q = \rho \ddot{r}(s,t) \tag{2.6}$$

where q is the distributed external loadings and ρ is the mass density of the water.

The derivation of equation 2.6 was made possible by the assumptions made in section 2.1. See Appendix I for the some detailed derivations.

2.3 Formulation of External and Internal Forces

As was mentioned in the previous sections, the external and internal forces on the rod are:

- Gravity force on the mass of the rod.
- The hydrostatic forces on the rod (both due to the water outside of the rod and to any fluids which may inside the rod[3]).
- The hydrodynamic forces due to the rod motion and wave motion.

---

[2] The superscript prime, ', is used to represent the partial derivative with respect to arc length s. The superscript dot, . , is used to represent the time derivative.

[3] A mooring line used in the marine applications is usually not hollow. The assumption that the rod may be hollow expands the application of the theory.

The geometrical definition of the rod is given in figure 2.3.

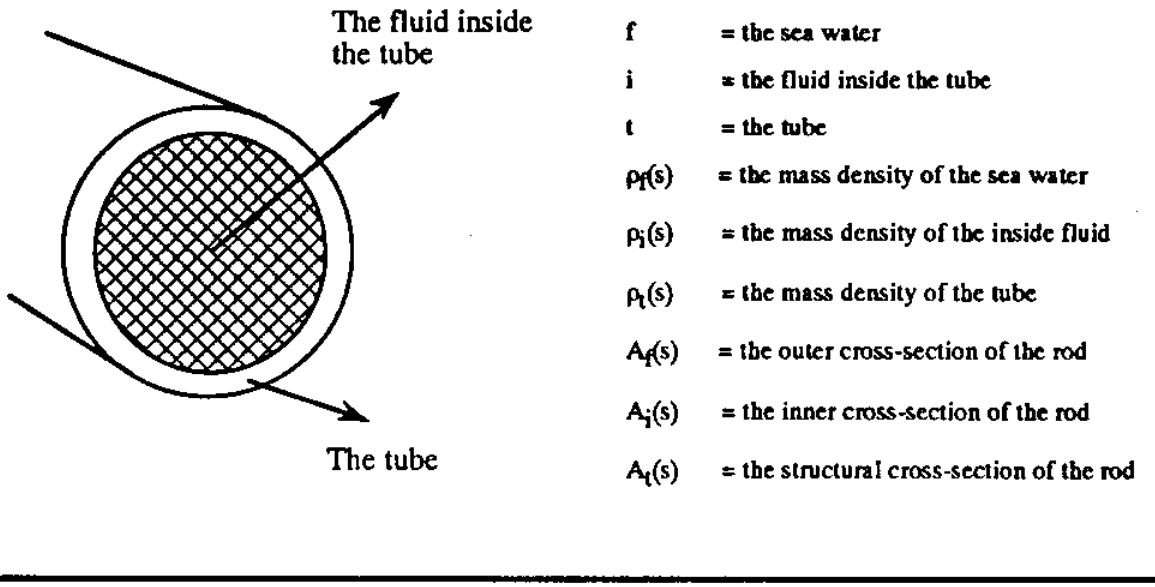


Figure 2.3 Geometrical definition of the rod

The effect of gravity on the mass of the rod leads to a distributed load given by:

$$q_t(s) = -\rho_t g(A_f - A_i)e_y \tag{2.7}$$

The hydrodynamic forces acting on the rod consist of an added-mass force, a drag force, and a Froude-Krylov force. The Morrison equation is used to predict the first two terms:

$$q_f^I(s,t) = \rho_f A_f C_M N(\dot{V}_f - \ddot{r}) \tag{2.8a}$$

$$q_f^D(s,t) = \frac{\rho_f}{2} D_f C_D N(V_f - \dot{r})\left|N(V_f - \dot{r})\right| \tag{2.8b}$$

where C_M and C_D are added-mass coefficient and drag coefficient andV_f is the velocity of the wave motion. N is a transfer matrix defined by:

$$N = I - (r')^T \otimes r' \tag{2.8c}$$

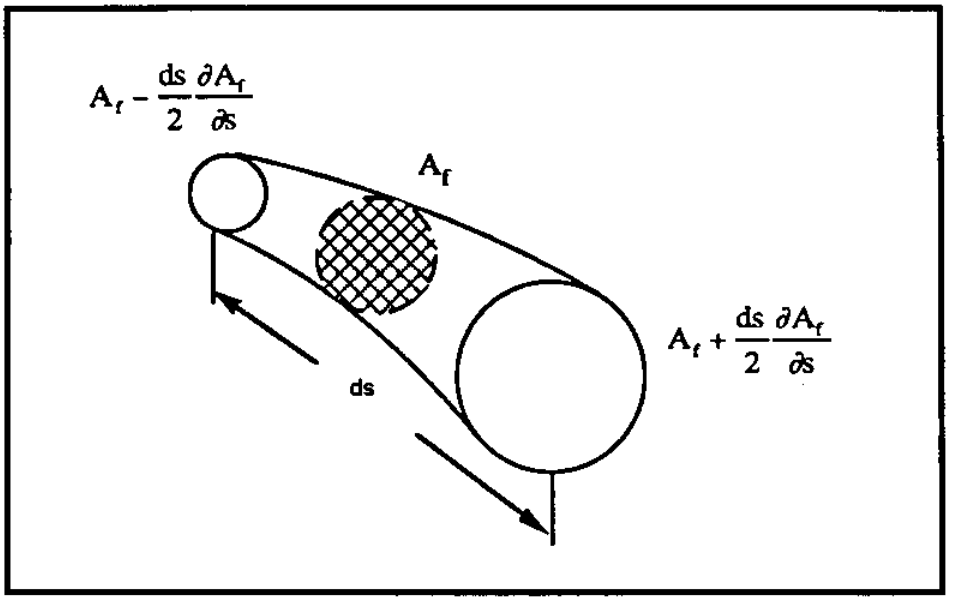


Figure 2.4 Schematic of the segment

The Froude-Krylov force arises from water pressure on the surface of the rod. Therefore, it is sometimes called generalized buoyancy force. Considering a segment of the rod shown in figure 2.4, the Froude-Krylov force on this segment is the summation of the buoyancy force and the acceleration force (high order terms are neglected). Remember that the two ends of the segment are not exposed to the fluid. Thus, minus the surface pressure on the ends from the Froude-Krylov forces of the segment with two ends exposed to the fluid, we have the Froude-Krylov force due to the sea water outside the rod:

- Buoyancy: $\rho_f A_f g e_y$
- Pressure due to wave acceleration: $\rho_f A_f \dot{v}_f$
- Pressure at two ends of the segment:

$$\text{pressure at } A_f - \frac{ds}{2}\frac{\partial A_f}{\partial s}: P_f A_f r' - \frac{ds}{2}\frac{\partial (P_f A_f r')}{\partial s}$$

$$\text{pressure at } A_f + \frac{ds}{2}\frac{\partial A_f}{\partial s}: -P_f A_f r' - \frac{ds}{2}\frac{\partial (P_f A_f r')}{\partial s}$$

where Pf is the pressure at the mid-section of the segment.

Subtract the pressure at ends from the summation of the buoyancy force and acceleration force, we have:

$$q_f^{F\text{-}K} = \rho_f(g\mathbf{e}_y + \dot{\mathbf{v}}_f)A_f + (P_f A_f \mathbf{r}')' \tag{2.9a}$$

Similarly, we have the Froude-Krylov force for the fluid inside the rod:

$$q_i^{F\text{-}K} = -\rho_i g A_i \mathbf{e}_y - (P_i A_i \mathbf{r}')' \tag{2.9b}$$

The governing equation is finally obtained by substituting equation 2.7, 2.8 and 2.9 into equation 2.6:

$$M\ddot{\mathbf{r}} + (B\mathbf{r}'')'' - (\tilde{\lambda}\mathbf{r}')' = \begin{cases} \rho_f A_f (I + NC_m)\dot{\mathbf{v}}^{fluid} \\ \quad + \dfrac{1}{2} D\rho_f C_d \left|\mathbf{v}_n^{rel}\right| \mathbf{v}_n^{rel} \\ + g[\rho_f A_f - \rho_i A_i - \rho_t A_t]\mathbf{e}_y \end{cases} \tag{2.10a}$$

with a stretch constrained equation

$$\mathbf{r}' \cdot \mathbf{r}' = 1 + 2\frac{T}{EA_t} \tag{2.10b}$$

Equation 2.10b is a statement of the extendibility of the rod. The definitions of the terms used in equation 2.10 are given below. Some of the definitions were given before. The others were not.

Definition:

- A_f : the outer cross-section of the rod,
- A_i : the inner cross-section of the rod (if it is hollow),
- A_t : = (A_f-A_i), the structural cross-section of the rod,
- B : = EI, the bending rigidity of the rod,
- C_d : the drag coefficient,
- C_m : the added-mass coefficient,
- M : = $(\rho_t A_t + \rho_i A_i)\mathbf{I} + \rho_f A_f C_m \mathbf{N}$, the virtual mass matrix,
- N : = $\mathbf{I} - \mathbf{r}'\mathbf{r}'$, the normal operator,
- T : the tension,
- $\dot{\mathbf{v}}^{fluid}$: the fluid acceleration at s,

$\mathbf{v}_n^{rel}$: the normal velocity of the rod at s with respective to the sea water,

ρ_f : the mass density of the sea water,

ρ_i : the mass density of the liquid inside the rod,

ρ_t : the mass density of the rod,

$\tilde{\lambda}$: $= (T + P_f A_f - P_i A_i) - B\kappa^2$, the effective tension.

2.4 Numerical Implementation

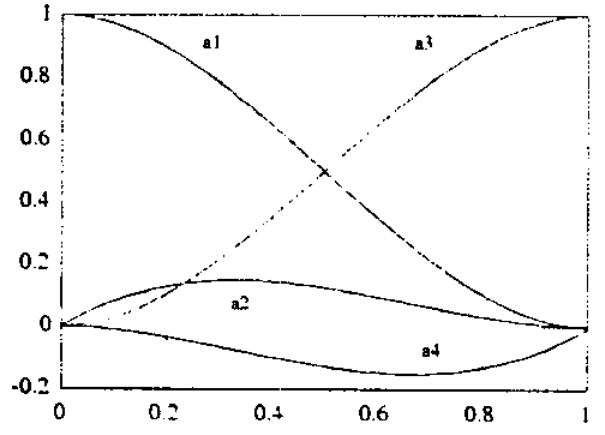
The purpose of this section is to discuss the numerical implementation of the governing equations derived in the previous section. A finite element solution method is employed to discretize the vector governing equations into algebraic equations. The method, based on the Galerkin's method, uses a set of shape functions to approximate the cable and the variations of the tensions (and other parameters) along it. By using this method, the original vectored equations are converted into 15 scalar equations with 15 unknowns.

Two sets of shape functions are used. One set of shape functions, a(s), describes the shape of the rod. The other set, $\Psi(s)$, describes the variation of λ and other material properties. Therefore, the position of the rod in the inertia coordinate system, the effective tension, and the distributed loadings are approximated as (summation rule is employed):

$$\begin{aligned} r(s,t) &= U_{in}(t)a_i(s)e_n \\ \lambda(s,t) &= \lambda_m(t)\psi'_m(s) \\ q(s,t) &= q_{mn}(t)\psi'_m(s)e_n \end{aligned} \tag{2.11}$$

where n=1,3, i=1,4 and m=1,3.

Shape functions a(s) and Ψ(s) are selected to be:



Shape Function a(s):

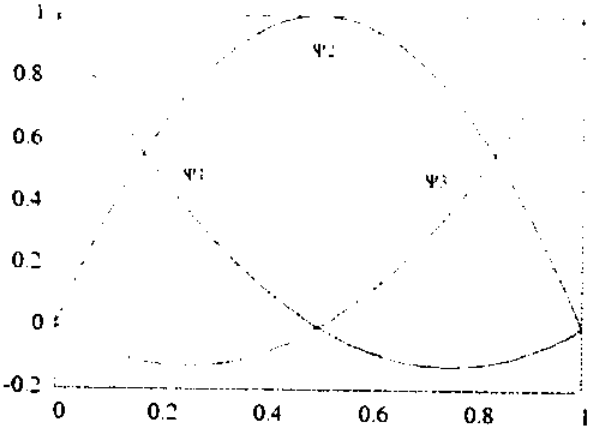
$$a_1(s) = 1 - 3(\frac{s}{L})^2 + 2(\frac{s}{L})^3$$

$$a_2(s) = \frac{s}{L} - 2(\frac{s}{L})^2 + (\frac{s}{L})^3$$

$$a_3(s) = 3(\frac{s}{L})^2 - 2(\frac{s}{L})^3$$

$$a_4(s) = -(\frac{s}{L})^2 + (\frac{s}{L})^3$$

Figure 2.5 Shape function a(s)



Shape Function Ψ(s):

$$\psi_1(s) = 1 - 3(\frac{s}{L}) + 2(\frac{s}{L})^2$$

$$\psi_2(s) = 4(\frac{s}{L})(1 - (\frac{s}{L}))$$

$$\psi_3(s) = (\frac{s}{L})(2(\frac{s}{L}) - 1)$$

Figure 2.6 Shape function Ψ(s)

We now use U_i, λ_i, and q_i to represent the nodal and mid-section values:

$$\begin{array}{lll} U_{1n}(t) = r_n(0,t) & \lambda_1(t) = \lambda(0,t) & q_1(t) = q(0,t) \\ U_{2n}(t) = Lr_n'(0,t) & & \\ U_{3n}(t) = r(L,t) & \lambda_2(t) = \lambda(\frac{L}{2},t) & q_2(t) = q(\frac{L}{2},t) \\ U_{4n}(t) = Lr_n'(L,t) & \lambda_3(t) = \lambda(L,t) & q_3(t) = q(L,t) \end{array} \tag{2.12}$$

It turns out that variables, such as $r(s,t)$, $q(s,t)$, and $\lambda(s,t)$, etc., can be approximated by their nodal values and shape functions as follows:

$$\begin{aligned} r(s,t) &= r(0,t)a_1(s) + Lr'(0,t)a_2(s) + r(L,t)a_3(s) + Lr'(L,t)a_4(s) \\ \varepsilon(s,t) &= \varepsilon(0,t)\psi_1(s) + \varepsilon(\frac{L}{2},t)\psi_2(s) + \varepsilon(L,t)\psi_3(s) \\ \lambda(s,t) &= \lambda(0,t)\psi_1(s) + \lambda(\frac{L}{2},t)\psi_2(s) + \lambda(L,t)\psi_3(s) \\ B(s) &= B(0)\psi_1(s) + B(\frac{L}{2})\psi_2(s) + B(L)\psi_3(s) \\ q(s,t) &= q(0,t)\psi_1(s) + q(\frac{L}{2},t)\psi_2(s) + q(L,t)\psi_3(s) \\ M(s,t) &= M(0,t)\psi_1(s) + M(\frac{L}{2},t)\psi_2(s) + M(L,t)\psi_3(s) \end{aligned} \tag{2.13}$$

To obtain the discretized form of the governing equation, we rewrite equation 2.10a as follows:

$$M\ddot{r} + (Br'')'' - (\tilde{\lambda}r')' - q = 0 \tag{2.14}$$

Multiplying both side of the equation with a(s) and integrating it with respect to s from 0 to L for a segment (or an element) of the rod with length L:

$$\int_0^L \{M\ddot{r} + (Br'')'' - (\tilde{\lambda}r')' - q\}a_i(s)ds = 0$$

Integrating the above equation by parts results in equation 2.15:

$$\int_0^L \{M\ddot{r}a_i(s) + Br''a_i'(s) + \tilde{\lambda}r'a_i'(s) - qa_i(s)\}ds$$
$$= Br''a_i'\Big|_0^L + \{\tilde{\lambda}r' - (Br'')'\}a_i\Big|_0^L \tag{2.15}$$

The first term on the right-hand side of the equation are the bending moments at the ends of the element. The second boundary term are the forces at the ends. The right side of the equation will vanish when the natural boundary condition of the element is applied.

Inserting equations 2.11, 2.12 and 2.13 into equation 2.15, we obtain the discretized form of the equation of motion for an element of a rod with length L:

$$\gamma_{ikm}M_m\ddot{U}_{kn} + \alpha_{ikm}B_mU_{kn} + \beta_{ikm}\tilde{\lambda}_mU_{kn} = \mu_{im}q_{mn} \tag{2.16a}$$

where the summation algorithm is employed. i and k run from 1 to 4; m and n runs from 1 to 3; and α_{ikm}, β_{ikm}, γ_{ikm}, and μ_{im} represent:

$$\alpha_{ikm} = \frac{1}{L^3}\int_0^1 a_i''(\xi)a_k''(\xi)\psi'_m(\xi)d\xi$$
$$\beta_{ikm} = \frac{1}{L}\int_0^1 a_i'(\xi)a_k'(\xi)\psi'_m(\xi)d\xi$$
$$\gamma_{ikm} = L\int_0^1 a_i(\xi)a_k(\xi)\psi'_m(\xi)d\xi$$
$$\mu_{im} = L\int_0^1 a_i(\xi)\psi'_m(\xi)d\xi$$

Similarly, we obtain the discretized form of the equation for equation 2.10b:

$$\frac{1}{2}\beta_{ikm}U_{in}U_{kn} = \frac{1}{2}\{\tau_m + 2\eta_{km}\varepsilon_k\} \tag{2.16b}$$

where η_{km} and τ_m represent:

$$\eta_{km} = L\int_0^1 \psi'_i(\xi)\psi'_m(\xi)d\xi$$
$$\tau_m = L\int_0^1 \psi'_m(\xi)d\xi$$

For three dimensional motions, equation 2.16a corresponds to 12 scalar equations and equation 2.16b corresponds to 3 scalar equations. There are a total of 15 equations for one element of a rod with 15 unknowns. A long cable can be divided into a series of smoothly connected elements. For each element, we have 15 equations of motion. Assembling these equations results in an algebraic equation system which can be solved by one of the famous numerical schemes.

2.5 The Static Problem

For the static problem, all the terms related to the time derivatives are zero. Therefore, equation 2.16a is reduced to

$$\alpha_{ikm} B_m U_{kn} + \beta_{ikm} \tilde{\lambda}_m U_{kn} = \mu_{im} q_{mn} \tag{2.17}$$

The fixed-point iteration method is employed to solve the equations. Let u^0 and λ^0 be a first guess, then the new values of U and λ are

$$U_{kn} = U^0_{kn} + \delta U_{kn}$$

$$\tilde{\lambda}_m = \tilde{\lambda}^0_m + \delta \tilde{\lambda}_m$$

Plugging the new expression of U and λ into equation 2.17 and equation 2.16b, and discarding all the high order terms, the static problem to be solved is then represented by the following equations:

$$\alpha_{ikm} B_m \delta U_{kn} + \beta_{ikm} \tilde{\lambda}^0_m \delta U_{kn} + \beta_{ikm} \delta \tilde{\lambda}_m U^0_{kn} = \mu_{im} q_{mn} - \alpha_{ikm} B_m U^0_{kn} - \beta_{ikm} \tilde{\lambda}^0_m U^0_{kn}$$

$$\beta_{ikm} U^0_{in} \delta U_{kn} - \eta_{lm} \{ \frac{\delta \tilde{\lambda}}{A_l E} \Big|_l - \rho g \frac{\delta y}{A_l E} \Big|_l \} = \frac{1}{2} (\tau_m + 2\eta_{lm} \epsilon^0_l - \beta_{ikm} U^0_{in} U^0_{kn}) \tag{2.18ab}$$

The terms in the bracket on the left hand side of equation 2.18b result from the deviation of the stretch, where $\delta y|$ represent:

$$\delta y_1 = \delta U_{12}$$
$$\delta y_2 = 0.5\delta U_{12} + 0.125\delta U_{22} + 0.5\delta U_{32} - 0.125\delta U_{42}$$
$$\delta y_3 = \delta U_{32}$$

Equation 2.18 follows the form $Ax = B$, where the unknown x consists of the 15 unknowns mentioned above. Gauss elimination is used to solve the equation system. However, the tensor expression of equation 2.18 is inconvenient for algebraic solution. Thus, a renumbering operation is performed. The unknown x is rearranged as :

$$\{\delta U_{11}, \delta U_{12}, \delta U_{13}, \delta U_{21}, \delta U_{22}, \delta U_{23}, \delta\lambda_1, \delta\lambda_2, \delta U_{31}, \delta U_{32}, \delta U_{33}, \delta U_{41}, \delta U_{42}, \delta U_{43}, \delta\lambda_3\}$$

2.6 The Dynamic Problem

As done in the static problem, we conclude that the dynamic equation of the motion of a rod is of the form $M\ddot{U} = c$, where vector c consists of all the terms which are not time derivatives. We use the Adams-Moulton method to solve the dynamic problem.

Let h be time step from $t=n$ to $t=n+1$. We can approximately represent M, $\ddot{U}$, and c by

$$\ddot{u}_k = \frac{\delta\dot{u}_k}{h}, c_i = \frac{c_i^{(n+1)} + c_i^{(n)}}{2}, M_m = \frac{M_m^{(n+1)} + M_m^{(n)}}{2} \tag{2.19}$$

where Taylor's expansion was employed to obtain the values at t = n+1:

$$M_m^{(n+1)} = M_m^{(n)} + J_{mk}^{(n+\frac{1}{2})}\delta u_k^{(n)}$$
$$c_i^{(n+1)} = c_i^{(n)} + {}_{11}K_{ik}^{(n+\frac{1}{2})}\delta u_k^{(n)} + {}_{12}K_{im}^{(n+\frac{1}{2})}\delta\lambda_m^{(n)}$$
$$J_{mk}^{(n+\frac{1}{2})} = \frac{\partial M_m}{\partial u_k}, {}_{11}K_{ik}^{(n+\frac{1}{2})} = \frac{\partial c_i}{\partial u_k}, {}_{12}K_{im}^{(n+\frac{1}{2})} = \frac{\partial c_i}{\partial \lambda_m}$$
$$u_k^{(n+\frac{1}{2})} = u_k^{(n)} + \frac{h}{2}\dot{u}_k^{(n)}, \lambda_m^{(n+\frac{1}{2})} = \frac{3\lambda_m^{(n)} - \lambda_m^{(n-1)}}{2}, \delta u_k^{(n)} = h\dot{u}_k^{(n)} + \frac{h}{2}\delta\dot{u}_k^{(n)}$$

With the help of equation 2.19, equation 2.16a can be written in the following form:

$$\{\frac{h^2}{4}{}_{11}K_{ik}^{(n+\frac{1}{2})} + \gamma_{ikm}(M_m^{(n)} + \frac{J_{mk}^{(n+\frac{1}{2})}\dot{u}_k^{(n)}h}{2})\}\{\frac{1}{h}\delta\dot{u}_k^{(n)}\} + \{\frac{h^2}{4}{}_{12}K_{im}^{(n+\frac{1}{2})}\}\{\frac{2}{h^2}\delta\lambda_m^{(n)}\} = c_i^{(n)} + \frac{h}{2}{}_{11}K_{ik}^{(n+\frac{1}{2})}\dot{u}_k^{(n)} \tag{2.20}$$

Equation 2.16b is of the form $d_j = 0$. The value of d_j at the end of any time step can be expressed by the value at the beginning of the time step plus high order terms. Setting d_j equal to zero at the new time step results in the following constraint equation for j =1, 3:

$$d_j^{n+1} = d_j^n + {}_{21}K_{jk}^{(n+\frac{1}{2})}\delta u_k + {}_{22}K_{jm}^{(n+\frac{1}{2})}\delta\lambda_m = 0 \tag{2.21}$$

Equation 2.21 is expanded to be of the form:

$$\{\frac{h^2}{4}{}_{21}K_{jk}^{(n+\frac{1}{2})}\}\{\frac{1}{h}\delta\dot{u}_k^{(n)}\} + \{\frac{h^2}{4}{}_{22}K_{jm}^{(n+\frac{1}{2})}\}\{\frac{2}{h^2}\delta\lambda_m^{(n)}\} = \frac{d_j^{(n)}}{2} + \frac{h}{2}{}_{21}K_{jk}^{(n+\frac{1}{2})}\dot{u}_k^{(n)} \tag{2.22}$$

where

$${}_{21}K_{jk}^{(n+\frac{1}{2})} = \frac{\partial d_j}{\partial u_k}\,{}_{22}K_{jm}^{(n+\frac{1}{2})} = \frac{\partial d_j}{\partial \lambda_m}$$

Equation 2.20 and 2.22 are solved to obtain the increment of the generalized velocity, $\dot{u}$, and the effective tension λ. The values of u and λ at the new time are obtained by the following approach:

$$\begin{aligned} u_k^{(n+1)} &= u_k^{(n)} + \frac{h}{2}\{\delta\dot{u}_k^{(n)} + 2\dot{u}_k^{(n)}\} \\ \lambda_m^{(n+1)} &= \lambda_m^{(n)} + \delta\lambda_m^{(n)} \end{aligned} \tag{2.23}$$

The values of u and λ at time t = n+1 serve as a starting point for the next calculation.

2.7 The Ocean Bottom Boundary Condition

A great deal of attention has been paid to the ocean bottom boundary condition. Assume that the ocean bottom is flat and elastic. It would be reasonable to use an elastic layer, or a spring mat in this case, to represent the ocean bottom. Therefore, the distributed bottom support force can be written in the following form:

$$q^{Spring} = \begin{cases} \frac{S}{D}\{D - (r \cdot e_y - D_{btm})\} & D - (r \cdot e_y - D_{btm}) > 0 \\ 0 & \text{otherwise} \end{cases} \tag{2.24}$$

where D is the distance to the outer most fiber of the rod; D_{btm} is the ocean depth; and S represents:

$$S = \rho_t g A_t + \rho_i g A_i - \rho_f g A_f$$

A schematic of the ocean bottom represented by an elastic layer is shown in figure 2.7.

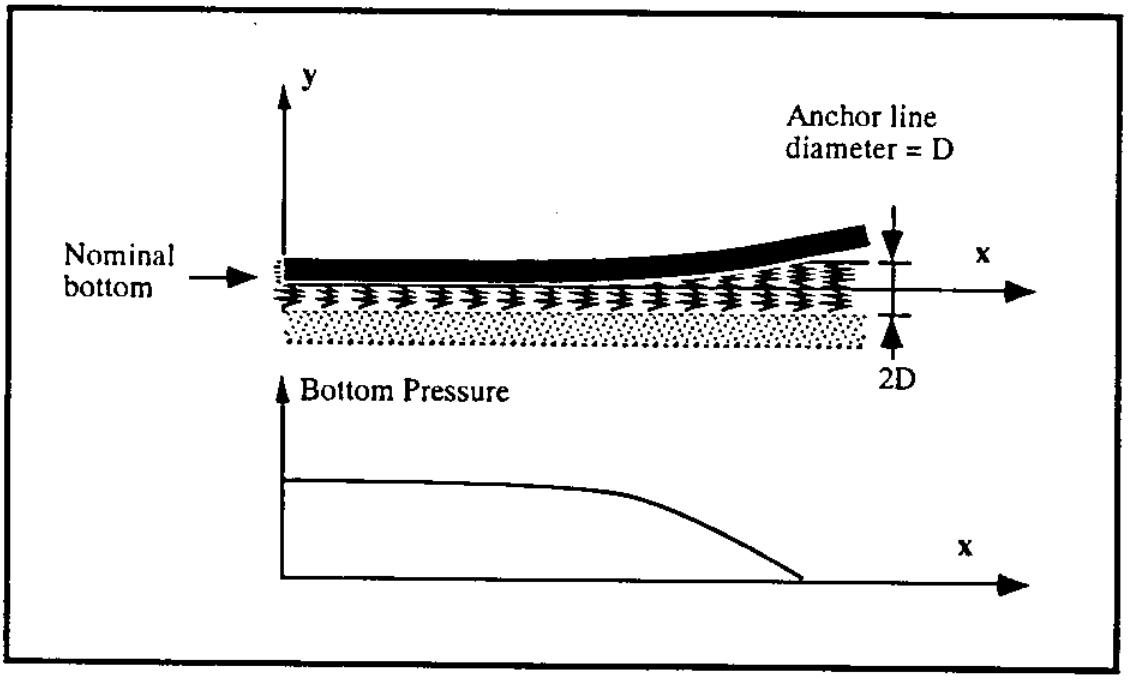


Figure 2.7 Schematic of the ocean bottom represented by an elastic layer

Due to the addition of the ocean bottom support force, an extra term, the distributed ocean bottom support force, is added to the equation of motion. Remember when we derived equation

2.16a, we multiplied both sides of the equation by the shape function a(s) and then integrated it with respect to s from 0 to L. We apply the same method to obtain the extra term:

$$\int_0^L q^{Spring} a_i ds = \mu_{im}\{\frac{S}{D}(D + D_{bottom})\}\Big|_m - \gamma_{ikm}\frac{S}{D}\Big|_m U_{k2} \tag{2.25}$$

Due to the presentation of the ocean bottom, the shape coefficients γ and μ are no longer constants for the element near the touch down point. Therefore, the two coefficients should be calculated for the element near the touch down point.

2.8 Energy Dissipation

In the previous sections, we made the assumption that the material studied is a pure elastic material. However, in the real world, materials do not behave in a perfectly elastic manner even at very low stresses. Energy is absorbed in an inelastic material system. The inelastic behavior of the material leads to the concept of energy dissipation.

Energy dissipation is defined to be proportional to the rate of the strain energy:

$$\text{Energy dissipation } D = -C\frac{d}{dt}\int_0^L T\varepsilon ds = -\frac{C}{h}{}_{22}K_{jm}^{(n+\frac{1}{2})}\lambda_j\delta\lambda_m \tag{2.26}$$

where C is the damping constant.

Multiplying both side of the stretch constraint equation by the tension λ, we may find that the equation represents the energy balance of the rod. By employing the concept of the energy dissipation, we modify equation 2.22 by a factor: FACT = $1-\frac{C}{h}$:

$$\{\frac{h^2}{4}{}_{21}K_{jk}^{(n+\frac{1}{2})}\}\{\frac{1}{h}\delta\dot{u}_k^{(n)}\} + \{-\text{FACT}\,\frac{h^2}{4}{}_{22}K_{jm}^{(n+\frac{1}{2})}\}\{\frac{2}{h^2}\delta\lambda_m^{(n)}\} = \frac{d_j^{(n)}}{2} + \frac{h}{2}{}_{21}K_{jk}^{(n+\frac{1}{2})}\dot{u}_k^{(n)} \tag{2.27}$$

FACT is suggested to be larger than 0.90.

Chapter 3

Results and Discussions

3.1 Validation

Several cases were studied, using the developed FORTRAN program CABLE, to validate and demonstrate the capabilities of the theory and the numerical approach employed in this project. We computed two problems for which analytical solutions could be obtained. One of the problems is a catenary moor. The other is a column exhibits post buckling behavior (elastica). The elastica has such a behavior that by increasing the external compressive load beyond the critical load, gradually, the column is not going to collapse but rather undergo large deformation. The theoretical results were obtained by Love (1927).

The configurations of the two cases are shown in figure 3.1.

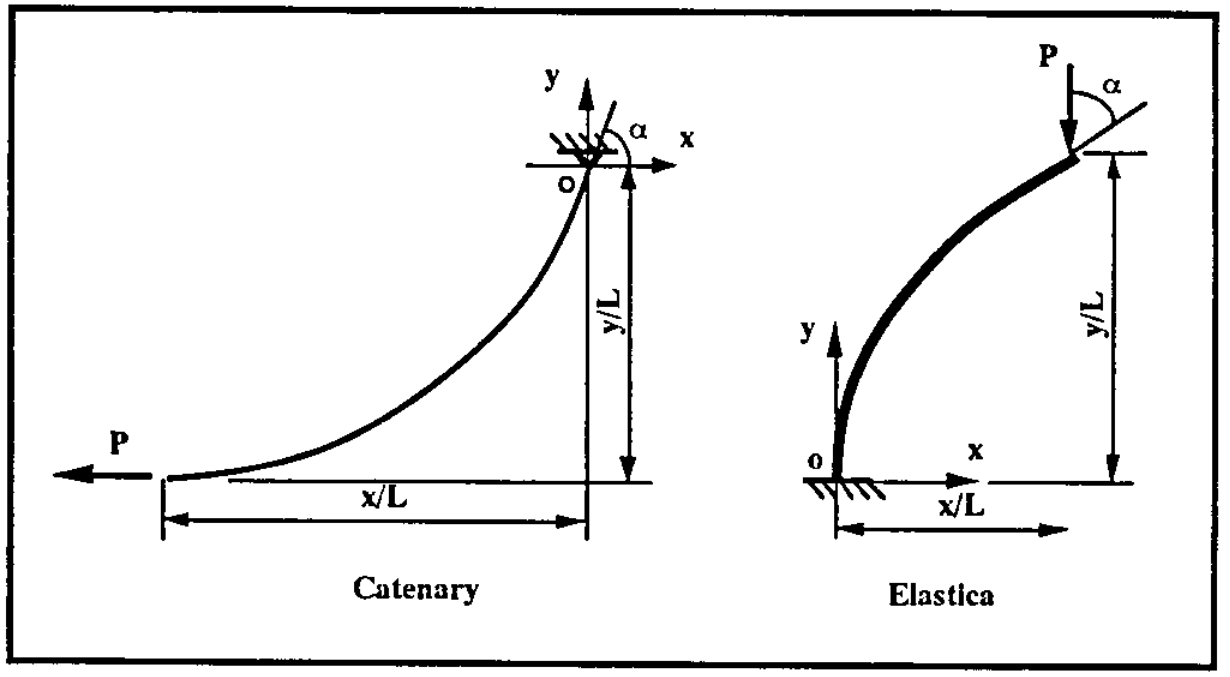


Figure 3.1 Schematic of the catenary moor and the elastica

Calculation results are shown in the following two tables. Table 3.1 is the comparison of the results of the catenary moor obtained from the program CABLE with the closed-form solution. Table 3.2 is the comparison of the results of the elastica. The error in the tables represents the

difference between the closed-form results and the numerical results. L is the length of the rod and P is the applied loading.

Table 3.1 Comparison of the numerical results with the closed-form results for a catenary (10 elements)

$w^4 l/p$	α (degree)	x/l	y/l
1	45.001	-0.88137	-0.41421
2	63.434	-0.72182	-0.61803
5	78.689	-0.46249	-0.81981
10	84.289	-0.29982	-0.90499
Error <	0.001	0.00001	0.00001

Table 3.2 Comparison of the numerical results with the closed-form results for an elastica (25 elements)

P/Pc[5]	α (degree)	x/l	y/l
2.542258	140.005	0.75037	-0.10696
9.116218	176.007	0.42151	-0.57720
Error <	0.007	0.00006	0.00004

In the above verification, a good match was found even for a coarse discretization.

We further demonstrated the capability of the theory by solving a more complex problem. Considering a marine cable, we performed a dynamic calculation. First, we performed the static calculation to obtain the equilibrium configuration of the cable (see figure 3.2). Then, the top

[4] w is the weight per unit length.

[5] Pc is the critical compressive load.

end of the cable was driven by a sinusoidal motion with the amplitude of the motion being 5.08 meters in x direction and 2.1 meters in z direction. Calculation was performed and the history of the motion of the cable was recorded by the program (See the work examples in Chapter 4). Figure 3.3 presents the time history of the top end tension of the cable.

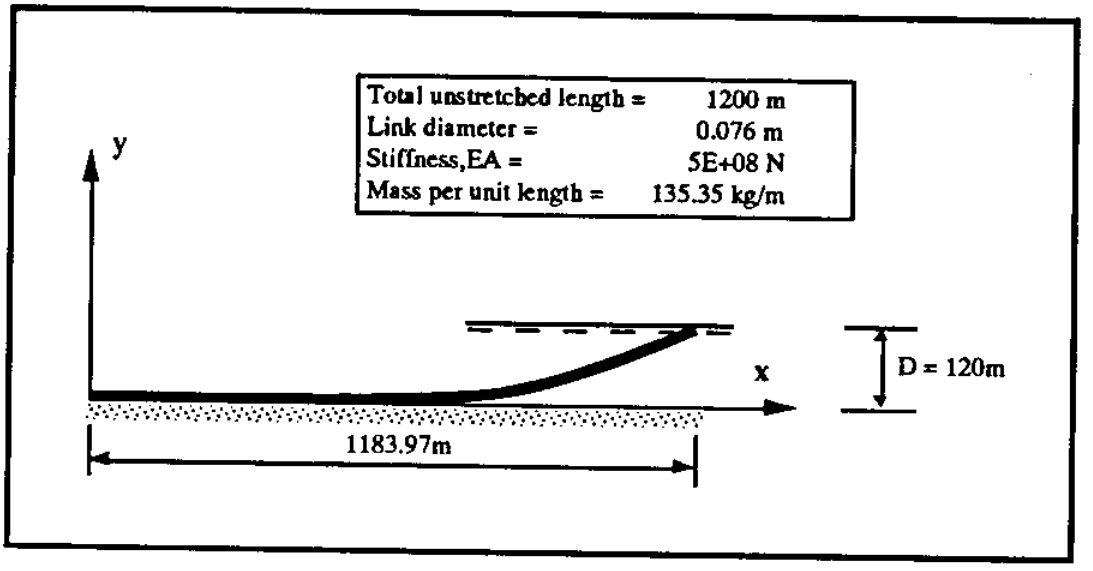


Figure 3.2 Schematic of a catenary marine cable

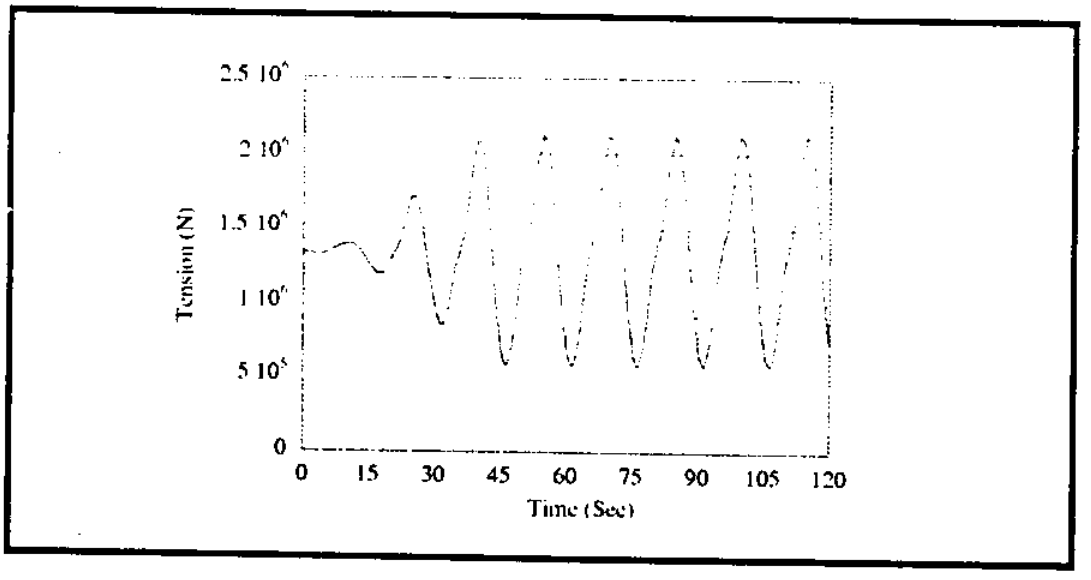


Figure 3.3 Time history of the tension at the top end of the cable

A comparison was made with the calculation results presented by Lindahl and Sjoberg (1983). A good agreement was obtained.

3.2 Analysis of a Catenary Anchor Leg Mooring System

The previously described computer program was used to perform the simulations of a catenary anchor leg mooring system (CALM). Two primary ocean environmental conditions were studied. One of the conditions was the condition of 2-year return period wind, wave and current. Under this condition, the dynamic analysis of the CALM with a San Diego class tanker was performed. The other environmental condition was the condition of 100-year return period wind, wave, and current. In this case, the dynamic tension of the CALM only was analyzed.

The most loaded line was analyzed based on the procedure recommended by API RP 2FP1. A quasi-static analysis was simulated under mean plus low frequency tension. The maximum tension was determined by the combination of the mean, the significant low frequency and the maximum wave frequency tension. The resulting tensions are given in Table 3.3. The reader is referred to Salancy (1994) for a detailed explanation of the environmental criteria used here.

Mooring parameters used in the analysis are given in Table 3.4. Figure 3.4-3.7 give the time history fairlead motion (surge only) and top end tension of the most loaded mooring line.

Table 3.3 Resulting Tension

Case I: CALM with the tanker		Case II: CALM only	
Mean Tension	98 kips	Mean Tension	73 kips
Mean + Low Frequency	100 kips	Mean + Low Frequency	80 kips
Maximum Tension	102 kips	Maximum Tension	96 kips

Table 3.4 Mooring Parameters

Top Segment:		
Elastic Stiffness	400,000	kips
Mass	4.6	slugs/ft
Buoyancy	19.1	lbs/ft
Added Mass Coefficient	3.8	
Drag Coefficient	1.0	
Diameter for Hydrodynamic force	0.625	ft
Line Length	100	ft
Mid-Segment:		
Elastic Stiffness	200,000	kips
Mass	0.652	slugs/ft
Buoyancy	2.7	lbs/ft
Added Mass Coefficient	3.8	
Drag Coefficient	1.0	
Diameter for Hydrodynamic force	0.292	ft
Line Length	1100	ft
Low Segment:		
Elastic Stiffness	400,000	kips
Mass	4.6	slugs/ft
Buoyancy	19.1	lbs/ft
Added Mass Coefficient	3.8	
Drag Coefficient	1.0	
Diameter for Hydrodynamic force	0.625	ft
Line Length	3300	ft

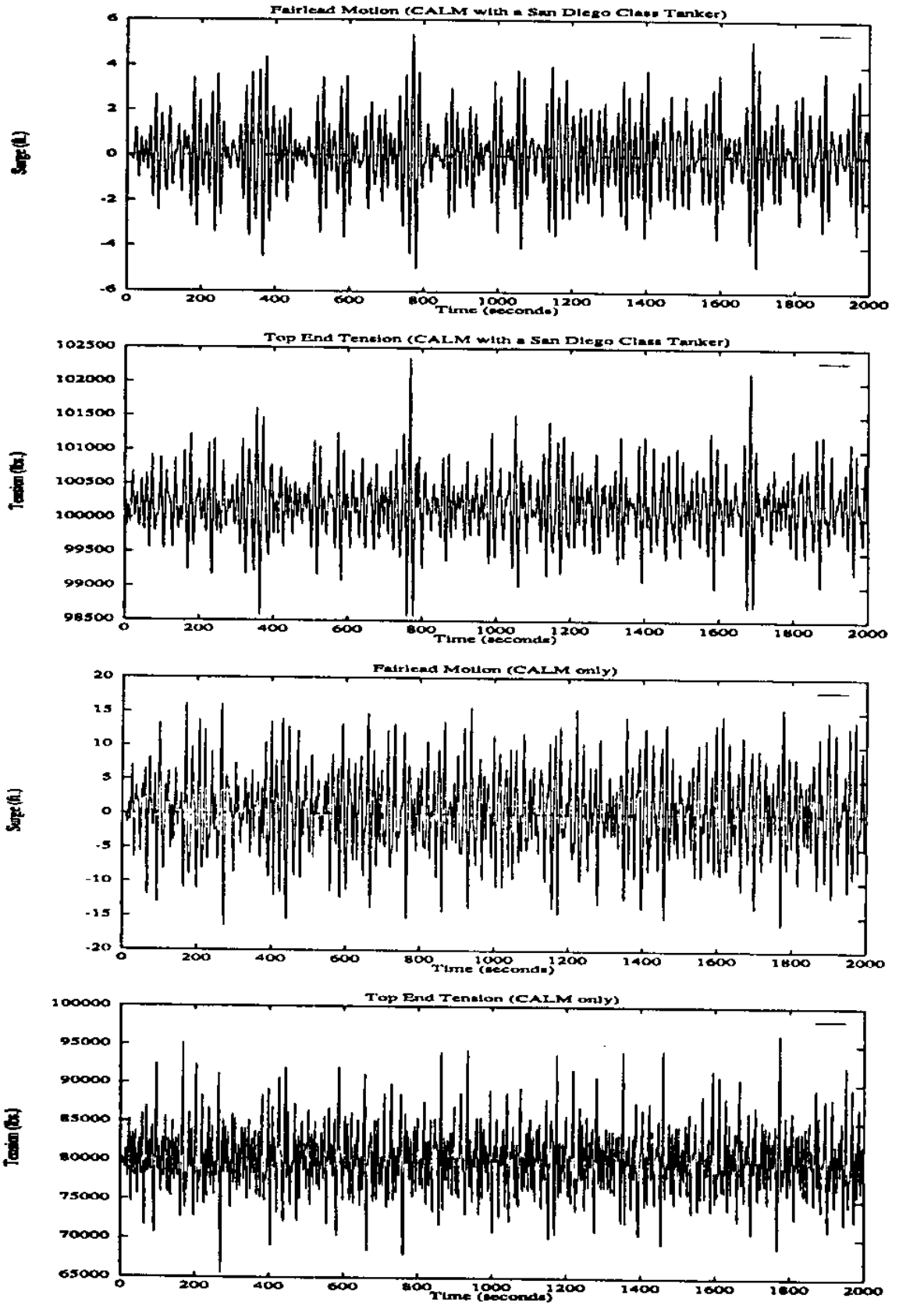


Figure 3.4 -3.7 Time history fairlead motion and top end tension

Chapter 4

User Manual

This part of the report serves as the user manual of the program CABLE (The source code is included as Appendix III to this report). It describes the input and the output format of the program.

CABLE is a cable analysis program for static and dynamic analysis of ocean cables. It is based on the theory of dynamics of slender rods introduced by Garrett (1982). The detailed theoretical background was documented in the previous chapters.

An ocean cable is exposed to loadings from many sources. The primary ones included in this program consist of the effect of gravity on the mass of the cable, the hydrostatic forces on the cable (both due to water outside of the cable and to any fluids which may be inside the cable), and the hydrodynamic forces due to the cable motion and wave motion.

Various boundary constraints are included in the programs. The program includes the contributions from the spring, damper, external static force, and concentrated mass at nodes. The ocean bottom is modeled by an elastic layer. A forced motion can also be enforced.

Several examples were studied to validate the program. Based on these validation studies and on the experience obtained during an intensive use of this program by the authors, it can be expected that the program is free of significant errors.

CABLE requires one ASCII input data file[6]. The program asks users to give the name of the input file. Calculation results are sent to an ASCII output file also named by the user. The program produces two supplementary data files as well to record some intermediate and time-domain results. File '**snapshot**' contains some intermediate results. File '**timeseries**' contains a time history of displacement, slopes, and tensions at the top end (always numbered node #1) of the cable.

A great deal of attention has been paid to the input/output format of the program. The program was developed so that the input of the program has the same format as that of the output.

---

[6] An additional input data file may be required to input the external forced motion.

Therefore, an output of the previous calculation could serve as an input for the next calculation. This is one of the great advantages of this program, which is very useful for iteration.

Any unit system can be used in the program. The only requirement is that the use of units be consistent.

This program was written in the standard FORTRAN 77 language, suitable for any IBM-compatible personal computer or workstations.

Several terms have been used frequently in the following text. These terms consist of a blank line, a text line, a new line, and a text phrase. A line, with the combination of a return, occupies one line in the input/output file. A blank line provides a spacing line between the previous line and the following line. A text line, with maximum of 80 characters, gives general information about the calculation to be carried out. Except for the text line in the first line of the input file, users are not required to give the information. Instead, a blank line may be used. The information will be printed out automatically by the program. A text phrase has the same function as a text line. But it only consists of a portion of a line. Like the text line, it may be omitted in the input file. Italicized fonts are used to mark the text phrases and text lines so that they can be different from other descriptions. A new line is required at some places in the file. It often marks some data input.

In the final section of this part, three work examples are presented.

Input/output data file

On the first line of the data file: Text Line

This is the only text line needed to be filled out by the user. General information about the calculation can be given in this line. For instance:

Single Point Mooring System : Static Analysis of A Mooring Line May 24, 1994

One Blank Line
Text Line: ***I. Global Constants***

Category I gives the constants involved in the calculation.

One Blank Line
Text Line: ***a. Indexes***
One Blank Line

New Line: NODE ***NODE (# of nodes)***

NODE is the number of nodes. As is mentioned above, the italicized phrase is a description and can be omitted in the input data file. **NODE** is an integer. The input format of **NODE** is I3, starting from column 2 of the line. The maximum number of nodes is preset to be 50, but it can be changed by modifying the dimensions of the corresponding array defined in the files named **'parametrics.inc'** and **'variables.inc'**.

New Line: NELTYP *NELTYP (# of different element types)*

NELTYP is the number of different element types. The maximum number of element types is 10 for a single rod. This preset dimension can be changed. **NELTYP** is an integer. The input format of **NELTYP** is I3, starting from column 2 of the line.

New Line: NDIM *NDIM (# of dimensions -2 or 3)*

NDIM is the dimensions of the space. This program is designed to be capable of handling both two dimensional and three dimensional spaces. **NDIM** is an integer. The input format of **NDIM** is I3, starting from column 2 of the line.

NDIM	= 2,	Two dimensional analysis. For two dimensional problem, only data related to x and y coordinate axes are needed.
NDIM	= 3,	Three dimensional analysis.

New Line: JOB *JOB (1 = static, 2 = dynamic)*

JOB is the job code. **JOB** is an integer. The input format of **JOB** is I3, starting from column 2 of the line.

JOB	= 1,	Static analysis.
JOB	= 2,	Dynamic analysis.

New Line: JSNAP *JSNAP (snapshot each JSNAP iterations)*

JSNAP is an index. Intermediate results at each **JSNAP** iteration in static analysis or each **JSNAP** time increments in dynamic analysis are recorded. The output of the intermediate results is sent to the file named 'snapshot'. **JSNAP** is an integer. The input format of **JSNAP** is I3, starting from column 2 of the line.

New Line: JMOTN *JMOTN (= 1, sin. motion, = 2, input forced motion)*

JMOTN is used to define the type of the external forced motion. **JMOTN** is an integer. The input format of **JMOTN** is I3, starting from column 2 of the line.

JMOTN = 1, Sinusoidal forced motion. Users are asked to input the amplitude and period of the motion.

JMOTN = 2, Arbitrary forced motion. The time domain data, stored in the file named "*forcedmotion*", should be given. See the explanation below for detail.

New Line: NCNSTR *NCNSTR(# of constrained nodes)*

NCNSTR is the number of constrained nodes. **NCNSTR** is an integer. The input format of **NCNSTR** is I3, starting from column 2 of the line. **NCNSTR** must be less than **NODE**.

One Blank Line
Text Line: *b. Parameters*
One Blank Line

New Line: RELAX, DAMP, ITER *relaxation , damping const. , # iteration*

RELAX is a parameter used to damp the speed of iteration. It is advised to set this parameter to be less than 0.5. **RELAX** is a real number. The input format of **RELAX** is F10.5, starting from column 1 of the line. **DAMP** is the coefficient of structural damping. The input format of **DAMP** is also F10.5, starting from column 11. **ITER** is the number of iterations used in the static analysis. Generally, the greater the number of iterations, the greater the accuracy. **ITER** is an integer. The input format of **ITER** is I5, starting from column 21 of the same line where **RELAX** and **DAMP** are.

New Line: TBGN, TEND, TSTEP *begin time, end time & time step*

TBGN, TEND, TSTEP are used to determine the starting time, ending time, and time increment in the dynamic calculation. It is obvious that **TEND > TBGN.** These three parameters are real numbers. The input formats of them are F9.5 with **TBGN** staring from column 2, **TEND** starting from column 12, and **TSTEP** starting from column 22, respectively.

New Line: CD, CM *drag and added-mass coefficients*

CD and **CM** are drag and added-mass coefficients of the rod. **CD** and **CM** are real numbers. The input formats of them are F10.5 with **CD** starting from column 1 and **CM** starting from column 11, respectively.

New Line: RUF, RUI, G *mass density of water, fluid,; gravity*

RUF is the mass density of the sea water. **RUI** is the mass density of the fluid inside the rod (if it is hollow). **G** is the acceleration of gravity. These three variables are real numbers. The input formats of them are F10.5 with **RUF** starting from column 1, **RUI** starting from column 11, and **G** starting from column 21, respectively.

New Line: BTMDPTH *bottom depth*
Two Blank Lines

BTMDPTH is the depth of the ocean bottom. Since the program requires the right-hand inertia coordinate system sit at the surface of still water with y axis pointing upward, **BTMDPTH** is always less than zero. **BTMDPTH** is a real number with input format of F10.2 starting from column 1 of the line.

Text Line: *II. Element and Nodal Information*

In Category II, material properties of the element type and nodal information, such as position and tension at the nodes, are defined.

One Blank Line
Text Line: *a. Physical Properties for Each Element Type*
One Blank Line

DO I = 1, NELTYP	1st End	Mid.	2nd End	*Phrase*
New Line: EEL (I, J)	(1)	(2)	(3)	*Young's mod, type #*
New Line: MOI (I, J)	(1)	(2)	(3)	*moment of inertia, (I)*
New Line: AFEL (I, J)	(1)	(2)	(3)	*cross-section area (Af)*
New Line: AIEL (I, J)	(1)	(2)	(3)	*internal cavity area (Ai)*
New Line: DFEL (I, J)	(1)	(2)	(3)	*dist. to outermost fiber*
New Line: MEL (I, J)	(1)	(2)	(3)	*mass of the rod per unit length*
New Line: DEL (I, J)	(1)	(2)	(3)	*diameters for hydrodynamic forces*
New Line: BUOYEL (I, J)	(1)	(2)	(3)	*buoyancy/unit length*
Blank Line				
----------				ENDDO

EEL = Young's module
MOI = moment of inertia of the rod
AFEL = cross-section area of the element
AIEL = internal cavity area of the element
DFEL = distance to the outermost fiber
MEL = mass of the rod per unit length

DEL = diameter for hydrodynamic forces

BUOYEL = buoyancy per unit length

EEL is a real number with the input format of 1PE11.5 starting from column 2 of the line.

MOI is the moment of inertia of the rod. Three values of **MOI** of an element should be given with **MOI (1)** is the moment of inertia at the first end of the element, **MOI (2)** is the moment of inertia at the mid-section, and **MOI (3)** is the moment of inertia at the second end. **MOI** is a real number. The input formats of **MOI** are 1PE11.5 with **MOI (1)** starting from column 2, **MOI (2)** starting from column 14, **MOI (3)** starting from column 26, respectively.

The remaining variables are all real. Each variable has three evaluations at the first end, mid-section, and second end of the element. Each variable consists of one line in the input data file and has the same input format as that of **MOI.**

One Blank Line
Text Line: *b. Coordinates and Lambdas for Each Node*
One Blank Line

DO I = 1, NODE	x	y	z	*Phrase*
New Line: R (I, J)	(1)	(2)	(3)	*x, y, and z position at node # I*
New Line: RPRM (I, J)	(1)	(2)	(3)	*x', y' and z' slopes*
New Line: RDOT (I, J)	(1)	(2)	(3)	*x-dot, y-dot, and z-dot*
New Line: RPMDT (I, J)	(1)	(2)	(3)	*x'-dot, y'-dot, and z'-dot*
New Line: LAMBDA (I, J)	(1)	(2)		*lambda at prev. ctr. and at node*
Blank Line				
----------				**ENDDO**

R = components of nodal position in x, y, and z direction

RPRM = components of slopes at node in x, y, and z direction

RDOT = components of time derivative of R in x, y, and z direction
RPMDT = components of time derivative of RPRM in x, y, and z direction
LAMBDA = tension at the previous mid-section and at the node

The inertia coordinate system is required to sit at the surface of the still water with its y axis pointing upward. It is also required that the top end of the rod be assigned the node #1. Variables **R, RPRM, RDOT,** and **RPMDT** are all expressed in this inertia coordinate system.

R, RPRM, RDOT, RPMDT, LAMBDA are real numbers. **R, RPRM, RDOT, RPMDT** have the same pattern of input format. The input format is F11.5 with the component in x direction starting from column 2, component in y direction starting from column 14, and component in z direction (3 D only) starting from column 26, respectively. **LAMBDA** has the input format of 1PE12.5 with the previous mid-section tension starting from column 1, followed by the nodal tension starting from column 13 of the line.

Text Line: *c. Connections between nodes*

Text Line: *length elmt type*

DO I = 1, NODE-1

New line: LNTHEL (I), JELTYP (I), *from node i to node j*

---------- ENDDO

Two Blank Lines

LNTHEL is the length of the element. **JELTYP** is the type of the element. Each element may have different length and different type (material and geometrical properties). The maximum element type, as mentioned above, is 10. **LNTHEL** is a real number with input format of F7.1. It is placed between column 2 and column 8. **JELTYP** is an integer. Its input format is I7 and it is placed between column 10 and column 16.

Text Line: ***III. Nodal Constraints***

One Blank Line

Boundary constraints at the nodes, such as spring, damper, initial static forces, and concentrated mass, will be set in Category III. Only transitional spring and damping constraints are considered in the program.

The transitional constraints are considered to be an arbitrarily oriented orthogonal spring and damper system. The directions of the system, along which the spring and damper act, are determined by the projection of the primary, 1st, and 2nd (3D only) axes of this system in the inertia coordinate system. The projection are read and stored in array **REFCOS (1), REFCOS (2),** and **REFCOS (3),** respectively. Since orthogonality is required, directions other than the primary axis of a 2-D space (or primary and 1st axes of a 3-D apace) will be modified automatically to achieve this.

Spring will be in effect when the specified node moves away from reference position and when the spring constant given in some directions is less than 1.0e10. When the spring constant is larger than 1.0e10 in a direction, there is absolutely no motion allowed in this direction. It is important to note that the locking of a node in one direction will result in locking the node in other directions unless the orthogonal spring system is parallel to the global coordinate system.

Damper and initial force are in action when the input values of them are not zero. Damping force is a linear function of velocity at the node. Initial force is considered as a static external force. Initial force is put on the right-hand side of the stiffness equation.

Forced motion of the reference point in one direction is in effect when the value of the spring constant in this direction is not larger than 1.0e10, and the amplitudes of motion (in-phase and/or out-phase) in the direction is not zero. If all the above conditions are satisfied, the magnitude of the spring constant is not important. This is because that the motion is a forced motion. The forced motion is a sinusoidal motion with the form:

$$DISAMP(1) \cdot SIN(FREQ \cdot t) + DISAMP(2) \cdot COS(FREQ \cdot t)$$

Other forms of forced motions are also allowed. Users are required to establish an ASCII input data file, named '*timeseries* ' to input the required information. Since the types of the forced

motions are different from the case to the case, it is suggested that the users develop their own scheme to read in the data. The corresponding subroutine, where the data being read, is the subroutine ***bondcns***. See detailed explanations given there.

DO I = 1, NODE

New Line: *Node No.* **NODEC (I)**

Blank Line

Text Line: ***a. linear constraints***

Text Line: ***direction cosines***

Text Line:		**primary**	*1st perp.*	*2nd perp.*	*ref. position*	
New Line:	**REFCOS (I, J, 1)**	**(I, 1, 1)**	**(I, 2, 1)**	**(I, 3, 1)**	**REFPOS (I, 1)**	*x direction*
New Line:	**REFCOS (I, J, 2)**	**(I, 1, 2)**	**(I, 2, 2)**	**(I, 3, 2)**	**REFPOS (I, 2)**	*y direction*
New Line:	**REFCOS (I, J, 3)**	**(I, 1, 3)**	**(I, 2, 3)**	**(I, 3, 3)**	**REFPOS (I, 3)**	*z direction*
New Line:	**SPGCONT (I, J)**	**(I, 1)**	**(I, 2)**	**(I, 3)**		*spring constant*
New Line:	**DMPCONT (I, J)**	**(I, 1)**	**(I, 2)**	**(I, 3)**		*damping constant*
New Line:	**FINITL (I, J)**	**(I, 1)**	**(I, 2)**	**(I, 3)**		*initial force*
New Line:	**CMASS (I)**	**(I)**				*concentrated mass*

Blank Line

Text Line: *b. Motion of Reference Point at Node I*

Blank Line

New Line: **FREQ (I)** *motion frequency (rad / sec)*

Blank Line

Text Line:	*in-phase*	*out-phase*	
	DISAMP (I, 1, 1)	**DISAMP (I, 1, 2)**	*x direction amplitude*
	DISAMP (I, 2, 1)	**DISAMP (I, 2, 2)**	*y direction amplitude*

Blank Line

---------- **ENDDO**

Blank Line

NODEC is an array containing the nodal number of the constrained nodes. Its input format is I3, starting from column 9 of the line.

REFCOS is the direction cosines of axes of the orthogonal system in the inertia coordinate system (ICS). For example, { **REFCOS (I, 1, 1), REFCOS (I, 1, 2), REFCOS (I, 1, 3)** } are the direction cosines of the primary axis in ICS. By induction, { **REFCOS (I, 2, 1), REFCOS (I, 2, 2), REFCOS (I, 2, 3)** } and { **REFCOS (I, 3, 1), REFCOS (I, 3, 2), REFCOS (I, 3, 3)** } are the direction cosines of the first perpendicular and second perpendicular axes in ICS, respectively. **REFPOS** is position of a reference point in ICS. If the position of the constrained node coincides with that of the reference point, there is no contribution force from the spring. Variable **REFCOS (I, 1, 1), REFCOS (I, 2, 1), REFCOS (I, 3, 1)** (3 D only), and **REFPOS (I, 1)** are placed in the same line. The input format of them is F8.5 for **REFCOS** and F10.3 for **REFPOS**. **REFCOS (I, 1, 1)** is placed between column 5 and column 12. **REFCOS (I, 2, 1)** is placed between column 17 and column 24 (for 2-D space, skip this step). You don't need to input **REFCOS (I, 3, 1)** since **REFCOS (I, 3, 1)** can be obtained by the cross product of **REFCOS (I, 1, 1)** and **REFCOS (I, 2, 1)**. **REFPOS (I, 1)** starts from column 41. Same rule applies for the set { **REFCOS (I, 1, 2), REFCOS (I, 2, 2), REFCOS (I, 3, 2), REFCOS (I, 2)** } and { **REFCOS (I, 1, 3), REFCOS (I, 2, 3), REFCOS (I, 3, 3), REFCOS (I, 3)** } (for a 2-D problem, use a blank line instead of it).

SPGCONT is a spring constant array. Its input format is 1PE12.5 with **SPGCONT (I, 1)** starting from column 1, **SPGCONT (I, 2)** starting from column 13, and **SPGCONT (I, 3)** starting from column 25 (for a 3 D problem, skip this step), respectively.

DMPCONT is a damping constant array. Its input format is F10.5 with **DMPCONT (I, 1)** starting from column 3, **DMPCONT (I, 2)** starting from column 15, and **DMPCONT (I, 3)** starting from column 27 (for a 3 D problem, skip this step), respectively.

FINITL is an applied external static force array. It follows the same input rule as **SPGCONT** does.

CMASS is a concentrated mass placed on the constrained node. It has the input format of F10.5 and is placed between column 3 and column 12.

FREQ is the frequency of the forced sinusoidal motion. It is placed between column 1 and column 15 with input format of F15.10.

DISAMP is the amplitude of the forced motion. **DISAMP (I, 1, 1)** and **DISAMP (I, 1, 2)** are the amplitudes of the in-phase and out-phase forced motion in x direction. **DISAMP (I, 2, 1)** and **DISAMP (I, 2, 2)** are the amplitudes in y direction. **DISAMP (I, 3, 1)** and **DISAMP (I, 3, 2)** are amplitudes in z direction. The input format of **DISAMP** is F9.3 with in-phase component starting from column 2 and out-phase component starting from column 12 of the line.

	Blank Line
Text Line:	***IV. Environmental Data***
	Blank Line

Environmental conditions are to be given in the Category IV. Current and linear waves are considered to be the major contributions from the environment.

Text Line:	***a. Currents at*** **NVC** ***depths***			
Blank Line				
Text Line:	***Depth***	***x-vel***	***y-vel***	***z-vel***
DO I = 1, NVC				
New Line:	**VC(I, 1)**	**VC(I, 2)**	**VC(I, 3)**	**VC(I, 4)**
				ENDDO
Blank Line				

Current is evaluated at several depths. The number of evaluations is defined by the variable NVC. VC is an array containing depths and components of velocity of current in x, y, and z directions. The input format of NVC is I3, starting from column 15. The input format of VC is F10.4, with the first components of VC starting from column 1 of the line. On the same line, the second, third, and the fourth components of VC are placed. The second components is placed between column 11 and 20. The third component is placed between column 21 and 30. The fourth component is placed between column 31 and 40. Skip the last step if you are dealing with a two dimensional problem.

Text Line:	*Waves*	
Blank Line		
New Line:	**AW, SIGMAW**	***wave amplitude and frequency (rad/sec)***
New Line:	**DIRW**	***wave direction relative to x axis***

AW is the amplitude of linear incoming wave. **SIGMAW** is the frequency of the wave. **DIRW** is the direction of wave relative to x axis. The input format of **AW** is F10.6, starting from column 1 of the line. On the same line, **SIGMAW** is placed. Its input format is F10.5, starting from column 11 of the line. **DIRW** is placed on a new line with input format of F10.7. It starts from column 1 of the line.

End of Input File

Three examples are given in Appendix II. As it was mentioned at the beginning of this manual, there is no difference between the format of the input file and that of the output file. Therefore, an output file can serve as an input file for another calculation. The following examples demonstrate advantages of this format. The examples were designed so that the next example uses the output of previous example as an input.

The first print-out, ***static-problem-without-bottom***, was an output of calculation of a mooring line hanging in the sea with its top end fixed at the sea level and its lower end fixed at the ocean bottom. Ocean bottom was not present in this case. Therefore some portion of the mooring line was below the ocean bottom. Absence of the ocean bottom didn't meet the physical reality. Hence, at the next step, the ocean bottom was modeled by a layer of springs. The output of the first example was used as an input file for the second case. Only the depth of the ocean bottom (**BTMDPTH**) was modified. The static equilibrium was then obtained, which was shown in the second print-out, ***static-problem-with-bottom*** . In the last example, dynamic analysis of this mooring line with its top end driven by a sinusoidal motion was performed. The third print-out, ***dynamic-problem,*** is the results of the calculation. The configuration of the cable shown in the last print-out is the configuration at time t = 120s. The time history of the tension of the cable at its top end was depicted in figure 3.3 shown in the previous chapter.

The charts shown in the previous chapter were drawn on the basis of the data obtained by these three examples.

Chapter 5

Conclusions

A finite-element based model has been developed to evaluate and determine the dynamics of cable. One of the important features of the model is that it has the ability to predict a variety of nonlinear marine mooring line motions. In this model, various loadings encountered in submerged marine applications are included. The loadings include the forces on the rod due to hydrodynamic effects resulting from the rod's motion and the exterior fluid motion (estimated here using the Morison equation), the effect of pressure gradients in both the external and internal fluid (the rod is perhaps hollow and filled with fluid), and the weight of the rod. Besides this, the model also incorporates bending rigidity which extends the scope of its application.

The model has been computerized and a code has been written in standard FORTRAN 77. The Code CABLE carries out a time domain calculation of the motion and tension of cables. The program can be operated on personal computers as well as work stations. A User Manual of the program is included as Chapter IV to this report. Several cases were studied, using the developed FORTRAN program CABLE, to validate and demonstrate the capabilities of the theory and the numerical approach employed in this report.

The computer code has been applied to a catenary anchor leg mooring system (CALM). Two primary ocean environmental conditions have been chosen. The example applications served to show the potential use of the code CABLE as well as to provide the engineering approach with necessary information.

Reference

[1] Garrett, D.L, Dynamic analysis of slender rods, Proc. 1st International OMAE Conf. Dallas, 1982.

[2] Lindahl, L. and Sjoberg, A., Dynamic analysis of mooring cables, The Second International Symposium on Ocean Engineering and Ship Handling, 1983.

[3] Love, A.E.H., A treatise on the mathematical and theory of elasticity, 4th Ed., Dover Publications, New York, 1927.

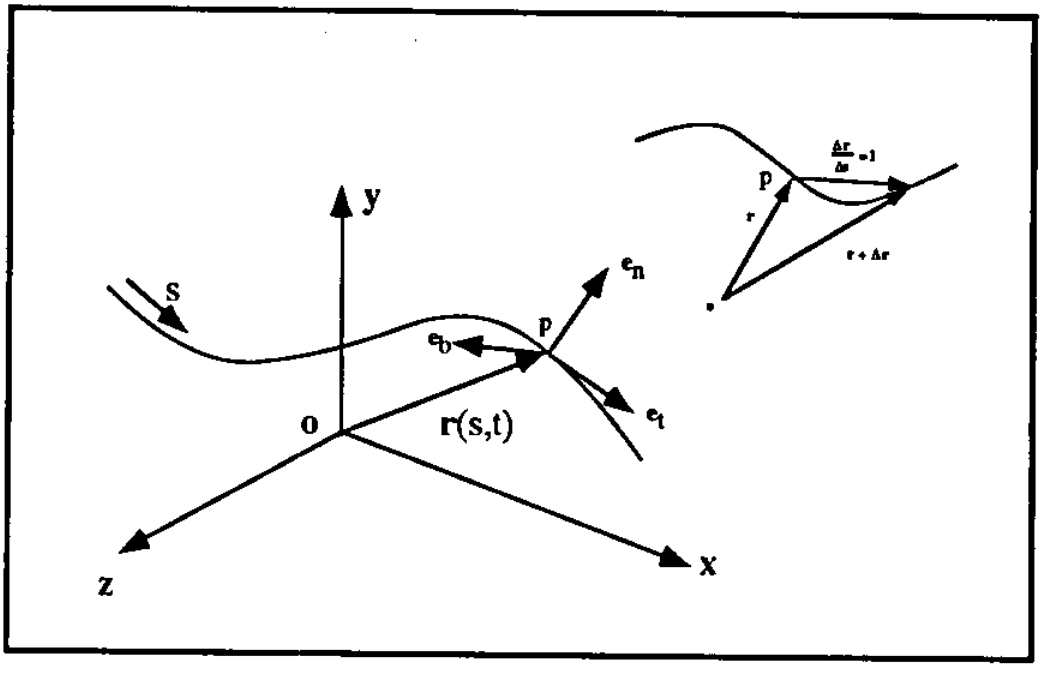
[4] Paulling, J.R. and Webster, W.C., A consistent, large-amplitude analysis of the coupled response of a TLP and tendon system, Proc. of the Fifth International Offshore Mechanics and Arctic Engineering Symposium, 1986

[5] Salancy, Aaron, Offshore single point mooring system for import of hazardous liquid cargoes, Master's thesis, Department of Naval Architecture and Offshore Engineering, University of California at Berkeley, May, 1994.

[6] US Department of Transportation, Coast Guard. Deepwater ports study, Office of Marine safety, Security and Environmental Protection, Washington, DC, 1993.

Appendix I. The Geometrical Relationships

The following geometrical relationships are useful in the derivation of equation 2.6. Let r(s,t) be a vector distance from the origin of the inertia coordinate system to the point P, located by the arc length s. Let $\{P; e_t, e_n, e_b\}$ forms a local coordinate system with its origin placed at P. The directions of this local coordinate system is defined by the base $\{e_t, e_n, e_b\}$, where e_n is the normal of the rod at the point P; e_t is the tangent ; and e_b is the cross product of e_t and e_n.



The geometrical relationships were derived with the help of the figure shown above:

$$r' = e_t \Rightarrow r''(s,t) = e_t' = \kappa e_n(s,t)$$

$$e_b = e_t \times e_n = r' \times \frac{1}{\kappa} r''$$

$$r' \cdot r' = |r'|^2 = 1$$

$$(r' \cdot r')' = 2r' \cdot r'' = 2e_t \cdot \kappa e_n(s,t) = 0 \Rightarrow r' \cdot r'' = e_t \cdot \kappa e_n(s,t) = 0$$

$$(r' \cdot r'')' = r' \cdot r''' + r'' \cdot r'' = r' \cdot r''' + \kappa^2 = 0 \Rightarrow r' \cdot r''' = -\kappa^2$$

Appendix II. Working Examples

Static analysis of a cable with the bottom presenting. 05/24/94

I. Global Constants

a. Indexs

41	NODE	(# of nodes)
1	NELTYP	(# of different element types)
3	NDIM	(# of dimensions - 2 or 3)
1	JOB	(1 = static, 2 = dynamic)
50	JSNAP	(snapshot each JSNAP iterations)
1	JMOTN	(1 = sin. motion, 2 = input forced motion)
2	NCNSTR	(# of constrained nodes)

b. Parameters

.20000	.00000	50	relaxation, # iterations
.00000	.00000	.00000	begin time, end time & time step
1.00000	3.80000		drag and added-mass coefficients
1000.00000	.00000	9.80665	mass density of water, fluid; gravity
-999.00			bottom depth

II. Element and Nodal Information

a. Physical Properties for Each Element Type

5.51090E+10			Young's mod: type 1
0.00000E+00	0.00000E+00	0.00000E+00	moment of inertia (I)
9.07292E-03	9.07292E-03	9.07292E-03	cross-section area (Af)
0.00000E+00	0.00000E+00	0.00000E+00	internal cavity area (Ai)
2.28000E-01	2.28000E-01	2.28000E-01	dist. to outermost fiber
1.35350E+02	1.35350E+02	1.35350E+02	mass of the rod per unit length
7.60000E-02	7.60000E-02	7.60000E-02	diameters for hydrodynamic forces
1.70171E+02	1.70171E+02	1.70171E+02	total buoyancy/unit length

b. Coordinates and Lambdas for Each Node

.00002	.00000	.00000	x, y and z position at node # 1
.93573	-.36704	.00000	x', y' and z' slopes
.00000	.00000	.00000	x-dot, y-dot and z-dot
.00000	.00000	.00000	lambda at prev. ctr. and at node
0.00000E+00	2.57439E+06		lambda at prev. ctr. and at node
28.13966	-10.83479	.00000	x, y and z position at node # 2
.94023	-.35525	.00000	x', y' and z' slopes
.00000	.00000	.00000	x-dot, y-dot and z-dot
.00000	.00000	.00000	lambda at prev. ctr. and at node
2.56817E+06	2.56205E+06		lambda at prev. ctr. and at node
56.41295	-21.31361	.00000	x, y and z position at node # 3
.94463	-.34330	.00000	x', y' and z' slopes
.00000	.00000	.00000	x-dot, y-dot and z-dot
.00000	.00000	.00000	lambda at prev. ctr. and at node
2.55603E+06	2.55011E+06		lambda at prev. ctr. and at node
84.81661	-31.43142	.00000	x, y and z position at node # 4
.94892	-.33119	.00000	x', y' and z' slopes
.00000	.00000	.00000	x-dot, y-dot and z-dot
.00000	.00000	.00000	lambda at prev. ctr. and at node
2.54429E+06	2.53858E+06		lambda at prev. ctr. and at node
113.34722	-41.18331	.00000	x, y and z position at node # 5
.95310	-.31891	.00000	x', y' and z' slopes
.00000	.00000	.00000	x-dot, y-dot and z-dot
.00000	.00000	.00000	lambda at prev. ctr. and at node
2.53297E+06	2.52747E+06		lambda at prev. ctr. and at node
142.00113	-50.56442	.00000	x, y and z position at node # 6
.95714	-.30647	.00000	x', y' and z' slopes
.00000	.00000	.00000	x-dot, y-dot and z-dot
.00000	.00000	.00000	lambda at prev. ctr. and at node
2.52207E+06	2.51678E+06		lambda at prev. ctr. and at node
170.77456	-59.57002	.00000	x, y and z position at node # 7
.96106	-.29388	.00000	x', y' and z' slopes
.00000	.00000	.00000	x-dot, y-dot and z-dot
.00000	.00000	.00000	lambda at prev. ctr. and at node
2.51160E+06	2.50652E+06		lambda at prev. ctr. and at node
199.66354	-68.19547	.00000	x, y and z position at node # 8
.96485	-.28113	.00000	x', y' and z' slopes
.00000	.00000	.00000	x-dot, y-dot and z-dot
.00000	.00000	.00000	lambda at prev. ctr. and at node
2.50155E+06	2.49669E+06		lambda at prev. ctr. and at node
228.66395	-76.43627	.00000	x, y and z position at node # 9
.96849	-.26823	.00000	x', y' and z' slopes
.00000	.00000	.00000	x-dot, y-dot and z-dot
.00000	.00000	.00000	lambda at prev. ctr. and at node
2.49194E+06	2.48730E+06		lambda at prev. ctr. and at node
257.77148	-84.28805	.00000	x, y and z position at node # 10
.97199	-.25519	.00000	x', y' and z' slopes
.00000	.00000	.00000	x-dot, y-dot and z-dot
.00000	.00000	.00000	lambda at prev. ctr. and at node
2.48277E+06	2.47835E+06		lambda at prev. ctr. and at node
286.98166	-91.74658	.00000	x, y and z position at node # 11
.97533	-.24202	.00000	x', y' and z' slopes
.00000	.00000	.00000	x-dot, y-dot and z-dot
.00000	.00000	.00000	lambda at prev. ctr. and at node
2.47404E+06	2.46984E+06		lambda at prev. ctr. and at node
316.28988	-98.80780	.00000	x, y and z position at node # 12
.97852	-.22871	.00000	x', y' and z' slopes
.00000	.00000	.00000	x-dot, y-dot and z-dot
.00000	.00000	.00000	lambda at prev. ctr. and at node
2.46576E+06	2.46180E+06		lambda at prev. ctr. and at node
345.69134	-105.46778	.00000	x, y and z position at node # 13
.98155	-.21527	.00000	x', y' and z' slopes
.00000	.00000	.00000	x-dot, y-dot and z-dot
.00000	.00000	.00000	lambda at prev. ctr. and at node
2.45794E+06	2.45420E+06		lambda at prev. ctr. and at node
375.18113	-111.72281	.00000	x, y and z position at node # 14
.98441	-.20171	.00000	x', y' and z' slopes
.00000	.00000	.00000	x-dot, y-dot and z-dot
.00000	.00000	.00000	lambda at prev. ctr. and at node
2.45058E+06	2.44707E+06		lambda at prev. ctr. and at node
404.75417	-117.56932	.00000	x, y and z position at node # 15
.98710	-.18804	.00000	x', y' and z' slopes
.00000	.00000	.00000	x-dot, y-dot and z-dot
.00000	.00000	.00000	lambda at prev. ctr. and at node
2.44368E+06	2.44041E+06		lambda at prev. ctr. and at node

434.40523	-123.00397	.00000	x, y and z position at node # 16
.98961	-.17425	.00000	x', y' and z' slopes
.00000	.00000	.00000	x-dot, y-dot and z-dot
.00000	.00000	.00000	lambda at prev. ctr. and at node
2.43725E+06	2.43421E+06		lambda at prev. ctr. and at node
464.12897	-128.02360	.00000	x, y and z position at node # 17
.99194	-.16037	.00000	x', y' and z' slopes
.00000	.00000	.00000	x-dot, y-dot and z-dot
.00000	.00000	.00000	lambda at prev. ctr. and at node
2.43129E+06	2.42849E+06		lambda at prev. ctr. and at node
493.91993	-132.62527	.00000	x, y and z position at node # 18
.99409	-.14639	.00000	x', y' and z' slopes
.00000	.00000	.00000	x-dot, y-dot and z-dot
.00000	.00000	.00000	lambda at prev. ctr. and at node
2.42581E+06	2.42324E+06		lambda at prev. ctr. and at node
523.77250	-136.80627	.00000	x, y and z position at node # 19
.99605	-.13233	.00000	x', y' and z' slopes
.00000	.00000	.00000	x-dot, y-dot and z-dot
.00000	.00000	.00000	lambda at prev. ctr. and at node
2.42080E+06	2.41847E+06		lambda at prev. ctr. and at node
553.68099	-140.56410	.00000	x, y and z position at node # 20
.99782	-.11818	.00000	x', y' and z' slopes
.00000	.00000	.00000	x-dot, y-dot and z-dot
.00000	.00000	.00000	lambda at prev. ctr. and at node
2.41627E+06	2.41419E+06		lambda at prev. ctr. and at node
583.63960	-143.89652	.00000	x, y and z position at node # 21
.99939	-.10397	.00000	x', y' and z' slopes
.00000	.00000	.00000	x-dot, y-dot and z-dot
.00000	.00000	.00000	lambda at prev. ctr. and at node
2.41223E+06	2.41039E+06		lambda at prev. ctr. and at node
613.64244	-146.80149	.00000	x, y and z position at node # 22
1.00077	-.08969	.00000	x', y' and z' slopes
.00000	.00000	.00000	x-dot, y-dot and z-dot
.00000	.00000	.00000	lambda at prev. ctr. and at node
2.40867E+06	2.40708E+06		lambda at prev. ctr. and at node
643.68354	-149.27726	.00000	x, y and z position at node # 23
1.00194	-.07535	.00000	x', y' and z' slopes
.00000	.00000	.00000	x-dot, y-dot and z-dot
.00000	.00000	.00000	lambda at prev. ctr. and at node
2.40560E+06	2.40425E+06		lambda at prev. ctr. and at node
673.75687	-151.32232	.00000	x, y and z position at node # 24
1.00291	-.06098	.00000	x', y' and z' slopes
.00000	.00000	.00000	x-dot, y-dot and z-dot
.00000	.00000	.00000	lambda at prev. ctr. and at node
2.40303E+06	2.40192E+06		lambda at prev. ctr. and at node
703.85633	-152.93541	.00000	x, y and z position at node # 25
1.00368	-.04656	.00000	x', y' and z' slopes
.00000	.00000	.00000	x-dot, y-dot and z-dot
.00000	.00000	.00000	lambda at prev. ctr. and at node
2.40094E+06	2.40008E+06		lambda at prev. ctr. and at node
733.97576	-154.11553	.00000	x, y and z position at node # 26
1.00425	-.03211	.00000	x', y' and z' slopes
.00000	.00000	.00000	x-dot, y-dot and z-dot
.00000	.00000	.00000	lambda at prev. ctr. and at node
2.39935E+06	2.39874E+06		lambda at prev. ctr. and at node
764.10899	-154.86195	.00000	x, y and z position at node # 27
1.00460	-.01765	.00000	x', y' and z' slopes
.00000	.00000	.00000	x-dot, y-dot and z-dot
.00000	.00000	.00000	lambda at prev. ctr. and at node
2.39825E+06	2.39789E+06		lambda at prev. ctr. and at node
794.24981	-155.17422	.00000	x, y and z position at node # 28
1.00475	-.00317	.00000	x', y' and z' slopes
.00000	.00000	.00000	x-dot, y-dot and z-dot
.00000	.00000	.00000	lambda at prev. ctr. and at node
2.39765E+06	2.39753E+06		lambda at prev. ctr. and at node
824.39199	-155.05214	.00000	x, y and z position at node # 29
1.00469	.01131	.00000	x', y' and z' slopes
.00000	.00000	.00000	x-dot, y-dot and z-dot
.00000	.00000	.00000	lambda at prev. ctr. and at node
2.39754E+06	2.39767E+06		lambda at prev. ctr. and at node
854.52930	-154.49579	.00000	x, y and z position at node # 30
1.00443	.02578	.00000	x', y' and z' slopes
.00000	.00000	.00000	x-dot, y-dot and z-dot
.00000	.00000	.00000	lambda at prev. ctr. and at node
2.39792E+06	2.39830E+06		lambda at prev. ctr. and at node
884.65554	-153.50550	.00000	x, y and z position at node # 31
1.00395	.04024	.00000	x', y' and z' slopes
.00000	.00000	.00000	x-dot, y-dot and z-dot
.00000	.00000	.00000	lambda at prev. ctr. and at node
2.39881E+06	2.39943E+06		lambda at prev. ctr. and at node
914.76450	-152.08190	.00000	x, y and z position at node # 32
1.00328	.05467	.00000	x', y' and z' slopes
.00000	.00000	.00000	x-dot, y-dot and z-dot
.00000	.00000	.00000	lambda at prev. ctr. and at node
2.40018E+06	2.40106E+06		lambda at prev. ctr. and at node
944.85002	-150.22586	.00000	x, y and z position at node # 33
1.00239	.06906	.00000	x', y' and z' slopes
.00000	.00000	.00000	x-dot, y-dot and z-dot
.00000	.00000	.00000	lambda at prev. ctr. and at node
2.40205E+06	2.40317E+06		lambda at prev. ctr. and at node
974.90599	-147.93853	.00000	x, y and z position at node # 34
1.00131	.08342	.00000	x', y' and z' slopes
.00000	.00000	.00000	x-dot, y-dot and z-dot
.00000	.00000	.00000	lambda at prev. ctr. and at node
2.40442E+06	2.40578E+06		lambda at prev. ctr. and at node
1004.92632	-145.22129	.00000	x, y and z position at node # 35
1.00002	.09772	.00000	x', y' and z' slopes
.00000	.00000	.00000	x-dot, y-dot and z-dot
.00000	.00000	.00000	lambda at prev. ctr. and at node
2.40727E+06	2.40888E+06		lambda at prev. ctr. and at node
1034.90503	-142.07582	.00000	x, y and z position at node # 36
.99853	.11197	.00000	x', y' and z' slopes
.00000	.00000	.00000	x-dot, y-dot and z-dot
.00000	.00000	.00000	lambda at prev. ctr. and at node
2.41061E+06	2.41247E+06		lambda at prev. ctr. and at node
1064.83619	-138.50401	.00000	x, y and z position at node # 37
.99685	.12614	.00000	x', y' and z' slopes

```
      .00000       .00000       .00000      x-dot, y-dot and z-dot
      .00000       .00000       .00000      lambda at prev. ctr. and at node
 2.41444E+06  2.41654E+06                   lambda at prev. ctr. and at node

  1094.71395   -134.50802       .00000      x, y and z position at node # 38
      .99497       .14024       .00000      x', y' and z' slopes
      .00000       .00000       .00000      x-dot, y-dot and z-dot
      .00000       .00000       .00000      lambda at prev. ctr. and at node
 2.41876E+06  2.42110E+06                   lambda at prev. ctr. and at node

  1124.53258   -130.09024       .00000      x, y and z position at node # 39
      .99291       .15426       .00000      x', y' and z' slopes
      .00000       .00000       .00000      x-dot, y-dot and z-dot
      .00000       .00000       .00000      lambda at prev. ctr. and at node
 2.42355E+06  2.42613E+06                   lambda at prev. ctr. and at node

  1154.28644   -125.25328       .00000      x, y and z position at node # 40
      .99065       .16819       .00000      x', y' and z' slopes
      .00000       .00000       .00000      x-dot, y-dot and z-dot
      .00000       .00000       .00000      lambda at prev. ctr. and at node
 2.42883E+06  2.43165E+06                   lambda at prev. ctr. and at node

  1183.97000   -120.00000       .00000      x, y and z position at node # 41
      .98822       .18201       .00000      x', y' and z' slopes
      .00000       .00000       .00000      x-dot, y-dot and z-dot
      .00000       .00000       .00000      lambda at prev. ctr. and at node
 2.43456E+06  2.43764E+06                   lambda at prev. ctr. and at node
```

c. Connections between Nodes

length	elmt type	
30.0	1	from node 1 to 2
30.0	1	from node 2 to 3
30.0	1	from node 3 to 4
30.0	1	from node 4 to 5
30.0	1	from node 5 to 6
30.0	1	from node 6 to 7
30.0	1	from node 7 to 8
30.0	1	from node 8 to 9
30.0	1	from node 9 to 10
30.0	1	from node 10 to 11
30.0	1	from node 11 to 12
30.0	1	from node 12 to 13
30.0	1	from node 13 to 14
30.0	1	from node 14 to 15
30.0	1	from node 15 to 16
30.0	1	from node 16 to 17
30.0	1	from node 17 to 18
30.0	1	from node 18 to 19
30.0	1	from node 19 to 20
30.0	1	from node 20 to 21
30.0	1	from node 21 to 22
30.0	1	from node 22 to 23
30.0	1	from node 23 to 24
30.0	1	from node 24 to 25
30.0	1	from node 25 to 26
30.0	1	from node 26 to 27
30.0	1	from node 27 to 28
30.0	1	from node 28 to 29
30.0	1	from node 29 to 30
30.0	1	from node 30 to 31
30.0	1	from node 31 to 32
30.0	1	from node 32 to 33
30.0	1	from node 33 to 34
30.0	1	from node 34 to 35
30.0	1	from node 35 to 36
30.0	1	from node 36 to 37
30.0	1	from node 37 to 38
30.0	1	from node 38 to 39
30.0	1	from node 39 to 40
30.0	1	from node 40 to 41

III. Nodal Constraints

Node No. 1

a. Linear Constraints

direction consines reference	1st perp.	2nd perp.	reference position	
1.00000	.00000	.00000	.000	x direction
.00000	1.00000	.00000	.000	y direction
.00000	.00000	1.00000	.000	z direction
1.00000E+11	1.00000E+11	1.00000E+11		spring constant
.00000	.00000	.00000		damping constant
0.00000E+00	0.00000E+00	0.00000E+00		initial force
.00000				concentrated mass

b. Motion of Reference Point at Node 1

.0000000000 motion frequencey (rad/sec)

in-phase	out-phase	
.000	.000	x displacement amplitude
.000	.000	y displacement amplitude
.000	.000	z displacement amplitude

Node No. 41

a. Linear Constraints

direction consines reference	1st perp.	2nd perp.	reference position	
1.00000	.00000	.00000	1183.970	x direction
.00000	1.00000	.00000	120.000	y direction
.00000	.00000	1.00000	.000	z direction
1.00000E+11	1.00000E+11	1.00000E+11		spring constant
.00000	.00000	.00000		damping constant
0.00000E+00	0.00000E+00	0.00000E+00		initial force
.00000				concentrated mass

b. Motion of Reference Point at Node 41

.0000000000 motion frequencey (rad/sec)

in-phase	out-phase	
.000	.000	x displacement amplitude
.000	.000	y displacement amplitude
.000	.000	z displacement amplitude

IV. Environmental Data

a. Currents at 4 depths

depth	x-vel	y-vel	z-vel
.0000	.0000	.0000	.0000
-40.0000	.0000	.0000	.0000
-80.0000	.0000	.0000	.0000
-120.0000	.0000	.0000	.0000

b. Waves

```
   .000000     .00000        wave amplitude and frequency (rad/sec)
  .0000000                   wave direction relative to x axis
```

Static analysis of a cable with the bottom presenting. 05/24/94

I. Global Constants

a. Indexs

41	NODE	(# of nodes)
1	NELTYP	(# of different element types)
3	NDIM	(# of dimensions - 2 or 3)
1	JOB	(1 = static, 2 = dynamic)
50	JSNAP	(snapshot each JSNAP iterations)
1	JMOTN	(1 = sin. motion, 2 = input forced motion)
2	NCNSTR	(# of constrained nodes)

b. Parameters

.20000	.00000	50	relaxation, # iterations
.00000	.00000	.00000	begin time, end time & time step
1.00000	3.80000		drag and added-mass coefficients
1000.00000	.00000	9.80665	mass density of water, fluid; gravity
-120.00			bottom depth

II. Element and Nodal Information

a. Physical Properties for Each Element Type

5.51090E+10			Young's mod: type 1
0.00000E+00	0.00000E+00	0.00000E+00	moment of inertia (I)
9.07292E-03	9.07292E-03	9.07292E-03	cross-section area (Af)
0.00000E+00	0.00000E+00	0.00000E+00	internal cavity area (Ai)
2.28000E-01	2.28000E-01	2.28000E-01	dist. to outermost fiber
1.35350E+02	1.35350E+02	1.35350E+02	mass of the rod per unit length
7.60000E-02	7.60000E-02	7.60000E-02	diameters for hydrodynamic forces
1.70171E+02	1.70171E+02	1.70171E+02	total buoyancy/unit length

b. Coordinates and Lambdas for Each Node

.00002	.00000	.00000	x, y and z position at node # 1
.89855	-.44488	.00000	x', y' and z' slopes
.00000	.00000	.00000	x-dot, y-dot and z-dot
.00000	.00000	.00000	lambda at prev. ctr. and at node
0.00000E+00	1.32931E+06		lambda at prev. ctr. and at node
27.11070	-13.02786	.00000	x, y and z position at node # 2
.90878	-.42353	.00000	x', y' and z' slopes
.00000	.00000	.00000	x-dot, y-dot and z-dot
.00000	.00000	.00000	lambda at prev. ctr. and at node
1.32174E+06	1.31435E+06		lambda at prev. ctr. and at node
54.52380	-25.40435	.00000	x, y and z position at node # 3
.91871	-.40145	.00000	x', y' and z' slopes
.00000	.00000	.00000	x-dot, y-dot and z-dot
.00000	.00000	.00000	lambda at prev. ctr. and at node
1.30715E+06	1.30015E+06		lambda at prev. ctr. and at node
82.22988	-37.10792	.00000	x, y and z position at node # 4
.92830	-.37867	.00000	x', y' and z' slopes
.00000	.00000	.00000	x-dot, y-dot and z-dot
.00000	.00000	.00000	lambda at prev. ctr. and at node
1.29333E+06	1.28671E+06		lambda at prev. ctr. and at node
110.21812	-48.11729	.00000	x, y and z position at node # 5
.93751	-.35518	.00000	x', y' and z' slopes
.00000	.00000	.00000	x-dot, y-dot and z-dot
.00000	.00000	.00000	lambda at prev. ctr. and at node
1.28029E+06	1.27407E+06		lambda at prev. ctr. and at node
138.47631	-58.41161	.00000	x, y and z position at node # 6
.94629	-.33100	.00000	x', y' and z' slopes
.00000	.00000	.00000	x-dot, y-dot and z-dot
.00000	.00000	.00000	lambda at prev. ctr. and at node
1.26805E+06	1.26225E+06		lambda at prev. ctr. and at node
166.99083	-67.97062	.00000	x, y and z position at node # 7
.95459	-.30616	.00000	x', y' and z' slopes
.00000	.00000	.00000	x-dot, y-dot and z-dot
.00000	.00000	.00000	lambda at prev. ctr. and at node
1.25665E+06	1.25127E+06		lambda at prev. ctr. and at node
195.74662	-76.77482	.00000	x, y and z position at node # 8
.96237	-.28068	.00000	x', y' and z' slopes
.00000	.00000	.00000	x-dot, y-dot and z-dot
.00000	.00000	.00000	lambda at prev. ctr. and at node
1.24611E+06	1.24116E+06		lambda at prev. ctr. and at node
224.72725	-84.80564	.00000	x, y and z position at node # 9
.96957	-.25461	.00000	x', y' and z' slopes
.00000	.00000	.00000	x-dot, y-dot and z-dot
.00000	.00000	.00000	lambda at prev. ctr. and at node
1.23644E+06	1.23194E+06		lambda at prev. ctr. and at node
253.91489	-92.04558	.00000	x, y and z position at node # 10
.97616	-.22797	.00000	x', y' and z' slopes
.00000	.00000	.00000	x-dot, y-dot and z-dot
.00000	.00000	.00000	lambda at prev. ctr. and at node
1.22767E+06	1.22362E+06		lambda at prev. ctr. and at node
283.29041	-98.47842	.00000	x, y and z position at node # 11
.98209	-.20081	.00000	x', y' and z' slopes
.00000	.00000	.00000	x-dot, y-dot and z-dot
.00000	.00000	.00000	lambda at prev. ctr. and at node
1.21981E+06	1.21623E+06		lambda at prev. ctr. and at node
312.83346	-104.08939	.00000	x, y and z position at node # 12
.98732	-.17318	.00000	x', y' and z' slopes
.00000	.00000	.00000	x-dot, y-dot and z-dot
.00000	.00000	.00000	lambda at prev. ctr. and at node
1.21289E+06	1.20979E+06		lambda at prev. ctr. and at node
342.52254	-108.86528	.00000	x, y and z position at node # 13
.99182	-.14515	.00000	x', y' and z' slopes
.00000	.00000	.00000	x-dot, y-dot and z-dot
.00000	.00000	.00000	lambda at prev. ctr. and at node
1.20693E+06	1.20430E+06		lambda at prev. ctr. and at node
372.33512	-112.79466	.00000	x, y and z position at node # 14
.99555	-.11676	.00000	x', y' and z' slopes
.00000	.00000	.00000	x-dot, y-dot and z-dot
.00000	.00000	.00000	lambda at prev. ctr. and at node
1.20192E+06	1.19979E+06		lambda at prev. ctr. and at node
402.24781	-115.86794	.00000	x, y and z position at node # 15
.99849	-.08809	.00000	x', y' and z' slopes
.00000	.00000	.00000	x-dot, y-dot and z-dot
.00000	.00000	.00000	lambda at prev. ctr. and at node
1.19790E+06	1.19626E+06		lambda at prev. ctr. and at node

432.23645	-118.07767	.00000	x, y and z position at node # 16
1.00061	-.05929	.00000	x', y' and z' slopes
.00000	.00000	.00000	x-dot, y-dot and z-dot
.00000	.00000	.00000	lambda at prev. ctr. and at node
1.19487E+06	1.19372E+06		lambda at prev. ctr. and at node
462.27636	-119.41901	.00000	x, y and z position at node # 17
1.00190	-.03049	.00000	x', y' and z' slopes
.00000	.00000	.00000	x-dot, y-dot and z-dot
.00000	.00000	.00000	lambda at prev. ctr. and at node
1.19283E+06	1.19219E+06		lambda at prev. ctr. and at node
492.34234	-119.91078	.00000	x, y and z position at node # 18
1.00234	-.00546	.00000	x', y' and z' slopes
.00000	.00000	.00000	x-dot, y-dot and z-dot
.00000	.00000	.00000	lambda at prev. ctr. and at node
1.19179E+06	1.19167E+06		lambda at prev. ctr. and at node
522.41297	-119.98716	.00000	x, y and z position at node # 19
1.00236	-.00084	.00000	x', y' and z' slopes
.00000	.00000	.00000	x-dot, y-dot and z-dot
.00000	.00000	.00000	lambda at prev. ctr. and at node
1.19165E+06	1.19165E+06		lambda at prev. ctr. and at node
552.48373	-119.99815	.00000	x, y and z position at node # 20
1.00236	-.00013	.00000	x', y' and z' slopes
.00000	.00000	.00000	x-dot, y-dot and z-dot
.00000	.00000	.00000	lambda at prev. ctr. and at node
1.19164E+06	1.19164E+06		lambda at prev. ctr. and at node
582.55451	-119.99973	.00000	x, y and z position at node # 21
1.00236	-.00002	.00000	x', y' and z' slopes
.00000	.00000	.00000	x-dot, y-dot and z-dot
.00000	.00000	.00000	lambda at prev. ctr. and at node
1.19164E+06	1.19164E+06		lambda at prev. ctr. and at node
612.62528	-119.99996	.00000	x, y and z position at node # 22
1.00236	.00000	.00000	x', y' and z' slopes
.00000	.00000	.00000	x-dot, y-dot and z-dot
.00000	.00000	.00000	lambda at prev. ctr. and at node
1.19164E+06	1.19164E+06		lambda at prev. ctr. and at node
642.69606	-119.99999	.00000	x, y and z position at node # 23
1.00236	.00000	.00000	x', y' and z' slopes
.00000	.00000	.00000	x-dot, y-dot and z-dot
.00000	.00000	.00000	lambda at prev. ctr. and at node
1.19164E+06	1.19164E+06		lambda at prev. ctr. and at node
672.76683	-120.00000	.00000	x, y and z position at node # 24
1.00236	.00000	.00000	x', y' and z' slopes
.00000	.00000	.00000	x-dot, y-dot and z-dot
.00000	.00000	.00000	lambda at prev. ctr. and at node
1.19164E+06	1.19164E+06		lambda at prev. ctr. and at node
702.83761	-120.00000	.00000	x, y and z position at node # 25
1.00236	.00000	.00000	x', y' and z' slopes
.00000	.00000	.00000	x-dot, y-dot and z-dot
.00000	.00000	.00000	lambda at prev. ctr. and at node
1.19164E+06	1.19164E+06		lambda at prev. ctr. and at node
732.90838	-120.00000	.00000	x, y and z position at node # 26
1.00236	.00000	.00000	x', y' and z' slopes
.00000	.00000	.00000	x-dot, y-dot and z-dot
.00000	.00000	.00000	lambda at prev. ctr. and at node
1.19164E+06	1.19164E+06		lambda at prev. ctr. and at node
762.97916	-120.00000	.00000	x, y and z position at node # 27
1.00236	.00000	.00000	x', y' and z' slopes
.00000	.00000	.00000	x-dot, y-dot and z-dot
.00000	.00000	.00000	lambda at prev. ctr. and at node
1.19164E+06	1.19164E+06		lambda at prev. ctr. and at node
793.04993	-120.00000	.00000	x, y and z position at node # 28
1.00236	.00000	.00000	x', y' and z' slopes
.00000	.00000	.00000	x-dot, y-dot and z-dot
.00000	.00000	.00000	lambda at prev. ctr. and at node
1.19164E+06	1.19164E+06		lambda at prev. ctr. and at node
823.12070	-120.00000	.00000	x, y and z position at node # 29
1.00236	.00000	.00000	x', y' and z' slopes
.00000	.00000	.00000	x-dot, y-dot and z-dot
.00000	.00000	.00000	lambda at prev. ctr. and at node
1.19164E+06	1.19164E+06		lambda at prev. ctr. and at node
853.19148	-120.00000	.00000	x, y and z position at node # 30
1.00236	.00000	.00000	x', y' and z' slopes
.00000	.00000	.00000	x-dot, y-dot and z-dot
.00000	.00000	.00000	lambda at prev. ctr. and at node
1.19164E+06	1.19164E+06		lambda at prev. ctr. and at node
883.26225	-120.00000	.00000	x, y and z position at node # 31
1.00236	.00000	.00000	x', y' and z' slopes
.00000	.00000	.00000	x-dot, y-dot and z-dot
.00000	.00000	.00000	lambda at prev. ctr. and at node
1.19164E+06	1.19164E+06		lambda at prev. ctr. and at node
913.33303	-120.00000	.00000	x, y and z position at node # 32
1.00236	.00000	.00000	x', y' and z' slopes
.00000	.00000	.00000	x-dot, y-dot and z-dot
.00000	.00000	.00000	lambda at prev. ctr. and at node
1.19164E+06	1.19164E+06		lambda at prev. ctr. and at node
943.40380	-120.00000	.00000	x, y and z position at node # 33
1.00236	.00000	.00000	x', y' and z' slopes
.00000	.00000	.00000	x-dot, y-dot and z-dot
.00000	.00000	.00000	lambda at prev. ctr. and at node
1.19164E+06	1.19164E+06		lambda at prev. ctr. and at node
973.47458	-120.00000	.00000	x, y and z position at node # 34
1.00236	.00000	.00000	x', y' and z' slopes
.00000	.00000	.00000	x-dot, y-dot and z-dot
.00000	.00000	.00000	lambda at prev. ctr. and at node
1.19164E+06	1.19164E+06		lambda at prev. ctr. and at node
1003.54535	-120.00000	.00000	x, y and z position at node # 35
1.00236	.00000	.00000	x', y' and z' slopes
.00000	.00000	.00000	x-dot, y-dot and z-dot
.00000	.00000	.00000	lambda at prev. ctr. and at node
1.19164E+06	1.19164E+06		lambda at prev. ctr. and at node
1033.61613	-120.00000	.00000	x, y and z position at node # 36
1.00236	.00000	.00000	x', y' and z' slopes
.00000	.00000	.00000	x-dot, y-dot and z-dot
.00000	.00000	.00000	lambda at prev. ctr. and at node
1.19164E+06	1.19164E+06		lambda at prev. ctr. and at node
1063.68690	-120.00000	.00000	x, y and z position at node # 37
1.00236	.00000	.00000	x', y' and z' slopes

```
      .00000       .00000       .00000       x-dot, y-dot and z-dot
      .00000       .00000       .00000       lambda at prev. ctr. and at node
 1.19164E+06  1.19164E+06                     lambda at prev. ctr. and at node

   1093.75768   -120.00000       .00000       x, y and z position at node # 38
      1.00236       .00000       .00000       x', y' and z' slopes
      .00000       .00000       .00000       x-dot, y-dot and z-dot
      .00000       .00000       .00000       lambda at prev. ctr. and at node
 1.19164E+06  1.19164E+06                     lambda at prev. ctr. and at node

   1123.82845   -120.00000       .00000       x, y and z position at node # 39
      1.00236       .00000       .00000       x', y' and z' slopes
      .00000       .00000       .00000       x-dot, y-dot and z-dot
      .00000       .00000       .00000       lambda at prev. ctr. and at node
 1.19164E+06  1.19164E+06                     lambda at prev. ctr. and at node

   1153.89923   -120.00000       .00000       x, y and z position at node # 40
      1.00236       .00000       .00000       x', y' and z' slopes
      .00000       .00000       .00000       x-dot, y-dot and z-dot
      .00000       .00000       .00000       lambda at prev. ctr. and at node
 1.19164E+06  1.19164E+06                     lambda at prev. ctr. and at node

   1183.97000   -120.00000       .00000       x, y and z position at node # 41
      1.00236       .00000       .00000       x', y' and z' slopes
      .00000       .00000       .00000       x-dot, y-dot and z-dot
      .00000       .00000       .00000       lambda at prev. ctr. and at node
 1.19164E+06  1.19164E+06                     lambda at prev. ctr. and at node

c. Connections between Nodes
  length   elmt type
    30.0        1        from node  1 to  2
    30.0        1        from node  2 to  3
    30.0        1        from node  3 to  4
    30.0        1        from node  4 to  5
    30.0        1        from node  5 to  6
    30.0        1        from node  6 to  7
    30.0        1        from node  7 to  8
    30.0        1        from node  8 to  9
    30.0        1        from node  9 to 10
    30.0        1        from node 10 to 11
    30.0        1        from node 11 to 12
    30.0        1        from node 12 to 13
    30.0        1        from node 13 to 14
    30.0        1        from node 14 to 15
    30.0        1        from node 15 to 16
    30.0        1        from node 16 to 17
    30.0        1        from node 17 to 18
    30.0        1        from node 18 to 19
    30.0        1        from node 19 to 20
    30.0        1        from node 20 to 21
    30.0        1        from node 21 to 22
    30.0        1        from node 22 to 23
    30.0        1        from node 23 to 24
    30.0        1        from node 24 to 25
    30.0        1        from node 25 to 26
    30.0        1        from node 26 to 27
    30.0        1        from node 27 to 28
    30.0        1        from node 28 to 29
    30.0        1        from node 29 to 30
    30.0        1        from node 30 to 31
    30.0        1        from node 31 to 32
    30.0        1        from node 32 to 33
    30.0        1        from node 33 to 34
    30.0        1        from node 34 to 35
    30.0        1        from node 35 to 36
    30.0        1        from node 36 to 37
    30.0        1        from node 37 to 38
    30.0        1        from node 38 to 39
    30.0        1        from node 39 to 40
    30.0        1        from node 40 to 41


III. Nodal Constraints

Node No.  1

a. Linear Constraints
            direction consines                reference
   reference   1st perp.   2nd perp.         position
     1.00000      .00000      .00000             .000    x direction
      .00000     1.00000      .00000             .000    y direction
      .00000      .00000     1.00000             .000    z direction
 1.00000E+11 1.00000E+11 1.00000E+11                     spring constant
      .00000      .00000      .00000                     damping constant
 0.00000E+00 0.00000E+00 0.00000E+00                     initial force
      .00000                                             concentrated mass

b. Motion of Reference Point at Node  1

    .0000000000            motion frequencey (rad/sec)

  in-phase out-phase
      .000       .000      x displacement amplitude
      .000       .000      y displacement amplitude
      .000       .000      z displacement amplitude

Node No. 41

a. Linear Constraints
            direction consines                reference
   reference   1st perp.   2nd perp.         position
     1.00000      .00000      .00000          1183.970    x direction
      .00000     1.00000      .00000           120.000    y direction
      .00000      .00000     1.00000              .000    z direction
 1.00000E+11 1.00000E+11 1.00000E+11                     spring constant
      .00000      .00000      .00000                     damping constant
 0.00000E+00 0.00000E+00 0.00000E+00                     initial force
      .00000                                             concentrated mass

b. Motion of Reference Point at Node 41

    .0000000000            motion frequencey (rad/sec)

  in-phase out-phase
      .000       .000      x displacement amplitude
      .000       .000      y displacement amplitude
      .000       .000      z displacement amplitude


IV. Environmental Data

a. Currents at  4 depths

    depth       x-vel      y-vel      z-vel
     .0000      .0000      .0000      .0000
  -40.0000      .0000      .0000      .0000
  -80.0000      .0000      .0000      .0000
 -120.0000      .0000      .0000      .0000
```

b. Waves

```
   .000000     .00000        wave amplitude and frequency (rad/sec)
  .0000000                   wave direction relative to x axis
```

Static analysis of a cable with the bottom presenting. 05/24/94

I. Global Constants

a. Indexs

41	NODE	(# of nodes)
1	NELTYP	(# of different element types)
3	NDIM	(# of dimensions - 2 or 3)
2	JOB	(1 = static, 2 = dynamic)
50	JSNAP	(snapshot each JSNAP iterations)
1	JMOTN	(1 = sin. motion, 2 = input forced motion)
2	NCNSTR	(# of constrained nodes)

b. Parameters

.20000	.00040	1201	relaxation, # iterations
.00000	120.00000	.10000	begin time, end time & time step
1.00000	3.80000		drag and added-mass coefficients
1000.00000	.00000	9.80665	mass density of water, fluid; gravity
-120.00			bottom depth

II. Element and Nodal Information

a. Physical Properties for Each Element Type

5.51090E+10			Young's mod: type 1
0.00000E+00	0.00000E+00	0.00000E+00	moment of inertia (I)
9.07292E-03	9.07292E-03	5.07292E-03	cross-section area (Af)
0.00000E+00	0.00000E+00	0.00000E+00	internal cavity area (Ai)
2.28000E-01	2.28000E-01	2.28000E-01	dist. to outermost fiber
1.35350E+02	1.35350E+02	1.35350E+02	mass of the rod per unit length
7.60000E-02	7.60000E-02	7.60000E-02	diameters for hydrodynamic forces
1.70171E+02	1.70171E+02	1.70171E+02	total buoyancy/unit length

b. Coordinates and Lambdas for Each Node

.00003	.00000	.00000	x, y and z position at node # 1
.89072	-.45749	-.01310	x', y' and z' slopes
2.12791	.00000	.87965	x-dot, y-dot and z-dot
-.02296	-.04366	-.00514	lambda at prev. ctr. and at node
0.00000E+00	7.09831E+05		lambda at prev. ctr. and at node
27.01444	-13.13239	-.37755	x, y and z position at node # 2
.90974	-.41833	-.01212	x', y' and z' slopes
1.46889	-1.32102	.72920	x-dot, y-dot and z-dot
-.01519	-.03146	-.00500	lambda at prev. ctr. and at node
7.02353E+05	6.94954E+05		lambda at prev. ctr. and at node
54.56179	-25.10610	-.72094	x, y and z position at node # 3
.92619	-.38050	-.01067	x', y' and z' slopes
.99737	-2.35524	.57993	x-dot, y-dot and z-dot
-.01010	-.02325	-.00473	lambda at prev. ctr. and at node
6.88107E+05	6.81067E+05		lambda at prev. ctr. and at node
82.55543	-35.99595	-1.00965	x, y and z position at node # 4
.93944	-.34647	-.00857	x', y' and z' slopes
.68532	-3.11737	.44533	x-dot, y-dot and z-dot
-.00717	-.01921	-.00404	lambda at prev. ctr. and at node
6.74636E+05	6.67855E+05		lambda at prev. ctr. and at node
110.89820	-45.94264	-1.23386	x, y and z position at node # 5
.94965	-.31737	-.00640	x', y' and z' slopes
.50002	-1.62028	.33234	x-dot, y-dot and z-dot
-.00737	-.02134	-.00341	lambda at prev. ctr. and at node
6.61970E+05	6.55187E+05		lambda at prev. ctr. and at node
139.50954	-55.08780	-1.39701	x, y and z position at node # 6
.95769	-.29218	-.00444	x', y' and z' slopes
.38668	-3.94322	.24079	x-dot, y-dot and z-dot
-.00858	-.02609	-.00286	lambda at prev. ctr. and at node
6.49568E+05	6.44045E+05		lambda at prev. ctr. and at node
168.33420	-63.53572	-1.50534	x, y and z position at node # 7
.96427	-.26962	-.00271	x', y' and z' slopes
.31208	-4.14472	.17146	x-dot, y-dot and z-dot
-.00833	-.02688	-.00222	lambda at prev. ctr. and at node
6.39236E+05	6.35111E+05		lambda at prev. ctr. and at node
197.33777	-71.36494	-1.56552	x, y and z position at node # 8
.96967	-.24944	-.00127	x', y' and z' slopes
.25317	-4.27529	.11802	x-dot, y-dot and z-dot
-.00524	-.01848	-.00159	lambda at prev. ctr. and at node
6.31217E+05	6.26882E+05		lambda at prev. ctr. and at node
226.47651	-78.63347	-1.58516	x, y and z position at node # 9
.97410	-.23145	-.00005	x', y' and z' slopes
.20898	-4.36336	.07742	x-dot, y-dot and z-dot
-.00091	-.00181	-.00108	lambda at prev. ctr. and at node
6.22335E+05	6.16450E+05		lambda at prev. ctr. and at node
255.74597	-85.37520	-1.56905	x, y and z position at node # 10
.97778	-.21530	.00108	x', y' and z' slopes
.16906	-4.43857	.04841	x-dot, y-dot and z-dot
.00242	.01321	-.00084	lambda at prev. ctr. and at node
6.10551E+05	6.04469E+05		lambda at prev. ctr. and at node
285.12534	-91.61736	-1.52054	x, y and z position at node # 11
.98097	-.20012	.00211	x', y' and z' slopes
.13628	-4.49707	.02450	x-dot, y-dot and z-dot
.00426	.02256	-.00077	lambda at prev. ctr. and at node
5.99730E+05	5.94504E+05		lambda at prev. ctr. and at node
314.60359	-97.37155	-1.44204	x, y and z position at node # 12
.98193	-.18491	.00308	x', y' and z' slopes
.11488	-4.51709	.00454	x-dot, y-dot and z-dot
.00427	.02585	-.00074	lambda at prev. ctr. and at node
5.92048E+05	5.87994E+05		lambda at prev. ctr. and at node
344.17208	-102.63982	-1.33623	x, y and z position at node # 13
.98670	-.16947	.00394	x', y' and z' slopes
.09748	-4.52007	-.01321	x-dot, y-dot and z-dot
.00313	.02263	-.00054	lambda at prev. ctr. and at node
5.86635E+05	5.83805E+05		lambda at prev. ctr. and at node
373.82320	-107.41865	-1.20648	x, y and z position at node # 14
.98939	-.15290	.00465	x', y' and z' slopes
.09425	-4.48293	-.02862	x-dot, y-dot and z-dot
.00214	.01654	-.00047	lambda at prev. ctr. and at node
5.82688E+05	5.80974E+05		lambda at prev. ctr. and at node
403.55765	-111.68037	-1.05858	x, y and z position at node # 15
.99199	-.13483	.00515	x', y' and z' slopes
.09808	-4.40726	-.04479	x-dot, y-dot and z-dot
.00074	.00713	-.00045	lambda at prev. ctr. and at node
5.79670E+05	5.76854E+05		lambda at prev. ctr. and at node

433.35654	-115.38193	-.89883	x, y and z position at node # 16
.99449	-.11471	.00545	x', y' and z' slopes
.10177	-4.27268	-.05824	x-dot, y-dot and z-dot
-.00059	-.00171	-.00052	lambda at prev. ctr. and at node
5.75772E+05	5.71522E+05		lambda at prev. ctr. and at node
463.23697	-118.39296	-.73448	x, y and z position at node # 17
.99720	-.08675	.00550	x', y' and z' slopes
.13525	-3.84939	-.07278	x-dot, y-dot and z-dot
-.00139	-.01004	-.00053	lambda at prev. ctr. and at node
5.70905E+05	5.66773E+05		lambda at prev. ctr. and at node
493.21898	-119.96990	-.57480	x, y and z position at node # 18
1.00120	-.00095	.00512	x', y' and z' slopes
.28497	-.46326	-.09987	x-dot, y-dot and z-dot
-.00227	.12399	-.00061	lambda at prev. ctr. and at node
5.65040E+05	5.63159E+05		lambda at prev. ctr. and at node
523.25187	-119.99414	-.41707	x, y and z position at node # 19
1.00111	.00309	.00535	x', y' and z' slopes
.27391	-.09188	-.10111	x-dot, y-dot and z-dot
-.00097	.02276	.00098	lambda at prev. ctr. and at node
5.61605E+05	5.61822E+05		lambda at prev. ctr. and at node
553.28450	-120.00156	-.25376	x, y and z position at node # 20
1.00109	.00297	.00553	x', y' and z' slopes
.25619	-.03783	-.09920	x-dot, y-dot and z-dot
-.00074	.00630	.00111	lambda at prev. ctr. and at node
5.61978E+05	5.61568E+05		lambda at prev. ctr. and at node
583.31715	-120.00632	-.08854	x, y and z position at node # 21
1.00109	.00172	.00555	x', y' and z' slopes
.23700	.02715	-.08984	x-dot, y-dot and z-dot
-.00044	.00477	.00094	lambda at prev. ctr. and at node
5.63617E+05	5.61440E+05		lambda at prev. ctr. and at node
613.34978	-120.00226	.07483	x, y and z position at node # 22
1.00108	.00032	.00542	x', y' and z' slopes
.22627	.05430	-.07868	x-dot, y-dot and z-dot
-.00031	.01685	.00059	lambda at prev. ctr. and at node
5.62806E+05	5.60564E+05		lambda at prev. ctr. and at node
643.38233	-119.99911	.23151	x, y and z position at node # 23
1.00108	-.00052	.00508	x', y' and z' slopes
.21091	.02566	-.06200	x-dot, y-dot and z-dot
-.00013	.03287	.00034	lambda at prev. ctr. and at node
5.60318E+05	5.58331E+05		lambda at prev. ctr. and at node
673.41478	-120.00056	.37865	x, y and z position at node # 24
1.00108	-.00063	.00473	x', y' and z' slopes
.19325	-.01242	-.04603	x-dot, y-dot and z-dot
.00004	.03858	.00003	lambda at prev. ctr. and at node
5.58002E+05	5.55676E+05		lambda at prev. ctr. and at node
703.44728	-120.00142	.51399	x, y and z position at node # 25
1.00108	-.00059	.00424	x', y' and z' slopes
.18002	-.02212	-.02478	x-dot, y-dot and z-dot
-.00061	.03381	.00057	lambda at prev. ctr. and at node
5.57774E+05	5.57345E+05		lambda at prev. ctr. and at node
733.48003	-120.00004	.63195	x, y and z position at node # 26
1.00110	-.00061	.00360	x', y' and z' slopes
.16572	-.02709	.00615	x-dot, y-dot and z-dot
-.00104	.01952	.00099	lambda at prev. ctr. and at node
5.60858E+05	5.62765E+05		lambda at prev. ctr. and at node
763.51303	-119.99935	.72879	x, y and z position at node # 27
1.00110	-.00073	.00293	x', y' and z' slopes
.14928	-.03681	.03584	x-dot, y-dot and z-dot
-.00056	.00031	.00109	lambda at prev. ctr. and at node
5.63793E+05	5.64344E+05		lambda at prev. ctr. and at node
793.54610	-119.99895	.80631	x, y and z position at node # 28
1.00110	-.00099	.00233	x', y' and z' slopes
.13209	-.05042	.05815	x-dot, y-dot and z-dot
.00003	-.01960	.00089	lambda at prev. ctr. and at node
5.63947E+05	5.62223E+05		lambda at prev. ctr. and at node
823.57900	-120.00059	.86701	x, y and z position at node # 29
1.00109	-.00139	.00178	x', y' and z' slopes
.11452	-.04294	.07444	x-dot, y-dot and z-dot
-.00007	-.03967	.00082	lambda at prev. ctr. and at node
5.60469E+05	5.57890E+05		lambda at prev. ctr. and at node
853.61169	-120.00092	.91257	x, y and z position at node # 30
1.00109	-.00162	.00123	x', y' and z' slopes
.10307	-.02771	.08514	x-dot, y-dot and z-dot
-.00079	-.05799	.00021	lambda at prev. ctr. and at node
5.56121E+05	5.54807E+05		lambda at prev. ctr. and at node
883.64432	-119.99955	.94217	x, y and z position at node # 31
1.00109	-.00122	.00072	x', y' and z' slopes
.09861	-.00581	.09459	x-dot, y-dot and z-dot
-.00060	-.07027	-.00023	lambda at prev. ctr. and at node
5.54775E+05	5.54782E+05		lambda at prev. ctr. and at node
913.67705	-119.99842	.95474	x, y and z position at node # 32
1.00109	-.00034	.00015	x', y' and z' slopes
.10184	.02352	.09802	x-dot, y-dot and z-dot
.00029	-.06462	-.00033	lambda at prev. ctr. and at node
5.56120E+05	5.57303E+05		lambda at prev. ctr. and at node
943.70994	-119.99866	.94818	x, y and z position at node # 33
1.00110	.00031	-.00055	x', y' and z' slopes
.10294	.04757	.09599	x-dot, y-dot and z-dot
.00050	-.04350	-.00060	lambda at prev. ctr. and at node
5.58636E+05	5.60366E+05		lambda at prev. ctr. and at node
973.74290	-119.99999	.92064	x, y and z position at node # 34
1.00110	-.00013	-.00133	x', y' and z' slopes
.08867	.07458	.08799	x-dot, y-dot and z-dot
-.00039	-.01448	-.00053	lambda at prev. ctr. and at node
5.60240E+05	5.60351E+05		lambda at prev. ctr. and at node
1003.77581	-120.00441	.86900	x, y and z position at node # 35
1.00109	-.00115	-.00215	x', y' and z' slopes
.06469	.06800	.06896	x-dot, y-dot and z-dot
-.00098	.01700	-.00100	lambda at prev. ctr. and at node
5.60455E+05	5.59236E+05		lambda at prev. ctr. and at node
1033.80861	-120.00313	.79245	x, y and z position at node # 36
1.00109	-.00147	-.00304	x', y' and z' slopes
.04432	.03110	.04232	x-dot, y-dot and z-dot
-.00006	.03955	-.00094	lambda at prev. ctr. and at node
5.59249E+05	5.58650E+05		lambda at prev. ctr. and at node
1063.84119	-119.99603	.68624	x, y and z position at node # 37
1.00108	-.00180	-.00405	x', y' and z' slopes

```
     .03817      -.04791       .01048       x-dot, y-dot and z-dot
     .00069       .04376      -.00154       lambda at prev. ctr. and at node
 5.56502E+05 5.57253E+05                    lambda at prev. ctr. and at node

  1093.87357  -119.99523       .55325       x, y and z position at node # 38
     1.00108      -.00059      -.00482       x', y' and z' slopes
      .04006      -.02377      -.03574       x-dot, y-dot and z-dot
      .00006       .03330      -.00151       lambda at prev. ctr. and at node
 5.54736E+05 5.56480E+05                    lambda at prev. ctr. and at node

  1123.90583  -119.99470       .39595       x, y and z position at node # 39
     1.00107       .00210      -.00572       x', y' and z' slopes
      .03676       .02095      -.06038       x-dot, y-dot and z-dot
     -.00041       .01103      -.00008       lambda at prev. ctr. and at node
 5.55211E+05 5.56316E+05                    lambda at prev. ctr. and at node

  1153.93796  -119.99787       .20932       x, y and z position at node # 40
     1.00107       .00408      -.00670       x', y' and z' slopes
      .02624      -.06405      -.04276       x-dot, y-dot and z-dot
     -.00004       .00576       .00125       lambda at prev. ctr. and at node
 5.56911E+05 5.57558E+05                    lambda at prev. ctr. and at node

  1183.97000  -120.00000       .00000       x, y and z position at node # 41
     1.00106       .00449      -.00714       x', y' and z' slopes
      .00000       .00000       .00000       x-dot, y-dot and z-dot
      .00023       .01035       .00169       lambda at prev. ctr. and at node
 5.58501E+05 5.58853E+05                    lambda at prev. ctr. and at node

c. Connections between Nodes
  length   elmt type
    30.0       1       from node  1 to  2
    30.0       1       from node  2 to  3
    30.0       1       from node  3 to  4
    30.0       1       from node  4 to  5
    30.0       1       from node  5 to  6
    30.0       1       from node  6 to  7
    30.0       1       from node  7 to  8
    30.0       1       from node  8 to  9
    30.0       1       from node  9 to 10
    30.0       1       from node 10 to 11
    30.0       1       from node 11 to 12
    30.0       1       from node 12 to 13
    30.0       1       from node 13 to 14
    30.0       1       from node 14 to 15
    30.0       1       from node 15 to 16
    30.0       1       from node 16 to 17
    30.0       1       from node 17 to 18
    30.0       1       from node 18 to 19
    30.0       1       from node 19 to 20
    30.0       1       from node 20 to 21
    30.0       1       from node 21 to 22
    30.0       1       from node 22 to 23
    30.0       1       from node 23 to 24
    30.0       1       from node 24 to 25
    30.0       1       from node 25 to 26
    30.0       1       from node 26 to 27
    30.0       1       from node 27 to 28
    30.0       1       from node 28 to 29
    30.0       1       from node 29 to 30
    30.0       1       from node 30 to 31
    30.0       1       from node 31 to 32
    30.0       1       from node 32 to 33
    30.0       1       from node 33 to 34
    30.0       1       from node 34 to 35
    30.0       1       from node 35 to 36
    30.0       1       from node 36 to 37
    30.0       1       from node 37 to 38
    30.0       1       from node 38 to 39
    30.0       1       from node 39 to 40
    30.0       1       from node 40 to 41


III. Nodal Constraints

Node No.  1

a. Linear Constraints
          direction consines                reference
   reference   1st perp.   2nd perp.        position
     1.00000      .00000      .00000            .000    x direction
      .00000     1.00000      .00000            .000    y direction
      .00000      .00000     1.00000            .000    z direction
 0.00000E+00 1.00000E+11 0.00000E+00                    spring constant
      .00000      .00000      .00000                    damping constant
 0.00000E+00 0.00000E+00 0.00000E+00                    initial force
      .00000                                            concentrated mass

b. Motion of Reference Point at Node  1

    .4188790205            motion frequencey (rad/sec)

  in-phase out-phase
     5.080       .000       x displacement amplitude
      .000       .000       y displacement amplitude
     2.100       .000       z displacement amplitude

Node No. 41

a. Linear Constraints
          direction consines                reference
   reference   1st perp.   2nd perp.        position
     1.00000      .00000      .00000         1183.970    x direction
      .00000     1.00000      .00000          120.000    y direction
      .00000      .00000     1.00000             .000    z direction
 1.00000E+11 1.00000E+11 1.00000E+11                    spring constant
      .00000      .00000      .00000                    damping constant
 0.00000E+00 0.00000E+00 0.00000E+00                    initial force
      .00000                                            concentrated mass

b. Motion of Reference Point at Node 41

    .0000000000            motion frequencey (rad/sec)

  in-phase out-phase
      .000       .000       x displacement amplitude
      .000       .000       y displacement amplitude
      .000       .000       z displacement amplitude


IV. Environmental Data

a. Currents at  4 depths

    depth       x-vel      y-vel      z-vel
     .0000      .0000      .0000      .0000
   -40.0000     .0000      .0000      .0000
   -80.0000     .0000      .0000      .0000
  -120.0000     .0000      .0000      .0000
```

b. Waves

```
   .000000     .00000        wave amplitude and frequency (rad/sec)
  .0000000                   wave direction relative to x axis
```

Appendix III. The Source Code

parametrics.inc

```
      common/index/     jmotn,job,jsnap,nconstr,ndim,neltyp,node
      common/paramt/    btmdpth,cd,cm,g,relax,damp,ruf,rul,
     1                  tbgn,tend,tstep,iter
      common/coefshp/   alpha(4,4,3),beta(4,4,3),eta(3,3),gamma(4,4,3),
     1                  omega(4,4,3),tau(3),mu(4,3)
      common/conctn/    lnthel(50),jeltyp(50)
      common/prptyel/   afel(10,3),   ajel(10,3),   buoyel(10,3),
     1                  del(10,3),    dfel(10,3),   eel(10),
     2                  mel(10,3),    moi(10,3)
      common/ndconstr/  cmass(50),        finitl(50,3),
     1                  refdir(50,3),     refcos(50,3,3),  refpos(50,3),
     1                  spgcont(50,3),    dmpcont(50,3),   freq(50),
     2                  dlsamp(50,3,2),     nodec(50)
      common/environ/   aw,dirw,sigmaw,vc(10,4),nvc
      common/tapes/     lkey,iscr,idatin,idatout,isnap,itimhis,ifmotn
      double precision mu,lnthel,mel,moi
```

variables.inc


```
      common/varlq/    e(3),    lmda(3),    lmda05(3), massmat(4,3,3),
     1                 q(3,3), wbd(3),
     2                 u(4,3), udot(4,3), uddot(4,3),u05(4,3)
      common/coord/    lambda(50,2), lmdap(50,2),
     1                 r(50,3),      rdot(50,3),   rdotp(50,3),
     2                 rprm(50,3),   rpmdt(50,3), rpmdtp(50,3)
      common/eqtion/   a(400,400),b(400)
      double precision lmda,lmda05,massmat,lambda,lmdap
```

```
      PROGRAM cable
c
c     Program CABLE solves cable staitc and dynamic problems.
c     In this section of the program, names of the input
c     and output files are defined. Routine INPUT, OUTPUT, and
c     SLVEQ16 are called. Routine INPUT reads all the necessary
c     data required for calculation. Routine OUPUT writes out
c     calculated results. Routine SLVEQ16 is the key routine in
c     this program and is used to solve cable static and
c     dynamic problems.
c
c     Variables   -
c
c        idatin   - access code to the file named "namein"
c        idatout  - access code to the file named "nameout"
c        ikey     - access code to the keyboard
c        iscr     - access code to the screen
c        isnap    - access code to file named "snapshot"
c        itimhis  - access code to file named "timeseries"
c        ifmotn   - access code to file named "forcedmotion"
c        namein   - name of the input file defined by the user
c        nameout  - name of the output file defined by the user
c
      include 'parametrics.inc'
      character namein*80,nameout*80

      ikey   = 5
      iscr   = 6
      idatin = 7
      idatout = 8
      isnap  = 9
      itimhis = 10
      ifmotn = 11
c
c     Input names of input and output files.
c
      write(iscr,1000)
      read(ikey,'(a)') namein
      write(iscr,1010)
      read(ikey,'(a)') nameout

      open(idatin, file = namein)
      open(idatout,file = nameout)
      open(isnap,  file = 'snapshot')
      open(itimhis,file = 'timeseries')
      open(ifmotn, file = 'forcedmotion')
c
c     Read initial configuration of the cables from the input file
c
      call input
c
c     Call routine SLVQE16 to solve the problem
c
      call slveq16
c
c     Output calculated results to the output file
c
      call output

      stop
1000  format(1x,'Welcome to the CABLE Computer Code'/
     1       1x,'Please type in the name of the input file:')
1010  format(1x,'and also the name of the output file:')
      end


      SUBROUTINE assemb(s,f,nel,nd)
c
c     Routine ASSEMB assembles linear element equilibrium equations
c     S*du = F into a global equilibrium equation Ax = B.
c
c     Variables   -
c        a        -   global stiffness matrix
c        b        -   global right-hand side vector
c        f        -   element right-hand side vector
c        nd       -   dimension of freedom of the element
c        nel      -   element #
c        s        -   element stiffness matrix
c
c     Subroutine called by SLVEQ15
c
      implicit double precision (a-h,o-z)
      include 'variables.inc'
      dimension f(16),s(16,16)

      do 2 i = 1,nd
         k = (nel-1)*nd/2+i
         b(k) = b(k)+f(i)
         do 1 j = 1,nd
            l = (nel-1)*nd/2+j
            a(k,l) = a(k,l)+s(i,j)
1        continue
2     continue

      return
      end


      SUBROUTINE bandcmp(a,acmp,n,m1,m2,np,mp)
c
c     Given an n*n band diagonal matrix a with m1 subdiagonal and
c     m2 superdiagonal rows, compactly stored in an array
c     acmp(1:n,1:m1+m2+1) described below: the diagonal element
c     are in acmp(1:n,m1+1).Subdiagonal elements are in acmp(j:n,1:m1)
c     (with j>1 appropriate to the number of elements on each
c     subdiagonal).Superdiagonal elements arein acmp(1:j,m1+2:m1+m2+1)
c     with j<n appropriate to the number of elements on each
c     superdiagonal.
c
c     Subroutine called by SLVEQ15
c
      implicit double precision (a-h,o-z)
      integer m1,m2,mp,n,i,j
      dimension a(np,np),acmp(np,mp)
c
c     substitute diagonal elements into acmp(1:n,m1+1)
c
      do 1 i = 1,n
         acmp(i,m1+1) = a(i,i)
1     continue
c
c     substitute subdiagonal elements into acmp(j:n,1:m1)
c
      do 3 i = 1,m1
         do 2 j = i+1,n
```

```
            acmp(j,m1-i+1) = a(j,j-i)
    2       continue
    3     continue
c
c       substitute superdiagonal elements into acmp(1:j,m1+2:m1+m2+1)
c
        do 5 i = 1,m2
           do 4 j = 1,n-1
              acmp(j,m1+i+1) = a(j,j+i)
    4       continue
    5     continue

        return
        end



        SUBROUTINE bandec(a,n,m1,m2,np,mp,al,mpl,indx,d)
c
c       Given an n*n band diagonal matrix a with m1 subdiagonal rows
c       and m2 superdiagonal rows, compactly stored in the array
c       a(1:n,1:m1+m2+1) as described in the routine bandcmp. np, mp
c       and mpl are physical dimensions of a and al,with the requirement
c       of np>=n,mp>=m1+m2+1, and mpl>=m1. This routine
c       constructs an LU decomposition of a rowwise permutation in
c       al(1:n,1:m1). Indx(1:n) is an output vector which records the
c       row permutation effected by the partial pivoting; d is output
c       as +-1 depending on whether the number of row interchanges was
c       even or odd, respectively. This routine is used in combination
c       with BANDKS to solve band-diagonal sets of equations.
c
c       Subroutine called by SLVEQ15
c
        implicit double precision (a-h,o-z)
        integer m1,m2,mp,mpl,n,np,indx(np),i,j,k,l,mm
        dimension a(np,mp),al(np,mpl)
        parameter(tiny=1.0d-20)

        mm = m1+m2+1
        if (mm.gt.mp.or.m1.gt.mpl.or.n.gt.np) pause 'bad args in bandec'
        l = m1
c
c       rearrange the storage a bit
c
        do 3 i = 1,m1
           do 1 j = m1+2-i,mm
              a(i,j-l) = a(i,j)
    1   continue
           l = l-1
           do 2 j = mm-l,mm
              a(i,j) = 0.0d0
    2      continue
    3   continue

        d = 1.0d0
        l = m1
c
c       for each row ...
c
        do 6 k = 1,n
           dum = a(k,1)
           i = k
           if (l .lt. n) l = l+1
c
```

```
c       find the pivot element
c
           do 4 j = k+1,i
              if (abs(a(j,1)) .gt. abs(dum)) then
                 dum = a(j,1)
                 i = j
              endif
    4      continue
           indx(k) = i
           if (dum .eq. 0.1d-50) pause 'matrix is singular'
c
c       matrix is algorithmically singular, but proceed anyway with tiny
c       pivot or stop
c
c       interchange rows

           if (i .ne. k) then
              d = -d
              do 5 j = 1,mm
                 dum = a(k,j)
                 a(k,j) = a(i,j)
                 a(i,j) = dum
    5         continue
           endif
c
c       do the elimination
c
           do 7 i = k+1,l
              dum = a(i,1)/a(k,1)
              al(k,i-k) = dum
              do 6 j = 2,mm
                 a(i,j-1) = a(i,j)-dum*a(k,j)
    6         continue
              a(i,mm) = 0.0d0
    7      continue
    8   continue

        return
        end



        SUBROUTINE bandks(a,n,m1,m2,np,mp,al,mpl,indx,b)
c
c       Given arrays a, al, and indx as returned from bandec, and given
c       a right-hand side vector b(1:n),solves the band diagonal linear
c       equation a*x=b.The solution vector x overwrites b(1:n).The other
c       input arrays are not modified, and can be left in place for
c       sucessive calls with different right-hand sides.
c
        implicit double precision (a-h,o-z)
        integer m1,m2,mp,mpl,n,np,indx(np),i,k,l,mm
        dimension a(np,mp),al(np,mpl),b(np)

        mm = m1+m2+1
        if (mm.gt.mp.or.m1.gt.mpl.or.n.gt.np) pause 'bad args in bandks'
        l = m1
c
c       forward subsitition, unscrambling the permuted rows as we go.
c
        do 2 k = 1,n
           i = indx(k)
           if (i .ne. k) then
              dum  = b(k)
```

```
            b(k) = b(i)
            b(i) = dum
         endif
         if (i .lt. n) i = i+1
         do 1 i = k+1,1
            b(i) = b(i)-a(k,i-k)*b(k)
1        continue
2     continue
      i = 1
c
c     backsubstition
c
      do 4 i = n,1,-1
         dum = b(i)
         do 3 k = 2,1
            dum = dum-a(i,k)*b(k+i-1)
3        continue
         b(i) = dum/a(i,1)
         if (i .lt. mm) i = i+1
4     continue

      return
      end



      SUBROUTINE bondcns(nd,t)
c
c     Routine BONDCNS enforces all the boundary constraints to the
c     assembled global equation. The boundary constraints, including
c     spring and damping constant, initial forcas (static force and/or
c     moment), and concentrated mass, are translational constraints.
c
c     Translational constraints:
c
c     The tanslational constraints are considered as an arbitarily
c     oriented orthogonal spring and damping system. The orientation
c     is defined by the cosines of the primary, 1st, and 2nd (for 3-D
c     space), which are read in and stored in array refcos(1),
c     refcos(2), and refcos(3), respectively. Since the system is an
c     system, refcos(2), and/or refcos(3) are not necessary as an
c     input for 2-D and/or 3-D problem. In fact, refcos(3) is a cross
c     product of refcos(1) and refcos(2). Calculation of the refcos(2)
c     refcos(3) is completed in the routine INPUT.
c
c     Spring will be put in effective when the specified node moves
c     away from referernce position and the spring constant given
c     in some directions are less than 1.0e10 and larger than
c     zero. Otherwise, the node is either absolutely locked
c     for spring constant larger than or equal to 1.0e10, or
c     totally free for zero spring constant. It is important to
c     note that locking a node in some directions (value of spring
c     constants in these directions are larger than 1.0e+10) will
c     result in locking the node in other directions unless the
c     orthogonal spring system is parallel to the COORDINATE system.
c
c     Damping and initial force/moment are enforced when the input
c     values of them are not zero. Damping force is a linear function
c     of velocity. Initial force/moment are considered as static
c     external forces. Initial forces are put at the right-hand
c     side of the global equation.
c
c     Forced motion of the reference point in one direction is in
c     effective when the value of the spring constant in this
c     direction is not larger than 1.0e+10, the amplitudes of
c     motions (in-phase and/or out phase) in the direction is
c     not zero, and the jmotn is 2. If all the above mentioned
c     conditions are satisfied, the magnitude of the spring
c     constant is no longer important. Because the motion is an
c     enforced motion. The enforced motion is a sinuisodial motion
c     with the form as follows:
c
c                    a1*sin(freq*t)+a2*cos(freq*t)
c
c     where, a1 is in-phase amplitude and a2 is out-phase amplitude.
c
c     Variables   -
c        h        - = tstep, time step
c        idspg    - index referring to the spring coordinate and
c                   spring constants
c        jmotn    - index indicates types of the constrained nodes,
c                   = 1, no forced motion
c                   = 2, forced motion
c        k        - euqation # for which the translational
c                   constraints are applied
c        nd       - # of freedom of the element equation.
c        refvel   - reference velocity
c        dmpmat   - damping matrix,
c                   = Sum[dmpcont[p]*transmat[m,p]*transmat[n,p],
c                     [p,1,3]]
c        spgmat   - spring matrix,
c                   = Sum[spgcont[p]*transmat[m,p]*transmat[n,p],
c                     [p,1,3]]
c        transmat - coordinate transfer matrix
c
c     Subroutine called by SLVEQ16
c
      implicit double precision (a-h,o-z)
      include 'parametrics.inc'
      include 'variables.inc'
      dimension dmpmat(3,3),spgmat(3,3),transmat(3,3),idspg(3)
      parameter(pi=3.141592645)


      h = tstep
      do 15 i = 1,nconstr
c
c     transfer matrix transmat
c
         do 2 m = 1,ndim
            do 1 n = 1,ndim
               transmat(m,n) = refcos(i,n,m)
1           continue
            idspg(m) = 0
2        continue
c
c     spring matrix spgmat
c     damping matrix dmpmat
c
         do 5 m = 1,ndim
            do 4 n = 1,ndim
               spgmat(m,n) = 0.0d0
               dmpmat(m,n) = 0.0d0
               do 3 np = 1,ndim
                  spgmat(m,n) = spgmat(m,n)+spgcont(i,np)*
     1                          transmat(m,np)*transmat(n,np)
                  dmpmat(m,n) = dmpmat(m,n)+dmpcont(1,np)*
     1                          transmat(m,np)*transmat(n,np)
```

```
      3           continue
      4         continue
      5       continue

              do 7 m = 1,ndim
                 if (spgcont(i,m) .ge. 1.0d+10) then
                    do 6 n = m,ndim
                       if (dabs(transmat(m,n)) .gt. 1.0d-7) idspg(n) = 1
      6             continue
                 endif
      7       continue

              do 14 n = 1,ndim
c
c       find the equation where the constraints are applied
c
                 k = (nodec(i)-1)*nd/2+n
                 if (nodec(i) .eq. 1) then
                    ka = 1
                    kb = nd
                 else
                    ka = (nodec(i)-2)*nd/2+1
                    kb = ka+nd*3/2-1
                    if (nodec(i) .eq. node) kb = ka+nd-1
                 endif
c
c       translational constraints/motion, equation # k
c       k indicates the element and the diagonal element #
c       in array b and a (i.e. b(k), a(k,k).
c
c       *************************
c       Translational constraints:
c       *************************
c
c       i) Spring constant:
c       1) between zero and 1.0e10.
c          f[n] = spgmat[n,m](r[m]-refpos[m])
c
                 if (job .eq. 1) then
                    do 8 m = 1,ndim
                       b(k) =b(k)-spgmat(n,m)*(r(nodec(i),m)-refpos(i,m))
                       a(k,k+m-n) = a(k,k+m-n)+spgmat(n,m)
      8             continue
                 else
                    do 9 m = 1,ndim
                       b(k) =b(k)-spgmat(n,m)*(r(nodec(i),m)-refpos(i,m))
                       a(k,k+m-n) = a(k,k+m-n)+h*h/4*spgmat(n,m)
      9             continue
                 endif
c
c       2) larger than 1.0e10. The node is locked.
c
                 if (idspg(n) .eq. 1) then
                    do 10 kcol = ka,kb
                       if (kcol .ne. k) then
                          a(k,kcol) = 0.0d0
                          a(kcol,k) = 0.0d0
                       endif
     10             continue
                    a(k,k) = 1.0d0
                    b(k) = 0.0d0
                 else
                    if (job .eq. 2) then
c
c       ii) Damping constant:
c       1) translational constraints
c
                       do 11 m = 1,ndim
                          b(k) = b(k)-dmpmat(n,m)*rdot(nodec(i),m)
                          a(k,k+m-n) = a(k,k+m-n)+h/2*dmpmat(n,m)
     11                continue
c
c       iii) Forced translational motion of reference point(s)
c            at node(s)
c
                       if ((disamp(i,n,1) .ne. 0.0d0) .or.
     1                     (disamp(i,n,2) .ne. 0.0d0) .or.
     2                     (jmotn         .eq. 2    )) then
                          do 12 kcol = ka,kb
                             if (kcol .ne. k) then
                                a(k,kcol) = 0.0d0
                                a(kcol,k) = 0.0d0
                             endif
     12                   continue
                          a(k,k) = 1.0d0
                          b(k) = 0.0d0
c
c       transition stage:
c       y=3xi^2-2xi^3,xi=t/ttrans
c       ttrans=6.5*pi/freq
c
                          ttrans = 6.5*pi/freq(i)
                          if (jmotn .eq. 2) ttrans = 10*pi
                          if (t .le. ttrans) then
                             xi = t/ttrans
                             if (jmotn .eq. 1) then
c
c       in-phase: a1*sin(freq*t)
c       out-phase: a2*cos(freq*t)
c       a1=disamp(i,j,1), a2=disamp(i,j,2)
c
                                refvel1 = freq(i)*dcos(freq(i)*t)*xi*xi*
     1                                    (3-2*xi)+dsin(freq(i)*t)/ttrans
     2                                    *6.0*xi*(1-xi)
                                refvel2 =-freq(i)*dsin(freq(i)*t)*xi*xi*
     1                                    (3-2*xi)+dcos(freq(i)*t)/ttrans
     2                                     *6.0*xi*(1-xi)
                                refvel = disamp(i,n,1)*refvel1+
     1                                   disamp(i,n,2)*refvel2
                             else
c
c       forced motion has the transistion period
c           10 x^3 - 15 x^4 + 6 x^5
c
                                read(ifmotn,*) refvel
                                refvel = refvel*(10*xi**3-15*xi**4+6*xi**5)
                             endif
                          else
                             if (jmotn .eq. 1) then
c
c       sinuisoidal motion:
c
                                refvel = freq(i)*
     1                                 ( disamp(i,n,1)*dcos(freq(i)*t)
     2                                  -disamp(i,n,2)*dsin(freq(i)*t))
                             else
c
c       arbitrary forced motion
```

```
c       The forced motion is read in from the file named 'forcedmotion'.
c       In this unformated file, the reference velocity of the top end
c       (motions of other nodes are also allowed, but users are require
c       to develop their own scheme) is arranged in one column with one
c       set of the velocity, (refvel(x), refvel(y), refvel(z)),
c       consisting of three rows or less, depending on the constrainted
c       conditions.
c
                            read(ifmotn,*) refvel
                          endif
                        endif
                        b(k) = (refvel-rdot(nodec(i),n))/h
                      endif
                    endif
                  endif
c
c       iv) Initial force/moment(s):
c
            if (finit1(i,n) .ne. 0.0d0) b(k) = b(k)+finit1(i,n)
c
c       v. concentrated mass
c
            if ((cmass(i) .gt. 0.0d0) .and. (j.eq.2))
     1        b(k) = b(k)-cmass(i)*g

   14     continue
   15   continue

        return
        end


        block data cshpfn
c
c       All the constant coefficients of shape functions, including
c       alpha, beta, eta, gamma, mu, tau, and omega are initialed
c       in this data block.
c
c       Shape function:
c
c          a[1]=1-3xi^2+2xi^3
c          a[2]=xi-2xi^2+xi^3
c          a[3]=3xi^2-2xi^3
c          a[4]=-xi^2+xi^3
c          p[1]=1-3xi+2xi^2
c          p[2]=4xi(1-xi)
c          p[3]=xi(2xi-1)
c          where  0<= xi <= 1
c
        implicit double precision (a-h,o-z)
        include 'parametrics.inc'
c
c       gamma = integrate[a[i]*a[k]*p[m], (xi,0,1)]
c
        data ((gamma(i,k,1),k=1,4),i=1,4)
     1  / 1.7460317460317459d-01,          1.5873015873015872d-02,
     2    8.7301587301587304d-03,         -3.1746031746031746d-03,
     3    1.5873015873015872d-02,          1.9841269841269840d-03,
     4    7.9365079365079365d-04,         -3.9682539682539683d-04,
     5    8.7301587301587304d-03,          7.9365079365079365d-04,
     6   -2.5396825396825397d-02,          3.1746031746031746d-03,
     7   -3.1746031746031746d-03,         -3.9682539682539683d-04,
     8    3.1746031746031746d-03,         -3.9682539682539683d-04/
        data ((gamma(i,k,2),k=1,4),i=1,4)
     1  / 2.2222222222222221d-01,          3.9682539682539680d-02,
     2    1.1111111111111110d-01,         -2.6984126984126985d-02,
     3    3.9682539682539680d-02,          7.9365079365079361d-03,
     4    2.6984126984126985d-02,         -6.3492063492063492d-03,
     5    1.1111111111111110d-01,          2.6984126984126985d-02,
     6    2.2222222222222221d-01,         -3.9682539682539680d-02,
     7   -2.6984126984126985d-02,         -6.3492063492063492d-03,
     8   -3.9682539682539680d-02,          7.9365079365079361d-03/
        data ((gamma(i,k,3),k=1,4),i=1,4)
     1 / -2.5396825396825397d-02,         -3.1746031746031746d-03,
     2    8.7301587301587304d-03,         -7.9365079365079365d-04,
     3   -3.1746031746031746d-03,         -3.9682539682539683d-04,
     4    3.1746031746031746d-03,         -3.9682539682539683d-04,
     5    8.7301587301587304d-03,          3.1746031746031746d-03,
     6    1.7460317460317459d-01,         -1.5873015873015872d-02,
     7   -7.9365079365079365d-04,         -3.9682539682539683d-04,
     8   -1.5873015873015872d-02,          1.9841269841269840d-03/
c
c       beta= integrate[a'[i]*a'[k]*p[m], (xi,0,1)]
c
        data ((beta(i,k,1),k=1,4),i=1,4)
     1  / 8.5714285714285715d-02,         -5.7142857142857141d-02,
     2   -8.5714285714285715d-02,          4.2857142857142858d-02,
     3   -5.7142857142857141d-02,          7.1428571428571425d-02,
     4    5.7142857142857141d-02,         -1.1904761904761904d-02,
     5   -8.5714285714285715d-02,          5.7142857142857141d-02,
     6    8.5714285714285715d-02,         -4.2857142857142858d-02,
     7    4.2857142857142858d-02,         -1.1904761904761904d-02,
     8   -4.2857142857142858d-02,          4.7619047619047623d-03/
        data ((beta(i,k,2),k=1,4),i=1,4)
     1  / 1.0285714285714285d+00,          1.1428571428571428d-01,
     2   -1.0285714285714285d+00,          1.1428571428571428d-01,
     3    1.1428571428571428d-01,          5.7142857142857141d-02,
     4   -1.1428571428571428d-01,         -9.5238095238095247d-03,
     5   -1.0285714285714285d+00,         -1.1428571428571428d-01,
     6    1.0285714285714285d+00,         -1.1428571428571428d-01,
     7    1.1428571428571428d-01,         -9.5238095238095247d-03,
     8   -1.1428571428571428d-01,          5.7142857142857141d-02/
        data ((beta(i,k,3),k=1,4),i=1,4)
     1  / 8.5714285714285715d-02,          4.2857142857142858d-02,
     2   -8.5714285714285715d-02,         -5.7142857142857141d-02,
     3    4.2857142857142858d-02,          4.7619047619047623d-03,
     4   -4.2857142857142858d-02,         -1.1904761904761904d-02,
     5   -8.5714285714285715d-02,         -4.2857142857142858d-02,
     6    8.5714285714285715d-02,          5.7142857142857141d-02,
     7   -5.7142857142857141d-02,         -1.1904761904761904d-02,
     8    5.7142857142857141d-02,          7.1428571428571425d-02/
c
c       alpha= integrate[a''[i]*a''[k]*p[m], (xi,0,1)]
c
        data ((alpha(i,k,1),k=1,4),i=1,4)
     1  / 3.6000000000000001d+00,          2.7999999999999998d+00,
     2   -3.6000000000000001d+00,          8.0000000000000004d-01,
     3    2.7999999999999998d+00,          2.0666666666666669d+00,
     4   -2.7999999999999998d+00,          7.3333333333333328d-01,
     5   -3.6000000000000001d+00,         -2.7999999999999998d+00,
     6    3.6000000000000001d+00,         -8.0000000000000004d-01,
     7    8.0000000000000004d-01,          7.3333333333333328d-01,
     8   -8.0000000000000004d-01,          6.6666666666666666d-02/
        data ((alpha(i,k,2),k=1,4),i=1,4)
     1  / 4.7999999999999998d+00,          2.3999999999999999d+00,
     2   -4.7999999999999998d+00,          2.3999999999999999d+00,
```

```
     3    2.3999999999999999d+00,          1.8666666666666667d+00,
     4   -2.3999999999999999d+00,          5.3333333333333333d-01,
     5   -4.7999999999999998d+00,         -2.3999999999999999d+00,
     6    4.7999999999999998d+00,         -2.3999999999999999d+00,
     7    2.3999999999999999d+00,          5.3333333333333333d-01,
     8   -2.3999999999999999d+00,          1.8666666666666667d+00/
      data ((alpha(i,k,3),k=1,4),i=1,4)
     1  / 3.6000000000000001d+00,          8.0000000000000004d-01,
     2   -3.6000000000000001d+00,          2.7999999999999998d+00,
     3    8.0000000000000004d-01,          6.6666666666666666d-02,
     4   -8.0000000000000004d-01,          7.3333333333333328d-01,
     5   -3.6000000000000001d+00,         -8.0000000000000004d-01,
     6    3.6000000000000001d+00,         -2.7999999999999998d+00,
     7    2.7999999999999998d+00,          7.3333333333333328d-01,
     8   -2.7999999999999998d+00,          2.0666666666666669d+00/
c
c     real mu = integrate(a[i]*p[m], (xi,0,1))
c
      data ((mu(i,m),i=1,4),m=1,3)
     1  / 1.8333333333333332d-01,          1.6666666666666666d-02,
     2   -1.6666666666666666d-02,          0.0000000000000000d+00,
     3    3.3333333333333331d-01,          6.6666666666666666d-02,
     4    3.3333333333333331d-01,         -6.6666666666666666d-02,
     5   -1.6666666666666666d-02,          0.0000000000000000d+00,
     6    1.8333333333333332d-01,         -1.6666666666666666d-02/
c
c     tau = integrate(p[m], (xi,0,1))
c
      data (tau(m),m=1,3)
     1  / 1.6666666666666666d-01,          6.6666666666666663d-01,
     2    1.6666666666666666d-01/
c                     --
c     eta = integrate(p[l]*p[m], (xi,0,1))
c
      data ((eta(l,m),l=1,3),m=1,3)
     1  / 1.3333333333333333d-01,          6.6666666666666666d-02,
     2   -3.3333333333333333d-02,          6.6666666666666666d-02,
     3    5.3333333333333333d-01,          6.6666666666666666d-02,
     4   -3.3333333333333333d-02,          6.6666666666666666d-02,
     5    1.3333333333333333d-01/
c
c     omega = [a'[i]a'[k], (xi=0, 0.5, 1)]
c
      data ((omega(i,k,1),k=1,4),i=1,4) / 5*0.0d0,1.0d0,10*0.0d0/
      data ((omega(i,k,3),k=1,4),i=1,4) /15*0.0d0,1.0d0/
      data ((omega(i,k,2),k=1,4),i=1,4)
     1       / 2.25d+00, 3.75d-01,-2.25d+00, 3.75d-01,
     2         3.75d-01, 6.25d-02,-3.75d-01, 6.25d-02,
     3        -2.25d+00,-3.75d-01, 2.25d+00,-3.75d-01,
     4         3.75d-01, 6.25d-02,-3.75d-01, 6.25d-02/

      end




      double precision function dmass(i,j,m,k,n,nel)
c
c     Routine RJ evaluates the derivative of mass matrix M(m)
c     with respect to vector u(kn). RJ(ijmkn) is an element of a
c     matrix of 3 by 3 (or 2 by 2 in 2-D problem). The location of
c     RJ in the matrix is specified by two indices i and j.
c
c     dmass=j(qsmkn)=dM(qsm)/dU(kn),
c     where:
c        dmass=-ruf*Af(m)*Cm*|2*omega(lkm)u05(li)  n=i=j
c                            |omega(lkm)u05(lj)        n=i,n<>j
c                            |omega(lkm)u05(li)    n=j,n<>i
c                            |0                          n<>i,n<>j
c     Variables   -
c        RJ       -  =dM(ijm)/dU(kn)
c        i,j      -  indices used to define location of the element
c        m,k,n    -  indices of M(m) and U(kn), respectively
c        nel      -  element #
c        u05      -  u at time t=n+0.5
c
c     Subroutine called by DYEQ16
c
      implicit double precision (a-h,o-z)
      include 'parametrics.inc'
      include 'variables.inc'
      parameter(pi=3.141592654d0)

      dmass = 0.0d0

      if ((n .eq. i) .and. (n .eq. j)) then
         do 1 l = 1,4
            dmass = dmass+
     1              2*omega(l,k,m)*u05(l,i)/lnthel(nel)/lnthel(nel)
     1    continue
      endif

      if ((n .eq. i) .and. (n .ne. j)) then
         do 2 l = 1,4
            dmass = dmass+
     1              omega(l,k,m)*u05(l,j)/lnthel(nel)/lnthel(nel)
     2    continue
      endif

      if ((n .ne. i) .and. (n .eq. j)) then
         do 3 l = 1,4
            dmass = dmass+
     1              omega(l,k,m)*u05(l,i)/lnthel(nel)/lnthel(nel)
     3    continue
      endif

      dmass = -ruf*0.25*pi*del(jeltyp(nel),m)*del(jeltyp(nel),m)*cm*
     1        dmass

      return
      end




      SUBROUTINE dyeq16(s,f,nel,xa1,xb1,xa2,xb2)
c
c     ****************
c     dynamic analysis
c     ****************
c
c     Vectorial motion equations for cable dynamics are
c     decomposed into 16 scalar equations with the form S*dUdot=F.
c     Routine DYEQ16 forms the element matrix S and the right-hand
c     side vector F, where S is a 16 by 16 (or 12 by 12) coefficient
c     matrix, and F is a 16 by 1 (or 12 by 1) vector in 3-D (or 2-D)
c     problem.
```

```
c
c       Variables   -
c          at       -  cross-section area of the tube(element),
c                      at=afel-ael
c          dUdot    -  unknown vector arranged as follows:
c
c                      3 dimensions  -            2 dimensions -
c           (dUdot) =  |  dUdot[11] |  (dUdot) =  |  dUdot[11] |
c                      |  dUdot[12] |             |  dUdot[12] |
c                      |  dUdot[13] |             |  dUdot[21] |
c                      |  dUdot[21] |             |  dUdot[22] |
c                      |  dUdot[22] |             | dlambda[1] |
c                      |  dUdot[23] |             | dlambda[2] |
c                      | dlambda[1] |             |  dUdot[31] |
c                      | dlambda[2] |             |  dUdot[32] |
c                      |  dUdot[31] |             |  dUdot[41] |
c                      |  dUdot[32] |             | dlambda[3] |
c                      |  dUdot[33] |             | dlambda[1] |
c                      |  dUdot[41] |
c                      |  dUdot[42] |
c                      |  dUdot[43] |
c                      | dlambda[3] |
c                      | dlambda[1] |
c          et       -  Young's modulus of the tube (element),
c                      et = eel(jeltyp(nel))
c          f        -  element right-hand side vector
c          h        -  time step (=tstep)
c          lt       -  length of the tube(element), lt = lnthel(nel)
c          massmat  -  mass matrix
c          mu1,mu2,gamma1,gamma2  -
c                      shape coefficient mu and gamma in the interval
c                      [xa1,xb1], and [xa2,xb2], respectively.
c          nel      -  element #
c          s        -  element stifness matrix
c          wbd      -  (net weight of the element in water)
c                      ------------------------------------, wbd(m)
c                             (dist to out most fiber)
c          [xa1,xb1], [xa2,xb2]  -
c                      range of the 1st and 2nd under bottom portions of
c                      the element
c
c       Subroutine called by SLVEQ16
c
        implicit double precision (a-h,o-z)
        include 'parametrics.inc'
        include 'variables.inc'
        dimension at(3),f(16),s(16,16)
        dimension gamma1(4,4,3),gamma2(4,4,3)
        double precision mu1(4,3),mu2(4,3),lt

        h = tstep
        fact = 1-damp/h
        do 2 ii = 1,16
           do 1 nn = 1,16
              s(ii,nn) = 0.0d0
    1      continue
           f(ii) = 0.0d0
    2   continue
c
c       at=af-ai
c       et=eel(jeltyp(nel)
c       lt=lnthel(nel)
c
        do 3 m = 1,3
           at(m) = afel(jeltyp(nel),m)-ael(jeltyp(nel),m)
    3   continue
        et = eel(jeltyp(nel))
        lt = lnthel(nel)
c
c       compute mu and gamma in the interval defined by [xa1,xb1],
c       and [xa2,xb2]
c
        call mubtm(xa1,xb1,mu1)
        call mubtm(xa2,xb2,mu2)
        call gammabtm(xa1,xb1,gamma1)
        call gammabtm(xa2,xb2,gamma2)
c
c ******************************
c       stiffness matrix s
c ******************************
c
        do 13 i = 1,4
           do 12 n = 1,ndim
              ii = ndim*(i-1)+n
              if (i .ge. 3) ii = ii+2
              do 9 k = 1,4
c
c       -h^2/4*(11)k(ik)|(t=n+0.5)
c       (11)k(ik)=-alpha(ikm)b(m)-beta(ikm)lmda05(m)
c       (11)k(ik)=-gamma(ikm)*wbd(m)  (bottom support: in y direction)
c
                 nn = ndim*(k-1)+n
                 if (k .ge. 3) nn = nn+2
                 do 4 m = 1,3
                    s(ii,nn) = s(ii,nn)+h*h/4.0*
     1                       (alpha(i,k,m)/lt**3*et*moi(jeltyp(nel),m)
     2                        +beta(i,k,m)/lt*lmda05(m))
                    if (n.eq.2)
     1                 s(ii,nn) = s(ii,nn)+h*h/4.0*
     2                          (gamma1(i,k,m)+gamma2(i,k,m))*lt*wbd(m)
    4               continue
c
c       gamma(ikm)*[m(m)|(t=n)+
c                  dmass(m,ks,ns)|(t=n+0.5)*udot(ks,ns)*h/2]
c       m[3*3]=m(is,js,m)
c       dmass[3*3]=j(is,js,m,ks,ns)
c       is,js,ks,ns corresponding to i,j,k,n in routine RJ
c
                 is = n
                 do 8 js = 1,ndim
                    nn = ndim*(k-1)+js
                    if (k .ge. 3) nn = nn+2
                    do 7 m = 1,3
                       sumdmass = 0.0d0
                       do 6 ks = 1,4
                          do 5 ns = 1,ndim
                             sumdmass =sumdmass+
     1                                dmass(is,js,m,ks,ns,nel)*udot(ks,ns)
    5                     continue
    6                  continue
                       s(ii,nn) = s(ii,nn)+gamma(i,k,m)*lt*
     1                             (massmat(is,js,m)+sumdmass*h/2)
    7               continue
    8            continue
    9         continue
c
c       -h^2/4(12)k(im)|(t=n+0.5)
c       (12)k(im)|(t=n+0.5)=-beta(ikm)*u05(kn)
```

```
c
            do 11 m = 1,3
               nn = 2*ndim+m
               if (m .eq. 3) nn = nn+2*ndim
               do 10 k = 1,4
                  s(ii,nn)=s(ii,nn)+h*h/4*beta(i,k,m)/lt*u05(k,n)
   10          continue
   11       continue

   12     continue
   13  continue
c
c      -h^2/4*(21)k(jk)|(t=n+0.5)
c      (21)k(jk)|(t=n+0.5)
c      =(21)k(jkn)=-beta(ikj)*u05(i,n)
c
       do 18 j=1,3
          ii = 2*ndim+j
          if (j .eq. 3) ii = ii+2*ndim
          do 16 k = 1,4
             do 15 n = 1,ndim
                nn = ndim*(k-1)+n
                if (k .ge. 3) nn = nn+2
                do 14 i = 1,4
                 s(ii,nn) = s(ii,nn)+h*h/4*beta(i,k,j)/lt*u05(i,n)
   14           continue
   15        continue
   16     continue
c
c      -h^2/4(22)k(jm)|(t=n+0.5)
c      (22)k(jm)|(t=n+0.5)=eta(mj)/[af(m)-ai(m)]e
c                          ~
          do 17 m = 1,3
             nn = 2*ndim+m
             if (m .eq. 3) nn = nn+2*ndim
             s(ii,nn) = s(ii,nn)-fact*h*h/4*eta(m,j)*lt/(at(m)*et)
   17     continue
   18  continue
c
c ***********************************
c      right-hand side vector f
c ***********************************
c
       do 22 i = 1,4
          do 21 n = 1,ndim
             ii = ndim*(i-1)+n
             if (i .ge. 3) ii = ii+2
             do 20 m = 1,3
c
c      q(mn)*mu(im)
c      mu(im)*wbd(m)*(D(m)+btmdpth)  (bottom support: in y direction)
c
                f(ii) = f(ii)+mu(i,m)*lt*q(m,n)
                if (n .eq. 2)
     1             f(ii) = f(ii)+(mu1(i,m)+mu2(i,m))*lt*wbd(m)*
     2                     (dfel(jeltyp(nel),m)+btmdpth)

                do 19 k = 1,4
c
c      -alpha(ikm)*b(m)*u(kn)
c      -beta(ikm)*lmda(m)*u(kn)
c      -gamma(ikm)*wbd(m)*u(k2)  (bottom support: in y direction)
c
                   f(ii) = f(ii)-u(k,n)*
     1                     (alpha(i,k,m)/lt**3*et*moi(jeltyp(nel),m)
     2                      +beta(i,k,m)/lt*lmda(m))
                if (n .eq. 2)
     1             f(ii) = f(ii)-(gamma1(i,k,m)+gamma2(i,k,m))*lt*
     2                     wbd(m)*u(k,2)
c
c      h/2(11)k(ik)|(t=n+0.5)*udot(kn)|(t=n)
c      =-h/2*[alpha(ikm)b(m)+beta(ikm)lmda(m)]*udot(kn)|(t=n)
c      -h/2*gamma(ikm)*wbd(m)*udot(k2)|(t=n)  (bottom support)
c
                f(ii) = f(ii)-h/2*udot(k,n)*
     1                     (alpha(i,k,m)/lt**3*et*moi(jeltyp(nel),m)
     2                      +beta(i,k,m)/lt*lmda05(m))
                if (n .eq. 2)
     1             f(ii)=f(ii)-h/2*(gamma1(i,k,m)+gamma2(i,k,m))*lt*
     2                     wbd(m)*udot(k,2)
   19          continue
   20        continue
   21     continue
   22  continue


       do 30 j = 1,3
          ii = 2*ndim+j
          if (j .eq. 3) ii = ii+2*ndim
c
c      d(j)/2|(t=n)=[tau(j)+2eta(mj)e(m)-beta(ikj)u(in)u(kn)|(t=n)]/4
c
          f(ii) = f(ii)+0.25*tau(j)*lt
          do 23 m = 1,3
             f(ii) = f(ii)+0.5*eta(m,j)*e(m)*lt
   23     continue
          do 26 i = 1,4
             do 25 k = 1,4
                do 24 n = 1,ndim
                  f(ii) = f(ii)-0.25*beta(i,k,j)/lt*u(i,n)*u(k,n)
   24           continue
   25        continue
   26     continue
c
c      h/2*(21)k(jkn)|(t=n+0.5)*udot(kn)|(t=n)
c      =-h/2*beta(ikj)u(in)|(t=n+0.5)*udot(kn)|(t=n)
c
          do 29 n = 1,ndim
             do 28 k = 1,4
                do 27 i = 1,4
                   f(ii)=f(ii)-0.5*h*beta(i,k,j)/lt*u05(i,n)*udot(k,n)
   27           continue
   28        continue
   29     continue

   30  continue

       do 32 ii = 1,4*ndim+4
          do 31 nn = 1,4*ndim+4
             if ((nn .ge. ndim+1)   .and. (nn .le. 2*ndim) .or.
     1           (nn .ge. 3*ndim+3) .and. (nn .le. 4*ndim+2))
     2        s(ii,nn) = s(ii,nn)*lt
   31     continue
   32  continue


       return
       end
```

```
      SUBROUTINE epsilon(nel,pf,pi)
c
c     Routine EPSILON evaluates the stretch epsilon e
c
c     Variables   -
c        e        -  stretch,            lmda-Pf*Af+Pi*Ai    |
c                           e[m] = ( ------------------- )|
c                                        E(Af-Ai)          |m
c        nel      -  element #
c        pf       -  pressure on the outer cross-section of the tube
c                    at location xi = 0, 0.5, & 1, pf[m]
c        pi       -  pressure on inner cross-sectin area of the tube
c                    at location xi=0, 0.5 & 1, pi[m]
c
c     Subroutine called by SLVEQ16
c
      implicit double precision (a-h,o-z)
      include 'parametrics.inc'
      include 'variables.inc'
      dimension pf(3),pi(3)

      do 1 m = 1,3
         e(m) = (lmda(m)-pf(m)*afel(jeltyp(nel),m)+
     1                  pi(m)*aiel(jeltyp(nel),m))/
     2           (eel(jeltyp(nel))*
     3            (afel(jeltyp(nel),m)-aiel(jeltyp(nel),m)))
    1 continue

      return
      end




      SUBROUTINE gammabtm(a,b,gammab)
c
c     Routine GAMMABTM calculats coefficients gamma, named gammab,
c     in the interval (a,b), where 0 <= a < b <= 1.
c     gamma=integrate(a[i]*a[k]*p[m],(xi,a,b))
c
c     Global Variables -
c        a,b    -  interval bounds
c        gammab-  coefficients defined by the integral, gammab[4,4,3].
c
c     Subroutine called by STEQ15, DYEQ15
c
      double precision a,b,gammab(4,4,3)

      gammab(1,1,1) =
     1    (-1d0+(3d0/2+(4d0/3+(-11d0/2+(3d0+(31d0/6+(-58d0/7+(9d0/2
     2     -8d0/9*a)*a)*a)*a)*a)*a)*a)*a)*a-
     3    (-1d0+(3d0/2+(4d0/3+(-11d0/2+(3d0+(31d0/6+(-58d0/7+(9d0/2
     4     -8d0/9*b)*b)*b)*b)*b)*b)*b)*b)*b

      gammab(1,1,2) = (-2d0+(4d0/3+(6d0+(-8d0+(-10d0/3+(12d0+
     1                 (-8d0+16d0/9*a)*a)*a)*a)*a)*a)*a)*a*a-
     2                 (-2d0+(4d0/3+(6d0+(-8d0+(-10d0/3+(12d0+
     3                 (-8d0+16d0/9*b)*b)*b)*b)*b)*b)*b)*b*b

      gammab(1,1,3) = (1d0/2+(-2d0/3+(-3d0/2+(16d0/5+(1d0/6+
     1                 (-30d0/7+(7d0/2-8d0/9*a)*a)*a)*a)*a)*a)*a)*a*a-
     2                 (1d0/2+(-2d0/3+(-3d0/2+(16d0/5+(1d0/6+
     3                 (-30d0/7+(7d0/2-8d0/9*b)*b)*b)*b)*b)*b)*b)*b*b

      gammab(1,2,1) = (-1d0/2+(5d0/3+(-3d0/2+(-2d0+(35d0/6+
     1                 (-39d0/7+(5d0/2-4d0/9*a)*a)*a)*a)*a)*a)*a)*a*a-
     2                 (-1d0/2+(5d0/3+(-3d0/2+(-2d0+(35d0/6+
     3                 (-39d0/7+(5d0/2-4d0/9*b)*b)*b)*b)*b)*b)*b)*b*b

      gammab(1,2,2) = (-4d0/3+(3d0+((-20d0/3+(60d0/7+(-9d0/2+8d0/9
     1                 *a)*a)*a)*a)*a)*a)*a*a*a-
     2                 (-4d0/3+(3d0+((-20d0/3+(60d0/7+(-9d0/2+8d0/9
     3                 *b)*b)*b)*b)*b)*b)*b*b*b

      gammab(1,2,3) = (1d0/3+(-1d0+(2d0/5+(2d0+(-23d0/7+(2d0-4d0/9
     1                 *a)*a)*a)*a)*a)*a)*a*a*a-
     2                 (1d0/3+(-1d0+(2d0/5+(2d0+(-23d0/7+(2d0-4d0/9
     3                 *b)*b)*b)*b)*b)*b)*b*b*b

      gammab(1,3,1) = (-1d0+(11d0/4+(-3d0/5+(-35d0/6+(58d0/7+
     1                 (-9d0/2+8d0/9*a)*a)*a)*a)*a)*a)*a*a*a-
     2                 (-1d0+(11d0/4+(-3d0/5+(-35d0/6+(58d0/7+
     3                 (-9d0/2+8d0/9*b)*b)*b)*b)*b)*b)*b*b*b

      gammab(1,3,2) =
     1    (-3d0+(4d0+(14d0/3+(-12d0+(8d0-16d0/9*a)*a)*a)*a)*a)*a**4-
     2    (-3d0+(4d0+(14d0/3+(-12d0+(8d0-16d0/9*b)*b)*b)*b)*b)*b**4

      gammab(1,3,3) = (3d0/4+(-8d0/5+(-5d0/6+(30d0/7+(-7d0/2+8d0/9
     1                 *a)*a)*a)*a)*a)*a**4-
     2                 (3d0/4+(-8d0/5+(-5d0/6+(30d0/7+(-7d0/2+8d0/9
     3                 *b)*b)*b)*b)*b)*b**4

      gammab(1,4,1) = (1d0/3+(-1d0+(2d0/5+(2d0+(-23d0/7+(2d0-4d0/9
     1                 *a)*a)*a)*a)*a)*a)*a*a*a-
     2                 (1d0/3+(-1d0+(2d0/5+(2d0+(-23d0/7+(2d0-4d0/9
     3                 *b)*b)*b)*b)*b)*b)*b*b*b

      gammab(1,4,2) = (1d0+(-8d0/5+(-4d0/3+(32d0/7+(-7d0/2+8d0/9
     1                 *a)*a)*a)*a)*a)*a**4-
     2                 (1d0+(-8d0/5+(-4d0/3+(32d0/7+(-7d0/2+8d0/9
     3                 *b)*b)*b)*b)*b)*b**4

      gammab(1,4,3) = (-1d0/4+(3d0/5+(1d0/6+(-11d0/7+(3d0/2-4d0/9
     1                 *a)*a)*a)*a)*a)*a**4-
     2                 (-1d0/4+(3d0/5+(1d0/6+(-11d0/7+(3d0/2-4d0/9
     3                 *b)*b)*b)*b)*b)*b**4

      gammab(2,2,1) = (-1d0/3+(7d0/4+(-4d0+(5d0+(-25d0/7+(11d0/8-2d0/9
     1                 *a)*a)*a)*a)*a)*a)*a*a*a-
     2                 (-1d0/3+(7d0/4+(-4d0+(5d0+(-25d0/7+(11d0/8-2d0/9
     3                 *b)*b)*b)*b)*b)*b)*b*b*b

      gammab(2,2,2) = (-1d0+(4d0+(-20d0/3+(40d0/7+(-5d0/2+4d0/9
     1                 *a)*a)*a)*a)*a)*a**4-
     2                 (-1d0+(4d0+(-20d0/3+(40d0/7+(-5d0/2+4d0/9
     3                 *b)*b)*b)*b)*b)*b**4

      gammab(2,2,3) = (1d0/4+(-6d0/5+(7d0/3+(-16d0/7+(9d0/8-2d0/9
     1                 *a)*a)*a)*a)*a)*a**4-
     2                 (1d0/4+(-6d0/5+(7d0/3+(-16d0/7+(9d0/8-2d0/9
     3                 *b)*b)*b)*b)*b)*b**4

      gammab(2,3,1) = (-3d0/4+(17d0/5+(-37d0/6+(39d0/7+(-5d0/2+4d0/9
```

```
     1                    *a)*a)*a)*a)*a)*a**4-
     2                    (-3d0/4+(17d0/5+(-37d0/6+(39d0/7+(-5d0/2+4d0/9
     3                    *b)*b)*b)*b)*b)*b**4

        gammab(2,3,2) =
     1      (-12d0/5+(22d0/3+(-60d0/7+(9d0/2-8d0/9*a)*a)*a)*a)*a**5-
     2      (-12d0/5+(22d0/3+(-60d0/7+(9d0/2-8d0/9*b)*b)*b)*b)*b**5

        gammab(2,3,3) =
     1      (3d0/5+(-7d0/3+(23d0/7+(-2d0+4d0/9*a)*a)*a)*a)*a**5-
     2      (3d0/5+(-7d0/3+(23d0/7+(-2d0+4d0/9*b)*b)*b)*b)*b**5

        gammab(2,4,1) = (1d0/4+(-6d0/5+(7d0/3+(-16d0/7+(9d0/8-2d0/9
     1                  *a)*a)*a)*a)*a)*a**4-
     2                  (1d0/4+(-6d0/5+(7d0/3+(-16d0/7+(9d0/8-2d0/9
     3                  *b)*b)*b)*b)*b)*b**4

        gammab(2,4,2) =
     1      (4d0/5+(-8d0/3+(24d0/7+(-2d0+4d0/9*a)*a)*a)*a)*a**5-
     2      (4d0/5+(-8d0/3+(24d0/7+(-2d0+4d0/9*b)*b)*b)*b)*b**5

        gammab(2,4,3) =
     1      (-1d0/5+(5d0/6+(-9d0/7+(7d0/8-2d0/9*a)*a)*a)*a)*a**5-
     2      (-1d0/5+(5d0/6+(-9d0/7+(7d0/8-2d0/9*b)*b)*b)*b)*b**5

        gammab(3,3,1) =
     1      (-9d0/5+(13d0/2+(-58d0/7+(9d0/2-8d0/9*a)*a)*a)*a)*a**5-
     2      (-9d0/5+(13d0/2+(-58d0/7+(9d0/2-8d0/9*b)*b)*b)*b)*b**5

        gammab(3,3,2) = (-6d0+(12d0+(-8d0+16d0/9*a)*a)*a)*a**6-
     1                  (-6d0+(12d0+(-8d0+16d0/9*b)*b)*b)*b**6

        gammab(3,3,3) = (3d0/2+(-30d0/7+(7d0/2-8d0/9*a)*a)*a)*a**6-
     1                  (3d0/2+(-30d0/7+(7d0/2-8d0/9*b)*b)*b)*b**6

        gammab(3,4,1) =
     1      (3d0/5+(-7d0/3+(23d0/7+(-2d0+4d0/9*a)*a)*a)*a)*a**5-
     2      (3d0/5+(-7d0/3+(23d0/7+(-2d0+4d0/9*b)*b)*b)*b)*b**5

        gammab(3,4,2) = (2d0+(-32d0/7+(7d0/2-8d0/9*a)*a)*a)*a**6-
     1                  (2d0+(-32d0/7+(7d0/2-8d0/9*b)*b)*b)*b**6

        gammab(3,4,3) = (-1d0/2+(11d0/7+(-3d0/2+4d0/9*a)*a)*a)*a**6-
     1                  (-1d0/2+(11d0/7+(-3d0/2+4d0/9*b)*b)*b)*b**6

        gammab(4,4,1) =
     1      (-1d0/5+(5d0/6+(-9d0/7+(7d0/8-2d0/9*a)*a)*a)*a)*a**5-
     2      (-1d0/5+(5d0/6+(-9d0/7+(7d0/8-2d0/9*b)*b)*b)*b)*b**5

        gammab(4,4,2) = (-2d0/3+(12d0/7+(-3d0/2+4d0/9*a)*a)*a)*a**6-
     1                  (-2d0/3+(12d0/7+(-3d0/2+4d0/9*b)*b)*b)*b**6

        gammab(4,4,3) = (1d0/6+(-4d0/7+(5d0/8-2d0/9*a)*a)*a)*a**6-
     1                  (1d0/6+(-4d0/7+(5d0/8-2d0/9*b)*b)*b)*b**6

        do 3 l =2,4
           do 2 k =1,l-1
              do 1 m =1,3
                 gammab(l,k,m) = gammab(k,l,m)
     1        continue
     2     continue
     3  continue

        return
        end



        SUBROUTINE input
c
c       Routine INPUT reads in all initial data about the physical
c       features and configuration of the cable. The position
c       of the cable and the tension in the cable can be given by
c       either an roughly guess or an output from the previous
c       calculation.
c
c       The initial data are stored in the file of which its name
c       is provided by the user. All variables used in this
c       routine are defined just before the excecutation of the
c       'read' command.
c
c       Variables   -
c          vdfn     -  definition of variables
c
c       Subroutine called by the main program.
c
        implicit double precision (a-h,o-z)
        include 'parametrics.inc'
        include 'variables.inc'
        character title*80,vdfn*40
c
c       1. global constants
c
        read(idatin,'(a)') title
        read(idatin,*)
        read(idatin,'(a)') title
        read(idatin,*)
c
c       a. indexs
c       1) node          (# of nodes)
c       2) neltyp        (# of different element types)
c       3) ndim          (# of dimensions- 2 or 3)
c       4) job           (1 = static, 2 = dynamic)
c       5) jsnap         (snapshot each jsnap iterations)
c       6) jmotn         (1 = no motion, 2 = seas + motion)
c       7) ncnstr        (# of constrained nodes)
c
        read(idatin,'(a)') title
        read(idatin,*)
        read(idatin,1000) node,   vdfn
        read(idatin,1000) neltyp, vdfn
        read(idatin,1000) ndim,   vdfn
        read(idatin,1000) job,    vdfn
        read(idatin,1000) jsnap,  vdfn
        read(idatin,1000) jmotn,  vdfn
        read(idatin,1000) ncnstr, vdfn
        read(idatin,*)
c
c       b. parameters
c       1) relax,damp,iter       (relaxtion, damping const., # iterations)
c       2) tbgn,tend,tstep       (begin time, end time & time stepi)
c       3) cd,cm                 (drag and added-mass coefficients)
c       4) ruf,rui,g             (mass density of water, fluid; gravity)
c       5) btmdpth               (bottom depth)
c
        read(idatin,'(a)') title
        read(idatin,*)
        read(idatin,1010) relax,damp,iter,vdfn
```

```
      read(idatin,1020) tbgn,tend,tstep,vdfn
      read(idatin,1030) cd,cm,vdfn
      read(idatin,1035) ruf,rui,g,vdfn
      read(idatin,1037) btmdpth,vdfn
      read(idatin,*)
      read(idatin,*)
c
c     ii. element and nodal information
c
c     a. physical properties for each element type
c     1) eel                  (young's mod)
c     2) moi                  (moment of inertia)
c     3) afel(1-3)            (cross-section area)
c     4) aiel(1-3)            (internal cavity area)
c     5) dfel(1-3)            (dist. to outermost fiber
c     6) mel(1-3)             (mass of the rod per unit length)
c     7) del(1-3)             (diameters for hydrodynamic forces)
c     8) buoyel(1-3)          (total buoyancy/unit length)
c
      read(idatin,'(a)') title
      read(idatin,*)
      read(idatin,'(a)') title
      read(idatin,*)
      do 1 i = 1,neltyp
         read(idatin,1040) eel(i),vdfn
         read(idatin,1050) (moi(i,j),    j = 1,3), vdfn
         read(idatin,1050) (afel(i,j),   j = 1,3), vdfn
         read(idatin,1050) (aiel(i,j),   j = 1,3), vdfn
         read(idatin,1050) (dfel(i,j),   j = 1,3), vdfn
         read(idatin,1050) (mel(i,j),    j = 1,3), vdfn
         read(idatin,1050) (del(i,j),    j = 1,3), vdfn
         read(idatin,1050) (buoyel(i,j), j = 1,3), vdfn
         read(idatin,*)
    1 continue
c
c     b. coordinates and lambdas for each node
c     1) r(1-3)           (x, y and z positions at node)
c     2) rprm(1-3)        (x', y' and z' slopes)
c     3) rdot(1-3)        (x-dot, y-dot and z-dot)
c     4) rpmdt(1-3)       (x'-dot, y'-dot and z'-dot)
c     5) lambda(1-2)      (lambda at prev. ctr. and at node)
c
      read(idatin,'(a)') title
      do 2 i = 1,node
         read(idatin,*)
         if (ndim .eq. 2) then
            read(idatin,1070) (r(i,j),     j = 1,2), vdfn
            read(idatin,1070) (rprm(i,j),  j = 1,2), vdfn
            read(idatin,1070) (rdot(i,j),  j = 1,2), vdfn
            read(idatin,1070) (rpmdt(i,j), j = 1,2), vdfn
         else
            read(idatin,1080) (r(i,j),     j = 1,3), vdfn
            read(idatin,1080) (rprm(i,j),  j = 1,3), vdfn
            read(idatin,1080) (rdot(i,j),  j = 1,3), vdfn
            read(idatin,1080) (rpmdt(i,j), j = 1,3), vdfn
         endif
         read(idatin,1090) (lambda(i,j), j = 1,2), vdfn
    2 continue
      read(idatin,*)
c
c     c. connections between nodes
c     1) lnthel        (element length)
c     2) jeltyp        (element type)
c
      read(idatin,'(a)') title
      read(idatin,'(a)') title
      read(idatin,1110) (lnthel(i),jeltyp(i),vdfn, i = 1,node-1)
      read(idatin,*)
      read(idatin,*)
c
c     iii. nodal constraints
c
c     1) nodec                (constrainted nodal #)
c     a. linear constraints
c     2) direction cosines
c        refcos(1)             (reference axis)
c        refcos(2)             (1st perpendicular axis)
c        refcos(3)             (2nd perpendicular axis)
c        note:
c        . in 2 dimensional problem, only the direction cosines of
c          the reference axis are needed. The 1st perpendicular
c          axis is then defined by
c              refcos(2,1) = -refcos(1,2)
c              refcos(2,2) =  refcos(1,1)
c        . in 3 dimensional problem, the 2nd perpendicular axis
c          is defined by the cross product of the reference axis
c          and the 1st perpendicular axis
c              refcos(3) = refcos(1) x refcos(2)
c     3) refpos(1-3)          (reference position)
c     4) spgcont(1-3)         (spring constant)
c     5) dmpcont(1-3)         (damping constant)
c     6) finit(1-3)           (initial force)
c     7) cmass                (concentrated mass)
c     8) freq                 (motion frequency (rad/sec))
c     b. motion of reference point at node #
c        1 = in-phase, 2 = out-phase
c     9) disamp(x,1-2)        (x displacement amplitude)
c              (y,1-2)        (y displacement amplitude)
c              (z,1-2)        (z displacement amplitude)
c
      read(idatin,'(a)') title
      read(idatin,*)
      do 5 i = 1,ncnstr
         read(idatin,1160) title,nodec(i)
         read(idatin,*)
         read(idatin,'(a)') title
         read(idatin,'(a)') title
         read(idatin,'(a)') title
         if (ndim .eq. 2) then
            do 3 j = 1,2
               read(idatin,1170) refcos(i,1,j),refpos(i,j),vdfn
    3       continue
            read(idatin,1180) vdfn
            refcos(i,2,1) = -refcos(i,1,2)
            refcos(i,2,2) =  refcos(i,1,1)
         else
            do 4 j = 1,3
               read(idatin,1175) refcos(i,1,j),refcos(i,2,j),
     1                           refpos(i,j),vdfn
    4       continue
            refcos(i,3,1) = refcos(i,1,2)*refcos(i,2,3)-
     1                      refcos(i,2,2)*refcos(i,1,3)
            refcos(i,3,2) = refcos(i,1,3)*refcos(i,2,1)-
     1                      refcos(i,2,3)*refcos(i,1,1)
            refcos(i,3,3) = refcos(i,1,1)*refcos(i,2,2)-
     1                      refcos(i,2,1)*refcos(i,1,2)
         endif
         if (ndim .eq. 2) then
```

```
            read(idatin,1190) (spgcont(i,j), j = 1,2), vdfn
            read(idatin,1200) (dmpcont(i,j), j = 1,2), vdfn
            read(idatin,1190) (finitl(i,j),  j = 1,2), vdfn
         else
            read(idatin,1195) (spgcont(i,j), j = 1,3), vdfn
            read(idatin,1205) (dmpcont(i,j), j = 1,3), vdfn
            read(idatin,1195) (finitl(i,j),  j = 1,3), vdfn
         endif
         read(idatin,1210) cmass(i),vdfn
         read(idatin,*)
         read(idatin,'(a)') title
         read(idatin,*)
         read(idatin,1220) freq(i),vdfn
         read(idatin,*)
         read(idatin,'(a)') title
         read(idatin,1230) disamp(i,1,1),disamp(i,1,2),vdfn
         read(idatin,1230) disamp(i,2,1),disamp(i,2,2),vdfn
         if (ndim .eq. 2) then
            read(idatin,1240) vdfn
         else
            read(idatin,1230) disamp(i,3,1),disamp(i,3,2),vdfn
         endif
         read(idatin,*)
    5  continue
       read(idatin,*)
c
c      iv. environmental data
c
c      nvc        (# of evaluations at which velocities of current
c                  are given)
c      vc(*,4)    (depths, x-vel, y-vel, and z-vel)
c      aw         (wave amplitude)
c      sigmaw     (circular wave frequency(rad/sec))
c      dirw       (wave direction with respective to x-axis)
c
c      a. currents at nvc depths
c
       read(idatin,'(a)') title
       read(idatin,*)
       read(idatin,1120) title,nvc,title
       read(idatin,*)
       read(idatin,'(a)') title
       read(idatin,1130) ((vc(i,j),j=1,4),i=1,nvc)
       read(idatin,*)
c
c      b. wave
c
       read(idatin,'(a)') title
       read(idatin,*)
       read(idatin,1140) aw,sigmaw,vdfn
       read(idatin,1150) dirw,vdfn
       read(idatin,*)
       read(idatin,*)

       return
1000   format(1x,13,7x,a)
1010   format(2f10.5,15,12x,a)
1020   format(3f10.5,7x,a)
1030   format(2f10.5,17x,a)
1035   format(3f10.5,7x,a)
1037   format(1f10.2,27x,a)
1040   format(1x,1pe11.5,29x,a)
1050   format(3(1x,1pe11.5),5x,a)
1070   format(2(1x,f11.5),19x,a)
1080   format(3(1x,f11.5),7x,a)
1090   format(2(1pe12.5),19x,a)
1110   format(1x,f7.1,1x,17,7x,a)
1120   format(a14,13,a7)
1130   format(4f10.4)
1140   format(f10.6,f10.5,7x,a)
1150   format(f10.7,17x,a)
1160   format(a8,13)
1170   format(4x,f8.5,28x,f10.3,4x,a)
1175   format(4x,2(f8.5,4x),12x,f10.3,4x,a)
1180   format(54x,a)
1190   format(2(1pe12.5),28x,a)
1195   format(3(1pe12.5),16x,a)
1200   format(2x,2(f10.5,2x),28x,a)
1205   format(2x,3(f10.5,2x),16x,a)
1210   format(2x,f10.5,42x,a)
1220   format(f15.10,12x,a)
1230   format(2(1x,f9.3),7x,a)
1240   format(27x,a)
       end



       SUBROUTINE mass(nel,normal)
c
c      Routine MASS evaluates the virtual mass matrix Massmat:
c      massmat=(rut*At+rui*Ai)I+ruf*Af*cm*Normal
c      where massmat=massmat(i,j,index), i,j = [1,ndim]
c                                        index = [1,3]
c
c      Variables   -
c         ndim     -  dimensions
c         massmat  -  virtual mass matrix for an element at
c                     specified cross-section defined by index
c         nel      -  element #
c         normal   -  trnasfer matrix, = I -r'*r'
c
c      Subroutine called by DYEQ16
c
       implicit double precision (a-h,o-z)
       include 'parametrics.inc'
       include 'variables.inc'
       double precision normal(3,3,3)
       parameter(pi=3.141592654d0)

       do 3 m = 1,3
          do 2 i = 1,ndim
             do 1 j = 1,ndim
                massmat(i,j,m) = ruf*0.25*pi*del(jeltyp(nel),m)*
     1                           del(jeltyp(nel),m)*cm*normal(i,j,m)
                if (i .eq. j) massmat(i,j,m) = massmat(i,j,m)+
     1                                         mel(jeltyp(nel),m)+
     2                                         rui*alel(jeltyp(nel),m)
     1          continue
     2       continue
     3    continue

          return
          end
```

```
      SUBROUTINE matrix(a,b,c,i,j,k,ni,nj,nk)
c
c     Matrix mutiplication:  C = AB
c     The maximum dimensions of matrice A,B, and C are defined by
c     A[ni,nj], B[nj,nk], and C[ni,nk], respectively.
c
c     Variables   -
c        a        -  matrix i*j
c        b        -  matrix j*k
c        c        -  matrix i*k
c
c     Subroutine called by QFORCE,QGFORCE
c
      implicit double precision (a-h,o-z)
      dimension a(ni,nj),b(nj,nk),c(ni,nk)

      do 3 m = 1,i
         do 2 n = 1,k
            c(m,n) = 0.0d0
            do 1 l = 1,j
               c(m,n) = c(m,n)+a(m,l)*b(l,n)
     1      continue
     2   continue
     3 continue

      return
      end



      SUBROUTINE mubtm(a,b,muab)
c
c     Routine MUBTM calculats coefficients mu, named muab, in the
c     interval [a,b], where 0 <= a < b <= 1.
c     mu=integrate[a[m]*p[n],{xi,a,b}]
c
c     Global Variables -
c        a,b    -  interval bounds
c        muab   -  coefficients defined by the integral, muab[4,3].
c
c     Subroutine called by STEQ15, DYEQ15
c
      double precision a,b,muab(4,3)

      muab(1,1) =
     1    (-1d0+(3d0/2+(1d0/3+(-11d0/4+(12d0/5-2d0/3*a)*a)*a)*a)*a)*a-
     2    (-1d0+(3d0/2+(1d0/3+(-11d0/4+(12d0/5-2d0/3*b)*b)*b)*b)*b)*b

      muab(1,2) = (-2d0+(4d0/3+(3d0+(-4d0+4d0/3*a)*a)*a)*a)*a*a-
     1            (-2d0+(4d0/3+(3d0+(-4d0+4d0/3*b)*b)*b)*b)*b*b

      muab(1,3) = (1d0/2+(-2d0/3+(-3d0/4+(8d0/5-2d0/3*a)*a)*a)*a)*a*a-
     1            (1d0/2+(-2d0/3+(-3d0/4+(8d0/5-2d0/3*b)*b)*b)*b)*b*b

      muab(2,1) = (-1d0/2+(5d0/3+(-9d0/4+(7d0/5-1d0/3*a)*a)*a)*a)*a*a-
     1            (-1d0/2+(5d0/3+(-9d0/4+(7d0/5-1d0/3*b)*b)*b)*b)*b*b

      muab(2,2) = (-4d0/3+(3d0+(-12d0/5+2d0/3*a)*a)*a)*a*a*a-
     1            (-4d0/3+(3d0+(-12d0/5+2d0/3*b)*b)*b)*b*b*b

      muab(2,3) = (1d0/3+(-1d0+(1d0-1d0/3*a)*a)*a)*a*a*a-
     1            (1d0/3+(-1d0+(1d0-1d0/3*b)*b)*b)*b*b*b

      muab(3,1) = (-1d0+(11d0/4+(-12d0/5+2d0/3*a)*a)*a)*a*a*a-
     1            (-1d0+(11d0/4+(-12d0/5+2d0/3*b)*b)*b)*b*b*b

      muab(3,2) = (-3d0+(4d0-4d0/3*a)*a)*a**4-
     1            (-3d0+(4d0-4d0/3*b)*b)*b**4

      muab(3,3) = (3d0/4+(-8d0/5+2d0/3*a)*a)*a**4-
     1            (3d0/4+(-8d0/5+2d0/3*b)*b)*b**4

      muab(4,1) = (1d0/3+(-1d0+(1d0-1d0/3*a)*a)*a)*a*a*a-
     1            (1d0/3+(-1d0+(1d0-1d0/3*b)*b)*b)*b*b*b

      muab(4,2) = (1d0+(-8d0/5+2d0/3*a)*a)*a**4-
     1            (1d0+(-8d0/5+2d0/3*b)*b)*b**4

      muab(4,3) = (-1d0/4+(3d0/5-1d0/3*a)*a)*a**4-
     1            (-1d0/4+(3d0/5-1d0/3*b)*b)*b**4

      return
      end



      SUBROUTINE output
c
c     Routine OUTPUT copies data describing physical features of
c     the cable, from the input file to the output file. The
c     The equilibrium position of the cable and tension in the cable
c     are copied directly from the newly calculated data.
c
c     The output data are written to the file of which its name
c     is provided by the user. The output file can be used as an
c     input file for another calculation, since the format of output
c     is as same as that of the input file.
c
c     Although all the variables used in this routine are previously
c     defined in the subroutine INPUT, the definitions of the
c     variables are as well given in this routine for the convenience
c     of the users.
c
c     Subroutine called by the main program.
c
      implicit double precision (a-h,o-z)
      include 'parametrics.inc'
      include 'variables.inc'
      character*80 title

      rewind idatin
c
c     1. global constants
c
      read(idatin,'(a)')   title
      write(idatout,'(a)') title
      write(idatout,*)
      write(idatout,'(a)') 'I. Global Constants'
      write(idatout,*)
c
c     a. indexs
c     1) node          (# of nodes)
c     2) neltyp        (# of different element types)
c     3) ndim          (# of dimensions- 2 or 3)
c     4) job           (1 - static, 2 - dynamic)
```

```
c       5) jsnap        (snapshot each jsnap iterations)
c       6) jmotn        (1 = no motion, 2 = seas + motion)
c       7) ncnstr       (# of constrained nodes)
c
        write(idatout,'(a)') 'a. Indexs'
        write(idatout,*)
        write(idatout,'(1x,13,7x,a)')
     1     node,    'NODE    (# of nodes)'
        write(idatout,'(1x,13,7x,a)')
     1     neltyp, 'NELTYP (# of different element types)'
        write(idatout,'(1x,13,7x,a)')
     1     ndim,    'NDIM    (# of dimensions - 2 or 3)'
        write(idatout,'(1x,13,7x,a)')
     1     job,     'JOB     (1 = static, 2 = dynamic)'
        write(idatout,'(1x,13,7x,a)')
     1     jsnap,   'JSNAP   (snapshot each JSNAP iterations)'
        write(idatout,'(1x,13,7x,a)')
     1     jmotn,   'JMOTN   (1 = sin. motion, 2 = input forced motion)'
        write(idatout,'(1x,13,7x,a)')
     1     ncnstr, 'NCNSTR (# of constrained nodes)'
        write(idatout,*)
c
c       b. parameters
c       1) relax,damp,iter      (relaxtion, damping const., # iterations)
c       2) tbgn,tend,tstep      (begin time, end time & time step)
c       3) cd,cm                (drag and added-mass coefficients)
c       4) ruf,rui,g            (mass density of water, fluid; gravity)
c       5) vx,vz                (x and z components of forward speed)
c       6) btmdpth              (bottom depth)
c
        write(idatout,'(a)') 'b. Parameters'
        write(idatout,*)
        write(idatout,'(2f10.5,15,12x,a)')
     1     relax,damp,iter,'relaxation, # iterations'
        write(idatout,'(3f10.5,7x,a)')
     1     tbgn,tend,tstep,'begin time, end time & time step'
        write(idatout,'(2f10.5,17x,a)')
     1     cd,cm,          'drag and added-mass coefficients'
        write(idatout,'(3f10.5,7x,a)')
     1     ruf,rui,g,      'mass density of water, fluid; gravity'
        write(idatout,'(1f10.2,27x,a)')
     1     btmdpth,        'bottom depth'
        write(idatout,*)
        write(idatout,*)
c
c       ii. element and nodal information
c
c       a. physical properties for each element type
c       1) eel                  (young's mod)
c       2) moi                  (moment of inertia)
c       3) afel(1-3)            (cross-section area)
c       4) aiel(1-3)            (internal cavity area)
c       5) dfel(1-3)            (dist. to outermost fiber
c       6) mel(1-3)             (mass of the rod per unit length)
c       7) del(1-3)             (diameters for hydrodynamic forces)
c       8) buoyel(1-3)          (total buoyancy/unit length)
c
        write(idatout,'(a)') 'II. Element and Nodal Information'
        write(idatout,*)
        write(idatout,'(a)')
     1     'a. Physical Properties for Each Element Type'
        write(idatout,*)
        do 1 i = 1,neltyp
           write(idatout,'(1x,1pe11.5,29x,a,i3)')
     1        eel(i),'Young''s mod: type',i
           write(idatout,'(3(1x,1pe11.5),5x,a)') (moi(i,j),    j = 1,3),
     1        'moment of inertia (I)'
           write(idatout,'(3(1x,1pe11.5),5x,a)') (afel(i,j),   j = 1,3),
     1        'cross-section area (Af)'
           write(idatout,'(3(1x,1pe11.5),5x,a)') (aiel(i,j),   j = 1,3),
     1        'internal cavity  area (Ai)'
           write(idatout,'(3(1x,1pe11.5),5x,a)') (dfel(i,j),   j = 1,3),
     1        'dist. to outermost fiber'
           write(idatout,'(3(1x,1pe11.5),5x,a)') (mel(i,j),    j = 1,3),
     1        'mass of the rod per unit length'
           write(idatout,'(3(1x,1pe11.5),5x,a)') (del(i,j),    j = 1,3),
     1        'diameters for hydrodynamic forces'
           write(idatout,'(3(1x,1pe11.5),5x,a)') (buoyel(i,j), j = 1,3),
     1        'total buoyancy/unit length'
           write(idatout,*)
     1  continue
c
c       b. coordinates and lambdas for each node
c       1) r(1-3)               (x, y and z positions at node)
c       2) rprm(1-3)            (x', y' and z' slopes)
c       3) rdot(1-3)            (x-dot, y-dot and z-dot)
c       4) rpmdt(1-3)           (x'-dot, y'-dot and z'-dot)
c       5) lambda(1-2)          (lambda at prev. ctr. and at node)
c
        write(idatout,'(a)') 'b. Coordinates and Lambdas for Each Node'
        do 2 i = 1,node
           write(idatout,*)
           if (ndim .eq. 2) then
              write(idatout,'(2(1x,f11.5),19x,a,i3)')
     1           (r(i,j),     j = 1,2),'x and y position at node #',i
              write(idatout,'(2(1x,f11.5),19x,a)')
     1           (rprm(i,j), j = 1,2),'x'' and y'' slopes'
              write(idatout,'(2(1x,f11.5),19x,a)')
     1           (rdot(i,j), j = 1,2),'x-dot and y-dot'
              write(idatout,'(2(1x,f11.5),19x,a)')
     1           (rpmdt(i,j),j = 1,2),'x''-dot and y''-dot'
           else
              write(idatout,'(3(1x,f11.5),7x,a,i3)')
     1           (r(i,j),     j = 1,3),'x, y and z position at node #',i
              write(idatout,'(3(1x,f11.5),7x,a)')
     1           (rprm(i,j), j = 1,3),'x'', y'' and z'' slopes'
              write(idatout,'(3(1x,f11.5),7x,a)')
     1           (rdot(i,j), j = 1,3),'x-dot, y-dot and z-dot'
              write(idatout,'(3(1x,f11.5),7x,a)')
     1           (rpmdt(i,j),j = 1,3),'lambda at prev. ctr. and at node'
           endif
           write(idatout,'(2(1pe12.5),19x,a)')
     1        (lambda(i,j), j = 1,2),'lambda at prev. ctr. and at node'
     2  continue
        write(idatout,*)
c
c       c. connections between nodes
c       1) lnthel           (element length)
c       2) jeltyp           (element type)
c
        write(idatout,'(a)') 'c. Connections between Nodes'
        write(idatout,'(a)') '  length   elmt type'
        do 3 i = 1,node-1
           write(idatout,'(1x,f7.1,1x,i7,7x,a,i3,a,i3)')
     1        lnthel(i),jeltyp(i),'from node',i,' to',i+1
     3  continue
        write(idatout,*)
        write(idatout,*)
```

```
c
c       iii. nodal constraints
c
c       1) nodec                  (constrainted nodal #)
c       a. linear constraints
c       2) direction cosines
c          refcos(1)              (reference axis)
c          refcos(2)              (1st perpendicular axis)
c          refcos(3)              (2nd perpendicular axis)
c       3) refpos(1-3)            (reference position)
c       4) spgcont(1-3)           (spring constant)
c       5) dmpcont(1-3)           (damping constant)
c       6) finitl(1-3)            (initial force)
c       7) cmass                  (concentrated mass)
c       8) freq                   (motion frequency (rad/sec))
c       b. motion of reference point at node #
c          1 = in-phase, 2 = out-phase
c       9) disamp(x,1-2)          (x displacement amplitude)
c                (y,1-2)          (y displacement amplitude)
c                (z,1-2)          (z displacement amplitude)
c
        write(idatout,'(a)') 'III. Nodal Constraints'
        write(idatout,*)
        do 6 i = 1,ncnstr
           write(idatout,'(a,i3)') 'Node No.',nodec(i)
           write(idatout,*)
           write(idatout,'(a)') 'a. Linear Constraints'
           write(idatout,'(a)')
     1        '              direction consines             reference'
           write(idatout,'(a)')
     1        '    reference   1st perp.   2nd perp.       position'
           if (ndim .eq. 2) then
              write(idatout,'(4x,2(f8.5,4x),12x,f10.3,4x,a)')
     1           refcos(i,1,1),refcos(i,2,1),refpos(i,1),'x direction'
              write(idatout,'(4x,2(f8.5,4x),12x,f10.3,4x,a)')
     1           refcos(i,1,2),refcos(i,2,2),refpos(i,2),'y direction'
              write(idatout,'(54x,a)') 'z direction'
           else
              write(idatout,'(4x,3(f8.5,4x),f10.3,4x,a)')
     1           refcos(i,1,1),refcos(i,2,1),refcos(i,3,1),refpos(i,1),
     2           'x direction'
              write(idatout,'(4x,3(f8.5,4x),f10.3,4x,a)')
     1           refcos(i,1,2),refcos(i,2,2),refcos(i,3,2),refpos(i,2),
     2           'y direction'
              write(idatout,'(4x,3(f8.5,4x),f10.3,4x,a)')
     1           refcos(i,1,3),refcos(i,2,3),refcos(i,3,3),refpos(i,3),
     2           'z direction'
5             continue
           endif
           if (ndim .eq. 2) then
              write(idatout,'(2(1pe12.5),30x,a)')
     1           (spgcont(i,j), j = 1,2),'spring constant'
              write(idatout,'(2x,2(f10.5,2x),28x,a)')
     1           (dmpcont(i,j), j = 1,2),'damping constant'
              write(idatout,'(2(1pe12.5),30x,a)')
     1           (finitl(i,j),  j = 1,2), 'initial force'
           else
              write(idatout,'(3(1pe12.5),18x,a)')
     1           (spgcont(i,j), j = 1,3),'spring constant'
              write(idatout,'(2x,3(f10.5,2x),16x,a)')
     1           (dmpcont(i,j), j = 1,3),'damping constant'
              write(idatout,'(3(1pe12.5),18x,a)')
     1           (finitl(i,j),  j = 1,3),'initial force'
           endif
           write(idatout,'(2x,f10.5,42x,a)')cmass(i),'concentrated mass'
           write(idatout,*)
           write(idatout,'(a,i3)')
     1        'b. Motion of Reference Point at Node',nodec(i)
           write(idatout,*)
           write(idatout,'(f15.10,12x,a)')
     1        freq(i),'motion frequencey (rad/sec)'
           write(idatout,*)
           write(idatout,'(a)') '  in-phase out-phase'
           write(idatout,'(2(1x,f9.3),7x,a)')
     1        disamp(i,1,1),disamp(i,1,2),'x displacement amplitude'
           write(idatout,'(2(1x,f9.3),7x,a)')
     1        disamp(i,2,1),disamp(i,2,2),'y displacement amplitude'
           if (ndim .eq. 2) then
              write(idatout,'(27x,a)') 'z displacement amplitude'
           else
              write(idatout,'(2(1x,f9.3),7x,a)')
     1           disamp(i,3,1),disamp(i,3,2),'z displacement amplitude'
           endif
           write(idatout,*)
6       continue
        write(idatout,*)
c
c       iv. environmental data
c
c       nvc       (# of evaluations at which velocities of current
c                  are given)
c       vc(*,4)   (depths, x-vel, y-vel, and z-vel)
c       aw        (wave amplitude)
c       sigmaw    (circular wave frequency(rad/sec))
c       dirw      (wave direction with respective to x-axis)
c
c       a. currents at nvc depths
c
        write(idatout,'(a)') 'IV. Environmental Data'
        write(idatout,*)
        write(idatout,'(a,i3,a)') 'a. Currents at',nvc,' depths'
        write(idatout,*)
        write(idatout,'(a)') '    depth      x-vel      y-vel      z-vel'
        write(idatout,'(4f10.4)') ((vc(i,j), j = 1,4), i = 1,nvc)
        write(idatout,*)
c
c       b. wave
c
        write(idatout,'(a)') 'b. Waves'
        write(idatout,*)
        write(idatout,'(f10.6,f10.5,7x,a)') aw,sigmaw,
     1     'wave amplitude and frequency (rad/sec)'
        write(idatout,'(f10.7,17x,a)') dirw,
     1     'wave direction relative to x axis'
        write(idatout,*)
        write(idatout,*)

        return
        end




        SUBROUTINE polint(xa,ya,n,x,y,dy)
c
c       Given arrays xa and ya, each of length n, and given a value
c       x, this routine returns a value y, and an error estimate dy.
```

```
c       if p(x) is the polynomial of degree n-1 such that p(xa(j))=ya(j)
c       ,j=1,...,n, then the return value y=p(x).
c
c       Subroutine called by VELOCITY, PRESSURE
c
        implicit double precision (a-h,o-z)
        dimension xa(10),ya(10),c(10),d(10)

        ns = 1
        dif = abs(x-xa(1))
c
c       here we find the index ns of the closest table entry,
c
        do 1 i = 1,n
           dift = abs(x-xa(i))
           if (dift .lt. dif) then
              ns = i
              dif = dift
           endif
c
c       and initialize the tableau of c's and d's.
c
           c(i) = ya(i)
           d(i) = ya(i)
    1   continue
c
c       this is the initial approximation of y.
c
        y = ya(ns)
        ns = ns-1
c
c       for each column of the tableau, we loop over the current
c       c's and d's and update them.
c
        do 3 m = 1,n-1
           do 2 i = 1,n-m
              ho = xa(i)-x
              hp = xa(i+m)-x
              w = c(i+1)-d(i)
              den = ho-hp
c
c       this error can occur if two input xa's are(to within roundoff)
c       identical.
c
              if (den .eq. 0.0d0) pause
              den = w/den
c
c       here the c's and d's are updated.
c
              d(i) = hp*den
              c(i) = ho*den
    2      continue
c
c       After each column in the tableau is completed, we decide
c       which correction, c or d, we want to add to our
c       accumulating value of y, i.e. which path to take through
c       the tableau-forking up or down. We do this in such a way
c       as to take the most "straight line" route through the
c       tableau to its apex, updating ns accordingly to keep
c       track of where we are. This route keeps the partial
c       approximations centered(insofar as possible) on the
c       target x. The last dy added is thus the error indication.
c
           if (2*ns .lt. n-m) then
              dy = c(ns+1)
           else
              dy = d(ns)
              ns = ns-1
           endif
           y = y+dy
    3   continue

        return
        end



        Subroutine presprm(lt,at,pt,pprm)
c
c       Routine PPRSPRM evalautes partial derivative of pt*at*rprm
c       with respective to s.
c
c       Variables   -
c          at       -  cross-section area of the tube (element)
c          lt       -  element length
c          pprm     -  partial derivative of pt*at*rprm with respective
c                      to s
c          pt       -  pressure on the ends and mid-section of the tube
c                      pt can either be pressure due to the outer fluid
c                      filed or pressure due to the inner fluid.
c          rprm     -  r'
c
c       Subroutine called by QFORCE
c
        implicit double precision (a-h,o-z)
        include 'parametrics.inc'
        include 'variables.inc'
        dimension at(3),pprm(3,3),pt(3)
        double precision lt
        parameter(pi = 3.141592654d0)
c
c       partial derivative of pt*at*rprm w.s.t. s at xi = 0, 0.5, &1
c
        do 1 n = 1,ndim
           pprm(1,n) = (-3*at(1)+4*at(2)-at(3))*pt(1)*u(2,n)/lt/lt+
     1                 (-3*pt(1)+4*pt(2)-pt(3))*at(1)*u(2,n)/lt/lt+
     2                 (-6*u(1,n)-4*u(2,n)+6*u(3,n)-2*u(4,n))*
     3                 at(1)*pt(1)/lt/lt
           pprm(2,n) = (-u(2,n)+u(4,n))*at(2)*pt(2)/lt/lt+
     1                 (-1.5*u(1,n)-0.25*u(2,n)+1.5*u(3,n)-0.25*u(4,n))*
     2                 (-at(1)+at(3))*pt(2)/lt/lt+
     3                 (-1.5*u(1,n)-0.25*u(2,n)+1.5*u(3,n)-0.25*u(4,n))*
     4                 (-pt(1)+pt(3))*at(2)/lt/lt
           pprm(3,n) = (at(1)-4*at(2)+3*at(3))*pt(3)*u(4,n)/lt/lt+
     1                 (pt(1)-4*pt(2)+3*pt(3))*at(3)*u(4,n)/lt/lt+
     2                 (6*u(1,n)+2*u(2,n)-6*u(3,n)+4*u(4,n))*
     3                 at(3)*pt(3)/lt/lt
    1   continue

        return
        end



        SUBROUTINE pressure(pf,pi,vf,phit)
c
```

```
c       Routine PRESSURE evaluates the pressure PI which would exist in
c       an undisturbed flow field at a point given on the tube.
c       The pressure is evaluated by Euler's equation.
c       p=ruf*(c-phit-g*y-0.5*v*v)
c
c       Variables   -
c          c        -  constant in Euler's equation
c          pf       -  pressures on outer cross-section of the element
c                      at location xi = 0, 0.5, & 1, pf(m)
c          phit     -  partial derivative of velocity potential
c                      with respective to t
c          pi       -  pressures on inner cross-section of the element
c                      at location xi = 0, 0.5, & 1, pi(m)
c          v        -  velocity
c          vf       -  velocities of fluid (sea water) due to currents
c                      and waves, vf(mn)
c          vsqr     -  velocity square
c
c       Subroutine called by SLVEQ16
c
        implicit double precision (a-h,o-z)
        include 'parametrics.inc'
        include 'variables.inc'
        dimension vf(3,3),pf(3),pi(3),x0(10),y0(10)
c
c       It is assumed that there is no wave in the infinity. Constant c
c       in Euler's equation is evaluated at infinite free surface
c       where y = 0. Only the influence of current is considered.
c
        c = 0.0d0
        do 2 n = 1,ndim
           do 1 k = 1,nvc
              x0(k) = vc(k,1)
              y0(k) = vc(k,n+1)
    1      continue
           call polint(x0,y0,nvc,0.0d0,v,dy)
           c = c + 0.5*v*v
    2   continue
c
c       evaluate press pf
c
        do 4 m = 1,3
c
c       The inner pressure is tentativly assumed to be zero!!!
c
           vsqr  = 0.0d0
           do 3 n = 1,ndim
              vsqr = vsqr+vf(m,n)*vf(m,n)
    3      continue
           y = u(m,2)
           if (m.eq.2) y=0.5*u(1,2)+0.125*u(2,2)+0.5*u(3,2)-0.125*u(4,2)
           if (y .le. 0.0d0) then
              pf(m) = ruf*(c-phit-g*y-0.5*vsqr)
              pi(m) = -rui*g*y
           else
              pf(m) = 0.0d0
              pi(m) = 0.0d0
           endif
    4   continue

        return
        end
```

```
        SUBROUTINE qforce(nel,normal,vf,af,xa1,xb1,xa2,xb2)
c
c       Routine QFORCE evaluates the generalized force Q at location
c       xi=0,0.5 & 1 of the tube (element).
c
c       Variables   -
c          af       -  acceleration of fluid (sea water), af(mn)
c          ndis, dis-
c                      number of points(i.e. two ends and mid point of
c                      the element) and distance of these points to the
c                      sea floor
c          nel      -  element #
c          nroot, xroot   -
c                      # of points and location of points at which
c                      the element acrossing the bottom line.
c          q        -  generalized force (except for bottom support
c                      force)
c          accel    -  acceleration of fluid in one of the 1
c                      cross-sections of location xi = 0, 0.5, & 1
c          acceln   -  acceleration in the normal direction
c          normal   -  transfer matrix, normal(3,3,index)
c          natxi    -  transfer matrix at one of cross-sections of
c                      location xi = 0, 0.5, & 1
c          niplsnatxi  -  = I +Cm*Normal
c          rvf      -  relative velocity of the element in flow field
c          rvn      -  relative velocity in the normal direction
c          rvnsqr   -  square of relative velocity in the
c                      normal direction
c          vf       -  velocity of fluid(i.e. sea water), vf(mn)
c          wb       -  summation of buoyance and gravity forces
c          wbd      -  (net weight of the element in water)
c                      ------------------------------------, wbd(m)
c                            (dist to out most fiber)
c          (xa1,xb1), (xa2,xb2)  -
c                      range of the 1st and 2nd under bottom portions of
c                      the element
c          y0       -  simulated ocean depth
c
c       Subroutine called by SLVEQ15
c
        implicit double precision (a-h,o-z)
        include 'parametrics.inc'
        include 'variables.inc'
        dimension rvf(3),rvn(3),vf(3,3)
        dimension af(3,3),accel(3),acceln(3),xroot(3)
        double precision normal(3,3,3),natxi(3,3),niplsnatxi(3,3)
        parameter (pi=3.141592654)

        do 2 n = 1,ndim
           do 1 m = 1,3
              q(m,n) = 0.0d0
    1      continue
           rvf(n)   = 0.0d0
           accel(n) = 0.0d0
    2   continue

        do 10 m = 1,3
c
c       obtain transfer matrix n and relative velocity vf for mode m
c
           do 4 i = 1,ndim
```

```
c
c       rvf = vf-rdot
c
              rvf(i) = vf(m,i)
              if (job .eq. 2) then
                 accel(i) = af(m,i)
                 rvf(i) = vf(m,i)-udot(m,i)
                 if (m .eq. 2) rvf(i) = vf(m,i)-(0.5*udot(1,i)+0.125*
     1                         udot(2,i)+0.5*udot(3,i)-0.125*udot(4,i))
              endif

              do 3 j = 1,ndim
                 natxi(i,j) = normal(i,j,m)
    3         continue
    4       continue
c
c       1) inertia term: ruf*af*(1+cm*n)*vf dot
c
           if (job .eq. 2) then
              do 6 i = 1,ndim
                 do 5 j = 1,ndim
                    niplsnatxi(i,j) = cm*natxi(i,j)
                    if (i .eq. j) niplsnatxi(i,j) = niplsnatxi(i,j)+1
    5            continue
    6         continue
              call matrix(niplsnatxi,accel,acceln,ndim,ndim,1,3,3,1)
              do 7 n = 1,ndim
                 q(m,n) = q(m,n)+ruf*
     1                    0.25*pi*del(jeltyp(nel),m)*del(jeltyp(nel),m)*
     2                    acceln(n)
    7         continue
           endif
c
c       2) drag term: 0.5*ruf*df*cd*(n*(vf-r dot)||n*(vf-r dot)|
c
           call matrix(natxi,rvf,rvn,ndim,ndim,1,3,3,1)
           rvnsqr = 0.0d0
           do 8 n = 1,ndim
              rvnsqr = rvnsqr+rvn(n)*rvn(n)
    8      continue

           do 9 n = 1,ndim
              q(m,n) = q(m,n)+0.5*ruf*del(jeltyp(nel),m)*cd*rvn(n)
     1                 *sqrt(rvnsqr)
    9      continue
c
c       3) gravity and buoyancy: [B-g*(Rui*Ai+Rut*At)]ey
c
           wb = g*(mel(jeltyp(nel),m)+rui*aiel(jeltyp(nel),m))-
     1          buoyel(jeltyp(nel),m)
           q(m,2) = q(m,2) - wb
           if (dabs(dfel(jeltyp(nel),m)) .le. 1.0d-5) then
              pause 'Distance to the outmost fiber is zero. STOP!!!'
              stop
           endif
           wbd(m) = wb/dfel(jeltyp(nel),m)

   10   continue
c
c       4) elastic sea floor: uniform distributed spring system
c          wbd*dis
c
        ndis = 0
        do 11 m = 1,3
           dis = u(m,2)
           if (m .eq. 2) then
              dis = 0.5*u(1,2)+0.125*u(2,2)+0.5*u(3,2)-0.125*u(4,2)
           endif
           if (dis-btmdpth .le. dfel(jeltyp(nel),m)) ndis = ndis+1
   11   continue

        xa1 = 0.0d0
        xb1 = 0.0d0
        xa2 = 0.0d0
        xb2 = 0.0d0
        nroot = 0
        do 12 m = 1,3
           xroot(m) = 0.0d0
   12   continue
c
c       No portion of the element below sea bottom line: ndis = 0
c       Full element below sea bottom line: ndis = 3
c       Some portions of the element below sea bottom line:ndis = 1 or 2
c
        if (ndis .eq. 0) return
        if (ndis .eq. 3) then
           xa1 = 0.0d0
           xb1 = 1.0d0
           return
        endif
        y0 = btmdpth+dfel(jeltyp(nel),2)
        if (ndis .gt. 0 .and. ndis .lt. 3) call root(u(1,2),u(2,2),
     1     u(3,2),u(4,2),y0,0.0d0,1.0d0,xroot,nroot)
c
c       hit bottom once
c
        if (nroot .eq. 1) then
           if (u(1,2) .lt. y0) then
              xa1 = 0.0d0
              xb1 = xroot(1)
           else
              xa1 = xroot(1)
              xb1 = 1.0d0
           endif
        endif
c
c       hit bottom twice
c
        if (nroot .eq. 2) then
           if (u(1,2) .lt. y0) then
              xa1 = 0.0d0
              xb1 = xroot(1)
              xa2 = xroot(2)
              xb2 = 1.0d0
           else
              xa1 = xroot(1)
              xb1 = xroot(2)
           endif
        endif
c
c       hit bottom triple times
c
        if (nroot .eq. 3) then
           if (u(1,2) .gt. y0) then
              xa1 = xroot(1)
              xb1 = xroot(2)
              xa2 = xroot(3)
              xb2 = 1.0d0
```

```
      else
         xa1 = 0.0d0
         xb1 = xroot(1)
         xa2 = xroot(2)
         xb2 = xroot(3)
      endif
   endif

   return
   end



   SUBROUTINE root(u1,u2,u3,u4,y0,xa,xb,xroot,nroot)
c
c     Routine ROOT evaluates location at which an element crossing
c     ocean bottom line. The routine is to solve the linear
c     polynomial equations of order 3:
c
c     u1 a[1]+u2 a[2]+u3 a[3]+u4 a[4] = y0
c     or a x^3 + b x^2 + c x + d = 0
c     where:
c            a = 2u1+u2-2u3+u4
c            b = -3u1-2u2+3u3-u4
c            c = u2
c            d = u1-y0
c
c     Bisection method is used in the routine to obtain the first zero
c     of the equation. The other two are gotten analytically. Only
c     roots in the interval [xa,xb] are kept.
c
c     Global Variables:
c        noort, xroot  -
c                  # of crossing points and location of the points in
c                  the specified interval [xa,xb]
c        u1, u2, u3, u4   -
c                  parameters describing the shape of the element
c        xa,xb  -  interval for which crossing points are searched
c        y0     -  simulated ocean depth
c
c     Subroutine called by QFORCE
c


      implicit double precision (a-f,o-z)
      dimension xroot(3)

      a = 2*u1+u2-2*u3+u4
      b = -3*u1-2*u2+3*u3-u4
      c = u2
      d = u1-y0
      xstep = 1.0d-2
c
c     Special case: a = 0 and/or b = 0
c
      if (d .eq. 0) d = 1.0d0-7
      if (a .eq. 0) then
         if (b .eq. 0) then
            x1 = -d/c
            nroot = 1
            xroot(1) = x1
            return
         else
            xsqrt = c*c-4*b*d
            if (xsqrt .eq. 0) then
               x1 = -c/(2*b)
               nroot = 1
               xroot(1) = x1
               return
            endif
            x1 = (-c+sqrt(xsqrt))/(2*b)
            x2 = (-c-sqrt(xsqrt))/(2*b)
            goto 5
         endif
      endif
c
c     Find the first zero of the equation: First, find the interval
c     where there is a zero. Then use Bisection method to get the
c     zero.
c
      x1 = xa
    1 y1 = a*x1*x1*x1+b*x1*x1+c*x1+d
      if (abs(y1) .lt. 1.0d-10) then
         xroot(1) = x1
         nroot = 1
         goto 4
      endif

    2 x2 = x1+xstep
      if (x2 .gt. xb) return
      y2 = a*x2*x2*x2+b*x2*x2+c*x2+d
      if (abs(y2) .lt. 1.0d-10) then
         xroot(1) = x2
         nroot = 1
         goto 4
      endif

      if (y1*y2 .lt. 0) goto 3
      x1 = x2
      y1 = y2
      goto 2

    3 x = (x1+x2)/2
      y = a*x*x*x+b*x*x+c*x+d
      if (abs(y) .lt. 1.0d-10) then
         xroot(1) = x
         nroot = 1
         goto 4
      endif

      if (y1*y .le. 0) then
         x2 = x
         y2 = y
         goto 3
      else
         x1 = x
         y1 = y
         goto 3
      endif
c
c     Find the other two roots and keep any root(s) in the interval
c     [xa,xb]
c
    4 xsqrt = (b/a+xroot(1))*(b/a+xroot(1))+4*d/(a*xroot(1))
      if (xsqrt .lt. 0) return
      if (xsqrt .eq. 0) then
         x = -(b/a+xroot(1))/2
         if ((x. ge.xa) .and. (x .le. xb)) then
```

```
            nroot = 2
            xroot(2) = x
         endif
         return
      endif
      x1 = (-(b/a+xroot(1))+sqrt(xsqrt))/2
      x2 = (-(b/a+xroot(1))-sqrt(xsqrt))/2

5     if (x1 .gt. x2) then
         x = x1
         x1 = x2
         x2 = x
      endif
      if ((x1 .ge. xa) .and. (x1 .le. xb)) then
         nroot = nroot+1
         xroot(nroot) = x1
      endif
      if ((x2 .ge. xa) .and. (x2 .le. xb)) then
         nroot = nroot+1
         xroot(nroot) = x2
      endif

      return
      end


      SUBROUTINE slveq16
c
c     Routine SLVEQ16 evaluates r,rprm,rdot,rpmdt, and lambda
c     for both static and dynamic problems.
c
c     For the static problem, initial guess of r,rprm,rdot,rpmdt,
c     and lambda are given. After several steps of iteration, the r
c     rprm, rdot, rpmdt, and lambda of an static equilibrium
c     status are calculated. The r, rprm, rdot, rpmdt, and lambda
c     for the dynamic problem are given at each time step.
c
c     Variables   -
c        af       -  acceleration of fluid
c        big      -  the biggest error (or increment)
c        dr       -  increment of r
c        drdot    -  increment of r-dot
c        drprm    -  increment of r'
c        drpmdt   -  increment of r'-dot
c        dt       -  increment of tension
c        eps      -  accuracy of iteration. Iteration exits if
c                    convergence reachs, or the iteration continues
c                    until a preset iteration number (iter) is
c                    reached.
c        f        -  right-hand side vector
c        h        -  time step, = tstep
c        isnapcnt -  snapshot count
c        job      -  1 = static, 2 = dynamic
c        jsnap    -  snapshot at each jsnap iteration
c        nd       -  freedom # of the element
c        ndim     -  # of dimensions, 2 or 3
c        neq      -  # of equations in the assembled global equation
c                    set
c        node     -  # of nodes
c        pi       -  pressure due to inner fluid
c        phit     -  partial derivative of velocity potential
c                    with respective to time t
c        pf       -  pressure due to outside fluid field
c        lmda05   -  lambda at time t=(n+1/2)
c        r        -  position of nodes, i.e. x, y, and z
c        rdot     -  time derivative of r, i.e. x-dot, y-dot,
c                    and z-dot
c        normal   -  transfer matrix
c        rpmdt    -  time derivative of the slope, i.e. x'-dot,
c                    y'-dot, and z'-dot,
c        rprm     -  slope at nodes, i.e. x', y', and z'
c        s        -  coefficient matrix
c        t        -  time
c        tbgn,tend,tstep  -
c                    begin time, end time & time step
c        lambda   -  effective tension at nodes
c        lmdap    -  lambda at previous time step
c        tx,ty,tz -  top end tension in x, y, and z direction
c        u,lmda   -  coefficients determined by
c                    r(s,t) = u[xn]*a[k]
c                    lambda(s,t) = lmda[m]*p[m]
c                    where, u and lmda is a function of t, and
c                    a and p is a function of s.
c        u05      -  u at time t=(n+1/2)
c        vf       -  velocity of fluid
c        wbd      -  (net weight of the element in water)
c                    ------------------------------------, wbd[m]
c                          (dist to out most fiber)
c        (xa1,xb1), (xa2,xb2)  -
c                    range of the 1st and 2nd under bottom portions
c                    of the element
c
c     Subroutine called by the main routine
c
      implicit double precision (a-h,o-z)
      include 'parametrics.inc'
      include 'variables.inc'
      dimension af(3,3),f(16),pf(3),pi(3),s(16,16),vf(3,3)
      dimension acmp(400,31),al(400,15),indx(400)
      double precision normal(3,3,3)
      parameter(eps=1.0d-10)
c
c     define some constants
c
      nd  = 4*ndim+4
      neq = nd*node/2
      m1  = nd-1
      m2  = nd-1
      np  = 400
      mp  = 31
      mp1 = 15
      h   = tstep
c
c     set initial data
c
      if (job .eq. 2) then
         t = tbgn
         iter = (tend-tbgn)/h+1
         do 2 n = 1,node
            do 1 m = 1,ndim
               rdotp(n,m)  = rdot(n,m)
               rpmdtp(n,m) = rpmdt(n,m)
1           continue
            lmdap(n,1) = lambda(n,1)
            lmdap(n,2) = lambda(n,2)
2        continue
         lmdap(1,1) = 0.0d0
      endif
```

```
      lsnapcnt = 1
      do 11 i = 1,iter

         do 6 n = 1,neq
            b(n) = 0.0d0
            indx(n) = 0
            do 3 m = 1,neq
               a(n,m) = 0.0d0
 3          continue
            do 4 m = 1,mp
               acmp(n,m) = 0.0d0
 4          continue
            do 5 m = 1,mpl
               al(n,m) = 0.0d0
 5          continue
 6       continue
c
c     assemble gloabal matrix a(left-hand side matrix)
c
         do 8 n=1,node-1
c
c     assign u(kn), udot(kn), & lmda
c
            do 7 m = 1,ndim

               u(1,m) = r(n,m)
               u(2,m) = rprm(n,m)*lnthel(n)
               u(3,m) = r(n+1,m)
               u(4,m) = rprm(n+1,m)*lnthel(n)

               if (job .eq. 2) then
                  udot(1,m) = rdot(n,m)
                  udot(2,m) = rpmdt(n,m)*lnthel(n)
                  udot(3,m) = rdot(n+1,m)
                  udot(4,m) = rpmdt(n+1,m)*lnthel(n)

                  uddot(1,m)=(rdot(n,m)-rdotp(n,m))/h
                  uddot(2,m)=(rpmdt(n,m)-rpmdtp(n,m))/h*lnthel(n)
                  uddot(3,m)=(rdot(n+1,m)-rdotp(n+1,m))/h
                  uddot(4,m)=(rpmdt(n+1,m)-rpmdtp(n+1,m))/h*lnthel(n)

                  u05(1,m) = u(1,m)+h*udot(1,m)/2
                  u05(2,m) = u(2,m)+h*udot(2,m)/2
                  u05(3,m) = u(3,m)+h*udot(3,m)/2
                  u05(4,m) = u(4,m)+h*udot(4,m)/2
               endif

 7          continue
            lmda(1) = lambda(n,2)
            lmda(2) = lambda(n+1,1)
            lmda(3) = lambda(n+1,2)
            if (job .eq. 2) then
               lmda05(1) =(  3*lambda(n,2)-lmdap(n,2)  )/2
               lmda05(2) =(3*lambda(n+1,1)-lmdap(n+1,1))/2
               lmda05(3) =(3*lambda(n+1,2)-lmdap(n+1,2))/2
            endif
c
c     evaluate velocity vf, acceleration af, pressure pf, transfer
c     matrix normal, mass matrix rm, stretch e
c
            call velocity(vf,af,phit,t)
            call pressure(pf,pi,vf,phit)
            call tnormal(n,normal)
            call qforce(n,normal,vf,af,xa1,xb1,xa2,xb2)
            call epsilon(n,pf,pi)
c
c     form element stiffness matrix s and right-hand side vector f
c     assemble global matrix a and global right-hand side vector b
c
            if (job .eq. 1) call steq16(s,f,n,xa1,xb1,xa2,xb2)
            if (job .eq. 2) then
               call mass(n,normal)
               call dyeq16(s,f,n,xa1,xb1,xa2,xb2)
            endif
            if (n .eq. node-1) s(nd,nd)=1.0d0

            call assemb(s,f,n,nd)

 8       continue
c
c     enforce boundary conditions
c
         call bondcns(nd,t)
c
c     slove unkowns
c
         call bandcmp(a,acmp,neq,m1,m2,np,mp)
         call bandec(acmp,neq,m1,m2,np,mp,al,mpl,indx,d)
         call bandks(acmp,neq,m1,m2,np,mp,al,mpl,indx,b)
c
c     backsubstitute solution
c
         big = 0.0d0
         do 10 n = 1,node
            k = (n-1)*nd/2
            do 9 m = 1,ndim

               if (job .eq. 1) then
                  dr = b(k+m)
                  if (abs(dr) .gt. abs(big)) big = dr
                  r(n,m) = r(n,m)+relax*dr
                  drprm = b(k+ndim+m)
                  if (abs(drprm) .gt. abs(big)) big = drprm
                  rprm(n,m) = rprm(n,m)+relax*drprm
               endif

               if (job .eq. 2) then
                  drdot = h*b(k+m)
                  dr = h/2*(drdot+2*rdot(n,m))
                  if (abs(dr) .gt. abs(big)) big = dr
                  r(n,m) = r(n,m)+dr
                  rdotp(n,m) = rdot(n,m)
                  rdot(n,m) = rdot(n,m)+drdot
c                 r(n,m) = r(n,m)+(rdot(n,m)+rdotp(n,m))*h/2

                  drpmdt = h*b(k+ndim+m)
                  drprm = h/2*(drpmdt+2*rpmdt(n,m))
                  if (abs(drprm) .gt. abs(big)) big = drprm
                  rprm(n,m) = rprm(n,m)+drprm
                  rpmdtp(n,m) = rpmdt(n,m)
                  rpmdt(n,m) = rpmdt(n,m)+drpmdt
c                 rprm(n,m) = rprm(n,m)+(rpmdt(n,m)+rpmdtp(n,m))*h/2
               endif

 9          continue

            if (n .eq. 1) then
```

```
                lambda(n,1) = 0.0d0
                lmdap(n,1) = 0.0d0
                if (job .eq. 1) then
                   dt = b(2*ndim+1)
                   lambda(n,2) = lambda(n,2)+relax*dt
                endif
                if (job .eq. 2) then
                   dt = h*h/2*b(2*ndim+1)
                   lmdap(n,2)  = lambda(n,2)
                   lambda(n,2) = lambda(n,2)+dt
                endif
             else
                if (job .eq. 1) then
                   dt = b(k)
                   lambda(n,1) = lambda(n,1)+relax*dt
                   dt = b(k+2*ndim+1)
                   lambda(n,2) = lambda(n,2)+relax*dt
                endif
                if (job .eq. 2) then
                   dt = h*h/2*b(k)
                   lmdap(n,1)  = lambda(n,1)
                   lambda(n,1) = lambda(n,1)+dt
                   dt = h*h/2*b(k+2*ndim+1)
                   lmdap(n,2)  = lambda(n,2)
                   lambda(n,2) = lambda(n,2)+dt
                endif
             endif
  10      continue
c
c       write top end tension
c
             if (job .eq. 1) then
                tx = lambda(1,2)*rprm(1,1)
                ty = lambda(1,2)*rprm(1,2)
                tz = lambda(1,2)*rprm(1,3)
                write(iscr,1010) i,big,tx,ty,tz
                write(isnap,1010) i,big,tx,ty,tz
             endif
c
c       snapshot at every jsnap
c
             if (i/jsnap .eq. isnapcnt) then
                isnapcnt = isnapcnt+1
                call snapshot(i,t)
             endif

             if ((job .eq. 1) .and. (abs(big) .lt. eps)) return
             if (job.eq.2) then
                write(iscr,1000) t,big
                write(itimhis,1030) t,(r(i,m),   m = 1,ndim),
     1                                (rprm(i,m),m = 1,ndim),lambda(i,2)
                t = t+h
             endif

  11   continue

       return
1000   format(1x,'t= ',f8.3,2x,'inf norm =',1pe10.3)
1010   format(/1x,'Iter. #= ',i4,2x,'Infinite Norm= ',1pe10.3/
     1 1x,'1st end tension in x, y, ans z = ',3(1pe12.5,1x))
1020   format(f8.3,7(1pe12.5))
1030   format(8(1pe12.5))
       end
```

```
       SUBROUTINE snapshot(i,t)
c
c      Routine SNAPSHOT outputs the intermediate results
c      at the ith iteration. The frequency of writing is
c      by the jsnap.
c
c      Variables   -
c         i        -  iteration #i
c         t        -  time
c
c      Subroutine called by SLVEQ16
c
       implicit double precision (a-h,o-z)
       include 'parametrics.inc'
       include 'variables.inc'

       write(isnap,1000) i,t
       if (ndim .eq. 2) write(isnap,1010)
       if (ndim .eq. 3) write(isnap,1020)

       do i n = 1,node
          write(isnap,1030) n,lambda(n,2), (r(n,m),    m = 1,ndim),
     1                                     (rprm(n,m),m = 1,ndim)
     1  continue

       return
1000   format(/'Snapshot at iteration #    ',i4,' (time = ',f10.3,' s)'
     1        /'Coordinates:')
1010   format(1x,4hnode,4x,7hTension,9x,1hx,11x,1hy,11x,2hx',10x,2hy')
1020   format(1x,4hnode,4x,7hTension,9x,1hx,11x,1hy,11x,1hz,11x,2hx',
     1        10x,2hy',10x,2hz')
1030   format(1x,i4,1x,1f12.2,1x,6(f11.5,1x))
       end
```

```
       SUBROUTINE steq16(s,f,nel,xa1,xb1,xa2,xb2)
c
c      ***************
c      static analysis
c      ***************
c
c      Vectorial motion equations for the cable statics are
c      decomposed into 16 scalar equations with the form S*dU=F.
c      Routine STEQ16 forms the element matrix S and the right-hand
c      side vector F, where S is a 16 by 16 (or 12 by 12) coefficient
c      matrix, and F is a 16 by 1 (or 12 by 1) vector in 3-D (or 2-D)
c      space.
c
c      Variables   -
c         at       -  cross-section area of the tube(element),
c                     at=afel-alel
c         dU       -  unknown vector arranged in the following way:
c
c                  3 dimensions  -          2 dimensions -
c              {dU} = |   du[11]   |  {dU} = |   du[11]   |
c                     |   du[12]   |         |   du[12]   |
c                     |   du[13]   |         |   du[21]   |
c                     |   du[21]   |         |   du[22]   |
c                     |   du[22]   |         | dlambda[1] |
```

```
c                       |   du[23]    |       =  | dlambda[2] |
c                       | dlambda[1]  |       =  |   du[31]   |
c                       | dlambda[2]  |       =  |   du[32]   |
c                       |   du[31]    |       =  |   du[41]   |
c                       |   du[32]    |       =  |   du[42]   |
c                       |   du[33]    |       =  | dlambda[3] |
c                       |   du[41]    |       =  | dlambda[1] |
c                       |   du[42]    |
c                       |   du[43]    |
c                       | dlambda[3]  |
c                       | dlambda[1]  |
c          et       -  Young's modulus of the tube(element),
c                      et = eel(jeltyp(nel))
c          f        -  element right-hand side vector
c          lt       -  length of the tube(element), lt = lnthel(nel)
c          mu1,mu2,gamma1,gamma2  -
c                      shape coefficient mu and gamma in the interval
c                      [xa1,xb1], and [xa2,xb2], respectively.
c          nel      -  element #
c          s        -  element coefficient matrix
c          wbd      -  (net weight of the element in water)
c                      ----------------------------------, wbd(m)
c                           (dist to out most fiber)
c          [xa1,xb1], [xa2,xb2]  -
c                      range of the 1st and 2nd under bottom portions
c                      of the element
c
c       Subroutine called by SLVEQ16
c
        implicit double precision (a-h,o-z)
        include 'parametrics.inc'
        include 'variables.inc'
        dimension at(3),f(16),s(16,16),gamma1(4,4,3),gamma2(4,4,3)
        double precision lt,mu1(4,3),mu2(4,3)

        do 2 ii=1,16
           do 1 nn=1,16
              s(ii,nn)=0.0d0
    1     continue
          f(ii)=0.0d0
    2   continue
c
c       at=af-al
c       et=eel(jeltyp(nel))
c       lt=lnthel(nel)
c
        do 3 m = 1,3
           at(m) = afel(jeltyp(nel),m)-alel(jeltyp(nel),m)
    3   continue
        et = eel(jeltyp(nel))
        lt = lnthel(nel)
c
c       compute mu and gamma in the interval defined by [xa1,xb1],
c       and [xa2,xb2]
c
        call mubtm(xa1,xb1,mu1)
        call mubtm(xa2,xb2,mu2)
        call gammabtm(xa1,xb1,gamma1)
        call gammabtm(xa2,xb2,gamma2)
c
c ********************************
c       stiffness matrix s
c ********************************
c
        do 9 l = 1,4
           do 8 n = 1,ndim

              ii = ndim*(i-1)+n
              if (i .ge. 3) ii = ii+2
c
c       beta(ikm)*lmda(m)+alpha(ikm)*b(m)
c       gamma(ikm)*wbd(m)  (bottom support: in y direction)
c
              do 5 k = 1,4
                 nn = ndim*(k-1)+n
                 if (k .ge. 3) nn = nn+2
                 do 4 m = 1,3
                    s(ii,nn) = s(ii,nn)+
     1                          (beta(i,k,m)/lt*lmda(m)+
     2                          alpha(i,k,m)/lt**3*et*mol(jeltyp(nel),m))
                    if (n .eq. 2)
     1                 s(ii,nn) = s(ii,nn)+(gammal(i,k,m)+gamma2(i,k,m))
     2                                    *lt*wbd(m)
    4            continue
    5           continue
c
c       beta(ikm)*u(kn)
c
              do 7 m = 1,3
                 nn = 2*ndim+m
                 if (m .eq. 3) nn = nn+2*ndim
                 do 6 k=1,4
                    s(ii,nn) = s(ii,nn)+beta(i,k,m)/lt*u(k,n)
    6             continue
    7           continue

    8        continue
    9     continue
c
c       beta(ikm)*u(in)
c       ruf*q*eta/(At*E)
c       -eta(lm)/(At*E)
c
        do 14 m = 1,3
           ll = 2*ndim+m
           if (m.eq.3) ll = ll+2*ndim

           do 12 k = 1,4
              do 11 n = 1,ndim
                 nn = ndim*(k-1)+n
                 if (k .ge. 3) nn = nn+2
                 do 10 i = 1,4
                    s(ii,nn) = s(ii,nn)+beta(i,k,m)/lt*u(i,n)
   10            continue

                 if (n .eq. 2) then
                    if (k.eq.1) s(ii,nn) = s(ii,nn)+ruf*q*lt*
     1                          (eta(1,m)/at(1)+0.5*eta(2,m)/at(2))/et
                    if (k.eq.2) s(ii,nn) = s(ii,nn)+
     1                          0.125*ruf*q*eta(2,m)*lt/(et*at(2))
                    if (k.eq.3) s(ii,nn) = s(ii,nn)+ruf*q*lt*
     1                          (eta(3,m)/at(3)+0.5*eta(2,m)/at(2))/et
                    if (k.eq.4) s(ii,nn) = s(ii,nn)-
     1                          0.125*ruf*q*eta(2,m)*lt/(et*at(2))
                 endif

   11         continue
   12        continue
```

```
            do 13 l = 1,3
               nn = 2*ndim+l
               if (l .eq. 3) nn = nn+2*ndim
               s(ii,nn) = s(ii,nn)-eta(l,m)*lt/(et*at(l))
   13       continue

   14    continue
c
c ***********************************
c       right-hand side vector f
c ***********************************
c
c       mu(im)*q(mn)
c       mu(im)*wbd(m)*(D(m)+btmdpth)  (bottom support: in y direction)
c
        do 17 i = 1,4
           do 16 n = 1,ndim
              ii = ndim*(i-1)+n
              if (i .ge. 3) ii = ii+2
              do 15 m = 1,3
                 f(ii) = f(ii)+mu(i,m)*lt*q(m,n)
                 if (n .eq. 2)
     1              f(ii) = f(ii)+(mu1(i,m)+mu2(i,m))*lt*wbd(m)*
     2                      (dfel(jeltyp(nel),m)+btmdpth)
   15         continue
   16       continue
   17    continue
c
c       beta(ikm)*lmda(m)*u(kn)
c       alpha(ikm)*b(m)*u(kn)
c       -gamma(ikm)*wbd(m)*u(k2)  (bottom support: in y direction)
c
        do 21 i = 1,4
           do 20 n = 1,ndim
              ii = ndim*(i-1)+n
              if (i .ge. 3) ii = ii+2
              do 19 k = 1,4
                 do 18 m = 1,3
                   f(ii) = f(ii)-beta(i,k,m)/lt*lmda(m)*u(k,n)
                   f(ii) = f(ii)-alpha(i,k,m)/lt**3*
     1                           et*moi(jeltyp(nel),m)*u(k,n)
                   if (n .eq. 2)
     1                f(ii) = f(ii)-(gamma1(i,k,m)+gamma2(i,k,m))*lt*
     2                        wbd(m)*u(k,2)
   18            continue
   19         continue
   20       continue
   21    continue
c
c       0.5*tau(m)
c
        do 22 m = 1,3
           ii = 2*ndim+m
           if (m .eq. 3) ii = ii+2*ndim
           f(ii) = f(ii)+0.5*tau(m)*lt
   22    continue
c
c       eta(lm)*e(l)
c
        do 24 m = 1,3
           ii = 2*ndim+m
           if (m .eq. 3) ii = ii+2*ndim
           do 23 l = 1,3
              f(ii) = f(ii)+eta(l,m)*lt*e(l)
   23       continue
   24    continue
c
c       -0.5*beta(ikm)*u(in)*u(kn)
c
        do 28 m = 1,3
           ii = 2*ndim+m
           if (m .eq. 3) ii = ii+2*ndim
           do 27 i = 1,4
              do 26 k = 1,4
                 do 25 n = 1,ndim
                    f(ii) = f(ii)-0.5*beta(i,k,m)/lt*u(i,n)*u(k,n)
   25            continue
   26         continue
   27       continue
   28    continue

        do 30 ii = 1,4*ndim+4
           do 29 nn = 1,4*ndim+4
              if ((nn .ge. ndim+1)   .and. (nn .le. 2*ndim) .or.
     1            (nn .ge. 3*ndim+3) .and. (nn .le. 4*ndim+2))
     2         s(ii,nn) = s(ii,nn)*lt
   29       continue
30      continue

        return
        end




        SUBROUTINE tnormal(nel,normal)
c
c       Rotuine TNORMAL provides transfer matrix NORMAL which yields the
c       normal component of ossicilation.
c
c       Variables   -
c          index    -  = 1, cross-section at xi = 0
c                      = 2, cross-section at xi = 0.5
c                      = 3, cross-section at xi = 1
c          nel      -  element #
c          normal   -  transfer matrix,  = i - r'*r'
c
c       Subroutine called by SLVEQ16
c
        implicit double precision (a-h,o-z)
        include 'parametrics.inc'
        include 'variables.inc'
        double precision normal(3,3,3)

        do 5 index = 1,3
           do 4 m = 1,ndim
              do 3 n = 1,ndim
                 normal(m,n,index) = 0.0d0
                 do 2 i = 1,4
                    do 1 k = 1,4
                       normal(m,n,index) = normal(m,n,index)+
     1                                     omega(i,k,index)*
     1                                     u(i,m)*u(k,n)/lnthel(nel)**2
    1                  continue
    2               continue
                    normal(m,n,index) = -normal(m,n,index)
                    if (m .eq. n) normal(m,n,index) = 1+normal(m,n,index)
```

```
    3         continue
    4       continue
    5     continue

          return
          end



          SUBROUTINE velocity(vf,af,phit,t)
c
c         Routine VELOCITY evaluates current and wave velocities.
c
c         Variables  -
c            af       -  acceleration of fluid due to waves
c            ax,ay,az    -
c                        components of water accelerations in x, y, and
c                        z directions
c            aw       -  wave amplitude
c            dirw     -  wave direction relative to x axis
c            ndim     -  dimensions
c            nvc      -  # of evaluations at which the velocities
c                        of current are given.
c            phit     -  partial derivative of velocity potential
c                        with respective to t
c            sigmaw   -  circular wave frequency (rad/sec)
c            t        -  time
c            ux,uy,uz    -
c                        components of water velocities in x, y , and z
c                        directions due to waves
c            vc       -  velocity of current
c            vcc      -  component of current velocity in x, y, or z
c                        directions
c            vf       -  velocity of fluid due to current and waves at
c                        cross-section xi = 0, 0.5 & 1 of the element
c            x,y,z    -  position of the cross-sections
c            yc       -  water evaluations
c
c         Subroutine called by SLVEQ16
c
          implicit double precision (a-h,o-z)
          include 'parametrics.inc'
          include 'variables.inc'
          dimension af(3,3),vf(3,3),vcc(10),yc(10)

          z    = 0.0d0
          phit = 0.0d0
          do 3 m = 1,3
c
c         Position of the cross-sections
c
             x = u(m,1)
             y = u(m,2)
             if (ndim .eq. 3) z = u(m,3)
             if (m .eq. 2) then
                x = 0.5*u(1,1)+0.125*u(2,1)+0.5*u(3,1)-0.125*u(4,1)
                y = 0.5*u(1,2)+0.125*u(2,2)+0.5*u(3,2)-0.125*u(4,2)
                if (ndim .eq. 3) z = 0.5*u(1,3)+0.125*u(2,3)+
     1                               0.5*u(3,3)-0.125*u(4,3)
             endif

             do 2 n = 1,ndim
c
c            1) velocity due to currents
c
                   do 1 k = 1,nvc
                      yc(k) = vc(k,1)
                      vcc(k) = vc(k,n+1)
     1             continue
                   call polint(yc,vcc,nvc,y,vf(m,n),dy)
     2          continue
c
c            2) velocity due to waves
c            velocity potential:
c            phi = aw*g*Exp(k*y)*Sin(k(x*Cos(dirw)+z*Sin(dirw))-sigmaw*t)
c                  ---------------------------------------------------
c                                       sigmaw
c            where:
c            i)  k     -  wave number, k = waveno = sigmaw^2/g
c            ii) g     -  gravity acceleration
c
                if (job .eq. 2) then
                   waveno = sigmaw*sigmaw/g
                   angle = waveno*(x*cos(dirw)+z*sin(dirw))-sigmaw*t
                   phit = -aw*g*exp(waveno*y)*cos(angle)
c
c            velocity of fluid
c
                   ux = aw*sigmaw*exp(waveno*y)*cos(dirw)*cos(angle)
                   uy = aw*sigmaw*exp(waveno*y)*sin(angle)
                   if (ndim .eq. 3)
     1             uz = aw*sigmaw*exp(waveno*y)*sin(dirw)*cos(angle)

                   vf(m,1) = vf(m,1)+ux
                   vf(m,2) = vf(m,2)+uy
                   if (ndim .eq. 3) vf(m,3) = vf(m,3)+uz
c
c            acceleration of fluid
c
                   af(m,1) = aw*exp(waveno*y)*waveno*cos(dirw)*
     1                       (g*sin(angle)-ux*sigmaw*cos(dirw)*sin(angle)
     2                                    +uy*sigmaw*cos(angle)
     3                                    -uz*sigmaw*sin(dirw)*sin(angle))
                   af(m,2) = aw*exp(waveno*y)*waveno*
     1                       (-g*cos(angle)+ux*sigmaw*cos(dirw)*cos(angle)
     2                                     +uy*sigmaw*sin(angle)
     3                                     +uz*sigmaw*sin(dirw)*cos(angle))
                   if (ndim .eq. 3)
     1             af(m,3) = aw*exp(waveno*y)*waveno*sin(dirw)*
     2                       (g*sin(angle)-ux*sigmaw*cos(dirw)*sin(angle)
     3                                    +uy*sigmaw*cos(angle)
     4                                    -uz*sigmaw*sin(dirw)*sin(angle))
                endif

     3       continue

             return
             end
```