

EFFECTS OF OFFSHORE ENVIRONMENT ON DYNAMIC RESPONSE OF PILE FOUNDATIONS

**Final Report Submitted to
Minerals Management Service, Department of Interior**

Toyoaki Nogami

**Scripps Institution of Oceanography
University of California at San Diego
La Jolla, CA 92093**

and

Motoki Kazama

**Port and Harbor Research Institute
Japan Ministry of Transport
Yokosuka, Japan**

TABLE OF CONTENTS

	Page
1. INTRODUCTION	1
1.1 Introduction	1
1.2 Methods for Dynamic Response Analysis of Fluid-Saturated Porous Medium	2
2. GOVERNING DIFFERENTIAL EQUATIONS	4
3. SOLUTION BY THIN LAYER ELEMENT METHOD	11
3.1 Plane strain Conditions with Cartesian Coordinate System	11
3.2 Three-dimensional Conditions with cylindrical coordinate System	17
4. COMPUTED RESULTS AND REMARKS	26
4.1 Conditions Considered for Soil Deposits	26
4.2 Plane Strain Conditions in Cartesian Coordinate System	27
4.3 Three-Dimensional Conditions in Cylindrical Coordinate System	30
5. CONCLUSIONS	34
REFERENCES	36
APPENDIX A	Listing of Computer Program "PST"
APPENDIX B	Listing of Computer Program "AXS"

1. INTRODUCTION

1.1 Introduction

Soil deposits in the offshore environment are located at some depth in the water and are saturated with fluid. The dynamic load applied to submerged soil generates the transient pore-fluid motion relative to the skeleton for the transient redistribution of pore fluid. This occurs in various degrees depending on loading rate, soil permeability, pressure gradient and boundary conditions. The dynamic response behavior of submerged soil is affected by such pore fluid behavior and thus is extremely complex. The water body above the soil deposits also may affect the dynamic response of soil. At the present time, however, it is not well understood how and how significantly those factors associated with the offshore environment affect the dynamic response of soil deposits and offshore pile foundations.

The objectives of the research are 1) to develop an efficient numericam method, the thin layer element solution, for the dynamic response analysis of saturated soil deposits and 2) to examine the effects of the offshore environment on the dynamic response of soil deposits and pile foundations. This report contains 1) formulation of the thin layer element solution developed for the dynamic response analysis of saturated soil deposits, 2) numerical results computed by the developed formulation, 3) and discussions of the offshore environment effects on the dynamic responses of soil deposits and pile foundations. Both the two-dimensional plane strain conditions and the three-dimensional conditions in the cylindrical coordinates are considered in the study.

The results reported herein are a part of the study to develop a rational numerical model of pile foundations for seismic response analysis of pile-supported offshore

structures. The study is being conducted with the collaboration from the Port and Harbor Research Institute, Japan Ministry of Transport.

1.2 Methods for Dynamic Response Analysis of Fluid-Filled Porous Medium

Analytical studies are useful to understand the complex dynamic response behavior of saturated soil. Biot (1962) has made a framework in the formulation of dynamic response of a fluid-filled elasto-porous medium . This formulation has been generally used for dynamic response analysis of submerged soil and evaluated typically by either analytical solutions obtained by solving the differential equations or the numerical finite element method. Considerable difficulty exists in obtaining analytical solutions of Biot's equation in general and thus the solutions have been developed only for very simple conditions (e.g. Biot, 1956; Jones, 1961; Deresiewics, 1960; Foda and Mei, 1982). Those conditions are generally far from those commonly encountered in the real situation. The finite element method has been applied for the numerical evaluation of Biot's equation (e.g. Ghaboussi and Wilson, 1973; Prevost, 1982; Simon et al., 1986; Zienkiewicz et al., 1977). Contrary to the former approach, this approach can account for complex geometry and inhomogeneity without increasing the degree of difficulty and amount of computation. However, compared with the finite element scheme applied to a single-phase medium, the computation effort increases substantially due to the additional degrees of freedom associated with pore fluid. The thin layered element method has been developed by combining the finite element method and analytical solution, to compute the responses by using Rayleigh wave modes (Kausel and Roessel, 1975; Lysmer and Waas, 1972; Tajimi and Shimomura, 1976). This approach requires a computation effort far less than the regular finite element approach and yet has a capacity of accommodating complex conditions far more than the analytical solution approach. Since a large computation effort

is generally required in the dynamic response analysis of a two-phase mixture by the regular finite element method, this approach appears to be very attractive for such analysis. However, the thin layer element method does not appear to have been applied to a fluid-saturated porous medium in the past.

2. GOVERNING DIFFERENTIAL EQUATIONS

Cylindrical and Cartesian coordinates are considered as shown in Fig. 2.1. Stress and strain vectors and force and displacement vectors are arranged as follows:

$$\text{stress and strain vectors} \quad (\sigma_{xx}, \sigma_{zz}, \sigma_{xz}) \quad (\epsilon_{rr}, \epsilon_{\theta\theta}, \epsilon_{zz}, \epsilon_{rz}, \epsilon_{r\theta}, \epsilon_{z\theta})$$

$$\text{force and displacement vectors} \quad (f_x, f_z) \quad (u_r, u_z)$$

Then the operator \mathbf{L} , which relates the strain vector with the displacement vector through $\boldsymbol{\epsilon} = \mathbf{L}\mathbf{u}$, is

$$\mathbf{L} = \begin{bmatrix} \frac{\partial}{\partial x} & 0 \\ 0 & \frac{\partial}{\partial z} \\ \frac{\partial}{\partial z} & \frac{\partial}{\partial x} \end{bmatrix}$$

$$\mathbf{L} = \begin{bmatrix} \frac{\partial}{\partial r} & 0 & 0 \\ \frac{1}{r} & 0 & \frac{1}{r} \frac{\partial}{\partial \theta} \\ 0 & \frac{\partial}{\partial z} & 0 \\ \frac{\partial}{\partial z} & \frac{\partial}{\partial r} & 0 \\ \frac{1}{r} \frac{\partial}{\partial \theta} & 0 & \frac{\partial}{\partial r} - \frac{1}{r} \\ 0 & \frac{1}{r} \frac{\partial}{\partial \theta} & \frac{\partial}{\partial z} \end{bmatrix}$$

(2.1a)

and also the Laplacian operators ∇^2 are defined as

$$\nabla = \left(\frac{\partial}{\partial x} \quad \frac{\partial}{\partial z} \right)^T$$

$$\nabla = \left(\frac{\partial}{\partial r} \quad \frac{\partial}{\partial z} \quad \frac{1}{r} \frac{\partial}{\partial \theta} \right)^T$$

(2.1b)

The soil medium is assumed to be an elastic porous medium saturated with pore fluid. The displacement of the pore fluid relative to the displacement of the solid skeleton is defined as

$$\mathbf{w} = n(\mathbf{U} - \mathbf{u}) \quad (2.2)$$

where n = porosity; \mathbf{w} = vector containing relative displacements of fluid; \mathbf{u} = vector containing displacements of solid skeleton; and \mathbf{U} = vector containing the average displacements of the fluid over a unit area of the mixture perpendicular to the displacements. The total normal stresses acting on a unit area of mixture is

$$\sigma = (1-n)\sigma_s - mn\pi = \sigma' - mn\pi \quad (2.3)$$

where σ_s = vector containing a normal stresses acting on the solid skeleton averaged over the unit area; σ = vector containing total normal stresses ; σ' = vector containing effective normal stresses; π = pore fluid pressure; and $\mathbf{m} = (1,1,0)^T$.

The equilibrium condition of forces acting on the soil skeleton in a unit volume of soil is described as

$$(1-n) \mathbf{L}^* \sigma_s + (1-n)\rho_s \mathbf{b} + nk^{-1} \dot{\mathbf{w}} = (1-n)\rho_s \ddot{\mathbf{u}} \quad (2.4)$$

where $\dot{w} = \partial w / \partial t$; $\ddot{u} = \partial^2 u / \partial t^2$; b = body forces per unit mass; ρ_s = density of a unit volume of the solid material in the skeleton; and

$$L^* = L^T = \begin{bmatrix} \frac{\partial}{\partial x} & 0 & \frac{\partial}{\partial z} \\ 0 & \frac{\partial}{\partial z} & \frac{\partial}{\partial x} \end{bmatrix}$$

$$L^* = \begin{bmatrix} \frac{\partial}{\partial x} + \frac{1}{r} & -\frac{1}{r} & 0 & \frac{\partial}{\partial z} & \frac{1}{r} \frac{\partial}{\partial \theta} & 0 \\ 0 & 0 & \frac{\partial}{\partial z} & \frac{\partial}{\partial r} + \frac{1}{r} & 0 & \frac{1}{r} \frac{\partial}{\partial \theta} \\ 0 & \frac{1}{r} \frac{\partial}{\partial \theta} & 0 & 0 & \frac{\partial}{\partial r} + \frac{2}{r} & \frac{\partial}{\partial z} \end{bmatrix} \quad (2.5)$$

The equilibrium condition of the forces acting on the pore fluid domain in a unit volume of soil is given by

$$n\nabla\pi - k^{-1}n\dot{w} + n\rho_f b = \rho_f \dot{w} + n\rho_f \ddot{u} \quad (2.6)$$

where ρ_f = density of unit volume of pore fluid. Combining Eqs. 2.4 and 2.6, the equilibrium condition of the pore fluid and solid skeleton mixture is expressed as

$$L^* \sigma + \rho b = \rho \ddot{u} + \rho_f \dot{w} \quad (2.7)$$

where ρ = density of unit volume of mixture = $(1-n)\rho_s + n\rho_f$. Since linear elastic conditions are considered, body forces will be neglected hereafter.

The rate of the fluid stored in a unit volume mixture is equal to the summation of the rate of volumetric strain in the solid frame, the rate of change due to compression of the solid by pore fluid pressure, the rate of change due to compression of the solid by the effective stresses, and the rate of change due to compressibility of the fluid. According to Simon et al. (1984), this is expressed as

$$\nabla^* \mathbf{w} = -\alpha \mathbf{m}^T \boldsymbol{\varepsilon} + Q^{-1} \boldsymbol{\pi} \quad (2.8)$$

where $\boldsymbol{\varepsilon}$ = strains; α and Q = material property parameters defined later in Eq. 2.10; and

$$\nabla^* = \nabla^T$$

$$\nabla^* = \left(\frac{\partial}{\partial r} + \frac{1}{r} \frac{\partial}{\partial z} - \frac{1}{r} \frac{\partial}{\partial \theta} \right) \quad (2.9)$$

The parameters α and Q are defined as

$$\alpha = 1 - \frac{K_d}{K_s} \quad \text{and} \quad Q^{-1} = \frac{1}{K_f} + \frac{\alpha \cdot n}{K_s} \quad (2.10)$$

where K_s = elastic volumetric modulus of solid; K_f = volumetric modulus of fluid; and K_d = elastic volumetric modulus of solid skeleton. The strains in a soil mass result from deformation of skeleton and deformation of solid particles. The former and latter are caused respectively by the particle-particle contact pressure and all around pore pressure. Therefore, the strains of a gross soil mass are expressed by

$$\boldsymbol{\epsilon} = \mathbf{D}^{-1} \boldsymbol{\sigma}' + \frac{\pi}{3K_s} \mathbf{m} \quad (2.11)$$

where \mathbf{D} = elastic stiffness matrix of soil skeleton. Substituting Eq. 2.11 into Eq. 2.3 and $\mathbf{D}\mathbf{m} = 3K_d\mathbf{m}$, the total stresses, $\boldsymbol{\sigma}$, can be correlated with $\boldsymbol{\epsilon}$ and π .

$$\boldsymbol{\sigma} = \mathbf{D}\boldsymbol{\epsilon} + (\mathbf{m} - \frac{\pi}{3K_s} \mathbf{m})\pi = \mathbf{D}\boldsymbol{\epsilon} + \alpha\pi\mathbf{m} \quad (2.12)$$

After substituting π in Eq. 2.8 into Eq. 2.12 and using $\boldsymbol{\epsilon} = \mathbf{L}\mathbf{u}$, Eqs. 2.8 and 2.12 result in the expressions of the stresses $\boldsymbol{\sigma}$ and porewater pressure π as

$$\begin{bmatrix} \boldsymbol{\sigma} \\ \pi \end{bmatrix} = \begin{bmatrix} (\mathbf{D} + \alpha^2 Q \mathbf{m} \mathbf{m}^T) \mathbf{L} & \alpha Q \mathbf{m} \nabla^* \\ \alpha Q \mathbf{m}^T \mathbf{L} & Q \nabla^* \end{bmatrix} \begin{bmatrix} \mathbf{u} \\ \mathbf{w} \end{bmatrix} \quad (2.13)$$

The operations inside the matrix on the right hand side of Eq. 2.13 result in

$$[] = \begin{bmatrix} A \frac{\partial}{\partial x} & (A-2G) \frac{\partial}{\partial z} & \alpha Q \frac{\partial}{\partial x} & \alpha Q \frac{\partial}{\partial z} \\ (A-2G) \frac{\partial}{\partial x} & A \frac{\partial}{\partial z} & \alpha Q \frac{\partial}{\partial x} & \alpha Q \frac{\partial}{\partial z} \\ G \frac{\partial}{\partial z} & G \frac{\partial}{\partial x} & 0 & 0 \\ \alpha Q \frac{\partial}{\partial x} & \alpha Q \frac{\partial}{\partial z} & Q \frac{\partial}{\partial x} & Q \frac{\partial}{\partial z} \end{bmatrix}$$

$$[] = \begin{bmatrix} A \frac{\partial}{\partial r} + \frac{A-2G}{r} & (A-2G) \frac{\partial}{\partial z} & \frac{A-2G}{r} \frac{\partial}{\partial \theta} & \alpha Q \left(\frac{\partial}{\partial r} + \frac{1}{r} \right) & \alpha Q \frac{\partial}{\partial z} & \frac{\alpha Q}{r} \frac{\partial}{\partial \theta} \\ (A-2G) \frac{\partial}{\partial r} + \frac{A}{r} & (A-2G) \frac{\partial}{\partial z} & \frac{A}{r} \frac{\partial}{\partial \theta} & \alpha Q \left(\frac{\partial}{\partial r} + \frac{1}{r} \right) & \alpha Q \frac{\partial}{\partial z} & \frac{\alpha Q}{r} \frac{\partial}{\partial \theta} \\ (A-2G) \left(\frac{\partial}{\partial r} + \frac{1}{r} \right) & A \frac{\partial}{\partial z} & \frac{A-2G}{r} \frac{\partial}{\partial \theta} & \alpha Q \left(\frac{\partial}{\partial r} + \frac{1}{r} \right) & \alpha Q \frac{\partial}{\partial z} & \frac{\alpha Q}{r} \frac{\partial}{\partial \theta} \\ G \frac{\partial}{\partial z} & G \frac{\partial}{\partial r} & 0 & 0 & 0 & 0 \\ \frac{G}{r} \frac{\partial}{\partial \theta} & 0 & \frac{G}{r} \frac{\partial}{\partial r} & 0 & 0 & 0 \\ \alpha Q \left(\frac{\partial}{\partial r} + \frac{1}{r} \right) & \alpha Q \frac{\partial}{\partial z} & \frac{\alpha Q}{r} \frac{\partial}{\partial \theta} & Q \left(\frac{\partial}{\partial r} + \frac{1}{r} \right) & Q \frac{\partial}{\partial z} & \frac{Q}{r} \frac{\partial}{\partial \theta} \end{bmatrix}$$

(2.14)

Using π and σ defined in Eq. 2.13, Eqs. 2.6 and 2.7 can be rewritten in the following matrix form after using the relationship $\boldsymbol{\varepsilon} = \mathbf{L}\mathbf{u}$:

$$-\mathbf{M}\begin{Bmatrix} \dot{\mathbf{u}} \\ \ddot{\mathbf{w}} \end{Bmatrix} - \mathbf{C}\begin{Bmatrix} \dot{\mathbf{u}} \\ \dot{\mathbf{w}} \end{Bmatrix} + \mathbf{K}\begin{Bmatrix} \mathbf{u} \\ \mathbf{w} \end{Bmatrix} = \begin{Bmatrix} \mathbf{0} \\ \mathbf{0} \end{Bmatrix} \quad (2.15)$$

where

$$\mathbf{M} = \begin{bmatrix} \rho & \rho_f \\ \rho_f & \rho_f/n \end{bmatrix} \quad \mathbf{C} = \begin{bmatrix} 0 & 0 \\ 0 & k^{-1} \end{bmatrix} \quad \mathbf{K} = \begin{bmatrix} \mathbf{L}^*(\mathbf{D} + \alpha^2 Q \mathbf{m} \mathbf{m}^T) \mathbf{L} & \alpha Q \mathbf{L}^* \mathbf{m} \nabla^* \\ \alpha Q \nabla \mathbf{m}^T \mathbf{L} & Q \nabla \nabla^* \end{bmatrix} \quad (2.16)$$

Operations inside \mathbf{K} result in

$$\begin{aligned} \mathbf{L}^*(\mathbf{D} + \alpha^2 Q \mathbf{m} \mathbf{m}^T) \mathbf{L} &= \begin{bmatrix} A \frac{\partial^2}{\partial x^2} + G \frac{\partial^2}{\partial z^2} & (A - G) \frac{\partial^2}{\partial x \partial z} \\ (A - G) \frac{\partial^2}{\partial x \partial z} & A \frac{\partial^2}{\partial z^2} + G \frac{\partial^2}{\partial x^2} \end{bmatrix} \\ \mathbf{L}^*(\mathbf{D} + \alpha^2 Q \mathbf{m} \mathbf{m}^T) \mathbf{L} &= \begin{bmatrix} A \left(\frac{\partial^2}{\partial r^2} + \frac{1}{r} \frac{\partial}{\partial r} \frac{1}{r^2} \right) + G \frac{\partial^2}{r^2 \partial \theta^2} + G \frac{\partial^2}{\partial z^2} & (A - G) \frac{\partial^2}{\partial r \partial z} & -\frac{A+G}{r^2} \frac{\partial}{\partial \theta} + \frac{A-G}{r} \frac{\partial^2}{\partial r \partial \theta} \\ (A - G) \left(\frac{\partial^2}{\partial r \partial z} + \frac{1}{r} \frac{\partial}{\partial z} \right) & A \frac{\partial^2}{\partial z^2} + G \left(\frac{\partial^2}{\partial r^2} + \frac{1}{r} \frac{\partial}{\partial r} + \frac{1}{r^2} \frac{\partial^2}{\partial \theta^2} \right) & \frac{A-G}{r} \frac{\partial^2}{\partial z \partial \theta} \\ \frac{A+G}{r^2} \frac{\partial}{\partial \theta} + \frac{A-G}{r} \frac{\partial^2}{\partial r \partial \theta} & \frac{A-G}{r} \frac{\partial^2}{\partial \theta \partial z} & \frac{A}{r^2} \frac{\partial^2}{\partial \theta^2} + G \left(\frac{\partial^2}{\partial r^2} + \frac{1}{r} \frac{\partial}{\partial r} \frac{1}{r^2} \right) + G \frac{\partial^2}{\partial z^2} \end{bmatrix} \end{aligned}$$

$$\begin{aligned}
L^* m \nabla^* &= \nabla m^T L = \nabla \nabla^* = \begin{bmatrix} \frac{\partial^2}{\partial x^2} & \frac{\partial^2}{\partial x \partial z} \\ \frac{\partial^2}{\partial x \partial z} & \frac{\partial^2}{\partial z^2} \end{bmatrix} \\
L^* m \nabla^* &= \nabla m^T L = \nabla \nabla^* = \begin{bmatrix} \frac{\partial^2}{\partial r^2} + \frac{1}{r} \frac{\partial}{\partial r} \frac{1}{r^2} & \frac{\partial^2}{\partial r \partial z} & \frac{1}{r} \frac{\partial^2}{\partial \theta \partial r} \frac{1}{r^2} \frac{\partial}{\partial \theta} \\ \frac{\partial^2}{\partial z \partial r} + \frac{1}{r} \frac{\partial}{\partial z} & \frac{\partial^2}{\partial z^2} & \frac{1}{r} \frac{\partial^2}{\partial \theta \partial z} \\ \frac{1}{r} \frac{\partial^2}{\partial \theta \partial r} + \frac{1}{r^2} \frac{\partial}{\partial \theta} & \frac{1}{r} \frac{\partial^2}{\partial \theta \partial z} & \frac{1}{r^2} \frac{\partial^2}{\partial \theta^2} \end{bmatrix}
\end{aligned}$$

(2.17)

3 SOLUTIONS BY THIN LAYER ELEMENT METHOD

3.1. Plane Strain Conditions in Cartesian Coordinates

Consider a horizontally layered submerged soil. The displacements of the medium in the horizontally propagating wave field is expressed in the form of $(u(x,z,t), w(x,z,t)) = (u(z), w(z))e^{i(\omega t - hx)}$ in which ω = circular frequency and h = wave number. Thus, omitting the time factor, the displacements of the j th layer in the wave field are assumed to be expressed as

$$\begin{Bmatrix} u_j(x,z) \\ w_j(x,z) \end{Bmatrix} = H(hx) \begin{Bmatrix} u_j(z) \\ w_j(z) \end{Bmatrix} \quad (3.1)$$

where $H(hx)$ = 4x4 diagonal matrix containing e^{-ihx} ; and u_j and w_j = displacement vectors containing x and z components in each of u_j and w_j . Also, using a shape function for expressing the displacements in the z coordinate, displacements within the j -th layer are approximately expressed as

$$\begin{Bmatrix} u_j(x,z) \\ w_j(x,z) \end{Bmatrix} = V(z) \begin{Bmatrix} U_j(x) \\ W_j(x) \end{Bmatrix} \quad (3.2)$$

where $U_j(x)^T = (u_j(x,0)^T, u_j(x,L_j)^T)^T$ and $W_j(x)^T = (w_j(x,0)^T, w_j(x,L_j)^T)^T$; and $z = 0$ and L_j are respectively the locations of the upper end and lower ends of the j th layer; ; and $V(z)$ = matrix containing shape function. Thus, we will have

$$\begin{aligned} \begin{Bmatrix} \mathbf{u}_j(z) \\ \mathbf{w}_j(z) \end{Bmatrix} &= \mathbf{V}(z) \begin{Bmatrix} \mathbf{U}_j \\ \mathbf{W}_j \end{Bmatrix} \\ \begin{Bmatrix} \mathbf{U}_j(x) \\ \mathbf{W}_j(x) \end{Bmatrix} &= \mathbf{H}(hx) \begin{Bmatrix} \mathbf{U}_j \\ \mathbf{W}_j \end{Bmatrix} \end{aligned} \quad (3.3)$$

When continuous linear variations of the displacements are assumed along z, the shape function matrix \mathbf{V} is

$$\mathbf{V}(z) = \left(1 - \frac{z}{L_j}, \frac{z}{L_j} \right) \mathbf{I} \quad (3.4)$$

where $\mathbf{I} = 4 \times 4$ identity matrix. It is noted in Eq. 3.4 that the factor $(1-z/L_j, z/L_j)$ is simply multiplied to the numbers in \mathbf{I} and thus \mathbf{V} is a matrix with 4 rows and 8 columns.

With Eqs. 3.1 and 3.4, Galarkin's procedure applied to Eq. 2.15 results in

$$\sum_{j=1}^J \int_0^{L_j} \mathbf{V}^T \left(\mathbf{K}_j + i\omega \mathbf{C}_j - \omega^2 \mathbf{M}_j \right) \mathbf{H} \mathbf{V} \begin{Bmatrix} \mathbf{U}_j \\ \mathbf{W}_j \end{Bmatrix} dz - \sum_{j=1}^J \mathbf{V}^T \tilde{\mathbf{K}}_j \mathbf{H} \mathbf{V} \begin{Bmatrix} \mathbf{U}_j \\ \mathbf{W}_j \end{Bmatrix} \Big|_0^{L_j} = \mathbf{0} \quad (3.5)$$

where $J = \text{numbers of layers}$; and $\tilde{\mathbf{K}}_j = \int \mathbf{K}_j dz$. The inside of the second summation in Eq. 3.5, \mathbf{S} , can be rewritten as

$$\mathbf{S} = \int_0^{L_j} \frac{\partial}{\partial z} \left[\mathbf{V}^T \tilde{\mathbf{K}}_j \mathbf{H} \mathbf{V} \right] \begin{Bmatrix} \mathbf{U}_j \\ \mathbf{W}_j \end{Bmatrix} dz$$

$$= \int_0^{L_j} \frac{\partial \mathbf{V}^T \tilde{\mathbf{K}}_j}{\partial z} \mathbf{H} \mathbf{V} \begin{Bmatrix} \mathbf{U}_j \\ \mathbf{W}_j \end{Bmatrix} dz + \int_0^{L_j} \mathbf{V}^T \frac{\partial \tilde{\mathbf{K}}_j}{\partial z} \mathbf{H} \mathbf{V} \begin{Bmatrix} \mathbf{U}_j \\ \mathbf{W}_j \end{Bmatrix} dz \quad (3.6)$$

Substituting Eq. 3.6 into Eq. 3.5 and integrating it with respect to z result in the characteristics equation in the discretized form such that

$$\sum_{j=1}^J \int_0^{L_j} \left\{ \mathbf{V}^T \left(\mathbf{K} - i\omega C + \omega^2 M - \frac{\partial \tilde{\mathbf{K}}}{\partial z} \right) \mathbf{H} \mathbf{V} - \frac{\partial \mathbf{V}^T \tilde{\mathbf{K}}}{\partial z} \mathbf{H} \mathbf{V} \right\} \begin{Bmatrix} \mathbf{U}_j \\ \mathbf{W}_j \end{Bmatrix} dz = \mathbf{0} \quad (3.7)$$

Then, after the elimination of a diagonal matrix \mathbf{H} , partial differentiations and integration result in

$$\sum_{j=1}^J \left(h^2 \alpha_j + ih\beta_j + \gamma_j \right) \begin{Bmatrix} \mathbf{U}_j \\ \mathbf{W}_j \end{Bmatrix} = \mathbf{0}$$

or , this can be rewritten as

$$(h^2 \alpha + ih\beta + \gamma) \begin{Bmatrix} \mathbf{U} \\ \mathbf{W} \end{Bmatrix} = \mathbf{0} \quad (3.8)$$

where \mathbf{U} and \mathbf{W} = vectors containing respectively $\mathbf{u}_j(z=0)$ and $\mathbf{w}_j(z=0)$ of all layers

$$\alpha_j = \frac{L_j}{6} \begin{bmatrix} Aa & 0a & \alpha Qa & 0a \\ 0a & Ga & 0a & 0a \\ \alpha Qa & Qa & Qa & 0a \\ 0a & 0a & 0a & 0a \end{bmatrix}$$

$$\beta_j = \frac{1}{2} \begin{bmatrix} 0b & (A-2G)b & 0b & \alpha Qb \\ Gb & 0b & 0b & 0b \\ 0b & \alpha Qb & 0b & Qb \\ 0b & 0b & 0b & 0b \end{bmatrix}^T \cdot \frac{1}{2} \begin{bmatrix} 0b & (A-2G)b & 0b & \alpha Qb \\ Gb & 0b & 0b & 0b \\ 0b & \alpha Qb & 0b & Qb \\ 0b & 0b & 0b & 0b \end{bmatrix}$$

$$\gamma_j = \frac{1}{L_j} \begin{bmatrix} Gc & 0c & 0c & 0c \\ 0c & Ac & 0c & \alpha Qc \\ 0c & 0c & 0c & 0c \\ 0c & \alpha Qc & 0c & Qc \end{bmatrix} + i\omega \frac{L_j}{6} \begin{bmatrix} 0a & 0a & 0a & 0a \\ 0a & 0a & 0a & 0a \\ 0a & 0a & k^{-1}a & 0a \\ 0a & 0a & 0a & k^{-1}a \end{bmatrix} - \omega^2 \frac{L_j}{6} \begin{bmatrix} pa & 0a & p_f a & 0a \\ 0a & pa & 0a & p_f a \\ p_f a & 0a & p_f/na & 0a \\ 0a & p_f a & 0a & p_f/na \end{bmatrix} \quad (3.9)$$

in which $A = \lambda + 2G + \alpha^2 Q$; and

$$\mathbf{a} = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix} \quad \mathbf{b} = \begin{bmatrix} 1 & -1 \\ 1 & -1 \end{bmatrix} \quad \mathbf{c} = \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \quad (3.10)$$

Wave numbers, h , and their associated mode shape vectors, $(\phi_u^T, \phi_w^T)^T$, can be determined by solving the characteristic equation given by Eq. 3.8. Since the fluid pressure is an all round equal pressure, there is a constraint between the freedoms associated with w_x and w_z and therefore the characteristic equation for J layers results in the number of eigenvalues equal to $3J$ conjugate pairs instead of $4J$. In order to satisfy the wave scattering conditions, only those with the plus sign in the imaginary part are selected among the conjugate pairs. Then, the displacements of the submerged layered-soil is expressed along x at the nodal points, $\mathbf{U}(x)$ and $\mathbf{W}(x)$, as

$$\begin{Bmatrix} \mathbf{U}(x) \\ \mathbf{W}(x) \end{Bmatrix} = \sum_{s=1}^{3J} H(h_s x) \begin{Bmatrix} \mathbf{U} \\ \mathbf{W} \end{Bmatrix}_s = \sum_{s=1}^{3J} e^{-ih_s x} \eta_s \begin{Bmatrix} \phi_u \\ \phi_w \end{Bmatrix}_s \quad (3.11)$$

and thus the displacements within the j -th layer as

$$\begin{Bmatrix} \mathbf{u}_j(x, z) \\ \mathbf{w}_j(x, z) \end{Bmatrix} = V(z) \sum_{s=1}^{3J} e^{-ih_s x} \eta_s \begin{Bmatrix} \phi_{uj} \\ \phi_{wj} \end{Bmatrix}_s \quad (3.12)$$

where $(\phi_u^T, \phi_w^T)_s^T$ = s-th eigenvector in which ϕ_u and ϕ_w = vectors of size $2J$; ϕ_{uj} and ϕ_{wj} = vectors containing the values at the locations corresponding the jth layer in ϕ_u and ϕ_w , respectively; h_s = s-th eigenvalue; and α_s = s-th mode participation factor. Then, using Eqs. 2.13 and 3.12 together with Eq. 3.4, the stresses and pore fluid pressure at the middle of the jth layer are

$$\begin{Bmatrix} \sigma_j(x, 0.5H_j) \\ \pi_j(x, 0.5H_j) \end{Bmatrix} = -i A_j \sum_{s=1}^{3J} h_s e^{-ih_s x} \eta_s \begin{Bmatrix} \phi_{uj} \\ \phi_{wj} \end{Bmatrix}_s - B_j \sum_{s=1}^{3J} e^{-ih_s x} \eta_s \begin{Bmatrix} \phi_{uj} \\ \phi_{wj} \end{Bmatrix}_s \quad (3.13)$$

where

$$A_j = \frac{1}{2} \begin{bmatrix} A & A & 0 & 0 & \alpha Q & \alpha Q & 0 & 0 \\ A-2G & A-2G & 0 & 0 & \alpha Q & \alpha Q & 0 & 0 \\ 0 & 0 & G & G & 0 & 0 & 0 & 0 \\ \alpha Q & \alpha Q & 0 & 0 & Q & Q & 0 & 0 \end{bmatrix} \quad (3.14)$$

$$B_j = \frac{1}{L_j} \begin{bmatrix} 0 & 0 & (A-2G) & -(A-2G) & 0 & 0 & \alpha Q & -\alpha Q \\ 0 & 0 & A & -A & 0 & 0 & \alpha Q & -\alpha Q \\ G & -G & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \alpha Q & -\alpha Q & 0 & 0 & Q & -Q \end{bmatrix}$$

A vertical cut is considered at $x=0$ in a layered soil. After substituting Eqs. 3.4 and 3.12 into Eq. 2.12, the nodal forces acting on the vertical cut of the jth layer for $x \geq 0$ can be obtained as

$$\begin{Bmatrix} P_{xj} \\ P_{zj} \\ P_{\pi j} \end{Bmatrix} = - \int_0^{H_j} V^T \begin{Bmatrix} \sigma_{xj}(0,z) \\ \tau_{xzj}(0,z) \\ \pi_j(0,z) \end{Bmatrix} dz = i E_j \sum_{s=1}^{3J} h_s \eta_s \begin{Bmatrix} \phi_{uj} \\ \phi_{wxj} \end{Bmatrix}_s + F_j \sum_{s=1}^{3J} \eta_s \begin{Bmatrix} \phi_{uj} \\ \phi_{wzj} \end{Bmatrix}_s \quad (3.15)$$

where $\mathbf{P}_{xj} = (P_{xj}(z=0), P_{xj}(z=L_j))^T$; $\mathbf{P}_{zj} = (P_{zj}(z=0), P_{zj}(z=L_j))^T$; $\mathbf{P}_{\pi j} = (P_{\pi j}(z=0), P_{\pi j}(z=L_j))^T$; and

$$\mathbf{E}_j = \frac{H_j}{6} \begin{bmatrix} Aa & 0a & \alpha Qa \\ 0a & Ga & 0a \\ \alpha Qa & 0a & Qa \end{bmatrix} \quad \text{with } \mathbf{a} = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}$$

$$\mathbf{F}_j = \frac{1}{2} \begin{bmatrix} 0b & (A-2G)b & \alpha Qb \\ Gb & 0b & 0b \\ 0b & \alpha b & Qb \end{bmatrix} \quad \text{with } \mathbf{b} = \begin{bmatrix} 1 & -1 \\ 1 & -1 \end{bmatrix}$$
(3.16)

Therefore those of the entire layered system for $x \geq 0$ are

$$\begin{Bmatrix} \mathbf{P}_x \\ \mathbf{P}_z \\ \mathbf{P}_\pi \end{Bmatrix} = i \mathbf{E} \sum_{s=1}^{3J} h_s \eta_s \begin{Bmatrix} \phi_u \\ \phi_{wx} \\ \phi_{wz} \end{Bmatrix}_s + \mathbf{F} \sum_{s=1}^{3J} \eta_s \begin{Bmatrix} \phi_u \\ \phi_{wx} \\ \phi_{wz} \end{Bmatrix}_s$$

or

$$\begin{Bmatrix} \mathbf{P}_x \\ \mathbf{P}_z \\ \mathbf{P}_\pi \end{Bmatrix} = \Gamma \eta \quad (3.17)$$

where η = vector with a size $3J$ containing η_s at the s -th location; and $\Gamma = 3J \times 3J$ matrix in which the s -th column is

$$ih_s \mathbf{E} \begin{Bmatrix} \phi_u \\ \phi_{wx} \\ \phi_{wz} \end{Bmatrix}_s + \mathbf{F} \begin{Bmatrix} \phi_u \\ \phi_{wx} \\ \phi_{wz} \end{Bmatrix}_s \quad (3.18)$$

Given external loads at $x=0$, \mathbf{Q}_x , \mathbf{Q}_z and \mathbf{Q}_π , the participation factors, α , can be determined from Eq. 3.17 with $(\mathbf{P}_x, \mathbf{P}_z, \mathbf{P}_\pi) = (\mathbf{Q}_x, \mathbf{Q}_z, \mathbf{Q}_\pi)/2$ and $(\mathbf{P}_x, \mathbf{P}_z, \mathbf{P}_\pi) = (\mathbf{Q}_x,$

Q_z, Q_π) for the soil medium extending respectively through $-\infty \leq x \leq +\infty$ and $0 \leq x \leq +\infty$. With those participation factors, the displacements can be determined from Eq. 3.11 and the stresses and pore pressure from Eq. 3.13.

3.2. Three-Dimensional Conditions in Cylindrical Coordinates

Assuming the out-going cylindrical wave field, the displacements of the j-th layer are expressed as

$$\begin{Bmatrix} u_j(r, \theta, z) \\ w_j(r, \theta, z) \end{Bmatrix} = H(hr, \theta) \begin{Bmatrix} u_j(z) \\ w_j(z) \end{Bmatrix} \quad (3.19)$$

where u_j and w_j = displacement vectors of size 3 containing the r, z and θ components; and

$$H(hr, \theta) = \sum_{m=0}^{\infty} e^{im\theta} \begin{bmatrix} H_m(hr) & 0 \\ 0 & H_m(hr) \end{bmatrix} \quad (3.20a)$$

$$H_m(hr) = \begin{bmatrix} H_m^{(2)}(hr) & 0 & \frac{m}{r} H_m^{(2)}(hr) \\ 0 & h H_m^{(2)}(hr) & 0 \\ \frac{i m}{r} H_m^{(2)}(hr) & 0 & i H_m^{(2)}(hr) \end{bmatrix} \quad (3.20b)$$

where $H_m^{(2)}$ = second kind m-th order Hankel function. With a shape function defined by Eq. 3.4, $u_j(r, \theta, z)$ and $w_j(r, \theta, z)$ are similarly expressed as

$$\begin{Bmatrix} \mathbf{u}_j(r,z,\theta) \\ \mathbf{w}_j(r,z,\theta) \end{Bmatrix} = \mathbf{V}(z) \begin{Bmatrix} \mathbf{U}_j(r,\theta) \\ \mathbf{W}_j(r,\theta) \end{Bmatrix} \quad (3.21)$$

in which $\mathbf{U}_j(r,\theta)^T = (\mathbf{u}_j(r,0,\theta)^T, \mathbf{u}_j(r,L_j,\theta)^T)^T$ and $\mathbf{W}_j(r,\theta)^T = (\mathbf{w}_j(x,0,\theta)^T, \mathbf{w}_j(x,L_j,\theta)^T)^T$. With a continuous linear variation of displacements within a layer, $\mathbf{V}(z)$ is identical to Eq. 3.4 but \mathbf{I} is a 6x6 identity matrix. Eqs. 3.19 and 3.21 lead to

$$\begin{aligned} \begin{Bmatrix} \mathbf{u}_j(z) \\ \mathbf{w}_j(z) \end{Bmatrix} &= \mathbf{V}(z) \begin{Bmatrix} \mathbf{U}_j \\ \mathbf{W}_j \end{Bmatrix} \\ \begin{Bmatrix} \mathbf{U}_j(r,\theta) \\ \mathbf{W}_j(r,\theta) \end{Bmatrix} &= \mathbf{H}(hr,\theta) \begin{Bmatrix} \mathbf{U}_j \\ \mathbf{W}_j \end{Bmatrix} \end{aligned} \quad (3.22)$$

Applying the procedure similar to that used for the plane strain problem, the following discretized characteristics equation is obtained:

$$\left(h^2 \alpha + ih\beta + \gamma \right) \begin{Bmatrix} \mathbf{U} \\ \mathbf{W} \end{Bmatrix} = \mathbf{0} \quad (3.23)$$

where α , β and γ are obtained respectively from α_j , β_j and γ_j . Rearranging the matrices α_j , β_j and γ_j so as to correspond to displacement vector $(u_{rj}, u_{zj}, w_{rj}, w_{zj}, u_{\theta j}, w_{\theta j})^T$, matrices α_j , β_j and γ_j are respectively

$$\begin{aligned}
\alpha_j &= \mathbf{a} \begin{bmatrix} A & 0 & \alpha Q & 0 \\ 0 & G & 0 & 0 \\ \alpha Q & 0 & Q & 0 \\ 0 & 0 & 0 & 0 \\ & & & G & 0 \\ & & & 0 & 0 \end{bmatrix} \\
\beta_j &= \frac{1}{2} \mathbf{b} \begin{bmatrix} 0 & -(A-2G) & 0 & -\alpha Q \\ G & 0 & 0 & 0 \\ 0 & -\alpha Q & 0 & -Q \\ 0 & 0 & 0 & 0 \\ & & 0 & 0 \\ & & 0 & 0 \end{bmatrix} + \frac{1}{2} \mathbf{b}^T \begin{bmatrix} 0 & -(A-2G) & 0 & -\alpha Q \\ G & 0 & 0 & 0 \\ 0 & -\alpha Q & 0 & -Q \\ 0 & 0 & 0 & 0 \\ & & 0 & 0 \\ & & 0 & 0 \end{bmatrix}^T \\
\gamma_j &= \frac{1}{L_j} \mathbf{c} \begin{bmatrix} A & 0 & \alpha Q & 0 \\ 0 & G & 0 & 0 \\ \alpha Q & 0 & Q & 0 \\ 0 & 0 & 0 & 0 \\ & & & G & 0 \\ & & & 0 & 0 \end{bmatrix} + i \omega \frac{L_j}{6} \mathbf{a} \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & k^{-1} & 0 \\ 0 & 0 & 0 & k^{-1} \\ & & 0 & 0 \\ & & 0 & k^{-1} \end{bmatrix} - \\
&\quad \omega^2 \frac{L_j}{6} \mathbf{a} \begin{bmatrix} \rho & 0 & \rho_f & 0 \\ 0 & \rho & 0 & \rho_f \\ \rho_f & 0 & \rho_f/n & 0 \\ 0 & \rho_f & 0 & \rho_f/n \\ & & \rho & \rho_f \\ & & \rho_f & \rho_f/n \end{bmatrix}
\end{aligned} \tag{3.24}$$

where \mathbf{a}, \mathbf{b} and \mathbf{c} are 2×2 matrices multiplied to each numbers in the immediately following matrices and defined respectively as

$$\mathbf{a} = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix} \quad \mathbf{b} = \begin{bmatrix} -1 & 1 \\ -1 & 1 \end{bmatrix} \quad \mathbf{c} = \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \tag{3.25}$$

The expressions of α_j , β_j and γ_j indicate that the displacements in the θ coordinate direction are uncoupled with the displacements in the r and z coordinate directions. Thus,

the characteristic equation Eq. 3.22 can be split into the following two independent equations:

$$(h^2 \alpha^{rz} + ih\beta^{rz} + \gamma^{rz}) \begin{Bmatrix} \mathbf{U}^{rz} \\ \mathbf{W}^{rz} \end{Bmatrix} = \mathbf{0} \quad (3.26a)$$

$$(h^2 \alpha^\theta + \gamma^\theta) \begin{Bmatrix} \mathbf{U}^\theta \\ \mathbf{W}^\theta \end{Bmatrix} = \mathbf{0} \quad (3.26b)$$

where $\mathbf{U}^{rz} = (\mathbf{U}_r^T, \mathbf{U}_z^T)^T$; $\mathbf{W}^{rz} = (\mathbf{W}_r^T, \mathbf{W}_z^T)^T$; $\mathbf{U}^\theta = \mathbf{U}_\theta$; and $\mathbf{W}^\theta = \mathbf{W}_\theta$.

When the circular cross section of pile is maintained and the lateral displacement of pile is considered, m in Eq. 3.20a is equal to one. Solving Eq. 3.26 for wave numbers and their associated mode shapes, the displacements along the horizontal plane at the nodal points, $\mathbf{U}(r, \theta)$ and $\mathbf{W}(r, \theta)$, are expressed as

$$\begin{aligned} \begin{Bmatrix} \mathbf{U}(r, \theta) \\ \mathbf{W}(r, \theta) \end{Bmatrix} &= \sum_{s=1}^{3J} \eta_s^{rz} H(h_s^{rz} r, \theta) \begin{Bmatrix} \phi_u^{rz} \\ 0 \\ \phi_w^{rz} \end{Bmatrix}_s + \sum_{s=3J+1}^{4J} \eta_s^\theta H(h_s^\theta r, \theta) \begin{Bmatrix} 0 \\ \phi_\theta^\theta \\ \phi_u^\theta \\ 0 \end{Bmatrix}_s \\ &= \sum_{s=1}^{4J} \eta_s H(h_s r, \theta) \begin{Bmatrix} \phi_u \\ \phi_w \end{Bmatrix}_s \end{aligned} \quad (3.27)$$

where h^{rz} , ϕ_u^{rz} and ϕ_w^{rz} are determined from Eq. 3.26a and h^θ , ϕ_u^θ and ϕ_w^θ are from Eq. 3.26b; $\phi_u = ((\phi_u^{rz})^T, (\phi_u^\theta)^T)^T$; $\phi_w = ((\phi_w^{rz})^T, \mathbf{0}^T)^T$. The displacements within the j -th layer are

$$\begin{Bmatrix} \mathbf{u}_j(r, z, \theta) \\ \mathbf{w}_j(r, z, \theta) \end{Bmatrix} = V(z)e^{i\theta} \sum_{s=1}^{4J} \eta_s H(h_s r) \begin{Bmatrix} \phi_{uj} \\ \phi_{wj} \end{Bmatrix}_s \quad (3.28)$$

where

$$H(h_s r, \theta) = e^{i\theta} H(h_s r) \quad (3.29)$$

Substituting Eq. 3.28 into Eq. 2.13, the stresses and pore pressure at the middle of the layer are

$$\begin{Bmatrix} \sigma_j(r, \theta, 0.5H_j) \\ \pi_j(r, \theta, 0.5H_j) \end{Bmatrix} = -A_j e^{i\theta} \sum_{s=1}^{4J} h_s^2 \eta_s H_a(h_s r) \begin{Bmatrix} \phi_{uj} \\ \phi_{wj} \end{Bmatrix}_s - B_j e^{i\theta} \sum_{s=1}^{4J} h_s \eta_s H_b(h_s r) \begin{Bmatrix} \phi_{uj} \\ \phi_{wj} \end{Bmatrix}_s - C_j e^{i\theta} \sum_{s=1}^{4J} h_s \eta_s H_b(h_s r) \begin{Bmatrix} \phi_{uj} \\ \phi_{wj} \end{Bmatrix}_s - D_j e^{i\theta} \sum_{s=1}^{4J} \eta_s H_a(h_s r) \begin{Bmatrix} \phi_{uj} \\ \phi_{wj} \end{Bmatrix}_s - E_j e^{i\theta} \sum_{s=1}^{4J} \eta_s H_a(h_s r) \begin{Bmatrix} \phi_{uj} \\ \phi_{wj} \end{Bmatrix}_s \quad (3.30)$$

where

$$A_j = \frac{1}{2} \begin{bmatrix} A & A & 0 & 0 & 0 & 0 & \alpha Q & \alpha Q & 0 & 0 & 0 & 0 \\ A-2G & A-2G & 0 & 0 & 0 & 0 & \alpha Q & \alpha Q & 0 & 0 & 0 & 0 \\ A-2G & A-2G & 0 & 0 & 0 & 0 & \alpha Q & \alpha Q & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & G & G & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & G & G & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \alpha Q & \alpha Q & 0 & 0 & 0 & 0 & Q & Q & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\mathbf{B}_j = \frac{G}{2r} \begin{bmatrix} -2 & -2 & 0 & 0 & 2 & 2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 2 & 2 & 0 & 0 & -2 & -2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 2 & 2 & 0 & 0 & -2 & -2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\mathbf{C}_j = \frac{1}{L_j} \begin{bmatrix} 0 & 0 & -(A-2G) & (A-2G) & 0 & 0 & 0 & 0 & -\alpha Q & \alpha Q & 0 & 0 \\ 0 & 0 & -(A-2G) & (A-2G) & 0 & 0 & 0 & 0 & -\alpha Q & \alpha Q & 0 & 0 \\ 0 & 0 & -A & A & 0 & 0 & 0 & 0 & -\alpha Q & \alpha Q & 0 & 0 \\ G & -G & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & G & -G & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -\alpha Q & \alpha Q & 0 & 0 & 0 & 0 & -Q & Q & 0 & 0 \end{bmatrix}$$

$$\mathbf{D}_j = \frac{2G}{r^2} \begin{bmatrix} -1 & -1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\mathbf{E}_j = \frac{G}{r L_j} \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & -1 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

(3.31)

and \mathbf{H}_a and \mathbf{H}_b are diagonal matrices containing the following Hankel functions in the diagonal terms: respectively

$$\left\{ \begin{array}{l} H_a(1,1) \\ H_a(3,3) \\ H_a(5,5) \\ H_a(7,7) \\ H_a(9,9) \\ H_a(11,11) \end{array} \right\} = \left\{ \begin{array}{l} H_a(2,2) \\ H_a(4,4) \\ H_a(6,6) \\ H_a(8,8) \\ H_a(10,10) \\ H_a(12,12) \end{array} \right\} = \left\{ \begin{array}{l} H_1(h_s r) \\ -H_0(h_s r) \\ H_1(h_s r) \\ H_1(h_s r) \\ -H_0(h_s r) \\ H_1(h_s r) \end{array} \right\} \quad (3.32)$$

$$\left\{ \begin{array}{l} H_b(1,1) \\ H_b(3,3) \\ H_b(5,5) \\ H_b(7,7) \\ H_b(9,9) \\ H_b(11,11) \end{array} \right\} = \left\{ \begin{array}{l} H_b(2,2) \\ H_b(4,4) \\ H_b(6,6) \\ H_b(8,8) \\ H_b(10,10) \\ H_b(12,12) \end{array} \right\} = \left\{ \begin{array}{l} -H_0(h_s r) \\ H_1(h_s r) \\ -H_0(h_s r) \\ -H_0(h_s r) \\ H_1(h_s r) \\ -H_0(h_s r) \end{array} \right\}$$

After substituting Eqs. 3.4 and 3.28 into Eq. 2.13, the nodal forces acting along a cylindrical surface with the radius r_0 at the j -th layer are

$$\left\{ \begin{array}{l} P_{rj} \\ P_{zj} \\ P_{\theta j} / i \\ P_{\pi j} \end{array} \right\} = - \int_0^{H_j} V^T \left\{ \begin{array}{l} \sigma_{rj}(r_0, \theta, z) \\ \tau_{zrj}(r_0, \theta, z) \\ \tau_{\theta rj}(r_0, \theta, z) / i \\ \pi_j(r_0, \theta, z) \end{array} \right\} dz$$

$$= A_j^* e^{i\theta} \sum_{s=1}^{4J} h_s^2 \eta_s H_a(h_s r_0) \left\{ \begin{array}{l} \phi_{uj} \\ \phi_{wrij} \end{array} \right\}_s + B_j^* e^{i\theta} \sum_{s=1}^{4J} h_s \eta_s H_b(h_s r_0) \left\{ \begin{array}{l} \phi_{uj} \\ 0 \end{array} \right\}_s + C_j^* e^{i\theta} \sum_{s=1}^{4J} h_s \eta_s H_b(h_s r_0) \left\{ \begin{array}{l} \phi_{uj} \\ \phi_{wzj} \end{array} \right\}_s$$

$$+ D_j^* e^{i\theta} \sum_{s=1}^{4J} \eta_s H_a(h_s r_0) \left\{ \begin{array}{l} \phi_{uj} \\ 0 \end{array} \right\}_s + E_j^* e^{i\theta} \sum_{s=1}^{4J} \eta_s H_a(h_s r_0) \left\{ \begin{array}{l} \phi_{uj} \\ 0 \end{array} \right\}_s \quad (3.33)$$

where, with **a** and **b** defined in Eq. 3.25,

$$A_j^* = \frac{L_j}{6} \begin{bmatrix} Aa & 0a & 0a & \alpha Qa \\ 0a & Ga & 0a & 0a \\ 0a & 0a & Ga & 0a \\ \alpha Qa & 0a & 0a & Qa \end{bmatrix}$$

$$B_j^* = \frac{G L_j}{6r_0} \begin{bmatrix} -2a & 0a & 2a & 0a \\ 0a & a & 0a & 0a \\ 2a & 0a & -2a & 0a \\ 0a & 0a & 0a & 0a \end{bmatrix}$$

$$C_j^* = \frac{1}{2} \begin{bmatrix} 0b & -(A-2G)b & 0b & -\alpha Qb \\ Gb & 0b & 0b & 0b \\ 0b & 0b & 0b & 0b \\ 0b & -\alpha Qb & 0b & Qb \end{bmatrix}$$

$$D_j^* = \frac{2GL_j}{3r_0} \begin{bmatrix} -1a & 0a & 1a & 0a \\ 0a & 0a & 0a & 0a \\ 1a & 0a & -1a & 0a \\ 0a & 0a & 0a & 0a \end{bmatrix}$$

$$E_j^* = \frac{G}{2r_0} \begin{bmatrix} 0b & 0b & 0b & 0b \\ 1b & 0b & -1b & 0b \\ 0b & 0b & 0b & 0b \\ 0b & 0b & 0b & 0b \end{bmatrix}$$

(3.34)

Therefore the nodal forces of the entire layered system are

$$\left\{ \begin{array}{l} P_r \\ P_z \\ P_\theta / i \\ P_\pi \end{array} \right\} = A^* e^{i\theta} \sum_{s=1}^{4J} h_s^2 \eta_s H_a(h_s r_0) \begin{Bmatrix} \phi_u \\ \phi_{wz} \end{Bmatrix}_s + B^* e^{i\theta} \sum_{s=1}^{4J} h_s \eta_s H_b(h_s r_0) \begin{Bmatrix} \phi_u \\ 0 \end{Bmatrix}_s + C^* e^{i\theta} \sum_{s=1}^{4J} h_s \eta_s H_b(h_s r_0) \begin{Bmatrix} \phi_{uj} \\ \phi_{wz} \end{Bmatrix}_s + D^* e^{i\theta} \sum_{s=1}^{4J} \eta_s H_a(h_s r_0) \begin{Bmatrix} \phi_u \\ 0 \end{Bmatrix}_s + E^* e^{i\theta} \sum_{s=1}^{4J} \eta_s H_a(h_s r_0) \begin{Bmatrix} \phi_u \\ 0 \end{Bmatrix}_s$$

or

$$\begin{Bmatrix} \mathbf{P}_r \\ \mathbf{P}_z \\ \mathbf{P}_\theta / i \\ \mathbf{P}_\pi \end{Bmatrix} = e^{i\theta} \Gamma \boldsymbol{\eta} \quad (3.35)$$

where $\boldsymbol{\eta}$ = vector with a size $4J$ containing η_s at the s -th location; and $\Gamma = 4J \times 4J$ matrix in which the s -th column is

$$\eta_s \mathbf{A}^* \mathbf{H}_a(h_s r_0) \begin{Bmatrix} \phi_u \\ \phi_{wx} \end{Bmatrix}_s + h_s \mathbf{B}^* \mathbf{H}_b(h_s r_0) \begin{Bmatrix} \phi_u \\ 0 \end{Bmatrix}_s + h_s \mathbf{C}^* \mathbf{H}_b(h_s r_0) \begin{Bmatrix} \phi_u \\ \phi_{wz} \end{Bmatrix}_s + \mathbf{D}^* \mathbf{H}_a(h_s r_0) \begin{Bmatrix} \phi_u \\ 0 \end{Bmatrix}_s + \mathbf{E}^* \mathbf{H}_a(h_s r_0) \quad (3.36)$$

Given external nodal forces at $r=r_0$, (Q_x, Q_z, Q_π) , the participation factors, $\boldsymbol{\eta}$, can be determined from Eq. 3.35 with $(P_x, P_z, P_\pi) = (Q_x, Q_z, Q_\pi)$. With those participation factors, the displacements can be determined from Eq. 3.27 and the stresses and pore pressure from Eq. 3.30.

4. COMPUTED RESULTS AND REMARKS

4.1 Conditions Considered for Soil Deposits

The following three conditions are considered for soil deposits:

- (1) Dry soil bed
- (2) Completely submerged soil bed with no water above the soil bed
- (3) Completely submerged soil bed at certain depth in the water

Either drained or undrained conditions are considered for submerged soil bed. Fluid alone exists above the soil bed for the third case. Fluid alone, dry soil and submerged soil with undrained conditions are special cases of the aforesaid formulations for submerged soil with drained conditions and correspond respectively to:

fluid alone	with $u = 0$, $k = \infty$, $n = 1$, $\rho_s = 0$, and $Q = K_f$
dry soil	with $w = 0$, $\rho_f = 0$, and $Q = K_s/(\alpha \cdot n)$
submerged soil in undrained condition	with $w = 0$ and $k = 0$

Homogeneous and inhomogeneous soil profiles assumed are shown in Fig. 4.1. The fundamental natural frequency of those two profile soil deposits are identical to each other. Soil bed permeabilities range generally as shown in Table. 4.1 and 10^{-5} m/s \sim 10^{-1} m/s are used for the computations. The profiles are divided into a number of horizontal homogeneous layers as shown in Fig. 4.1. In addition to those, the behavior in homogeneous profiles are also computed without dividing the soil deposits into a number of layers and it will be stated so whenever this is the case. For the aforementioned last

case, fluid above the soil bed is divided into a number of equal thickness fluid layers of which the number is identical to that of soil bed.

4.2 Plane Strain Conditions in Cartesian Coordinates

Effects of submerged conditions on the dynamic response of soil deposits are examined for a homogeneous soil profile. The present formulation requires dividing the soil into multi-layers for computation. In the first study, however, only one layer is used in order to avoid complexity in the results and to see the essence of coupled behavior of solid skeleton and pore fluid. Fig. 4.2 shows wave number dispersion curves for various conditions. The real and imaginary parts of h define respectively the wave length and the rate of decay of Rayleigh waves according to e^{-ihx} . The modes 1 and 2 denoted in the dispersion curves for dry soil are associated respectively with S-wave and P-wave. When the soil is dry, those mode waves are not progressive at the frequencies below the natural frequencies of the soil deposits associated with S-wave and P-wave respectively for the first and second modes. Above those natural frequencies, they are progressive to form waves in x direction. When the condition is submerged and undrained, the high pore fluid stiffness increases the P-wave velocity and therefore the mode 2 wave does not propagate until the frequency substantially higher than that for dry soil. When this mode wave is not progressive, the displacements decay somewhat more quickly with distance than those for dry condition . The pore fluid stiffness affects the S-wave velocity very little and therefore so does the natural frequencies of the soil associated S-waves. When the condition is submerged and drained, one additional mode exists in this case as seen in Figs. 4.2c and 4.2d because of the additional degree of the freedom due to the fluid motion. In the case $k = 10^{-3}$ m/s, the first and second mode waves are very similar to those observed for the undrained condition but the third mode wave is a progressive wave at any frequency and decays very quickly with x . As is seen in Fig. 4.3, the fluid motion relative to the solid

motion in the third mode is rather independent of permeability although those in other two modes decrease with decreasing permeability. This is because the motions are predominantly governed by S waves in the first mode, P waves mainly transmitted through the soil skeleton in the third mode and P waves transmitted through both the pore fluid and soil skeleton. When the relative fluid motions are large, a strong coupling develops between the pore fluid and soil skeleton. As a result, the dispersion curves associated with the mode 1 and 2 are significantly affected by the coupling and thus aforementioned difference between the two types modes are not clear, as is seen for $k = 10^1$ m/s case.

A completely flexible massless vertical impermeable wall is assumed to be inserted at $x = 0$ in the ground and subjected to a lateral concentrated harmonic load at the location of the ground surface. The top and tip of the wall are assumed to be located respectively at the top and bottom of the soil deposits. Ground displacements are computed along the ground surface by using the above computed wave numbers and mode shapes and are shown in Fig. 4.4 for $\omega H/v_s = 2$. The horizontal and vertical motions are uncoupled with each other for dry soil with $v = .25$ and are governed respectively by the first mode and second mode waves. Therefore, according to the real and imaginary parts of h_s at $\omega H/v_s = 2$ in Fig. 4.2a, the vertical displacements monotonically decay with distance x because of no real part h_s in the first mode, whereas horizontal displacements form wave pattern with the wave length defined by the real part of the first mode wave h_s . When the soil is submerged, the horizontal and vertical motions are coupled each other even with $v = 0.25$ and thus both horizontal and vertical motions form wave pattern in x as soon as the first mode wave propagates. The difference in the imaginary part of h_s in the first mode wave between the dry and submerged conditions can be clearly seen in the difference in wave lengths along x . The amplitudes and phase shifts of the displacements of the wall are shown in Fig. 4.5a at various frequencies for dry soil and submerged soil in undrained

condition. The first peaks are due to the first mode wave and are located around the fundamental natural frequency of the soil deposits associated with S-wave ($\omega H/v_s = \pi/2$). The second peaks, clearly seen in z direction displacements, are due to the second mode wave and are located around the fundamental natural frequency of the soil deposits associated with P-wave ($\omega H/v_p = \pi/2$ with $v_p^2 = (\lambda + 2G + \alpha Q)/\rho$). The difference in pore fluid rigidity affects not only the amplitudes of the response but the location of the second peak. It very little affects the location of the first peak because of its association with S-wave. The relative fluid motions generates damping and therefore suppress the peak as seen in Fig. 4.5b. As frequency increases, however, the relative fluid motions becomes smaller and the difference between the undrained and drained conditions diminishes. Contrary to this, when the motion becomes slow, there is enough time for pore fluid diffusion and thus the submerged soil behavior is closer to that of the dry soil as frequency decreases.

Now the soil responses to the wall motions are re-computed by dividing the soil into 10 layers (see Fig. 4.1), in order to see the distributions of responses along the depth. The computed amplitudes of displacements and pressures at the wall are shown respectively in Figs. 4.6 and 4.7 at $\omega H/v_s = 2$ for various permeabilities of the soil. It is noted that the displacement responses for $k = 10^{-1}$ m/s and 10^{-5} m/s are very little different respectively from those for dry case and undrained case at the frequency considered. The lower the permeability is, the higher the pore fluid pressure is induced due to more difficulty in pore fluid diffusion. In addition, free drainage at the ground surface affects the distribution of pore fluid pressure giving further complexity in the pore fluid pressure in the soil. The pore fluid is far stiffer compared with the soil skeleton stiffness and thus the soil responses along the depth are affected by the difference in permeability as seen in those figures.

Three different conditions are considered as shown in Fig. 4.8 for an identical nonhomogeneous soil profile shown in Fig. 4.1: they include dry soil (case A), submerged soil with water level at the ground surface (case B) and submerged soil with water level above the ground surface (case C). The shear wave velocities of the inhomogeneous soil are defined so that its fundamental natural frequency is identical to that of the homogeneous soil. Similar to the previous study, a completely flexible massless vertical wall is assumed to be inserted in the soil and is subjected to a lateral harmonic motion at the surface of the soil. Both of the soil and the water above the soil is divided into 5 layers each for the computation. The computed displacement amplitudes of the wall are shown in Fig. 4.9 at the ground surface. Clear peaks can be observed around the fundamental natural frequency of the soil associated with S-wave in all three cases. It is interesting to notice that one additional peak exists below this frequency when the water exists above the soil. Again the submerged condition reduces the response significantly. It is also seen that the water body above the soil affects the soil response significantly to reduce the soil response even further.

4.3 Three-Dimensional Conditions in Cylindrical Coordinates

A vertical cylindrical body with a circular cross section is assumed to be embedded from the surface through the bottom of the soil bed. The lateral harmonic excitation at the cylindrical body generates cylindrical waves in the soil medium. Wave numbers of those waves are computed at various frequencies for conditions identical to those considered for the plane strain conditions. Similarly to the plane strain conditions previously studied, a homogeneous soil profile as shown in Fig. 4.1 is used without divided into multi-layer in the computation for simplicity. For cylindrical waves, two independent characteristics equations given by Eqs. 3.26a and 3.26b define the wave numbers. The first equation defines the Rayleigh waves formed by P-waves and SV-waves, of which motions are

parallel to the r - z plane. On the other hand, the second defines the Love waves formed by SH-waves of which the motions are perpendicular to the r - z plane (θ direction).

Computed results are shown in Fig. 4.10 in which the solid and broken curves are respectively the wave numbers for the Rayleigh waves and Love waves. The dispersion curves associated with the Rayleigh waves are generally not much different from those for the plane strain conditions and the dispersion curves associated with the Love waves are hardly affected by the fluid in soil pores. The relative fluid motions in the first and third modes of the Rayleigh waves (Fig. 4.11) are nearly identical to those for the plane strain conditions (Fig. 4.3) except the relative motion in the second mode. The relative fluid motions for the Love waves are close to those in the first mode of the Rayleigh waves (Fig. 4.11). The Love waves are formed by SH waves and therefore this indicates that the first mode of the Rayleigh wave is mainly concerned with the shear wave propagation.

A massless, impervious and completely flexible cylindrical body is assumed to be subjected to horizontal vibration at the location of the soil surface. The responses of the system are computed without dividing the soil into a number of layers for the identical soil conditions to those considered for the plane strain conditions. The vibrations of the cylindrical body generates both the Rayleigh and Love waves simultaneously. The motions perpendicular to the Rayleigh waves (motions in the θ direction) are due to the Love waves and are not generated in the soil mass under the plane strain conditions previously studied. As are seen in Fig. 4.12, the soil motions associated with the Rayleigh waves along the surface are somewhat different from those generated by the plane Rayleigh waves. Particularly, the motions associated with the cylindrical Rayleigh waves (motions in the r - z plane) decays with the distance much more rapidly with the distance than those associated with the plane Rayleigh waves. The pores fluid damps especially the motions associated with the Love waves. The dynamic responses of the cylinder at the top are shown in Fig. 4.13. The amplitudes of the horizontal motions of

the pile in saturated soil decreased as the fluid stiffness increases. However, the vertical motions show the opposite trends which do not occur for the plane strain conditions. Similarly to the responses for the plane strain conditions, the fluid motions relative to the soil skeleton motions reduces the amplifications in soil motions. The vertical motions are more affected by the pore fluid for the cylindrical structure than the plane strain structure. Fig. 4.14 shows the effects of the variation of the water level on the response. The water levels are assumed to be 10 m and 20 m above the ground surface. As observed for the plane strain structure, the response of the structure is affected by the variation of the water level.

The response of the soil and cylindrical bodies are computed along the depth. The soil is divided into 10 layers in this computation. Fig. 4.15 shows the distributions of the structure response along the depth. The trends of the effects of the pore fluid are very similar to, but are less pronounced than, those seen for the plane strain structure. Fig. 4.16 shows the distribution of the maximum stresses in the soil at the structure along the depth. The maximum stresses herein are the maximum values of the stresses along the circumference of the cylindrical structure.

The analytical expression is formulated for the lateral stiffness of fluid-saturated porous medium under the cylindrical plane strain conditions, in which freedoms of displacements are only in the horizontal r and θ directions. Such a stiffness has been developed previously for a single phase solid (Novak et al., 1978). This approach is a well accepted approximation for a single phase solid to compute the dynamic pile response at frequencies above the fundamental resonant frequency of the soil deposits. The pile-head stiffness for the lateral motion is computed with this analytical expression of soil stiffness assuming EI of pile equal to 2×10^4 tf.m 2 . Fig. 4.17b shows the variation of the stiffness computed with this approximate expression of the soil stiffness and that

computed by the developed thin layer method. For the comparison, those stiffnesses are also shown for a single phase solid soil in Fig. 4.17a. This approximation for the fluid-saturated porous soil medium yields the accuracy more or less very similar to the accuracy previously found for a single phase solid.

5. CONCLUSIONS

A semi-analytical method is developed for dynamic response analysis of fluid-saturated porous medium. The plane strain conditions in the Cartesian coordinate system and three-dimensional conditions in the cylindrical coordinate system are considered. With a separation of variables, the method adopts the finite element discretization only along the depth and analytical forms in the lateral direction. The method is found to be numerically very efficient and therefore is particularly useful for two-phase mixture problems since they generally require computations much larger than those for single phase medium problems.

The pore fluid in the soil mass affects the dynamic response of the soil deposits by not simply increasing the stiffness of the soil but also by coupling the soil skeleton motions with pore fluid motions. All those are affected by the loading rate relative to the pore fluid diffusion rate, boundary conditions and stress gradient. The coupling effects are more predominant for higher permeability soils. When the permeability is low, a mode wave transmitted primarily to the fluid is distinctly different from those primarily transmitted to the soil skeleton and decays very quickly with distance. Under the static and drained conditions, the submerged soil response is identical to the dry soil response. Under the dynamic condition, however, the transient pore fluid redistribution depends on the rate of loading and permeability of the soil. The response of submerged soil is closer to the undrained conditions when the combination of those are less favorable for pore fluid movement. The larger the pore fluid motions relative to the soil skeleton are, the higher the damping is generated. When the soil is submerged below the water table, the water above the soil deposits can affect the dynamic response of the soil significantly and thus has to be taken into account in the dynamic response analysis of submerged soil under the water. It is well known that the Winkler model based on the cylindrical plane strain

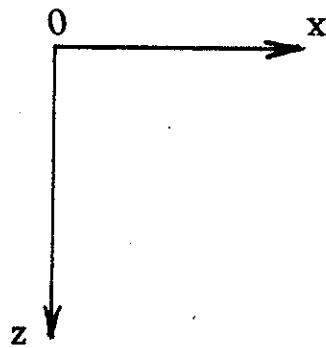
conditions can produce the dynamic pile response reasonable well for piles in a single phase soil medium. It is found that the pore fluid and soil skeleton coupling effects in the transient motions also can be reasonably well produced by such Winkler model formulated for a fluid-saturated porous medium.

REFERENCES

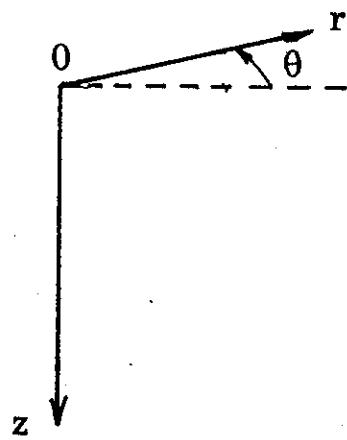
- Biot, M. A., 1956, "Theory of Wave Propagation of Elastic Waves in a Fluid-Saturated Porous Solid, Part I: Low Frequency Range and Part II High Frequency Range," J. Acoust. Soc. America, Vol. 28, pp. 168-191.
- Dresiewicz, H., 1970, "The effect of boundaries on Wave Propagation in a Liquid-filled Porous Solid," Bull. Seism. Soc. America, Vol 15, pp. 599-607.
- Foda, M. and Mei, C. C., 1982, "Boundary Layer Theory for Rayleigh Waves in a Porous, Fluid-Filled Half Space," Proc. Soil Dynamics and Earthquake Engineering Conference, Southampton, July, pp. 239-249.
- Ghaboussi, J. and Wilson, E. L., 1978, "Variational Formulation of Dynamics of Fluid-Saturated Porous Elastic Solids," J. Eng. Mech. Div., ASCE, EM4, pp.947-963.
- Jones, J. P., 1961, "Rayleigh Waves in a Porous Elastic Fluid Saturated Solid," J. Acoust. Soc. America, Vol. 33, pp. 969-962.
- Kausel, E. and Roesset, J. M., 1975, "Dynamic Stiffness of Circular Foundations," J. Eng. Mech. Div., ASCE, Vol. 101, EM6, pp. 771-785.
- Lysmer, J. and Waas, G., 1972, "Shear Waves in Plane Infinite Structures," J. Eng. Mech. Div., ASCE, Vol. 98, EM1, pp. 85-105.
- Novak, M., Nogami, T. and Aboul-Ella, F., 1978, "Dynamic Soil Reaction for Plane Strain Case," J. Eng. Mech. Div., ASCE, Vol.104, No. EM4, pp. 953-959.
- Prevost, J. H., 1982, "Nonlinear Transient Phenomena in Saturated Porous Media," Comp. Mech. Appl. Mech. Eng. Vol. 20, pp. 3-8.
- Simon, B. R., Zienkiewicz, O. C. and Paul, D. K., 1986, "Evaluation of u-w and u- π Finite Element Methods for the Dynamic Response of Saturated Porous Media Using One-Dimensional Models," Int. Numer. Anal. Methods Geomech., Vol. 10, pp. 461-482.
- Simon, B.R., Zienkiewicz, O. C. and Paul, D. K., 1984, "An Analytical Solution for the Transient Response of Saturated Porous Elastic Solids," Int. Numer. Anal. Methods Geomech., Vol. 8, pp. 381-398.
- Tajimi, H. and Shimomura, Y., 1976, "Dynamic Analysis of Soil-Structure Interaction by the Then Layered Element Method," Transactions of the Architectural Institute of Japan, Vol.243, pp.41-51 (in Japanese).
- Zienkiewics, O. C. and Shiomi, T., 1984, "Dynamic Behavior of Saturated Porous Media: Generalized Biot Formulation and Numerical Solution," Int. J. Numer. Anal. Methods Geomech., Vol. 8, pp. 71-96.

TABLE 1
Permeabilities of Soils

permeabilities (m/s)				
10^{-0}	10^{-2}	10^{-5}	10^{-9}	10^{-11}
clean gravels	clean sands	silts, mixtures of sands, silts and clays		clays



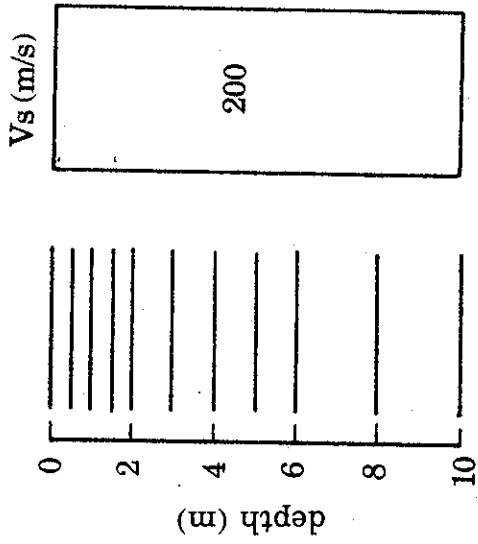
Cartesian coordinates



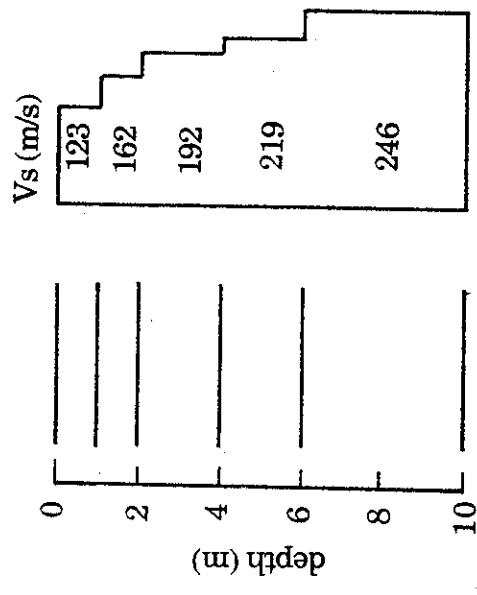
Cylindrical coordinates

Fig. 2.1 Cartesian and cylindrical coordinate systems

(a) Homogeneous profile



(b) Inhomogeneous Profile



$$K_s = 3.7 \times 10^6 \text{ tf/m}^2$$

$$\nu = 0.25$$

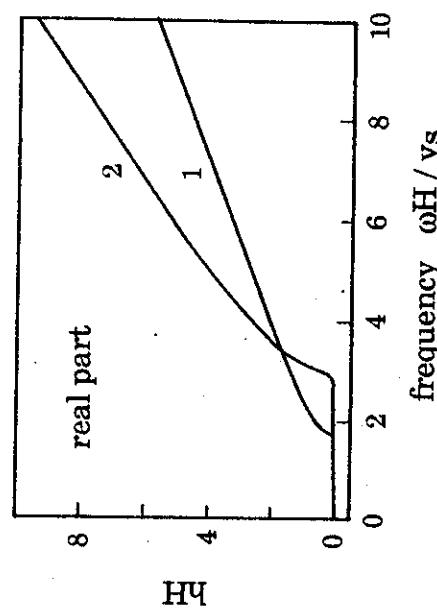
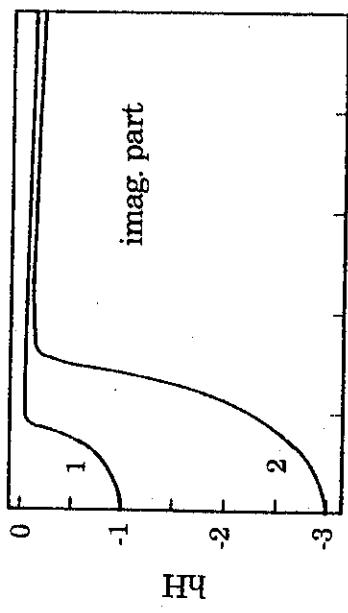
$$K_f = 2.08 \times 10^5 \text{ tf/m}^2$$

$$D = 2\%$$

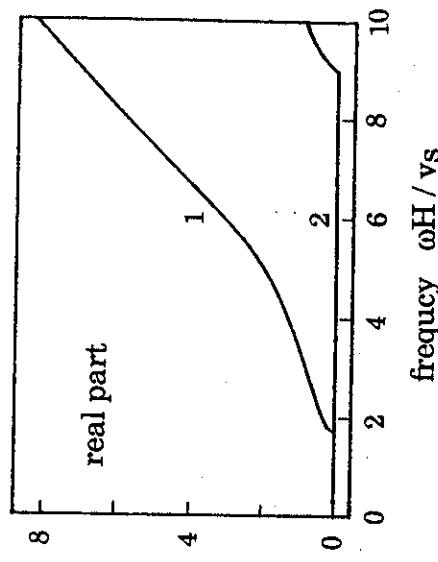
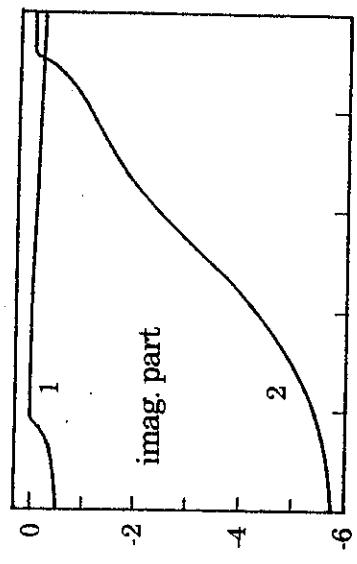
$$n = 0.375$$

$$\gamma = 2.6 \text{ tf/m}^3$$

Fig. 4.1 Homogeneous and inhomogeneous soil profiles

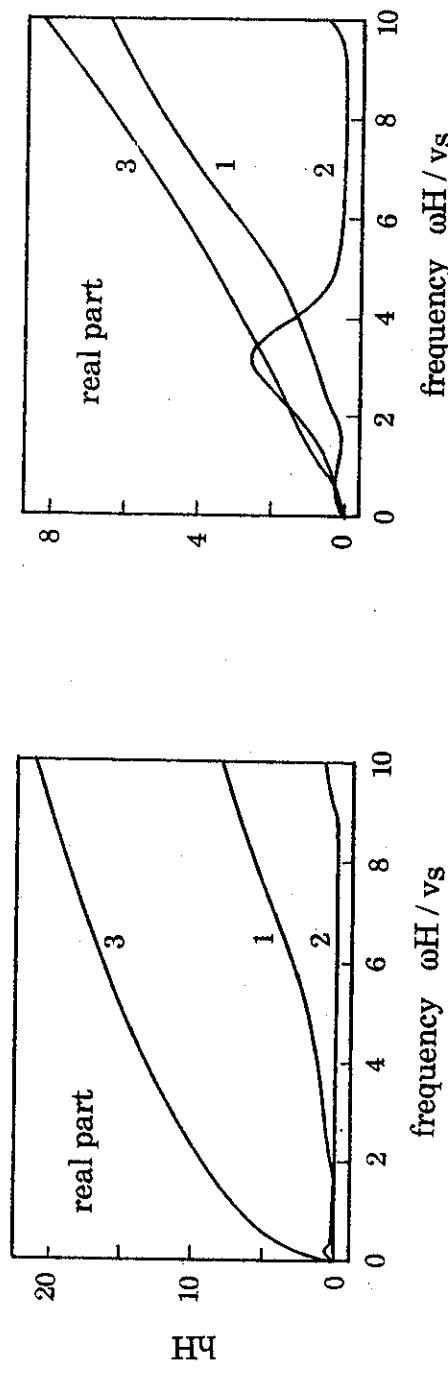
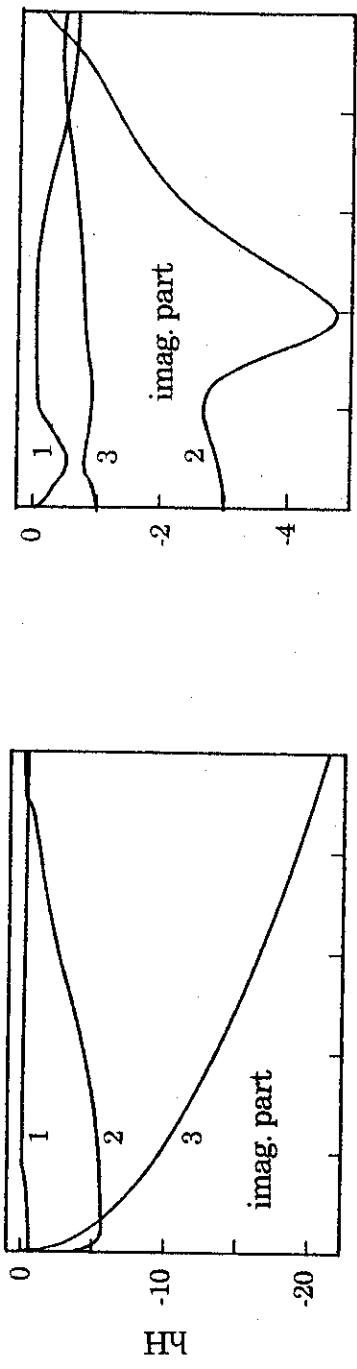


(a) Dry soil



(b) Submerged soil in undrained condition

Fig. 4.2a and 2b Wave number dispersion curves



(c) Submerged soil in drained condition ($k = 10^{-3} \text{ m/s}$)

(d) Submerged soil in drained condition ($k = 10^{-1} \text{ m/s}$)

Fig. 4.2c and 2d Wave number dispersion curves

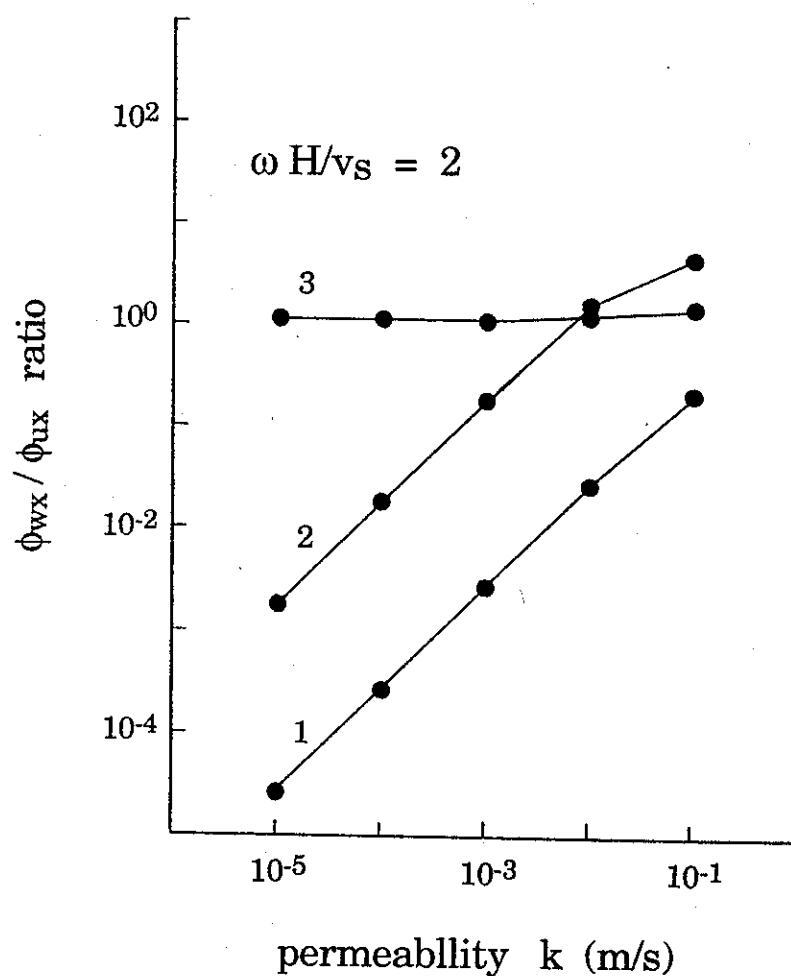
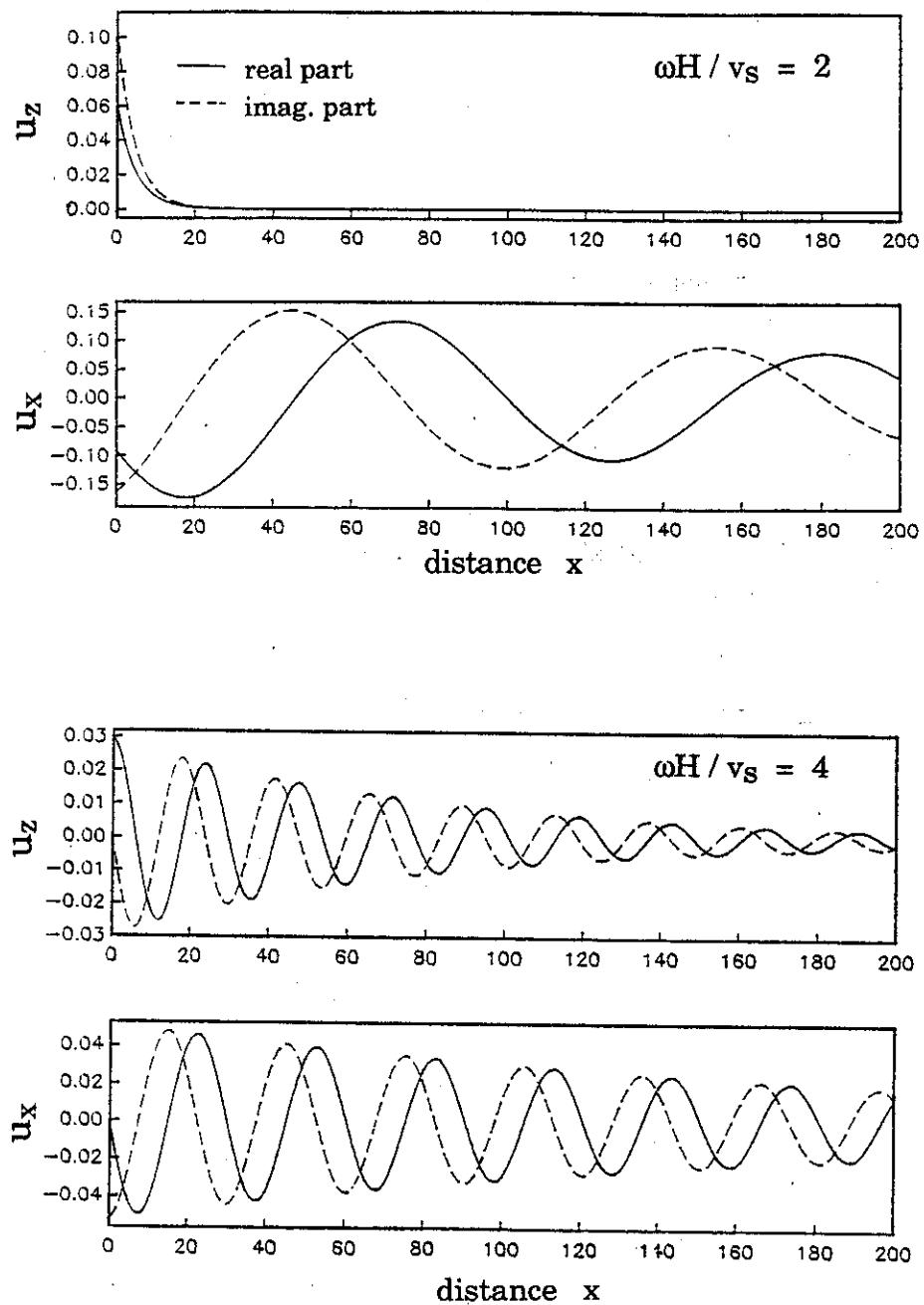
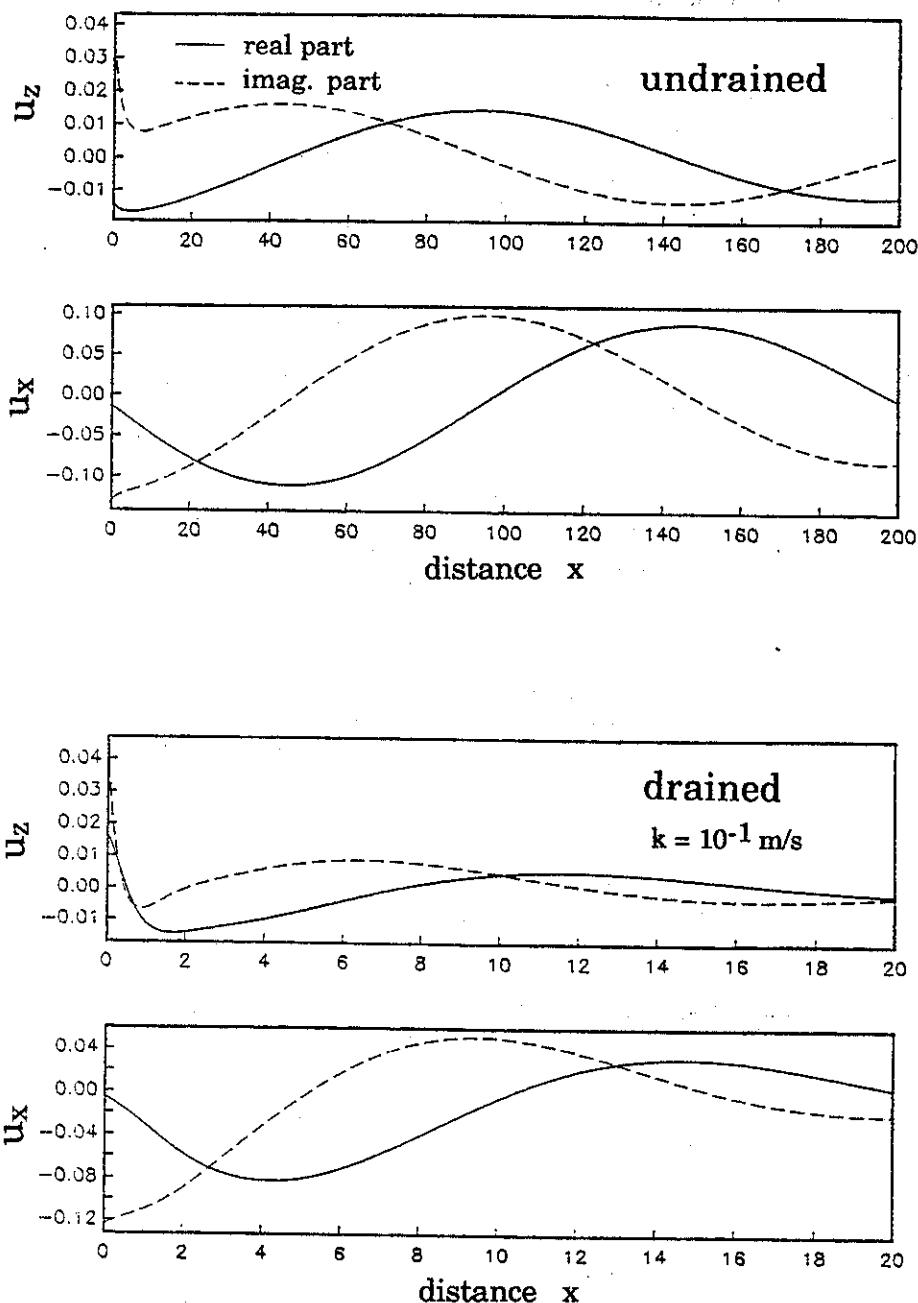


Fig. 4.3 Variation of ϕ_{wx} / ϕ_{ux} ratio with permeability



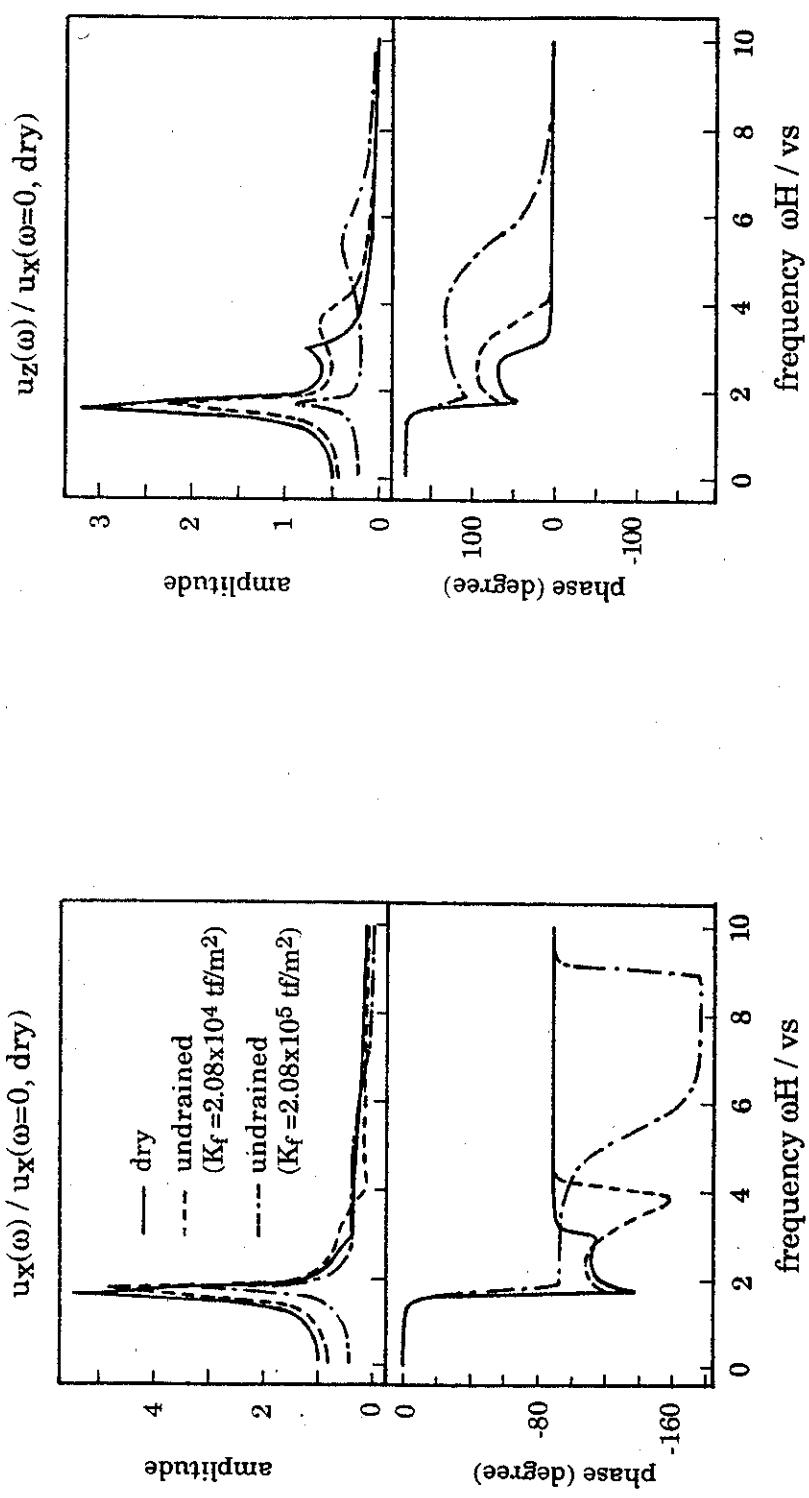
(a) Dry soil ($\omega H / v_s = 2$ and $\omega H / v_s = 4$)

Fig. 4.4a Displacement amplitudes along the ground surface



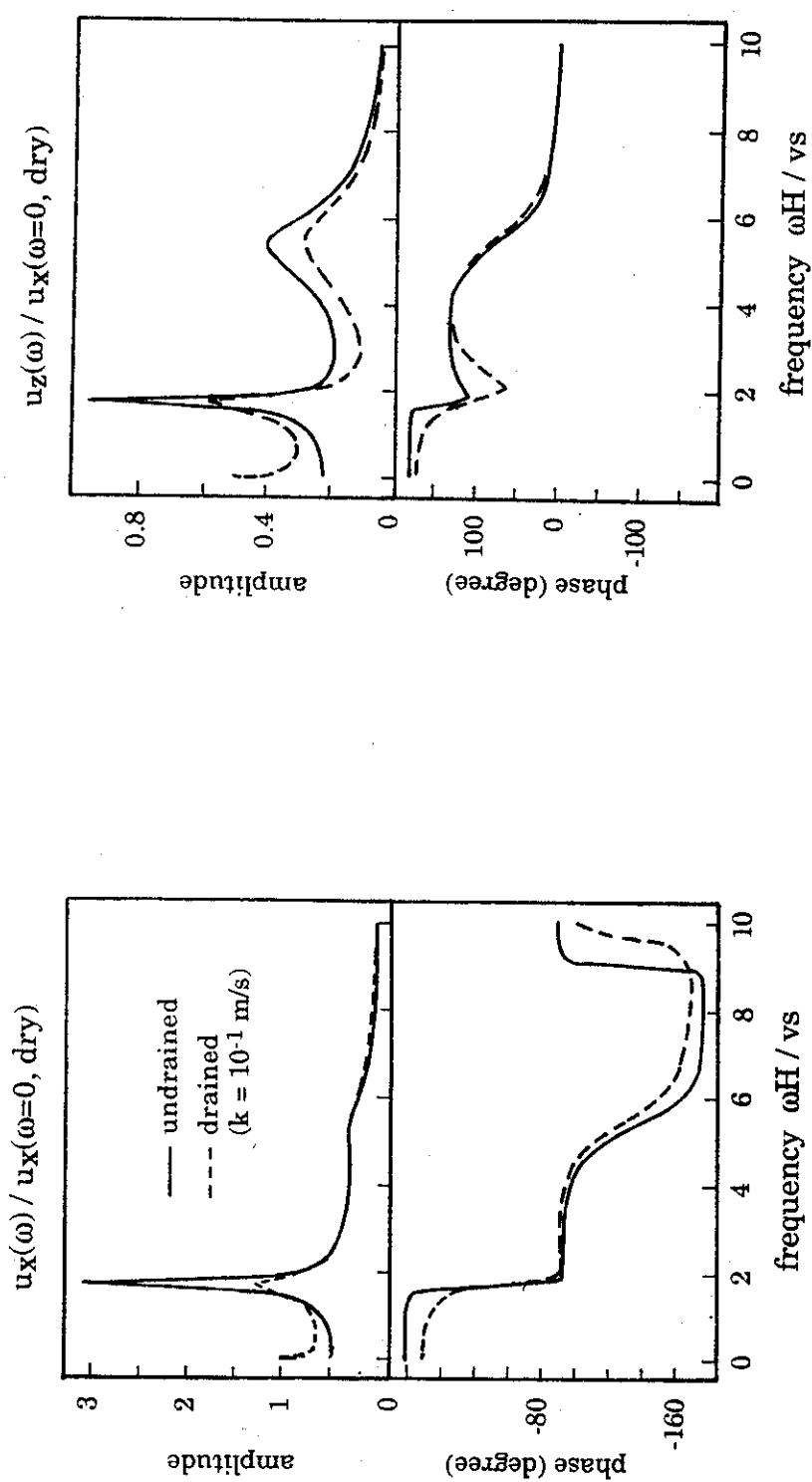
- (b) Submerged soil in undrained and drained condition
 $(\omega H / v_s = 2)$

Fig. 4.4b Displacement amplitudes along the ground surface



(a) dry soil and submerged soil in undrained condition

Fig. 4.5a Displacements at the top of the vertical wall subjected to lateral vibration



(b) submerged soil in undrained and drained condition

Fig. 4.5b Displacements at the top of the vertical wall subjected to lateral vibration

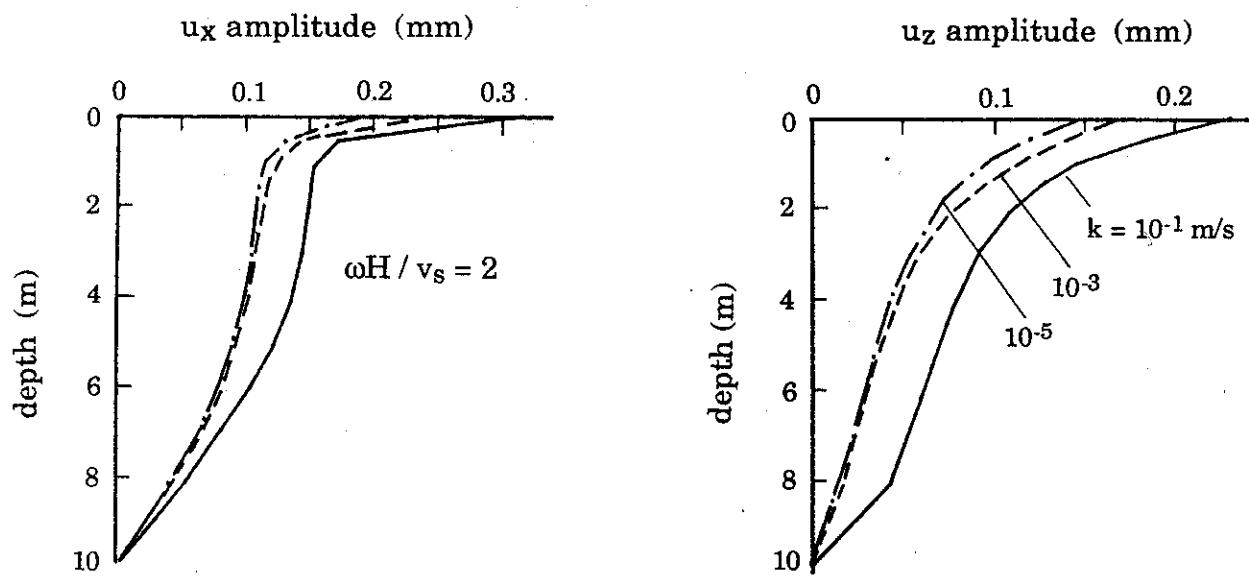


Fig. 4.6 Displacement response amplitudes of the wall along depth

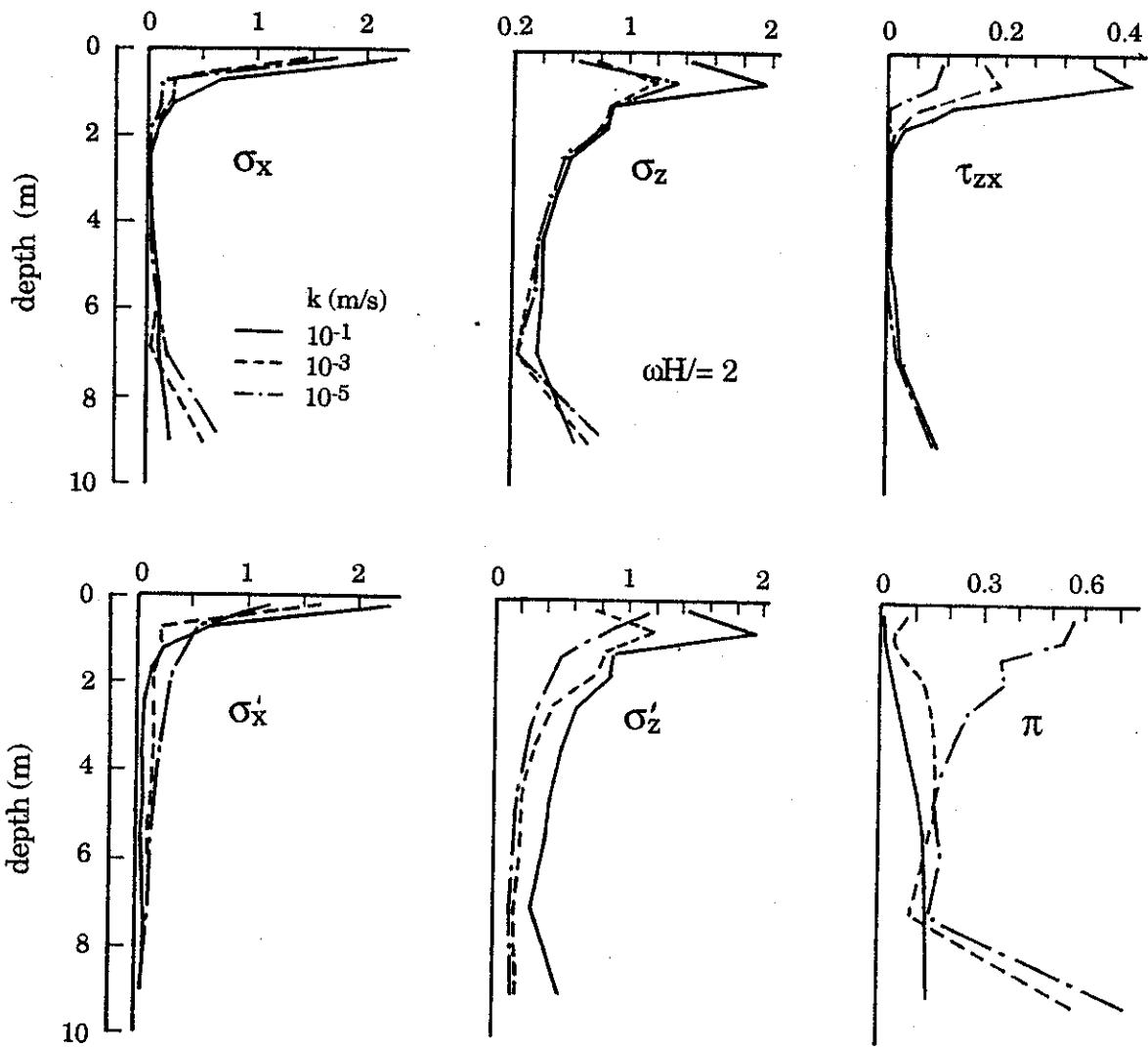


Fig. 4.7 Pressure response amplitudes at soil-wall interface along the wall

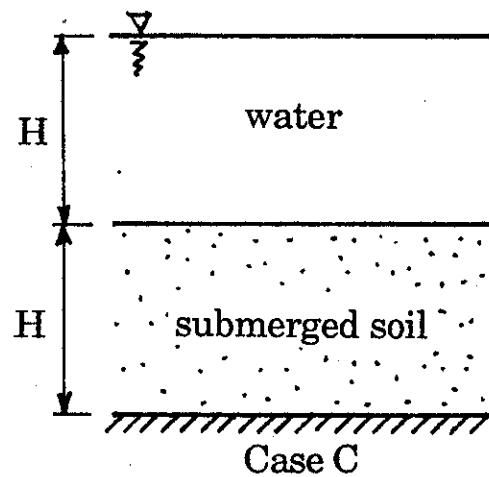
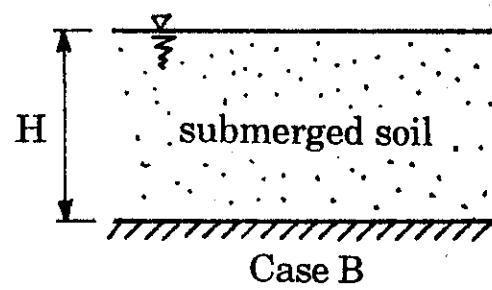
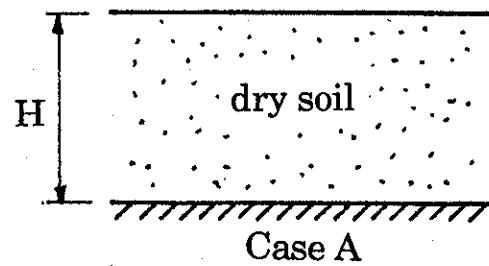


Fig. 4.8 Various cases considered for examples

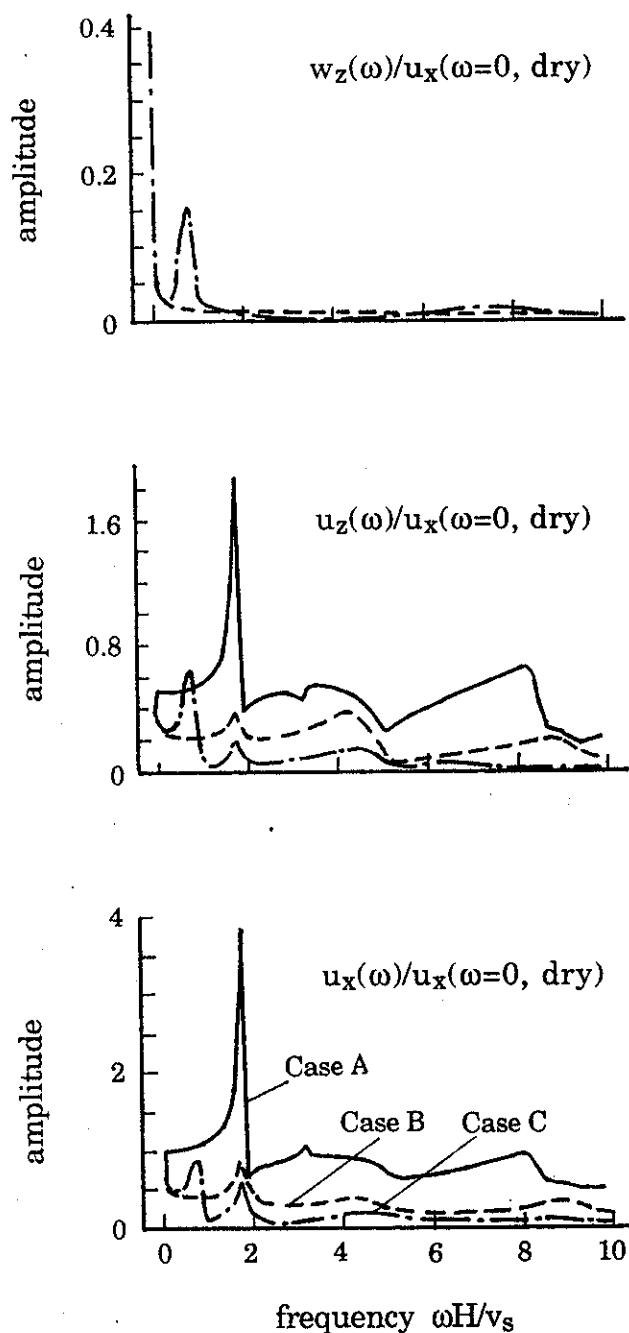
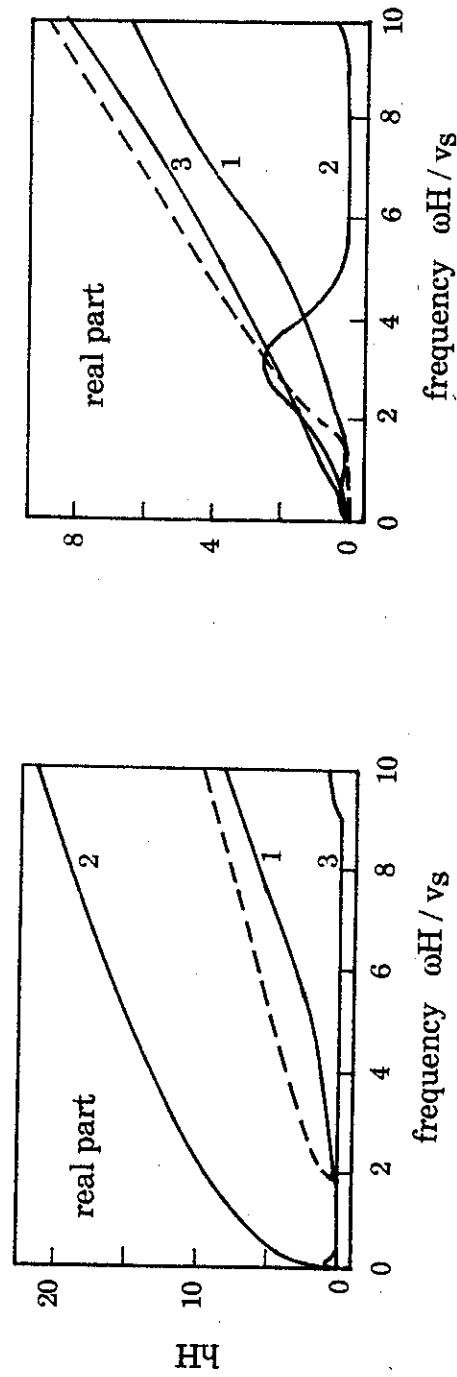
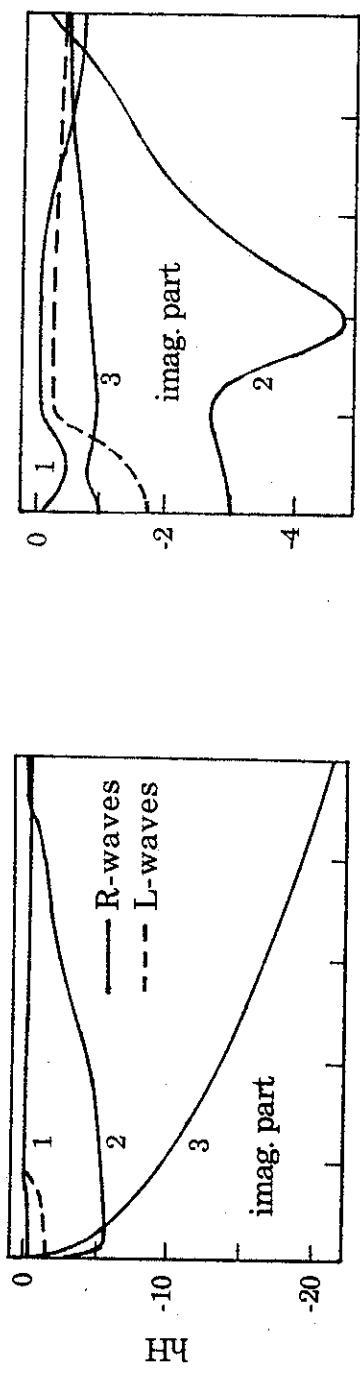


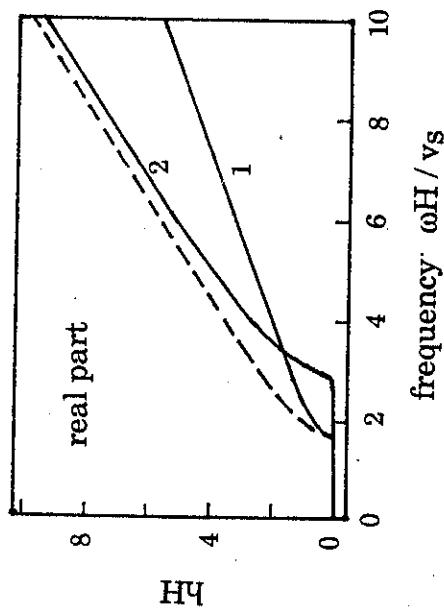
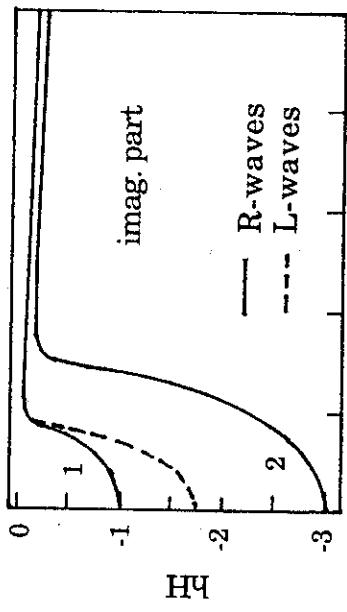
Fig. 4.9 Lateral displacement amplitudes at the top of the wall subjected to lateral vibration



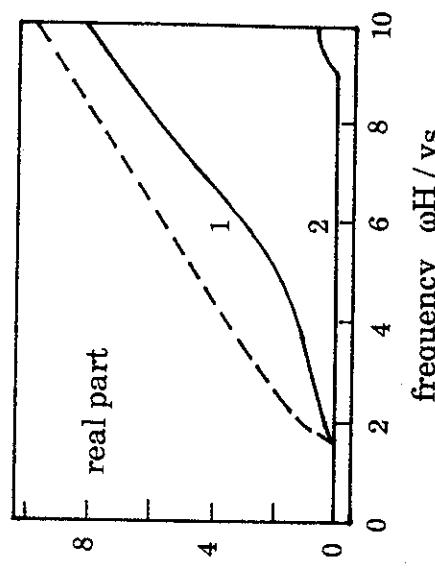
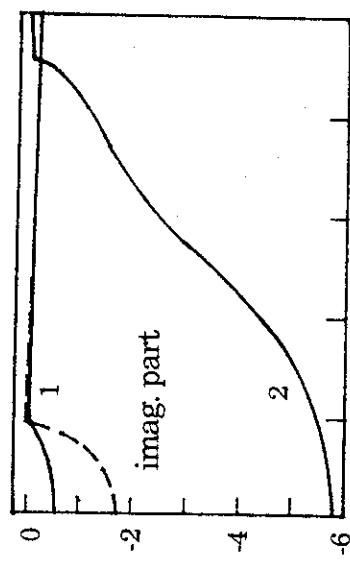
(c) Submerged soil in drained condition ($K = 10^{-3} \text{ m/s}$)

(d) Submerged soil in drained condition ($K = 10^{-1} \text{ m/s}$)

Fig. 4.10c and 10d Wave number dispersion curves



(a) Dry soil



(b) Submerged soil in undrained condition

Fig. 4.10a and 10b Wave number dispersion curves

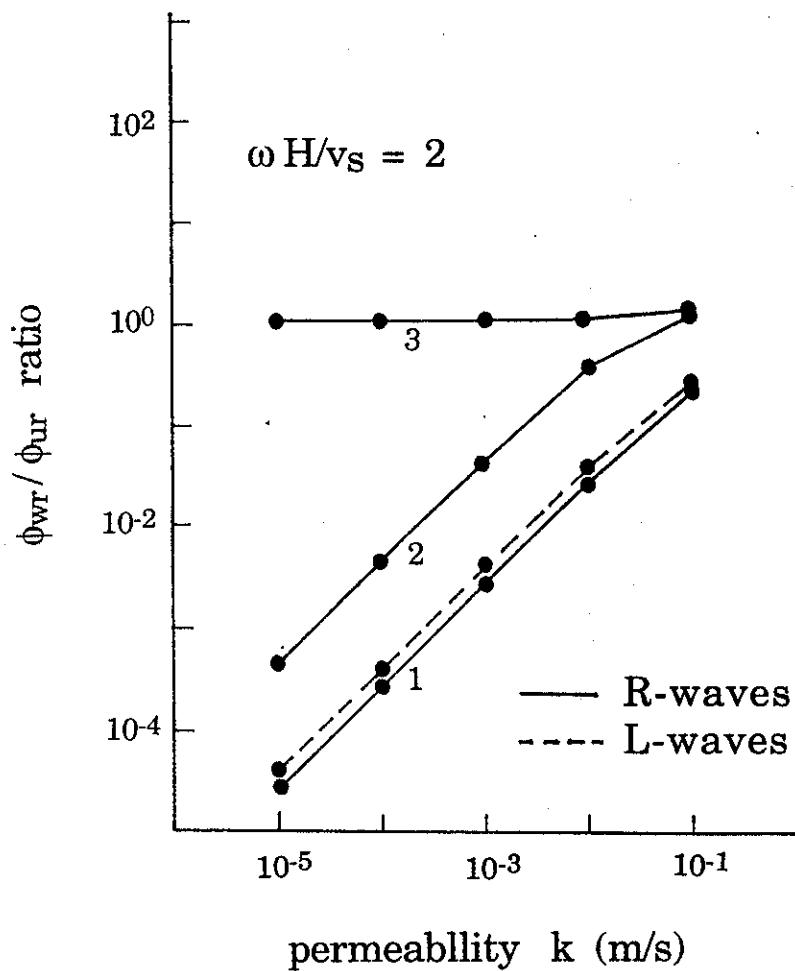
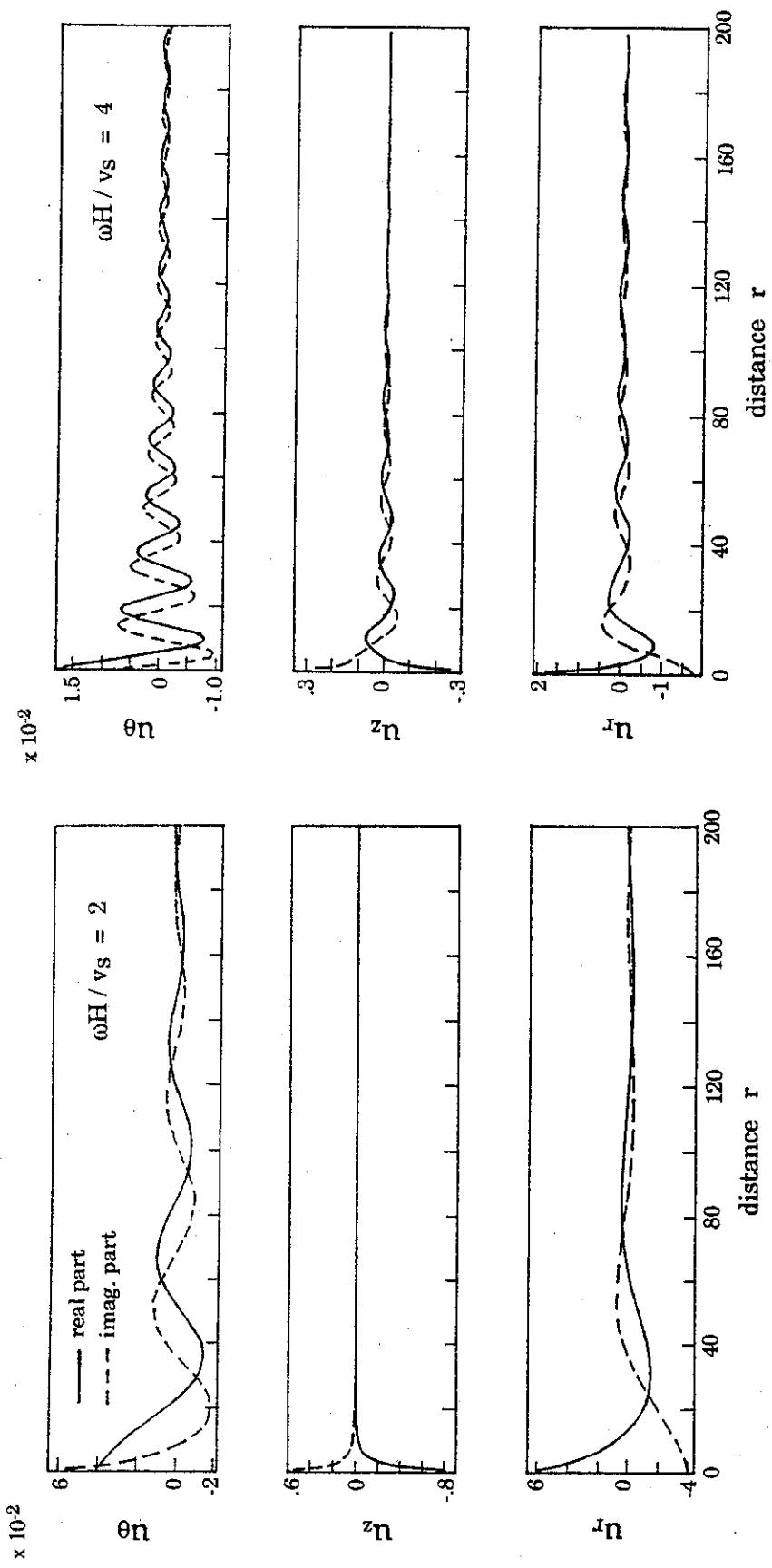
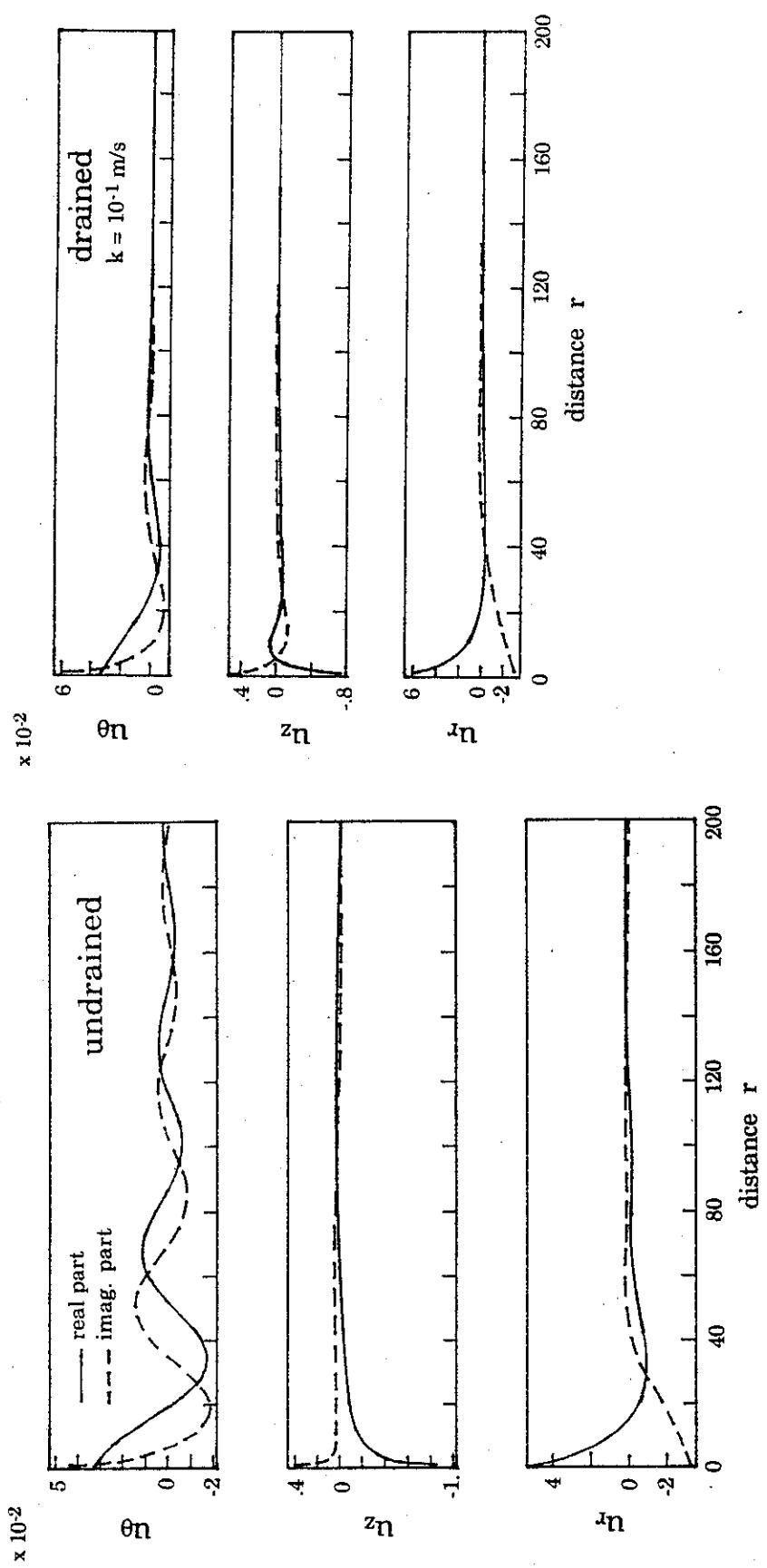


Fig. 4.11 Variation of ϕ_{wr} / ϕ_{ur} ratio with permeability



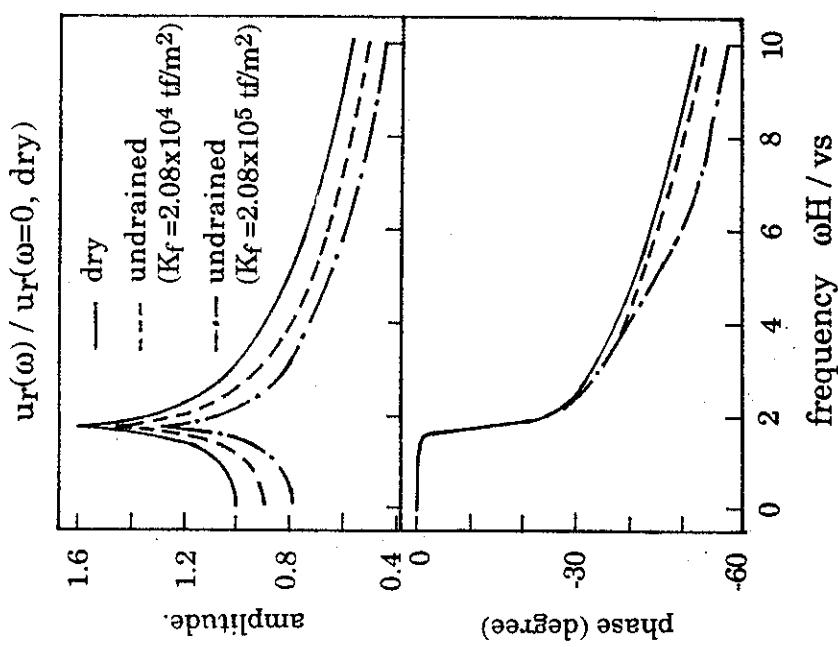
(a) Dry soil ($\omega H / v_s = 2$ and $\omega H / v_s = 4$)

Fig. 4.12a Displacement amplitudes along the ground surface



(b) Submerged soil in undrained and drained condition
($\omega H / v_s = 2$)

Fig. 4.12b Displacement amplitudes along the ground surface



(a) dry soil and submerged soil in undrained condition

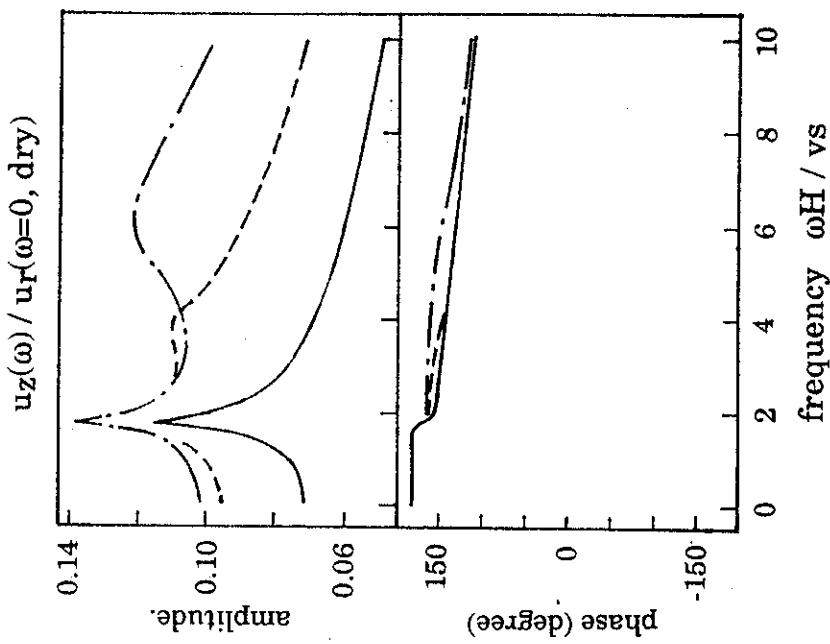
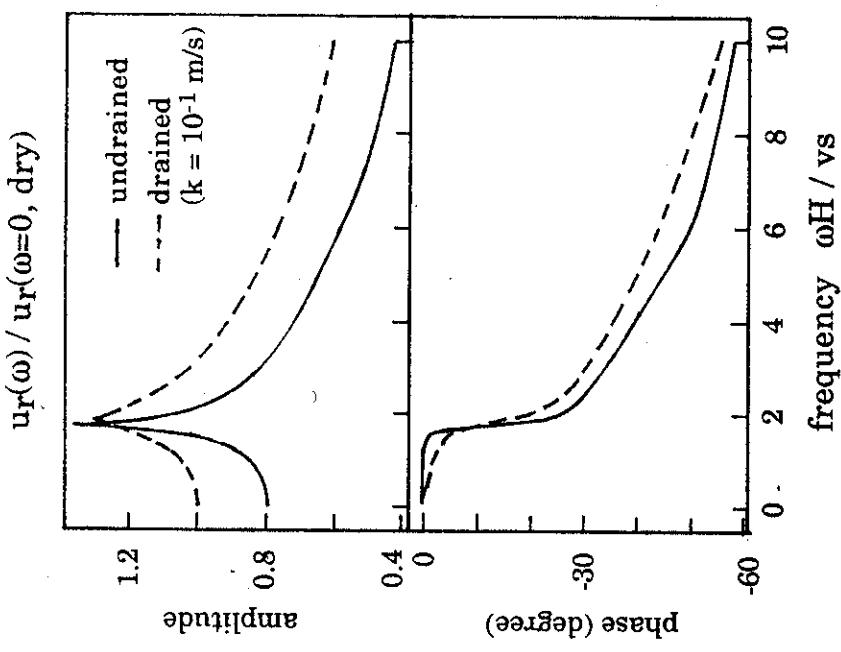


Fig. 4.13a Displacements at the top of cylindrical body subjected to lateral vibration



(b) submerged soil in undrained and drained condition

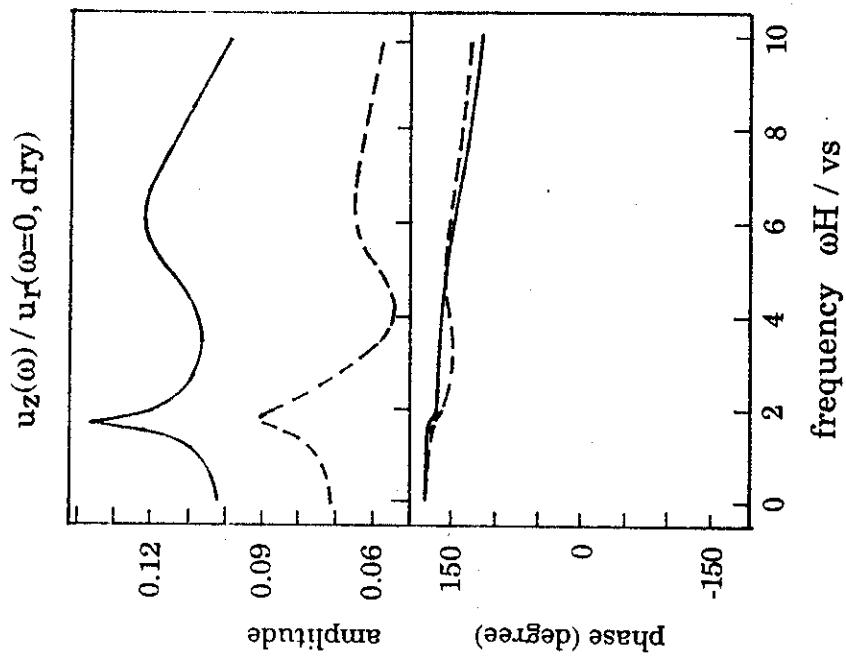


Fig. 4.13b Displacements at the top of cylindrical body subjected to lateral vibration

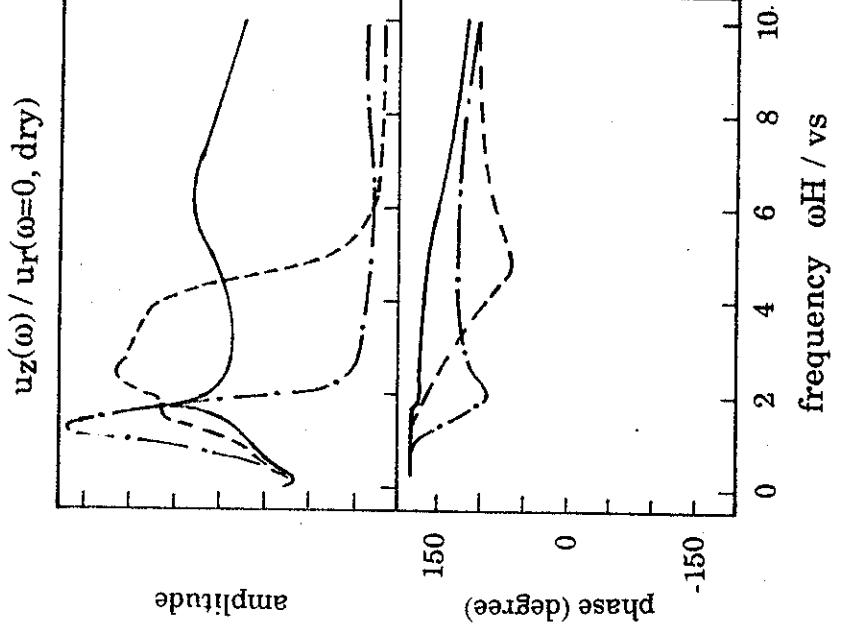
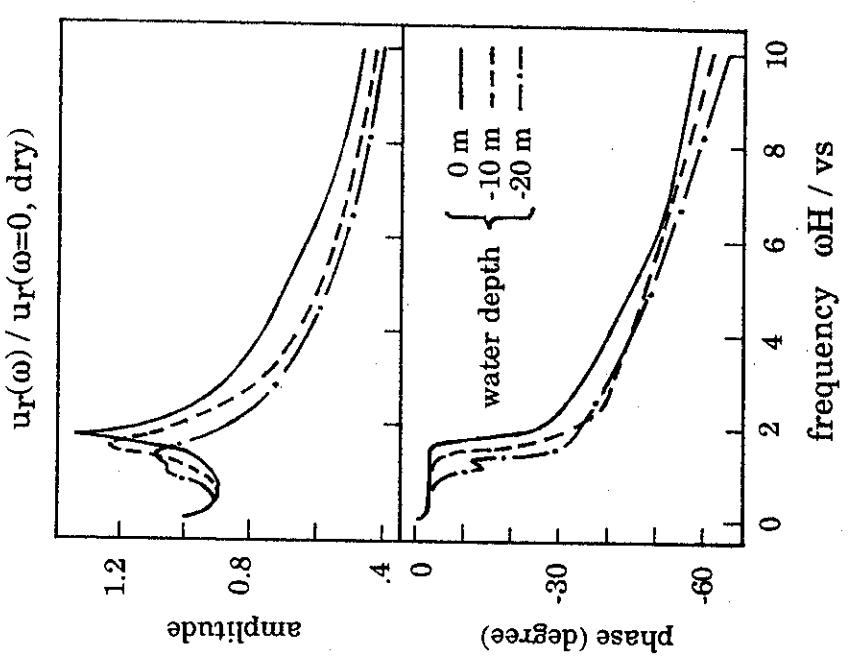


Fig. 4.14 Effects of water depth on displacement of cylindrical body subjected to lateral vibration

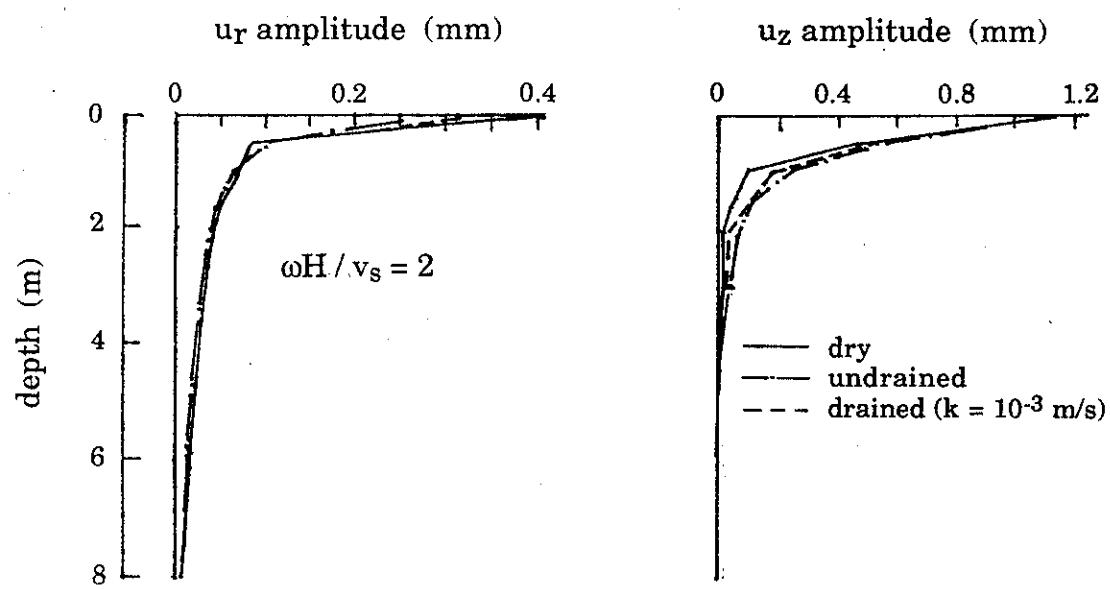
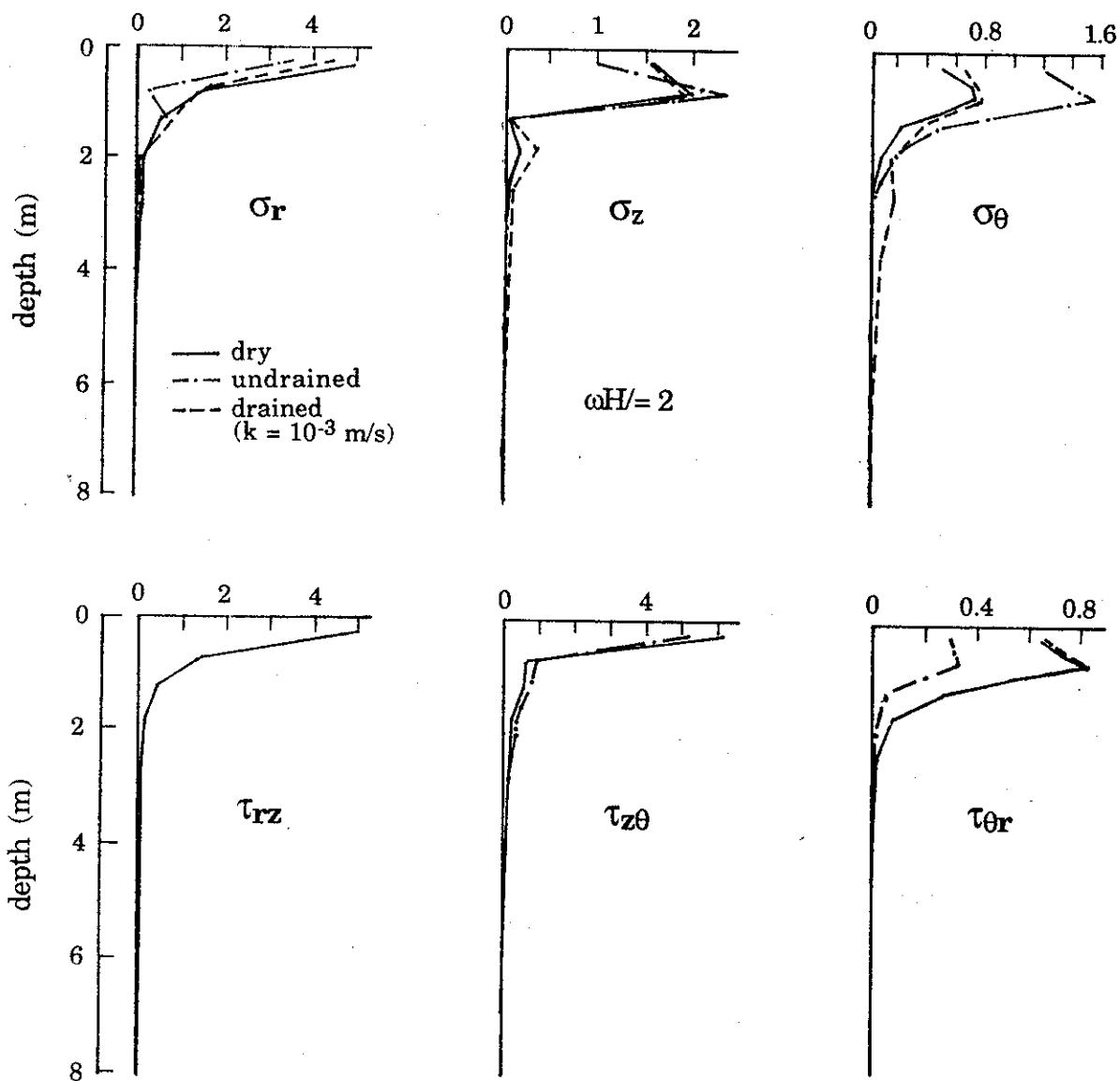
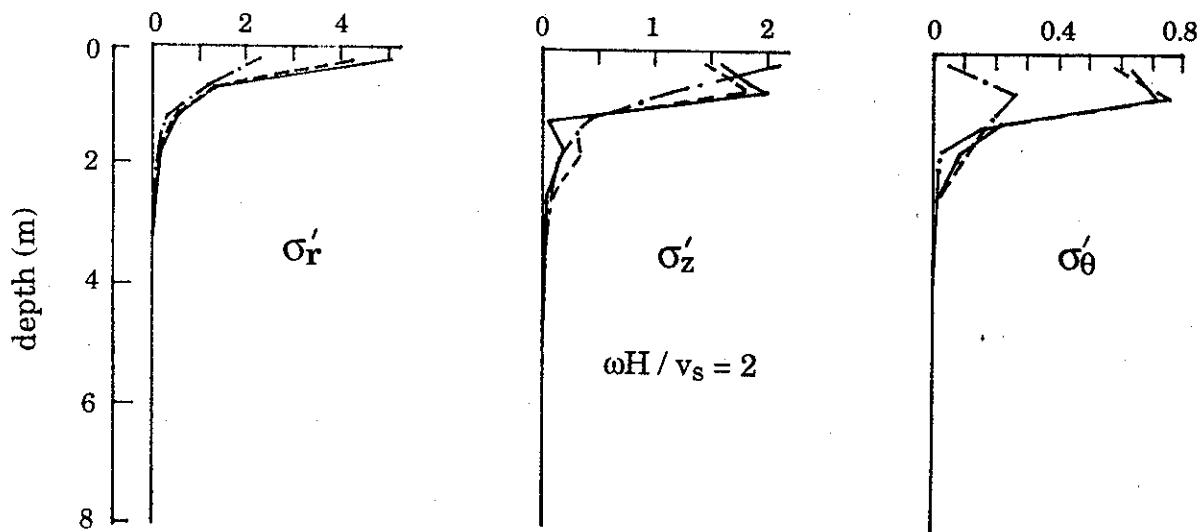
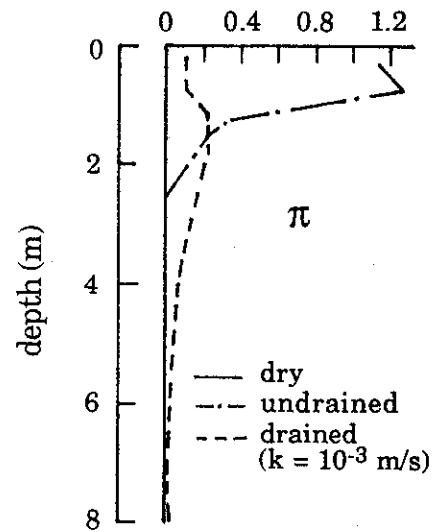


Fig. 4.15 Displacement response amplitudes of the cylindrical body along depth



(a) Distributions of total stresses

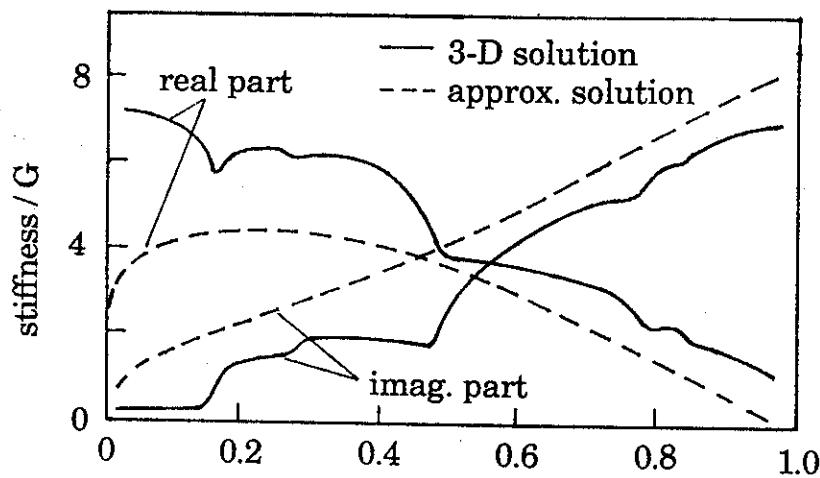
Fig. 4.16a Pressure response amplitudes along the cylindrical body



(b) Distributions of porewater pressure and effective stresses

Fig. 4.16b Pressure response amplitudes along the cylindrical body

(a) dry soil



(b) drained condition ($k = 10^{-5}$ m/s)

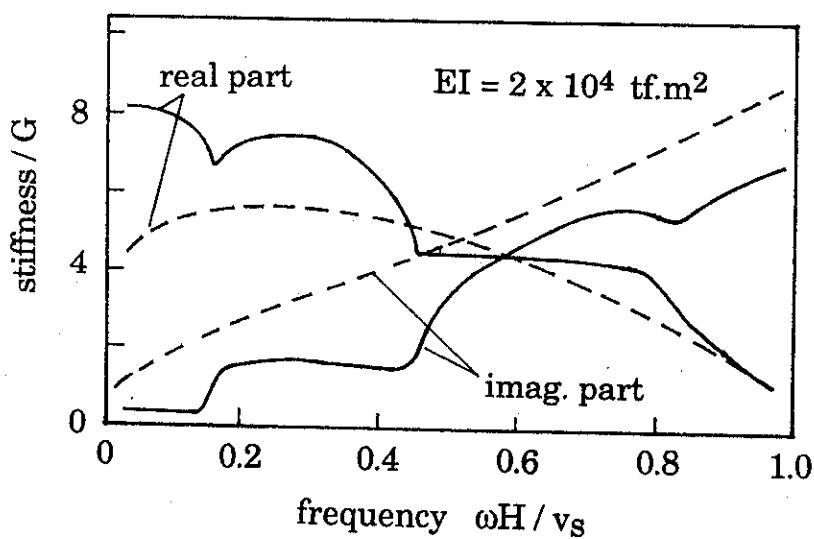


Fig. 4.17 Dynamic stiffnesses of the cylindrical body computed by present solution and approximate solution

APPENDIX A

Computer Program "PST"

A program to compute dynamic response of submerged soil bed
for two-dimensional plane strain conditions in Cartesian coordinates

```

1: C
2: C
3: C
4: C
5: C
6: C
7: C
8: C
9: C
10: C
11: C
12: C
13: C
14: C
15: C
16: C
17: C
18: C
19: C
20: C
21: C
22: C
23: C
24: C
25: C
26: C
27: C
28: C
29: C
30: C
31: C
32: C
33: C
34: C
35: C
36: C
37: C
38: C
39: C
40: C
41: C
42: C
43: C
44: C
45: C
46: C
47: C
48: C
49: C
50: C
51: C
52: C
53: C
54: C
55: C
56: C
57: C
      PROGRAM *****PST***** CODED BY KAZAMA 1990 3/13
      MAIN PROGRAM FOR CALCULATE 2D PLANE STRAIN CASE
      DRAIN CASE
      IMPLICIT COMPLEX*16(Z)
      PARAMETER (MLY=10)
      PARAMETER (M4=(MLY+1)*4)
      PARAMETER (M3=MLY*3)

      COMMON /NOCH1/ ZA(M4,M4),ZD(M4,M4)
      COMMON /NOCH2/ ZK(M4,M4),ZCM(M4,M4)
      COMMON /NOCH3/ ZPH1(M4,M3),ZPTN(M3,M3)
      COMMON /NOCH4/ ZB(M4,M4),ZRB(M4/2,M4/2)
      COMMON /NOCH5/ ZCINV(M4,M4)
      DIMENSION IND1(MLY,4),FRG(100),NXOUT(10),ZHS(M3),NFORC(MLY*2)
      COMMON /GROUND/ ZAA(12),ZG(12),ZAL(12),ZG(12),PERM(12),
      CNE(12),RO(12),R0F(12),E1(12),DEP(12),DEM(12)
      COMMON /DMYL/ ZDUMY1(3200)
      COMMON /DMY2/ ZDUMY2(3200)
      COMMON /DMY3/ ZDUMY3(3200)
      COMMON /DMY4/ ZDUMY4(3200)
      COMMON /DMY5/ ZDUMY5(1000)
      COMMON /COND1/ KC,KP,KG(4),NLAY,MJYD,MRI1,NFLG1,IND2(2)
      COMMON /COND2/ GAL,PAI,DEPTH,NOT,VGB,NFRQ,DO
      COMMON /CONST/ CHARACTER INFL*20,QUTFL*20,YNF*1
      COMMON /PLOTD1*20,PLOTD2*20,PLOTD3*20,PLOTD4*20
      DATA PLOTD1/*>QUT>PLOUT.FRG
      DATA PLOTD2/*>QUT>PLOUT.MODE
      DATA PLOTD3/*>QUT>PLOUT.DISP
      DATA PLOTD4/*>QUT>PLOUT.SIGMA
      DATA QUTFL/*>PST.DUTPUT
      KG(1)=11
      KG(2)=12
      KG(3)=13
      KG(4)=14
      DIMENSION /DMY1/... 32MLY**2
      /DMY2/... 32MLY**2
      /DMY3/... 32MLY**2
      /DMY4/... 32MLY**2
      /DMY5/... 48MLY
      /DMY6/... 32MLY**2+64MLY*32
      INFORMATION ABOUT VARIABLES
      PARAMETER MLY,... MAXIMUM USAGE LAYER NUMBER
      M4,... DIMENSION OF THE MATRIX
      DIMENSION 2D ARRAY
      ZA(M4,M4)..... MATRIX A
      ZB(M4,M4)..... MATRIX B
      ZCM(M4,M4)..... MATRIX C
      ZD(M4,M4)..... MATRIX D

```

```

58: C ZK(MA,MA)..... MATRIX W
59: C ZRB(MA,2,MA/2)..... MATRIX RB BEAM STIFFNESS
60: C ZPHI(MA,MA)..... MATRIX PHAI CONSISTED OF EIGEN VECTOR
61: C ZCINV(MA,MA)..... INVERSE MATRIX OF C
62: C ZPINV(MA,MA)..... INVERSE MATRIX OF PHAI
63: C ----- ID ARRAY -----
64: C EI(12)..... BENDING STIFFNESS OF YAITA (TF*M**2)
65: C ZAA(12)..... A VALUE CALCULATED BY LAME CONSTANTS AND ALPHA, Q
66: C ZG(12)..... SHEAR STIFFNESS OF THE GROUND
67: C ZAL(12)..... ALPHA VALUE OF BIOT EQUATION
68: C ZQ(12)..... Q VALUE OF BIOT EQUATION
69: C H(12)..... LAYER HEIGHT (H)
70: C PERM(12)..... PERMEABILITY COEFFICIENT OF THE GROUND
71: C ----- DIVIDED BY ROF*QAL
72: C CNE(12)..... POLOSITY
73: C 'RO(12)..... UNIT WEIGHT OF THE SOLID AND FLUID-MIXTURE
74: C 'ROF(12)..... UNIT WEIGHT OF THE FLUID MATERIAL
75: C
76: C INPUT AND OUTPUT DEVICE CODE SETTING
77: C (KC) (KP)
78: C
79: KC=2
80: KP=10
81: WRITE(*,*)
82: WRITE(*,*), 'INPUT DATA FILENAME=?'
83: READ(*,'(A20)',INFL)
OPEN(KC,FILE=INFL)
84: OPEN(KP,FILE=OUTFL)
OPEN(KC,FILE=CARDIM(KC,KP))
85: CALL CARDIM(KC,KP)
86: READ(KC,500) VN,NCASE
87: IF(VN.EQ.'Y') WRITE(*,*) 'WAVE NUMBER AND MODE WILL BE CALCULATED'
88: IF(VN.EQ.'N') WRITE(*,*) 'DISP. AND STRESS WILL BE CALCULATED'
89: IF(NCASE.LE.0) NCASE=-1
90: WRITE(*,*) 'NCASE=',NCASE
91: WRITE(*,*) 'NCASE=',NCASE
92: OPEN(KG(1),FILE=PLT0D1)
OPEN(KG(2),FILE=PLT0D2)
OPEN(KG(3),FILE=PLT0D3)
OPEN(KG(4),FILE=PLT0D4)
93: C &&&&&&&&& SET CONSTANT FOR NORMALIZE &&&&&&&&&&&&
94: C VSB : AVERAGE SHEAR WAVE VELOCITY FOR DIMENSIONLESS FREQUENCY
95: C
96: C
97: C
98: C
99: C VSB=200.0
100: DO : DEPTH FOR DIMENSIONLESS FACTOR
101: C DO=10.0
102: C
103: C
104: C DO 1000 NSET=1,NCASE
105: C
106: C CALL INPUT(IND1,FRG,MA,NXOUT,MLY,NFDRC)
108: C----- READ DATA AND SETTING MATERIAL CONSTANT -----
109: C
110: C CALL MATRX(IND1)
111: C----- MAKE MATRIX A,D,B,K,CM,RB -----
112: C
113: C CALL RDMAT(IND1)
114: C----- REDUCE DEGREE OF FREEDOM AND REARRANGE MATRIX -----

```

```

115: C
116: DO 2 N=1, NFREQ
117: WRITE(10, 600) N, FRQ(N)
118: OMG=2.0*PAI*FRQ(N)
119: OMG2=OMG*OMG
120: WRITE(*, '(//2X, ''CASE= '', I3, '' FREQUENCY= '', FB, 2)') N, FRQ(N)
121: CALL CMATRX(ZCM, ZCINV, ZK, OMG, OME2, M4)
122: C-----MAKE MATRIX C AND INVERSE OF C -----
123: C
124: CALL EIGEN(ZA, ZB, ZHS, DEPTH)
125: C-----SOLVE EIGEN VALUE PROBLEM -----
126: C
127: WRITE(*, *) DEPTH
128: XA=OMG*DO/VSB
129: IF(YN, EQ, 'Y') THEN
130: DO 20 J=1, MRI
131: WRITE(KG(1), 650) XA, DO*ZHS(J), ZPHI(1, J), ZPHI(2, J), ZPHI(3, J)
132: +
133: 20 CONTINUE
134: CALL WAVEOT(ZHS, ZPHI)
135: ELSE
136: C
137: CALL RMATRIX(ZA, ZPHI, ZHS, ZD)
138: C-----MAKE STIFFNESS MATRIX OF THE GROUND -----
139: C
140: CALL SOLVE(ZRB, NXOUT, IND1, ZPHI, ZPINV, ZHS, OMG, INFOC)
141: C-----SOLVE PROBLEM AND RESULTS OUTPUT -----
142: C
143: IF(N, NE, 1, OR, NSET, NE, 1) GO TO 100
144: ZUX0=ZUX1
145: ZSTKU=1.0/ZUX0
146: 100 CONTINUE
147: ZDYKU=1.0/ZUX1
148: ZCOEF=ZDYKU*REAL(ZSTKU)
149: Z1=ZUX1/ZUX0
150: Z2=ZUZ1/ZUX0
151: Z3=ZWZ1/ZUX0
152: A1=AIMAG(Z1)
153: A2=REAL(Z1)
154: APX=57.29578*ATAN2(A1, A2)
155: A1=AIMAG(Z2)
156: A2=REAL(Z2)
157: APZ=57.29578*ATAN2(A1, A2)
158: APWZ=0.0
159: IF(IND2(2), EQ, 0) GO TO 110
160: A1=AIMAG(Z3)
161: A2=REAL(Z3)
162: APWZ=57.29578*ATAN2(A1, A2)
163: 110 WRITE(KG(1), 650) OMG*DO/VSB, ABS(Z1), APX, ABS(Z2), APZ,
164: +
165: END IF
166: 2 CONTINUE
167: 1000 CONTINUE
168: CLOSE(KC, STATUS='KEEP')
169: C
170: CLOSE(KC(1), STATUS='KEEP')
171: CLOSE(KC(2), STATUS='KEEP')

```

```

172:      CLOSE(KG(3), STATUS='KEEP')
173:      CLOSE(KG(4), STATUS='KEEP')
174:      WRITE(*, ('//', 'OUTPUT LIST FILE=//', A20)) QUTFL
175:      WRITE(*, ('//', 'GRAPHIC OUT FILE=//', A20)) PLOTD1, PLOTD2, PLOTD3
176:      +
177:      PLOTD4
178:      WRITE(*, ('//', 'INPUT DATA FILE=//', A20)) INF1
179:      500 FORMAT(A1, I4)
180:      600 FORMAT(IHL, 'N=', 15, ' FREQUENCY=', F10.5, 'HZ')
181:      650 FORMAT(F10.5, 10E15.5)
182:      660 FORMAT(F10.5, 2E15.5, F10.5)
183:      STOP
184:      END
185:      ****
186:      SUBROUTINE INPUT (IND1, FRQ, M4, NXOUT, MLY, NFORC)
187:      ****
188:      IMPLICIT COMPLEX*16(12)
189:      COMMON /GROUND/ ZAA(12), ZG(12), ZQ(12), H(12), PERM(12),
190:      + CNE(12), CNE(12), ROF(12), E1(12), DEP(12), DEM(12)
191:      COMMON /CONST/ KC, XP, KG(4), NFT, VSB, NFRC, DO
192:      COMMON /COND/ KC, XP, KG(4), NLAY, MJYD, MR1, NFLG1, IND2(2)
193:      COMMON /DMY1/VSM(12), DAMPM(12), PSNRM(12), PERMM(12), CNEM(12),
194:      + ROM(12), ROFM(12), CKFM(12), CKSM(12), EIM(12)
195:      C
196:      CHARACTER AM*5, TITLE*60
197:      DIMENSION FRQ(*), IND(MLY,*), AM(4), NXOUT(*), NFORC(*)
198:      C
199:      DATA DROF/1.0/, DCKF/2.08E+5/, DCKS/3.7E+6/
200:      DATA AM(1)/*SOLID//, AM(2)/*FLUID//, AM(3)/*MIX. // /
201:      DATA AM(4)/*UNDRN//,
202:      DATA DEI/20000.0/
203:      C
204:      CAL=9.8
205:      PA1=3.14159265
206:      CCC
207:      WRITE(*, *) 'SUBROUTINE INPUT NOW'
208:      READ(KC, 499) TITLE
209:      READ(KC, 500) NLAY, NFRC, NMAT, (NXOUT(I), I=1, 10), NFLG1
210:      + , (NFORC(I), I=1, NLAY*2)
211:      WRITE(KP, 650) (NFORC(I), I=1, NLAY*2)
212:      NOT=0
213:      DO 99 I=1, 10
214:      NOT=NOT+NXOUT(I)
215:      READ(KC, 501) (FRQ(I), I=1, NFRC)
216:      WRITE(KP, 600) NFRC
217:      WRITE(KP, 630) (FRQ(I), I=1, NFRC)
218:      IF(NFLG1.EQ.0) WRITE(KP, 601)
219:      WRITE(KP, 602)
220:      DO 2 NM=1, NMAT
221:      READ(KC, 501) VSM(N), DAMPM(N), PSNRM(N), PERMM(N), CNEM(N),
222:      + , ROM(N), ROFM(N), CKFM(N), CKSM(N), EIM(N)
223:      IF(ROFM(N).LT.0.1) ROFM(N)=DROF
224:      IF(CKFM(N).LT.1.0) CKFM(N)=DCKF
225:      IF(CKSM(N).LT.1.0) CKSM(N)=DCKS
226:      IF(EIM(N).LT.1.0) EIM(N)=DEI
227:      WRITE(KP, 603) N, VSM(N), DAMPM(N), PSNRM(N), PERMM(N), CNEM(N),
228:      + , ROM(N), ROFM(N), CKFM(N), CKSM(N), EIM(N)
229:      2 CONTINUE

```

```

229:      WRITE(KP,605)
230:      WRITE(KP,606)
231:      DEPTH=0.0
232:      TT=0.0
233:      DO 3 J=1,NLAY
234:      DEP(J)=DEPTH
235:      READ(UC,503) NO,MKIND,MAT,H(J)
236:      DEM(J)=DEPTH+H(J)/2.0
237: C
238: C      SET KEY INDEX FOR DEGREE OF FREEDOM
239: C
240: C      MKIND = 1 ... JUST SOLID
241: C      MKIND = 2 ... JUST FLUID
242: C      MKIND = 3 ... MIXTURE DRAIN CONDITION
243: C      MKIND = 4 ... MIXTURE UNDRAIN CONDITION
244: C      TO IND(J,1) ... DISPLACEMENT (X-DIRECTION) OF SOLID
245: C      IND(J,2) ... DISPLACEMENT (Z-DIRECTION) OF SOLID
246: C      IND(J,3) ... RELATIVE DIS. (X-DIRECTION) OF FLUID
247: C      IND(J,4) ... RELATIVE DIS. (Z-DIRECTION) OF FLUID
248: C
249:      DO 1 N=1,4
250:      IND1(J,N)=1
251:      1 CONTINUE
252:      IF(MKIND.EQ.3) GO TO 10
253:      IND1(J,3)=0
254:      IND1(J,4)=0
255:      10 CONTINUE
256:      IF(MKIND.NE.2) GO TO 11
257:      IND1(J,1)=0
258:      IND1(J,2)=0
259:      IND1(J,3)=1
260:      IND1(J,4)=1
261:      11 CONTINUE
262: C
263: C      SET MATERIAL CONSTANT OF EACH LAYER
264: C
265:      IF(MAT.GT.NMAT) GO TO 901
266:      VS=VSM(MAT)
267:      DAMP=DAMP(MAT)
268:      PSNR=PSNR(MAT)
269:      PERM(PERM(MAT))/ROFM(MAT)
270:      CNE(CNE(MAT))
271:      IF(MKIND.EQ.2) CNE(J)=1.0
272:      E1(J)=E1(MAT)
273:      RO(J)=RO(MAT)/GAL
274:      ROF(J)=ROFM(MAT)/GAL
275:      CKS=CKS(MAT)
276:      CKF=CKF(MAT)
277:      GO TO 902
278:      STOP 'SPECIFIED MATERIAL IS NOT EXIST'
279:      902 CONTINUE
280:      W1=RO(J)*VS*VS
281:      W2=1.0-2.0*DAMP*DAMP
282:      W3=SGRT11.0-DAMP*DAMP
283:      ZG(J)=B1*CMPLX(W2,2.0*DAMP*W3)
284:      IF(MKIND.EQ.2) ZG(J)=CMPLX(0.0,0.0)
285:      ZRAD=ZG(J)*B2.0*PSNR/(1.0-2.0*PSNR)

```

```

286: ZKD=ZRAMD+ZG(J)*2.0/3.0
287: ZAL(J)=1.0-ZKD/CKS
288: C
289: IF(MKIND.EQ.1) ZAL(J)=CMPLX(0,0,0,0)
290: IF(MKIND.EQ.2) ZAL(J)=CMPLX(1,0,0,0)
291: C
292: ZW1=(ZAL(J)-CNE(J))/CKS+1.0/CKF
293: ZG(J)=1.0/ZW1
294: IF(MKIND.EQ.1) ZG(J)=CMPLX(0,0,0,0)
295: ZAA(J)=ZRAMH*2.0*ZG(J)+ZAL(J)*ZAL(J)*ZG(J)
296: WRITE(*,603) J,AM(MKIND),MAT,H(J),DEPTH,ZAA(J),ZG(J),ZAL(J),ZG(J)
297: DEPTH=DEPTH+H(J)
298: IF(VS.LT.0.001) GO TO 3
299: TT=TT+H(J)/VS
300: C
301: CONTINUE
302: C
303: C
304: C
305: 499 FORMAT(ABD)
306: 500 FORMAT(315,5X,10I1,15,5X,20I1)
307: 501 FORMAT(10FB,0)
308: 503 FORMAT(315,F10.0)
309: 600 FORMAT(IH//,'NUMBER OF THE CALCULATION FREQUENCY = ',15,/)
310: 600 FORMAT(IH,'AT FREQUENCY = ',20F5.1)
311: 601 FORMAT(IH,'THERE IS NO FILE IN THIS CALCULATION')
312: 602 FORMAT(IH//,'MATERIAL ',2X,'VS',10X,'DAMPING ',5X,
313: + 'POISSON-R ',3X,'PERMEABILITY ',IX,'POLOSITY ',3X,
314: + 'UNIT WEIGHT ',1X,'UNIT WEIGHT ',2X,'KF ',9X,'KS',
315: + '9X,EI OF PILE ',3X,'NO ',4X,'(m/sec)',29X,'(m/sec)',
316: + '17X, QF MIXTURE ',2X,'QF FLUID ')
317: 603 FORMAT(IH,'14.2X,10(PE12.3)')
318: 605 FORMAT(IH//,'LAYER ',2X,'MATERIAL ',2X,'THICKNESS',
319: + '2X,'DEPTH ',7X,'A-VALUE ',18X,'Q-VALUE ',18X,'ALPHA ',18X,
320: + 'Q-VALUE ')
321: 604 FORMAT(IH,'12.2X,A5,13.2F9.2,8(1PE12.4)
322: 606 FORMAT(IH,'3X,(TYPE)',8X,(m),',3X,
323: + '4('REAL,IMAG,'))
324: 650 FORMAT(IH,'***** FORCE VECTOR ',2X,20I1)
325: C
326: CCC WRITE(*,*),
327: RETURN
328: END
329: C ****
330: SUBROUTINE MATRIX(IND1)
331: C ****
332: IMPLICIT COMPLEX*16(Z)
333: PARAMETER (MLY=10)
334: PARAMETER (M4=(MLY+1)*4)
335: COMMON /GROUND/ ZAA(112),ZG(112),ZAL(112),H(112),PERM(112),
336: + CNE(112),RD(112),RF(112),DEM(112)
337: COMMON /COND1/ KC,KP,KG(4),NLAY,MJYD,MRL,NFLG1,IND2(2),
338: COMMON /DMY1/ ZD(M4,M4),ZWK1(4,4),ZWK(4,4),ZWK8(16,8)
339: COMMON /NOCH1/ ZA(M4,M4),ZDD(M4,M4)
340: COMMON /NOCH2/ ZK(M4,M4),ZCM(M4,M4)
341: COMMON /NOCH4/ ZB(M4,M4),ZRB(M4/2,M4/2)
342: DIMENSION IND1(MLY,*)

```

```

343: C
344: C ZERO=CMPLX(0, 0, 0, 0)
345: C
346: CCC WRITE(*, *) 'SUBROUTINE MATRIX NOW'
347: C
348: C INITIAL ZERO CLEAR
349: C
350: CALL ZEROCL(ZWK, 4, 4, 4)
351: CALL ZEROCL(ZA, M4, M4, M4)
352: CALL ZEROCL(ZB, M4, M4, M4)
353: CALL ZEROCL(ZD, M4, M4, M4)
354: CALL ZEROCL(ZH, M4, M4, M4)
355: CALL ZEROCL(ZCM, M4, M4, M4)
356: CALL ZEROCL(ZDD, M4, M4, M4)
357: CALL ZEROCL(ZRB, M4/2, M4/2, M4/2)
358: DO 2 J=1,NLAY
359: ZAG=ZAA(JJ)-2.0*ZG(J)
360: ZAG=ZAL(JJ)*ZB(J)
361: KO=JJ*4-3
362: RNE=RDF(JJ)/CNE(JJ)
363: C
364: C MATRIX A ELEMENT (UPPER LEFT SIDE 1/4)
365: C
366: ZWK(1, 1)=2.0*ZAA(JJ)
367: ZWK(1, 3)=2.0*ZAG
368: ZWK(2, 2)=2.0*ZAG
369: ZWK(3, 1)=2*WKA(1, 3)
370: ZWK(3, 3)=2.0*ZG(J)
371: ZWK(4, 3)=CMPLX(1.0, 0, 0)
372: CALL COPY(ZWK, ZWK, 4, 0, 0, 5, 1, 0, H(J)/6, 0)
373: CALL STRE(ZWK, B, ZA, KO, M4)
374: C
375: C MATRIX D ELEMENT (UPPER LEFT SIDE 1/4) AND DD ELEMENT
376: C
377: ZWK(1, 2)=2*ZG
378: ZWK(1, 4)=ZAG
379: ZWK(2, 1)=ZG(J)
380: ZWK(3, 2)=ZAG
381: ZWK(3, 4)=ZG(J)
382: DO 1 L=1, 4
383: DO 1 K=1, 4
384: 1 ZWK1(K, L)=ZWK(K, L)
385: ZWK1(4, 4)=CMPLX(1.0, 0, 0, 0)
386: CALL COPY4(ZWK, ZWK, 4, 1, 0, -1, 0, 0, 0, 5)
387: CALL STRE(ZWK, B, ZD, KO, M4)
388: CALL COPY4(ZWK1, ZWK8, 4, 1, 0, -1, 0, 0, 0, 5)
389: CDDDD WRITE(*, *) ZA(3, 1), ZD(5, 6), KO
390: CDDDD
391: CDDDD CALL STRE(ZWK8, B, ZDD, KO, M4)
392: CDDDD
393: CCC WRITE(*, *) ZA(3, 1), ZD(5, 6), KO
394: CCC
395: CDDDD
396: C
397: C MATRIX K ELEMENT (UPPER LEFT SIDE 1/4)
398: C
399: ZWK(1, 1)=ZG(J)

```

```

400: ZWK(2,2)=ZAA(J)
401: ZWK(2,4)=ZAG
402: ZWK(4,2)=ZAG
403: ZWK(4,4)=ZG(J)
404: CALL COPYA(ZWK,ZWK8,4,-1,0,-1,0,1,0,H(J))
405: CALL STRE(ZWK8,B,ZK,KO,M4)
406: C----- MATRIX CM ELEMENT (UPPER LEFT SIDE 1/4)
407: C----- 408: C----- 409: ZWK(1,1)=CMPLX(2,0*RDF(J),0,0)
410: ZWK(1,3)=CMPLX(2,0*RDF(J),0,0)
411: ZWK(2,2)=CMPLX(2,0*RDF(J),0,0)
412: ZWK(2,4)=CMPLX(2,0*RDF(J),0,0)
413: ZWK(3,1)=ZWK(1,3)
414: PP=0.0
415: IF(IND1(J,1).EQ.1.AND.IND1(J,3).EQ.1) PP=2.0/PERM(J)
416: ZWK(3,3)=CMPLX(2,0*RNIE,PP)
417: IF(CNE(J).GT.0.999) ZWK(3,3)=CMPLX(2,0*RNIE,0,0)
418: ZWK(4,2)=ZWK(2,4)
419: ZWK(4,4)=ZWK(3,3)
420: CALL COPY4(ZWK,ZWK8,4,0,5,0,5,1,0,H(J)/6,0)
421: CALL STRE(ZWK8,B,ZCM,KO,M4)
422: C----- 423: IF(INFLG1.EQ.0) GO TO 2
424: C----- 425: C----- MATRIX RB ELEMENT
426: C----- 427: DO 3 K=1,4
428: DO 3 L=1,4
429: 3 ZWK(K,L)=ZERO
430: ZWK(1,1)=CMPLX(6,0,0,0)
431: ZWK(2,1)=CMPLX(3,0.4*H(J),0,0)
432: ZWK(2,2)=CMPLX(2,0.4*H(J)*H(J),0,0)
433: ZWK(3,1)=CMPLX(-6,0,0,0)
434: ZWK(3,2)=ZWK(2,1)
435: ZWK(3,3)=CMPLX(6,0,0,0)
436: ZWK(4,1)=ZWK(2,1)
437: ZWK(4,2)=ZWK(2,2)/2.0
438: ZWK(4,3)=ZWK(3,2)
439: ZWK(4,4)=ZWK(2,2)
440: CALL ZTM1(ZWK,4,4)
441: CFAC=2.0*EI(J)/H(J)/H(J)
442: DO 5 L=1,4
443: DO 5 K=1,4
444: 5 ZWK(K,L)=ZWK(K,L)*CFAC
445: KB=J*2-1
446: CALL STRE(ZWK,4,ZRB,KB,M4/2)
447: CALL ZEROL((ZWK,4,4,4)
448: 2 CONTINUE
449: C----- 450: C----- MAKE B MATRIX
451: C----- 452: MM=NLAY*4
453: CALL ZTENCH((ZD,ZB,MM,M4)
454: DO 4 L=1,MM
455: DO 4 K=1,MM
456: ZB(K,L)=ZB(K,L)-ZD(K,L)

```

```

457:    4 CONTINUE
458:    CCC CALL POUTM(ZA, MM, MM, KP, M4)
459:    CCC CALL POUTM(ZD, MM, MM, KP, M4)
460:    CCC WRITE(*,*), ZA(3,1), ZD(5,6), KO
461:    CCC CALL POUTM(ZK, MM, MM, KP, M4)
462:    CCC CALL POUTM(ZCM, MM, MM, KP, M4)
463:    CCC CALL POUTM(ZB, MM, MM, KP, M4)
464:    CCC WRITE(*,*), MATRIX COMPLETE'
465:    RETURN
466: END
467: C ****SUBROUTINE RDMAT(IND1)
468: C ****IMPLICIT COMPLEX*16(Z)
469: C ****PARAMETER (MLV=10)
470: C ****DIMENSION IND1(MLV,*)
471: C ****COMMON /NOCH1/ ZA(M4,M4), ZD(M4,M4)
472: C ****COMMON /NOCH2/ ZK(M4,M4), ZCM(M4,M4)
473: C ****COMMON /NOCH4/ ZB(M4,M4), ZRB(M4,2, M4/2)
474: C ****COMMON /COND1/ KC, KP, KG(4), NLAY, NJVD, MR1, NFLG1, IND2(2)
475: C ****COMMON /DMY1/ ZW(M4,M4)
476: C ****COMMON /DMY2/ ZW(M4,M4)
477: C ****COMMON /DMY3/ ZW(M4,M4)
478: C ****COMMON /DMY4/ ZW(M4,M4)
479: CCC WRITE(*,*), 'SUBROUTINE RDMAT NOW '
480: C
481: C COUNT DEGREE OF FREEDOM
482: C
483: N1=0
484: DO 10 N=1, 3, 2
485: 1. J=0
486: N1=N1+1
487: DO 11 J=1, NLAY
488: IF (IND1(J,N).NE. 0) 1=J+1
489: 1.1 CONTINUE
490: IND2(1)=1J
491: 10 CONTINUE
492: WRITE(KP, 602)
493: DO 12 J=1, NLAY
494: 12 WRITE(KP, 603) J, (IND1(J,I), I=1, 4)
495: 12 WRITE(KP, 604) (IND2(I), I=1, 2)
496: C
497: C REARRANGE MATRIX
498: C
499: CALL REARRG(ZA, ZW, NLAY*4, M4, 4)
500: NM=NLAY*4
501: CCC CALL POUTM(ZA, MM, MM, KP, M4)
502: CCC CALL REARRG(ZD, ZW, NLAY*4, M4, 4)
503: CCC CALL REARRG(ZK, ZW, NLAY*4, M4, 4)
504: CCC CALL REARRG(ZCM, ZW, NLAY*4, M4, 4)
505: CCC CALL REARRG(ZB, ZW, NLAY*2, M4, 2)
506: CCC CALL POUTM(ZB, MM, MM, KP, M4)
507: CCC
508: C
509: C REDUCE DEGREE OF FREEDOM
510: C
511: K=0
512: K1=0
513: DO 20 N=1, 4

```

```

514: DO 1 J=1, NLAY
515: K=K+1
516: IF(IND1(J,N).LE.0) GO TO 1
517: K1=K1+1
518: L1=0
519: L=0
520: DO 21 N1=1, 4
521: DO 2 J1=1, NLAY
522: L=L+1
523: IF(IND1(J1,N1).LE.0) GO TO 2
524: L1=L1+1
525: ZA(K1,L1)=ZA(K,L)
526: ZD(K1,L1)=ZD(K,L)
527: ZB(K1,L1)=ZB(K,L)
528: ZH(K1,L1)=ZH(K,L)
529: ZCM(K1,L1)=ZCM(K,L)
530: 2 CONTINUE
531: 21 CONTINUE
532: 1 CONTINUE
533: 20 CONTINUE
534: WRITE(KP,600) L1
535: MJYD=L1
536: MR1=IND2(1)*2+IND2(2)
537: CCC CALL POUT(ZA,MJYD,KP,JYD,KP,M4)
538: WRITE(*,*) 'RDNAT COMPLETE'
539: 600 FORMAT(1HO, / **** NUMBER OF DEGREE OF FREEDOM =', I5, ', ****')
540: IF(K1.NE.L1) WRITE(KP,601) K1
541: 601 FORMAT(1HO, 'K1= ', I5)
542: 602 FORMAT(1H1, 'DEGREE OF FREEDOM', //, 'LAYER NO. ', /,
543: *2X, 'SOLID-X SOLID-Z FLUID-X FLUID-Z ')
544: 603 FORMAT(1H , 'TOTAL= ', 2X, 16.8X, 1B)
545: RETURN
546: END
547: C ****
548: C ****SUBROUTINE EIGENZA, ZB, ZHS, ADEP, ****
549: C ****
550: C ****
551: IMPLICIT REAL*8(B-H, D-Y), COMPLEX*16(Z)
552: PARAMETER (MLY=10)
553: PARAMETER (M3=MLY*3)
554: PARAMETER (M4=(MLY+1)*4)
555: PARAMETER (MB=MLY*8)
556: DIMENSION ZA(1M, *), ZB(1M, *), ZHS(*)
557: COMMON /NOCH3/ ZRHI(M4, M3), ZPINV(M3, M3)
558: COMMON /NOCH5/ ZCINV(M4, M4)
559: COMMON /DMY1/ EGR(MB, MB)
560: COMMON /DMY2/ EG1(MB, MB)
561: COMMON /DMY3/ PR(MB, MB)
562: COMMON /DMY4/ PI(MB, MB)
563: COMMON /DMY5/ HSR(MB), HS1(MB), FV1(MB), FV2(MB), FV3(MB)
564: COMMON /DMY6/ ZW(M4, M4), ZW1(M4, M4)
565: COMMON /COND1/ KC, KP, KG(4), NLAY, MJYD, MR1, NFLG1, MUX, MAX
566: ZERO=CHPLX(0.0, 0.0)
567: ZAI=CHPLX(0.0, 1.0)
568: CCC WRITE(*,*) 'SUBROUTINE EIGEN NOW'
569: C
570: CALL ZEROL(ZW1, M4, M4, M4)

```

```

571: CALL ZEROC1(ZPHI,M4,M3,M4)
572: CALL ZEROC1(ZPINV,M2,M3,M3)
573: CALL ZEROC1(ZW,M4,M4,M4)
574: C
575: DO 1 L=1,MJYD
576:   DO 1 K=1,MJYD
      ZWK,L)=ZBK(K,L)/ADEP
577: 1 CONTINUE
578: 1 CONTINUE
579: CALL 2MULT1(ZCINV,ZW,ZWI,MJYD,MJYD,KP,M4)
580: CCC CALL POUTM(ZWI,MJYD,MJYD,KP,M4)
581: DO 2 L=1,MJYD
582: DO 2 K=1,MJYD
      ECR(K,L)=REAL(ZWI(K,L))
583: EGI(K,L)=AIMAG(ZWI(K,L))
584: 2 CONTINUE
585: CALL ZEROC1(ZW,M4,M4,M4)
586: DO 4 L=1,MR1
587:   DO 4 K=1,MR1
      ZWK,L)=ZAI(K,L)/(ADEP*ADEP)
588: 4 CONTINUE
589: CALL 2MULT1(ZCINV,ZW,ZWI,MJYD,MJYD,M4,M4,M4)
590: DO 3 L=1,MJYD
591:   DO 3 K=1,MJYD
      EGR(K,L+MJYD)=REAL(ZWI(K,L))
592: EGI(K,L+MJYD)=AIMAG(ZWI(K,L))
593: 3 CONTINUE
594: 3 CONTINUE
595: CCC CALL POUTM(ZWI,MJYD,MJYD,KP,M4)
596: CCC CALL POUTM(ZWI,MJYD,MJYD,KP,M4)
597: DO 10 L=1,MJYD
598:   DO 10 K=1,MJYD
      VAL=0.0
599: 10 CONTINUE
600: IF(K.EQ.L) VAL=1.0
601: EGR(K+MJYD,L)=VAL
602: EGI(K+MJYD,L)=0.0
603: EGR(K+MJYD,L+MJYD)=0.0
604: EGI(K+MJYD,L+MJYD)=0.0
605: 10 CONTINUE
606: C
607: C
608: C
609: MSDL=MJYD+MR1
610: CALL CG(MLY*B,MSDL,EGR,EGI,HSR,HSI,I,PR,P1,FV1,FV2,FV3,IERR)
611: C
612: DO 5 J=1,MSDL
      HSR(J)=HSR(J)*ADEP
613: HSI(J)=HSI(J)*ADEP
614: 5 CONTINUE
615: WRITE(KP,600) IERR
616: IF(IERR.NE.0) GO TO 99
617: C
618: C
619: C
620: C
621: CALL SLCTS1(ZPHI,ZHS,HSR,HSI,FR,P1,MSDL,MLY,M4)
622: CCC
623: CCC CALL POUTM(ZPHI,MJYD,MRI,KP,M4)
624: C
625: C
626: C
627: C

```

```

628: DO 100 J=1, M41
629: DO 100 N=1, M41
630: ZPINV(N, J)=ZPHI(N, J)
631: 100 CONTINUE
632: CALL ZINV1(ZPINV, M41, M41, M3*2)
633: CALL ZOUTM(ZPINV, M41, M41, KP, M3)
634: WRITE(*,*) ' EIGEN COMPLETE'
635: C
636: C FORMAT
637: C
638: 600 FORMAT(1H , '*** TERR= ', I4, ' ***')
639: GO TO 98
640: 99 STOP 'STOP TERR NE. 0'
641: 98 CONTINUE
642: RETURN
643: END
644: C **** SUBROUTINE RMATRIX(ZA, ZPHI, ZPINV, ZHS, ZD)
645: ****
646: C ****
647: IMPLICIT COMPLEX*16(Z)
648: PARAMETER (MLY=10)
649: PARAMETER (M3=MLY*3)
650: PARAMETER (M4=(MLY+1)*4)
651: DIMENSION ZD(M4,*), ZHS(*,*), ZA(M4,*), ZPHI(M4,*), ZPINV(M3,*)
652: COMMON /COND1/ KC, KP, KO(4), NLAY, MJYD, MR1, NFLG1, IND2(2)
653: COMMON /DMY1/ ZR(M4, M3)
654: COMMON /DMY6/ ZWRK1(M4, M3), ZWRK2(M4, M3)
655: C
656: CCC WRITE(*,*) ' SUBROUTINE RMATRIX NOW'
657: CALL ZMULT1(ZA, ZPHI, ZWRK1, MJYD, MR1, M4, M4)
658: DO 1 L=1, MR1
659: DO 2 K=1, MJYD
660: ZWRK1(K, L)=ZWRK1(K, L)*ZHS(L)
661: 2 CONTINUE
662: 1 CONTINUE
663: CALL ZMULT1(ZWRK1, ZPINV, ZR, MJYD, MR1, M4, M3, M4)
664: CALL ZMULT1(ZD, ZPHI, ZWRK1, MJYD, MJYD, MR1, M4, M4)
665: CALL ZMULT1(ZWRK1, ZPINV, ZWRK2, MJYD, MR1, M4, M3, M4)
666: DO 10 K=1, MJYD
667: DO 10 L=1, MR1
668: ZR(K,L)=ZR(K,L)*CMPLX(0.0, 1.0)+ZWRK2(K, L)
669: 10 CONTINUE
670: CCC WRITE(KP,*) ' MATRIX R'
671: CCC CALL POUTM(ZR, MJYD, MR1, KP, M4)
672: CCC WRITE(*,*) ' RMATRIX COMPLETE'
673: RETURN
674: END
675: C **** SUBROUTINE CMATRIX(ZCM, ZCINV, ZK, CMG, DMG2, M4)
676: ****
677: C ****
678: IMPLICIT COMPLEX*16(Z)
679: DIMENSION ZCM(M4,*), ZCINV(M4,*), ZK(M4,*)
680: COMMON /COND1/ KC, KP, KO(4), NLAY, MJYD, MR1, NFLG1, IND2(2)
681: C
682: CCC WRITE(*,*) ' SUBROUTINE CMATRIX NOW'
683: DO 1 K=1, MJYD
684: DO 1 L=1, MJYD

```



```

742: C-----+
743: DO 30 L=1, MJN
    DO 30 K=1, MJN
    ZW(K,L)=ZRI(K,L)
30 CONTINUE
744: C-----+
745: CASE=1
746: C-----+
747: C-----+
748: C-----+
749: C-----+
750: CALL ZINV1(ZW,MJN,MJN,M4*2)
    CALL ZMULT1(ZW,ZPUX,ZUW,MJN,MJN,1,M4,1,1)
751: WRITE(10P,600)
752: CALL WPAI(ZUW,ZC34,ZPHI,ZPINV,ZALP,ZPUX,ZW2,M4,MLY,OMG)
753: CALL OUTPT(ZUW,ZALP,ZRUX,ZHS,ZPHI,ZW,ZW1,ZW2,ZR,NXOUT
+ ,IND1,OMG)
754: C-----+
755: C-----+
756: C-----+
757: GO TO 99
758: C-----+
759: C-----+
760: C-----+
761: 102 CONTINUE
    CALL ZINV1(ZC32,MUX,MLY)
    CALL ZMULT1(ZC32,ZC32,Z2221,MUX,MLY,MLY,MLY)
762: CALL ZMULT1(ZC12,Z2221,ZWORK,MLAY,MUX,MLY,MLY,MLY)
763: DO 50 L=1,MLAY
    DO 50 K=1,MLAY
50 ZC11(K,L)=ZC11(K,L)-ZWORK(K,L)

764: C-----+
765: DO 51 L=1,MLAY*2
    DO 51 K=1,MLAY*2
51 ZW(K,L)=ZRW(K,L)
766: DO 52 K=1,MLAY
    DO 52 L=1,MLAY
52 ZW(K,L)=ZWK(L)+ZC11(K,L)
767: C-----+
768: CASE=2
769: C-----+
770: DO 53 L=1,MLAY*2
    DO 53 K=1,MLAY*2
53 ZW(K,L)=ZRW(K,L)
771: CALL ZMULT1(ZW,ZPUX,ZUW,MLAY*2,MLAY*2,1,M4,1,1)
772: CALL ZMULT1(ZC11,ZUW,ZPUX,MLAY,MLAY,1,MLY,1,1)
773: CALL ZMULT1(Z2221,ZUW,ZPUX,MLAY,MLAY,1,MLY,1,1)
774: CALL ZMULT1(Z2221,ZUW,ZW2,MUX,MLY,1,MLY,1,1)
775: C-----+
776: C-----+
777: C-----+
778: CALL ZINV1(ZW,MLAY*2,MLAY*2,M4*2)
779: CALL ZMULT1(ZW,ZPUX,ZUW,MLAY*2,MLAY*2,1,M4,1,1)
780: C-----+
781: CALL ZMULT1(ZC11,ZUW,ZPUX,MLAY,MLAY,1,MLY,1,1)
782: CALL ZMULT1(Z2221,ZUW,ZW2,MUX,MLAY,1,MLY,1,1)
783: DG 60 N=1,MUX
    ZUX(MLAY+N)=-ZW2(N)
    ZPUX(MLAY+N)=ZERO
60 CONTINUE
784: WRITE(10P,603)
785: CALL WPAI(ZUW,ZC34,ZPHI,ZPINV,ZALP,ZPUX,ZW2,M4,MLY,OMG)
786: CALL OUTPT(ZUW,ZALP,ZRUX,ZHS,ZPHI,ZW,ZW1,ZW2,ZR,NXOUT
+ ,IND1,OMG)
787: C-----+
788: 99 CONTINUE
789: C-----+
790: C-----+
791: C-----+
792: C-----+
793: C-----+
794: WRITE(*,*), ' SOLVE COMPLETE'
795: 600 FORMAT(1H , 'NO FILE CASE=1 FORCED VIBRATION ')
796: 603 FORMAT(1H , 'FILE EXIST CASE=1 FORCED VIBRATION ')
797: C-----+
    RETURN

```

```

799: END
800: C ****SUBROUTINE WZPAI(ZUM,ZC34,ZPH1,ZPINV,ZALP,ZPUX,ZW2,M4,MLY,DLG;
801: C ****
802: C ****IMPLICIT COMPLEX*16(Z)
803: C ****
804: DIMENSION ZUM(*),ZC34(MLY*2,*),ZPINV(MLY*3,*),ZW2(*),ZPUX(*)
805: C
806: COMMON /COND1/ KC,KP,KC(4),NLAY,MUX,MUX,NWX
807: COMMON /RESP/ ZUX1,ZUZ1,ZWZ1
808: ZERO=CMPLX(0,0,0,0)
809: ZAI=CMPLX(0,0,1,0)
810: C-----
811: C-----CALCULATE OF PAI
812: C-----
813: MUN=MUX+NLAY
814: MS=MUX*2+MUX-NLAY
815: MAU=NLAY-MUX
816: MUX=NLAY-MUX
817: CALL ZMULT1(ZC34,ZUM,ZW2,MS,MUX+NLAY,1,MLY*2,1,1)
818: DO 4 N=1,MUX+MUX-NLAY
819: 4 ZPUX(N+MUN)=ZW2(N)
820: DO 3 N=1,MUX
821: ZPUX(N+MR1)=-ZAI*DNG*ZW2(MUX+MUX-NLAY+N)
822: 3 CONTINUE
823: C-----
824: C-----CALCULATE ALPHA
825: C-----
826: IF(MAU.EQ.0) GO TO 90
827: DO 5 N=1,MAU
828: ZALP(N+MUX*2)=ZPUX(N)
829: 5 ZW2(N+MUX*2)=ZUW(N)
830: DO 6 N=1,MUX*2
831: ZALP(N)=ZPUX(N+MAU)
832: 6 ZW2(N)=ZUW(N+MAU)
833: DO 7 N=1,MUN
834: ZPUX(N)=ZALP(N)
835: 7 ZUW(N)=ZW2(N)
836: 90 CONTINUE
837: DO 10 N=1,MUX+MUX-NLAY
838: ZUW(N+MUN)=ZERO
839: 10 CONTINUE
840: CALL ZMULT1(ZPINV,ZUM,ZALP,MR1,MR1,1,MLY*3,1,1)
841: CC CALL WRITE(KP,*,'VECTOR ALPHA'
842: CC CALL PDOUT(ZALP,MR1,1,KP,MR3)
843: C-----CHECK WRITE UX,UZ,WX
844: C-----WRITE(KP,600)
845: C-----WRITE(KP,600)
846: CC DD 1 N=1,MUX
847: CC 1 WRITE(KP,601) N+MAU,ZUW(N),ZUW(N+MUX)
848: CC DD 2 N=1,MUX
849: CC 2 WRITE(KP,601) N+MUN,ZUW(MUX*2+N)
850: CC CG600 FORMAT(IH,'*** DISP. ***',/15X,'X-DIRECTION',16X,'Z-DIRECTION')
851: CG601 FORMAT(IH,'***',/15X,'X-DIRECTION',16X,'Z-DIRECTION')
852: CC CG601 FORMAT(IH,15,5X,4E15.5)
853: C-----CALCULATE WZ
854: C-----
```

```

855: CALL ZMULT1(ZPH1, ZALP, ZUN, MJYD, MR1, 1, M4, 1, 1)
856: ZUX1=ZUN(1)
857: C
858: ZUZ1=ZUN(1+MUX)
859: ZWZ1=ZUN(1+MUX*2+MAX+MAU)
860: RETURN
861: C
862: END
863: C*****SUBROUTINE OUTPT(ZUN, ZALP, ZPUX, ZHS, ZPH1, ZU, ZW1, ZWB, ZR,
864: + NMOUT, IND1, DMQ)
865: C*****IMPLICIT COMPLEX*16(Z)
866: C
867: PARAMETER (MLY=10)
868: PARAMETER (M4=(MLY+1)*4)
869: DIMENSION XDUT(10), IND1(MLY,*), CX(10), X(11),
870: ZHS(*), ZALP(*), ZUM(*), ZPUX(*), ZPH1(M4,*),
871: + ZW1(*), ZR(M4,*), ZH2(M4,*), ZH42(*),
872: COMMON /DMY5/ ZS0Z(MLY), ZTAU(MLY), ZPAI(MLY),
873: + ZSGX1(MLY,11), ZSGZ1(MLY,11), ZTAU1(MLY,11),
874: COMMON /COND1/ KC, KP, KG(4), MLAY, MJYD, MR1, NFLQ1, MUX, MWX
875: COMMON /GROUND/ ZAA(112), ZG(112), ZAL(112), H(112), PERM(112),
876: + ONE(112), RO(112), RGF(112), EI(112), DEP(112), DEM(112),
877: COMMON /CONST/ QAL, PAI, DEP, NOT, VGB, NFRC, DO
878: DATA C/0.1, 0.2, 0.3, 0.5, 0.1, 0.10, 0.20, 0/
879: ZAI=CHPLX(0, 0, 1, 0)
880: C
881: C
882: C OUTPUT UK, UZ, WZ, PUX, PPAI AT X=0, 0
883: C
884: DO 12 J=1, MR1
885: 12 ZW1(J)=ZAI*ZH5(J)*ZALP(J)
886: X(1)=0, 0
887: CALL OUT1(0, 0, ZUM, ZPUX)
888: CALL S0MA(ZSGX, ZSGZ, ZTAU, ZPAI, ZALP, ZPH1, ZW1, ZWB, IND1, M4, MLY)
889: DO 10 J=1, MLAY
890: ZD1=ZUR(J)
891: ZD2=ZUK(J++)
892: IF ((J, E0, MLAY) ZD2=CHPLX(0, 0, 0)
893: ZAVE=(ZD1+ZD2)/2, 0
894: ZKJ=ZSGX(J)/ZAVE
895: WRITE(KO(2), 650) DEM(J), ZKJ
896: ZSGX1(J, 1)=ZSGX(J)
897: ZSGZ1(J, 1)=ZSGZ(J)
898: ZTAU1(J, 1)=ZTAU(J)
899: ZPAI1(J, 1)=ZPAI(J)
900: 10 CONTINUE
901: C
902: C CALCULATION STRESS AND PORE PRESSURE AT X=CX(1)*DEPTH
903: C
904: N1=1
905: IF (NOT, EQ, 0) GO TO 90
906: DO 15 N=N1, 10
907: IF (NMOUT(N), EQ, 0) GO TO 15
908: N1=N1+1
909: X(N1)=DEPTH*CX(N)
910: DO 1 J=1, MR1
911: ZEX=EXP(-ZAI*ZH5(J)*X(N1))
912: ZWR1(J)=ZEX*ZALP(J)

```

```

(
  913: ZW1(J)=ZW2(J)*ZHS(J)*(-ZAI)
  914: 1 CONTINUE
  915: CALL ZMULT1(ZPH1, ZW2, ZUW, MJYD, MR1, 1, M4, 1, 1)
  916: CALL ZMULT1(ZR, ZUW, ZPUX, MJYD, MR1, 1, M4, 1, 1)
  917: DO 30 NN=1, MUX
  918: 30 ZPUX(NN+MR1)=ZPUX(NN+MR1)*(-ZAI)*OMG
  919: CALL OUT1(X(N1), ZUW, ZPUX)
  920: C
  921: CALL SOMA(ZSGX, ZSGZ, ZTAU, ZPAI, ZW1, ZPH1, ZW2, ZPUX, M4, MLY)
  922: DO 11 J=1, NLAY
  923: ZSG1(J, N1)=ZSGX(J)
  924: ZSG2(J, N1)=ZSGZ(J)
  925: ZTAU1(J, N1)=ZTAU(J)
  926: ZPAI1(J, N1)=ZPAI(J)
  927: 11 CONTINUE
  928: 15 CONTINUE
  929: 90 CALL OUT2(ZSGX1, ZSGZ1, ZTAU1, ZPAI1, X, NOTT+1, IND1)
  930: IF (INFRQ .NE. 1) GO TO 102
  931: IF (NOTT .NE. 0) GO TO 102
  932: CCC  IF (NLAY .GT. 2) GO TO 102
  933: MUX=NLAY-MUX
  934: READ(IND, K0(3))
  935: DX=0.01*DO
  936: XX=DX
  937: DO 100 N=1, 781
  938: IF (N .GT. 201) DX=0.1*DO
  939: IF (N .GT. 381) DX=0.2*DO
  940: XX=DX+XX
  941: DQ 101 J=1, MR1
  942: ZEX=EXP(-ZAI*ZHS(J)**XX)
  943: ZW2(J)=ZEX*ZALP(J)
  944: 101 CONTINUE
  945: CALL ZMLR(T1(2PH1, ZH2, ZUW, MJYD, MR1, 1, M4, 1, 1)
  946: 21=(ZUW(1))*1000.0
  947: 22=(ZUW(MUX+1))*1000.0
  948: 23=(ZUW(MUX*2+MA+1))*1000.0
  949: 24=(ZUW(MUX*2+MA+MUX+1))*1000.0
  950: WRITE(K0(3), 600) XX/DO, Z1, Z2, Z3, Z4
  951: 100 CONTINUE
  952: 102 CONTINUE
  953: 600 FORMAT(F10.3, 8E15.5)
  954: 650 FORMAT(F5.2, 4E15.5)
  955: RETURN
  956: END
  957: ****
  958: SUBROUTINE OUT1(X, ZUW, ZPUX)
  959: ****
  960: IMPLICIT COMPLEX*16(Z)
  961: DIMENSION ZUW(*), ZPUX(*)
  962: COMMON /COND1/ KC, KP, KG(4), NLAY, MJYD, MR1, NFLQ1, MUX, MHW,
  963: COMMON /GROUND/ ZAA(112), ZB(112), ZC(112), ZD(112), PERM(112),
  964: + CNE(112), RQF(112), E1(112), DEP(112), DEM(112)
  965: C
  966: MUX=NLAY-MUX
  967: MHW=MUX-MHX
  968: ZERO=CMPLX(0.0, 0.0)
  969: DZ=0.0

```

```

970:      WRITE (NP, 600) X
971:      WRITE (NP, 601)
972:      DO 1 N=1, MUX
973:      AB=ABS(ZUM(N))
974:      WRITE (NP, 605) N+MAU, ZPUX(N), ZUM(N), AB
975:      1 CONTINUE
976:      WRITE (NP, 603)
977:      DO 3 N=1, MUX
978:      NM=MUX+N
979:      AB=ABS(ZUM(NN))
980:      WRITE (NP, 605) N+MAU, ZPUX(NN), ZUM(NN), AB
981:      3 CONTINUE
982:      IF (MAY.EQ. 0) GO TO 4
983:      WRITE (NP, 602)
984:      DO 2 N=1, MAX
985:      NM=MUX*2+N
986:      AB=ABS(ZUM(NNN))
987:      WRITE (NP, 605) N+MAW, ZPUX(NNN), ZUM(NNN), AB
988:      2 CONTINUE
989:      WRITE (NP, 604)
990:      DO 5 N=1, MAX
991:      NM=N+MAY
992:      AB=ABS(ZUM(NNN))
993:      WRITE (NP, 605) N+MAW, ZPUX(NNN), ZUM(NNN), AB
994:      5 CONTINUE
995:      4 CONTINUE
996:      DO 6 N=1, MAY
997:      J1=N-MAU
998:      J2=N-MAW
999:      IF (J1.LE. 0) THEN
1000:      ZX=ZERO
1001:      ZU2=ZERO
1002:      ELSE
1003:      ZX=ZUM(J1)*1000.0
1004:      ZU2=ZUM(J1+MUX)*1000.0
1005:      END IF
1006:      IF (J2.LE. 0) THEN
1007:      ZX=ZERO
1008:      ZW2=ZERO
1009:      ELSE
1010:      ZX=ZUM(J2+MUY*2)*1000.0
1011:      ZW2=ZUM(J2+MUY*2+MWX)*1000.0
1012:      END IF
1013:      WRITE (KG(3), 650) DEP(N), ABS(ZUX), ABS(ZUZ), ABS(ZWX), ABS(ZWZ)
1014:      6 CONTINUE
1015:      WRITE (KG(3), 650) DEP(NLAY+1), DZ, DZ, DZ, DZ
1016:      C
1017:      FORMAT
1018:      C
1019:      600 FORMAT(IH, 'X', F10.2, '(M)', /
1020:      + 'RELATION BETWEEN DISP. AND LOAD OF SOLID')
1021:      601 FORMAT(IH, 'LAYER NO.', 3X, 'LOAD(X)', 17X, 'DISP.(X)', 16X,
1022:      + 'DISP.(ABS)')
1023:      603 FORMAT(IH, 'LAYER NO.', 3X, 'LOAD(Z)', 17X, 'DISP.(Z)', 16X,
1024:      + 'DISP.(ABS)')
1025:      602 FORMAT(IH, 'LAYER NO.', P-LOAD, '17X, 'DISP.(WX)', 15X,
1026:      + 'DISP.(ABS)')

```

```

      1027: 604 FORMAT(1H , 'LAYER NO.   QUANTITY OF FLOW', BX, 'DISP. (WZ) ')
      1028: 605 FORMAT(1H , 13, 5X, 10E1.2, 3)
      1029: 650 FORMAT(F5. 2, 4F10. 5)
      1030: RETURN
      1031: END

      1032: C *****
      1033: C *****
      1034: C *****
      1035: C *****
      1036: C *****
      1037: C *****
      1038: C *****
      1039: C *****
      1040: C *****
      1041: C *****
      1042: C *****
      1043: C *****
      1044: C *****
      1045: C *****
      1046: C *****
      1047: C *****
      1048: C *****
      1049: CCC  WRITE(*,*), 'SUBROUTINE SCHA NOW'
      1050: C *****
      1051: DO 1 J=1, NLAY
      1052: C *****
      1053: C *****
      1054: C SET T1 AND T2
      1055: C *****
      1056: DO 2 K=1, 4
      1057: DO 2 L=1, 4
      1058: ZT1(K, L)=ZERO
      1059: ZT2(K, L)=ZERO
      1060: 2 CONTINUE
      1061: ZT1(1, 1)=ZAA(J)-2.0*ZB(J)
      1062: ZT1(2, 1)=ZAA(J)-2.0*ZB(J)
      1063: ZT1(4, 1)=ZAL(J)*ZB(J)
      1064: ZT1(3, 2)=ZB(J)
      1065: ZT1(1, 3)=ZAL(J)*ZB(J)
      1066: ZT1(2, 3)=ZT1(1, 3)
      1067: ZT1(4, 3)=ZB(J)
      1068: ZT2(1, 2)=-(ZAA(J)-2.0*ZB(J))
      1069: ZT2(2, 2)=-ZAA(J)
      1070: ZT2(3, 1)=-ZB(J)
      1071: ZT2(4, 2)=-ZAL(J)*ZB(J)
      1072: ZT2(1, 4)=-ZAL(J)*ZB(J)
      1073: ZT2(2, 4)=ZT2(1, 4)
      1074: ZT2(4, 4)=-ZB(J)
      1075: C *****
      1076: DO 3 K=1, 4
      1077: DO 3 L=1, 4
      1078: ZT1(K, L+4)=ZT1(K, L)
      1079: ZT2(K, L+4)=-ZT2(K, L)
      3 CONTINUE
      1080: 4 K=1, 4
      1081: DO 4 L=1, 8
      1082: ZT1(K, L)=ZT1(K, L)/2.0
      1083: ZT2(K, L)=ZT2(K, L)/2.0

```

```

1084:      ZT2(W,L)=ZT2(W,L)/H(J)
1085:      4 CONTINUE
1086: C
1087: C      CALCULATE STRESS AND PORE PRESSURE AT THE CENTER OF LAYER
1088: C
1089: IF(IND1(J,1).EQ.1) THEN
1090:   LU=LU+1
1091:   DO 10 L=1,MRI
1092:     ZW(1,L)=ZPHI(LU,L)
1093:   10 ZW(2,L)=ZPHI(LU+MUX,L)
1094: ELSE
1095:   DO 11 L=1,MRI
1096:     ZW(1,L)=ZERO
1097:   11 ZW(2,L)=ZERO
1098: END IF
1099: IF(IND1(J,3).EQ.1) THEN
1100:   LW=LW+1
1101:   DO 21 L=1,MRI
1102:     ZW(3,L)=ZPHI(MUX*2+L,W,L)
1103:   21 ZW(4,L)=ZPHI(MUX*2+MUX+L,W,L)
1104: ELSE
1105:   DO 25 L=1,MRI
1106:     ZW(3,L)=ZERO
1107:   25 ZW(4,L)=ZERO
1108: 25 CONTINUE
1109: END IF
1110: IF(J.EQ.NLAY) GO TO 30
1111: IF(IND1(J+1,1).EQ.1) THEN
1112:   DO 12 L=1,MRI
1113:     ZW(5,L)=ZPHI(LU+1,L)
1114:   12 ZW(6,L)=ZPHI(MUX+LU+1,L)
1115: ELSE
1116:   DO 13 L=1,MRI
1117:     ZW(5,L)=ZERO
1118:   13 ZW(6,L)=ZERO
1119: END IF
1120: IF(IND1(J+1,3).EQ.1) THEN
1121:   DO 22 L=1,MRI
1122:     ZW(7,L)=ZPHI(MUX*2+L,W+1,L)
1123:   22 ZW(8,L)=ZPHI(MUX*2+MUX+L,W+1,L)
1124: 22 CONTINUE
1125: ELSE
1126:   DO 23 L=1,MRI
1127:     ZW(7,L)=ZERO
1128:   23 ZW(8,L)=ZERO
1129: CONTINUE
1130: END IF
1131: GO TO 40
1132: 30 CONTINUE
1133:   DO 24 L=1,MRI
1134:     ZW(5,L)=ZERO
1135:     ZW(6,L)=ZERO
1136:     ZW(7,L)=ZERO
1137:     ZW(8,L)=ZERO
1138: 24 CONTINUE
1139: 40 CONTINUE
1140: CALL ZMULT1(ZW,ZAHS,ZT14,B,MRI,1,M4,S,6)

```

```

1141: CALL ZMULT1(ZW,ZALX,ZT24,B,MRI,1,M4,B,8)
1142: CALL ZMULT1(ZT1,ZT14,ZW41,4,B,1,4,B,4)
1143: CALL ZMULT1(ZT2,ZT24,ZW42,4,B,1,4,B,4)
1144: ZS9X(J)=Z441(1)+Z442(1)
1145: ZS9Z(J)=Z442(2)+Z442(2)
1146: ZTAU(J)=Z441(3)+Z442(3)
1147: ZPAI(J)=Z441(4)+Z442(4)
1148: 1 CONTINUE
1149: C
1150: RETURN
END
1152: C ****SUBROUTINE WAVEOT(ZHS,ZPHI)
1153: C ****SUBROUTINE WAVEOT(ZHS,ZPHI)
1154: C ****
1155: C IMPLICIT COMPLEX*16(Z)
1156: C PARAMETER (MLY=10)
1157: C
1158: DIMENSION ZPH1(M4,*),ZHS(*)
1159: CC
1160: COMMON /COND1/ KC,KP,KQ(4),NLAY,NJYD,MRI,NFLG1,MUX,MUX
1161: COMMON /DMY1/ ZALX(M4),ZAHS(M4),ZW(M4,M4),
1162: CC
1163: CC
1164: CC
1165: CC
1166: CC
1167: CC
1168: CC
1169: C
1170: C
1171: ZERO=CMPLX(0,0,0)
1172: WRITE(NP,602)
1173: IF (INFRG .GT. 20) RETURN
1174: MAU=NLAY-MUX
1175: MAW=NLAY-MMAX
1176: DO 21 L=1,MRI
1177: PMAXB=0.0
1178: PMAXF=0.0
1179: DO 20 J=1,NLAY
1180: J1=J-MAU
1181: J2=J-MAW
1182: IF (J1.LE.0) THEN
1183: ZUX=ZERO
1184: ZUZ=ZERO
1185: ELSE
1186: ZUX=ZPH1(J1,L)
1187: ZUZ=ZPH1(J1+MUX,L)
1188: END IF
1189: IF (J2.LE.0) THEN
1190: ZXW=ZERO
1191: ZWZ=ZERO
1192: ELSE
1193: ZXW=ZPH1((J2+MUX)*2,L)
1194: ZWZ=ZPH1((J2+MUX)*2+MUX,L)
1195: END IF
1196: P1=ABS(ZUX)
1197: P2=ABS(ZWX)

```



```

C 1255: C OUTPUT FOR STRESS AND PORE PRESSURE
C 1256: C
C 1257: C
C 1258: IMPLICIT COMPLEX*16(Z)
C 1259: PARAMETER (MLY=10)
C 1260: DIMENSION ZSGX(MLY,*), ZTAU(MLY,*), ZPAI(MLY,*)
C 1261: DIMENSION X(*), IND1(MLY,*)
C 1262: C
C 1263: COMMON /COND1/ KC, MP, KG(4), NLAY, MJYD, MRY1, NFLG1, MUX, MNX
C 1264: COMMON /GROUND/ ZAA(12), ZG(12), ZAL(12), ZQ(12), H(12), PERM(12),
C 1265: + CNE(12), RO(12), ROF(12), EI(12), DEP(12), DEM(12)
C 1266: COMMON /DMY3/ ZDSQX(MLY,11), ZDSQZ(MLY,11), YCC(MLY,11)
C 1267: C
C 1268: WRITE(KP,600) (X(N),N=1,NC)
C 1269: WRITE(KP,601)
C 1270: DO 1 J=1,NLAY
C 1271: 1 WRITE(KP,602) (ABS(ZSGX(J,N)),N=1,NC)
C 1272: WRITE(KP,603)
C 1273: DO 2 J=1,NLAY
C 1274: 2 WRITE(KP,602) (ABS(ZSGZ(J,N)),N=1,NC)
C 1275: WRITE(KP,604)
C 1276: DO 3 J=1,NLAY
C 1277: 3 WRITE(KP,602) (ABS(ZTAU(J,N)),N=1,NC)
C 1278: WRITE(KP,605)
C 1279: DO 4 J=1,NLAY
C 1280: 4 WRITE(KP,602) (ABS(ZPAI(J,N)),N=1,NC)
C 1281: C
C 1282: ADB=0.0
C 1283: Z2=CMPXL(X,O,O,O)
C 1284: DO 5 J=1,NLAY
C 1285: AJS=(RO(J)-ROF(J))*#9. B#H(J)/2. O+ADB
C 1286: ADB=ADB+(RO(J)-ROF(J))*#9. B#H(J)
C 1287: 21=CMPXL(X,ADB,O,O)
C 1288: ZDSQX(J,L)=ZSGX(J,L)-ZPAI(J,L)
C 1289: ZDSQZ(J,L)=ZSGZ(J,L)-ZPAI(J,L)
C 1290: IF(IND1(J,1).EQ.0) GO TO 6
C 1291: Z2=ZPAI(J,L)/(Z1-ZDSQZ(J,L))
C 1292: YCC(J,L)=AB6(Z2)
C 1293: 6 CONTINUE
C 1294: WRITE(KP,606)
C 1295: DO 7 J=1,NLAY
C 1296: 7 WRITE(KP,602) (ABS(ZDSQX(J,N)),N=1,NC)
C 1297: WRITE(KP,607)
C 1298: DO 8 J=1,NLAY
C 1299: 8 WRITE(KP,602) (ABS(ZDSQZ(J,N)),N=1,NC)
C 1300: WRITE(KP,608)
C 1301: WRITE(KP,609)
C 1302: DO 9 J=1,NLAY
C 1303: 9 WRITE(KP,602) (YCC(J,N),N=1,NC)
C 1304: C
C 1305: DO 5 N=1,NC
C 1306: DO 5 J=1,NLAY
C 1307: WRITE(KG(4),650) DEM(J), ABS(ZDSQX(J,N)), ABS(ZSGZ(J,N))
C 1308: + ABS(ZTAU(J,N)), ABS(ZDSQZ(J,N)), ABS(ZPAI(J,N))
C 1309: + ABS(ZDSQX(J,N)), ABS(ZDSQZ(J,N)), YCC(J,N)
C 1310: 5 CONTINUE
C 1311: C

```

```

C 1312: C
C 1313: C
C 1314: 600 FORMAT(IH , 'DISTANCE FROM CENTER',/,11F11.2)
C 1315: 601 FORMAT(IH , '*****' SIGMA-XX (TF/M**2) *****/')
C 1316: 603 FORMAT(IH , '*****' SIGMA-ZZ (TF/M**2) *****/')
C 1317: 604 FORMAT(IH , '*****' TAU-ZX (TF/M**2) *****/')
C 1318: 605 FORMAT(IH , '*****' PORE PRESSURE (TF/M**2) *****/')
C 1319: 602 FORMAT(IH , '*****' 111(PE11.3))
C 1320: 606 FORMAT(IH , '*****' EFFECTIVE STRESS SIGMA-X (TF/M**2) *****/')
C 1321: 607 FORMAT(IH , '*****' EFFECTIVE STRESS SIGMA-Z (TF/M**2) *****/')
C 1322: 608 FORMAT(IH , '*****' RATIO PURE PRES. AND SG2(ST.)+SG2(DV) *****/')
C 1323: 650 FORMAT(F5.2,7E15.5)
C 1324: C
C 1325: RETURN
C 1326: END
C 1327: C *****
C 1328: SUBROUTINE SLCTHS(ZPHI,ZHS,HSR,HSI,PR,PI,MSOL,MLY,M4)
C 1329: C *****
C 1330: IMPLICIT REAL*8(A-H,D-Y), COMPLEX*16(Z)
C 1331: DIMENSION ZPHI(M4,*),ZHS(*),HSR(*),HSI(*)
C 1332: + PR(MLY*6,*),PI(MLY*6,*),MIND(200)
C 1333: COMMON /COND1/ KC,KP,KG(4),NLAY,MJYD,MRI,NFLG1,IND2(2)
C 1334: C
C 1335: ZAI=CMPLX(0.0,1.0)
C 1336: EPS=1.0E-07
C 1337: JCNT=0
C 1338: C
C 1339: DO 10 N=1,MSOL
C 1340: 10 MIND(N)=0
C 1341: 199 JCNT=JCNT+1
C 1342: DO 1 N=1,MSOL
C 1343: IF(MIND(N).NE.0) GO TO 1
C 1344: 201=CMPLX(HSI(N),HSI(N))
C 1345: IF(ABS(Z01).LT.1.0E-12) GO TO 20
C 1346: ZH1=ZAI/Z01
C 1347: IF(ABS(Z02).LT.EPS) GO TO 21
C 1348: IF(N.EQ.MSOL) GO TO 20
C 1349: DB 2 J=N+1,MSOL
C 1350: IF(MIND(J).NE.0) GO TO 2
C 1351: Z02=CMPLX(HSI(J),HSI(J))
C 1352: IF(ABS(Z02).LT.1.0E-12) GO TO 20
C 1353: ZH2=ZAI/Z02
C 1354: ZSI=ZH1+ZH2
C 1355: IF(ABS(ZSI).LT.EPS) GO TO 3
C 1356: 2 CONTINUE
C 1357: 20 MIND(N)=2
C 1358: 20 MIND(N)=3
C 1359: 21 MIND(N)=3
C 1360: 20 TO 1
C 1361: 3 IF(AIMAG(ZH2).GE.0.0) THEN
C 1362: MIND(J)=-1
C 1363: MIND(N)=1
C 1364: ELSE
C 1365: MIND(J)=1
C 1366: MIND(N)=-1
C 1367: END IF
C 1368: 1 CONTINUE

```

```

1388: C ----- CHECK WRITE
1389: C
1390: C
1391: C
1392: CC WRITE (KP,700) MSOL
1393: CC DO 101 N=1,MSOL
1394: CC Z2=ZAI/CMPLX(HSR(N),HSI(N))
1395: CC WRITE (KP,701) N,MIND(N),HSR(N),HSI(N),Z2
1396: CC DD 102 J=1,MSOL
1397: CC Z1=CMPLX(PR(J,N),PI(J,N))
1398: CC WRITE (KP,702) Z1,ABS(Z1)
1399: CC CONTINUE
CC101 CONTINUE
1400: L1=0
DO 22 N=1,MSOL
IF (MIND(N).NE.1) GO TO 22
L1=L1+1
1401: Z1=CMPLX(HSR(N),HSI(N))
1402: IF (ABS(Z1).LT.1.0E-12) GO TO 22
ZHS(L1)=ZAI/Z1
VMAX=0.0
DO 30 J=1,MJYD
1403: Z1=CMPLX(PR(J,N),PI(J,N))
1404: IF (ABS(Z1).GT.1.0E-12) VMAX=ABS(Z1)
30 CONTINUE
1405: DO 40 J=1,MJYD
1406: P1=PR(J,N)/VMAX
1407: P2=PI(J,N)/VMAX
1408: ZPHI(J,L1)=CMPLX(P1,P2)
40 CONTINUE
22 CONTINUE
1409: C
1410: WRITE (KP,600) L1
DO 50 N=1,MSOL
1411: IF (MIND(N).EQ.2) MIND(N)=0
50 CONTINUE
1412: EPB=EPS*0.0
1413: IF (JCNT.GT.6) GO TO 99
1414: IF (L1.NE.MR1) GO TO 199
1415: GO TO 100
1416: STOP 'EIGEN CHECK'
1417: 100 CONTINUE
1418: DO 23 N=1,L1
1419: WRITE (KP,601) N,ZHS(N)
23 CONTINUE
1420: C
1421: FORMAT(1H , 'NUMBER OF SELECTED WAVE NUMBER =',15)
1422: 600 FORMAT(1H , 'WAVE NUMBER =',2E15.5)
1423: END *****
1424: C SUBROUTINE CDNR1(ZH,ZR1,ZC11,ZC12,ZC21,M4,MLY,MAU)
1425: *****

```

```

C 1426: C ****IMPLICIT COMPLEX*16(Z)
C 1427: IMPLICIT COMPLEX*16(Z)
C 1428: DIMENSION ZR(MA,*),ZR1(MA,*)
C 1429: DIMENSION ZC11(MLY,*),ZC12(*MLY,*),ZC21(*MLY,*),ZC22(*MLY,*)
C 1430: COMMON /COND1/ KC,KP,KQ(4),NLAY,MJYD,MJYD,MFLG1,MUX,MWX
C 1431: C
C 1432: DO 1 L=1,MR1
C 1433: DO 1 K=1,MJYD
C 1434: ZR1(K,L)=ZR(K,L)
C 1435: 1 CONTINUE
C 1436: IF (MAU.EQ.0) GO TO 50
C 1437: DO 10 L=1,MAU
C 1438: DO 11 K=1,MAU
C 1439: 11 ZR1(K,L)=ZR(MUX*K+K,MUX*K+L)
C 1440: DO 12 K=1,MUX
C 1441: ZR1(K+MAU,L)=ZR(K,L+MUX*K)
C 1442: 12 ZR1(K+NLAY,L)=ZR(K+MUX,L+MUX*K)
C 1443: 10 CONTINUE
C 1444: C
C 1445: DO 13 L=1,MUX
C 1446: DO 14 K=1,MAU
C 1447: ZR1(K,L+MAU)=ZR(K+MUX*K,L)
C 1448: 14 ZR1(K,L+NLAY)=ZR(K+MUX*K, L+MUX)
C 1449: DO 15 K=1,MUX
C 1450: ZR1(K+MAU,L+MAU)=ZR(K,L)
C 1451: ZR1(K+NLAY,L+MAU)=ZR(K+MUX,L)
C 1452: ZR1(K+MAU,L+NLAY)=ZR(K,L+MUX)
C 1453: 15 ZR1(K+NLAY,L+NLAY)=ZR(K+MUX,L+MUX)
C 1454: 13 CONTINUE
C 1455: 50 CONTINUE
C 1456: C
C 1457: IF (NFLQ1.EQ.0) GO TO 100
C 1458: DO 4 L=1,NLAY
C 1459: DO 41 K=1,NLAY
C 1460: 41 ZC11(K,L)=ZR1(K,L)
C 1461: DO 42 K=1,MUX
C 1462: 42 ZC11(K,L)=ZR1(K+NLAY,L)
C 1463: 4 CONTINUE
C 1464: C
C 1465: DO 5 L=1,MUX
C 1466: DO 6 K=1,NLAY
C 1467: 6 ZC12(K,L)=ZR1(K,L+NLAY)
C 1468: DO 7 K=1,MUX
C 1469: 7 ZC22(K,L)=ZR1(K+NLAY,L+NLAY)
C 1470: 5 CONTINUE
C 1471: 100 CONTINUE
C 1472: C
C 1473: RETURN
C 1474: END

```

C ****

```

1: C ****
2: SUBROUTINE CARDIM(NFT, KP)
3: C ****
4: CHARACTER A$60, AD$95, AINP$19
5: DATA AD/, CARD NO. /, 1, 2, 3, 4, 5, 6, 7, 8/
6: DATA AINP/, INPUT DATA ECHO /, 0
7: NT = 0
8: READ(NFT, 500, END=101) A
9: IF(AD.EQ.' ') STOP
10: NT = NT + 1
11: NC=MOD(NT, 50)
12: IF(AD.EQ.'1') WRITE(KP, 600) NFT, AINP, AD
13: WRITE(KP, 602) NT, A
14: IF(AD.EQ.'0') WRITE(KP, 650) AD
15: GO TO 10
16: 101 CONTINUE
17: IF(AD.EQ.'0') WRITE(KP, 650) AD
18: WRITE(KP, 601) NT
19: REWIND NFT
20: 500 FORMAT(A80)
21: 600 FORMAT(1H1, /, 15X, 20(' *'), ' FILE CODE=' , I2, A19, 20(' *'), //, A95, /)
22: 601 FORMAT(1H , 30(' *'), 'DATA CARD END TOTAL CARD NUMBER =', 15)
23: 602 FORMAT(1H , 4X, 17, ' ;', A80)
24: 650 FORMAT(//, A95)
25: RETURN
26: END
27: C ****
28: SUBROUTINE COP4(ZWK, ZMK, MM, C2, C3, C4, CKK)
29: C ****
30: IMPLICIT COMPLEX*16(Z)
31: DIMENSION ZWK(MM,*), ZMKB(MM*2,*)
32: DIMENSION CK(4)
33: ZERO=COMPLX(0.0,0.0)
34: CK(1)=I_0
35: CK(2)=C2
36: CK(3)=C3
37: CK(4)=C4
38: C
39: DD_1 N=1, 4
40: C=CK(N)*CKK
41: K0=NM*MOD(N-1, 2)
42: L0=(N-1)/2
43: L0=L*MM
44: DD_2 K=1, MM
45: KK=K0+K
46: DD_2 L=1, MM
47: LL=L*MM
48: ZMB(KK, LL)=C*ZWK(K, L)
49: 2 CONTINUE
50: 1 CONTINUE
51: DD_3 K=1, MM
52: DD_3 L=1, MM
53: ZMK(K, L)=ZERO
54: 3 CONTINUE
55: RETURN
56: END
57: C ****

```

C ****

```

58: SUBROUTINE POUTM(ZA, N, M, MP, M2)
59: C ****
60:   IMPLICIT COMPLEX*16(Z)
61:   DIMENSION ZA(M2,*)
62: C
63:   IP=(M-1)/10+1
64:   MN=MOD(M-1, 10)+1
65:   WRITE(KP, 600) N, M
66: C
67: C   REAL PART
68: C
69:   WRITE(KP, 601)
70:   DO 1 11=1, IP
71:   IE=10
72:   IF(11, EQ, IP) IE=MNN
73:   DD(2,J)=1, N
74:   IO=(11-1)*10
75:   2 WRITE(KP, 602) J, (REAL(ZA(J, I+IO)), I=1, IE)
76:   1 CONTINUE
77: C
78: C   IMAGINARY PART
79: C
80:   WRITE(KP, 603)
81:   DO 3 11=1, IP
82:   IE=10
83:   IF(11, EQ, IP) IE=MNN
84:   DD(4,J)=1, N
85:   IO=(11-1)*10
86:   4 WRITE(KP, 602) J, (AIMAG(ZA(J, I+IO)), I=1, IE)
87:   3 CONTINUE
88:   600 FORMAT(1H/, 'CHECK WRITE MATRIX N= ', IE, 5X, 'M= ', 15)
89:   601 FORMAT(1H, '--- REAL PART ---')
90:   602 FORMAT(1H, '15.5X, 10E12, 3)
91:   603 FORMAT(1H, '--- IMAGINARY PART ---')
92:   RETURN
93: C ****
94: C ****
95:   SUBROUTINE REBARR(ZIN, ZTEMP, MN, M2, MMOD)
96: C ****
97:   IMPLICIT COMPLEX*16(Z)
98:   DIMENSION ZIN(M2,*), ZTEMP(M2,*)
99: C
100:  DO 1  J=1, MN
101:  DO 1  N=1, MN
102:  1  ZTEMP(N, J)=ZIN(N, J)
103:  N1=MN/MMOD
104:  DO 10  I=1, MN
105:  1, J=MOD(I-1, MMOD)
106:  11=(J-1)/MMOD+N1*I, J=1
107:  DO 20  J=1, MN
108:  1, J=MOD(J-1, MMOD)
109:  J1=(J-1)/MMOD+N1*I, J=1
110:  ZIN(J1, 11)=ZTEMP(J, I)
111:  20 CONTINUE
112:  10 CONTINUE
113:  RETURN
114: END

```

```

115: C ****
116:      SUBROUTINE ZTRN( M1, ZOT, MO, M2 )
117: C ****
118:      IMPLICIT COMPLEX*16(Z)
119:      DIMENSION ZIN(M1,*),ZOT(M2,*)
120: C
121:      DO 1 L=1, M1
122:      LL=L+MO-1
123:      DO 2 K=1, M1
124:      KK=K+MO-1
125:      ZOT(KK,LL)=ZOT(KK,LL)+ZIN(K,L)
126:      2 CONTINUE
127:      1 CONTINUE
128:      RETURN
129: END
130: C ****
131:      SUBROUTINE ZINV1( A, N, M, NM2 )
132: C ****
133:      REAL*8 A(NM2,1),PA,PR,PI,WR,WI, ONE, ZERO
134:      DATA ONE, ZERO / 1. D+00, 0.D+00 /
135: C
136:      DO 40 K=1, N
137:      PA = A(2*K-1,K)*ZOT(2*K, K)*#2
138:      PR = A(2*K-1,K)/PA
139:      P1 = -A(2*K ,K)/PA
140:      A(2*K-1,K) = ONE
141:      A(2*K ,K) = ZERO
142:      DO 10 J=1, M
143:      PA = A(2*K-1,J)
144:      A(2*K-1,J) = A(2*K-1,J)*PR-A(2*K, J)*PI
145:      A(2*K ,J) = PA*P1+A(2*K, J)*PR
146:      10 CONTINUE
147:      DO 30 I=1, N
148:      IF (K, EQ, 1) GO TO 30
149:      WR = -A(2*I-1,K)
150:      WI = -A(2*I ,K)
151:      A(2*I-1,K) = ZERO
152:      A(2*I ,K) = ZERO
153:      DO 20 J=1, M
154:      A(2*I-1,J) = A(2*I-1,J)+A(2*K-1,J)*WR-A(2*K, J)*WI
155:      A(2*I ,J) = A(2*I ,J)+A(2*K-1,J)*WI+A(2*K, J)*WR
156:      20 CONTINUE
157:      30 CONTINUE
158:      40 CONTINUE
159:      RETURN
160: END
161: C ****
162:      SUBROUTINE ZMULT1(ZA, ZB, ZC, M1, M2, MK, MC)
163: C ****
164:      IMPLICIT COMPLEX*16(Z)
165:      DIMENSION ZA(MA,*), ZB(MB,*), ZC(MC,*)
166: C
167:      DO 10 L=1, MK
168:      DO 20 K=1, M1
169:      Z1=CMPLX(0.0,0.0)
170:      DO 30 N=1, M2
171:      Z2=ZA(K,N)*ZB(N,L)

```

```

C ****
172: Z3=Z1
173: Z1=Z2+Z3
174: 30 CONTINUE
175: ZC(K,L)=Z1
176: 20 CONTINUE
177: 10 CONTINUE
178: RETURN
179: END
180: C ****SUBROUTINE ZTENCH(ZIN,ZTEN,MM,MM)
181: C ****
182: C ****
183: IMPLICIT COMPLEX*16(Z)
184: DIMENSION ZIN(MM,*),ZTEN(MM,*)
185: C
186: DO 1 K=1,MM
187: DO 1 L=1,MM
188: ZTEN(K,L)=ZIN(L,K)
189: 1 CONTINUE
190: RETURN
191: END
192: C ****
193: C ****SUBROUTINE ZTM1(ZIN,MM,MA)
194: C ****
195: IMPLICIT COMPLEX*16(Z)
196: DIMENSION ZIN(MA,*)
197: C
198: DO 1 K=1,MM-1
199: L1=K+1
200: DO 2 L=L1,MM
201: ZIN(K,L)=ZIN(L,K)
202: 2 CONTINUE
203: 1 CONTINUE
204: RETURN
205: END
206: C ****
207: C ****SUBROUTINE ZERCL(ZA,N,M,MDIM)
208: C ****
209: IMPLICIT COMPLEX*16(Z)
210: DIMENSION ZA(MDIM,*)
211: ZERO=COMPLX(0.0,0.0)
212: DO 1 K=1,N
213: DO 1 L=1,M
214: ZA(K,L)=ZERO
215: 1 CONTINUE
216: RETURN
217: END
218: C ****
219: C ****SUBROUTINE CG(NMN,AR,AI,WR,WI,MATZ,ZR,ZI,FV1,FV2,FV3,IERR)
220: C
221: INTEGER N,NM,IS1,IS2,IERR,MATZ
222: REAL*B AR(NM,N),AI(NM,N),WR(N,N),WI(N,N),ZR(NM,N),ZI(NM,N),
223:      $ FV1(N),FV2(N),FV3(N)
224: C
225: C
226: C This subroutine calls the recommended sequence of
227: C subroutines from the eigensystem subroutine package (EISPACK)
228: C to find the eigenvalues and eigenvectors (if desired)

```

C ****

```

229: C of a complex general matrix.
230: C
231: C On input:
232: C
233: C NM must be set to the row dimension of the two-dimensional
array parameters as declared in the calling program
dimension statement;
234: C
235: C
236: C
237: C N is the order of the matrix A*(AR,AI);
238: C
239: C AR and AI contain the real and imaginary parts,
respectively, of the complex general matrix;
240: C
241: C
242: C MATZ is an integer variable set equal to zero if
only eigenvalues are desired; otherwise it is set to
any non-zero integer for both eigenvalues and eigenvectors.
243: C
244: C
245: C On output:
246: C
247: C WR and WI contain the real and imaginary parts,
248: C respectively, of the eigenvalues;
249: C
250: C ZR and ZI contain the real and imaginary parts,
251: C respectively, of the eigenvectors if MATZ is not zero;
252: C
253: C JERR is an integer output variable set equal to an
error completion code described in section 2B of the
254: C documentation. The normal completion code is zero;
255: C
256: C
257: C FV1, FV2, and FV3 are temporary storage arrays.
258: C
259: C Questions and comments should be directed to R. S. Garbow,
260: C Applied Mathematics division, Argonne National Laboratory
261: C
262: C
263: C
264: C
265: C IF (N .LE. -NM) GO TO 10
266: C JERR = 10 * N
267: C GO TO 50
268: C
269: 10 CALL CBAL(NM,N,AR,AI,IS1,IS2,FV1)
270:     CALL CORTH(NM,N,IS1,IS2,AR,AI,FV2,FV3)
271:     IF (MATZ .NE. 0) GO TO 20
272:     ::::: find eigenvalues only :::::
273:     CALL COMQR(NM,N,IS1,IS2,AR,AI,WR,WI,IERR)
274:     GO TO 50
275:     ::::: find both eigenvalues and eigenvectors :::::
276:     20 CALL CDQR2(NM,N,IS1,IS2,FV2,FV3,AR,AI,WR,WI,ZR,ZI,IERR)
277:     IF (IERR .NE. 0) GO TO 50
278:     CALL CRBK2(NM,N,IS1,IS2,FV1,N,ZR,ZI)
279:     GO RETURN
280: C
281:     ::::: last card of CG ::::::
282: C ****
283: SUBROUTINE CBABK2(NM,N,LOW,IGH,SCALE,M,ZR,ZI)
284: C
285: INTEGER I,J,K,M,N,II,NM,IGH,LOW

```

C ****

```
286: REAL*8 SCALE(N),ZR(NM,M),ZI(NM,M)
287: REAL*8 S
288: C
289: C This subroutine is a transiation of the algol procedure
290: C CBALIC, which is a complex version of BALBAK.
291: C Num. Math. 13, 293-304(1969) by Parlett and Reinsch.
292: C Handbook for auto. comp., Vol. II-Linear Algebra, 315-326(1971).
293: C
294: C This subroutine forms the eigenvectors of a complex general
295: C matrix by back transforming those of the corresponding
296: C balanced matrix determined by CBAL.
297: C
298: C On input:
299: C
300: C NM must be set to the row dimension of two-dimension one.
301: C array parameters as declared in the calling program
302: C DIMENSION statement;
303: C
304: C N is the order of the matrix;
305: C
306: C LOW and IGH are integers determined by CBAL;
307: C
308: C SCALE contains information determining the permutations
309: C and scaling factors used by CBAL;
310: C
311: C M is the number of eigenvectors to be back transformed;
312: C
313: C ZR and ZI contain the real and imaginary parts,
314: C respectively, of the eigenvectors to be
315: C back transformed in their first M columns.
316: C
317: C On output:
318: C
319: C ZR and ZI contain the real and imaginary parts,
320: C respectively, of the transformed eigenvectors
321: C in their first M columns.
322: C
323: C Questions and comments should be directed to B. S. Garbow,
324: C Applied Mathematics Division, Argonne National Laboratory
325: C
326: C
327: C
328: IF (M .EQ. 0) GO TO 200
329: IF (IGH .EQ. LOW) GO TO 120
330: C
331: DO 110 I = LOW, IGH
332: S = SCALE(I)
333: C :.....: left hand eigenvectors are back transformed
334: C if the foregoing statement is replaced by
335: C S=1.0D0/SCALE(I).
336: DO 100 J = 1, M
337: ZR(I,J) = ZR(I,J) * S
338: ZI(I,J) = ZI(I,J) * S
339: 100 CONTINUE
340: C
341: 110 CONTINUE :.....: FOR I=LOW-1 STEP -1 UNTIL 1,
342: C
```

C ****

```

343: C          IGH+1 STEP 1 UNTIL N DO ----
344: 120 DO 140 I1 = 1, N
345:   I = I1
346:   IF (I .GE. LOW .AND. I .LE. IGH) GO TO 140
347:   IF (I .LT. LOW) I = LOW - I1
348:   K = SCALE(I)
349:   IF (K .EQ. 1) GO TO 140
350: C
351:   DO 130 J = 1, M
352:     S = ZR(I,J)
353:     ZR(I,J) = ZR(K,J)
354:     ZR(K,J) = S
355:     S = ZI(I,J)
356:     ZI(I,J) = ZI(K,J)
357:     ZI(K,J) = S
358:   130 CONTINUE
359: C
360: 140 CONTINUE
361: C
362: 200 RETURN
363: C      last card of CBABK2
364: END
365: C*****
366: C***** SUBROUTINE CBAL(NM,N,AR,AI,LOW,IGH,SCALE)
367: C
368:      INTEGER I, J, K, L, M, N, JJ, NM, IGH, LOW, IEXC
369:      REAL*8 AR(NM,N), AI(NM,N), SCALE(N)
370:      REAL*8 C, F, G, R, S, B2, RADIX
371:      REAL*8 DABS
372:      LOGICAL NOCONV
373: C
374: C
375: C This subroutine is a translation of the ALGOL procedure
376: C BALANCE, which is a complex version of BALANCE,
377: C Num. Math. 13, 293-304(1969) by Parlett and Reinsch.
378: C Handbook for Auto. Comp., Vol. II-Linear Algebra, 315-326(1971).
379: C
380: C This subroutine balances a complex matrix and isolates
381: C eigenvalues whenever possible.
382: C
383: C On input:
384: C
385: C NM must be set to the row dimension of two-dimensional
386: C array parameters as declared in the calling program.
387: C DIMENSION statement;
388: C
389: C N is the order of the matrix;
390: C
391: C AR and AI contain the real and imaginary parts,
392: C respectively, of the complex matrix to be balanced.
393: C
394: C On output:
395: C
396: C AR and AI contain the real and imaginary parts,
397: C respectively, of the balanced matrix;
398: C
399: C LOW and IGH are two integers such that AR(I,J) and AI(I,J)
```


卷之三

```

514: IF (C .EQ. 0. ODO .OR. R .EQ. 0. ODO) GO TO 270
515: C = R / RADIX
516: F = 1. ODO
517: S = C + R
518: 210 IF (C .GE. 0) GO TO 220
519: F = F * RADIX
520: C = C * B2
521: GO TO 210
522: 220 G = R * RADIX
523: IF (C .LT. G) GO TO 240
524: F = F / RADIX
525: C = C / B2
526: GO TO 230
527: C : : : Now balance : : :
528: 240 IF ((C + R) / F .GE. 0.95DO * S) GO TO 270
529: C = 1. ODO / F
530: SCALE(I) = SCALE(I) * F
531: NDCONV = .TRUE.
532: C
533: DD 250 J = K, N
534: AR(I,J) = AR(I,J) * C
535: AI(I,J) = AI(I,J) * C
536: CONTINUE
537: C
538: DD 260 J = 1, L
539: AR(J,I) = AR(J,I) * F
540: AI(J,I) = AI(J,I) * F
541: CONTINUE
542: C
543: 270 CONTINUE
544: C
545: IF (NDCONV) GO TO 190
546: C
547: 280 LOW = K
548: IGH = L
549: RETURN
550: C
551: **** last card of CBAL ****
552: C*****
553: 952: C***** SUBROUTINE CGNR(NM, NLW, IGH, ITL, LOW, LP1, ENM1, IEF)
554: C*****
      INTEGER I, J, L, NM, IGH, ITL, LOW, LP1, ENM1, IEF
      REAL#B HR(NM,N), HI(NM,N), WR(N), WI(N)
      REAL#B SI, SR, TI, TR, XI, XR, YI, YR, Z21, Z22R, NORM, MACHM
      REAL#B DSGRT, CDABS, DBBS
      INTEGER MINO
      COMPLEX#16 ZB3
      REAL#B DREAL, DIMAG
      COMPLEX#16 CDSORT, DCMPLEX
      REAL#B DREAL, DIMAG
      :::::::::: Statement functions enable extraction
      :::::::::: imaginary parts of double precision complex numbers
      :::::::::: Statement functions enable extraction
      :::::::::: imaginary parts of double precision complex numbers
      DREAL(Z3) = Z3
      DIMAG(Z3) = (0. ODO, -1. ODO) * Z3
      :::::::::: This subroutine is a translation of a unitary and
      :::::::::: ALGOL procedure COMLR, Num. Math. 12, 369-376(1968)
      :::::::::: and Wilkinson.

```

C ****

Handbook for Auto. Comp., Vol. II-Linear Algebra, 396-403(1971).
 The unitary analogue substitutes the QR algorithm of Francis.
 (Comp. Jour. 4, 332-345(1962) for the LR Algorithm.

575: C This subroutine finds the eigenvalues of a complex
 upper Hessenberg matrix by the QR method.

577: C

On input:

579: C NM must be set to the row dimension of two-dimensional
 array parameters as declared in the calling program.

580: C

581: C

582: C

583: C

584: C N is the order of the matrix;

585: C

586: C LOW and IGH are integers determined by the balancing
 subroutine CBAL. If CBAL has not been used,
 dimension statement,

587: C set LOW=1, IGH=N.

588: C

589: C

590: C HR and HI contain the real and imaginary parts,
 respectively, of the complex upper Hessenberg matrix.

591: C 592: C their lower triangles below the subdiagonals contain
 593: C information about the unitary transformations used in
 594: C the reduction by CGUTH, if performed.

595: C

On output:

597: C

598: C The upper Hessenberg portions of HR and HI have been
 599: C destroyed. Therefore, they must be saved before
 calling CDQR if subsequent calculation of
 eigenvectors is to be performed;

600: C

601: C

602: C WR and WI contain the real and imaginary parts,
 respectively, of the eigenvalues. If an error
 exit is made, the eigenvalues should be correct
 for indices IERR+1,...,N.

603: C

604: C

605: C

606: C

607: C

608: C IERR is set to

609: C ZERO for normal return,
 610: C J if the J-th eigenvalue has not been
 611: C determined after 30 iterations.

612: C

613: C arithmetic is real except for the replacement of the ALGOL
 procedure CDIV by complex division and use of the subroutines
 CDSQRT and DCMPXL in computing complex square roots.

614: C

615: C

616: C

617: C Questions and comments should be directed to B. S. Garbow,
 Applied Mathematics Division, Argonne National Laboratory

618: C

619: C

620: C

621: C

622: C

623: C MACHEP is a machine dependent parameter specifying
 the relative precision of floating point arithmetic.

624: C MACHEP = 16. QDO**(-13) for long form arithmetic.

625: C on S360

626: C DATA MACHEP / 1. 421D-14/

627: C

```

C ****
      IERR = 0
      IF (LOW .EQ. IGH) GO TO 180
      630: C   ::::: create real subdiagonal elements
      L = LOW + 1
      632: C
      633: DO 170 I = L, IGH
            LL = MIN(I,I+1,IGH)
            IF (HI(I,I-1) .EQ. 0.0D0) GO TO 170
            NORM = CDABS(DCMPLX(HR(I,I-1),HI(I,I-1)))
            YR = HR(I,I-1) / NORM
            VI = HI(I,I-1) / NORM
            HR(I,I-1) = NORM
            HI(I,I-1) = 0.0D0
      634: CONTINUE
      635: DO 155 J = I, IGH
            SI = YR * HI(I,J) - VI * HR(I,J)
            HR(I,J) = YR * HR(I,J) + VI * HI(I,J)
            HI(I,J) = SI
      636: CONTINUE
      637: 155
      638: CONTINUE
      639: 160
      640: CONTINUE
      641: C
      642: DO 155 J = LOW, LL
            SI = YR * HI(J,1) + VI * HR(J,1)
            HR(J,1) = YR * HR(J,1) - VI * HI(J,1)
            HI(J,1) = SI
      643: CONTINUE
      644: 155
      645: CONTINUE
      646: 160
      647: C
      648: DO 160 J = LOW, LL
            SI = YR * HI(J,1) + VI * HR(J,1)
            HR(J,1) = YR * HR(J,1) - VI * HI(J,1)
            HI(J,1) = SI
      649: CONTINUE
      650: 160
      651: CONTINUE
      652: 160
      653: C
      654: 170 CONTINUE
      655: C   ::::: Store root isolated by CBAL ::::::::::::
      656: 180 DO 200 I = 1, N
            IF (I .GE. LOW .AND. I .LE. IGH) GO TO 200
            WR(I) = HR(I,I)
            WI(I) = HI(I,I)
      657: 200 CONTINUE
      658: 180
      659: 180
      660: 200
      661: C
      662: EN = IGH
      663: TR = 0.0D0
      664: TI = 0.0D0
      665: C   ::::: Search for next eigenvalue ::::::::::::
      666: 220 IF (EN .LT. LOW) GO TO 1001
      667: ITS = 0
      668: ENM1 = EN - 1
      669: C   ::::: Look for single small sub-diagonal element
            FOR L=EN STEP -1 UNTIL LOW DO -- ::::::::::::
      670: C   240 DO 260 LL = LOW, EN
            L = EN + LOW - LL
      671: 260
      672: IF (L .EQ. LL) GO TO 300
            IF (L .EQ. LOW) GO TO 300
            IF (DABS(HR(L,L-1)) .LE.
                MACHEP * (DABS(HR(L-1,L-1)) + DABS(HI(L-1,L-1))
                X + DABS(HR(L,L))) +DABS(HI(L,L))) GO TO 300
      673: 300
      674: 260 CONTINUE
      675: C   ::::: Form shift ::::::::::::
      676: 300 IF (L .EQ. EN) GO TO 660
            IF (ITS .EQ. 30) GO TO 1000
      677: 680: IF (ITS .EQ. 10 .OR. ITS .EQ. 20) GO TO 320
            SR = HR(EN,EN)
            SI = HI(EN,EN)
            XR = HR(ENM1,EN) * HR(EN,ENM1)
      678: C
      679: 300
      680: 681: IF (ITS .EQ. 10 .OR. ITS .EQ. 20) GO TO 320
            SR = HR(EN,EN)
            SI = HI(EN,EN)
            XR = HR(ENM1,EN) * HR(EN,ENM1)
      682: 683: 684:

```

C *****

```

585: XI = HI(ENM1, EN) * HR(EN, ENM1)
      IF (XR .EQ. 0.0D0) AND. XI .EQ. 0.0D0) GO TO 340
586: YR = (HR(ENM1, ENM1) - SR) / 2.0D0
587: YI = (HI(ENM1, ENM1) - SI) / 2.0D0
588: Z3 = CDSQRT(DCMPLX(YR**2+YI**2+XR, 2.0D0*YR*YI+XI))
589: ZZR = DREAL(Z3)
590: ZZI = DIMAG(Z3)
591: IF (YR * ZZR + YI * ZZI .GE. 0.0D0) GO TO 310
592: ZZR = -ZZR
593: ZZI = -ZZI
594: Z3 = DCMPLX(XR, XI) / DCMPLX(YR+ZZR, YI+ZZI)
595: SR = SR - DREAL(Z3)
596: SI = SI - DIMAG(Z3)
597: GO TO 340
598: C ::::::::::::::: Form exceptional shift ::::::::::::::
599: 320 SR = DABS(HR(EN, ENM1)) + DABS(HR(ENM1, EN-2))
600: 320 SR = 0.0D0
601: SI = 0.0D0
602: C
603: 340 DO 360 I = LOW, EN
      HR(I, I) = HR(I, I) - SR
      HI(I, I) = HI(I, I) - SI
360 CONTINUE
604: TI = TI + SI
605: ITS = ITS + 1
606: C :::::::::::::: Reduce to triangle (rows) ::::::::::::::
607: C
608: TR = TR + SR
609: TI = TI + SI
610: ITS = ITS + 1
611: C
612: LP1 = L + 1
613: C
614: DO 500 I = LP1, EN
      SR = HR(I, I-1)
      HR(I, I-1) = 0.0D0
      NORM = DSQRT(HR(I-1, I-1)*HR(I-1, I-1)+HI(I-1, I-1))
      X
      XR = HR(I-1, I-1) / NORM
      WR(I-1) = XR
      XI = HI(I-1, I-1) / NORM
      WI(I-1) = XI
      HR(I-1, I-1) = NORM
      HI(I-1, I-1) = 0.0D0
      HI(I, I-1) = SR / NORM
500 CONTINUE
615: C
616: DO 490 J = I, EN
      VR = HR(I-J, J)
      YI = HI(I-J, J)
      ZZR = HR(I-J, J)
      ZZI = HI(I-J, J)
      HR(I-J, J) = XR * VR + XI * YI + HI(I, I-1) * ZZR
      HI(I-1, J) = XR * YI - XI * VR + HI(I, I-1) * ZZI
      HR(I, J) = XR * ZZR - XI * ZZI - HI(I, I-1) * YR
      HI(I, J) = XR * ZZI + XI * ZZR - HI(I, I-1) * YI
490 CONTINUE
617: C
618: 500 CONTINUE
619: C
620: SI = HI(EN, EN)
621: IF (SI .EQ. 0.0D0) GO TO 340
622: C
623: C
624: C
625: C
626: C
627: C
628: C
629: C
630: C
631: C
632: C
633: C
634: C
635: C
636: C
637: C
638: C
639: C
640: C
641: C

```

C ****

```

742:      NORM = CDABS(DCMPLX(HR(EN,EN),SI))
743:      SR = HR(EN,EN) / NORM
744:      SI = SI / NORM
745:      HR(EN,EN) = NORM
746:      HI(EN,EN) = 0.0D0
747:      C   747. C  Inverse operation (columns) ::::::::::::
748:      540 DO 600 J = LP1, EN
749:      XR = WR(J-1)
750:      XI = WI(J-1)
751:      C
752:      DO 580 I = L, J
753:      YR = HR(I,J-1)
754:      YI = 0.0D0
755:      ZZR = HR(I,J)
756:      ZZI = HI(I,J)
757:      IF (I .EQ. J) GO TO 560
758:      YI = HI(I,J-1)
759:      HI(I,J-1) = XR * YI + XI * YR + HI(J,J-1) * ZZI
760:      560  HR(I,J-1) = XR * YR - XI * YI + HI(J,J-1) * ZZR
761:      HR(I,J) = XR * ZZR + XI * ZZI - HI(J,J-1) * YR
762:      HI(I,J) = XR * ZZI - XI * ZZR - HI(J,J-1) * YI
763:      580  CONTINUE
764:      C
765:      600  CONTINUE
766:      C
767:      IF (SI .EQ. 0.0D0) GO TO 240
768:      C
769:      DO 630 I = L, EN
770:      YR = HR(I,EN)
771:      YI = HI(I,EN)
772:      HR(I,EN) = SR * YR - SI * YI
773:      HI(I,EN) = SR * YI + SI * YR
774:      630  CONTINUE
775:      C
776:      GO TO 240
777:      C  :::::::::::: A root found ::::::::::::
778:      660  (HR(EN)) = HR(EN,EN) + TR
779:      (WI(EN)) = HI(EN,EN) + TI
780:      EN = EN+1
781:      GO TO 220
782:      C  :::::::::::: Set error -- no convergence to an
783:      C  eigenvalue after 30 iterations ::::::::::::
784:      1000 TERR = EN
785:      1001 RETURN
786:      C  :::::::::::: last card of CDM9R ::::::::::::
787:      END
788:      C **** SUBROUTINE CONGR2NM, N, LOW, IGH, ORTR, ORTI, HR, HI, WR, WI, ZR, ZI, TERR ****
789:      C
790:      C
791:      INTEGER I,J,K,L,M,N,EN,II,JJ,LL,NM,NH,IGH,IP1,
792:      X     ITS,LOW,LP1,ENM1,IEND,IERR
793:      REAL*8 HR(NM,N),HI(NM,N),WR(N),WI(N),ZR(NM,N),ZI(NM,N),
794:      X     ORTR(IGH),ORTI(IGH)
795:      REAL*8 SI,SR,TL,TR,XI,XR,YI,YR,ZZI,ZZR,NORM,MACHEP
796:      REAL*8 DSGRT,CDABS,DABS
797:      INTEGER MINO
798:      COMPLEX*16 Z3

```

```

COMPLEX*16 CDSORT,DCMPLX
REAL*8 DREAL,DIMAG
      :: Statement functions enable extraction of real and
      :: imaginary parts of double precision complex numbers
      :: DREAL(Z3) = Z3
      :: DIMAG(Z3) = (0.0D0,-1.0D0)*Z3

      :: This subroutine is a translation of a unitary analogue of the
      :: ALGOL procedure COMLR2, Num. Math. 16, 181-204(1970) by Peters
      :: and Wilkinson.
      :: Handbook for Auto. Comp., Vol. III-linear Algebra, 372-395(1971).
      :: The unitary analogue substitutes the QR algorithm of Francis
      :: (Comp. Jour. 4, 332-345(1962)) for the LR algorithm.

      :: This subroutine finds the eigenvalues and eigenvectors
      :: of a complex upper Hessenberg matrix by the QR
      :: method. The eigenvectors of a complex general matrix
      :: can also be found if CORTH has been used to reduce
      :: this general matrix to Hessenberg form.

      :: On input:
      :: NM must be set to the row dimension of two-dimensional
      :: array parameters as declared in the calling program
      :: DIMENSION statement;

      :: N is the order of the matrix;
      :: LOW and IGH are integers determined by the balancing
      :: subroutine CBAL. If CBAL has not been used,
      :: set LOW=1, IGH=N;
      :: ORTR and ORTI contain information about the unitary trans-
      :: formations used in the reduction by CORTH; if performed,
      :: only elements LOW through IGH are used. If the eigenvectors
      :: of the hessenberg matrix are desired, set ORTR(J) and
      :: ORTI(J) to 0.0D0 for these elements;
      :: HR and HI contain the real and imaginary parts,
      :: respectively, of the complex upper Hessenberg matrix.
      :: Their lower triangles below the subdiagonal contain further
      :: information about the transformations which were used in the
      :: reduction by CORTH; if performed, the eigenvectors of
      :: the Hessenberg matrix are desired, these elements may be
      :: arbitrary.

      :: On output:
      :: ORTR, ORTI, and the upper Hessenberg portions of HR and HI
      :: have been destroyed;
      :: WR and WI contain the real and imaginary parts,
      :: respectively, of the eigenvalues. If an error
      :: exit is made, the eigenvalues should be correct
      :: for indices IERR+1,...,N;
      :: ZR and ZI contain the real and imaginary parts;

```

C ****

```

855: C   respectively, of the eigenvectors. The eigenvectors
856: C   are unnormalized. If an error exit is made, none of
857: C   the eigenvectors has been found.
858: C
859: C   IERR is set to
860: C     zero      for normal return,
861: C     J        if the J-th eigenvalue has not been
862: C     determined after 30 iterations.
863: C
864: C   Arithmetic is real except for the replacement of the ALGOL
865: C   procedure CDIV by complex division and use of the subroutines
866: C   CDSQRT and DCMPLEX in computing complex square roots.
867: C
868: C
869: C   Questions and comments should be directed to B. S. Garbow,
870: C   Applied Mathematics Division, Argonne National Laboratory
871: C
872: C
873: C
874: C   ..... MACHEP is a machine dependent parameter specifying
875: C   the relative precision of floating point arithmetic.
876: C   MACHEP = 16.0D0**(-13) for long form arithmetic
877: C   on S360 : : : : :
878: C   DATA MACHEP/1.421D-14/
879: C
880: C   IERR = 0
881: C   ..... Initialize eigenvector matrix : : : : :
882: DO 100 I = 1, N
883: C
884: DO 100 J = 1, N
885: ZR(I,J) = 0.0D0
886: ZI(I,J) = 0.0D0
887: IF (I .EQ. J) ZR(I,J) = 1.0D0
888: 100 CONTINUE
889: C   ..... Form the matrix of accumulated transformations
890: C   from the information left by CORTH : : : : :
891: C   IEND = IGH - LOW - 1
892: IF (IEND) 180, 150, 105
893: 105 DO 140 IHIGH-1 STEP -1 UNTIL LCHM+1 DO -- : : : : :
894: 105 DO 140 II = 1, IEND
895: I = IGH - II
896: IF (ORTR(I) .EQ. 0.0D0 .AND. ORTI(I) .EQ. 0.0D0) GO TO 140
897: IF (HR(I,I-1) .EQ. 0.0D0 .AND. HI(I,I-1) .EQ. 0.0D0) GO TO 140
898: C   ..... Norm below is negative of H formed in CORTH : : : : :
899: NORM = HR(I,I-1) * ORTR(I) + HI(I,I-1) * ORTI(I)
900: IP1 = I + 1
901: C
902: DO 110 K = IP1, IGH
903: ORTR(K) = HR(K,I-1)
904: ORTI(K) = HI(K,I-1)
905: 110  CONTINUE
906: C
907: DO 130 J = 1, IGH
908: SR = 0.0D0
909: SI = 0.0D0
910: C
911: DO 115 K = 1, IGH
912: SR = SR + ORTR(K) * ZR(K,J) + ORTI(K) * ZI(K,J)

```

C ****

Page 17

```
913:    SI = SI + ORTR(K) * ZI(K,J) - ORTI(K) * ZR(K,J)
914: 115      CONTINUE
915: C
916:      SR = SR / NORM
917:      SI = SI / NORM
918: C
919:      DO 120 K = I, IGH
920:        ZR(K,J) = ZR(K,J) + SR * ORTR(K) - SI * ORTI(K)
921:        ZI(K,J) = ZI(K,J) + SR * ORTI(K) + SI * ORTR(K)
922:      120      CONTINUE
923: C
924: 130      CONTINUE
925: C
926: 140      CONTINUE
927: C      :: Create real subdiagonal elements ::::::::::::
928: 150      L = LOW + 1
929: C
930:      DO 170 I = L, IGH
931:        LL = MIN(I+1,IGH)
932:        IF (HI(I,I-1).EQ.0.0D0) GO TO 170
933:        NORM = CDABS(DCMPLX(HR(I,I-1),HI(I,I-1)))
934:        YR = HR(I,I-1) / NORM
935:        YI = HI(I,I-1) / NORM
936:        HR(I,I-1) = NORM
937:        HI(I,I-1) = 0.0D0
938: C
939:      DO 155 J = I, N
940:        SI = YR * HI(I,J) - YI * HR(I,J)
941:        HR(I,J) = YR * HR(I,J) + YI * HI(I,J)
942:        HI(I,J) = SI
943:      155      CONTINUE
944: C
945:      DO 160 J = I, LL
946:        SI = YR * HI(J,I) + YI * HR(J,I)
947:        HR(J,I) = YR * HR(J,I) - YI * HI(J,I)
948:        HI(J,I) = SI
949:      160      CONTINUE
950: C
951:      DO 165 J = LOW, IGH
952:        SI = YR * ZI(J,I) + YI * ZR(J,I)
953:        ZR(J,I) = YR * ZR(J,I) - YI * ZI(J,I)
954:        ZI(J,I) = SI
955:      165      CONTINUE
956: C
957: 170      CONTINUE
958: C      :: Store roots isolated by CBAL ::::::::::::
959: 180      DO 200 I = 1, N
960:        IF (I.GE. LOW .AND. I.LE. IGH) GO TO 200
961:        WR(I) = HR(I,I)
962:        WI(I) = HI(I,I)
963: 200      CONTINUE
964: C
965:      EN = IGH
966:      TR = 0.0D0
967:      TI = 0.0D0
968: C      :: Search for next eigenvalue ::::::::::::
969: 220      IF (EN .LT. LOW) GO TO 480
```

C ****

```

970:    ITS = 0
971:    ENM1 = EN - 1
972:    C      Look for single small sub-diagonal element
973:    FOR L=EN STEP -1 UNTIL LOW DO ::::::::::::
974:    240 DO 260 LL = LOW, EN
975:    L = EN + LOW - LL
976:    IF (L .EQ. LOW) GO TO 300
977:    IF (DABS(HR(L,L-1)) .LE.
978:        X     MACHEP * (DABS(HR(L-1,L-1)) + DABS(HI(L-1,L-1))
979:        X     + DABS(HR(L,L))) +DABS(HI(L,L))) GO TO 300
980:    260 CONTINUE
981:    C      Form shift ::::::::::::
982:    300 IF (L .EQ. EN) GO TO 660
983:    IF (ITS .EQ. 30) GO TO 1000
984:    IF (ITS .EQ. 10 .OR. ITS .EQ. 20) GO TO 320
985:    SR = HR(EN, EN)
986:    SI = HI(EN, EN)
987:    XR = HR(ENM1, EN) * HR(EN, ENM1)
988:    XI = HI(ENM1, EN) * HR(EN, ENM1)
989:    IF (XR .EQ. 0.0D0 .AND. XI .EQ. 0.0D0) GO TO 340
990:    YR = (HR(ENM1, ENM1) - SR) / 2.0D0
991:    YI = (HI(ENM1, ENM1) - SI) / 2.0D0
992:    Z3 = CDSQRT(DCMPLX(YR**2-YI**2+XR, 2.0D0*YR*YI+XI))
993:    ZZR = DREAL(Z3)
994:    ZZI = DIMAG(Z3)
995:    IF (YR * ZZR + YI * ZZI .GE. 0.0D0) GO TO 310
996:    ZZR = -ZZR
997:    ZZI = -ZZI
998:    310 Z3 = DCMPLX(XR, XI) / DCMPLX(YR+ZZR, YI+ZZI)
999:    SR = SR - DREAL(Z3)
1000:   SI = SI - DIMAG(Z3)
1001:   GO TO 340
1002:   C      Form exceptional shift ::::::::::::
1003:   320 SR = DABS(HR(EN, ENM1)) + DABS(HR(ENM1, EN-2))
1004:   SI = 0.0D0
1005:   C
1006:   340 DO 360 I = LOW, EN
1007:       HR(I,I) = HR(I,I) - SR
1008:       HI(I,I) = HI(I,I) - SI
1009:   360 CONTINUE
1010:   C
1011:   TR = TR + SR
1012:   TI = TI + SI
1013:   ITS = ITS + 1
1014:   C      Reduce to triangle (rows) ::::::::::::
1015:   LP1 = L + 1
1016:   C
1017:   DO 500 I = LP1, EN
1018:       SR = HR(I,I-1)
1019:       HR(I,I-1) = 0.0D0
1020:       NORM = DSQRT(HR(I-1,I-1)+HR(I-1,I-1)+HI(I-1,I-1))
1021:       X     +SR*NORM
1022:       XR = HR(I-1,I-1) / NORM
1023:       HR(I-1) = XR
1024:       XI = HI(I-1,I-1) / NORM
1025:       WI(I-1) = XI
1026:       HR(I-1,I-1) = NORM

```

C ****

```

1027:      HI(I-1,I-1) = 0.0D0
1028:      HI(I,I-1) = SR / NORM
1029: C
1030:      DO 490 J = I, N
1031:      YR = HR(I-1,J)
1032:      YI = HI(I-1,J)
1033:      ZZR = HR(I,J)
1034:      ZZI = HI(I,J)
1035:      HR(I-1,J) = XR * YR + XI * YI + HI(I,I-1) * ZZR
1036:      + HI(I-1,J) = XR * YI - XI * YR + HI(I,I-1) * ZZI
1037:      HR(I,J) = XR * ZZR - XI * ZZI - HI(I,I-1) * YR
1038:      HI(I,J) = XR * ZZI + XI * ZZR - HI(I,I-1) * YI
1039:      CONTINUE
1039: 490
1040: C
1041: 500 CONTINUE
1042: C
1043:      SI = HI(EN,EN)
1044:      IF ('SI' .EQ. 0.0D0) GO TO 540
1045:      NORM = CDABS(DCMPLX(HR(EN,EN), SI))
1046:      SR = HR(EN,EN) / NORM
1047:      SI = SI / NORM
1048:      HR(EN,EN) = NORM
1049:      HI(EN,EN) = 0.0D0
1050:      IF ('EN' .EQ. N) GO TO 540
1051:      IP1 = EN + 1
1052: C
1053:      DO 520 J = IP1, N
1054:      YR = HR(EN,J)
1055:      VI = HI(EN,J)
1056:      HR(EN,J) = SR * YR + SI * VI
1057:      HI(EN,J) = SR * VI - SI * YR
1058:      CONTINUE
1059: C
1060: 520      ::::: Inverse operation (columns) :::::::
1061:      540 DO 600 J = LP1, EN
1062:      XR = WR(J-1)
1063:      XI = WI(J-1)
1063: C
1064:      DO 580 I = 1, J
1065:      YR = HR(I,J-1)
1066:      YI = 0.0D0
1067:      ZZR = HR(I,J)
1068:      ZZI = HI(I,J)
1069:      IF (I .EQ. J) GO TO 560
1070:      YI = HI(I,J-1)
1071:      HI(I,J-1) = XR * YI + XI * YR + HI(J,J-1) * ZZI
1072:      HR(I,J-1) = XR * YR - XI * YI + HI(J,J-1) * ZZR
1073:      HR(I,J) = XR * ZZI + XI * ZZI - HI(J,J-1) * YR
1074:      HI(I,J) = XR * ZZI - XI * ZZR - HI(J,J-1) * YI
1075:      CONTINUE
1076: C
1077:      DO 590 I = LOW, IGH
1078:      YR = ZR(I,J-1)
1079:      YI = ZI(I,J-1)
1080:      ZZR = ZR(I,J)
1081:      ZZI = ZI(I,J)
1082:      ZR(I,J-1) = XR * YR - XI * YI + HI(J,J-1) * ZZR
1083:      ZI(I,J-1) = XR * YI + XI * YR + HI(J,J-1) * ZZI

```

```

C ****
1084      ZR(I,J) = XR * ZZR + XI * ZZI - HI(J,J-1) * YR
1085      ZI(I,J) = XR * ZZI - XI * ZZR - HI(J,J-1) * YI
1086      CONTINUE
1087      C   500 CONTINUE
1088      C
1089      C   1090      IF (SI .EQ. 0.0D0) GO TO 240
1091      C   1092      DO 630 I = 1, EN
1093          YR = HR(I,EN)
1094          YI = HI(I,EN)
1095          HR(I,EN) = SR * YR - SI * YI
1096          HI(I,EN) = SR * YI + SI * YR
1097      630 CONTINUE
1098      C
1099      DO 640 I = LOW, IGH
1100          YR = ZR(I,EN)
1101          YI = ZI(I,EN)
1102          ZR(I,EN) = SR * YR - SI * YI
1103          ZI(I,EN) = SR * YI + SI * YR
1104      640 CONTINUE
1105      C
1106      GO TO 240
1107      C   ::::::: A root found :::::::
1108      660 HR(EN,EN) = HR(EN,EN) + TR
1109      HR(EN) = HR(EN,EN)
1110      HI(EN,EN) = HI(EN,EN) + TI
1111      WI(EN) = HI(EN,EN)
1112      EN = ENM1
1113      GO TO 220
1114      C   ::::::: All roots found. Backsubstitute to find
1115      C   ::::::: vectors of upper triangular form :::::::
1116      680 NORM = 0.0D0
1117      C
1118      DO 720 I = 1, N
1119      C
1120      DO 720 J = 1, N
1121      1121      NORM = NORM + DABS(HR(I,J)) + DABS(HI(I,J))
1122      720 CONTINUE
1123      C
1124      IF (N .EQ. 1 .OR. NORM .EQ. 0.0D0) GO TO 1001
1125      C   ::::::: FOR EN=N STEP -1 UNTIL 2 DO ---
1126      DO 800 NN = 2, N
1127          EN = N + 2 - NN
1128          XR = WR(EN)
1129          XI = WI(EN)
1130          ENM1 = EN - 1
1131          ::::::: FOR I=EN-1 STEP -1 UNTIL 1 DO ---
1132          DO 780 II = 1, ENM1
1133              I = EN - II
1134              ZZR = HR(I,EN)
1135              ZZI = HI(I,EN)
1136              IF (I .EQ. ENM1) GO TO 760
1137              IP1 = I + 1
1138      C
1139      DO 740 J = IP1, ENM1
1140          ZZR = ZZR + HR(I,J) * HR(J,EN) - HI(I,J) * HI(J,EN)

```

C ****

```

1141:    Z2I = Z2I + HR(I,J) * HI(I,EN) + HI(I,J) * HR(J,EN)
1142:    740      CONTINUE
1143: C
1144:    760      YR = XR - WR(I)
1144:      YI = XI - WI(I)
1145:      IF (YR .EQ. 0.0D0 .AND. YI .EQ. 0.0D0) YR = MACHEP * NORM
1146:      Z3 = DCMPLX(ZZR,Z2I)
1147:      DO 840 I = 1, ENM1
1148:        IF (I .GE. LOW .AND. I .LE. IGH) GO TO 840
1149:        HR(I,EN) = DREAL(Z3)
1150:        HI(I,EN) = DIMAG(Z3)
1151:      CONTINUE
1152: 800      CONTINUE : End backsubstitution ::::::::::::
1153: C
1154:      ENM1 = N - 1
1155:      ::::: Vectors of isolated roots ::::::::::::
1156:      DO 840 I = 1, ENM1
1157:        IF (I .GE. LOW .AND. I .LE. IGH) GO TO 840
1158:        IP1 = I + 1
1159: C
1160:      DO 820 J = IP1, N
1161:        ZR(I,J) = HR(I,J)
1162:        ZI(I,J) = HI(I,J)
1163: 820      CONTINUE
1164: C
1165: 840      CONTINUE
1166: C      ::::::: Multiply by transformation matrix to give
1167: C      ::::::: vectors of original full matrix.
1168: C      FOR J=N STEP -1 UNTIL LOW+1 DO -- ::::::::::::
1169:      DO 880 JJ = LOW, ENM1
1170:        J = N + LOW - JJ
1171:        H = MIN(0,J-1, IGH)
1172: C
1173:      DO 880 I = LOW, IGH
1174:        ZR = ZR(I,J)
1175:        Z2I = Z2I(I,J)
1176: C
1177:      DO 860 K = LOW, N
1178:        ZZR = ZZR + ZR(I,K) * HR(K,J) - ZI(I,K) * HI(K,J)
1179:        Z2I = Z2I + ZR(I,K) * HI(K,J) + ZI(I,K) * HR(K,J)
1180: 860      CONTINUE
1181: C
1182:      ZR(I,J) = ZZR
1183:      ZI(I,J) = Z2I
1184: 880      CONTINUE
1185: C
1186: 880      GO TO 1001
1187: C      ::::::: Set error — no convergence to an
1188: C      eigenvalue after 30 iterations ::::::::::::
1189: 1000 IERR = EN
1190: 1001 RETURN
1191: C      ::::::: last card of COMGR2 ::::::::::::
1192: END
1193: C*****SUBROUTINE COMGR2(NM, N, LOW, IGH, AR, AI, ORTR, ORTI)
1194: C
1195: C      INTEGER I, J, M, N, II, JJ, LA, MP, NM, IGH, KPI, LOW
1196:      REAL*8 AR(NM,N), AI(NM,N), ORTR(IGH), ORTI(IGH)
1197:

```

C ****

```
1198: REAL*8 F, G, H, FI, FR, SCALE
1199: REAL*8 DSGRT, CDABS, DABS
1200: COMPLEX*16 DCMPXLX
```

```
1201: C
1202: C This subroutine is a translation of a complex analogue of
1203: C the ALGOL procedure ORTHES, Num. Math. 12, 349-368(1968)
1204: C by Martin and Wilkinson. Vol. II-Linear Algebra, 339-358(1971).
1205: C Handbook for Auto. Comp.
```

```
1206: C
1207: C Given a complex general matrix, this subroutine
1208: C reduces a submatrix situated in rows and columns
1209: C L0W through IGH to upper Hessenberg form by
1210: C unitary similarity transformations.
```

```
1211: C
1212: C On input:
```

```
1213: C
1214: C NM must be set to the row dimension of two-dimensional
1215: C array parameters as declared in the calling program
1216: C DIMENSION statement;
```

```
1217: C
1218: C N is the order of the matrix;
```

```
1219: C
1220: C L0W and IGH are integers determined by the balancing
1221: C subroutine CBAL. If CBAL has not been used,
1222: C set L0W=N, IGH=N;
```

```
1223: C
1224: C AR and AI contain the real and imaginary parts,
1225: C respectively, of the complex input matrix.
```

```
1226: C
1227: C On output:
```

```
1228: C
1229: C AR and AI contain the real and imaginary parts,
1230: C respectively, of the Hessenberg matrix. Information
1231: C about the unitary transformations used in the reduction
1232: C is stored in the remaining triangles under the
1233: C Hessenberg matrix;
```

```
1234: C
1235: C ORTR and ORTI contain further information about the
1236: C transformations. Only elements L0W through IGH are used.
```

```
1237: C
1238: C Questions and comments should be directed to B. S. Garbow,
1239: C Applied Mathematics Division, Argonne National Laboratory
1240: C
```

```
1241: C
1242: C
1243: C LA = IGH - 1
1244: KP1 = L0W + 1
1245: IF (LA .LT. KP1) GO TO 200
```

```
1246: C
1247: DO 180 M = KP1, LA
1248: H = 0.0D0
1249: ORTR(M) = 0.0D0
1250: ORTI(M) = 0.0D0
1251: SCALE = 0.0D0
1252: C : Scale column (ALGOL TOL then not needed) ::::::::::::
1253: DD 90 I = M, IGH
1254: SCALE = SCALE + DABS(AR(I, M-1)) + DABS(AI(I, M-1))
90
```

C ****

```

1255: C
1256:      IF (SCALE .EQ. 0.0D0) GO TO 180
1257:      MP = M + IGH
1258:      DO 100 II = M, IGH
1259:      DO 100 II = M, IGH
1260:      I = MP - 11
1261:      ORTR(I) = AR(I,M-1) / SCALE
1262:      ORTI(I) = AI(I,M-1) / SCALE
1263:      H = H + ORTR(I) * ORTI(I) + ORTI(I) * ORTR(I)
1264:      CONTINUE
1265: C
1266:      G = DSGRT(H)
1267:      F = CDAB5(DCMPLX(ORTR(M),ORTI(M)))
1268:      IF (F .EQ. 0.0D0) GO TO 103
1269:      H = H + F * G
1270:      G = G / F
1271:      ORTR(M) = (1.0D0 + G) * ORTR(M)
1272:      ORTI(M) = (1.0D0 + G) * ORTI(M)
1273:      GO TO 105
1274: C
1275:      103
1276:      ORTR(M) = G
1277:      AR(M,M-1) = SCALE
1278:      DO 105 Form (1-(U*UT)/H) * A ::::::::::::
1279:      105 DO 130 J = M, N
1280:      FR = 0.0D0
1281:      FI = 0.0D0
1282:      DO 110 II = M, IGH
1283:      I = MP - II
1284:      FR = FR + ORTR(I) * AR(I,J) + ORTI(I) * AI(I,J)
1285:      FI = FI + ORTR(I) * AI(I,J) - ORTI(I) * AR(I,J)
1286:      CONTINUE
1287: C
1288:      FR = FR / H
1289:      FI = FI / H
1290: C
1291:      DO 120 I = M, IGH
1292:      AR(I,J) = AR(I,J) - FR * ORTR(I) + FI * ORTI(I)
1293:      AI(I,J) = AI(I,J) - FR * ORTI(I) - FI * ORTR(I)
1294:      CONTINUE
1295: C
1296:      130
1297:      CONTINUE
1298:      DO 160 I = 1, IGH
1299:      FR = 0.0D0
1300:      FI = 0.0D0
1301:      DO 130 J=IGH STEP -1 UNTIL M DO ::::::::::::
1302:      DO 140 JU = M, IGH
1303:      J = MP - JU
1304:      FR = FR + ORTR(J) * AR(I,J) - ORTI(J) * AI(I,J)
1305:      FI = FI + ORTR(J) * AI(I,J) + ORTI(J) * AR(I,J)
1306:      CONTINUE
1307: C
1308:      FR = FR / H
1309:      FI = FI / H
1310: C
1311:      DO 150 J = M, IGH

```

C *****

```
1312:      AR(I,J) = AR(I,J) - FR * ORTR(J) - FI * ORTI(J)
1313:      AI(I,J) = AI(I,J) + FR * ORTI(J) - FI * ORTR(J)
1314:      CONTINUE
1315:      C
1316:      160  CONTINUE
1317:      C
1318:      ORTR(M) = SCALE * ORTR(M)
1319:      ORTI(M) = SCALE * ORTI(M)
1320:      AR(M,M-1) = -G * AR(M,M-1)
1321:      AI(M,M-1) = -G * AI(M,M-1)
1322:      160  CONTINUE
1323:      C
1324:      200  RETURN
1325:      C      ::::::: last card of CORTH
1326:      END
1327:      SUBROUTINE HANKEL(Z,ZHO,ZH1)
1328:      C
1329:      IMPLICIT COMPLEX*16(Z), REAL*8(A-H,D-Y)
1330:      C
1331:      PAI=3.14159265D0
1332:      ZAI=Cmplx(0,0,1,00)
1333:      ZZ=ZAI*Z
1334:      CALL_BSK((ZZ,ZKO,ZK1,1)
1335:      ZC1=Z*ZAI/PAI
1336:      ZHO=ZC1*ZKO
1337:      ZH1=-Z*ZK1/PAI
1338:      RETURN
1339:      END
1340:      C
1341:      SUBROUTINE BSK(AA,SK0,SK1,JUMAX)
1342:      C
1343:      IMPLICIT DOUBLE PRECISION (A-H,0-Z)
1344:      COMPLEX*16 AA,SK0,SK1,XX,ARH,AIRH,ARH2,AL,ALE,A2,A,B,C,
1345:      1SUM1,SUM2,SUM3,T
1346:      DIMENSION XX(1),ASN(1)
1347:      PI=3.14159265D0
1348:      E=0.5772156649D0
1349:      ASN(1)=CDABS(AA)
1350:      AAA=AA
1351:      SITA=DATAN2(DIMAG(AA),AAA)
1352:      XX(1)=AA
1353:      DO 200 JJ=1,JUMAX
1354:      ARH=XX(JJ)
1355:      DIN=ASN(JJ)*ZD0
1356:      IF(ASN(JJ)-3,DO 10,10,20
1357:      10 ARH2=ARH/2,DO
1358:      AL=DCMLPLX(BLDG(ASN(JJ)/2,DO),SITA)
1359:      ALE=EA_L
1360:      AIRH=1,DO/ARH
1361:      A2=ARH2*ARH2
1362:      A=AIRH#2,DO
1363:      B=ALE
1364:      SUM1=DCMLPLX(O,DO,O,DO)
1365:      SUM2=DCMLPLX(O,DO,O,DO)
1366:      SUM3=DCMLPLX(O,DO,O,DO)
1367:      DO 100 I=1,8
1368:      R1=1,DO/(FLDAT(I)*I,DO
```

```

1369: R12=RI*RI
1370: T=A*A2*R12
1371: A=T
1372: D=B+RI
1373: C=A*B
1374: SUM1=SUM1+A
1375: SUM2=SUM2+FLOAT(I)*C
1376: SUM3=SUM3-C
1377: 100 CONTINUE
1378: SK0=-ALE+SUM3*ARH2
1379: SK1=A*INRH+SUM1/2, DO-SUM2
1380: GO TO 11
1381: 20 DB=B, *ASN(J,J)
1382: DAO=1, DO
1383: DA1=1, DO
1384: SUM1=DCMPLX(1, DO, 0, DO)
1385: SUM2=DCMPLX(1, DO, 0, DO)
1386: JMAX=7
1387: DO 110 J=1, JMAX
1388: NODD=(2*(J-1))**2
1389: BO=FLOAT(NODD)*1, DO
1390: BI=FLOAT(4-NODD)
1391: S=SITA*FI, DAT(J)
1392: SS=DSIN(S)
1393: CS=DCOS(S)
1394: DDA=FLOAT(J)*1, DO*DB
1395: DAO=DAO*BO/DDA
1396: T=DCMPLX(DAO*CS, DAO*SS)
1397: SUM1=SUM1+T
1398: DA1=DA1*BI/DDA
1399: T=DCMPLX(DA1*CS, DA1*SS)
1400: SUM2=SUM2+T
1401: 110 CONTINUE
1402: T=CDSGRT(P1/(2, DO**AA))/CDEXP( AA)
1403: SK0=SUM1*T
1404: SK1=SUM2*T
1405: 11 CONTINUE
1406: 200 RETURN
1407: END
1408:

```

APPENDIX B

Computer Program "AXS"

A program to compute dynamic response of submerged soil bed
for three-dimensional conditions in cylindrical coordinates

PROGRAM *****AXS***** CODED BY KAZAMA 1990 5/30

```
1: C
2: C PROGRAM *****AXS***** CODED BY KAZAMA 1990 5/30
3: C MAIN PROGRAM FOR 2-PHASE DYNAMIC PROBLEM CYLINDRICAL COORDINATE
4: C DRAIN CASE
5: C
6: C MAXIMUM NUMBER OF LAYER IS 10
7: C
8: IMPLICIT COMPLEX*16(Z)
9: PARAMETER (MLY=10)
10: PARAMETER (M6=MLY*6)
11: PARAMETER (M4=(MLY+1)*4)
12: PARAMETER (M2=(MLY+1)*2)
13: C
14: COMMON /NOCH1/ ZA(M4,M4)
15: COMMON /NOCH2/ ZK(M4,M4),ZCM(M4,M4)
16: COMMON /NOCH3/ ZPHI(M6,MLY*4),ZPINV(MLY*4,MLY*4)
17: COMMON /NOCH4/ ZB(M4,M4),ZRB(M4,M4)
18: COMMON /NOCH5/ ZCINV(MLY*4,MLY*4),ZCTINV(MLY*2,MLY*2),
+ COMMON /NOCH6/ ZC2DMC5(MLY*3,MLY)
19: COMMON /NOCH7/ ZAT(M2,M2),ZKT(M2,M2),ZCHT(M2,M2)
20: C
21: C DIMENSION INDI(MLY,6),FRG(100),NXOUT(10),NFORC(MLY*4)
22: C
23: C COMMON /GROUND/ ZAA(10),ZG(10),ZAL(10),H(10),PERM(10),
24: + ROP(10),CNE(10),RD(10),RF(10),E1(10),DEP(11),DEM(10)
25: C
26: C
27: COMMON /DMY1/ ZDUMY1(3200)
28: COMMON /DMY2/ ZDUMY2(3200)
29: COMMON /DMY3/ ZDUMY3(3200)
30: COMMON /DMY4/ ZDUMY4(3200)
31: COMMON /DMY5/ ZDUMY5(2400)
32: COMMON /DMY6/ ZDUMY6(2400)
33: C
34: COMMON /HSFNC/ ZHS(MLY*4),ZHO(MLY*4),ZHI(MLY*4)
35: COMMON /COND1/ KC,KP,KG(4),NLAY,MMN
36: COMMON /JYD/ MJYD,MJRZ,MJTH,MRL,MRRZ,MRTTH,IND2(6)
37: COMMON /CONST/ GAL,PAI,DEPTH,NOT,VSB,NFFO,RR,DOO,NCASE
38: COMMON /RESP/ ZUR1,ZU21,ZUT1,ZNR1,ZWZ1,ZWT1,ZKXX,ZKXT,ZKTT
39: C
40: CHARACTER INFIL*24,OUTFL*22,YN*1
41: CHARACTER 'PLOTD1*21,PLOTD2*21,PLOTD3*21,PLOTD4*21,
42: C
43: DATA PLOTD1/'MOTOK1>OUT>PLOUT,FRG'/
44: DATA PLOTD2/'MOTOK1>OUT>PLOUT,MODE'/
45: DATA PLOTD3/'MOTOK1>OUT>PLOUT,DISP'/
46: DATA PLOTD4/'MOTOK1>OUT>PLOUT,SGMA'/
47: DATA OUTFL/'MOTOK1>OUT>PLOUT,OUT'/
48: KG(1)=11
49: KG(2)=2
50: KG(3)=13
51: KG(4)=14
52: C
53: C MJD=MJRZ+MJTH,.....TOTAL NUMBER OF DEGREE OF FREEDOM(RZ)
54: C MRZ=IND2(1234).....NUMBER OF DEGREE OF FREEDOM(RZ)
55: C MJTH=IND2(56).....NUMBER OF DEGREE OF FREEDOM(TH)
56: C MR1 =MNRZ+MRTH,.....TOTAL NUMBER OF EIGEN VALUE
57: C MRRZ=IND(123).....NUMBER OF EIGEN VALUE (RZ)
```

58: C MRTH=IND(5) NUMBER OF EIGEN VALUE (TH)
59: C
60: C
61: C DIMENSION /DMY1/ . . . 32MLY**2
62: C /DMY2/ . . . 32MLY**2
63: C /DMY3/ . . . 32MLY**2
64: C /DMY4/ . . . 32MLY**2
65: C /DMY5/ . . . 24MLY**2
66: C /DMY6/ . . . 24MLY**2
67: C
68: C INFORMATION ABOUT VARIABLES
69: C
70: C PARAMETER
71: C MLY MAXIMUM USAGE LAYER NUMBER
72: C DIMENSION
73: C ----- AD ARRAY -----
74: C EI (10) BENDING STIFFNESS OF YAITA (TFNM**2)
75: C ZAA (10) A VALUE CALCULATED BY LAME CONSTANTS AND ALPHA, Q
76: C ZG (10) SHEAR STIFFNESS OF THE GROUND
77: C ZAL (10) ALPHA VALUE OF BIOT EQUATION
78: C ZQ (10) Q VALUE OF BIOT EQUATION
79: C H (10) LAYER HEIGHT (M)
80: C PERM(10) PERMEABILITY COEFFICIENT OF THE GROUND
81: C DIVIDED BY RUF*GAL
82: C CNE (10) POLOSTY
83: C RO (10) UNIT WEIGHT OF THE SOLID AND FLUID MIXTURE
84: C RWF (10) UNIT WEIGHT OF THE FLUID MATERIAL
85: C
86: C INPUT AND OUTPUT DEVICE CODE SETTING
87: C (KC) (KP)
88: C KC=2
89: C
90: C KP=10
91: C WRITE(*,*)
92: C WRITE(*,*) ' INPUT DATA FILENAME=?'
93: C READ(*, '(A24)'), INFL
94: C OPEN(KC, FILE=INFL)
95: C OPEN(KP, FILE=OUTFL)
96: C OPEN(KG(1), FILE=PLOTD1)
97: C OPEN(KG(2), FILE=PLOTD2)
98: C OPEN(KG(3), FILE=PLOTD3)
99: C OPEN(KG(4), FILE=PLOTD4)
100: C &&&&&&&&& SET CONSTANT FOR NORMALIZE &&&&&&&&
101: C
102: C
103: C VSB : AVERAGE SHEAR WAVE VELOCITY FOR DIMENSIONLESS FRG.
104: C VSB=200.0
105: C D00 : DEPTH FOR DIMENSIONLESS FRG. AND WAVE NUMBER
106: C D00=10.0
107: C &&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
108: C
109: C CALL CARDIM(KC, NP)
110: C READ(KC, 500) YN, NCASE
111: C IF (NCASE, LE, 0) NCASE=1
112: C DO 1000 NSET=1, NCASE
113: C
114: C

```

115: CALL INPUT(IND1,FRQ,NXOUT,NFLG1,MLY,NFORC)
116: C----- READ DATA AND SETTING MATERIAL CONSTANT -----
117: C
118: CALL MATRIX(IND1,NFLG1)
119: C----- MAKE MATRIX A, D, B, K, CM, RB -----
120: C
121: CALL RDMAT(IND1)
122: C----- REDUCE DEGREE OF FREEDOM AND REARRANGE MATRIX -----
123: C
124: DO 2 N=1, NFRC
125:   WRITE(*,600) N, FRQ(N)
126:   DMG=2.0*PAI*FRQ(N)
127:   DMG2=DMG*DMG
128:   WRITE(*, '(/2X, ''CASE= '', I3, '' FREQUENCY= ', , F8. 2)' ) N, FRQ(N)
129:   CALL CMATRX(ZCM, ZKT, DMG, DMG2)
130: C----- MAKE MATRIX C AND INVERSE OF C -----
131: C
132: CALL EIGEN(ZA, ZAT, ZB)
133: C----- SOLVE EIGEN VALUE PROBLEM -----
134: C
135: XA=DMG*D00/VSB
136: IF(YN.EQ.'Y') THEN
137:   DO 20 J=1, MRI
138:     WRITE(KO(1), 650) XA, D00*ZHS(J)
139: 20 CONTINUE
140: CALL WAVEOT
141: ELSE
142:   CALL HNLST(MRI, RR)
143: C
144: CALL MATRIX2(RR, IND1)
145: C
146: CALL PINVS(O, RR)
147: C
148: CALL RMATRIX(IRR, ZPHI, ZPINV)
149: C----- MAKE STIFFNESS MATRIX OF THE GROUND -----
150: C
151: CALL SOLVE(ZRB, NXOUT, NFLG1, IND1, NFORC, DMG2)
152: C----- SOLVE PROBLEM AND RESULTS OUTPUT -----
153: C
154: IF(N.NE. 1. OR. NSET. NE. 1) GO TO 100
155: ZU0=ZUR1
156: ZSTKU=1.0/ZU0
157: WRITE(*, *) ZSTKU
158: 100 CONTINUE
159: Z1=ZUR1/ZU0
160: Z2=ZUZ1/ZU0
161: Z3=ZUT1/ZU0
162: Z4=ZWZ1/ZU0
163: Z5=ZWT1/ZU0
164: ZDYKU=1.0/ZUR1
165: ZCOEF=ZDYKU/REAL(ZSTKU)
166: C
167: A1=AIMAG(Z1)
168: A2=REAL(Z1)
169: APR=57.29578*ATAN2(A1, A2)
170: A1=A1MAQ(Z2)
171: A2=REAL(Z2)

```

```

172: APZ=57. 2957B*ATAN2(A1,A2)
173: A1=A1*AG(73)
174: A2=REAL(73)
175: APT=57. 2957B*ATAN2(A1,A2)
176: C
177: IF(IND2(4).EQ.0) GO TO 110
178: A1=A1*AG(74)
179: A2=REAL(74)
180: APWZ=57. 2957B*ATAN2(A1,A2)
181: A1=A1*AG(75)
182: A2=REAL(75)
183: APWT=57. 2957B*ATAN2(A1,A2)
184: C
185: 110 WRITE(KG(1),650) XA,ABS(Z1),APR,ABS(Z2),APZ,ABS(Z3),APT
186: +
187: C
188: Z1=ZKX*(RR**3)/EI(1)
189: Z2=ZKXT*(RR*RR)/EI(1)
190: Z3=ZKTT*(RR)/EI(1)
191: WRITE(KG(4),650) XA,Z1,Z2,Z3
192: END IF
193: 2 CONTINUE
194: 1000 CONTINUE
195: CLOSE(KC, STATUS='KEEP')
196: C
197: CLOSE(KC1), STATUS='KEEP')
198: CLOSE(KC2), STATUS='KEEP')
199: CLOSE(KC3), STATUS='KEEP')
200: CLOSE(KC4), STATUS='KEEP')
201: CLOSE(KP, STATUS='KEEP')
202: WRITE(*,'(/'OUTPUT LIST FILE='',A20)') OUTFL
203: WRITE(*,'(/'GRAPHIC OUT FILE='',A420)',PLOTD1,PLOTD2,PLOTD3
204: +,PLOTD4
205: WRITE(*,'(/'INPUT DATA FILE='',A20)',INFL
206: 300 FORMAT(IH1,'N= ',I5,' FREQUENCY= ',F10.2,'Hz')
207: 600 FORMAT(IH1,'N= ',I5,' ')
208: 650 FORMAT(F10.5,12E15.5)
209: STOP
210: END
211: C ****SUBROUTINE INPUT IND1,FRQ,NXOUT,NFLQ1,MLY,NFORC*****
212: ****
213: C ****IMPLICIT COMPLEX*16(Z)
214: COMMON /GROUND/ ZAA(10),ZB(10),ZAL(10),ZQ(10),H(10),PERM(10),
215: + ROP(10),CNE(10),RQ(10),ROP(10),EI(10),DEP(11),DEM(10)
216: + COMMON /CONST/ QAL,PA,DEPTH,NOT,VSB,NFRQ,RR,DOO,NCASE
217: COMMON /COND1/ KC,KP,KC4),NLAY,MNN
218: COMMON /JYD/ MJYD,MJRH,MJTH,MR2,MRT2,IND2(6),
219: COMMON /DMY1/VSM(10),DAMPM(10),PSNRM(10),PERMM(10),CNEM(10),
220: + ROPM(10),RDM(10),RDFM(10),CKFM(10),CKSM(10),EIM(10)
221: C
222: C CHARACTER AM*5,TITLE*80
223: DIMENSION FRQ(*),IND1(MLY,*),AM(4),NXOUT(*),NFORC(*)
224: C
225: C DATA DRDF/1.0/,DCMF/2.08E+5/,DCKS/3.7E+6/
226: DATA DRR/0.5/,DRDP/2.32/
227: DATA AM(1)/SOLID/,AM(2)/FLUID/,AM(3)/MIX. /
228: C

```

```

229: DATA AM(4)/ UNDRN /
230: DATA DE1/20000.0 /
231: C
232: C
233: C
234: C
235: C
236: C
237: C
238: C
239: C
240: C
241: C
242: C
243: C
244: C
245: C
246: C
247: C
248: C
249: C
250: C
251: C
252: C
253: C
254: C
255: C
256: C
257: C
258: C
259: C
260: C
261: C
262: C
263: C
264: C
265: C
266: C
267: C
268: C
269: C
270: C
271: C
272: C
273: C
274: C
275: C
276: C
277: C
278: C
279: C
280: C
281: C
282: C
283: C
284: C
285: C

```

233: GAL=9. B
PAI=3. 14159265
WRITE(*,*) 'ROUTINE INPUT NOW'
READ(KC, 499) TITLE
READ(KC, 500) NLAY, NFRO, NMAT, (NXOUT(I), I=1, 10), NFLG1,
+ (NFRC(I), I=1, 40), RR
IF (RR .LT. 0.01) RR=0.0
WRITE(KP, 650) (NFRC(I), I=1, NLAY*4)
IF(NLAY.GT.10) GO TO 999

243: SET M=1
NMN=1

244: IF (NMN.EQ.1) WRITE(KP, 655)
IF (NMN.EQ.0) WRITE(KP, 656)

245: NOT=0
NXOUT(1)=1

246: DO 99 I=1, 10
99 NOT=NOT+NXOUT(I)

247: READ(KC, 501) (FRQ(I), I=1, NFRO)
WRITE(KP, 600) NFRO
WRITE(KP, 630) (FRQ(I), I=1, NFRO)

248: NOT=0
NXOUT(1)=1

249: READ(KC, 502) ROM(N), DAMPM(N), PSNRM(N), PERMM(N), CNEM(N),
+ ROPM(N), ROFM(N), CKFM(N), CKSM(N), EIM(N), ROPM(N)

250: IF (ROFM(N).LT.0.1) ROFM(N)=DROP
IF (ROPM(N).LT.0.1) ROPM(N)=DROP
IF (CKFM(N).LT.1.0) CKFM(N)=DCKF
IF (CKSM(N).LT.1.0) CKSM(N)=DCKS
IF (EIM(N).LT.1.0) EIM(N)=DEI
WRITE(KP, 603) N, VSM(N), DAMPM(N), PSNRM(N), PERMM(N), CNEM(N),
+ RUM(N), ROPM(N), ROFM(N), CKFM(N), CKSM(N), EIM(N), ROPM(N)

251: DO 2 N=1, NMAT
252: READ(KC, 502) ROM(N), ROPM(N), CKFM(N), CKSM(N), EIM(N), ROPM(N)

253: IF (ROPM(N).LT.0.1) ROPM(N)=DROP
IF (ROPM(N).LT.0.1) ROPM(N)=DROP
IF (CKSM(N).LT.1.0) CKSM(N)=DCKS
IF (EIM(N).LT.1.0) EIM(N)=DEI
WRITE(KP, 603) N, VSM(N), DAMPM(N), PSNRM(N), PERMM(N), CNEM(N),
+ RUM(N), ROPM(N), ROFM(N), CKFM(N), CKSM(N), EIM(N), ROPM(N)

254: WRITE(KP, 602) (FRQ(I), I=1, NFRO)

255: WRITE(KP, 602)

256: READ(KC, 502) ROM(N), ROPM(N), CKFM(N), CKSM(N), EIM(N), ROPM(N)

257: IF (ROPM(N).LT.0.1) ROPM(N)=DROP
IF (ROPM(N).LT.0.1) ROPM(N)=DROP
IF (CKFM(N).LT.1.0) CKFM(N)=DCKF
IF (CKSM(N).LT.1.0) CKSM(N)=DCKS
IF (EIM(N).LT.1.0) EIM(N)=DEI
WRITE(KP, 603) N, VSM(N), DAMPM(N), PSNRM(N), PERMM(N), CNEM(N),
+ RUM(N), ROPM(N), ROFM(N), CKFM(N), CKSM(N), EIM(N), ROPM(N)

258: 2 CONTINUE

259: WRITE(KP, 605)
260: WRITE(KP, 606)
261: DEPTH=0.0
262: TTF=0.0
263: DO 3 J=1, NLAY
264: DEP(J)=DEPTH
265: READ(KC, 503) NO, MKIND, MAT, H(J)
266: DEM(J)=DEPTH*(J)/2.0

267: SET KEY INDEX FOR DEGREE OF FREEDOM

268: C

269: C

270: C

271: C

272: C

273: C

274: C

275: C

276: C

277: C

278: C

279: C

280: C

281: C

282: C

283: C

284: C

285: C

MKIND = 1 ... JUST SOLID
MKIND = 2 ... JUST FLUID
MKIND = 3 ... MIXTURE DRAIN CONDITION
MKIND = 4 ... MIXTURE UNDRAIN CONDITION
IND1(J,1) ... DISPLACEMENT (R-DIRECTION) OF SOLID
IND1(J,2) ... DISPLACEMENT (Z-DIRECTION) OF SOLID
IND1(J,3) ... RELATIVE DIS. (R-DIRECTION) OF FLUID

```

286: C      IND1(J,4) ... RELATIVE DIS. (Z-DIRECTION) OF FLUID
287: C      IND1(J,5) ... DISPLACEMENT (THETA-DIRC.) OF SOLID
288: C      IND1(J,6) ... RELATIVE DIS. (THETA-DIRC.) OF FLUID
289: C
290: DO 1 N=1,6
291:   IND1(J,N)=1
292:   1 CONTINUE
293:   IF(MKIND.EQ.3) GO TO 10
294:   IND1(J,3)=0
295:   IND1(J,4)=0
296:   IND1(J,6)=0
297: 10 CONTINUE
298:  IF(MKIND.NE.2) GO TO 11
299:  IND1(J,1)=0
300:  IND1(J,2)=0
301:  IND1(J,3)=1
302:  IND1(J,4)=1
303:  IND1(J,5)=0
304:  IND1(J,6)=1
305:  11 CONTINUE
306: C
307: C      SET MATERIAL CONSTANT OF EACH LAYER
308: C
309: IF(NMAT.GT.NPAT) GO TO 901
310: VS=VS/NPAT
311: DAMP=DAMP/NPAT
312: PSNR=PSNR/NPAT
313: PERM(J)=PERMM(NPAT)/ROFM(NPAT)
314: CNE(J)=CNE(NPAT)
315: IF(MKIND.EQ.2) CNE(J)=1.0
316: EI(J)=EI(NPAT)
317: RO(J)=RO(NPAT)/GAL
318: ROF(J)=ROFM(NPAT)/GAL
319: ROP(J)=ROP(NPAT)/GAL
320: CKS=CKSN(NPAT)
321: CKF=CKFN(NPAT)
322: GO TO 902
323: 901 STOP 'SPECIFIED MATERIAL IS NO EXIST'
324: 902 CONTINUE
325: W1=RD(J)*VS*VS
326: W2=1.0-2.0*DAMP*DAMP
327: W3=SOR(1.0-DAMP*DAMP)
328: ZG(J)=W1*CMPLX(W2,2.0*DAMP*W3)
329: IF(MKIND.EQ.2) ZG(J)=CMPLX(0.0,0.0)
330: ZRAMD=ZG(J)*2.0*PSNR/(1.0-2.0*PSNR)
331: ZKD=ZRAND+ZG(J)*2.0/3.0
332: ZAL(J)=1.0-ZKD/CKS
333: C
334: IF(MKIND.EQ.1) ZAL(J)=CMPLX(0.0,0.0)
335: IF(MKIND.EQ.2) ZAL(J)=CMPLX(1.0,0.0)
336: C
337: ZH1=(ZAL(J)-CNE(J))/CKS+1.0/CKF
338: ZO(J)=1.0/ZW1
339: IF(MKIND.EQ.1) ZG(J)=CMPLX(0.0,0.0)
340: ZAA(J)=ZRAND*2.0*ZG(J)+ZAL(J)*ZAL(J)*ZG(J)
341: WRITE(XP,604) J,AM(MKIND),MAT,H(J),DEPTH,ZA(J),ZG(J),ZAL(J),ZH1,ZQ(J)
342: IF(VS.LT.0.01) GO TO 3

```

```

343: DEPTH=DEPTH+H*(J/VS
344: TT=TT+H*(J/VS
345: 3 CONSTITUTE
346: DEP(NLAY+1)=DEPTH
347: VSB=DEPTH/TT
348: C
349: C FORMAT
350: C
351: 499 FORMAT(A80)
352: 500 FORMAT(3I5,5X,10I1,15,5X,4O11,F10.2)
353: 501 FORMAT(10FB.0)
354: 502 FORMAT(11FB.0)
355: 503 FORMAT(3I5,F10.0)
356: 600 FORMAT(1H//,'NUMBER OF THE CALCULATION FREQUENCY =',I5,/)
357: 630 FORMAT(1H,'AT FREQUENCY =',I5,/,10.5,1)
358: 601 FORMAT(1H,'THERE IS NO FILE IN THIS CALCULATION')
359: 602 FORMAT(1H//,'MATERIAL',2X,'VS',10X,'DAMPING',5X,
360: + 'POISSON-R',3X,'PERMEABILITY',1X,'POLODITY',3X,
361: + 'UNIT WEIGHT',1X,'UNIT WEIGHT',2X,'KF',9X,'KS',
362: + '9X,'EI OF PILE',3X,'NO.',4X,'(m/sec)',29X,'(m/sec)',
363: + '17X,'OF MIXTURE',2X,'OF FLUID')
364: 603 FORMAT(1H,'14,2X,10I1,F12.3),OF5.2)
365: 605 FORMAT(1H//,'LAYER',2X,'MATERIAL',2X,'THICKNESS',
366: + 2X,'DEPTH',7X,'A-VALUE',18X,'G-VALUE',18X,
367: + 'Q-VALUE')
368: 604 FORMAT(1H,'12,2X,A5,13,2F9.2,B1PE12.4)
369: 606 FORMAT(1H,'3X,'TYPE',18X,'(m)',6X,'(m)',3X,
370: + 4,'REAL',IMAG.)
371: 650 FORMAT(1H,'***** FORCE VECTOR= *****,40I1)
372: 655 FORMAT(1H,'CYLINDRICAL M=1 CASE')
373: 656 FORMAT(1H,'AXISYMMETRIC M=0 CASE')
374: C
375: WRITE(*,*),
376: RETURN
377: 999 STOP 'CALCULATION CONDITION NOT ALLOWED'
378: END
379: C *****
380: SUBROUTINE MATRIX(IND1,NFLQ1)
381: C *****
382: IMPLICIT COMPLEX*16(Z)
383: PARAMETER (MLY=10)
384: PARAMETER (M4=MLY+1)*4)
385: COMMON /GROUND/ ZAA(110),ZB(110),ZAL(10),ZQ(10),H(10),PERM(10),
386: + ROP(110),CNE(110),RO(110),RQF(110),E1(110),DEM(110),
387: + COMMON /COND1/ KC,KP,KC4),NLAY,MNN
388: COMMON /DMY1/ ZWK1(4,4),ZWK(4,4),ZWKBS(8,8),
389: + ZWK2(2,2),ZWK4(4,4),ZD(M4,M4),
390: COMMON /NOCH1/ ZA(M4,M4),
391: COMMON /NOCH2/ ZK(M4,M4),ZCM(M4,M4)
392: COMMON /NOCH4/ ZB(M4,M4),ZRB(M4,M4)
393: COMMON /NOCH6/ ZAT(M2,M2),ZKT(M2,M2),ZCMT(M2,M2)
394: DIMENSION IND1(MLY,*)
395: C
396: C ZERO=CNPLX(0,0,0,0)
397: C WRITE(*,*),'SUBROUTINE MATRIX NOW'
398: C
399: C

```

```

400: C ----- INITIAL ZERO CLEAR
401: C
402: C
403: C     CALL ZEROCL(ZWK, 4, 4)
404: C     CALL ZEROCL(ZWK2, 2, 2)
405: C     CALL ZEROCL(ZWK4, 4, 4)
406: C     CALL ZEROCL(ZA, M4, M4)
407: C     CALL ZEROCL(ZB, M4, M4)
408: C     CALL ZEROCL(ZD, M4, M4)
409: C     CALL ZEROCL(ZCM, M4, M4)
410: C     CALL ZEROCL(ZCR, M4, M4)
411: C     CALL ZEROCL(ZRB, M4, M4)
412: C     CALL ZEROCL(ZAT, M2, M2)
413: C     CALL ZEROCL(ZKT, M2, M2)
414: C     CALL ZEROCL(ZCMT, M2, M2)
415: C     JQ=0
416: DO 2 J=1, NLAY
417: ZAQ=ZAA(J)-2.0*ZG(J)
418: ZAG=ZAL(J)*ZG(J)
419: KO=J*4-3
420: KB=J*2-1
421: RNE=RDF(J)/RNE(J)
422: CFAC=2.0*EI(J)/H(J)/H(J)/H(J)
423: CAR=ROP(J)*H(J)/420.0
424: C ----- MATRIX A ELEMENT (UPPER LEFT SIDE 1/4)
425: C
426: C
427: ZWK(1,1)=2.0*ZAA(J)
428: ZWK(1,3)=2.0*ZAG
429: ZWK(2,2)=2.0*ZG(J)
430: ZWK(3,1)=ZWK(1,3)
431: ZWK(3,3)=2.0*ZG(J)
432: CALL COPY4(ZWK, ZWKB, 4, 0, 5, 1, 0, H(J)/6.0)
433: CALL STRE(ZWKB, B, ZA, KO, M4)
434: C
435: ZWK(1,1)=2.0*ZG(J)
436: CALL COPY4(ZWK2, ZWKA, 2, 0, 5, 1, 0, H(J)/6.0)
437: CALL STRE(ZWKA, A, ZAT, KB, M2)
438: C ----- MATRIX D ELEMENT (UPPER LEFT SIDE 1/4) AND DD ELEMENT
439: C
440: C
441: ZWK(1,2)=2*ZAG
442: ZWK(1,4)=ZAG
443: ZWK(2,1)=-ZG(J)
444: ZWK(3,2)=ZAG
445: ZWK(3,4)=ZG(J)
446: CALL COPY4(ZWK, ZWKB, 4, 1, 0, -1, 0, 0, 5)
447: CALL STRE(ZWKB, B, ZD, KO, M4)
448: C ----- MATRIX K ELEMENT (UPPER LEFT SIDE 1/4)
449: C
450: C
451: ZWK(1,1)=ZG(J)
452: ZWK(2,2)=ZAG
453: ZWK(2,4)=ZAG
454: ZWK(4,2)=ZAG
455: ZWK(4,4)=ZG(J)
456: CALL COPY4(ZWK, ZWKB, 4, -1, 0, 1, 0, 1, 0/H(J))

```

```

457: CALL STRE(ZWK8, B, ZK, KO, M4)
458: C
459: ZWK2(1, 1)=ZG(J)
460: CALL COPY(ZWK2, ZWK4, 2, -1, 0, -1, 0, 1, 0, 1, 0/H(J))
461: CALL STRE(ZWK4, 4, ZKT, KB, M2)
462: C
463: C MATRIX CM ELEMENT (UPPER LEFT SIDE 1/4)
464: C
465: ZWK(1, 1)=CMPLX(2, 0*R0(J), 0, 0)
466: ZWK(1, 3)=CMPLX(2, 0*R0F(J), 0, 0)
467: ZWK(2, 2)=CMPLX(2, 0*R0(J), 0, 0)
468: ZWK(2, 4)=CMPLX(2, 0*R0F(J), 0, 0)
469: ZWK(3, 1)=ZWK(1, 3)
470: PP=0, 0
471: IF(IND1(J, 1).EQ.1 .AND. IND1(J, 3).EQ.1) PP=2, 0/PERM(J)
472: ZWK(3, 3)=CMPLX(2, 0*RNE, PP)
473: IF(CNE(J).GT.0.999) ZWK(3, 3)=CMPLX(2, 0*RNE, 0, 0)
474: ZWK(4, 2)=ZWK(2, 4)
475: ZWK(4, 4)=ZWK(3, 3)
476: CALL COPY4(ZWK, ZWK8, 4, 0, 5, 0, 5, 1, 0, H(J)/6, 0)
477: CALL STRE(ZWK8, B, ZCM, KO, M4)
478: C
479: ZWK2(1, 1)=CMPLX(2, 0*R0(J), 0, 0)
480: ZWK2(1, 2)=CMPLX(2, 0*R0F(J), 0, 0)
481: ZWK2(2, 1)=ZWK2(1, 2)
482: ZWK2(2, 2)=CMPLX(2, 0*RNE, PP)
483: CALL COPY4(ZWK2, ZWK4, 2, 0, 5, 0, 5, 1, 0, H(J)/6, 0)
484: CALL STRE(ZWK4, 4, ZMT, KB, M2)
485: C
486: IF(INFLQ1.EQ.0) GO TO 2
487: C
488: C MATRIX RB ELEMENT
489: C
490: DO 31 K=1, 8
491: DO 31 L=1, 8
492: 31 ZWK(B(K, L))=ZERO
493: ZWK(B(1, 1))=CMPLX(6, 0*CFAC, 15d, 0*CKP)
494: ZWK(B(2, 2))=ZWK(B(1, 1))
495: ZWK(B(3, 1))=CMPLX(3, 0*H(J)*CFAC, 22, 0*H(J)*CKP)
496: ZWK(B(3, 3))=CMPLX(2, 0*H(J)*H(J)*CFAC, 4, 0*H(J)*H(J)*CKP)
497: ZWK(B(4, 2))=ZWK(B(3, 1))
498: ZWK(B(4, 4))=ZWK(B(3, 3))
499: ZWK(B(5, 1))=CMPLX(-6, 0*CFAC, 54, 0*CKP)
500: ZWK(B(5, 3))=CMPLX(-3, 0*H(J)*CFAC, 13, 0*H(J)*CKP)
501: ZWK(B(5, 5))=ZWK(B(1, 1))
502: ZWK(B(6, 2))=ZWK(B(5, 1))
503: ZWK(B(6, 4))=ZWK(B(5, 3))
504: ZWK(B(6, 6))=ZWK(B(5, 5))
505: ZWK(B(7, 1))=CMPLX(3, 0*H(J)*CFAC, -13, 0*H(J)*CKP)
506: ZWK(B(7, 3))=CMPLX(H(J)*H(J)*CFAC, -3, 0*H(J)*H(J)*CKP)
507: ZWK(B(7, 5))=CMPLX(-3, 0*H(J)*CFAC, -22, 0*H(J)*CKP)
508: ZWK(B(7, 7))=CMPLX(2, 0*H(J)*H(J)*CFAC, 4, 0*H(J)*H(J)*CKP)
509: ZWK(B(8, 2))=ZWK(B(7, 1))
510: ZWK(B(8, 4))=ZWK(B(7, 3))
511: ZWK(B(8, 6))=ZWK(B(7, 5))
512: ZWK(B(8, 8))=ZWK(B(7, 7))
513: CALL ZTM1(ZWK8, 8, 8)

```

```

514: CALL STRE(ZWKB, B, ZRB, KO, M4)
515: CALL ZEROCL(ZWKB, B, B, B)
516: 2 CONTINUE
517: C
518: C MAKE B MATRIX
519: C
520: MM=NLAY*4
521: CALL ZTENCH((ZD, ZB, MM, M4)
522: DO 4 L=1, MM
523: DO 4 K=1, MM
524: ZB(K,L)=ZB(K,L)+ZD(K,L)
525: 4 CONTINUE
526: CCC CALL POUTM(ZA, MM, MM, KP, M4)
527: CCC CALL POUTM(ZD, MM, MM, KP, M4)
528: CCC WRITE(*,*) ZA(3, 1), ZD(5, 6), KO
529: CCC CALL POUTM(ZA, MM, MM, KP, M4)
530: CCC CALL POUTM(ZC, MM, MM, KP, M4)
531: CCC CALL POUTM(ZB, MM, MM, KP, M4)
532: MM=NLAY*2
533: CCC CALL POUTM(ZAT, MM, MM, KP, M2)
534: CCC CALL POUTM(ZKT, MM, MM, KP, M2)
535: CCC CALL POUTM(ZCMT, MM, MM, KP, M2)
536: WRITE(*,*) ' MATRIX COMPLETE'
537: RETURN
538: END
539: C *****
540: SUBROUTINE RDMAT(IND1)
541: C *****
542: IMPLICIT COMPLEX*16(Z)
543: PARAMETER (MLY=10)
544: PARAMETER (M2=(MLY+1)*2)
545: PARAMETER (M4=(MLY+1)*4)
546: DIMENSION IND1(MLY,*)
547: COMMON /NOCH1/ ZA(M4, M4)
548: COMMON /NOCH2/ ZK(M4, M4), ZCM(M4, M4)
549: COMMON /NOCH4/ ZB(M4, M4), ZRB(M4, M4)
550: COMMON /NOCH6/ ZAT(M2, M2), ZKT(M2, M2), ZCMT(M2, M2)
551: COMMON /COND1/ KC, KP, KO(4), N, AV, MM
552: COMMON /JYD/ MJYD, MJR2, MOTH, MR1, MRR2, MRTH, IND2(6)
553: COMMON /DMY1/ ZB1MA, M4), ZW2(M2, M2)
554: WRITE(*,*) 'SUBROUTINE RDMAT NOW'
555: C COUNT DEGREE OF FREEDOM
556: C
557: C
558: N1=0
559: DO 10 N=1, 6
560: IJ=0
561: N1=N1+1
562: DO 11 J=1, NLAY
563: IF((IND1(IJ, N).NE. 0) IJ=IJ+1
564: 11 CONTINUE
565: IND2(N1)=IJ
566: 10 CONTINUE
567: WRITE(KP, 602)
568: DO 12 J=1, NLAY
569: 12 WRITE(KP, 603) J, (IND1(J, I), I=1, 6)
570: WRITE(KP, 604) (IND2(I), I=1, 6)

```

```

571: C
      MURZ=IND2(1)+IND2(2)+IND2(3)+IND2(4)
572: MUTH=IND2(5)+IND2(6)
573: MUJD=MURZ+MUTH
574: MRRZ=IND2(1)+IND2(2)+IND2(3)
575: MUTH=IND2(5)
576: MRR1=MURZ+MUTH
577: MRR2=MUTH
578: C
579: WRITE(MP,600) MUJD
580: C
581: C REARRANGE MATRIX
582: C
583: CALL REARRG(ZA, ZW, NLAY*4, M4, 4)
584: NH=NLAY*4
585: CCC CALL POUTH(ZA, MM, MN, KP, M4)
586: CALL REARRG(ZA, MM, MN, KP, M4, 4)
587: CALL REARRG(ZB, ZW, NLAY*4, M4, 4)
588: CALL REARRG(ZCM, ZW, NLAY*4, M4, 4)
589: CALL REARRG(ZRB, ZW, NLAY*4, M4, 4)
590: CALL REARRG(ZAT, ZW, NLAY*2, M2, 2)
591: CALL REARRG(ZKT, ZW, NLAY*2, M2, 2)
592: CALL REARRG(ZCMT, ZW, NLAY*2, M2, 2)
593: CCC CALL POUTH(ZB, MM, MN, KP, M4)
594: NH=NLAY*2
595: CCC CALL POUTH(ZAT, MM, MN, KP, M2)
596: CCC CALL POUTH(ZKT, MM, MN, KP, M2)
597: CCC CALL POUTH(ZCMT, MM, MN, KP, M2)
598: C
599: C REDUCE DEGREE OF FREEDOM
600: C
601: K1=0
602: K1=0
603: DO 20 N=1,4
604: DO 1 J=1,NLAY
605: K=K+1
606: IF(IND1(J,N).LE.0) GO TO 1
607: K1=K1+1
608: L1=0
609: L=0
610: DO 21 N1=1,4
611: DO 2 J1=1,NLAY
612: L=L+1
613: IF(IND1(J,N1).LE.0) GO TO 2
614: L1=L1+1
615: ZAK1,L1)=ZA(K,L)
616: ZBK1,L1)=ZB(K,L)
617: ZCK1,L1)=ZC(K,L)
618: ZCMK1,L1)=ZCM(K,L)
619: 2 CONTINUE
620: 21 CONTINUE
621: 1 CONTINUE
622: 20 CONTINUE
623: C ++++++ ++++++ ++++++ ++++++ ++++++ ++++++
624: C THETA DIRECTION
625: C ++++++ ++++++ ++++++ ++++++ ++++++ ++++++
626: K1=0
627: K1=0

```

```

628: DO 41 N=5,6
629: DO 41 J=1, NLAY
630: K=K+1
631: IF (IND1(J,N).LE.0) GO TO 41
632: K1=K1+1
633: L1=0
634: DO 43 N1=5,6
635: DO 43 J1=1, NLAY
636: L=L+1
637: IF (IND1(J1,N1).LE.0) GO TO 43
638: L1=L1+1
639: ZAT(K1,L1)=ZAT(K,L)
640: ZKT(K1,L1)=ZKT(K,L)
641: ZCMT(K1,L1)=ZCMT(K,L)
642: 43 CONTINUE
643: 41 CONTINUE
644: 41 CONTINUE
645: C
646: CCC CALL POUTM(ZA,MJZ,MJZ,KP,M4)
647: CCC CALL POUTM(ZAT,MJTH,MJTH,KP,M2)
648: WRITE(*,*), 'ROUTAT COMPLETE'
649: 600 FORMAT(1H0, '**** NUMBER OF DEGREE OF FREEDOM = ', I5, ' ***')
650: IF (K1.NE. L1) WRITE(1W, 601) K1
651: 601 FORMAT(1H0, 'K1= ', I5)
652: 602 FORMAT(1H1, 'DEGREE OF FREEDOM', //, 'LAYER NO.', '
653: *2X, ' SOLID-R SOLID-Z FLUID-R FLUID-Z SOLID-TH FLUID-TH')
654: 603 FORMAT(1H ,15, 3X, 6I8)
655: 604 FORMAT(1H ,TOTAL=', 2X, 6I8)
656: RETURN
657: END
658: C ****ROUTINE EIGENZA,ZAT,ZB
659: C ****IMPLICIT REAL*B(A-H,0-Y), COMPLEX*16(Z)
660: C ****
661: DIMENSION ZA(M4,*), ZB(M4,*), ZAT(M2,*)
662: PARAMETER (MLY=10)
663: PARAMETER (M2=(MLY+1)*2)
664: PARAMETER (MB=MLY+6)
665: PARAMETER (M4=(MLY+1)*4)
666: PARAMETER (MB=MLY+8)
667: COMMON /NDCH3/ ZPH1(M6,MLY*4), ZPINV(MLY*4,MLY*4)
668: COMMON /NDCH3/ ZPH2(M6,MLY*4), ZCTINV(MLY*2,MLY*2),
669: COMMON /NDCH3/ ZCINV(MLY*6,MLY*4), ZDMC3(MLY*3,MLY)
670: + ZC21(MLY,MLY), ZDMC5(MLY*3,MLY)
671: COMMON /DRY1/ EGR(MB, MB)
672: COMMON /DRY2/ ECI(MB, MB)
673: COMMON /DRY3/ PR(MB, MB)
674: COMMON /DRY4/ P1(MB, MB)
675: COMMON /DRY5/ ZW(MLY*4,MLY*4), ZDM5(2400-16*MLY*MLY)
676: COMMON /DRY6/ HSR(MB), HSI(MB), FV1(MB), FV2(MB), FV3(MB)
677: + ZW1(MLY*4,MLY*4), ZDMS(2400-16*MLY*MLY-20*MLY)
678: COMMON /COND1/ KC, KP, KG(4), NLAY, MM
679: COMMON /HSFNC/ ZHS(MLY*4), ZHO(MLY*4), ZH1(MLY*4)
680: COMMON /JYD/ MJYD, MJRZ, MJTH, MRL1, MRRZ, MRT, MUR, MUZ, MNR, MUZ, MWT, MWT
681: ZERO=CMPLX(0, 0, 0)
682: ZA1=CMPLX(0, 0, 1, 0)
683: WRITE(*,*), 'SUBROUTINE EIGEN NOW'
684: C

```

```

685: CALL ZEROCL(ZW1,MLY*4,MLY*4,MLY*4)
686: CALL ZEROCL(ZPH1,M6,MLY*4,MLY*4)
687: CALL ZEROCL(ZPINV,MLY*4,MLY*4,MLY*4)
688: CALL ZEROCL(ZW,MLY*4,MLY*4,MLY*4)
689: C DO 1 L=1,MRRZ
690: DO 1 K=1,MRRZ
691: ZW(K,L)=ZA(K,L)
692: 1 CONTINUE
693: CALL ZMULTI(ZCINV,ZB,ZW1,MJRZ,MJRZ,MJRZ,MLY*4,M4,MLY*4)
694: CALL POUTM(ZW1,MJRZ,MJRZ,KP,MLY*4)
695: CCC DO 2 L=1,MJRZ
696: DO 2 K=1,MJRZ
697: DO 2 K=1,MJRZ
698: EGR(K,L)=REAL(ZW1(K,L))
699: EQ1(K,L)=AIMAG(ZW1(K,L))
700: 2 CONTINUE
701: CALL ZMULTI(ZCINV,ZW,ZW1,MJRZ,MJRZ,MJRZ,MLY*4,MLY*4,MLY*4)
702: DO 3 L=1,MJRZ
703: DO 3 K=1,MJRZ
704: EGR(K,L+MJRZ)=REAL(ZW1(K,L))
705: EQ1(K,L+MJRZ)=AIMAG(ZW1(K,L))
706: 3 CONTINUE
707: CCC CALL POUTM(ZW1,MJRZ,MJRZ,KP,MLY*4)
708: DO 10 L=1,MJRZ
709: DO 10 K=1,MJRZ
710: VAL=0.0
711: IF(K.EQ.L) VAL=-1.0
712: EGR(K+MJRZ,L)=VAL
713: EQ1(K+MJRZ,L)=0.0
714: EGR(K+MJRZ,L+MJRZ)=0.0
715: EQ1(K+MJRZ,L+MJRZ)=0.0
716: 10 CONTINUE
717: C -----+
718: C -----+ SOLVE EIGEN VALUE PROBLEM
719: C -----+
720: MSQL=MJRZ+MRRZ
721: CALL CQ(MLY*8,MSOL,EGR,EQ1,HSH,HSI,1,PR,PI,FV1,FV2,FV3,IERR)
722: WRITE(KP,600) IERR
723: IF(IERR.NE.0) GO TO 99
724: C -----+
725: C -----+ SELECT EIGEN VALUE AND MAKE PHAI MATRIX
726: C -----+
727: CALL SLCTHS(ZPH1,ZHS,HSH,HSI,PR,PI,MSOL)
728: WRITE(KP,602) MRRZ
729: DO 23 N=1,MRRZ
730: WRITE(KP,603) N,ZHS(N)
731: 23 CONTINUE
732: C -----+
733: C ++++++ THE TA DIRECTION ++++++
734: C +++++++
735: C +++++++
736: CALL ZEROCL(ZW1,MLY*4,MLY*4,MLY*4)
737: CALL ZEROCL(ZW,MLY*4,MLY*4,MLY*4)
738: CALL ZMULTI(ZCINV,ZAT,ZW,MUT,MUT,MLY*2,M2,MLY*4)
739: DO 12 L=1,MUT
740: DO 12 K=1,MUT
741: EGR(K,L)=0.0

```

```

742: EQU(K,L)=0.0
743: VAL=0.0
744: IF (K.EQ.L) VAL=-1.0
745: EGR(K*MUT,L)=VAL
746: EQU(K+MUT,L)=0.0
747: EGR(K+MUT,L+MUT)=0.0
748: EQU(K+MUT,L+MUT)=0.0
749: EGR(K,L+MUT)=REAL(ZW(K,L))
750: EQU(K,L+MUT)=AIMAG(ZW(K,L))
751: 12 CONTINUE
752: MSOL=MUT**2
753: CALL COMLY*B,MSOL,EGR,EQU,HSL,1,PR,PI,FV1,FV2,FV3,IERR2)
754: WRITE(KP,601) IERR2
755: IF (IERR2.NE.0) GO TO 99
756: L1=0
757: DO 20 L=1,MSOL
758: Z1=-1.0*CMPLX(HSL(L),HSL(L))
759: C1=AIMAG(Z1)
760: IF (C1.LT.0.0) GO TO 20
761: L1=L1+1
762: ZHS(L1+MRRZ)=Z1
763: DO 21 K=1,MUT
764: ZPHI(K+MUT,MRRZ+L1)=CMPLX(PR(K,L),PI(K,L))
765: 21 ZW(K,L1)=CMPLX(PR(K,L),PI(K,L))
766: 20 CONTINUE
767: WRITE(KP,602) L1
768: DO 24 L=L1,LL
769: WRITE(KP,603) L+MRRZ,ZHS(MRRZ+L)
770: 24 CONTINUE
771: IF (L1.NE.MRTH) GO TO 97
772: C
773: CALL 2MULT1(ZC221,ZW,ZW1,MUT,MUT,MUT,MLY,MLY*4,MLY*4)
774: DO 16 L=1,MUT
775: DO 16 K=1,MUT
776: P1=-REAL(ZW1(K,L))
777: P2=-AIMAG(ZW1(K,L))
778: ZPHI(MRRZ+MUT+K,MRRZ+L)=CHPLX(P1,P2)
779: 16 CONTINUE
780: C
781: C-----NORMARIZE ZPHI MATRIX-----
782: C-----783: C-----784: C-----785: C-----786: C-----787: C-----788: C-----789: C-----790: C-----791: C-----792: C-----793: C-----794: CCC-----795: CCC-----796: C-----797: C-----798: C
    EQU(K,L)=0.0
    IF (K.EQ.L) VAL=-1.0
    EGR(K*MUT,L)=VAL
    EQU(K+MUT,L)=0.0
    EGR(K+MUT,L+MUT)=0.0
    EQU(K+MUT,L+MUT)=0.0
    EGR(K,L+MUT)=REAL(ZW(K,L))
    EQU(K,L+MUT)=AIMAG(ZW(K,L))
    MSOL=MUT**2
    CALL COMLY*B,MSOL,EGR,EQU,HSL,1,PR,PI,FV1,FV2,FV3,IERR2)
    WRITE(KP,601) IERR2
    IF (IERR2.NE.0) GO TO 99
    L1=0
    DO 20 L=1,MSOL
    Z1=-1.0*CMPLX(HSL(L),HSL(L))
    C1=AIMAG(Z1)
    IF (C1.LT.0.0) GO TO 20
    L1=L1+1
    ZHS(L1+MRRZ)=Z1
    DO 21 K=1,MUT
    ZPHI(K+MUT,MRRZ+L1)=CMPLX(PR(K,L),PI(K,L))
    ZW(K,L1)=CMPLX(PR(K,L),PI(K,L))
    20 CONTINUE
    WRITE(KP,602) L1
    DO 24 L=L1,LL
    WRITE(KP,603) L+MRRZ,ZHS(MRRZ+L)
    24 CONTINUE
    IF (L1.NE.MRTH) GO TO 97
    CALL 2MULT1(ZC221,ZW,ZW1,MUT,MUT,MUT,MLY,MLY*4,MLY*4)
    DO 16 L=1,MUT
    DO 16 K=1,MUT
    P1=-REAL(ZW1(K,L))
    P2=-AIMAG(ZW1(K,L))
    ZPHI(MRRZ+MUT+K,MRRZ+L)=CHPLX(P1,P2)
    16 CONTINUE
    DO 30 L=1,MRRZ+MRTH
    VMAX=ABS(ZPHI(1,L))
    DO 31 K=2,MJYD
    P1=ABS(ZPHI(K,L))
    V1=ABS(ZPHI(K,L))
    IF (V1.GT.VMAX) VMAX=V1
    31 CONTINUE
    DO 32 K=1,MJYD
    ZPHI(K,L)=ZPHI(K,L)/VMAX
    32 CONTINUE
    30 CONTINUE
    CCC WRITE(KP,*), 'MATRIX PHAI'
    CALL POUTMZPHI,MJYD,MJYD,M1,KP,M6)
    CCC WRITE(*,*), 'EIGEN COMPLETE'
    FORMAT

```

```

799: C      FORMAT(1H , '*** TERR= ', I4, ' ***')
800: 600 FORMAT(1H , '*** IERR2= ', I3, ' ***')
801: 601 FORMAT(1H , '*** NUMBER = ', I5)
802: 602 FORMAT(1H , 'NUMBER OF SELECTED WAVE NUMBER = ', I5)
803: 603 FORMAT(1H , 'N= ', I3, 'X, 'WAVE NUMBER= ', E15. 5)
804: GO TO 98
805: 97 STOP 'ERR IN EIGEN THETA DIRECTION'
806: 99 STOP 'STOP TERR NE. 0'
807: 98 CONTINUE
808: RETURN
809: END
810: C ****SUBROUTINE RMATRX (RR, ZPHI, ZPINV)
811: C ****SUBROUTINE RMATRX (RR, ZPHI, ZPINV)
812: C ****
813: IMPLICIT COMPLEX*16(Z)
814: PARAMETER (MLY=10)
815: PARAMETER (MRES=25*MLY**2+50*MLY+25)
816: PARAMETER (ML6=MLY*6)
817: DIMENSION ZPHI (ML6, *, ZPINV(MLY*4, *), ZH1 (MLY*4))
818: COMMON /HSFCNC/ ZHS(MLY*4), ZH0(MLY*4), ZH1 (MLY*4)
819: COMMON /COND1/ KC, KP, KG(4), MLLAY, MMH
820: COMMON /JYJD/ MJYD, MURZ, MJTH, MR1, MRRZ, MRTH, IND2(6)
821: COMMON /DMY1/ ZR(MLY*4, MLY*4), ZW(MLY*4, MLY*4)
822: COMMON /DMY2/ ZAMAT(M5, M5), ZDM2(3200-MRES)
823: COMMON /DMY3/ ZGMAP(M5, M5), ZDM3(3200-MRES)
824: COMMON /DMY4/ ZSMAT(M5, M5), ZDM4(3200-MRES)
825: COMMON /DMY5/ ZVMAT(M6, MLY*4)
826: COMMON /DMY6/ ZPMAT(M6, MLY*4)
827: COMMON /DMY7/ ZPMAT(M6, MLY*4)
828: C      WRITE(*,*) 'SUBROUTINE RMATRX NOW'
829: C
830: C
831: J=MJYD-IND2(6)
832: CALL ZMULT1(ZAMAT, ZVMAT, ZW, MR1, JJ, MR1, M5, M6, MLY*4)
833: DO 1 L=1, MRI
834: ZHS2=ZHS(L)*ZHS(L)
835: DO 1 K=1, MRI
836: ZRK(L)=ZWK(K, L)*ZHS2
837: 1 CONTINUE
838: C
839: CALL ZMULT1(ZGMAP, ZPHAT, ZW, MR1, JJ, MR1, M5, M6, MLY*4)
840: DO 2 L=1, MRI
841: ZHS1=ZHS(L)
842: DO 2 K=1, MRI
843: 2 ZRK(L)=ZRK(K, L)+ZWK(K, L)*ZHS1
844: CALL ZMULT1(ZSMAT, ZVMAT, ZW, MR1, JJ, MR1, M5, M6, MLY*4)
845: DO 3 L=1, MRI
846: DO 3 K=1, MRI
847: 3 ZRK(L)=ZRK(K, L)+ZWK(K, L)
848: CALL ZMULT1(ZR, ZPINV, ZW, MR1, MRI, MR1, MLY*4, MLY*4)
849: DO 4 L=1, MRI
850: DO 4 K=1, MRI
851: 4 ZRK(L)=ZWK(K, L)*RR
852: CCC WRITE(MR, *) 'MATRIX R'
853: CCC CALL POUTM(ZR, MRI, MRI, KP, MLY*4)
854: CCC WRITE(*, *) 'RMATRX COMPLETE'
855: RETURN

```

```

856: END
857: C ****ROUTINE CMATRIX(ZCM, ZCMT, ZK, ZKT, CMG, CMG2)
858: C ****ROUTINE ZCMT(M2, *, ZK(M2, *), ZKT(M2, *))
859: C ****IMPLICIT COMPLEX*16(Z)
860:
861: PARAMETER (MLY=10)
862: PARAMETER (M4=(MLY+1)*4)
863: PARAMETER (M2=(MLY+1)*2)
864: DIMENSION ZCM(M4,*), ZCMT(M2,*), ZK(M4,*), ZKT(M2,*)
865: COMMON /COND1/ KC, KP, KG(4), NLAY, MM
866: COMMON /JYD/ MJD, MURZ, MJTH, MR1, MRRZ, MRTH, IND2(6)
867: COMMON /NOCH5/ ZCINV(MLY*4,MLY*4), ZCTINV(MLY*2,MLY*2),
868: + ZC221(MLY,MLY) ZDM5(MLY*3,MLY)
869: COMMON /DMY1/ ZC11(MLY,MLY), ZC12(MLY,MLY), ZC21(MLY,MLY),
870: + ZC22(MLY,MLY), ZW(MLY,MLY)
871: C
872: WRITE(*,*) 'SUBROUTINE CMATRIX NOW'
873: CALL ZERDCL(ZC221,MLY,MLY,MLY)
874: DO 1 K=1,MJRZ
875: DO 1 L=1,MJRZ
876: RR=-REAL(ZCM(K,L))*OMG2
877: RI=AIMAG(ZCM(K,L))*OMG
878: ZCINV(K,L)=Cmplx(RR,RI)+2K(K,L)
879: 1 CONTINUE
880: CCC CALL POUT((ZCINV, MURZ, MJRZ, KP, MLY*4)
881: CALL ZINV((ZCINV, MURZ, MJRZ, KP, MLY*4*2)
882: CCC CALL POUT((ZCINV, MJTH, MURZ, KP, MLY*4)
883: C
884: DO 2 K=1,MOTH
885: DO 2 L=1,MOTH
886: RR=-REAL(ZCMT(K,L))*OMG2
887: RI=AIMAG(ZCMT(K,L))*OMG
888: ZCTINV(K,L)=Cmplx(RR,RI)+ZKT(K,L)
889: 2 CONTINUE
890: CCC CALL POUT((ZCTINV, MJTH, KP, MLY*2)
891: C
892: J=IND2(5)
893: MW=IND2(6)
894: IF (MWT.EQ.Q) GO TO 25
895: DO 3 K=1,JJ
896: DO 5 L=1,JJ
897: 5 ZC11(K,L)=ZCTINV(K,L)
898: DO 6 L=1,MWT
899: 6 ZC12(K,L)=ZCTINV(K,L+JJ)
900: 3 CONTINUE
901: DO 10 K=1,MWT
902: DO 11 L=1,JJ
903: 11 ZC21(K,L)=ZCTINV(K+JJ,L)
904: DO 12 L=1,MWT
905: 12 ZC22(K,L)=ZCTINV(K+JJ,L+JJ)
906: 10 CONTINUE
907: C
908: CALL ZINV1(ZC22, MW, MW, MWT, MWT, JJ, MLY, MLY, MLY)
909: CALL ZMULT1(ZC22, ZC21, ZC21, MWT, MWT, JJ, MLY, MLY, MLY)
910: CALL ZMULT1(ZC12, ZC221, ZC221, ZW, JJ, MWT, JJ, MLY, MLY, MLY)
911: C
912: DO 4 K=1,JJ

```

```

913: DO 4 L=1, JJ
914:   4 ZCTINV(K,L)=ZC11(K,L)-ZW(K,L)
915: 25 CONTINUE
916: CCC CALL POUTM(ZCTINV,JJ, JJ, KP, MLY*2)
917: CALL ZINV1(ZCTINV,JJ, JJ, MLY*2*2)
918: CCC CALL POUTM(ZCTINV,JJ, JJ, KP, MLY*2)
919: WRITE(*,*), CMATRX COMPLETE
920: RETURN
921: END
922: C ****
923: SUBROUTINE HANKEL(MR, R)
924: C ****
925: IMPLICIT COMPLEX*16(Z)
926: PARAMETER (MLY=10)
927: COMMON /HSFNC/ ZH5(MLY*4), ZH0(MLY*4), ZH1(MLY*4)
928: C
929: DO 1 N=1, MR
930: ZZ=ZHS(N)*R
931: CALL HANKEL(ZZ, ZHO(N), ZH1(N))
932: 1 CONTINUE
933: RETURN
934: END
935: C ****
936: SUBROUTINE MATR2(R1, IND1)
937: C ****
938: IMPLICIT COMPLEX*16(Z)
939: PARAMETER (MLY=10)
940: PARAMETER (M5=(MLY+1)*5)
941: PARAMETER (MRES=55*MLY**2+50*MLY+25)
942: DIMENSION IND1(MLY,*)
943: COMMON /JYD/ MJRD, MRRZ, MUTH, MR1, MRRZ, MUTH, IND2(6)
944: COMMON /GROUND/ ZAA(10), ZG(10), ZAL(10), ZQ(10), PERM(10),
945: + ROP(10), CNE(10), RD(10), ROF(10), EI(10), DEP(11), DEM(10),
946: COMMON /COND1/ KC, KP, KG(4), NLAY, MMN
947: COMMON /DMY1/ ZNM5, MB) ZWK(5, 5), ZWK10(10, 10),
948: COMMON /DMY2/ ZAMAT(M5, M5), ZDM2(3200-MRES)
949: COMMON /DMY3/ ZPAT(M5, M5), ZDM3(3200-MRES)
950: COMMON /DMY4/ ZPAT(M5, M5), ZDM4(3200-MRES)
951: C
952: ZERO=CHPLX(0, 0, 0)
953: C
954: WRITE(*,*), 'SUBROUTINE MATRIX2 NOW'
955: C
956: C INITIAL ZERO CLEAR
957: C
958: CALL ZEROCL(ZMK, 5, 5)
959: CALL ZEROCL(ZAMAT, M5, M5, M5)
960: CALL ZEROCL(ZGMAT, M5, M5, M5)
961: CALL ZEROCL(ZSMAT, M5, M5, M5)
962: CALL ZEROCL(ZW, M5, M5, M5)
963: DO 2 J=1, NLAY
964: ZA20=ZAA(J)-2, 0*ZG(J)
965: ZAQ=ZAL(J)*ZQ(J)
966: KO=J#5-4
967: C
968: C MATRIX A ELEMENT (UPPER LEFT SIDE 1/4)
969: C

```

```

970: ZWK(1, 1)=2. 0*ZAA(J)
971: ZWK(1, 4)=2. 0*ZAQ
972: ZWK(2, 2)=2. 0*ZQ(J)
973: ZWK(3, 3)=2. 0*ZQ(J)
974: ZWK(4, 1)=2. 0*ZAQ
975: ZWK(4, 4)=2. 0*ZQ(J)
976: CALL COPY4(ZWK, ZWK10, 5, 0, 5, 1, 0, H(J)/6, 0)
CALL STRE(ZWK10, 10, ZMAT, KO, M5)
977: C
978: C
----- MATRIX D ELEMENT (UPPER LEFT SIDE 1/4) AND DD ELEMENT
979: C
980: C
----- ZWK(1, 2)=ZA2G
981: ZWK(1, 5)=ZAQ
982: ZWK(2, 1)=ZQ(J)
983: ZWK(4, 2)=ZAQ
984: ZWK(4, 5)=ZAQ(J)
985: ZWK(4, 5)=ZQ(J)
986: CALL COPY4(ZWK, ZWK10, 5, 1, 0,-1, 0, -1, 0, 0, 5)
CALL STRE(ZWK10, 10, ZMAT, KO, M5)
987: C
----- MATRIX E ELEMENT
988: C
----- ZCC=ZG(J)*H(J)/(6, 0*R1)
991: ZC2=ZCC*NMM
992: ZW(K(1, 1)=-4. 0*ZCC
993: ZWK(1, 3)=4. 0*ZC2
994: ZWK(3, 1)=4. 0*ZC2
995: ZWK(3, 3)=-4. 0*ZCC
996: ZWK(2, 2)=2. 0*ZCC
997: CALL COPY4(ZWK, ZWK10, 5, 0, 5, 0, 5, 1, 0, 1, 0)
CALL STRE(ZWK10, 10, ZW, KO, M5)
998: C
1000: 2 CONTINUE
1001: DO 3 J=1, NLAY*5
1002: DO 3 L=1, NLAY*5
3 ZOMAT(J, L)=ZGMAT(J, L)+ZW(J, L)
1003: CALL ZEROCL(ZW, M5, M5)
1004: C
1005: C
----- MATRIX F ELEMENT
1006: C
1007: C
----- DO 23 J=1, NLAY
1008: DO 23 J=1, NLAY
1009: KO=J*5-4
1010: ZCC=ZG(J)*H(J)*MMM*(MMM+1)/(3, 0*R1*R1)
1011: ZWK(1, 1)=-2. 0*ZCC
1012: ZWK(1, 3)=2. 0*ZCC
1013: ZWK(3, 1)=2. 0*ZCC
1014: ZWK(3, 3)=-2. 0*ZCC
1015: CALL COPY4(ZWK, ZWK10, 5, 0, 5, 0, 5, 1, 0, 1, 0)
CALL STRE(ZWK10, 10, ZW, KO, M5)
1016: C
----- MATRIX G ELEMENT
1017: C
1018: C
----- ZCC=ZG(J)*NMM/R1
1019: C
1020: ZWK(2, 1)=-ZCC
1021: ZWK(2, 3)=ZCC
1022: CALL COPY4(ZWK, ZWK10, 5, 1, 0,-1, 0, -1, 0, 0, 5)
CALL STRE(ZWK10, 10, ZSMAT, KO, M5)
1023: C
1024: 23 CONTINUE
1025: C
1026: DO 24 J=1, NLAY*5

```

```

1027: DO 24 L=1,NLAY+5
1028: 24 ZSMAT(J,L)=ZSMAT(J,L)+ZW(J,L)
1029: CALL REARR(ZGMA1,ZW,NLAY*5,M5,5)
1030: CALL REARR(ZGMA1,ZW,NLAY*5,M5,5)
1031: CALL REARR(ZGMA1,ZW,NLAY*5,M5,5)

1032: C   CALL POUTM(ZAMAT,NLAY*4,NLAY*5,KP,M5)
1033: CCC  CALL POUTM(ZQMAT,NLAY*,NLAY*5,KP,M5)
1034: CCC  CALL POUTM(ZSMAT,NLAY*4,NLAY*5,KP,M5)
1035: CCC  CALL POUTM(ZSMAT,NLAY*4,NLAY*5,KP,M5)

1036: C
1037: C ----- REDUCE DEGREE OF FREEDOM -----
1038: C----- 1039: C -----
1040: L1=0
1041: L=0
1042: DO 40 N=1,5
1043:   DO 40 J=1,NLAY
1044:     III=IND1(J,3)
1045:     IF(N.LE.3) III=IND1(J,1)
1046:     L=L+1
1047:     IF(III.LE.0) GO TO 40
1048:     L1=L1+1
1049:     DO 41 K=1,NLAY+4
1050:       ZAMAT(K,L1)=ZAMAT(K,L)
1051:       ZGMA1(K,L1)=ZGMA1(K,L)
1052:       ZSMAT(K,L1)=ZSMAT(K,L)
41  CONTINUE
1053: 40  CONTINUE
1054: C
1055: C
1056: K1=0
1057: K=0
1058: DO 50 N=1,4
1059:   DO 50 J=1,NLAY
1060:     III=IND1(J,3)
1061:     IF(N.LE.3) III=IND1(J,1)
1062:     K=K+1
1063:     IF(III.EQ.0) GO TO 50
1064:     K1=K1+1
1065:     DO 51 L=1,NLAY*5
1066:       ZAMAT(K1,L)=ZAMAT(K,L)
1067:       ZGMA1(K1,L)=ZGMA1(K,L)
1068:       ZSMAT(K1,L)=ZSMAT(K,L)
51  CONTINUE
1069: 50  CONTINUE
1070: C
1071: C
1072: J=MYID-IND2(6)
1073: CCC  CALL POUTM(ZAMAT,MRI,J,J,KP,M5)
1074: CCC  CALL POUTM(ZGMA1,MRI,J,J,KP,M5)
1075: CCC  CALL POUTM(ZSMAT,MRI,J,J,KP,M5)
1076: C
1077: RETURN
1078: END
1079: C **** SUBROUTINE PINV(NNN,R,1) ****
1080: C ****
1081: C **** IMPLICIT COMPLEX*16(Z)
1082: C PARAMETER (MLy=10)
1083: C

```

```

1084: PARAMETER (M5=MLY*6)
1085: C
1086: DIMENSION ZSS(MLY*4, MLY)
1087: C
1088: COMMON /HSFNC/ ZHS(MLY*4), ZHO(MLY*4), ZHI(MLY*4)
1089: COMMON /JYD/ MJYD, MJRZ, MJTH, MR1, MRRZ, MRTH, MUR, MUZ, MWR, MNZ, MUT, MWT
1090: COMMON /COND/ GAL, PAI, DEPTH, NOT, VSB, NFRQ, RR, DDO, NCASE
1091: COMMON /NOCH3/ KC, KP, KG(4), NLAY, MNN
1092: COMMON /NOCH5/ ZPHI1(M5, MLY*4), ZPHI2(M5, MLY*4), ZPHI3(M5, MLY*4)
1093: COMMON /NOCH5/ ZPHIM1(M5, MLY*4), ZPHIM2(M5, MLY*4), ZPHIM3(M5, MLY*4)
1094: COMMON /DMYS/ ZVMAT(M5, MLY*4)
1095: COMMON /DMY6/ ZPMAT(M5, MLY*4)
1096: C
1097: J1=MJR
1098: J2=MIR+MU2
1099: J3=MIR+MU2+MUT
1100: CK=MNM/R1
1101: C
1102: CALL ZEROCL(ZVMAT, M6, MLY*4, M6)
1103: CALL ZEROCL(ZPMAT, M6, MLY*4, M6)
1104: CALL ZEROCL(ZPHIM, M6, MLY*4, M6)
1105: CALL ZEROCL(ZPHINV, M6, MLY*4, MLY*4)
1106: DO 10 L=1, MRRZ
1107: IF (MNM EQ. 0) THEN
1108: ZHM=ZHO(L)
1109: ZHMD=-ZHS(L)*ZHI(L)
1110: ZHM1=-ZHI(L)
1111: ELSE IF (MNM EQ. 1) THEN
1112: ZHM=ZHI(L)
1113: ZHMD=-ZHI(L)/R1+ZHS(L)*ZHO(L)
1114: ZHM1=ZHO(L)
1115: END IF
1116: DO 1 J=1, J1
1117: ZPHIM(J,L)=ZHMD*ZPHI(J,L)
1118: ZVMAT(J,L)=ZHM*ZPHI(J,L)
1119: 1 ZPMAT(J,L)=-ZHM1*ZPHI(J,L)
1120: DO 2 J=J+1, J2
1121: ZPHIM(J,L)=ZHS(L)*ZHM*ZPHI(J,L)
1122: ZVMAT(J,L)=-ZHM1*ZPHI(J,L)
1123: 2 ZPMAT(J,L)=ZHM*ZPHI(J,L)
1124: DO 3 J=1, J1
1125: J1=J+J2
1126: ZPHIM(J,J1,L)=ZHM*ZPHI(J,J1)*CK
1127: 3 CONTINUE
1128: DO 4 J=J3+1, J3+MNR
1129: ZPHIM(J,L)=ZHMD*ZPHI(J,L)
1130: ZVMAT(J,L)=ZHM*ZPHI(J,L)
1131: 4 ZPMAT(J,L)=-ZHM1*ZPHI(J,L)
1132: DO 5 J=J3+MNR+1, J3+MNZ
1133: ZPHIM(J,L)=ZHB(L)*ZHM*ZPHI(J,L)
1134: ZVMAT(J,L)=-ZHM1*ZPHI(J,L)
1135: 5 ZPMAT(J,L)=ZHM*ZPHI(J,L)
1136: DO 6 J=1, MWR
1137: J1=J+MZR+MUT
1138: J2=J+J3
1139: ZPHIM(J1,L)=ZHM*ZPHI(J,J2, L)*CK
1140: 6 CONTINUE

```

```

1141: DO 7 J=1, J3+MNR
1142:    7 ZPINV(J,L)=ZPHIM(J,L)
1143: 10 CONTINUE
1144: C
1145: DO 20 L=MNRZ+1, MNR
1146: IF (MMN.EQ.0) THEN
1147:  ZHM=ZHO(L)
1148:  ZHMD=-ZHS(L)*ZHI(L)
1149:  ZHMM1=-ZHI(L)
1150: ELSE IF (MMN.EQ.1) THEN
1151:  ZHM=ZHI(L)
1152:  ZHMD=-ZHI(L)/R1+ZHS(L)*ZHO(L)
1153:  ZHMM1=ZHO(L)
1154: END IF
1155: CCC
1156: CCC WRITE(KP,700) ZHM, ZHMD, ZHMM1
1157: CC700 FORMAT(1H , 'ZHM=' , 2E15. 5, 'ZHMD=' , 2E15. 5, 'ZHMM1=' , 2E15. 5)
1158: CCC
1159: DO 11 J=1, J1
1160:  ZPHIM(J,L)=ZHM*ZPHI(J+J2, L)*CK
1161: 11 CONTINUE
1162: DO 14 J=1, MNR
1163:  J1=J+J3
1164:  J2=J+MNRZ+MUT
1165:  ZPHIM(J,J1,L)=ZHM*ZPHI(JJ2, L)*CK
1166: 14 CONTINUE
1167: DO 13 J=1, MUT
1168:  ZPHIM(J+J2, L)=ZHMD*ZPHI(J+J2, L)
1169:  ZMAT(J+J2, L)=ZHM*ZPHI(J+J2, L)
1170:  ZPHAT(J+J2, L)=-2NHM1*ZPHI(J+J2, L)
1171: 13 CONTINUE
1172: DO 16 J=1, MNR
1173:  J1=J+MNRZ+MUT
1174:  ZPHIM(J,J1,L)=ZHMD*ZPHI(JJ1, L)
1175:  ZMAT(J,J1,L)=ZHM*ZPHI(JJ1, L)
1176:  ZPHAT(J,J1,L)=-2NHM1*ZPHI(JJ1, L)
1177: 16 CONTINUE
1178: DO 17 J=1, J3+MNR
1179:  ZPINV(J,L)=ZPHIM(J,L)
1180: 17 CONTINUE
1181: 20 CONTINUE
1182: C
1183: C
1184: C INVERSE OF PHIM MATRIX
1185: C
1186: CCC WRITE(KP, *) 'MATRIX PHIM'
1187: CCC CALL POUT(ZPHIM,MJYD,MRI1,KP,M6)
1188: CCC WRITE(KP, *) 'MATRIX V'
1189: CCC CALL POUT(ZVMAT,MJYD,MRI1,KP,M6)
1190: CCC WRITE(KP, *) 'MATRIX PSAI'
1191: CCC CALL POUT(ZPMAT,MJYD,MRI1,KP,M6)
1192: C
1193: IF (NNN.NE.0) GO TO 100
1194: CCC WRITE(KP, *) 'MATRIX ZPINV'
1195: IF (MMN.EQ.1) THEN
1196: CALL POUT(ZPINV,MRI1,MRI1,KP,MLY*4)
1197: CALL ZINV1ZPINV,MRI1,MRI1,MLY*4*2)

```

```

1198: ELSE
1199: DO 110 K=1, M1
1200: DO 111 K=1, M1R
1201: K1=K+MUR*3
1202: 111 ZPINV(K+MUR*2, L)=ZPHIM(K1, L)
1203: DO 112 K=1, M1T
1204: K1=K+MUR*2
1205: 112 ZPINV(K+MUR*2+MUR, L)=ZPHIM(K1, L)
1206: 110 CONTINUE
1207: C
1208: CALL ZINV1(ZPINV, M1, M1, MLY*4*2)
1209: C
1210: DO 210 K=1, M1
1211: DO 211 L=1, M1R
1212: 211 ZSS(K, L)=ZPINV(K, L+MUR*2)
1213: DO 212 L=1, M1R
1214: 212 ZPINV(K, L+MUR*2)=2PINV(K, L+MUR*2+MUR)
1215: DO 213 L=1, M1R
1216: 213 ZPINV(K, L+MUR*3)=ZSS(K, L)
1217: 210 CONTINUE
1218: C
1219: END IF
1220: C
1221: CCC CALL POUTM(ZPINV, M1, M1, KP, MLY*4)
1222: C
1223: 100 RETURN
1224: END
1225: C
1226: C **** SUBROUTINE SOLVE(ZRB, NYOUT, NFGL, IND1, NFORC, QMG2) ****
1227: C ****
1228: C ****
1229: C ****
1230: IMPLICIT COMPLEX*16(Z)
1231: PARAMETER (MLY=10)
1232: PARAMETER (MS=MLY*6)
1233: PARAMETER (M4=(MLY+1)*4)
1234: DIMENSION ZRB(MA,*), IND1(MLY,*), NXOUT(*), NFORC(*)
1235: COMMON /NDCHS/ ZPHIM(M6, MLY*4)
1236: COMMON /NDCHS/ ZPHIM(M6, MLY*4)
1237: COMMON /DMY1/ ZR(MLY*4, MLY*4), ZTEMP(MLY*4), ZZW2(MLY*4)
1238: COMMON /DMY2/ ZUW(M6), ZPUX(M4), ZALP(M4), ZW2(M4), ZW3(MLY*4),
1239: + ZC11(MLY, MLY), ZC12(MLY, MLY), ZC13(MLY, MLY),
1240: 1 ZC21(MLY, MLY), ZC22(MLY, MLY), ZC23(MLY, MLY),
1241: 2 Z1221(MLY, MLY), Z1223(MLY, MLY), Z3221(MLY, MLY),
1242: 3 Z22221(MLY, MLY), Z2223(MLY, MLY), Z3223(MLY, MLY),
1243: 4 ZR5(MLY, MLY*3), ZWORK(MLY*2, MLY*2), ZW5(MLY), ZPUX1(M4)
1244: 5 ZR5(MLY, MLY*3), ZWORK(MLY*2, MLY*2), ZW5(MLY), ZPUX1(M4)
1245: COMMON /DMY3/ ZSGR(MLY, 10), ZSG2(MLY, 10), ZSGT(MLY, 10),
1246: 1 , ZTUZZ(MLY, 10), ZTUTR(MLY, 10), ZPAI(MLY, 10), ZDSGR(MLY, 10),
1247: 2 ZDSC2(MLY, 10), ZDSGT(MLY, 10)
1248: COMMON /DMY4/ ZW(MLY*4, MLY*4), ZH1(MLY*4, MLY*4)
1249: COMMON /COND1/ KC, KP, KG(4), NLAY, MM
1250: COMMON /HSFNC/ ZHS(MLY*4), ZHO(MLY*4), ZHI(MLY*4)
1251: COMMON /JYD/ MJYD, MJRZ, MOTH, MR1, MRRZ, MRT, MIZ, MIZ, M1T, MWT
1252: COMMON /RESP/ ZUR1, ZUZ1, ZUT1, ZMFI1, ZWZ1, ZWZ2, ZXKT, ZXKT
1253: ZERO=CHPLX(0, 0, 0)
1254: ZONE=CHPLX(1, 0, 0)

```

```

1255: ZAI=CMPLX(0,0,1,0)
1256: WRITE(*,*) 'SUBROUTINE SOLVE NOW'
1257: DO 80 K=1, NLAY+2
1258: ZPUX(K,K)=ZERO
1259: 80 IF (NFDRC(K).EQ.1) ZPUX(K)=ZONE
1260: MAU=MUR-MUR
1261: IF (MAU.LE.0) MAU=0
1262: MUN=MUR*3+MAU
1263: DO 81 K=1, MUR-MAU
1264: DO 82 L=1, MUN
1265: 82 ZRS(K,L)=ZR(K+MUN,L)
1266: 81 CONTINUE
1267: IF (NFLG1.GT.0) GO TO 102
1268: C
1269: C CALCULATE UR, UT
1270: C
1271: C
1272: C THERE IS NO P FILE
1273: C
1274: DO 30 L=1, MUN
1275: DO 30 K=1, MUN
1276: ZW(K,L)=ZR(K,L)
1277: 30 CONTINUE
1278: CALL ZINV1(ZW,MUN,MUN,MLY*8)
1279: CALL ZMULT1(ZW,ZPUX,ZUM,MUN,MUN,1,MLY*4,1,1)
1280: C
1281: WRITE(10P,600)
1282: C
1283: GO TO 99
1284: C
1285: C PILE EXIST CASE
1286: C
1287: 102 CONTINUE
1288: CALL CNVER1(ZR,ZC11,ZC12,ZC13,ZC21,ZC22,ZC23,ZC31,ZC32,
1289: + ZC33,MUR,NLAY,MAU)
1290: CALL ZINV1(ZC22,MUR,MUR,MLY*2)
1291: CALL ZMULT1(ZC22,ZC21,Z2221,MUR,MUR,NLAY,MLY,MLY)
1292: CALL ZMULT1(ZC22,ZC23,Z2223,MUR,MUR,MLY,MLY,MLY)
1293: CALL ZMULT1(ZC12,Z2221,Z1221,NLAY,MUR,MLY,MLY,MLY)
1294: CALL ZMULT1(ZC12,Z2223,Z1223,NLAY,MUR,MLY,MLY,MLY)
1295: CALL ZMULT1(ZC32,Z2221,Z3221,MUR,MUR,NLAY,MLY,MLY)
1296: CALL ZMULT1(ZC32,Z2223,Z3223,MUR,MUR,MLY,MLY,MLY)
1297: C
1298: CALL ZEROL1(ZWORK,MLY*2,MLY*2)
1299: DO 50 L=1, NLAY
1300: DO 61 K=1, NLAY
1301: 61 ZWORK(K,L)=ZC11(K,L)-Z1221(K,L)
1302: DO 63 K=1, MUR
1303: 63 ZWORK(K+NLAY+MUR,L)=ZC31(K,L)-Z3221(K,L)
1304: 50 CONTINUE
1305: C
1306: DO 65 L=1, MUR
1307: DO 66 K=1, NLAY
1308: 66 ZWORK(K,L+NLAY+MUR)=ZC13(K,L)-Z1223(K,L)
1309: DO 68 K=1, MUR
1310: 68 ZWORK(K+NLAY+MUR,L+NLAY+MUR)=ZC33(K,L)-Z3223(K,L)
1311: 65 CONTINUE

```

```

1312: C
1313: DO 51 L=1, NLAY*4
      DO 51 K=1, NLAY*4
1314: PRRR=REAL(ZRB(K,L))-OMG2*AIMAG(ZRB(K,L))
1315: 51 ZW(K,L)=CMLX(PRRR,0,0)
1316:      DO 52 K=1, NLAY*2
      DO 52 L=1, NLAY*2
1317: 52 ZM(K,L)=ZN(K,L)+ZM0RK(K,L)
1318:
1319:      CASE=1 FORCED VIBRATION IN R-DIRECTION AT THE TOP OF PILE
1320: C
1321: C
1322: C CALL ZINV1(ZW,NLAY*4,NLAY*4,MLY*8)
1323:
1324: C
1325: C
1326: C STIFFNESS ZXXX, ZXXT, ZKTT
1327: C
1328: CALL COEFFX(ZW,ZW1,ZW2,MLY,ZXXX,ZXXT,ZKTT)
1329: CALL ZMULT1(ZW,ZPUX,ZUN,NLAY*4,1,MLY*4,1,1)
1330: CALL ZMULT1(ZW2221,ZW1,ZW2,MUR,NLAY,1,MLY,1,1)
1331: DD 57 K=1, MUR
1332: 57 ZW3(K)=ZUW(NLAY+MAU+K)
1333: CALL ZMULT1(ZW2223,ZW3,ZW4,MUR,MUR,1,MLY,1,1)
1334: C REARRANGE ZUW VECTOR
1335: C
1336: C
1337: DO 70 K=1, MAU
1338: 70 ZW5(K)=ZUW(K)
1339: DD 71 K=1, MUR
1340: 71 ZUW(K)=ZUW(K+MAU)
1341: DD 73 K=1, MUR
1342: ZUW(K+MUR)=ZW2(K)-ZW4(K)
1343: ZUW(K+MUR*2)=ZW3(K)
1344: 73 CONTINUE
1345: DD 72 K=1, MAU
1346: 72 ZUW(K+MUR*3)=ZW5(K)
1347: DD 74 K=1, MUR-MAU
1348: 74 ZUW(K+MUR*3+MAU)=ZERO
1349: C
1350: CALL ZMULT1(ZR,ZUW,ZPUX,MJN,MJN,1,MLY*4,1,1)
1351: WRITE(WP,603),
1352: 99 CONTINUE
1353: CALL WZPA1(ZUM,ZRS,ZALP,ZPUX,ZW2,MJN,MAU)
1354: ZU1=ZUW(1)
1355: ZU21=ZUW(1+MUR)
1356: ZUT1=ZUW(1+MUR*2)*ZAI
1357: ZMR1=ZUW(1+MUR*3+MAU)
1358: ZW1=ZUW(1+MUR*3+MAU+MUR)
1359: ZWT1=ZUW(1+MUR*3+MAU+MUR*2)*ZAI
1360: CALL OUTPT(ZUM,ZALP,ZPUX,ZR,NXOUT,IND1,MAU)
1361: C
1362: WRITE(*,*), 'SOLVE COMPLETE'
1363: 600 FORMAT(1H,'NO PILE CASE=1 FORCED VIBRATION IN X DIRECTION')
1364: 603 FORMAT(1H,'PILE EXIST CASE=1 FORCED VIBRATION IN X DIRECTION')
1365: C
1366: RETURN
1367: END
1368: C ****

```

```

1369: C SUBROUTINE WPAI (ZUW, ZR5, ZALP, ZPUX, ZW2, MUN, MAU)
1370: C *****
1371: C IMPLICIT COMPLEX*16(Z)
1372: C PARAMETER (MLY=10)
1373: C PARAMETER (M6=MLY*6)
1374: C DIMENSION ZUH(*), ZRE(MLY,*), ZALP(*), ZW2(*), ZPUX(*)
1375: C COMMON /COND/ KC, KP, KG(4), NL, Y, MMN
1376: C COMMON /YD/ MJYD, MURZ, MRTH, MUR, MUZ, MUR, MUZ, MUL, MUL
1377: C COMMON /NODCH3/ ZPH1(N6, MLY*4), ZPINV(MLY*4, MLY*4)
1378: C COMMON /NODCH5/ ZPHIM(M6, MLY*4)
1379: C COMMON /CONST/ GAL, PAI, DEPTH, NOTT, VSB, NRQ, RR, D00, NCASE
1380: C ZERO=CMPLX(0, 0, 0, 0)
1381: C ZA1=CMPLX(0, 0, 1, 0)
1382: C -----
1383: C CALCULATE OF PAI
1384: C -----
1385: C CALL ZMULT1(ZR5, ZUH, ZW2, MUR-MAU, MUN, 1, MLY, 1, 1)
1386: C -----
1387: C -----
1388: C -----
1389: DO 10 N=1, MUR-MAU
1390: ZUW(N+MUN)=ZERO
1391: ZPUX(N+MUN)=ZUW(N)
1392: 10 CONTINUE
1393: CALL ZMULT1(ZPINV, ZUH, ZALP, MR1, MR1, MR1, 1, MLY*4, 1, 1)
1394: CCC WRITE(KP, *) 'VECTOR ALPHA'
1395: CCC WRITE(KP, *) 'FORMAT(ZALP, MR1, 1, KP, 1)'
1396: CCC CALL POUTM(ZALP, MR1, 1, KP, 1)
1397: C -----
1398: C -----
1399: CCC WRITE(KP, 600)
1400: CCC DO 1 N=1, MUR
1401: CCC 1 WRITE(KP, 601) N, ZUW(N), ZUW(N+MUR), ZUW(N+MUR*2)
1402: C -----
1403: C -----
1404: C -----
1405: C -----
1406: CCC CALL ZMULT1(ZPHIM, ZALP, ZUW, MJYD, MR1, 1, M6, 1, 1)
1407: CCC 2 WRITE(KP, 601) N, ZUW(MUR*3+N), ZUW(MUR*3+MUR+N), ZUW(MUR*2+N)
1408: CCC -----
1409: CCC -----
1410: CCC WRITE(KP, 602) '(ZUW(J), J=1, 6)'
1411: CCC FORMAT(1H, '***DISP. ***', 15X, 'R-DIRECTION', 19X, 'Z-DIRECTION',
1412: CC + 19X, 'THETA-DIRECTION')
1413: CCC FORMAT(1H, 15, 5X, BE15, 5)
1414: CCC FORMAT(1H, 5E15, 5)
1415: RETURN
1416: END
1417: C*****
1418: C SUBROUTINE OUTPT(ZUW, ZALP, ZPUX, ZR, NYOUT, IND1, MAU)
1419: C*****
1420: C IMPLICIT COMPLEX*16(Z)
1421: C PARAMETER (MLY=10)
1422: C DIMENSION NXOUT(*), IND1(MLY,*), CX(10), X(10)
1423: C DIMENSION ZALP(*), ZUW(*), ZPUX(*), ZR (MLY*4, *)
1424: C COMMON /DMY3/ ZSGT(MLY, 10), ZSGZ(MLY, 10), ZTURZ(MLY, 10)

```

```

1: C ****
2: C ***** SUBROUTINE CARDIM(NFT, KP)
3: C ***** CHARACTER A$60, AD*95, AINP*19
4: C ***** DATA AD/, CARD NO. -1-----2-----3-----4---*
5: C ***** -5-----6-----7-----8 / *
6: C ***** -4-----5-----6-----7-----8 / *
7: C ***** DATA AINP/, INPUT DATA ECHO / /
8: C ***** NT = 0
9: C ***** READ(NFT, 500, END=101) A
10: C ***** NT = NT + 1
11: C ***** NC=MD(INT, 50)
12: C ***** IF(NC, EQ, 1) WRITE(KP, 600) NFT, AINP, AD
13: C ***** WRITE(KP, 602) NT, A
14: C ***** IF(NC, EQ, 0) WRITE(KP, 650) AD
15: C ***** GO TO 10
16: C ***** CONTINUE
17: C ***** IF(NC, EQ, 0) WRITE(KP, 650) AD
18: C ***** WRITE(KP, 601) NT
19: C ***** REWIND NFT
20: C ***** 500 FORMAT(ABO)
21: C ***** 600 FORMAT(1H /15X, 20(' *'), FILE CODE=', I2, A19, 20(' *'), //, A95, / )
22: C ***** 601 FORMAT(1H ,30(' *'), 'DATA CARD END TOTAL CARD NUMBER = ', I5)
23: C ***** 602 FORMAT(1H ,4X, I7, ' ', ABO)
24: C ***** 650 FORMAT(//, A95)
25: C ***** RETURN
26: C ***** END
27: C ****
28: C ***** SUBROUTINE CPY4(ZWK, ZMKB, MM, C2, C3, C4, CKW)
29: C ***** IMPLICIT COMPLEX*16(Z)
30: C ***** DIMENSION ZWK(MM,*), ZMKB(MM*2,*)
31: C ***** DIMENSION CK(4)
32: C ***** ZERO=CHPLX(0, 0, 0, 0)
33: C ***** CK(1)=1. 0
34: C ***** CK(2)=C2
35: C ***** CK(3)=C3
36: C ***** CK(4)=C4
37: C ***** 38: C
39: C ***** DO 1 N=1, 4
40: C ***** C=CK(N)*CKK
41: C ***** K0=MM*MD(N-1, 2)
42: C ***** L0=(N-1)/2
43: C ***** L0=L0*MM
44: C ***** DO 2 K=1, MM
45: C ***** KK=K0+K
46: C ***** DO 2 L=1, MM
47: C ***** LL=L0+L
48: C ***** ZMKB(MM, LL)=C*ZWK(K, L)
49: C ***** 2 CONTINUE
50: C ***** 1 CONTINUE
51: C ***** DO 3 K=1, MM
52: C ***** DO 3 L=1, MM
53: C ***** ZMK(K, L)=ZERO
54: C ***** 3 CONTINUE
55: C ***** RETURN.
56: C ***** END
57: C ****

```

C ****

```

58: SUBROUTINE POUTM(ZA, N, M, KP, M2)
59: C ****
60: IMPLICIT COMPLEX*16(Z)
61: DIMENSION ZA(M2,*)
62: C
63: IP=(M-1)/10+1
64: MM=MOD(M-1,10)+1
65: WRITE(KP,600) N,M
66: C
67: C-----REAL PART-----
68: C-----WRITE(KP,601)
69: DO 1 I1=1,IP
70: 1E=10
71: IF(I1.EQ.IP) IE=MM
72: DO 2 J=1,N
73: 10=(I1-1)*10
74: 2 WRITE(KP,602) J, (REAL(ZA(J, I+10)), I=1,IE)
75: 1 CONTINUE
76: C-----IMAGINARY PART-----
77: C-----WRITE(KP,603)
78: C-----DO 3 I1=1,IP
79: C-----1E=10
80: IF(I1.EQ.IP) IE=MM
81: DO 4 J=1,N
82: 10=(I1-1)*10
83: 4 WRITE(KP,602) J, (AIMAG(ZA(J, I+10)), I=1,IE)
84: 3 CONTINUE
85: 600 FORMAT(1H,'CHECK WRITE MATRIX N= ',15.5X, 'M= ',15)
86: 601 FORMAT(1H,'-- REAL PART --')
87: 602 FORMAT(1H,'15.5X,10E12.3')
88: 603 FORMAT(1H,'-- IMAGINARY PART --')
89: RETURN
90: END
91: C ****
92: C ****
93: C ****
94: C ****
95: SUBROUTINE REARRG(ZIN, ZTEMP, MM, M2, MMOD)
96: C ****
97: IMPLICIT COMPLEX*16(Z)
98: DIMENSION ZIN(M2,*), ZTEMP(M2,*)
99: C
100: DO 1 J=1,MM
101: DO 1 N=1,MM
102: 1 ZTEMP(N,J)=ZIN(N,J)
103: N1=MM/MMOD
104: DO 10 I=1,MM
105: 1J=MOD(I-1,MMOD)
106: 1I=(I-1)/MMOD+N1*IJ+1
107: DO 20 J=1,MM
108: 1J=MOD(J-1,MMOD)
109: 1I=(J-1)/MMOD+N1*IJ+1
110: ZIN(J1, I1)=ZTEMP(J, I)
111: 20 CONTINUE
112: 10 CONTINUE
113: RETURN
114: END

```

C ****

```

115: C ****
116:      SUBROUTINE ZTRI(ZIN,M1,ZOT,MO,M2)
117: C ****
118:      IMPLICIT COMPLEX*16(Z)
119:      DIMENSION ZIN(M1,*),ZOT(M2,*)
120: C
121:      DO 1 L=1,MI
122:      LL=L+MO-1
123:      DO 2 K=1,MI
124:      KK=K+MO-1
125:      ZDT(WK,LL)=ZDT(WK,LL)+ZIN(K,L)
126:      2 CONTINUE
127:      1 CONTINUE
128:      RETURN
129: C
130: C ****
131:      SUBROUTINE ZINV1(A,N,M,NC)
132: C ****
133:      REAL*8 A(NM2,1),PA,PR,PI,WR,WI,ONE,ZERO
134:      DATA ONE,ZERO /1,D+00,0,D+00/
135: C
136:      DO 40 K=1,N
137:      PA = A(2*K-1,K)*#2+A(2*K,K)*#2
138:      PR = A(2*K-1,K)/PA
139:      PI = -A(2*K ,K)/PA
140:      A(2*K-1,K)=ONE
141:      A(2*K ,K)=ZERO
142:      DO 10 J=1,M
143:      PA = A(2*K-1,J)
144:      A(2*K-1,J) = A(2*K-1,J)*PR-A(2*K,J)*PI
145:      A(2*K ,J) =PA*PI+A(2*K,J)*PR
146:      10 CONTINUE
147:      DO 30 I=1,N
148:      IF (K.EQ.I) GO TO 30
149:      WR = -A(2*I-1,K)
150:      WI = -A(2*I ,K)
151:      A(2*I-1,K) = ZERO
152:      A(2*I ,K) = ZERO
153:      DO 20 J=1,M
154:      A(2*I-1,J) = A(2*I-1,J)+A(2*K-1,J)*WR-A(2*K,J)*WI
155:      A(2*I ,J) = A(2*I ,J)+A(2*K-1,J)*WI+A(2*K,J)*WR
156:      20 CONTINUE
157:      30 CONTINUE
158:      40 CONTINUE
159:      RETURN
160: C
161: C ****
162:      SUBROUTINE ZMULTI(ZA,ZB,ZC,M1,M2,MK,MC)
163: C ****
164:      IMPLICIT COMPLEX*16(Z)
165:      DIMENSION ZA(MA,*),ZB(MB,*),ZC(MC,*)
166: C
167:      DO 10 L=1,MK
168:      DO 20 K=1,MI
169:      Z1=CMPLX(O,O,O,O)
170:      DO 30 N=1,MJ
171:      Z2=ZA(K,N)*ZB(N,L)

```

```

C ****
172: Z3=Z1
173: Z1=Z2+Z3
174: 30 CONTINUE
175: ZC(K,L)=Z1
176: 20 CONTINUE
177: 10 CONTINUE
178: RETURN
179: END

180: C ****
181: SUBROUTINE ZTENCH(ZIN, ZEN, MM, M2)
182: C ****
183: IMPLICIT COMPLEX*16(Z)
184: DIMENSION ZIN(M2,*), ZTEN(M2,*)
185: C
186: DO 1 K=1, MM
187: DO 1 L=1, MM
188: ZTEN(K,L)=ZIN(L,K)
189: 1 CONTINUE
190: RETURN
191: END

192: C ****
193: SUBROUTINE ZTM1(ZIN, MM, MA)
194: C ****
195: IMPLICIT COMPLEX*16(Z)
196: DIMENSION ZIN(MA,*)
197: C
198: DO 1 K=1, MM-1
199: L1=K+1
200: DO 2 L=L1, MM
201: ZIN(K,L)=ZIN(L,K)
202: 2 CONTINUE
203: 1 CONTINUE
204: 1 CONTINUE
205: RETURN
206: END

207: C ****
208: SUBROUTINE ZEROL(ZA, NM, NDIM)
209: C ****
210: IMPLICIT COMPLEX*16(Z)
211: DIMENSION ZA(NDIM,*)
212: ZERO=CMPLX(0.0, 0.0)
213: DO 1 K=1, N
214: DO 1 L=1, N
215: ZA(K,L)=ZERO
216: 1 CONTINUE
217: END

218: C ****
219: SUBROUTINE CG(NM, NR, AI, WR, WI, MATZ, ZR, ZI, FV1, FV2, FV3, IERR)
220: C ****
221: INTEGER NM, IS1, IS2, IERR, MATZ
222: REAL*8 AR(NM,N), AI(NM,N), WR(N), WI(N), ZR(NM,N), ZI(NM,N),
223:      FV1(N), FV2(N), FV3(N)
224: C ****
225: C This subroutine calls the recommended sequence of
226: C subroutines from the eigensystem subroutine package (EISPACK)
227: C to find the eigenvalues and eigenvectors (if desired)
228: C

```

C ****

229: C of a complex general matrix.
230: C
231: C On input:
232: C NM must be set to the row dimension of the two-dimensional
array parameters as declared in the calling program
dimension statement;
233: C
234: C
235: C
236: C N is the order of the matrix A=(AR,AI);
237: C AR and AI contain the real and imaginary parts,
238: C respectively, of the complex general matrix;
239: C
240: C MATZ is an integer variable set equal to zero if
241: C only eigenvalues are desired; otherwise it is set to
242: C any non-zero integer for both eigenvalues and eigenvectors.
243: C
244: C
245: C
246: C On output:
247: C WR and WI contain the real and imaginary parts,
248: C respectively, of the eigenvalues;
249: C ZR and ZI contain the real and imaginary parts,
250: C respectively, of the eigenvectors if MATZ is not zero;
251: C
252: C
253: C IERR is an integer output variable set equal to an
error completion code described in section 2B of the
documentation. The normal completion code is zero;
254: C
255: C FV1, FV2, and FV3 are temporary storage arrays.
256: C
257: C Questions and comments should be directed to B. S. Garbow,
258: C Applied Mathematics division, Argonne National Laboratory
259: C
260: C
261: C
262: C
263: C
264: C
265: IF (N .LE. NM) GO TO 10
266: IERR = 10.*N
267: GO TO 50
268: C
269: 10 CALL CBAL(NM,N,AR,AI,IS1,IS2,FV1)
270: CALL CORTH(NM,N,IS1,IS2,AR,AI,FV2,FV3)
271: IF (MATZ .NE. 0) GO TO 20
272: C : find eigenvalues only :::::::
273: CALL COMQR(NM,N,IS1,IS2,AR,AI,WR,WI,IERR)
274: GO TO 50
275: C ::::: find both eigenvalues and eigenvectors :::::::
276: CALL COMQR2(NM,N,IS1,IS2,FV2,FV3,AR,AI,WR,WI,ZR,ZI,IERR)
277: IF (IERR .NE. 0) GO TO 50
278: CALL CBAK2(NM,N,IS1,IS2,FV1,N,ZR,ZI)
279: 50 RETURN
280: C ::::: last card of CG :::::::
281: END
282: C*****
283: SUBROUTINE CBAK2 (NM, N, LOW, IGH, SCALE, M, ZR, ZI)
284: C-----
285: INTEGER I, J, K, M, N, II, NM, IGH, LOW

C ****

```
286: REAL*B SCALE(N),ZR(NM,M),ZI(NM,M)
 287: REAL*B S
 288: C
 289: C This subroutine is a translation of the algol procedure
 290: C CBALK2, which is a complex version of BALBAK,
 291: C Num. Math. 13, 293-304(1967) by Palett and Reinsch.
 292: C Handbook for auto. comp., Vol.II-Linear Algebra, 315-326(1971).
 293: C
 294: C This subroutine forms the eigenvectors of a complex general
 295: C matrix by back transforming those of the corresponding
 296: C balanced matrix determined by CBAL.
 297: C
 298: C
 299: C On input:
 300: C NM must be set to the row dimension of two-dimensional
 301: C array parameters as declared in the calling program
 302: C
 303: C
 304: C N is the order of the matrix;
 305: C
 306: C LOW and IGH are integers determined by CBAL;
 307: C
 308: C SCALE contains information determining the permutations
 309: C and scaling factors used by CBAL;
 310: C
 311: C M is the number of eigenvectors to be transformed;
 312: C
 313: C ZR and ZI contain the real and imaginary parts,
 314: C respectively, of the eigenvectors to be
 315: C back transformed in their first M columns.
 316: C
 317: C
 318: C
 319: C On output:
 320: C ZR and ZI contain the real and imaginary parts,
 321: C respectively, of the transformed eigenvectors
 322: C in their first M columns.
 323: C
 324: C Questions and comments should be directed to B. S. Garbow,
 325: C Applied Mathematics Division, Argonne National Laboratory
 326: C
 327: C
 328: IF (M .EQ. 0) GO TO 200
 329: IF (IGH .EQ. LOW) GO TO 120
 330: C
 331: DO 110 I = LOW, IGH
 332: C
 333: C S = SCALE(I)
 334: C ..... left hand eigenvectors are back transformed
 335: C if the foregoing statement is replaced by
 336: C      S=1.0D0/SCALE(I).
 337: C      DO 100 J = 1, M
 338: C      ZR(I,J) = ZR(I,J) * S
 339: C      ZI(I,J) = ZI(I,J) * S
 340: C      100 CONTINUE
 341: 110 CONTINUE
 342: C ..... FOR I=LOW-1 STEP -1 UNTIL 1,
```

C ****

```

343: C   120 DO 140 I1 = 1, N
344:   I = I1
345:   IF (I .GE. LOW .AND. I .LE. IGH) GO TO 140
346:   IF (I .LT. LOW) I = LOW - 11
347:   K = SCALE(I)
348:   IF (K .EQ. 1) GO TO 140
350: C
351:   DO 130 J = 1, M
352:   S = ZR(I,J)
353:   ZR(I,J) = ZR(K,J)
354:   ZR(K,J) = S
355:   S = ZI(I,J)
356:   ZI(I,J) = ZI(K,J)
357:   ZI(K,J) = S
358:   130  CONTINUE
359: C
360:   140  CONTINUE
361: C
362:   200 RETURN
363: C   **** last card of CBALK2 *****
364: C
365: C***** SUBROUTINE CBAL(NM,N,AR,AI,LOW,IGH,SCALE)
366: C
367: C
368:   INTEGER I, J, K, L, M, N, JJ, NM, IGH, LOW, IEXC
369:   REAL*B AR(NM,N),AI(NM,N),SCALE(N)
370:   REAL*B C, F, G, R, S, B2, RADIX
371:   REAL*B DABS
372:   LOGICAL NOCDNV
373: C
374: C
375: C This subroutine is a translation of the ALGOL procedure
376: C CBALANCE, which is a complex version of BALANCE,
377: C Num. Math. 13, 293-304 (1969)
378: C Handbook for Auto. Comp., Vol. II-Linear Algebra, 315-326 (1971).
379: C
380: C This subroutine balances a complex matrix and isolates
381: C eigenvalues whenever possible.
382: C
383: C On input:
384: C
385: C NM must be set to the row dimension of two-dimensional
386: C array parameters as declared in the calling program
387: C DIMENSION statement;
388: C
389: C N is the order of the matrix;
390: C
391: C AR and AI contain the real and imaginary parts,
392: C respectively, of the complex matrix to be balanced.
393: C
394: C On output:
395: C
396: C AR and AI contain the real and imaginary parts,
397: C respectively, of the balanced matrix;
398: C
399: C LOW and IGH are two integers such that AR(I,J) and AI(I,J)

```

```

1   400: C      are equal to zero if
1   401: C      (1) I is greater than J and
1   402: C      (2) J=1,...,LOW-1 or I=IGH+1,...,N;
1   403: C
1   404: C      SCALE contains information determining the
1   405: C      permutations and scaling factors used.
1
1   406: C
1   407: C      Suppose that the principal submatrix in rows LOW through IGH
1   408: C      has been balanced, that P(J) denotes the index interchanged
1   409: C      with J during the permutation step, and that the elements
1   410: C      of the diagonal matrix used are denoted by D(I,J). Then
1   411: C      SCALE(J) = P(J),      FOR J = 1,...,LOW-1
1   412: C      = D(J,J),        FOR J = LOW,...,IGH
1   413: C      = P(J) / N,      J = IGH+1,...,N.
1
1   414: C      The order in which the interchanges are made is N to IGH+1,
1   415: C      then 1 to LOW-1.
1
1   416: C
1   417: C      Note that I is returned for IGH if IGH is zero formally.
1
1   418: C
1   419: C      The ALGOL procedure EXC contained in cbalance appears in
1   420: C      CBAL in line. (Note that the ALGOL roles of identifiers
1   421: C      K, L have been reversed.)
1
1   422: C
1   423: C      Arithmetic is real throughout.
1
1   424: C      Questions and comments should be directed to B. S. Garbow,
1   425: C      Applied Mathematics Division, Argonne National Laboratory
1   426: C
1   427: C
1
1   428: C
1   429: C
1   430: C      :::::::::: RADIX is a machine dependent parameter specifying
1   431: C      the base of the machine floating point representation.
1
1   432: C      RADIX = 16.000 for long form arithmetic
1
1   433: C      DATA RADIX/2/
1
1   434: C
1   435: C
1
1   436: C      B2 = RADIX * RADIX
1
1   437: C      K = 1
1   438: C      L = N
1   439: C      GO TO 100
1
1   440: C      :::::::::: In-line procedure for row and
1   441: C      column exchange ::::::::::::
1
1   442: C      20 SCALE(M) = J
1   443: C      IF (J .EQ. M) GO TO 50
1
1   444: C      DO 30 I = 1, L
1   445: C      F = AR(I,J)
1   446: C      AR(I,J) = AR(I,M)
1   447: C      AR(I,M) = F
1
1   448: C      F = AI(I,J)
1   449: C      AI(I,J) = AI(I,M)
1   450: C      AI(I,M) = F
1
1   451: C      30 CONTINUE
1
1   452: C
1   453: C      DD 40'1 = K, N
1   454: C      F = AR(J,I)
1   455: C      AR(J,I) = AR(M,I)
1   456: C

```



```
457:      ARI(M,I) = F
458:      F = AI(J,I)
459:      AI(J,I) = AI(M,I)
460:      AI(M,I) = F
461:      40 CONTINUE
462: C   50 GO TO 60,130, IEXC
463: C   ::::::::::: Search for rows isolating an eigenvalue
464: C   ::::::::::: and push them down ::::::::::::
465: C   80 IF (L .EQ. 1) GO TO 280
466: C   80 IF (L .EQ. 1) GO TO 280
467: C   L = L - 1
468: C   ::::::::::: FOR J=L STEP -1 UNTIL 1 DO ---
469: C   100 DO 120 JJ = 1, L
470: C   J = L + 1 - JJ
471: C
472: C   DO 110 I = 1, L
473: C   IF (I .EQ. J) GO TO 110
474: C   IF (ARI(J,I) .NE. 0.0D0 .OR. AI(J,I) .NE. 0.0D0) GO TO 120
475: C   110 CONTINUE
476: C
477: C   M = L
478: C   IEXC = 1
479: C   GO TO 20
480: C   120 CONTINUE
481: C
482: C   90 TO 140
483: C   ::::::::::: Search for columns isolating an eigenvalue
484: C   ::::::::::: and push them left ::::::::::::
485: C   130 K = K + 1
486: C
487: C   140 DO 170 J = K, L
488: C
489: C   DO 150 I = K, L
490: C   IF (I .EQ. J) GO TO 150
491: C   IF (ARI(I,J) .NE. 0.0D0 .OR. AI(I,J) .NE. 0.0D0) GO TO 170
492: C   150 CONTINUE
493: C
494: C   M = K
495: C   IEXC = 2
496: C   GO TO 20
497: C   170 CONTINUE
498: C
499: C   ::::::::::: Now balance the submatrix in rows K to L ::::::::::::
500: C   180 SCALB(I) = 1.0D0
501: C
502: C   190 NOCONV = FALSE.
503: C
504: C   DO 270 I = K, L
505: C   C = 0.0D0
506: C   R = 0.0D0
507: C
508: C   DO 200 J = K, L
509: C   IF (J .EQ. I) GO TO 200
510: C   C = C + DABS(ARI(J,I)) + DABS(AI(J,I))
511: C   R = R + DABS(ARI(I,J)) + DABS(AI(I,J))
512: C   200 CONTINUE
513: C   ::::::::::: guard against zero C or R due to underflow ::::::::::::
```

```

514:      IF (C .EQ. 0.0D0 .OR. R .EQ. 0.0D0) GO TO 270
515:      Q = R / RADIX
516:      F = 1.0D0
517:      S = C + R
518:      210   IF (C .GE. 0) GO TO 220
519:      F = F * RADIX
520:      C = C * B2
521:      GO TO 210
522:      220   Q = R * RADIX
523:      IF (C .LT. 0) GO TO 240
524:      F = F / RADIX
525:      C = C / B2
526:      GO TO 230
527:      C      ::::: Now balance ::::::::::::
528:      240   IF ((C + R) / F .GE. 0.95D0 * S) GO TO 270
529:      G = 1.0D0 / F
530:      SCALE(I) = SCALE(I) * F
531:      NOCDNV = .TRUE.
532:      C
533:      DO 250 J = K, N
534:      AR(I,J) = AR(I,J) * G
535:      AI(I,J) = AI(I,J) * G
536:      250  CONTINUE
537:      C
538:      DO 260 J = 1, L
539:      AR(J,I) = AR(J,I) * F
540:      AI(J,I) = AI(J,I) * F
541:      260  CONTINUE
542:      C
543:      270  CONTINUE
544:      C
545:      IF (INDCONV) GO TO 190
546:      C
547:      280  LOW = K
548:      IGH = L
549:      RETURN
550:      C
551:      END
552:      C*****SUBROUTINE COMGR(NM, NLW, IGH, HR, HI, WR, WI, TERR)
553:      C
554:      C----- INTEGER J, J, L, N, EN, LL, NM, IGH, ITS, LOW, LP1, ENM1, TERR
555:      C----- REAL*B HR(NM,N), HI(NM,N), WR(N), WI(N)
556:      C----- REAL*B SI, SR, TI, TR, XI, XR, VI, YR, ZI, ZR, NORM, MACHEP
557:      C----- REAL*B DSORT, CDABS, DABS
558:      C----- INTEGER NMNO
559:      C----- COMPLEX#16 Z3
560:      C----- COMPLEX#16 CDSORT, DCMPXL
561:      C----- REAL*B DREAL, DIMAG
562:      C----- Statement functions enable extraction of real and
563:      C----- imaginary parts of double precision complex numbers ::::::::::::
564:      C----- DREAL(Z3) = Z3
565:      C----- DIMAG(Z3) = (0.0D0, -1.0D0) * Z3
566:      C
567:      C
568:      C This subroutine is a translation of a unitary analogue of the
569:      C ALGOL procedure COMLR, Num. Math. 12, 369-376(1968) by Martin
570:      C and Wilkinson.

```

C ****

571:C Handbook for Auto. Comp., Vol II-Linear Algebra, 396-403(1971).
 572:C The unitary analogue substitutes the QR algorithm of Francis.
 573:C (Comp. Jour. 4, 332-345(1962) for the LR Algorithm.
 574:C
 575:C This subroutine finds the eigenvalues of a complex
 576:C upper Hessenberg matrix by the QR method.
 577:C
 578:C On input:
 579:C
 580:C NM must be set to the row dimension of two-dimensional
 581:C array parameters as declared in the calling program
 582:C dimension statement;
 583:C
 584:C N is the order of the matrix;
 585:C
 586:C LOW and IGH are integers determined by the balancing
 587:C subroutine CBAL. If CBAL has not been used,
 588:C set LOW=1, IGH=N.
 589:C
 590:C HR and HI contain the real and imaginary parts,
 591:C respectively, of the complex upper Hessenberg matrix.
 592:C Their lower triangles below the subdiagonal contain
 593:C information about the unitary transformations used in
 594:C the reduction by CORTH, if performed.
 595:C
 596:C On output:
 597:C
 598:C The upper Hessenberg portions of HR and HI have been
 599:C destroyed. Therefore, they must be saved before
 600:C calling COMQR if subsequent calculation of
 601:C eigenvectors is to be performed;
 602:C
 603:C WR and WI contain the real and imaginary parts,
 604:C respectively, of the eigenvalues. If an error
 605:C exit is made, the eigenvalues should be correct
 606:C for indices IERR+1,...,N.
 607:C
 608:C IERR is set to
 609:C ZERO for normal return,
 610:C J if the J-th eigenvalue has not been
 611:C determined after 30 iterations.
 612:C
 613:C arithmetic is real except for the replacement of the ALGOL
 614:C procedure CDIV by complex division and use of the subroutines
 615:C CDSQRT and DCMPXL in computing complex square roots.
 616:C
 617:C Questions and comments should be directed to B. S. Garbow,
 618:C Applied Mathematics Division, Argonne National Laboratory
 619:C
 620:C
 621:C
 622:C MACHEP is a machine dependent parameter specifying
 623:C the relative precision of floating point arithmetic.
 624:C MACHEP = 16.0D0*(-13) for long form arithmetic
 625:C on S360 : : : : :
 626:C DATA MACHEP/1.421D-14/
 627:C

C ****

```

628:      IERR = 0
629:      IF (LOW .EQ. IGH) GO TO 180
630:      C      ::::::::::::::: create real subdiagonal elements ::::::::::::
631:      L = LOW + 1
632:      C
633:      DO 170 I = L, IGH
634:        LL = MIN(I,I+1), IGH)
635:        IF (HI(I,I-1) .EQ. 0.0D0) GO TO 170
636:        NORM = CDABS(DCMPLX(HR(I,I-1), HI(I,I-1)))
637:        YR = HR(I,I-1) / NORM
638:        VI = HI(I,I-1) / NORM
639:        HR(I,I-1) = NORM
640:        HI(I,I-1) = 0.0D0
641:      C
642:      DO 155 J = I, IGH
643:        SI = YR * HI(I,J) - VI * HR(I,J)
644:        HR(I,J) = YR * HR(I,J) + VI * HI(I,J)
645:        HI(I,J) = SI
646:      155    CONTINUE
647:      C
648:      DO 160 J = LOW, LL
649:        SI = YR * HI(J,I) + VI * HR(J,I)
650:        HR(J,I) = YR * HR(J,I) - VI * HI(J,I)
651:        HI(J,I) = SI
652:      160    CONTINUE
653:      C
654:      170 CONTINUE
655:      C      ::::::::::::::: Store root isolated by CBAL ::::::::::::
656:      180 DO 200 I = 1, N
657:        IF (I .GE. LOW .AND. I .LE. IGH) GO TO 200
658:        HR(I,I) = HR(I,I)
659:        WI(I) = HI(I,I)
660:      200 CONTINUE
661:      C
662:        EN = IGH
663:        TR = 0.0D0
664:        TI = 0.0D0
665:      C      ::::::::::::::: Search for next eigenvalue ::::::::::::
666:      220 IF (EN .LT. LOW) GO TO 1001
667:      ITS = 0
668:      ENM1 = EN - 1
669:      C      ::::::::::::::: Look for single small sub-diagonal element ::::::::::::
670:      FOR L=EN STEP -1 UNTIL LOW DO -- ::::::::::::
671:      240 DO 260 LL = LOW, EN
672:        L = EN + LOW - LL
673:        IF (L .EQ. LOW) GO TO 300
674:        IF (DABS(HR(L,L-1)) .LE.
675:          MACHEP * (DABS(HR(L-1,L-1)) + DABS(HI(L-1,L-1))
676:          X + DABS(HR(L,L)) + DABS(HI(L,L))) GO TO 300
677:      260 CONTINUE
678:      C      ::::::::::::::: Form shift ::::::::::::
679:      300 IF (L .EQ. EN) GO TO 660
680:      IF (ITS .EQ. 30) GO TO 1000
681:      IF (ITS .EQ. 10 .OR. ITS .EQ. 20) GO TO 320
682:      SR = HR(EN,EN)
683:      SI = HI(EN,EN)
684:      XR = HR(ENM1,EN) * HR(EN,ENM1)

```

```

685: XI = HI(ENM1, EN) * HR(EN, ENM1)
686: IF (XR .EQ. 0. ODO .AND. XI .EQ. 0. ODO) GO TO 340
687: YR = (HR(ENM1, ENM1) - SR) / 2. ODO
688: YI = (HI(ENM1, ENM1) - SI) / 2. ODO
689: Z3 = CD SORT(DCMPLX(YR**2-YI**2+XR, 2. ODO*YR*YI+XI))
690: ZZR = DREAL(Z3)
691: Z2I = DIMAG(Z3)
692: IF (YR * ZZR + YI * Z2I .GE. 0. ODO) GO TO 310
693: ZZR = -ZZR
694: Z2I = -Z2I
695: 310 Z3 = DCMPLX(XR, XI) / DCMPLX(YR+ZZR, YI+Z2I)
696: SR = SR - DREAL(Z3)
697: SI = SI - DIMAG(Z3)
698: GO TO 340
699: C :::::::::: Form exceptional shift ::::::::::::
700: 320 SR = DABS(HR(EN, ENM1)) + DABS(HR(ENM1, EN-2))
701: SI = 0. ODO
702: C
703: 340 DO 360 I = LOW, EN
704:      HR(I, 1) = HR(I, I) - SR
705:      HI(I, 1) = HI(I, I) - SI
706: 360 CONTINUE
707: C
708: TR = TR + SR
709: TI = TI + SI
710: ITS = 1ITS + 1
711: C :::::::::: Reduce to triangle (rows) ::::::::::::
712: LP1 = L + 1
713: C
714: DO 500 I = LP1, EN
715:      SR = HR(I, I-1)
716:      HR(I, I-1) = 0. ODO
717:      NORM = DSGRT(HR(I-1, I-1)*HR(I-1, I-1)+HI(I-1, I-1),
718:                      +SR*SR),
719:      XR = HR(I-1, I-1) / NORM
720:      XI = HI(I-1, I-1) / NORM
721:      XI = HI(I-1, I-1) / NORM
722:      HI(I-1, I-1) = XI
723:      HR(I-1, I-1) = NORM
724:      HI(I-1, I-1) = 0. ODO
725:      HI(I, I-1) = SR / NORM
726: C
727:      DO 490 J = I, EN
728:          YR = HR(I-1, J)
729:          YI = HI(I-1, J)
730:          ZZR = HR(I, J)
731:          Z2I = HI(I, J)
732:          HR(I-1, J) = XR * YR + XI * YI + HI(I, I-1) * ZZR
733:          HI(I-1, J) = XR * YI - XI * YR + HI(I, I-1) * Z2I
734:          HR(I, J) = XR * ZZR - XI * Z2I - HI(I, I-1) * YR
735:          HI(I, J) = XR * Z2I + XI * ZZR - HI(I, I-1) * YI
736: 490 CONTINUE
737: C
738: 500 CONTINUE
739: C
740:      SI = HI(EN, EN)
741:      IF (SI .EQ. 0. ODO) GO TO 540

```

C ****

```

742: NORM = CDABS(DCMPLX(HR(EN, EN), SI))
743: SR = HR(EN, EN) / NORM
744: SI = SI / NORM
745: HR(EN, EN) = NORM
746: HI(EN, EN) = 0.0D0
747: C : : : : : Inverse operation (columns)
748: 540 DO 600 J = LP1, EN
749:     XR = WR(J-1)
750:     XI = WI(J-1)
751: C
752:     DD 580 I = L, J
753:         YR = HR(I, J-1)
754:         YI = 0.0D0
755:         ZZR = HR(I, J)
756:         ZZI = HI(I, J)
757:         IF (I .EQ. J) GO TO 560
758:         YI = HI(I, J-1)
759:         HI(I, J-1) = XR * YI + XI * YR + HI(J, J-1) * ZZI
760:         HR(I, J-1) = XR * YR - XI * YI + HI(J, J-1) * ZZR
761:         HR(I, J) = XR * ZZR + XI * ZZI - HI(J, J-1) * YR
762:         HI(I, J) = XR * ZZI - XI * ZZR - HI(J, J-1) * YI
763:         CONTINUE
764: C
765: 600 CONTINUE
766: C
767: IF (SI .EQ. 0.0D0) GO TO 240
768: C
769: DO 630 I = L, EN
770:     YR = HR(I, EN)
771:     VI = HI(I, EN)
772:     HR(I, EN) = SR * YR - SI * YI
773:     HI(I, EN) = SR * VI + SI * YR
774:     630 CONTINUE
775: C
776: 660 GO TO 240
777: C : : : : : A root found
778:     660 WR(EN) = HR(EN, EN) + TR
779:     WI(EN) = HI(EN, EN) + TI
780:     EN = ENM1
781: 660 GO TO 220
782: C : : : : : Set error -- no convergence to an
783: C : : : : : eigenvalue after 30 iterations
784: C 1000 TERR = EN
785: 1001 RETURN
786: C : : : : : last card of COMQR
787: END
788: C ****
789: SUBROUTINE COMQR2(NM, N, LDW, IGH, ORTR, ORTI, HR, HI, WR, WI, ZR, ZI, IERR)
790: C
791: INTEGER I, J, K, L, M, N, EN, II, JJ, LL, NM, NN, IGH, IP1,
792:      ITS, LOW, LP1, ENM1, IEND, IERR
793:      REAL*8 HR(NM, NM), HI(NM, NM), WR(N), WI(N), ZR(NM, NM), ZI(NM, NM),
794:      ORTR(IGH), ORTI(IGH)
795:      REAL*8 SI, SR, TI, TR, XI, XR, YI, YR, ZZI, ZZR, NORM, MACHEP
796:      REAL*8 DSGRT, CDABS, DABS
797:      INTEGER MINO
798:      COMPLEX*16 Z3

```

C ****

```

799: COMPLEX*16 CDSQRT,DCMPLX
800: REAL*8 DREAL,DIMAG
801: C : Statement functions enable extraction of real and
802: C imaginary parts of double precision complex numbers
803: C DREAL(Z3) = Z3
804: DIMAG(Z3) = (0.0D0,-1.0D0)*Z3

805: C This subroutine is a translation of a unitary analogue of the
806: C ALGOL procedure CDMLR2, Num. Math. 16, 181-204(1970) by Peters
807: C and Wilkinson.
808: C Handbook for Auto. Comp., Vol III-Linear Algebra, 372-395(1971).
809: C The unitary analogue substitutes the QR algorithm of Francis
810: C (Comp. Jour. 4, 332-345(1962)) for the LR algorithm.
811: C
812: C
813: C This subroutine finds the eigenvalues and eigenvectors
814: C of a complex upper Hessenberg matrix by the QR
815: C method. The eigenvectors of a complex general matrix
816: C can also be found if CDRTH has been used to reduce
817: C this general matrix to Hessenberg form.
818: C
819: C On input:
820: C NM must be set to the row dimension of two-dimensional
821: C array parameters as declared in the calling program.
822: C
823: C DIMENSION statement;
824: C
825: C N is the order of the matrix;
826: C
827: C LQW and IGH are integers determined by the balancing
828: C subroutine CBAL. If CBAL has not been used,
829: C set LQW=1, IGH=N;
830: C
831: C ORTR and ORTI contain information about the unitary trans-
832: C formations used in the reduction by CDRTH, if performed.
833: C Only elements LQW through IGH are used. If the eigenvectors
834: C of the hessenberg matrix are desired, set ORTR(J) and
835: C ORTI(J) to 0.0D0 for these elements;
836: C
837: C HR and HI contain the real and imaginary parts,
838: C respectively, of the complex upper Hessenberg matrix.
839: C Their lower triangles below the subdiagonal contain further
840: C information about the transformations which were used in the
841: C reduction by CDRTH, if performed.
842: C
843: C
844: C On output:
845: C DRTR, ORTL, and the upper Hessenberg portions of HR and HI
846: C have been destroyed;
847: C
848: C
849: C WR and WI contain the real and imaginary parts,
850: C respectively, of the eigenvalues. If an error
851: C exit is made, the eigenvalues should be correct
852: C for indices JERR+1,...,N,
853: C
854: C
855: C ZR and ZI contain the real and imaginary parts,
```

```

856: C
857: C respectively, of the eigenvectors. The eigenvectors
858: C are unnormalized. If an error exit is made, none of
859: C the eigenvectors has been found.
860: C
861: C IERR is set to
862: C zero for normal return,
863: C if the J-th eigenvalue has not been
864: C determined after 30 iterations.
865: C
866: C Arithmetic is real except for the replacement of the ALGOL_
867: C procedure CDIV by complex division and use of the subroutines
868: C CDSQRT and DCMPLX in computing complex square roots.
869: C
870: C Questions and comments should be directed to B. S. Garbow,
871: C Applied Mathematics Division, Argonne National Laboratory.
872: C
873: C
874: C ..... MACHEP is a machine dependent parameter specifying
875: C the relative precision of floating point arithmetic.
876: C MACHEP = 16. ODO**(-13) for long form arithmetic
877: C
878: C DATA MACHEP/1.421D-14/
879: C
880: C IERR = 0
881: C ..... Initialize eigenvector matrix ::::::::::::
882: C DO 100 I = 1, N
883: C
884: C DO 100 J = 1, N
885: C ZR(I,J) = 0.0D0
886: C ZI(I,J) = 0.0D0
887: C IF (I .EQ. J) ZR(I,J) = 1.0D0
888: C
889: C 100 CONTINUE
890: C ..... Form the matrix of accumulated transformations
891: C from the information left by CORTH ::::::::::::
892: C IEND = IGH - LDW - 1
893: C IF (IEND) 160, 150, 105
894: C 160 DO 140 II = 1, IEND
895: C 140 I = IGH - II
896: C IF (ORTRI(I)) .EQ. 0.0D0 AND ORTI(I) .EQ. 0.0D0 GO TO 140
897: C IF (ORTRI(I-1)) .EQ. 0.0D0 AND HI(I,I-1) .EQ. 0.0D0 GO TO 140
898: C ..... NORM is negative of H formed in CORTH ::::::::::::
899: C NORM = HR(I,I-1) * ORTR(I) + HI(I,I-1) * ORTI(I)
900: C IP1 = I + 1
901: C
902: C DO 110 K = IP1, IGH
903: C ORTRIK = HR(K,I-1)
904: C ORTIK = HI(K,I-1)
905: C 110 CONTINUE
906: C
907: C DO 130 J = 1, IGH
908: C SR = 0.0D0
909: C SI = 0.0D0
910: C
911: C DO 115 K = I, IGH
912: C SR = SR + ORTR(K) * ZR(K,J) + ORTI(K) * ZI(K,J)

```

C ****

```

913:    SI = SI + ORTR(K) * ZI(K,J) - ORTI(K) * ZR(K,J)
914:    CONTINUE
915: C
916:    SR = SR / NORM
917:    SI = SI / NORM
918: C
919:    DO 120 K = I, IGH
920:      ZR(K,J) = ZR(K,J) + SR * ORTR(K) - SI * ORTI(K)
921:      ZI(K,J) = ZI(K,J) + SR * ORTI(K) + SI * ORTR(K)
922:    CONTINUE
923: C
924:    130  CONTINUE
925: C
926:    140  CONTINUE   Create real subdiagonal elements ::::::::::::
927: C
928:    150  L = LOW + 1
929: C
930:    DO 170 I = L, IGH
931:      LL = MINO(I+1,IGH)
932:      IF (HI(I,I-1) .EQ. 0.0D0) GO TO 170
933:      NORM = CDABS(DCMPLX(HR(I,I-1),HI(I,I-1)))
934:      YR = HR(I,I-1) / NORM
935:      YI = HI(I,I-1) / NORM
936:      HR(I,I-1) = NORM
937:      HI(I,I-1) = 0.0D0
938: C
939:    DD 155 J = I, N
940:      SI = YR * HI(I,J) - YI * HR(I,J)
941:      HR(I,J) = YR * HR(I,J) + YI * HI(I,J)
942:      HI(I,J) = SI
943:    155  CONTINUE
944: C
945:    DO 160 J = I, LL
946:      SI = YR * HI(J,I) + YI * HR(J,I)
947:      HR(J,I) = YR * HR(J,I) - YI * HI(J,I)
948:      HI(J,I) = SI
949:    160  CONTINUE
950: C
951:    DO 165 J = LOW, IGH
952:      SI = YR * ZI(J,I) + YI * ZR(J,I)
953:      ZR(J,I) = YR * ZR(J,I) - YI * ZI(J,I)
954:      ZI(J,I) = SI
955:    165  CONTINUE
956: C
957:    170  CONTINUE   Store roots isolated by CBAL ::::::::::::
958: C
959:    180  DO 200 I = 1, N
960:      IF (I .GE. LOW .AND. I .LE. IGH) GO TO 200
961:      HR(I) = HR(I,I)
962:      WI(I) = HI(I,I)
963:    200  CONTINUE
964: C
965:    EN = IGH
966:    TR = 0.0D0
967:    TI = 0.0D0
968: C   Search for next eigenvalue ::::::::::::
969:    220  IF (EN .LT. LOW) GO TO 680

```

C ****

```

970: ITS = 0
971: ENM1 = EN - 1
972: C      ::::::: Look for single small sub-diagonal element
973: C      ::::::: for L=EN STEP -1 UNTIL LDW DO -- :::::::
974: 240 DO 260 LL = LDW, EN
975:      L = EN + LDW - LL
976:      IF (L .EQ. LDW) GO TO 300
977:      IF (DABS(HR(L,L-1)) .LE.
978:           X      MACHEP * (DABS(HR(L-1,L-1)) + DABS(HI(L-1,L-1)))
979:           X      + DABS(HR(L,L))) GO TO 300
980: 260 CONTINUE
981: C      ::::::: Form shift :::::::
982: 300 IF (L .EQ. EN) GO TO 600
983:      IF (ITS .EQ. 30) GO TO 1000
984:      IF (ITS .EQ. 10 .OR. ITS .EQ. 20) GO TO 320
985:      SR = HR(EN,EN)
986:      SI = HI(EN,EN)
987:      XR = HR(ENM1,EN) * HR(EN,ENM1)
988:      XI = HI(ENM1,EN) * HR(EN,ENM1)
989:      IF (XR .EQ. 0.0D0 .AND. XI .EQ. 0.0D0) GO TO 340
990:      YR = (HR(ENM1,ENM1) - SR) / 2.0D0
991:      YI = (HI(ENM1,ENM1) - SI) / 2.0D0
992:      Z3 = DC SORT(DCMPLX(YR*Y2-Y1**2+IR, 2.0D0*YR*Y1+XI))
993:      ZZR = DREAL(Z3)
994:      Z2I = DIMAG(Z3)
995:      IF (YR * ZZR + YI * Z2I .GE. 0.0D0) GO TO 310
996:      ZZR = -ZZR
997:      Z2I = -Z2I
998: 310 Z3 = DCMPLX(XR,XI) / DCMP LX(YR+ZZR, YI+Z2I)
999:      SR = SR - DREAL(Z3)
1000:     SI = SI - DIMAG(Z3)
1001: GO TO 340
1002: C      ::::::: Form exceptional shift :::::::
1003: 320 SR = DABS(HR(EN,ENM1)) + DABS(HR(ENM1,EN-2))
1004: SI = 0.0D0
1005: C
1006: 340 DO 360 I = LDW, EN
1007:      HR(I,I) = HR(I,I) - SR
1008:      HI(I,I) = HI(I,I) - SI
1009: 360 CONTINUE
1010: C
1011: TR = TR + SR
1012: TI = TI + SI
1013: ITS = ITS + 1
1014: C      ::::::: Reduce to triangle (rows) :::::::
1015: LP1 = L + 1
1016: C
1017: DO 500 I = LP1, EN
1018:     SR = HR(I,I-1)
1019:     HR(I,I-1) = 0.0D0
1020:     NORM = DSQRT(HR(I-1,I-1)*HR(I-1,I-1)+HI(I-1,I-1)*HI(I-1,I-1))
1021:     X      SR*SR
1022:     XR = HR(I-1,I-1) / NORM
1023:     HR(I,I-1) = XR
1024:     XI = HI(I-1,I-1) / NORM
1025:     HI(I-1,I-1) = XI
1026:     HR(I-1,I-1) = NORM

```

```

1027:      HI(I-1, J-1) = 0.0D0
1028:      HI(I, J-1) = SR / NORM
1029: C      DO 490 J = I, N
1030:      YR = HR(I-1, J)
1031:      YI = HI(I-1, J)
1032:      ZZR = HR(I, J)
1033:      Z2I = HI(I, J)
1034:      HR(I-1, J) = XR * YR + XI * YI + HI(I, J-1) * ZZR
1035:      HI(I-1, J) = XR * YI - XI * YR + HI(I, J-1) * Z2I
1036:      HR(I, J) = XR * ZZR - XI * Z2I - HI(I, J-1) * YR
1037:      HI(I, J) = XR * Z2I + XI * ZZR - HI(I, J-1) * YI
1038: 490      CONTINUE
1039:      490      CONTINUE
1040: C      500 CONTINUE
1041:      500 CONTINUE
1042: C
1043:      SI = HI(EN, EN)
1044:      IF (SI .EQ. 0.0D0) GO TO 540
1045:      NORM = CDABS(DCOMPLX(HR(EN, EN), SI))
1046:      SR = HR(EN, EN) / NORM
1047:      SI = SI / NORM
1048:      HR(EN, EN) = NORM
1049:      HI(EN, EN) = 0.0D0
1050:      IF (EN .EQ. N) GO TO 540
1051:      IP1 = EN + 1
1052: C      DO 520 J = IP1, N
1053:      YR = HR(EN, J)
1054:      YI = HI(EN, J)
1055:      HR(EN, J) = SR * YR + SI * YI
1056:      HI(EN, J) = SR * YI - SI * YR
1057: 520      CONTINUE
1058: 520      ::::: Inverse operation (columns) ::::::::::::
1059: C      540 DO 600 J = LP1, EN
1060:      XR = WR(J-1)
1061:      XI = WI(J-1)
1062: C      580 DO 580 I = 1, J
1063: C
1064:      YR = HR(I, J-1)
1065:      YI = 0.0D0
1066:      ZZR = HR(I, J)
1067:      Z2I = HI(I, J)
1068:      1F (I : EQ, J) GO TO 560
1069:      YI = HI(I, J-1)
1070:      HI(I, J-1) = XR * YI + XI * YR + HI(J, J-1) * Z2I
1071:      HR(I, J-1) = XR * YR - XI * YI + HI(J, J-1) * ZZR
1072: 560      HR(I, J) = XR * ZZR + XI * Z2I - HI(J, J-1) * YR
1073:      HI(I, J) = XR * Z2I - XI * ZZR - HI(J, J-1) * YI
1074: 580      CONTINUE
1075: 580      CONTINUE
1076: C
1077:      DO 590 I = LOW, IGH
1078:      YR = ZR(I, J-1)
1079:      YI = ZI(I, J-1)
1080:      ZZR = ZR(I, J)
1081:      Z2I = ZI(I, J)
1082:      ZR(I, J-1) = XR * YR - XI * YI + HI(J, J-1) * ZZR
1083:      ZI(I, J-1) = XR * YI + XI * YR + HI(J, J-1) * Z2I

```

C ****

```

1084:      ZR(I,J) = XR * ZZR + XI * ZZI - HI(J,J-1) * YR
1085:      ZI(I,J) = XR * ZZI - XI * ZZR - HI(J,J-1) * YI
1086:      590  CONTINUE
1087:      C
1088:      600  CONTINUE
1089:      C
1090:      IF (SI .EQ. 0.0D0) GO TO 240
1091:      C
1092:      DO 430 I = 1, EN
1093:        YR = HR(I,EN)
1094:        YI = HI(I,EN)
1095:        HR(I,EN) = SR * YR - SI * YI
1096:        HI(I,EN) = SR * YI + SI * YR
1097:      630  CONTINUE
1098:      C
1099:      DO 640 I = LDM, IGH
1100:        YR = ZR(I,EN)
1101:        YI = ZI(I,EN)
1102:        ZR(I,EN) = SR * YR - SI * YI
1103:        ZI(I,EN) = SR * YI + SI * YR
1104:      640  CONTINUE
1105:      C
1106:      GO TO 240
1107:      C
1108:      660  HR(EN,EN) = HR(EN,EN) + TR
1109:      WR(EN) = HR(EN,EN)
1110:      HI(EN,EN) = HI(EN,EN) + TI
1111:      WI(EN) = HI(EN,EN)
1112:      EN = ENM1
1113:      GO TO 220
1114:      C
1115:      C
1116:      680  NORM = 0.0D0
1117:      C
1118:      DO 720 I = 1, N
1119:      C
1120:      DO 720 J = 1, N
1121:      C
1122:      720  CONTINUE
1123:      C
1124:      IF (N .EQ. 1 .OR. NORM .EQ. 0.0D0) GO TO 1001
1125:      C
1126:      DO 800 NN = 2, N
1127:        EN = N + 2 - NN
1128:        XR = WR(EN)
1129:        XI = WI(EN)
1130:        ENM1 = EN - 1
1131:        C
1132:        FOR I=EN-1 STEP -1 UNTIL 1 DO -- :::::::
1133:          DO 780 II = 1, ENM1
1134:            I = EN - II
1135:            ZZR = HI(I,EN)
1136:            ZZI = HI(I,EN)
1137:            IF (I .EQ. ENM1) GO TO 760
1138:            IP1 = I + 1
1139:            DO 740 J = IP1, ENM1
1140:              ZZR = ZZR + HR(I,J) * HR(J,EN) - HI(I,J) * HI(J,EN)

```

C ****

```

1141:      Z2I = Z2I + HR(I,J) * HI(J,EN) + HI(I,J) * HR(J,EN)
1142:      CONTINUE
1143: C
1144:      YR = XR - WR(I)
1145:      YI = XI - WI(I)
1146:      IF (YR .EQ. 0.0D0 .AND. YI .EQ. 0.0D0) YR = MACHEP * NORM
1147:      Z3 = DCMPLEX(ZZR,Z2I) / DCMPLEX(YR,YI)
1148:      HR(I,EN) = DREAL(Z3)
1149:      HI(I,EN) = DIMAG(Z3)
1150:      CONTINUE
1151: C
1152:      B80 CONTINUE
1153: C      ::::::: End backsubstitution :::::::
1154:      ENM1 = N - 1
1155: C      ::::::: Vectors of isolated roots :::::::
1156:      DO B80 I = 1, ENM1
1157:          IF (I .GE. LOW .AND. I .LE. IGH) GO TO 840
1158:          IP1 = I + 1
1159: C
1160:          DO B20 J = IP1, N
1161:              ZR(I,J) = HR(I,J)
1162:              ZI(I,J) = HI(I,J)
1163:          CONTINUE
1164: C
1165:      B80 CONTINUE
1166: C      ::::::: Multiply by transformation matrix to give
1167:      C      ::::::: vectors of original full-matrix.
1168: C      DO B80 JJ = LOW, ENM1
1169:          FOR J=N STEP -1 UNTIL LOW+1 DO ---
1170:              J = N + LOW - JJ
1171:              M = MIN0(J-1, IGH)
1172: C
1173:          DO B80 I = LOW, IGH
1174:              ZZR = ZR(I,J)
1175:              Z2I = ZI(I,J)
1176: C
1177:          DO B60 K = LOW, N
1178:              ZZR = ZZR + ZR(I,K) * HR(K,J) - ZI(I,K) * HI(K,J)
1179:              Z2I = Z2I + ZR(I,K) * HI(K,J) + ZI(I,K) * HR(K,J)
1180:          CONTINUE
1181: C
1182:          ZR(I,J) = ZZR
1183:          ZI(I,J) = Z2I
1184:      B80 CONTINUE
1185: C
1186:      GO TO 1001
1187: C      ::::::: Set error -- no convergence to an
1188: C      ::::::: eigenvalue after 30 iterations :::::::
1189:      1000 IERR = EN
1190:      1001 RETURN
1191: C      ::::::: last card of COMOR2 :::::::
1192: END
1193: C*****
1194:      SUBROUTINE CORTH(NP, N, LOW, IGH, AR, AI, ORTR, ORTI)
1195: C
1196:      INTEGER I, J, M, N, II, JJ, LA, MP, NM, IGH, KP1, LOW
1197:      REAL*8 AR(NM,N), AI(NM,N), ORTR(IGH), ORTI(IGH)

```

C ****

```

1198      REAL*8 F,G,H,FI,FR,SCALE
1199      REAL*8 DSGR1,CDABS,DABS
1200      COMPLEX*16 DCMPXL
1201      C
1202      C This subroutine is a translation of a complex analogue of
1203      C the ALGOL procedure ORTHES, Num. Math. 12, 349-368(1968),
1204      C by Martin and Wilkinson.
1205      C Handbook for Auto. Comp., Vol. II-Linear Algebra, 339-358(1971).
1206      C
1207      C Given a complex general matrix, this subroutine
1208      C reduces a submatrix situated in rows and columns
1209      C LOW through IGH to upper Hessenberg form by
1210      C unitary similarity transformations.
1211      C
1212      C On input:
1213      C
1214      C NM must be set to the row dimension of two-dimensional
1215      C array parameters as declared in the calling program.
1216      C
1217      C
1218      C N is the order of the matrix;
1219      C
1220      C LOW and IGH are integers determined by the balancing
1221      C subroutine CBAL. If CBAL has not been used,
1222      C set LOW=1, IGH=N;
1223      C
1224      C AR and AI contain the real and imaginary parts,
1225      C respectively, of the complex input matrix.
1226      C
1227      C On output:
1228      C
1229      C AR and AI contain the real and imaginary parts,
1230      C respectively, of the Hessenberg matrix. Information
1231      C about the unitary transformations used in the reduction
1232      C is stored in the remaining triangles under the
1233      C Hessenberg matrix.
1234      C
1235      C ORTR and ORTI contain further information about the
1236      C transformations. Only elements LOW through IGH are used.
1237      C
1238      C Questions and comments should be directed to B. S. Garbow,
1239      C Applied Mathematics Division, Argonne National Laboratory
1240      C
1241      C
1242      C
1243      C LA = IGH - 1
1244      C KPI = LOW + 1
1245      C IF (LA .LT. KPI) GO TO 200
1246      C
1247      C DO 180 M = KPI, LA
1248      C     H = 0.0D0
1249      C     ORTR(M) = 0.0D0
1250      C     ORTI(M) = 0.0D0
1251      C     SCALE = 0.0D0
1252      C     .... Scale column (ALGOL TOL then not needed) .....
1253      C     DO 90 I = M, IGH
1254      C       SCALE = SCALE + DABS(AI(I,M-1)) + DABS(AI(I,M-1))

```

C ****

Page 25

```
1255: C      IF (SCALE .EQ. 0.0D0) GO TO 180
1256:      MP = M + IGH
1257:      DO 100 II = M, IGH
1258:      FOR I=IGH STEP -1 UNTIL M DO -- ::::::::::::
1259:      I = MP - II
1260:      ORTR(I) = AR(I,M-1) / SCALE
1261:      ORTI(I) = AI(I,M-1) / SCALE
1262:      H = H + ORTR(I) * ORTR(I) + ORTI(I) * ORTI(I)
1263:      CONTINUE
1264: 100
1265: C      G = DSGRT(H)
1266:      F = CDABSFDCMPLX(ORTR(M),ORTI(M))
1267:      IF (F .EQ. 0.0D0) GO TO 103
1268:      H = H + F * G
1269:      G = G / F
1270:      ORTR(M) = (1.0D0 + G) * ORTR(M)
1271:      ORTI(M) = (1.0D0 + G) * ORTI(M)
1272:      GO TO 105
1273: 105
1274: C      ORTR(M) = G
1275:      AR(M,M-1) = SCALE
1276:      DO 130 J = M, N
1277:      FOR (I-(U*UT)/H) * A -- ::::::::::::
1278: 105
1279:      DO 130 J = M, N
1280:      FR = 0.0D0
1281:      FI = 0.0D0
1282:      DO 110 II = M, IGH
1283:      I = MP - II
1284:      FR = FR + ORTR(I) * AR(I,J) + ORTI(I) * AI(I,J)
1285:      FI = FI + ORTR(I) * AI(I,J) - ORTI(I) * AR(I,J)
1286:      CONTINUE
1287: C      FR = FR / H
1288:      FI = FI / H
1289: 120
1290: C      DO 120 I = M, IGH
1291:      AR(I,J) = AR(I,J) - FR * ORTR(I) + FI * ORTI(I)
1292:      AI(I,J) = AI(I,J) - FR * ORTI(I) - FI * ORTR(I)
1293:      CONTINUE
1294: 120
1295: C      CONTINUE
1296: 130
1297: C      DO 160 I = 1, IGH
1298:      FR = 0.0D0
1299:      FI = 0.0D0
1300:      DO 140 JJ = M, IGH
1301:      FOR J=IGH STEP -1 UNTIL M DO -- ::::::::::::
1302:      J = MP - JJ
1303:      FR = FR + ORTR(J) * AR(I,J) - ORTI(J) * AI(I,J)
1304:      FI = FI + ORTR(J) * AI(I,J) + ORTI(J) * AR(I,J)
1305:      CONTINUE
1306: 140
1307: C      FR = FR / H
1308: 1309:      FI = FI / H
1310: C      DO 150 J = M, IGH
```

C ****

```
1312:      AR(I,J) = AR(I,J) - FR * ORTR(J) - FI * ORTI(J)
1313:      AI(I,J) = AI(I,J) + FR * ORTI(J) - FI * ORTR(J)
1314:      CONTINUE
1315:      C
1316:      160  CONTINUE
1317:      C
1318:      ORTR(M) = SCALE * ORTR(M)
1319:      ORTI(M) = SCALE * ORTI(M)
1320:      AR(M,M-1) = -G * AR(M,M-1)
1321:      AI(M,M-1) = -G * AI(M,M-1)
1322:      180  CONTINUE
1323:      C
1324:      200  RETURN
1325:      C      ::::: last card of CORTH ::::::::::::
1326:      END
1327:      SUBROUTINE HANKEL(Z, ZHO, ZHI)
1328:      C
1329:      IMPLICIT COMPLEX*16(Z), REAL*B(A-H,0-Y)
1330:      C
1331:      PAI=3.14159265D0
1332:      ZAI=CMPLX(0, 1, 0)
1333:      ZZ= ZAI*Z
1334:      CALL BSK(ZZ, ZKO, ZK1, 1)
1335:      ZC1=Z.0*ZAI/PAI
1336:      ZHO=ZC1*ZHO
1337:      ZHI=--Z.0*ZAI/PAI
1338:      RETURN
1339:      END
1340:      C
1341:      SUBROUTINE BSK(AA, SKO, SK1, JUMAX)
1342:      C
1343:      IMPLICIT DOUBLE PRECISION (A-H,0-Z)
1344:      COMPLEX*16 AA, SKO, SK1, XX, ARH, AIRH, ARH2, AL, ALE, A2, A, B, C,
1345:      1SUM1, SUM2, SUM3, T
1346:      DIMENSION XX(1), ASN(1)
1347:      P1=3. 14159265D0
1348:      E=0. 5772156649D0
1349:      ASN(1)=CDABS(AA)
1350:      AAAA
1351:      SITA=DATAN2(DIMAG(AA), AAA)
1352:      XX(1)=AA
1353:      DO 200 JJ=1, JUMAX
1354:      ARH=XX(JJ)
1355:      DINASN(JJ)*#2D0
1356:      IF (ASN(JJ)-3, DO) 10, 10, 20
1357:      10 ARH=ARH/2, DO
1358:      AL=DCMPLX(DL,DO*(ASN(JJ)/2, DO), SITA)
1359:      ALEE+AL
1360:      AIRH=1, DO, ARH
1361:      A2=ARH*#ARH2
1362:      A=AIRH*#2, DO
1363:      B=-ALE
1364:      SUM1=DCMPLX(0, DO, 0, DO)
1365:      SUM2=DCMPLX(0, DO, 0, DO)
1366:      SUM3=DCMPLX(0, DO, 0, DO)
1367:      DO 100 I=1, B
1368:      RI=I, DO, (FLDAT(I)*1, DO)
```

C *****

```
1369:     RI2=I1*RI
1370:     T=A*I2*RI2
1371:     A=T
1372:     B=B+RI
1373:     C=A*B
1374:     SUM1=SUM1+A
1375:     SUM2=SUM2+FLOAT(1)*C
1376:     SUM3=SUM3+C
1377: 100  CONTINUE
1378:     SK0=-ALE+SUM3*ARH2
1379:     SK1=AINRH+SUM1/2. DO-SUM2
1380:     GO TO 11
1381: 20  DB=8.*ASN(JJ)
1382:     DAO=1. DO
1383:     DA1=1. DO
1384:     SUM1=DCMPLX(1,DO,0,DO)
1385:     SUM2=DCMPLX(1,DO,0,DO)
1386:     JMAX=7
1387:     DO 110 J=1,JMAX
1388:     NDDD=(2*j-1)*#2
1389:     BO=-FLOAT(NDDD)*1. DO
1390:     B1=FLOAT(4-NDDD)
1391:     S=-S1*T*FLOAT(J)
1392:     SS=D1*S
1393:     CS=DCOS(S)
1394:     DD=A*FLOAT(J)*1. DO*D8
1395:     DAO=DAO*BO/DDA
1396:     T=DCMPLX(DAO*CS, DAO*SS)
1397:     SUM1=SUM1+T
1398:     DA1=DA1*B1/DDA
1399:     T=DCMPLX(DA1*CS, DA1*SS)
1400:     SUM2=SUM2+T
1401: 110  CONTINUE
1402:     T=CDBORT(P1/(2. DO*AA))/CDEXP(AA)
1403:     SK0=SUM1*T
1404:     SK1=SUM2*T
1405: 11  CONTINUE
1406: 200  CONTINUE
1407:     RETURN
1408: END
```

```

1 , ZTUZT(MLY,10), ZTUTR(MLY,10), ZPAI(MLY,10), ZDSGR(MLY,10),
2 ZDSGZ(MLY,10), ZDSGT(MLY,10)
* COMMON /COND1/ KC, KP, KG(4), NLAY, MMH
COMMON /GROUND/ ZAA(10), ZG(10), H(10), PERM(10),
+ ROP(10), CNE(10), RD(10), RQF(10), EI(10), DEP(11), DEM(10)
* COMMON /NOCH3/ ZPHI(M6,MLY*4), ZPINV(MLY*4,MLY*4)
COMMON /NOCH5/ ZPHIM(M6,MLY*4)
COMMON /DRY6/ ZPHAT(M6,MLY*4)
COMMON /YND/ MJYD, MZRZ, MJTH, MR1, MRRZ, MRTB, MUR, MUZ, MHZ, MWZ, MUT, MWT
COMMON /CONST/ QAL, PAI, DEPTH, NOT, VSB, NFRG, RR, DOO, NCASE
COMMON /HSFCNC/ ZHS(MLY*4), ZHO(MLY*4), ZHI(MLY*4)
DATA CX/1.0, 2.0, 3.0, 5.0, 10.0, 20.0, 30.0, 50.0, 100.0, 200.0/
1439: ZAI=CMPLX(0.0, 1.0)

1440: C----- CALCULATION STRESS AND PORE PRESSURE AT R=CX(1)*RR
1441: C----- -----
1442: C----- -----
1443: NC=10
1444: DO 15 N=1, NC
1445: X(N)=RR*CX(N)
1446: IF(NXOUT(N).EQ.0) GO TO 15
1447: IF(N.EQ.1) GO TO 10
1448: C
1449: CALL HNLST(MR1,X(N))
1450: C
1451: CALL PINVS(1,X(N))
1452: C
1453: 10 CALL ZMULTI(ZPHIM,ZALP,ZUM,MJYD,MR1,1,M6,1,1)
1454: C
1455: CALL OUT1(N,X(N),ZUM,ZPUX,MAU)
1456: C
1457: CALL SCMAN(IND1,ZALP,ZHS,X(N))
1458: IF(N.NE.1) GO TO 15
1459: DO 13 J=1,NLAY
1460: ZD1=ZUW(J)
1461: ZD2=ZUW(J+1)
1462: IF(J.EQ.NLAY) ZD2=CMPLX(0.0,0.0)
1463: ZAVE=(ZD1+ZD2)/2.0
1464: ZK=ZSGR(J,1)/ZAVE
1465: WRITE(KG(2),650) DEM(J), ZK,J
1466: 11 CONTINUE
1467: 15 CONTINUE
1468: CALL OUT2(X,NOT,IND1)
1469: IF(NCASE.NE.1) GO TO 102
1470: IF(NFRG.NE.1) GO TO 102
1471: IF(INDT.NE.1) GO TO 102
1472: REWIND KG(3)
1473: DX=0.01*D00
1474: XX=RR-DX
1475: DO 100 N=1,381
1476: IF(N.GT.201) DX=0.1*1000
1477: XX=DX+XX
1478: C
1479: CALL HNLST(MR1,XX)
1480: C
1481: CALL PINVS(1,XX)
1482: C

```

```

1483: CALL ZMULTI(ZPHIM,ZALP,ZUM,MJYD,MRI,1,M6,1,1)
1484: Z1=ZUM*(MUR+1)*1000.0
1485: Z2=ZUM*(MUR*2+1)*1000.0
1486: Z3=ZUM*(MUR*2+1)*ZAI*1000.0
1487: Z4=ZUM*(MUR*3+1+MUR)*1000.0
1488: Z5=ZUM*(MUR*3+MUR+1+MUR)*1000.0
1489: Z6=ZUM*(MUR*3+MUR*2+1+MUR)*1000.0
1490: WRITE(IKQ(3),600) XX/D00,Z1,Z2,Z3,Z4,Z5,Z6
1491: 100 CONTINUE
1492: 102 CONTINUE
1493: 600 FORMAT(F10.3,12E15.5)
1494: 650 FORMAT(F5.2,4E15.5)
1495: RETURN
1496: END
1497: C*****
1498: SUBROUTINE QUT1(RN1,X,ZUM,ZRUX,MAU)
1499: C*****
1500: IMPLICIT COMPLEX*6(Z)
1501: DIMENSION ZUM(*),ZPUX(*)
1502: COMMON /COND1/ KC,KP,KQ(4),NLAY,MMH
1503: COMMON /JYD/ MJYD,MJRZ,MJTH,MRI,MRRZ,MRT,MUR,MWZ,MUT,MWT
1504: COMMON /GROUND/ ZAA(10),ZB(10),ZAL(10),ZD(10),PERM(10),
1505: + ROP(10),CNE(10),RD(10),RDF(10),E1(10),DEP(11),DEM(10)
1506: C
1507: DZ=0.0
1508: ZAI=CMPLX(0,0,1,0)
1509: MAW=NLAY-MUR
1510: WRITE(KP,600) X
1511: IF(NN1.NE.1) GO TO 5
1512: WRITE(KP,603)
1513: DO 3 N=1,MUR
1514: WRITE(KP,605) N+MAU,ZPUX(N),ZPUX(N+MUR),ZPUX(N+MUR*2)
1515: + ZPUX(N+MUR*3)
1516: 3 CONTINUE
1517: 5 WRITE(KP,601)
1518: DO 1 N=1,MUR
1519: AB=ABS(ZUM(N))
1520: AB=ABS(ZUM(N+MUR))
1521: AB2=ABS(ZUM(N+MUR*2))
1522: WRITE(KP,605) N+MAU,ZUM(N),AB,ZUM(N+MUR),AB1
1523: + ZUM(N+MUR*2)*ZAI,AB2
1524: 1 CONTINUE
1525: IF(MUR.EQ.0) GO TO 4
1526: WRITE(KP,602)
1527: WRITE(KP,601)
1528: DO 2 N=1,MUR
1529: NN=MUR*3+N
1530: AB=ABS(ZUM(NNN))
1531: AB=ABS(ZUM(NNN+MUR))
1532: AB2=ABS(ZUM(NNN+MUR*2))
1533: WRITE(KP,605) N+MAU,ZUM(NNN),AB,ZUM(NNN+MUR),AB1
1534: + ZUM(NNN+MUR*2)*ZAI,AB2
1535: 2 CONTINUE
1536: 4 CONTINUE
1537: DD 6 N=1,NLAY
1538: S1=ABS(ZUM(N))*1000.0
1539: S2=ABS(ZUM(N+MUR))*1000.0

```

```

1540: S3=ABS(ZUM(N+MUR*2))*1000.0
1541: S4=ABS(ZUM(N*MUR*3))*1000.0
1542: S5=ABS(ZUM(N*MUR*3+MUR))*1000.0
1543: S6=ABS(ZUM(N*MUR*3+MUR*2))*1000.0
1544: WRITE(KG(3),650) DEP(N),S1,S2,S3,S4,S5,S6
1545: C CONTINUE
1546: C WRITE(KG(3),650) DEP(NLAY+1),DZ,D2,DZ,DZ,DZ,DZ
1547: C
1548: C FORMAT
1549: C
1550: 600 FORMAT(1H,'X=',F10.2,'(M)',/)
1551: + 'SOLID DISPLACEMENT')
1552: 601 FORMAT(1H,'LAYER NO.',2X,'DISP(R)',29X,'DISP(Z)',2BX,
1553: + 'DISP.(THETA)')
1554: 602 FORMAT(1H,'FLUID DISPLACEMENT')
1555: 603 FORMAT(1H,'LOAD VECTOR',5X,'PR',17X,'PZ',22X,'PT',22X
1556: + , 'PORE PRESSURE')
1557: 605 FORMAT(1H,13.5X,10E12,3)
1558: 650 FORMAT(F5.2,6F10.5)
1559: RETURN
1560: END
1561: C ****SUBROUTINE SONAIS, INDI1, ZALP, ZHS, RI)
1562: ****SUBROUTINE SONAIS, INDI1, ZALP, ZHS, RI)
1563: C*****
1564: IMPLICIT COMPLEX*16(Z)
1565: PARAMETER (MLY=10)
1566: PARAMETER (MSA=MLY*6)
1567: DIMENSION INDI(MLY,*),ZALP(*),ZHS(*)
1568: COMMON /COND1/ KC,KP,KQ(4),NLAY,MMN
1569: COMMON /CONST/ QAL,PA,DEPTH,NOT,VSB,NFRG,RR,DOO,INCASE
1570: COMMON /JYD/ MJYD,MURZ,MJTH,MR1,MRRZ,MTH,MUR,MUZ,MUT,MWT
1571: COMMON /GROUND/ ZAA(10),ZG(10),ZL(10),ZD(10),PERM10,
1572: + ROP(10),CNE(10),RD(10),RDF(10),DEM(10),
1573: + ROP(10),CNE(10),RD(10),RDF(10),DEM(10),
1574: 1, ZTUT(MLY,10),ZTUTR(MLY,10),ZSGT(MLY,10),ZTRZ(MLY,10),
1575: 2, ZDSQR(MLY,10),ZDSQT(MLY,10),ZALPH(MLY*4),ZALPH2(MLY*4),
1576: 3, ZT1(7,5),ZT2(7,5),ZT3(7,5),ZT4(7,5),ZT5(7,5)
1577: 4, ZVJ(5,MLY*4),ZVJ(5,MLY*4),ZPJ(5,MLY*4),ZPJ(5,MLY*4)
1578: 5, ZWH(5,MLY*4),ZW1(7),ZW1(7),ZW2(7),ZW3(7),ZW5(7)
1579: COMMON /DMY5/ ZUMAT(MS,MLY*4)
1580: COMMON /DMY6/ ZPMAT(MS,MLY*4)
1581: C
1582: ZA1=CNPLX(0,0,1,0)
1583: ZERO=CNPLX(0,0,0,0)
1584: ZONE=CNPLX(1,0,0,0)
1585: C
1586: L=0
1587: L=0
1588: WRITE(*,*) 'SUBROUTINE ZGMAN'
1589: DO 11 L=1,MR1
1590: ZALPH(L)=ZALP(L)*ZHS(L)
1591: ZALPH2(L)=ZALP(L)*ZHS(L)*ZHS(L)
1592: 11 CONTINUE
1593: C
1594: DO 1 J=1,NLAY
1595: C
1596: C

```

```

1597: C      SET T1 AND T6
1598: C      CALL ZEROCL(ZT1, 7, 5, 7)
1599: C      CALL ZEROCL(ZT2, 7, 5, 7)
1600: C      CALL ZEROCL(ZT3, 7, 5, 7)
1601: C      CALL ZEROCL(ZT4, 7, 5, 7)
1602: C      CALL ZEROCL(ZT5, 7, 5, 7)
1603: C      CALL ZEROCL(ZPJ, 5, NLY*4, 5)
1604: C      CALL ZEROCL(ZVJ, 5, NLY*4, 5)
1605: C      CALL ZEROCL(ZPJI, 5, NLY*4, 5)
1606: C      CALL ZEROCL(ZVJI, 5, NLY*4, 5)
1607: C
1608: C      CALL ZAL(J)*ZG(J)
1609: C      ZAG=ZAL(J)*ZG(J)
1610: C      ZA2G=ZAA(J)-2.0*ZG(J)
1611: C      ZT1(1,1)=ZAA(J)
1612: C      ZT1(1,4)=ZAG
1613: C      ZT1(2,2)=ZG(J)
1614: C      ZT1(3,3)=ZG(J)
1615: C      ZT1(4,1)=ZAG
1616: C      ZT1(4,4)=ZG(J)
1617: C      ZT1(5,1)=ZA2G
1618: C      ZT1(5,4)=ZAG
1619: C      ZT1(6,1)=ZA2G
1620: C      ZT1(6,4)=ZAG
1621: C
1622: C      ZCC=ZG(J)/R1
1623: C      ZT2(1,1)=-2.0*ZCC
1624: C      ZT2(1,3)=2.0*MM*ZCC
1625: C      ZT2(3,1)=ZT2(1,3)
1626: C      ZT2(3,3)=ZT2(1,1)
1627: C      ZT2(2,2)=MM*ZCC
1628: C      ZT2(5,1)=2.0*ZCC
1629: C      ZT2(5,3)=-2.0*MM*ZCC
1630: C      ZT2(7,2)=-MM*ZCC
1631: C
1632: C      ZCC=2.0*ZG(J)*MM*(MM+1)/(R1*R1)
1633: C      ZT3(1,1)=-ZCC
1634: C      ZT3(1,3)= ZCC
1635: C      ZT3(3,1)= ZCC
1636: C      ZT3(3,3)=-ZCC
1637: C      ZT3(5,1)= ZCC
1638: C      ZT3(5,3)=-ZCC
1639: C
1640: C      ZT4(1,2)=-ZA2G
1641: C      ZT4(1,5)=-ZAG
1642: C      ZT4(2,1)=ZG(J)
1643: C      ZT4(4,2)=-ZAG
1644: C      ZT4(4,5)=-ZG(J)
1645: C      ZT4(5,2)=-ZA2G
1646: C      ZT4(5,5)=-ZAG
1647: C      ZT4(6,2)=-ZAA(J)
1648: C      ZT4(6,5)=-ZAG
1649: C      ZT4(7,3)=ZG(J)
1650: C
1651: C      ZCC=ZG(J)*MM/R1
1652: C      ZT5(2,1)=ZCC
1653: C      ZT5(2,3)=-ZCC

```

```

1654: ZT5(7,1)=-ZCC
1655: ZT5(7,3)=ZCC
1656: C
1657: DO 4 K=1,7
1658: DO 4 L=1,5
1659: ZT1(K,L)=-ZT1(K,L)/2.0
1660: ZT2(K,L)=-ZT2(K,L)/2.0
1661: ZT3(K,L)=-ZT3(K,L)/2.0
1662: ZT4(K,L)=-ZT4(K,L)/H(J)
1663: ZT5(K,L)=-ZT5(K,L)/H(J)
1664: 4 CONTINUE
1665: C
1666: C----- CALCULATE STRESS AND PORE PRESSURE AT THE CENTER OF LAYER
1667: C----- IF (IND1(J,1),NE.1) GO TO 9
1668: C----- LU=LU+1
1669: IF (IND1(J,1),NE.1) GO TO 9
1670: LU=LU+1
1671: DO 10 L=1,MR1
1672: ZVJ(1,L)=ZVMAT(LU,L)
1673: ZVJ(2,L)=ZVMAT(MUR+LU,L)
1674: ZVJ(3,L)=ZVMAT(MUR+MUZ+LU,L)
1675: ZPJ(1,L)=ZPMAT(LU,L)
1676: ZPJ(2,L)=ZPMAT(MUR+LU,L)
1677: ZPJ(3,L)=ZPMAT(MUR+MUZ+LU,L)
1678: 10 CONTINUE
1679: 9 CONTINUE
1680: IF (IND1(J,3),NE.1) GO TO 12
1681: LW=LW+1
1682: DO 21 L=1,MR1
1683: ZVJ(4,L)=ZVMAT(MUR*3+LW,L)
1684: ZVJ(5,L)=ZVMAT(MUR*3+MUR+LW,L)
1685: ZPJ(4,L)=ZPMAT(MUR*3+LW,L)
1686: ZPJ(5,L)=ZPMAT(MUR*3+MUR+LW,L)
1687: 21 CONTINUE
1688: 12 CONTINUE
1689: IF (J.EQ.NLAY) GO TO 40
1690: IF (IND1(J+1,1),NE.1) GO TO 51
1691: DO 50 L=1,MR1
1692: ZVJ(1,L)=ZVMAT(LU+1,L)
1693: ZVJ(2,L)=ZVMAT(MUR+LU+1,L)
1694: ZVJ(3,L)=ZVMAT(MUR+MUZ+LU+1,L)
1695: ZPJ(1,L)=ZPMAT(LU+1,L)
1696: ZPJ(2,L)=ZPMAT(MUR+LU+1,L)
1697: ZPJ(3,L)=ZPMAT(MUR+MUZ+LU+1,L)
1698: 50 CONTINUE
1699: 51 CONTINUE
1700: IF (IND1(J+1,3),NE.1) GO TO 23
1701: DO 22 L=1,MR1
1702: ZVJ(4,L)=ZVMAT(MUR*3+LW+1,L)
1703: ZVJ(5,L)=ZVMAT(MUR*3+MUR+LW+1,L)
1704: ZPJ(4,L)=ZPMAT(MUR*3+LW+1,L)
1705: ZPJ(5,L)=ZPMAT(MUR*3+MUR+LW+1,L)
1706: 22 CONTINUE
1707: 23 CONTINUE
1708: 40 CONTINUE
1709: C
1710: DO 60 L=1,MR1

```

```

1711: DO 60 K=1,5
1712:   60 ZWJK(K,L)=ZVJK(L)+ZVJ1(K,L)
1713:   CALL ZMULT1(ZWM, ZALPH2, ZWI, 5, MRL, 1, 5, 1, 1)
1714:   CALL ZMULT1(ZT1, ZWI, ZWI1, 7, 5, 1, 7, 1, 1)
1715: C
1716:   CALL ZMULT1(ZWM, ZALP, ZWI, 5, MRL, 1, 5, 1, 1)
1717:   CALL ZMULT1(ZT3, ZWI, ZW33, 7, 5, 1, 7, 1, 1)
1718: C
1719: DO 61 L=1, MRL
1720:   61 ZWJK(K,L)=ZPJK(K,L)+ZPJK1(K,L)
1721:   CALL ZMULT1(ZWM, ZALPH, ZWI, 5, MRL, 1, 5, 1, 1)
1722:   CALL ZMULT1(ZT2, ZWI, ZW22, 7, 5, 1, 7, 1, 1)
1723: C
1724: DO 62 L=1, MRL
1725:   62 ZWJK(K,L)=ZVJK(L)-ZVJ1(K,L)
1726:   CALL ZMULT1(ZWM, ZALP, ZWI, 5, MRL, 1, 5, 1, 1)
1727:   CALL ZMULT1(ZT5, ZWI, ZW44, 7, 5, 1, 7, 1, 1)
1728: C
1729: DO 63 L=1, MRL
1730:   63 ZWJK(K,L)=ZVJK(L)+ZVJ1(K,L)
1731:   CALL ZMULT1(ZT4, ZWI, ZW55, 7, 5, 1, 7, 1, 1)
1732: C
1733:   63 ZWJK(L,J)=-ZVJ1(K,L)+ZVJ1(K,L)
1734:   CALL ZMULT1(ZWM, ZALP, ZWI, 5, MRL, 1, 5, 1, 1)
1735:   CALL ZMULT1(ZT5, ZWI, ZW55, 7, 5, 1, 7, 1, 1)
1736: C
1737:   ZSGR(J,NS)=ZWI1(1)*ZW22(1)+ZW33(1)*ZW44(1)+ZW55(1)
1738:   ZTUR(J,NS)=ZWI1(2)*ZW22(2)+ZW33(2)*ZW44(2)+ZW55(2)*ZAI
1739:   ZTOR2(J,NS)=ZWI1(3)*ZW22(3)+ZW33(3)*ZW44(3)+ZW55(3)
1740:   ZPAI(J,NS)=ZWI1(4)*ZW22(4)+ZW33(4)*ZW44(4)+ZW55(4)
1741:   ZSGT(J,NS)=ZWI1(5)*ZW22(5)+ZW33(5)*ZW44(5)+ZW55(5)
1742:   ZSGZ(J,NS)=ZWI1(6)*ZW22(6)+ZW33(6)*ZW44(6)+ZW55(6)
1743:   ZTU2T(J,NS)=(ZWI1(7)*ZW22(7)+ZW33(7)*ZW44(7)+ZW55(7))*ZAI
1744: 1 CONTINUE
1745: C
1746: RETURN
1747: END
1748: C ****
1749: SUBROUTINE WAVEOT
1750: C ****
1751: IMPLICIT COMPLEX*16(Z)
1752: PARAMETER (MLY=10)
1753: COMMON /HSFNC/ ZHS(MLY*4), ZHO(MLY*4), ZH1(MLY*4)
1754: COMMON /NOCH3/ ZPH1(M6, MLY*4), ZPINV(MLY**, MLY*4)
1755: COMMON /COND1/ KC, KP, KQ(4), NLAY, MM
1756: COMMON /JYD/ MJD, MURZ, MJTH, MRL, MRRZ, MRTN, MUR, MUZ, MUR, MUZ, MUT, MUT
1757: COMMON /CONST/ QAL, PAI, DEPTH, NOT, VSB, NFRG, RR, D00, NCASE
1758: COMMON /GROUND/ ZAA(10), ZG(10), ZAL(10), ZG(10), H10, PERM10,
+ ROP(10), CNE(10), R0F(10), E1(10), DEP(11), DEM(10),
1759: C
1760: MODE SHAPE OUTPUT
1761: C
1762: C
1763: C
1764: ZERO=CMPLX(0, 0, 0)
1765: WRITE(KP, 602)
1766: IF (INFRQ .GT. 10) RETURN
1767: MAU=MLAY-MUR

```

```

1768:      MAINLAY-MUR
1769:      DD 21 L=1,MR1
1770:      PMAXS=0.0
1771:      PMAXF=0.0
1772:      DO 20 J=1,NLAY
1773:      J1=J-MAU
1774:      J2=J-MAW
1775:      IF (J1.LE.0) THEN
1776:      ZUT=ZERO
1777:      ZUZ=ZERO
1778:      ZWR=ZERO
1779:      ELSE
1780:      ZUR=2PHI (J1,L)
1781:      ZUZ=2PHI (J1+MUR,L)
1782:      ZUT=2PHI (J1+MUR*2,L)
1783:      END IF
1784: C
1785:      IF (J2.LE.0) THEN
1786:      ZWR=ZERO
1787:      ZWZ=ZERO
1788:      ZWT=ZERO
1789:      ELSE
1790:      ZUR=2PHI (J2+MUR*3,L)
1791:      ZWZ=2PHI (J2+MUR*3+MUR,L)
1792:      ZWT=2PHI (J2+MUR*3+MUR*2,L)
1793:      END IF
1794: C
1795:      IF (ABS(ZUR).GT.PMAXS) PMAXS=ABS(ZUR)
1796:      IF (ABS(ZUZ).GT.PMAXS) PMAXS=ABS(ZUZ)
1797:      IF (ABS(ZUT).GT.PMAXS) PMAXS=ABS(ZUT)
1798:      IF (ABS(ZWR).GT.PMAXF) PMAXF=ABS(ZWR)
1799:      IF (ABS(ZWZ).GT.PMAXF) PMAXF=ABS(ZWZ)
1800:      IF (ABS(ZWT).GT.PMAXF) PMAXF=ABS(ZWT)
1801:      WRITE(KG(2),650) DEP(J),ZUR,ZUZ,ZWT
1802:      20 CONTINUE
1803:      WRITE(KG(2),650) DEP(NLAY+1),ZERO,ZERO,ZERO,ZERO
1804:      P1=0.0
1805:      IF (PMAXS.GT.1.0E-12) P1=PMAXF/PMAXS
1806:      WRITE(NP,601) L,PMAXS,PMAXF,P1
1807:      21 CONTINUE
1808:      650 FORMAT(F5.2,12F10.5)
1809:      601 FORMAT(1H ,15.5X,3E15.5)
1810:      602 FORMAT(1H ,*** RATIO BETWEEN MAX(SOLID-R) AND MAX(FLUID-R) ***')
1811:      RETURN
1812:      END
1813: C ****
1814:      SUBROUTINE OUT2(X,NC,IND1)
1815: C ****
1816: C
1817: C      IMPLICIT COMPLEX*16(Z)
1818: C      PARAMETER (MLY=0)
1819: C      DIMENSION X(*),IND1(MLY,*)
1820: C
1821: C      OUTPUT FOR STRESS AND PORE PRESSURE
1822: C
1823: COMMON /COND1/ KC,KP,KG(4),NLAY,MM
1824: COMMON /CONST/ GAL,PAI,DEPTH,NOT,USB,NFRQ,RR,DOO,NCASE

```

*** C PROGRAM *****AXS***** CODED BY KAZAMA 1990 5/30

```
1825: COMMON /JYD/, MJRD, MJDZ, MUTH, MR1, MR2, MUR, MUTH, MUR, MWZ, MUR, MWZ, MUR, MWZ
1826: COMMON /ROUND/, ZAA(10), ZG(10), ZAL(10), H(10), R(10), PERM(10),
+ RGP(10), CNE(10), RO(10), RDF(10), E(10), DEM(10), DEM(11), DEM(10)
1827: COMMON /DRY3/, ZSGR(MLY,10), ZSGZ(MLY,10), ZSGT(MLY,10), ZTURZ(MLY,10),
1     , ZTUZT(MLY,10), ZTUTR(MLY,10), ZPAI(MLY,10), ZDSGR(MLY,10),
1827: 2     , ZDSGZ(MLY,10), ZDSET(MLY,10), YCC(MLY,10)
1830: C
1831: C
1832: ADB=0.0
1832: Z2=CMPLX(0,0,0,0)
1833: DO 6 J=1, NLAY
1834:  ODB=RO(J)*RCF(J)
1835:  AJSB=CD*9, B*B(H(J)/2, 0+ADB
1836:  ADB=ADB+OD*9, B*B(H(J)
1837:  Z1=CMPLX(AJSB,0,0)
1838:  DO 6 L=1, NC
1839:  YCC(J,L)=0.0
1840:  ZDSGR(J,L)=ZSGR(J,L)-ZPAI(J,L)
1841:  ZDSGZ(J,L)=ZSGZ(J,L)-ZPAI(J,L)
1842:  ZDSGT(J,L)=ZSGT(J,L)-ZPAI(J,L)
1843:  IF(IND1(J,L).EQ.0) GO TO 6
1844:  Z2=ZPAI(J,L)/(Z1+ZDSGZ(J,L))
1845:  YCC(J,L)=ABS(Z2)
1846:  CONTINUE
1847: 6 CONTINUE
1848: C
1849: IF(NFREQ.NE.1) GO TO 7
1850: C
1851: DO 5 N=1, NC
1852:  DO 5 J=1, NLAY
1853:  WRITE(KG(4), 650) DEM(J), ABS(ZSGR(J,N)), ABS(ZSGZ(J,N)),
1     , ABS(ZTURZ(J,N)), ABS(ZTUZT(J,N)), ABS(ZDSGR(J,N)),
1854: 1     , ABS(ZTUTR(J,N)), ABS(ZDSGT(J,N)), ABS(ZPAI(J,N)),
1855: 2     , ABS(ZDSST(J,N)), ABS(ZPAI(J,N)), YCC(J,N)
1856: 3     , ABS(ZDSGT(J,N)), ABS(ZPAI(J,N)), YCC(J,N)
1857: 5 CONTINUE
1858: 7 CONTINUE
1859: C
1860: WRITE(KP, 600) (X(N), N=1, NC)
1861: DD 10 J=1, NLAY
1862: WRITE(KP, 601) J
1863: WRITE(KP, 602) (ABS(ZSGR(J,N)), N=1, NC)
1864: WRITE(KP, 603) (ABS(ZSGZ(J,N)), N=1, NC)
1865: WRITE(KP, 604) (ABS(ZTURZ(J,N)), N=1, NC)
1866: WRITE(KP, 605) (ABS(ZTUZT(J,N)), N=1, NC)
1867: WRITE(KP, 606) (ABS(ZTUTR(J,N)), N=1, NC)
1868: WRITE(KP, 607) (ABS(ZDSGR(J,N)), N=1, NC)
1869: WRITE(KP, 608) (ABS(ZPAI(J,N)), N=1, NC)
1870: WRITE(KP, 609) (ABS(ZDSGT(J,N)), N=1, NC)
1871: WRITE(KP, 610) (ABS(ZDSGZ(J,N)), N=1, NC)
1872: WRITE(KP, 611) (ABS(ZDSGT(J,N)), N=1, NC)
1873: WRITE(KP, 612) (YCC(J,N), N=1, NC)
1874: 10 CONTINUE
1875: C
1876: C
1877: 600 FORMAT(1H, 'DISTANCE FROM CENTER', //, 9X, 11F11.2)
1878: 601 FORMAT(1H, 'LAYER NO. ', 15)
1879: 602 FORMAT(1H, 'SIGMA-R ', '101PE11.3')
1879: 603 FORMAT(1H, 'SIGMA-Z ', '101PE11.3')
1880: 604 FORMAT(1H, 'SIGMA-THETA ', '101PE11.3')
```

```

1882: 605 FORMAT(1H , 'TAU-RZ      ', 10(IPE11,3))
1883: 606 FORMAT(1H , 'TAU-ZT      ', 10(IPE11,3))
1884: 607 FORMAT(1H , 'TAU-TR      ', 10(IPE11,3))
1885: 608 FORMAT(1H , 'PORE-PRESSURE', 10(IPE11,3))
1886: 609 FORMAT(1H , 'SIGMA-R<EFFT>', 10(IPE11,3))
1887: 610 FORMAT(1H , 'SIGMA-Z<EFFT>', 10(IPE11,3))
1888: 611 FORMAT(1H , 'SIGMA-T<EFFT>', 10(IPE11,3))
1889: 612 FORMAT(1H , 'EFCT STRES RT', 10(IPE11,3))
1890: 650 FORMAT(1F5.2,1E15.5)
1891: C
1892:      RETURN
1893:      END
1894: C*****
1895:      SUBROUTINE CNVER1(ZR,ZC11,ZC12,ZC13,ZC21,ZC22,ZC23,ZC31,ZC32,
1896:      +
1897: C*****
1898:      IMPLICIT COMPLEX*16(Z)
1899:      PARAMETER (MLY=10)
1900:      DIMENSION ZR(MLY*4,MLY*4)
1901:      1   ,ZC11(MLY,MLY),ZC12(MLY,MLY),ZC13(MLY,MLY)
1902:      2   ,ZC21(MLY,MLY),ZC22(MLY,MLY),ZC23(MLY,MLY)
1903:      3   ,ZC31(MLY,MLY),ZC32(MLY,MLY),ZC33(MLY,MLY)
1904: C
1905:      CALL ZEROCL(ZC11,ML,Y,ML,Y,MLY)
1906:      CALL ZEROCL(ZC12,ML,Y,ML,Y,MLY)
1907:      CALL ZEROCL(ZC13,ML,Y,ML,Y,MLY)
1908:      CALL ZEROCL(ZC21,ML,Y,ML,Y,MLY)
1909:      CALL ZEROCL(ZC22,ML,Y,ML,Y,MLY)
1910:      CALL ZEROCL(ZC23,ML,Y,ML,Y,MLY)
1911:      CALL ZEROCL(ZC31,ML,Y,ML,Y,MLY)
1912:      CALL ZEROCL(ZC32,ML,Y,ML,Y,MLY)
1913:      CALL ZEROCL(ZC33,ML,Y,ML,Y,MLY)
1914: C
1915:      DO 1 L=1,MUR
1916:      DO 1 K=1,MUR
1917:      ZC21(K,L)=ZR(K+MUR ,L+MUR )
1918:      ZC22(K,L)=ZR(K+MUR ,L+MUR*2)
1919:      ZC23(K,L)=ZR(K+MUR*2,L+MUR )
1920:      ZC31(K,L)=ZR(K+MUR*2,L+MUR*2)
1921:      1 CONTINUE
1922: C
1923:      IF (MAU.LE.0) GO TO 40
1924:      DO 2 L=1,MAU
1925:      DO 21 K=1,MAU
1926:      21 ZC11(K ,L)=ZR(K+MUR*3,L+MUR*3)
1927:      DO 22 K=1,MUR
1928:      ZC11(K+MAU,L)=ZR(K ,L+MUR*3)
1929:      ZC21(K ,L)=ZR(K+MUR ,L+MUR*3)
1930:      ZC22(K ,L)=ZR(K+MUR*2,L+MUR*3)
1931:      22 ZC31(K ,L)=ZR(K+MUR*2,L+MUR*3)
1932:      2 CONTINUE
1933:      40 CONTINUE
1934:      DO 3 L=1,MUR
1935:      DO 31 K=1,MAU
1936:      31 ZC11(K ,L+MAU)=ZR(K+MUR*3,L+MUR*3)
1937:      ZC12(K ,L )=ZR(K+MUR*3,L+MUR )
1938:      ZC13(K ,L )=ZR(K+MUR*3,L+MUR*2)

```

```

1939: 31 CONTINUE
1940: DO 32 K=1, MUR
1941: ZC1(K+MAU,L+MAU)=ZR(K+MAU ,L
1942: ZC2(K ,L+MAU)=ZR(K+MAU ,L
1943: ZC3(K ,L+MAU)=ZR(K+MAU*2,L
1944: ZC12(K+MAU,L )=2R(K+MUR*2,L
1945: ZC13(K+MAU,L )=2R(K ,L+MUR )
1946: 32 CONTINUE
1947: 3 CONTINUE
1948: RETURN
1949: END
1950: C *****
1951: SUBROUTINE SLCTHS(2PHI,ZHS,HSR,HSI,PR,P,MSOL)
1952: C *****
1953: *****
1954: IMPLICIT REAL*8(A-H,O-7) , COMPLEX*16(Z)
1955: PARAMETER (MLY=10)
1956: PARAMETER (MB=MLY*6)
1957: DIMENSION 2PHI(MB,*),ZHS(*),HSR(*),HSI(*),
1958: + PRMLY*B,*),PI(MLY*B,*),MIND(100)
1959: COMMON /COND1/ HC,KP,KQ(4),NLAY,MM
1960: C
1961: MCHA=MRRZ
1962: EPS=1.0E-07
1963: JCNT=0
1964: JD_10_N=1,MSOL
1965: 10 MIND(N)=0
1966: 199 JCNT=JCNT+1
1967: DO 1 N=1,MSOL
1968: IF (MIND(N).NE.0) GO TO 1
1969: 201=CMPLX(HSR(N),HSI(N))
1970: ZH1=-1.0/201
1971: IF (ABS(ZH1).LT.EPS) GO TO 21
1972: IF (N.EQ.MSOL) GO TO 20
1973: DO 2 J=N+1,MSOL
1974: IF (MIND(J).NE.0) GO TO 2
1975: 202=CMPLX(HSR(J),HSI(J))
1976: ZH2=-1.0/202
1977: ZS1=ZH1+ZH2
1978: IF (ABS(ZS1).LT.EPS) GO TO 3
1979: 2 CONTINUE
1980: 20 MIND(N)=2
1981: GO TO 1
1982: 21 MIND(N)=3
1983: GO TO 1
1984: 3 IF (AIMAG(ZH2).GE.0.0) THEN
1985: MIND(J)=-1
1986: MIND(N)=1
1987: ELSE
1988: MIND(J)=1
1989: MIND(N)=-1
1990: END IF
1991: 1 CONTINUE
1992: C
1993: C -----CHECK WRITE-----
1994: C
1995: C -----
```

```

1996: CCC WRITE(KP,700) MSOL
1997: CCC DD 101 N=1,MSOL
1998: CCC Z2=-1.0/CMPLX(HSI(N),HSI(N))
1999: CCC WRITE(KP,701) N,MIND(N),HSI(N),HSI(N),Z2
2000: CCC DD 102 J=1,MSOL
2001: CCC Z1=CMPLX(PR(J,N),PI(J,N))
2002: CCC WRITE(KP,702) Z1,ABS(Z1)
2003: CC102 CONTINUE
2004: CC101 CONTINUE
2005: C
2006: L1=0
2007: DO 22 N=1,MSOL
2008: IF(MIND(N).NE.1) GO TO 22
2009: L1=L1+1
2010: Z1=CMPLX(HSI(N),HSI(N))
2011: ZHS(L1)=-1.0/Z1
2012: DO 40 J=1,MUR+MUZ
2013: 40 ZPHI(J,L1)=CMPLX(PR(J,N),PI(J,N))
2014: DO 41 J=1,MUR+MUZ
2015: 41 P=PR(J+MUZ+MUR,N)
2016: P2=PI(J+MUZ+MUR,N)
2017: 41 ZPHI(J+MUR+MUZ+MUT,L1)=CMPLX(P1,P2)
2018: 22 CONTINUE
2019: C
2020: DO 50 N=1,MSOL
2021: IF(MIND(N).EQ.2) MIND(N)=0
2022: 50 CONTINUE
2023: EPS=EPS*10.0
2024: IF(JCNT.GT.6) GO TO 99
2025: IF(L1.NE.MCHK) GO TO 199
2026: GO TO 100
2027: 99 STOP 'EIGEN CHECK'
2028: 100 CONTINUE
2029: C
2030: C FORMAT
2031: C
2032: CC700 FORMAT(1H , 'NUMBER OF EIGEN VALUE= ',15)
2033: CC701 FORMAT(1H , 'N= ',13.5X,'EIGEN VALUE= ',2E15.5,5X,
2034: CC + 'HAVE NUMBER= ',2E15.5)
2035: CC702 FORMAT(1H , 'E15.5')
2036: RETURN
2037: END
2038: C ****
2039: SURROUN COEFXX(ZN,ZW1,ZW2,NLAY,MLY,ZXXX,ZNXT,2KTT)
2040: C ****
2041: IMPLICIT COMPLEX*16(Z)
2042: DIMENSION ZW1(MLY*4,MLY*4),ZW1(MLY*4),ZW2(MLY*4)
2043: C
2044: ZONE=CMPLX(1.0,0.0)
2045: C
2046: CALL ZEROCL(ZW1,NLAY*4,1,1)
2047: ZW1(1)=ZONE
2048: ZW1(NLAY+1)=ZONE
2049: CALL ZMULT1(ZW1,ZW1,ZW2,NLAY*4,1,NLAY*4,1,1)
2050: ZA=ZW2(1)
2051: ZB=ZW2(NLAY*2+1)
2052: CALL ZEROCL(ZW1,NLAY*4,1,1)

```

```
      ZW1(NLAY*2+1)=ZONE  
      ZW1(NLAY*3+1)=ZONE  
      CALL ZMULT1(ZW,ZW1,ZW2,NLAY*4,NLAY*4,NLAY*4,1,MLY*4,1,1)  
      ZC1=ZW2(1)  
      ZD1=ZW2(NLAY*2+1)  
      ZDIV=ZB1*ZC1-ZA1*ZD1  
      ZXKX=-ZD1/ZDIV  
      ZXXT= ZC1/ZDIV  
      ZXTT= ZA1*ZC1/ZB1/ZDIV  
      RETURN  
END  
2053:  
2054:  
2055:  
2056:  
2057:  
2058:  
2059:  
2060:  
2061:  
2062:  
2063:
```