FINITE ELEMENT MODELLING OF THREE-MODULUS ANISOTROPIC HYPOELASTIC SOIL MEDIA

BY

ROU-LEI ZHANG

A Dissertation

Submitted to The University of Manitoba

in Partial Fulfillment of the Requirements

for the Degree of

MASTER OF SCIENCE

Department of Civil Engineering

University of Manitoba

Winnipeg, Manitoba, Canada.

ISBN 0-315-77950-0

Canada

FINITE ELEMENT MODELLING OF THREE-MODULUS ANISOTROPIC HYPOELASTIC SOIL MEDIA

BY

ROU-LEI ZHANG

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

MASTER OF SCIENCE

ACKNOWLEDGMENTS

I would like to express my deep gratitude to my advisor, Professor R.K.N.D. Rajapakse, and my co-advisor, Professor J. Graham, for their support, encouragement, and guidance during the course of my graduate study at UM.

I convey my sincere thanks to Dr. J.H. Yin for providing much assistance towards the comprehension of the hypoelasticity model used throughout the thesis. Thanks are extended to Mr. B.E. Lingnau for helping with the interpretation of the laboratory data in a few instances, and all of my friends for their encouragement.

This work was made possible by the financial support received in the form of a research assistantship from the NSERC operating grant of Prof. Rajapakse and a teaching assistantship from the Department of Civil Engineering, all of which I greatly appreciate.

September 1992 Rou-Lei Zhang

Winnipeg, Canada.

ABSTRACT

A finite element code is presented for analyzing the stress-strain behavior of a buffer-structure interaction problem raised in the Canadian nuclear fuel waste disposal management program. A three-modulus anisotropic model of hypoelasticity by Yin, Graham and Saadat (1989), is implemented in the code. Generalization is performed to transfer the three modulus constitutive relationships expressed in two principal effective stresses and two principal strains into the multi-axial engineering stress and strain condition. A symmetric stiffness matrix is developed and thus the hypoelasticity model can be added to an existing FEM program without major modification. The finite element iterative scheme is developed from the saturated porous boundary value problem and then reduced to several simplified cases, providing options for different levels of requirements.

The code has been checked using a number of verification examples where either analytical solution or experimental data are available. Excellent agreement between the numerical and analytical results, and satisfactory matching of FEM and Experimental outcome confirm the validity of the finite element code developed.

The nuclear waste disposal vault proposed in the Canadian program is modelled by the numerical code under three possible extreme environmental conditions, namely, sudden high fluid pressure acting on the buffer through a fracture in rock mass; the loading from waste container weight and backfill; and swelling pressures developed and then released due to the contact with natural ground water. Discussions and conclusions are made from these numerical modelling results.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

1.1. GENERAL

Geomechanics and finite element methods (FEM) have exerted a mutual influence in their recent development (Desai and Gioda, 1990). The complexity of many geomechanical systems necessitiates the use of modern numerical methods such as finite element method and its branch approaches (such as boundary element method, finite difference method, etc.). With the almost unlimited power of the current and future computers, these methods can provide extremely powerful tools for analysis and design of engineering systems that are very difficult with the use of conventional methods, often based on closed-form analytical solutions.

The computational power available will be of limited use unless the mechanical response of materials constituting the engineering system is properly defined and implemented. Constitutive modelling of geological materials, which are usually more complex than many other engineering materials plays an important role in robust, consistent and reliable solutions from computational methods. Conversely, implementation of the

models in efficient finite element procedures assumes a vital role towards the solution of practical and realistic engineering problems. With the remarkable advances made in computers, a great many new geomechanical models are developed every year. To implement all of these models in a set of finite element codes is impossible and impractical. Too general approaches sometimes lead to nowhere. Usually it is more pertinent to solve a concrete problem using specific particular solutions. Laboratory models are set up for a stress-strain space in which the principal directions are predetermined and simplified, and the ambient boundary conditions are greatly idealized. Modelling specialists have to be concerned about the agreement of their models with real solutions if they are implemented to predict the performance of engineering systems with multi-dimensional, complex geometry and boundary conditions. There tends to be a gap between laboratory model tests and engineering applications which needs to be filled by predictive numerical analysis. The finite element method provides an efficient, versatile and powerful means to fill this gap. This is what the present work sets out to do. It implements a newly developed nonlinear, anisotropic hypoelastic model using the finite element method to model the geomechanical behavior of a problem raised by the Canadian nuclear fuel waste disposal management program.

1.2. BACKGROUND

The spent fuel from nuclear power plants is strongly toxic and a hazard to the social environment. Today many countries have extensive nuclear waste management programs for disposal of spent fuel (Olivier 1986). The Canadian program is based on the concept that the waste can be isolated effectively and permanently by disposal underground in deep stable geological formations. A compacted mixture of sand and bentonite (known as buffer) is proposed as an effective barrier to limit radionuclide escape to the biosphere.

Bentonites with their swelling property of absorbing water into their crystalline structure, can potentially provide a chemically and mechanically stable environment around the waste canisters (Saadat, 1989). The addition of sand to bentonite provides a high density material with low sensitivity of compaction density to moisture content, and decreases the shrinkage potential of the mixture. The buffer is an engineered mixture of 50/50 dry mass of crushed silica sand and sodium bentonite.

The mechanical behavior of the compacted buffer has been studied extensively by the group led by Dr. J. Graham at the University of Manitoba. Many conclusions and several constitutive models have been made on the basis of experimental observations (Saadat, 1989, Yin 1990). However, little information is available on numerical implementation of these newly developed constitutive models. Numerical analysis of the behavior of the buffer material is essential for assessing the safety for the containers of nuclear fuel waste in resisting groundwater pressures, pressure from swelling of the buffer, any ambient stresses transferred through the buffer from the surrounding rock, and other pressures acting on the container through the buffer.

Buffer mass tests at the Stripa mine in Sweden, and small-scale laboratory experiments, have indicated that the buffer can be expected to become fully saturated after emplacement. It is therefore necessary to study the buffer behavior as a two-phase materail, taking separate account of the solid phase and the liquid phase of the buffer. The research described in this thesis has focused on finite element modelling of saturated soil materials using constitutive relations in the form of a new nonlinear 3-modulus anisotropic hypoelastic model (called the KGJ model). Applications will be made to the rock-buffer-container-backfill problem in the Canadian nuclear fuel waste management program.

1.3. OUTLINE OF PRESENT WORK

The objectives of this research are as follows.

(1) To formulate a rational finite element scheme for tracing the stress-strain response of a two-phase problem representing saturated soil. The constitutive relations for the solid phase are initially expressed in a general form. This allows use of the KGJ model in this thesis, and later use of other nonlinear models.

(2) To formulate the finite element schemes for simple problems of fully drained shear, undrained shear, and decoupling behavior of saturated soil. This can be done by reduction from the general scheme developed in (1).

(3) To translate the original hypoelastic KGJ model which is expressed in two dimensional space ($\{p',\ q\}$ or $\{\epsilon_v,\ \epsilon_s\}$) to engineering six dimensional space, i.e., in $\{\sigma_x, \sigma_y, \sigma_z, \tau_{xy}, \tau_{yz}, \tau_{zx}\}$ or $\{\epsilon_x, \epsilon_y, \epsilon_z, \gamma_{xy}, \gamma_{yz}, \gamma_{zx}\}$ space.

(4) To develop a computer program related to the above formulations. The program is written in FORTRAN and developed under the UNIX operating system. An independent module is designed for the constitutive modelling modifications, and allows addition of further developed models.

(5) To test the finite element code. Numerical results will be compared with results from analytical solutions and experimental data. Results will also be compared with critical state modelling.

(6) To study the stress-strain behavior for rock-buffer-container-backfill interaction by means of the finite element code that has been developed. Special cases will include the conditions of sudden high fluid impact on the buffer through fissures or cracks existing in rock mass; gravity loading of the buffer, canister and backfill; and prescribed displacements due to partial release of swelling pressures in the buffer.

CHAPTER 2

THREE-MODULUS HYPOELASTICITY MODEL AND ITS SYMMETRIC STIFFNESS FORMULATION

After having briefly introduced the 3-modulus (K,G,J) hypoelastic model proposed by Yin, Graham and Saadat (1990), its generalized symmetric form is formulated in this chapter for finite element implementation. The model provides the constitutive relationship between the skeleton stress and the soil deformation for a two-phase porous medium. The formulation is based on the simplified anisotropic elastic theory proposed by Graham and Houlsby (1983) using a 3-modulus elastic model. The material anisotropy appears as the simplest cross-isotropic case, for after some simplifying assumptions, it can be described by only three independent parameters which are functions of mean effective stress, shear stress and volume strain. The numerical implications are discussed for the case where the compliance matrix of the constitutive equation becomes indefinite.

2.1 THE THREE-MODULUS HYPOELASTICITY MODEL

Yin, Graham, Saadat and Azizi (1989) proposed a new hypoelastic model developed mathematically from curves fitted from simple isotropic compression and undrained triaxial shear tests on saturated specimens (Fig. 2.1). The model describes non-linear incremental elasticity. It contains shear-generated volume strains (dilatancy) and the related phenomenon of shear strains arising from changes in mean effective pressure. This model was calibrated for two different densities of the sand-bentonite buffer proposed for nuclear fuel waste containment in Canada, and validated for medium dense sand from China and stiff plastic clay from France. It may also be used for sand-clay mixtures for highway construction in western Canada and for other less active soils. The hypoelasticity approach emphasizes the nonlinear nature of soil response to loading but does not readily include some of the transient behavior (for example, pore-water pressure generation or dissipation) that is perhaps better handled by elastoplasticity. Some criticism for its lack of incremental continuity of response along certain load paths was presented by Mroz (1980). This may be the topic for future discussion.

The model is characterized by three stress-strain dependent modulus functions that are referred to loosely as "moluli". These are a bulk modulus function K, a shear modulus function G and a coupling modulus function J, which relates strains to stresses in the following way

$$\begin{Bmatrix} d\epsilon_v \\ d\epsilon_s \end{Bmatrix} = \begin{bmatrix} 1/K & 1/J \\ 1/J & 1/3G \end{bmatrix} \begin{Bmatrix} dp' \\ dq \end{Bmatrix} \tag{2.1}$$

where

$$p' = \frac{1}{3}(\sigma_1' + 2\sigma_3'), \quad q = \sigma_1' - \sigma_3',$$
$$\epsilon_v = \epsilon_1 + 2\epsilon_3, \quad \epsilon_s = \frac{2}{3}(\epsilon_1 - \epsilon_3) \tag{2.2}$$

with σ_1', σ_3' being the principal effective stresses, and ϵ_1, ϵ_3 the principle strains. In the terminology of soil mechanics, p', q, ϵ_v and ϵ_s are referred to as effective mean stress, de-

viator stress, volume strain and shear strain, respectively (Graham, 1989; Wood, 1990). The hypoelastic model incorporates non-reversibility, non-linearity, dilatancy and the related phenomenon which produces shear strains from mean stress changes. In eq (2.1) the bulk modulus ($K > 0$) represents the volumetric stiffness of the soil with respect to dp'. The shear modulus G ($G > 0$) controls shear deformations with respect to dq. The coupling modulus J accounts both for the volumetric strain produced by an increment in deviator stress, dq, and for the shear strain produced by an increment in mean effective stress, dp'. This is actually a "compliance" approach with $1/K$ representing the volume compliance.

Generally the expressions for the moduli which are developed from curve fitting functions of triaxial tests differ from soil to soil. For the sand-bentonite buffer as well as the Paris stiff clay, Yin et al (1990) showed that they can be obtained from the following three elemetary curve fitting functions fitted to undrained shear data. (Note: In this thesis, some modifications to the original expressions have been made after personal contact with Prof. Graham and Dr. Yin. These changes are listed in Table 2.1 and 2.2.)

(i) $p'_{cons} - \epsilon_v$ relationship from isotropic compression tests,

$$\epsilon_v = \lambda/V_i \ln(p'_{cons}) + \epsilon_{v_0} \tag{2.3}$$

(ii) $q - p'$ relation from undrained triaxial compression tests (CIŪ-tests), with pore-water pressure measurement,

$$q/p'_{cons} = A(1 - p'/p'_{cons})^n \tag{2.4}$$

(iii) $q - \epsilon_s$ relation from CIŪ tests,

$$q/p'_{cons} = \epsilon_s/(E + F\epsilon_s) \tag{2.5}$$

In the above equations λ/V_i, ϵ_{v_0}, A, n, E and F are curve fitting constants, and p'_{cons} is the consolidation mean effective stress. Table 2.3 references the curve-fitting constants

Table 2.1. Reference changes needed to Yin, Saadat and Graham (1990)

to perform calculations in this thesis

Original	This thesis	Change for	Write
Eq(11)	Eqs(2.10),(2.14)	V	V_i
Eq(14)	Eq(2.11)	K^e	K
Eq(15)	Eq(5.1)	$80cp'$	$13.7cp'$
Eq(21)		κ/V_i	λ/V_i
Eqs(22)	Eq(5.3)	$\kappa d/(1+d)\lambda c$	$d/(1+d)c$
Eqs(23)		$\kappa d/(1+d)\lambda c$	$d/(1+d)c$
Eq(24)		0.0475	0.277
Table 2	Table 2.3	$n = 0.81$	$n = 0.71$
Fig.5 and Fig.6	Fig.2a and b	p'_{cons}	p'_i
7 lines below eq(11)		$K/K^e = 5.8$	$K/K^e = 0.1713$
3 lines above eq(15)		$K^e/p'_{cons} = 80$	$K/p' = 13.7$
2 lines below eq(25)		$K^e = 80p'$	$K = 13.7p'$
5 lines below eq(20)		$K^e = 80p'$	$K = 13.7p'$

Table 2.2. Reference changes needed to Yin (1990)

to perform calculations in this thesis

Original	This thesis	Change for	Write
Eq(4.13)	Eq(2.15)	$(6867 + 39.4p')^2/6867$	$(R + Sp')^2/R$
Eq(4.15)	Eq(2.18)	$U^2 - 2UU_{ult} - U_0U_{ult}$	$T(U^2 - 2UU_{ult} + U_0U_{ult})$
Eq(4.17)	Eq(2.18)	$(\sigma'_3)^m/V$	$(\sigma'_3)^m/3V$
1 line below eq(4.14)	Eq(2.9)	$T = 0.0015$	$T = 0.0015(\text{kPa})^{m-1}$
		$T_0 = 7$	$T_0 = 7(\text{kPa})^{1-m}$
		$U_{ult} = 10$	$U_{ult} = 10(\text{kPa})^{1-m}$

developed for two types of sand-bentonite soils and the Paris clay. For Wuhan sand, which was tested in drained shear, the three elementary relations are replaced by

(i) $\epsilon_v - p'_{cons}$, from isotropic compression,

$$\epsilon_v = p'_{cons}/(R + Sp'_{cons}) \tag{2.6}$$

(ii) $q - \epsilon_v$ behavior, from drained triaxial compression (CID) tests,

$$\epsilon_v = T(U - U_0)U/(U - U_{ult}) \tag{2.7}$$

(iii) $q - \epsilon_s$ behavior, from CID tests,

$$q/(\sigma'_3)^m = \epsilon_s/(V + W\epsilon_s) \tag{2.8}$$

where R, S, T, U_0, U_{ult}, m, V, and W are curve fitting constants which are

$$\begin{gathered} R = 6867.0(\text{kPa}), \quad S = 39.4, \quad T = 0.0015(\text{kPa})^{1-m}, \\ U_0 = 7.0(\text{kPa})^{m-1}, \quad U_{ult} = 10.0(\text{kPa})^{m-1}, \quad m = 0.85, \\ V = 0.001(\text{kPa}^{m-1}), \quad W = 0.10(\text{kPa}^{m-1}) \end{gathered} \tag{2.9}$$

and $U = q/(\sigma'_3)^m$.

The three curve fitting functions allow the moduli K, G, J to be developed mathematically from eq (2.1). The moduli are functions of the measured quantities of p', q, ϵ_v, and have different expressions for first-time loading, and unloading/reloading expressed in terms of unit strain energy. In the first-time loading range, they depend on p', q, ϵ_v and a small number of experimental constants.

For the sand-bentonite soils and Paris stiff clay, the expressions for K, G and J are given by

$$K_f = (V_\lambda)\, p' \tag{2.10}$$

$$J_f = Kc(p'_{cons}/q)^d \tag{2.11}$$

$$G_f = \frac{D_f J_f^2}{J_f^2 + 3 D_f K_f} \tag{2.12}$$

with

$$\begin{aligned} c = nA^{\frac{1}{n}}, \quad d = \frac{1-n}{n}, \quad V_\lambda = V_i/\lambda \\ D_f = \frac{p'_{cons}}{3E}\left(1 - F\frac{q}{p'_{cons}}\right)^2 \\ p'_{cons} = \exp[V_\lambda(\epsilon_v - \epsilon_{v_0})] \end{aligned} \tag{2.13}$$

where the subscript f denotes first-time loading. The unloading/reloading behavior of the soil is assumed isotropically elastic. Thus there is no dilatancy nor anisotropy for the unloading/reloading stage. The moduli are functions of p' and ϵ_v only

$$K_e = (V_i/\kappa)\, p', \quad J_e \to \infty, \quad G_e = D_e = s p'_{cons} \tag{2.14}$$

where κ/V_i and s are experimental constants obtained from the triaxial tests, and are shown in Table 2.3.

Table 2.3. Material constants

Dry density(Mg/m^3)	λ/V_i	κ/V_i	ϵ_{v_0}	A	n	E	F	s
1.50	0.073	0.0125	-0.003	1.32	0.71	0.0086	2.15	37.5
1.66	0.052	–	-0.039	1.25	0.65	0.0108	1.83	–
Paris clay	0.029	–	-0.172	-8.93	1.0	0.0039	0.92	–

For Wuhan sand, the coupling modulus J and shear modulus G were developed by means of CID tests. Taking into account the bulk modulus K developed from isotropic compression, J and G can be expressed as

$$K_f = (R + S\, p')^2/R, \tag{2.15}$$

$$J_f = 3KJ^e/(3K - J^e) \tag{2.16}$$

$$G_f = 3G^e J/(3J - 3G^e) \tag{2.17}$$

where

$$J^e = \frac{(\sigma_3')^m (U - U_{ult})^2}{T(U^2 - 2UU_{ult} + U_0 U_{ult})}$$

$$G^e = \frac{(\sigma_3')^m}{3V}[1 - Wq/(\sigma_3')^m]^2 \tag{2.18}$$

Appropriate moduli are chosen depending on whether the changes of stress state relates to first-time loading or unloading/reloading. The distinction of the state is made on the basis of the total unit strain energy (Yin et al, 1990) which can be expressed as

$$\mathcal{F} = \mathcal{F}(\sigma_{ij}, \epsilon_{ij}) = \frac{1}{2}\int \sigma_{ij}\, \dot{\epsilon}_{ij} dt \tag{2.19}$$

Where dt is an increment of time. Let $\mathcal{F}$ be the current total unit strain energy and $\mathcal{F}^*$ be the highest level of total unit strain energy previously counted. If $\mathcal{F} \geq \mathcal{F}^*$ the stress state is said to be in the first-time loading stage and the moduli K_f, J_f, G_f, e.g, eq (2.10), are to be used. On the countrary if $\mathcal{F} < \mathcal{F}^*$, the unloading/reloading takes place and the moduli K_e, J_e, G_e in eq (2.14) go into effect. This may be expressed as a criterion in terms of:

$$\mathcal{F} = \begin{cases} \geq \mathcal{F}^*, & \text{first-time loading} \\ < \mathcal{F}^*, & \text{unloading/reloading} \end{cases} \tag{2.20}$$

This is an analogue to the plastic loading criterion used primarily in plasticity. (Note that the neutral case when $\mathcal{F} = \mathcal{F}^*$ is defined as first-time loading in this work, following the practice in other branches of geotechnical engineering.) Other criteria, e.g. on the basis of previous higher values of p' or q, may be adopted (see, for example, Yin et al 1988). However the criterion defined in energy is considered superior (Mroz 1980; Duncan 1981). The total unit strain energy expressed in eq (2.19) will be computed on average over an element in the finite element approximation, and the maximum element

strain energy will be updated at each load step.

2.2 NUMERICAL IMPLEMENTATION

The KGJ model deals with the non-linearity, loading/unloading, anisotrpy, expansion during shear and shear strain accompanied in mean stress changes by the three moduli K, G, J. These three moduli can be determined either from routine undrained CIŪ triaxial tests or drained CID tests, depending on whichever is more conveniently available. These merits make the model very attractive. However, from the point of view of numerical work, two major problems raised by the new model have to be considered. The first one is that although the matrix of the model appears to be symmetric, it may not generate a symmetric compliance or stiffness matrix in six dimensional engineering stress-strain space (Yin,1990, Yin, Graham, Saadat and Azizi, 1989). In fact the engineering stiffness generalized from KGJ model is not unique. This problem will be dealt with in next section. The other problem concerns the definiteness of the stiffness matrix which will be formulated later. The necessary and sufficient conditions for positive definiteness of matrix in eq (2.1) requires that all principal minors of the matrix must be greater than zero; that is

$$\frac{1}{K} > 0 \text{ and } \frac{J^2 - 3GK}{3GKJ^2} > 0 \tag{2.21}$$

For most engineering soil foundations it may be pertinent to assume that the effective stress $p' > 0$ and $p'_{cons} > 0$. In the discussion in last section it can reasonably be inferred that $K > 0$ and $G \geq 0$. Therefore it is seen that if $J^2 > 3GK$ and $G > 0$, then the matrix of eq (2.1) remains positive definite. Otherwise it is indefinite. This is evident when the soil is stressed towards critical state or the multiplication of bulk modulus and shear modulus is increased beyond a certain amount. The possibility of

an indefinite matrix implies that some of the most popular solution algorithms, such as the Choleski method, may not always be applied to finite element implementation. Other alternatives, for instance Gauss-Seidel iteration scheme, should be applied. One more numerical consideration is the singularity existing in the KGJ model when $G \rightarrow 0$. For example the matrix in eq (2.1) becomes singular if $q \rightarrow p'_{cons}/F$ for the buffer soil, see eqs (2.11) and (2.12). Fortunately in compressive soils, this phenomenon generally means physically that the soil reaches its critical state (that is steady state failure) and plastic flow takes place.

2.3 ENGINEERING FORM OF THE MODEL

To make the new model applicable for use with finite element approximations, Yin et al (1989) extended it to the engineering form by

$$\begin{Bmatrix} d\epsilon_{11} \\ d\epsilon_{22} \\ d\epsilon_{33} \\ d\epsilon_{12} \\ d\epsilon_{23} \\ d\epsilon_{31} \end{Bmatrix} = \begin{bmatrix} a_1+2b_1 & a_2+b_1+b_2 & a_2+b_1+b_3 & c_1 & c_2 & c_3 \\ a_2+b_1+b_2 & a_1+2b_2 & a_2+b_2+b_3 & c_1 & c_2 & c_3 \\ a_2+b_1+b_3 & a_2+b_2+b_3 & a_1+2b_3 & c_1 & c_2 & c_3 \\ c_1/2 & c_2/2 & c_3/2 & 1/2G & 0 & 0 \\ c_1/2 & c_2/2 & c_3/2 & 0 & 1/2G & 0 \\ c_1/2 & c_2/2 & c_3/2 & 0 & 0 & 1/2G \end{bmatrix} \begin{Bmatrix} d\sigma'_{11} \\ d\sigma'_{22} \\ d\sigma'_{33} \\ d\sigma'_{12} \\ d\sigma'_{23} \\ d\sigma'_{31} \end{Bmatrix} \tag{2.22}$$

where

$$a_1 = \frac{1}{9K} + \frac{1}{3G}, \quad a_2 = \frac{1}{9K} - \frac{1}{6G},$$

$$b_1 = \frac{2\sigma'_{11} - \sigma'_{22} - \sigma'_{33}}{6qJ}, \quad b_2 = \frac{2\sigma'_{22} - \sigma'_{11} - \sigma'_{33}}{6qJ},$$

$$b_3 = \frac{2\sigma'_{33} - \sigma'_{11} - \sigma'_{22}}{6qJ}, \quad c_1 = \frac{\sigma'_{12}}{qJ}, \quad c_2 = \frac{\sigma'_{23}}{qJ}, \quad c_3 = \frac{\sigma'_{31}}{qJ}$$

Unfortunately the constitutive matrix is not symmetrical, and thus is not convenient for implementation in existing finite element programs. However, this problem may be solved using the following approach.

Graham and Houlsby (1983) proposed a $\bar{K}$, $\bar{G}$, $\bar{J}$ linear elasticity model for describing the transverse isotropy of soils as measured in the triaxial test, which may be written as

$$\begin{Bmatrix} dp' \\ dq \end{Bmatrix} = \begin{bmatrix} \bar{K} & \bar{J} \\ \bar{J} & 3\bar{G} \end{bmatrix} \begin{Bmatrix} d\epsilon_v \\ d\epsilon_s \end{Bmatrix} \tag{2.23}$$

Note that the hypoelastic compliance moduli in eq (2.1) are not directly comparable to the elastic stiffness moduli in above equation but can be related to them by inversion of

$$\bar{K} = \frac{1}{1-\xi}K, \quad \bar{J} = \frac{\xi J}{\xi - 1}, \quad \bar{G} = \frac{G}{1-\xi} \tag{2.24}$$

with

$$\xi = \frac{3GK}{J^2} \tag{2.25}$$

The essential difference between Graham, Houlsby' model (2.23) and Yin et al's model (2.1) is that the latter describes the transversely hypoelastic isotropy of soils by functioning $\bar{K}$, $\bar{G}$ and $\bar{J}$ to $K(p', q, \epsilon_v)$, $G(p', q, \epsilon_v)$ and $J(p', q, \epsilon_v)$ while the former treats them as elastic constants. The former model is a special case of the latter one. Therefore some of the capabilities of the elastic model should be held as a limiting capability for the establishment of stress-strain relations using the hypoelastic model. From eq (2.25), it is seen that if $J \to \infty$, then $\xi \to 0$ and

$$\lim_{J\to\infty} \bar{J} = 0 \tag{2.26}$$

Generally, anisotropy is expected if J varies within limited real number domain.

A transversely isotropic material possesses an axis of symmetry in the sense that its properties are independent of rotation about the axis. The behavior of a transversely isotropic elastic material may be described by five parameters which may be shown as

$$\begin{Bmatrix} d\sigma'_{11} \\ d\sigma'_{22} \\ d\sigma'_{33} \\ d\sigma'_{23} \\ d\sigma'_{31} \\ d\sigma'_{12} \end{Bmatrix} = \begin{bmatrix} A & B & B & & & \\ B & C & D & & & \\ B & D & C & & & \\ & & & C-D & & \\ & & & & F & \\ & & & & & F \end{bmatrix} \begin{Bmatrix} d\epsilon_{11} \\ d\epsilon_{22} \\ d\epsilon_{33} \\ d\epsilon_{23} \\ d\epsilon_{31} \\ d\epsilon_{12} \end{Bmatrix} \tag{2.27}$$

where zero terms have been omitted for clarity. (The "A" in eq (2.27) is different from "A" in Table 2.3. Both have appeared in earlier publications and have been retained here for consistency.) Returning to the calibration of Yin et al's model, the triaxial tests were carried out on transversely isotropic cylindrical specimens with the axis of symmetry of the material properties corresponding to vertical axis of the specimen. The tests applied no shear stresses (σ'_{12}, σ'_{23}, σ'_{31}) nor were the shear strains (ϵ_{12}, ϵ_{23}, ϵ_{31}) measured. So stress-strain relationship may be described as

$$\begin{Bmatrix} d\sigma'_{11} \\ d\sigma'_{22} \\ d\sigma'_{33} \end{Bmatrix} = \begin{bmatrix} A & B & B \\ B & C & D \\ B & D & C \end{bmatrix} \begin{Bmatrix} d\epsilon_{11} \\ d\epsilon_{22} \\ d\epsilon_{33} \end{Bmatrix} \tag{2.28}$$

It may be assumed that all horizontal stresses and all horizontal strains were equal, thus $\sigma'_{22} = \sigma'_{33}$ and $\epsilon_{22} = \epsilon_{33}$. Therefore the matrix may be further reduced to

$$\begin{Bmatrix} d\sigma'_{11} \\ d\sigma'_{33} \end{Bmatrix} = \begin{bmatrix} A & 2B \\ B & C+D \end{bmatrix} \begin{Bmatrix} d\epsilon_{11} \\ d\epsilon_{33} \end{Bmatrix} \tag{2.29}$$

The matrix is unsymmetric. In a triaxial test only the 2×2 matrix in eq (2.29) can be investigated. Since the elastic properties C and D only appear as their sum in this equation they can not be investigated separately. Thus if $H = C + D$ only three hypoelastic properties A, B and H may be found from triaxial tests on vertically cut samples. Because testing programmes commonly examine only vertically cut samples, calculations for general stress states require assumptions to be made regarding the relationship between the five parameters in eq (2.27). In order to do this, consider first an isotropic material, for which the stiffness matrix may be written:

$$\begin{Bmatrix} d\sigma'_{11} \\ d\sigma'_{22} \\ d\sigma'_{33} \end{Bmatrix} = \begin{bmatrix} A & B & B \\ B & A & B \\ B & B & A \end{bmatrix} \begin{Bmatrix} d\epsilon_{11} \\ d\epsilon_{22} \\ d\epsilon_{33} \end{Bmatrix} \tag{2.30}$$

This is the case when $C = A$ and $D = B$ in eq (2.29). Houlsby (1981) assumed that the anisotropy can be expressed by multiplying the stiffness coefficients in isotropic matrix (2.30) to increase the stiffness in a horizontal direction by an anisotropy factor α. While

preserving the symmetry of the matrix, this may be achieved by multiplying the second row, second column and the third row and third column by α to give the form

$$\begin{Bmatrix} d\sigma'_{11} \\ d\sigma'_{22} \\ d\sigma'_{33} \end{Bmatrix} = \begin{bmatrix} A & \alpha B & \alpha B \\ \alpha B & \alpha^2 A & \alpha^2 B \\ \alpha B & \alpha^2 B & \alpha^2 A \end{bmatrix} \begin{Bmatrix} d\epsilon_{11} \\ d\epsilon_{22} \\ d\epsilon_{33} \end{Bmatrix} \tag{2.31}$$

Houlsby's assumption was proved to represent reasonable hypotheses about certain aspects of soil behavior (Graham and Houlsby, 1983). When $\alpha = 1$ the material is isotropic. For $\alpha > 1$ the material is stiffer horizontally than vertically and for $\alpha < 1$ the material is stiffer vertically than horizontally. The factor α^2 is the ratio of the direct-stiffnesses in the horizontal and vertical directions, i.e., the ratio of the second and first terms on the leading diagonal of the stiffness matrix in eq (2.28).

The full 6×6 stiffness matrix with three independent parameters can be expressed, by the rule that from the matrix for isotropic material, for every occurrence of 2 or 3 in the labels identifying the row and column of the matrix entry, that entry is multiplied by $\sqrt{\alpha}$, as

$$\begin{Bmatrix} d\sigma'_{11} \\ d\sigma'_{22} \\ d\sigma'_{33} \\ d\sigma'_{23} \\ d\sigma'_{31} \\ d\sigma'_{12} \end{Bmatrix} = \mathcal{E} \begin{bmatrix} 1-\nu & \alpha\nu & -\alpha/\nu & & & \\ \alpha/\nu & \alpha^2(1-\nu) & \alpha^2\nu & & & \\ \alpha\nu & \alpha^2(1-\nu) & \alpha^2\nu & & & \\ & & & \alpha^2(1-2\nu) & & \\ & & & & \alpha(1-2\nu) & \\ & & & & & \alpha(1-2\nu) \end{bmatrix} \begin{Bmatrix} d\epsilon_{11} \\ d\epsilon_{22} \\ d\epsilon_{33} \\ d\epsilon_{23} \\ d\epsilon_{31} \\ d\epsilon_{12} \end{Bmatrix} \tag{2.32}$$

where $\mathcal{E} = E/(1+\nu)(1-2\nu)$. This formula can be inverted and expressed as a compliance matrix

$$\begin{Bmatrix} d\epsilon_{11} \\ d\epsilon_{22} \\ d\epsilon_{33} \\ d\epsilon_{23} \\ d\epsilon_{31} \\ d\epsilon_{12} \end{Bmatrix} = \frac{1}{E} \begin{bmatrix} 1 & -\nu/\alpha & \nu\alpha & & & \\ -\nu\alpha & 1/\alpha^2 & -\nu/\alpha^2 & & & \\ -\nu/\alpha & -\nu/\alpha^2 & 1/\alpha^2 & & & \\ & & & (1+\nu)/\alpha^2 & & \\ & & & & (1+\nu)/\alpha & \\ & & & & & (1+\nu)/\alpha \end{bmatrix} \begin{Bmatrix} d\sigma'_{11} \\ d\sigma'_{22} \\ d\sigma'_{33} \\ d\sigma'_{23} \\ d\sigma'_{31} \\ d\sigma'_{12} \end{Bmatrix} \tag{2.33}$$

The five parameters more commonly used to describe a transversely isotropic material are often chosen as E_v, E_h, ν_{vv}, ν_{vh} and G_{vh}, and are given in the following relation

$$\begin{Bmatrix} d\epsilon_{11} \\ d\epsilon_{22} \\ d\epsilon_{33} \\ d\epsilon_{23} \\ d\epsilon_{31} \\ d\epsilon_{12} \end{Bmatrix} = \begin{bmatrix} 1/E_v & -\nu_{vv}/E_v & -\nu_{vv}/E_v & & & \\ -\nu_{vv}/E_v & 1/E_h & -\nu_{vh}/E_h & & & \\ -\nu_{vv}/E_v & -\nu_{vh}/E_h & 1/E_h & & & \\ & & & (1+\nu_{vh})/E_h & & \\ & & & & 1/2G_{vh} & \\ & & & & & 1/2G_{vh} \end{bmatrix} \begin{Bmatrix} d\sigma'_{11} \\ d\sigma'_{22} \\ d\sigma'_{33} \\ d\sigma'_{23} \\ d\sigma'_{31} \\ d\sigma'_{12} \end{Bmatrix} \tag{2.34}$$

Comparing eqs (2.33) and (2.34), it can be established that $E_v = E$, $E_h = \alpha^2 E$, $\nu_{vv} = \nu/\alpha$, $\nu_{vh} = \nu$ and $2G_{vh} = \alpha E/(1+\nu)$. Therefore the 3-modulus model is a particular form of transversely isotropic models with only three independent parameters.

The three sets of parameters, (A, B, H), (K, G, J) and (E, ν, α) can be related to each other by the following procedure. Eq (2.1) can be rewritten by substituting eq (2.2) in components of stresses and strains as

$$\begin{aligned} d\epsilon_{11} + 2d\epsilon_{33} &= \frac{J+3K}{3KJ} d\sigma'_{11} + \frac{2J-3K}{3KJ} d\sigma'_{33} \\ \frac{2}{3}d\epsilon_{11} - \frac{2}{3}d\epsilon_{33} &= \frac{G+J}{3GJ} d\sigma'_{11} + \frac{2G-J}{3GJ} d\sigma'_{33} \end{aligned} \tag{2.35}$$

Since in the triaxial tests all horizontal stresses and horizontal strains are equal, eq (2.31) can be rewritten as

$$d\sigma'_{11} = A d\epsilon_{11} + 2\alpha B d\epsilon_{33}$$

$$d\sigma'_{33} = \alpha B d\epsilon_{11} + \alpha^2 (A+B) d\epsilon_{33}$$

Substituting the above equation into eq (2.21) leads to

$$\begin{aligned} (3KJ)d\epsilon_{11} + (6KJ)d\epsilon_{33} &= [(J+3K)A + (2J-3K)B\alpha] d\epsilon_{11} \\ &+ [2(J+3K)\alpha B + (2J-3K)(A+B)\alpha^2]\, d\epsilon_{33} \end{aligned} \tag{2.36}$$

and

$$(2GJ)d\epsilon_{11} - (2GJ)d\epsilon_{33} = [(G+J)A + (2G-J)\alpha B]d\epsilon_{11}$$

$$+[2(G+J)\alpha B + (2G-J)(A+B)\alpha^2]\, d\epsilon_{33} \tag{2.37}$$

Comparing the coefficients of incremental strains on both sides of eqs (2.36) and (2.37), yields

$$\begin{aligned} 3KJ &= A(J+3K) + \alpha B(2J-3K) \\ 6KJ &= 2\alpha B(J+3K) + \alpha^2(A+B)(2J-3K) \\ 2GJ &= A(G+J) + \alpha B(2G-J) \\ -2GJ &= 2\alpha B(G+J) + \alpha^2(A+B)(2G-J) \end{aligned} \tag{2.38}$$

The solution of equations (2.38) converts the K, G, J moduli parameters to

$$\begin{aligned} A &= \frac{4GKJ - KJ^2 - 4GJ^2/3}{3GK - J^2} \\ B &= \frac{3KJ - A(J+3K)}{(2J-3K)\alpha} \\ \alpha &= \frac{-HF \pm \sqrt{(FH)^2 + 8AH[3KJ - F(J+3K)]}}{2AH} \end{aligned} \tag{2.39}$$

where

$$H = 2J - 3K, \quad F = [3KJ - A(J+3K)]/H \tag{2.40}$$

Since in the special case of isotropic problems, $J \to \infty$, manipulation of the last equation of eq (2.39) results in

$$\alpha \to \frac{-F \pm (F+2A)}{2A} \tag{2.41}$$

Because the hypoelastic cross-isotropic model must include the case where the material is isotropic when $\alpha \to 1$, the only possible sign of '$\pm$' in eq (2.41) is positive. Hence

$$\alpha = \frac{\sqrt{(FH)^2 + 8AH[3KJ - F(J+3K)]} - HF}{2AH} \tag{2.42}$$

The first two equations in eq (2.39) as well as eq (2.42) are used in the first-loading stage when K, G, J are computed from eqs (2.10) through (2.13). For the unloading-reloading stage when K, G, J are replaced by K_e, G_e with $J_e \to \infty$, the following relations can be found through a similar procedure which gives

$$\begin{aligned} A_e &= K_e + \frac{4}{3}G_e \\ B_e &= \left(K_e - \frac{2}{3}G_e\right)/\alpha_e \\ \alpha_e &= 1 \end{aligned} \tag{2.43}$$

where the subscript e denotes the unloading-reloading stage. The relationships between (K, G, J) and (E, ν, α) can be established through (A, B, α) as follows

$$\begin{aligned} E &= \frac{(1+\nu)(1-2\nu)}{1-\nu}A \\ \nu &= \frac{B}{(A+B)} \end{aligned} \tag{2.44}$$

This relation is useful for computer codes which use E, ν as the isotropic elastic parameters.

In the finite element code developed by the author, engineering stress and strain notations are used, and eq (2.32) can be rewritten for 3-D problem as

$$\begin{Bmatrix} d\sigma'_x \\ d\sigma'_y \\ d\sigma'_z \\ d\tau_{xy} \\ d\tau_{yz} \\ d\tau_{zx} \end{Bmatrix} = \begin{bmatrix} \alpha^2 A & \alpha B & \alpha^2 B & & & \\ \alpha B & A & \alpha B & & & \\ \alpha^2 B & \alpha B & \alpha^2 A & & & \\ & & & \alpha(A-B)/2 & & \\ & & & & \alpha(A-B)/2 & \\ & & & & & \alpha^2(A-B)/2 \end{bmatrix} \begin{Bmatrix} d\epsilon_x \\ d\epsilon_y \\ d\epsilon_z \\ d\gamma_{xy} \\ d\gamma_{yz} \\ d\gamma_{zx} \end{Bmatrix} \tag{2.45}$$

Note here that the y-axis represents the direction of material symmetry. The stresses and strains defined in eq (2.2) for triaxial soil tests are generalized in terms of engineering

stresses and strains to give

$$p' = \frac{1}{3}\left(\sigma_x + \sigma_y + \sigma_z\right), \tag{2.46}$$

$$q = \left\{\frac{1}{2}\left[\left(\sigma_x - \sigma_y\right)^2 + \left(\sigma_y - \sigma_z\right)^2 + \left(\sigma_z - \sigma_x\right)^2 + 6\left(\tau_{xy}^2 + \tau_{yz}^2 + \tau_{zx}^2\right)\right]\right\}^{\frac{1}{2}} \tag{2.47}$$

$$\epsilon_v = \epsilon_x + \epsilon_y + \epsilon_z \tag{2.48}$$

$$\epsilon_s = \left\{\frac{2}{9}\left[\left(\epsilon_x - \epsilon_y\right)^2 + \left(\epsilon_y - \epsilon_z\right)^2 + \left(\epsilon_z - \epsilon_x\right)^2 + \frac{3}{2}\left(\gamma_{xy}^2 + \gamma_{yz}^2 + \gamma_{zx}^2\right)\right]\right\}^{\frac{1}{2}} \tag{2.49}$$

For axisymmetric problem where $d\tau_{r\theta} = d\tau_{y\theta} = 0$, $d\gamma_{r\theta} = d\gamma_{y\theta} = 0$, the relations of eq (2.45) become

$$\begin{Bmatrix} d\sigma'_r \\ d\sigma'_y \\ d\tau_{ry} \\ d\sigma'_\theta \end{Bmatrix} = \begin{bmatrix} \alpha^2 A & \alpha B & 0 & \alpha^2 B \\ \alpha B & A & 0 & \alpha B \\ 0 & 0 & \alpha(A-B)/2 & 0 \\ \alpha^2 B & \alpha B & 0 & \alpha^2 A \end{bmatrix} \begin{Bmatrix} d\epsilon_r \\ d\epsilon_y \\ d\gamma_{ry} \\ d\epsilon_\theta \end{Bmatrix} \tag{2.50}$$

These definitions are identical to the those defined in eq (2.2) under triaxial test conditions.

For plane strain problem, $d\tau_{zx} = d\tau_{zy} = 0$, $d\gamma_{zx} = d\gamma_{zy} = d\epsilon_z = 0$, thus

$$d\sigma'_z = \alpha B(\alpha d\epsilon_x + d\epsilon_y) \tag{2.51}$$

and

$$\begin{Bmatrix} d\sigma'_x \\ d\sigma'_y \\ d\tau_{xy} \end{Bmatrix} = \begin{bmatrix} \alpha^2 A & \alpha B & 0 \\ \alpha B & A & 0 \\ 0 & 0 & \alpha(A-B)/2 \end{bmatrix} \begin{Bmatrix} d\epsilon_x \\ d\epsilon_y \\ d\gamma_{xy} \end{Bmatrix} \tag{2.52}$$

For plane stress problem, $d\sigma'_z = d\tau_{zx} = \tau_{zy} = 0$, $d\gamma_{zx} = d\gamma_{zy} = 0$, therefore

$$d\epsilon_z = -\frac{B}{\alpha A}(\alpha d\epsilon_x + d\epsilon_y) \tag{2.53}$$

and

$$\begin{Bmatrix} d\sigma'_x \\ d\sigma'_y \\ d\tau_{xy} \end{Bmatrix} = \frac{A-B}{A} \begin{bmatrix} \alpha^2(A+B) & \alpha B & 0 \\ \alpha B & (A+B)/2 & 0 \\ 0 & 0 & \alpha A \end{bmatrix} \begin{Bmatrix} d\epsilon_x \\ d\epsilon_y \\ d\gamma_{xy} \end{Bmatrix} \tag{2.54}$$

In all of the previous discussion it is tacitly assumed that the normal stresses and normal strains are positive in compression. This is opposite to the convention commonly used in solid mechanics.

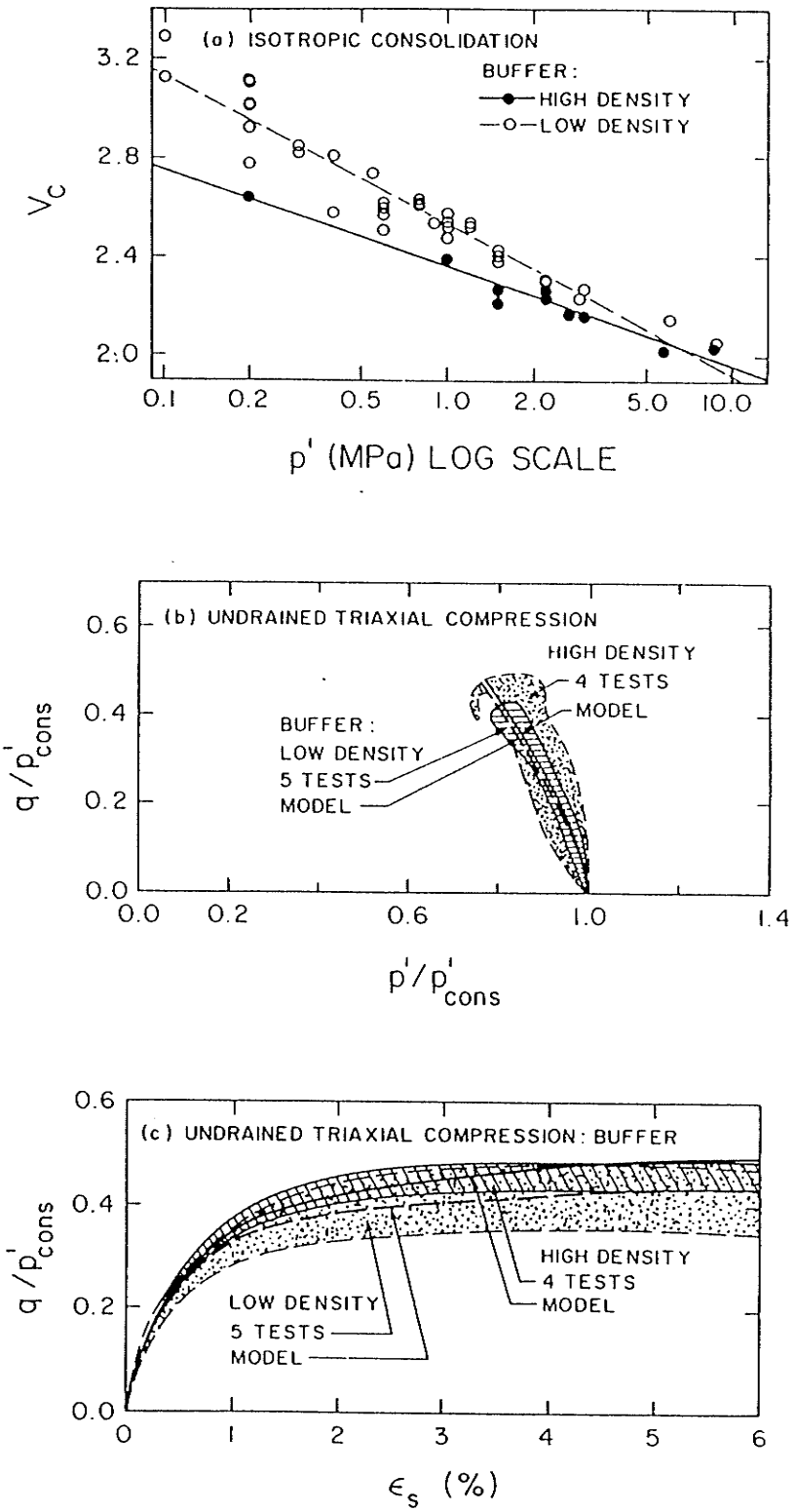


Figure 2.1. Modelling functions for compacted sand-bentonite buffer (Yin, Graham, Saadat and Aziz, 1989). (a) V vs $\log(p')$, (b) q/p'_i vs p'/p'_i, (c) q/p'_i vs ϵ_s.

CHAPTER 3

FINITE ELEMENT ITERATIVE SCHEME FOR TWO-PHASE MATERIAL

The main characteristic of soil under saturated conditions is its porous or two-phase nature: a skeleton of loose (or cemented) particles is surrounded by a fluid (usually water) in which shear stresses are small and which exerts a pressure u on the solid phase. The essential behavioral relationship of such materials with a linear elastic skeleton was formulated by Biot (1941), and Schiffman et al(1969). Schiffman and Chen (1969) and Zienkiewicz et al (1977) extended the formulation to the general non-linear case using finite element discretization. Here the basic governing equations and the finite element formulations involved in the present study will be presented. Following sections are based loosely on the references given previously. It is assumed that mass transformation may be neglected, the deformations are small and the fluid flow is very slow so that all acceleration forces are negligible. The formulation starts from a general case but a number of simplified cases are deduced for practical uses.

3.1 GOVERNING EQUATIONS

EFFECTIVE STRESSES

The total stress state under pure statics may be decomposed into two parts, one of these being the hydrostatic pressure $\mathbf{m}u$ acting externally and internally on the solid skeleton and the second being the unit loads (stress) transferred between the particles as an 'effective' stress. Thus

$$\boldsymbol{\sigma} = \boldsymbol{\sigma}' + \mathbf{m}u \tag{3.1}$$

where $\boldsymbol{\sigma}'$ is the effective stress vector, and $\mathbf{m} = \{1\ 1\ 1\ 0\ 0\ 0\}^{\mathrm{T}}$ is the unit vector. In this context, bold face letters represent matrix or vector quantities, while scalars are printed in italics.

Constitutive Law

Well established soil mechanics principles conclude that it is the effective stress $\boldsymbol{\sigma}'$ which is responsible for all major deformations, linear or non-linear, and indeed that failure states can only be adequately expressed in terms of such effective stresses. With this generally agreed assumption the constitutive relations of the soil skeleton can be written in the incremental form as

$$d\boldsymbol{\sigma}' = \mathbf{D}_{\mathrm{T}}(d\boldsymbol{\epsilon} - d\boldsymbol{\epsilon}^{c} - d\boldsymbol{\epsilon}^{u} - d\boldsymbol{\epsilon}^{\tau}) \tag{3.2}$$

in which $\mathbf{D}_{\mathrm{T}}$ is the tangent modulus matrix, τ represents temperature, and the strains on the right hand side are, in turn, the total incremental strains $d\boldsymbol{\epsilon}$, i.e.,

$$d\boldsymbol{\epsilon} = \mathbf{L}d\mathbf{a} \tag{3.3}$$

where $\mathbf{L}$ is a 6×3 matrix operator for 3-D problems and $\mathbf{a}$ is the displacement vector; the creep strains $d\boldsymbol{\epsilon}^{c}$, which with some generality can be written as

$$d\boldsymbol{\epsilon}^{c} = \mathbf{g}(\boldsymbol{\sigma}')dt \tag{3.4}$$

where $\mathbf{g}(\sigma')$ is a stress dependent vector; the strains due to pore water pressure changes, $d\epsilon^u$, which are generally neglected but in less porous material can be computed as

$$d\epsilon^u = \mathbf{m}\frac{dp}{3K_s} \tag{3.5}$$

in which K_s is the average bulk modulus of the solid phase; the thermal strains, $d\epsilon^\tau$, caused by the temperature change in the soil, which, if are assumed uncoupled with deformation, can ge generally expressed as

$$d\epsilon^\tau = \mathbf{h}(\mathbf{x})\, d\tau \tag{3.6}$$

where $\mathbf{h}(\mathbf{x})$ is a vector dependent on coordinates.

In the above, the modulus matrix $\mathbf{D}_\mathrm{T}$ is defined by the form of the constitutive relationship used and generally will depend on σ, ϵ, $\dot{\epsilon}$, etc. In this work the three-modulus nonlinear model introduced in previous chapter will be applied.

OVERALL EQUILIBRIUM AND EQUILIBRIUM OF FLUID FLOW

For the static state, the overall equilibrium must exist between the total stress gradients and body forces. That is, incremental equilibrium must satisfy

$$\mathbf{L}^\mathrm{T} d\sigma + d\mathbf{b} = 0 \tag{3.7}$$

or

$$\mathbf{L}^\mathrm{T} d\sigma' + \nabla du + d\mathbf{b} = 0$$

where $\mathbf{L}$ is the operator equivalent to strain definition, ∇ the Hamilton operator and $d\mathbf{b}$ the body force vector.

With the hydraulic head h defined as

$$h = z + u/\gamma_w \tag{3.8}$$

where γ_w is the fluid density and z the vertical coordinate, fluid flow velocities can be obtained by Darcy's law as

$$\mathbf{v} = -\mathbf{k}\nabla(h) = -\mathbf{k}\nabla(z + u/\gamma_w) \tag{3.9}$$

In the above, $\mathbf{k}$ is the hydraulic conductivity matrix, which in orthotropic situations can be reduced by a matrix with three independent parameters,

$$\mathbf{k} = \begin{bmatrix} k_x & & \\ & k_y & \\ & & k_z \end{bmatrix} \tag{3.10}$$

where k_x, k_y, k_z are the hydraulic conductivity coefficients in each Cartsian coordinate direction.

The basic statement of continuity of flow requires that the divergence of the flow velocity vector be equal to the rate of fluid accumulation per unit volume of space, i.e., $\nabla^{\mathrm{T}}\mathbf{v} = \textit{rate of fluid accumulation}$. Substituting the velocity expression (3.9) into the left hand side and adding the contribution to the rate of fluid accumulation, leads to

$$-\nabla^{\mathrm{T}}\mathbf{k}\nabla(\gamma_w z + u) = \mathbf{m}^{\mathrm{T}}\frac{\partial \epsilon}{\partial t} + (1-n)\frac{1}{K_s}\frac{\partial u}{\partial t} + \frac{n}{\beta}\frac{\partial u}{\partial t} + \frac{1}{3K_s}\mathbf{m}^{\mathrm{T}}\frac{\partial \boldsymbol{\sigma}'}{\partial t} \tag{3.11}$$

where n is the porosity and β the bulk modulus of the fluid.

The terms on the right hand side of eq (3.11) represent, from the first term to the third respectively, the change of total strain; the change of particle volume due to pressure change; the change of fluid volume contributed by fluid compressibility. The last term is the compression of solid grains due to effective stress changes $\partial\boldsymbol{\sigma}'/\partial t$. This strain is small but is important in rock flow associated with oil technology. Finally, the combined differential equation governing the flow becomes on substitution of eq (3.2)

$$\begin{aligned} &\nabla^{\mathrm{T}}\mathbf{k}\nabla(\gamma_w z + u) + (\mathbf{m}^{\mathrm{T}} + \frac{1}{3K_s}\mathbf{m}^{\mathrm{T}}\mathbf{D}_{\mathrm{T}})\frac{\partial \epsilon}{\partial t} \\ &+ \left[\frac{1-n}{K_s} + \frac{n}{\beta} - \frac{1}{(3K_s)^2}\mathbf{m}^{\mathrm{T}}\mathbf{D}_{\mathrm{T}}\mathbf{m}\right]\frac{\partial u}{\partial t} - \frac{1}{3K_s}\mathbf{m}^{\mathrm{T}}\mathbf{D}_{\mathrm{T}}(\mathbf{g} + \mathbf{h}) = 0 \end{aligned} \tag{3.12}$$

In the above, the sources of perturbations are not considered.

BOUNDARY CONDITIONS

A porous medium occupies a certain domain Ω within a certain boundary Γ of the domain. The boundary Γ may be divided into parts according to the physical nature prescribed. The boundary can be described as composed of a boundary Γ_a on which the displacements or their increments are prescribed as

$$d\mathbf{a} = d\bar{\mathbf{a}} \quad \text{on} \quad \Gamma_a \tag{3.13}$$

and a boundary Γ_t at which the tractions are prescribed, in incremental form, as

$$d\boldsymbol{\sigma}^{\mathrm{T}}\mathbf{n} = \mathbf{t} \quad \text{on} \quad \Gamma_t \tag{3.14}$$

where $\mathbf{n}$ is the unit vector normal to the boundary surface, and $\mathbf{t}$ is a traction vector on the boundary Γ_t.

If we only consider the physical behavior of fluid flow, the boundaries of the soil domain can be divided into a part consisting of pressure (head or potential) conditions on which the pore pressure is given to a known value, i.e.,

$$u = \bar{u} \quad \text{on} \quad \Gamma_u \tag{3.15}$$

and a part of flow boundary conditions where the fluid flux, ϱ, is equal to a known value $\bar{\varrho}(t)$, namely,

$$\nabla(\mathbf{k}u)^{\mathrm{T}}\mathbf{n} = -\bar{\varrho} \quad \text{on} \quad \Gamma_\varrho \tag{3.16}$$

If the finite element formulation is based on the displacement approach where nodal displacements and nodal pore pressures are the basic unkowns, the natural boundary conditions for traction (3.14) and for flow (3.16) will be automatically satisfied. The boundaries for displacement-force may overlap those for pressure-flux, that is

$$\Gamma_a \cup \Gamma_t = \Gamma_u \cup \Gamma_\varrho = \Gamma \tag{3.17}$$

3.2 FINITE ELEMENT FORMULATION IN SOIL SPACE

The system of governing equations given in last section is called '$\mathbf{a}-u$' formulation, for the displacements $\mathbf{a}$ and pore pressure u are the only retained basic variables. This may be seen clearly after the finite element discretization. Using the notation of Zienkiewicz (1977) with $\boldsymbol{\delta}$ and $\mathbf{u}$ defining the nodal displacements and pressure parameters

$$\mathbf{a} = \mathbf{N}\boldsymbol{\delta}, \quad \boldsymbol{\epsilon} = \mathbf{B}\boldsymbol{\delta}, \quad u = \bar{\mathbf{N}}\mathbf{u} \tag{3.18}$$

and for the incremental equilibrium equation in the total incremental stress state

$$\int_{\Omega} \mathbf{B}^{\mathrm{T}}\, d\boldsymbol{\sigma} - d\mathbf{f} = 0 \tag{3.19}$$

where $d\mathbf{f}$ stands for any changes of external forces due to boundary, body or thermal force loading

$$d\mathbf{f} = \int_{\Omega} \mathbf{N}^{\mathrm{T}}\, d\mathbf{b}\, d\Omega + \int_{\Gamma_t} \mathbf{N}^{\mathrm{T}}\, d\mathbf{t}\, dS + \int_{\Omega} \mathbf{N}^{\mathrm{T}}\, \tau\, d\Omega \tag{3.20}$$

Substitution of eqs (3.1), (3.2) and (3.18) into (3.19) gives immediately the incremental equilibrium conditions as

$$\mathbf{K}\frac{d\boldsymbol{\delta}}{dt} + \mathbf{R}\frac{d\mathbf{u}}{dt} - \frac{d\mathbf{f}}{dt} - \mathbf{c} = 0 \tag{3.21}$$

where $\mathbf{K}$ is the tangential stiffness matrix

$$\mathbf{K} = \int_{\Omega} \mathbf{B}^{\mathrm{T}}\mathbf{D}_{\mathrm{T}}\mathbf{B}\, d\Omega \tag{3.22}$$

and

$$\mathbf{R} = \int_{\Omega} \mathbf{B}^{\mathrm{T}}(\mathbf{m} - \mathbf{D}_{\mathrm{T}}\mathbf{m}/3K_s)\bar{\mathbf{N}}\, d\Omega \tag{3.23}$$

$$\mathbf{c} = \int_{\Omega} \mathbf{B}^{\mathrm{T}}\mathbf{D}_{\mathrm{T}}(\mathbf{g} + \mathbf{h})\, d\Omega \tag{3.24}$$

Equation (3.21) is the fundamental one. It is in general non-linear, with the tangential stiffness matrix dependent on stresses and strains which occur at a particular increment.

The pore pressure development is coupled with the strain changes which hence have to be found for the flow conditions.

The finite element discretization of the flow continuity condition of (3.12) can be performed by using the Galerkin procedure or variational principle in a standard way incorporating the flow boundary conditions (Sandha and Wilson, 1969). The discretization using the interpolation expression (3.18), results in

$$\mathbf{Hp} + \mathbf{R}^{\mathrm{T}} \frac{d\boldsymbol{\delta}}{dt} + \mathbf{S}\frac{d\mathbf{u}}{dt} = \bar{\mathbf{f}} \tag{3.25}$$

with

$$\mathbf{H} = \int_{\Omega} (\nabla \bar{\mathbf{N}})^{\mathrm{T}} \mathbf{k} (\nabla \bar{\mathbf{N}})\, d\Omega \tag{3.26}$$

$$\mathbf{S} = \int_{\Omega} \bar{\mathbf{N}}^{\mathrm{T}} \left[\frac{(1-n)}{K_s} + \frac{n}{\beta} - \frac{1}{(3K_s)^2} \mathbf{m}^{\mathrm{T}} \mathbf{D}_{\mathrm{T}} \mathbf{m} \right] \bar{\mathbf{N}}\, d\Omega \tag{3.27}$$

$$\bar{\mathbf{f}} = -\int_{\Omega} \bar{\mathbf{N}}^{\mathrm{T}} \nabla^{\mathrm{T}} \mathbf{k} (\gamma_w \nabla z)\, d\Omega + \int_{\Gamma_e} \bar{\mathbf{N}}^{\mathrm{T}} \bar{\varrho}\, dS - \int_{\Omega} \bar{\mathbf{N}}^{\mathrm{T}} \frac{\mathbf{m}^{\mathrm{T}}}{3K_s} \mathbf{D}_{\mathrm{T}} (\mathbf{g} + \mathbf{h})\, d\Omega \tag{3.28}$$

(details can be found in pp 424-427, 542-543, by Zienkiewicz, 1977) The first term on the right hand side of eq (3.28) exists only for non-linear hydraulic conductivity problems. It vanishes when $\mathbf{k}$ is a constant matrix. Eq (3.27) represents the compressibility term of the soil. If the soil is incompressible, $\mathbf{S} \rightarrow 0$.

Summarizing, the complete coupled behavior of the two phase skeleton-fluid state is governed by a system of ordinary differential equations which can be written concisely as

$$\begin{bmatrix} 0 & 0 \\ 0 & \mathbf{H} \end{bmatrix} \begin{Bmatrix} \boldsymbol{\delta} \\ \mathbf{u} \end{Bmatrix} + \begin{bmatrix} \mathbf{K} & \mathbf{R} \\ \mathbf{R}^{\mathrm{T}} & \mathbf{S} \end{bmatrix} \frac{d}{dt} \begin{Bmatrix} \boldsymbol{\delta} \\ \mathbf{u} \end{Bmatrix} = \begin{Bmatrix} d\mathbf{f}/dt + \mathbf{c} \\ \bar{\mathbf{f}} \end{Bmatrix} \tag{3.29}$$

Displacement boundary conditions (3.13) and pressure boundary conditions (3.15) have both been forced to satisfaction. Natural boundary conditions (3.14) and (3.16) are both automatically satisfied.

The general solution of eq (3.29) can be obtained provided initial conditions of σ_0, u_0 etc. are known. Solution of such transient equations can be accomplished by various

processes of time stepping and provide a complete consolidation history.

3.3 FINITE ELEMENT DISCRETIZATION IN TIME DOMAIN

Equation (3.25) is a first-order differential equation and various numerical schemes may be used to solve it. However, in this work, solutions will be restricted to the simplest types of two-point formulae established from a weighted residual approach.

Proceeding in the usual manner of discretization, with time as the independent variable, the unknowns $\boldsymbol{\delta}$ and $\mathbf{u}$ at any time may be written as

$$\boldsymbol{\delta} = \sum \mathcal{N}_i \boldsymbol{\delta}_i, \quad \mathbf{u} = \sum \mathcal{N}_i \mathbf{u}_i \tag{3.30}$$

where $\boldsymbol{\delta}_i$ (or $\mathbf{u}_i$) stands for a model set of $\boldsymbol{\delta}$ (or $\mathbf{u}$) at time t_i. The shape function $\mathcal{N}_i(t)$ in time domain is taken to be the same for each component of the vectors $\boldsymbol{\delta}$ and $\mathbf{u}$ and therefore $\mathcal{N}_i$ is a scalar.

Consider now a typical time element of length Δt with $\boldsymbol{\delta}_i$ taking on nodal values $\boldsymbol{\delta}_n$ and $\boldsymbol{\delta}_{n+1}$, corresponding to time t_n and t_{n+1} respectively. The interpolation functions and their first time derivatives can be written in terms of local variables as

$$\left.\begin{array}{ll} 0 \le \xi \le 1, & \xi = t/\Delta t \\ \mathcal{N}_n = 1 - \xi, & \dot{\mathcal{N}}_n = -1/\Delta t \\ \mathcal{N}_{n+1} = \xi, & \dot{\mathcal{N}}_{n+1} = 1/\Delta t \end{array}\right\} \tag{3.31}$$

A typical weighted residual equation to eq (3.25) can now be written assuming that the full domain of investigation corresponds with that of one element

$$\int_0^1 w_i [\, \mathbf{R}^{\mathrm{T}} (\dot{\mathcal{N}}_n \boldsymbol{\delta}_n + \dot{\mathcal{N}}_{n+1} \boldsymbol{\delta}_{n+1}) + \mathbf{S} (\dot{\mathcal{N}}_n \mathbf{u}_n + \dot{\mathcal{N}}_{n+1} \mathbf{u}_{n+1})$$

$$+ \mathbf{H} (\mathcal{N}_n \mathbf{u}_n + \mathcal{N}_{n+1} \mathbf{u}_{n+1}) + \bar{\mathbf{f}}] \, d\xi = 0 \tag{3.32}$$

As the problem is an initial value one, one of the parameter sets $\boldsymbol{\delta}_n$, $\mathbf{u}_n$ is assumed known and the equation will serve to determine $\boldsymbol{\delta}_{n+1}$, $\mathbf{u}_{n+1}$ approximately. On inserting

eq (3.31) a recurrence relation can be written as

$$\left(\mathbf{R}^{\mathrm{T}} \int_0^1 w_i d\xi/\Delta t\right) \delta_{n+1} - \left(\mathbf{R}^{\mathrm{T}} \int_0^1 w_i \, d\xi/\Delta t\right) \delta_n$$
$$+ \left(\mathbf{S} \int_0^1 w_i d\xi/\Delta t + \mathbf{H} \int_0^1 w_i \xi d\xi\right) \mathbf{u}_{n+1} - \left(\mathbf{S} \int_0^1 w_i \, d\xi/\Delta t$$
$$-\mathbf{H} \int_0^1 w_i (1-\xi) \, d\xi\right) \mathbf{u}_n + \int_0^1 w_i \bar{\mathbf{f}} \, d\xi = 0 \tag{3.33}$$

In the above matrices $\mathbf{R}$, $\mathbf{S}$ and $\mathbf{H}$ have been assumed to be independent of t within a time step.

Quite generally eq (3.33) can be written for any weighting function as

$$\mathbf{R}^{\mathrm{T}}/\Delta t(\delta_{n+1} - \delta_n) + (\mathbf{S}/\Delta t + \mathbf{H}\theta)\mathbf{u}_{n+1}$$
$$-(\mathbf{S}/\Delta t - \mathbf{H}(1-\theta))\mathbf{u}_n + \hat{\mathbf{f}} = 0 \tag{3.34}$$

where

$$\theta = \int_0^1 w_i \xi \, d\xi \Big/ \int_0^1 w_i \, d\xi \tag{3.35}$$

and

$$\hat{\mathbf{f}} = \int_0^1 w_i \bar{\mathbf{f}} \, d\xi \Big/ \int_0^1 w_i \, d\xi \tag{3.36}$$

The factor θ varies from 0 to 1. It is logical and convenient to assume that the same interpolation is applied to the function $\hat{\mathbf{f}}$ as that used for δ and $\mathbf{u}$, i.e.,

$$\hat{\mathbf{f}} = \theta \bar{\mathbf{f}}_{n+1} + (1-\theta)\bar{\mathbf{f}}_n \tag{3.37}$$

Various possible cases related by varying the transient factor θ were disscussed in Zienkiewicz's work (1977). Correspondingly, $\theta = 0$, $1/2$ and 1 give the forward difference (explicit), mid-difference and backward difference formulaes. If $0 \leq \theta < 1/2$, a criterion for numerical stability must be met for the success of solution. When $\theta \geq 1/2$ the schemes are proved to be unconditionally stable.

Let $\Delta\delta = \delta_{n+1} - \delta_n$, $\Delta\mathbf{u} = \mathbf{u}_{n+1} - \mathbf{u}_n$, on substitution of them with eq (3.34), we have

$$\mathbf{R}^{\mathrm{T}}\Delta\delta + (\mathbf{S} + \mathbf{H}\theta\Delta t)\Delta\mathbf{u} - \Delta\hat{\mathbf{f}} = 0 \tag{3.38}$$

where

$$\Delta\hat{\mathbf{f}} = -[\theta\bar{\mathbf{f}}_{n+1} + (1-\theta)\bar{\mathbf{f}}_n]\Delta t + (\Delta t\,\mathbf{H})\mathbf{u}_n \tag{3.39}$$

Replacing the differential notation "d" in the first expression of eq (3.29) with "Δ" and the second expression with eq (3.38), leads to

$$\begin{bmatrix} \mathbf{K} & \mathbf{R} \\ \mathbf{R}^{\mathrm{T}} & (\theta\Delta t\,\mathbf{H} + \mathbf{S}) \end{bmatrix} \begin{Bmatrix} \Delta\delta \\ \Delta\mathbf{u} \end{Bmatrix} = \begin{Bmatrix} \Delta\mathbf{f} + \mathbf{c}\Delta t \\ \Delta\hat{\mathbf{f}} \end{Bmatrix} \tag{3.40}$$

For some problems, such as incompressibility problem when $\mathbf{S} = 0$ with an explicit scheme when $\theta = 0$ or for an instantaneous phenomenon like an undrained problem, the term $\theta\Delta t\mathbf{H} + \mathbf{S}$ may tend to zero. This makes the direct solution of eq (3.40) difficult. This singlarity problem can be solved by resolving eq (3.40) to a non-singular formula as follows.

3.3. ITERATIVE SCHEMES

Since the stiffness matrix $\mathbf{K}$ is positive definite, from the first set of equations in eq (3.40), it gives

$$\Delta\delta = \mathbf{K}^{-1}(\Delta\mathbf{f} + \mathbf{c}\Delta t - \mathbf{R}\Delta\mathbf{u}) = \mathbf{y} + \mathbf{X}\Delta\mathbf{u} \tag{3.41}$$

where

$$\mathbf{y} = \mathbf{K}^{-1}(\Delta\mathbf{f} + \mathbf{c}\Delta t) \tag{3.42}$$

and

$$\mathbf{X} = \mathbf{K}^{-1}\mathbf{R} \tag{3.43}$$

are constant at the begining of each iterative loop. Substitution of eq (3.41) into the second set of equations in expression (3.40) leads to

$$(\theta\Delta t\mathbf{H} + \mathbf{S} + \mathbf{R}^{\mathrm{T}}\mathbf{X})\Delta\mathbf{u} = \Delta\hat{\mathbf{f}} - \mathbf{R}^{\mathrm{T}}\mathbf{y} \tag{3.44}$$

Let

$$\left.\begin{aligned} \mathbf{A} &= \theta\Delta t\,\mathbf{H} + \mathbf{S} + \mathbf{R}^{\mathrm{T}}\mathbf{X} \\ \Delta\mathbf{q} &= \Delta\hat{\mathbf{f}} - \mathbf{R}^{\mathrm{T}}\mathbf{y} \\ \Delta\mathbf{g} &= \Delta\mathbf{f} + \Delta t\mathbf{c} - \mathbf{R}\Delta\mathbf{u} \end{aligned}\right\} \tag{3.45}$$

then the resolved system of solution can be written as

$$\begin{aligned} &\mathbf{A}(\Delta t)\Delta\mathbf{u} = \Delta\mathbf{q}(\Delta t) \\ &\Delta\mathbf{g} = \Delta\mathbf{f} + \Delta t\mathbf{c} - \mathbf{R}\Delta\mathbf{u} \\ &\mathbf{K}\Delta\delta = \Delta\mathbf{g}(\Delta t) \end{aligned} \tag{3.46}$$

If the two-phase problem under consideration is linear, the solution at each time node can be computed directly by running through eq (3.46) only. The accuracy of the result depends on the numerical strategy adopted and the time interval length used. However, if the problem is considered nonlinear, an iterative algorithm is inevitable within each time step. The most obvious and direct solution procedure, starting from an initial guess $\mathbf{K}_0 = \mathbf{K}(\sigma_0, \epsilon_0)$, may be the *direct iteration scheme* presented below.

Consider just one loop of iteration. Suppose all quantities at the mth iteration are known. Then, starting from the known results at mth iteration, we can proceed to the $(m+1)$th iteration, where m= 0, 1, 2, $\cdots$. Moving through eqs (3.41) to (3.45), the improved approximation is obtained as

$$\begin{aligned} &\mathbf{y}^{(m+1)} = (\mathbf{K}^{-1})^{(m)}(\Delta\mathbf{f}^{(m+1)} + \Delta t\mathbf{c}^{(m)}) \\ &\mathbf{X}^{(m+1)} = (\mathbf{K}^{-1})^{(m)}\mathbf{R} \\ &\mathbf{A}^{(m+1)} = \theta\Delta t\mathbf{H}^{(m)} + \mathbf{S}^{(m)} + \mathbf{R}^{\mathrm{T}}\mathbf{X}^{(m)} \\ &\Delta\mathbf{q}^{(m+1)} = \Delta\hat{\mathbf{f}}^{(m+1)} - \mathbf{R}^{\mathrm{T}}\mathbf{y}^{(m+1)} \\ &\Delta\mathbf{u}^{(m+1)} = (\mathbf{A}^{-1})^{(m+1)}\Delta\mathbf{q}^{(m)} \\ &\Delta\mathbf{g}^{(m+1)} = \Delta\mathbf{f}^{(m+1)} + \Delta t\,\mathbf{c}^{(m)} - \mathbf{R}\Delta\mathbf{u}^{(m+1)} \\ &\Delta\delta^{(m+1)} = (\mathbf{K}^{-1})^{(m)}\Delta\mathbf{g}^{(m+1)} \end{aligned} \tag{3.47}$$

Repetition of the process until the convergence criterion

$$\|\Delta\delta^{(m+1)} - \Delta\delta^{(m)}\| \leq \varrho \tag{3.48}$$

is met, where ϱ is a specified convergence factor.

The formulation (3.47) derived from a direct interation scheme always directly leads to the original form of eq (3.40). And for the cases when the stiffness is not differentiable or the expression is too complicated to get its pertinent derivatives, as for example in the Yin and Graham' model, the direct iteration scheme manifests its peculiar merit. The disadvantage of the scheme is that it may lack convergence for some situations if large increments are used. A variety of alternatives of numerical schemes to the solution of eq (3.40) exist in the literature (Zienkiewicz, 1977). Each has its particular advantages, such as being computationally economic or faster in acceleration of convergence in certain cases. However it is not possible to identify a "best" scheme, for a process that is most economical in one context may be divergent in another. Nevertheless, it is certain that convegence can always be attained by using sufficiciently small increments. To focus attention on the aspect of finite element modelling to porous media with nonlinear constitutive models, the direct iteration approach will be used thoughout this work. No other choice of algorithm will be further discussed.

3.5 PARTICULAR FORMS

FOUNDATION CONSOLIDATION

The general theory of consolidation of an elastic porous medium was introduced by Biot (1941). It is based on a number of simplifying assumptions: (1). Isotropic of the material; (2). Reversibility of stress-strain relations under final equilibrium conditions; (3). Linearity of stress-strain relations; (4). The water contained in the pores is incompressible; (5). Small strains; (6). The deformation due to the change of pore water

pressure is negligible; (7). The water flows through the porous skeleton according to Darcy's law with constant linear permeability. Here we will generalize the solid phase as being anisotropic, nonlinear and irreversible and accept the remainder of the Biot assumptions.

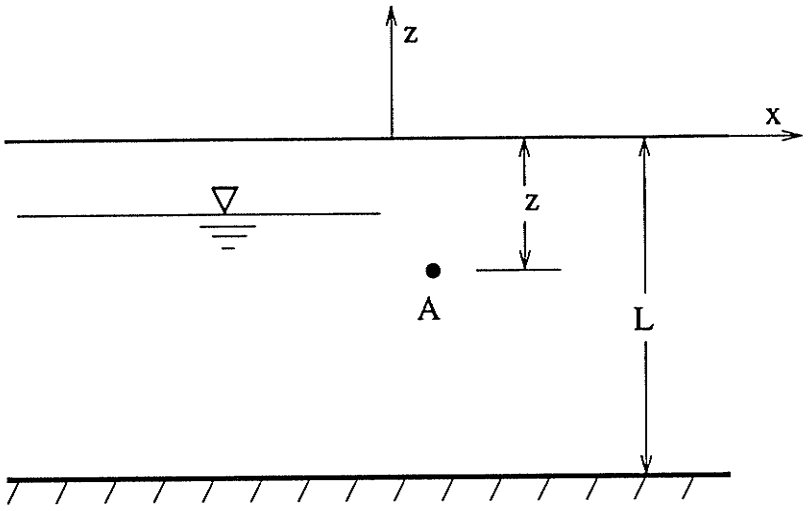


Figure 3.1 Reference frame of hydraulic head

Consider a foundation with the origin of the coordinates fixed on the top of a ground surface and the datum at $z = -l$, as shown in Fig. 3.1. The hydraulic head at any point A in the soil can be written as

$$h = l + z + u/\gamma_w \tag{3.49}$$

The body forces are caused due to gravity of soil

$$\mathbf{b} = \mathbf{b}_0 = \{0,\ 0,\ z\gamma_b\}^{\mathrm{T}} \tag{3.50}$$

where $\gamma_b = \gamma - \gamma_w$ is the buoyant unit weight of soil (γ is the total unit weight, and γ_w is the unit weight of water). During the transient consolidation process, the total pore pressure p can be taken as the superposition of an excessive pore pressure $\bar{u}$ and a

steady state pore pressure u_0, that is

$$u(z,t) = \bar{u}(z,t) + u_0(z) \tag{3.51}$$

in which u_0 is equal to the static water pressure

$$u_0(z) = -\gamma_w z \tag{3.52}$$

Replacing u in equilibrium equation (3.7) with eq (3.51), then

$$\mathbf{L}d\boldsymbol{\sigma}' - \nabla d\bar{u} = 0 \tag{3.53}$$

Substitution of eq (3.51) into the fluid continuity conditions (3.11) results in

$$\nabla^{\mathrm{T}}\mathbf{k}\nabla\bar{u} = \mathbf{m}^{\mathrm{T}}\frac{d\boldsymbol{\epsilon}}{dt} \tag{3.54}$$

where the solid particles and fluid are both assumed incompressible.

Through a similar process similar to that used in previous section, the finite element formula for the consolidation problem becomes

$$\begin{bmatrix} \mathbf{K} & \mathbf{R} \\ \mathbf{R}^{\mathrm{T}} & \theta\Delta t\,\mathbf{H} \end{bmatrix} \begin{Bmatrix} \Delta\boldsymbol{\delta} \\ \Delta\mathbf{u} \end{Bmatrix} = \begin{Bmatrix} \Delta\mathbf{f} \\ \Delta\hat{\mathbf{f}} \end{Bmatrix} \tag{3.55}$$

Note that in this equation, $\Delta u = \Delta\bar{u}$, hence the actual pore water pressure is

$$u = u_0 + \sum \Delta u \tag{3.56}$$

The incremental external forces at time t_n are due to change of tractions on Γ_t,

$$\Delta\mathbf{f}_n = \int_{\Gamma_t} \mathbf{N}^{\mathrm{T}}\Delta\mathbf{t}_n\, dS \tag{3.57}$$

with the initial forces at time t_0 being

$$\mathbf{f}_0 = \int_{\Omega} \mathbf{N}^{\mathrm{T}}\mathbf{b}_0\, d\Omega + \int_{\Gamma_t} \mathbf{N}^{\mathrm{T}}\mathbf{t}_0\, dS \tag{3.58}$$

Similarly, the incremental flux vector $\Delta\hat{\mathbf{f}}$ here is defined the same as eq (3.39) with reduced form for flux $\bar{\mathbf{f}}$ as

$$\bar{\mathbf{f}}_n = \int_{\Gamma_u} \bar{\mathbf{N}}^{\mathrm{T}} \bar{u}_n \, dS, \quad n = 0, 1, 2, 3, \cdots \tag{3.59}$$

If the constant external forces are acted on the foundation with constant flow velocity boundaries, then $\Delta\mathbf{f}_n = 0$ and $\bar{\mathbf{f}}_n = 0$, where $n = 1, 2, 3, \cdots$. From eqs (3.39) and (3.55), the solution of eq (3.40) is therefore reduced to

$$\begin{gathered} (\theta \Delta t \mathbf{H} - \mathbf{R}^{\mathrm{T}} \mathbf{K}^{-1} \mathbf{R}) \Delta \mathbf{u} = -\Delta t \mathbf{H} \mathbf{u}_n \\ \Delta \boldsymbol{\delta} = \mathbf{K}^{-1} \mathbf{R} \Delta \mathbf{u} \end{gathered} \tag{3.60}$$

It is seen that the consolidation is then merely due to the gradual dissipation of excessive pore water pressure in the soil.

FULLY DRAINED BEHAVIOR

Under steady state conditions, all transient behavior has ceased, i.e., $\dot{\boldsymbol{\delta}} = 0$, $\dot{\mathbf{u}} = 0$. In eq (3.29), complete uncoupling of equations occurs and they reduce to

$$\mathbf{H}\mathbf{u} = \bar{\mathbf{f}} \tag{3.61}$$

Now the pressures can be independently determined, provided that the hydraulic conductivity is known and is independent of strain. Eq (3.40) then results in

$$\mathbf{K} \Delta \boldsymbol{\delta} = \Delta \check{\mathbf{f}}, \quad \Delta \check{\mathbf{f}} = \Delta(\mathbf{f} + \mathbf{R}\mathbf{u}) \tag{3.62}$$

In eq (3.62) $\Delta\check{\mathbf{f}}$ is an increment of the resultant of external forces $\mathbf{f}$ and pore pressures $\mathbf{u}$ independent of time at steady state.

Now the solution of the system reduces to the nonlinear solid mechanics problem given in eq (3.62) with the addition of forces $\mathbf{R}\mathbf{u}$ due to pore pressures. The standard

programs and numerical approaches used for nonlinear problems can be applied directly. For a linear elastic skeleton, integration of eq (3.62) yields

$$\mathbf{K}\delta = \mathbf{f} + \mathbf{R}\mathbf{u} = \check{\mathbf{f}} \tag{3.63}$$

Therefore, the consolidation problem can be solved in terms of total quantities if the problem is linear. Otherwise an incremental form of eq (3.62) and must be applied using an iterative process.

UNDRAINED BEHAVIOR

This is an instantaneous phenomenon that can be modelled either by allowing $dt \rightarrow 0$, or by allowing the hydraulic conductivity to tend to zero. If eq (3.29) is multiplied by dt, and $dt \rightarrow 0$, all finite terms disappear. Reforming the equation gives

$$\begin{bmatrix} \mathbf{K} & \mathbf{R} \\ \mathbf{R}^{\mathrm{T}} & \mathbf{S} \end{bmatrix} \begin{Bmatrix} d\delta \\ d\mathbf{u} \end{Bmatrix} = \begin{Bmatrix} d\mathbf{f} \\ 0 \end{Bmatrix} \tag{3.64}$$

Once again, this can be solved using the iteration formula (3.44) by setting Δt to zero. This leads to

$$\begin{aligned} (\mathbf{R}^{\mathrm{T}}\mathbf{X} - \mathbf{S})\Delta\mathbf{u} &= \mathbf{R}^{\mathrm{T}}\mathbf{K}^{-1}\Delta\mathbf{f} \\ \mathbf{K}\Delta\delta &= \Delta\mathbf{f} - \mathbf{R}\Delta\mathbf{u} \end{aligned} \tag{3.65}$$

For the incompressible case when $\mathbf{S} = 0$, eq (3.64) indicates that $\Delta\mathbf{u}$ is in effect the Lagrangian multiplier inserted to ensure the incompressibility. However for the case when $\mathbf{S} \neq 0$ which corresponds to the type of formulation used in solid mechanics when a Poisson's ratio is not equal to 0.5, the solution of eq (3.64) can be further simplified.

Note that if no flow occurs, the continuity equation (3.12) in the absence of creep strain and thermal strains, becomes

$$\left(\mathbf{m}^{\mathrm{T}} + \frac{1}{3K_s}\mathbf{m}^{\mathrm{T}}\mathbf{D}_{\mathrm{T}}\right)\frac{\partial\epsilon}{\partial t} = \mathcal{M}\frac{\partial\mathbf{u}}{\partial t} \tag{3.66}$$

where

$$\mathcal{M} = \frac{1-n}{K_s} + \frac{n}{\beta} - \frac{1}{(3K_s)^2}\mathbf{m}^{\mathrm{T}}\mathbf{D}_{\mathrm{T}}\mathbf{m} \tag{3.67}$$

Substitution of eq (3.66) in eqs (3.1) and (3.2) results in the direct incremental stress-strain relationship

$$d\boldsymbol{\sigma} = d\boldsymbol{\sigma}' + \mathbf{m}d\mathbf{u} = \mathbf{D}_{\mathrm{T}}\, d\boldsymbol{\epsilon} + \left(\mathbf{m} - \frac{1}{K_s} D_{\mathrm{T}}\mathbf{m}\right) d\mathbf{u} = \bar{\mathbf{D}}_{\mathrm{T}}\, d\boldsymbol{\epsilon} \tag{3.68}$$

where

$$\bar{\mathbf{D}}_{\mathrm{T}} = \mathbf{D}_{\mathrm{T}} + \frac{1}{\mathcal{M}}\left(\mathbf{m} - \frac{1}{K_s}\mathbf{D}_{\mathrm{T}}\mathbf{m}\right)\left(\mathbf{m}^{\mathrm{T}} + \frac{1}{3K_s}\mathbf{m}^{\mathrm{T}}\mathbf{D}_{\mathrm{T}}\right) \tag{3.69}$$

Subsequently the deformation can be calculated directly from the solid phase

$$\bar{\mathbf{K}}\Delta\boldsymbol{\delta} = \Delta\mathbf{f} \tag{3.70}$$

where

$$\bar{\mathbf{K}} = \int_{\Omega} \mathbf{B}^{\mathrm{T}}\bar{\mathbf{D}}_{\mathrm{T}}\mathbf{B}\, d\Omega \tag{3.71}$$

Standard iteration approaches for solid mechanics can be used. Again if the problem is linear the solution can be obtained in terms of total quantities.

DECOUPLING ANALYSIS (SEEPAGE)

The finite element formulation (3.40) is in fact formed in a decoupled manner for two-phase material within an incremental step. For computation, the pore pressure is calculated first at each time step, and then the solution for the skeleton deformation follows. Therefore a computer program for a two-phase medium should be conveniently used to deal with uncoupled situations where the requirements of only one phase are needed. For instance, if the hydraulic effect is removed out from eq (3.40), the equation reduces immediately to

$$\mathbf{K}\Delta\boldsymbol{\delta} = \Delta\mathbf{f} + \Delta t\mathbf{c} \tag{3.72}$$

which accounts for the FE formula in solid mechanics. However in this work we will confine our discussion just to one of the essential problems – linear steady state seepage, and leave the rest for further study.

For seepage problem the water head h usually is a variable of more concern than the pore pressure u. At steady state, the flow continuity requirement (3.12) reduces to

$$\nabla^{\mathrm{T}}\mathbf{k}\nabla h = Q \tag{3.73}$$

in which Q is the source flux per unit volume. The boundary condition on Γ_u is replaced by

$$h = \bar{h} \quad on \quad \Gamma_u \tag{3.74}$$

where $\bar{h}$ is the specified head. After some manipulation the finite element formula can be obtained

$$\tilde{\mathbf{H}}\mathbf{u} = \tilde{\mathbf{f}} \tag{3.75}$$

where

$$\tilde{\mathbf{H}} = \int_{\Omega} (\nabla\bar{\mathbf{N}})^{\mathrm{T}}\mathbf{k}(\nabla\bar{\mathbf{N}})\, d\Omega$$

$$\tilde{\mathbf{f}} = \int_{\Gamma_u} \bar{\mathbf{N}}^{\mathrm{T}}\bar{h}\, dS + \int_{\Omega} \bar{\mathbf{N}}^{\mathrm{T}} Q\, d\Omega$$

Comparing eq (3.75) with eq (3.25), it is seen that $\tilde{\mathbf{H}}$ can be obtained by setting $\gamma_w = 1$ in $\mathbf{H}$, and $\tilde{\mathbf{f}}$ can be deduced by replacing $\bar{u}$ by $\bar{h}$ and $\mathbf{m}^{\mathrm{T}}\mathbf{D}_{\mathrm{T}}(\mathbf{g}+\mathbf{h})/3K_s$ by Q, respectively. All the particular forms discussed in this section can be developed from the general formulation (3.46). The program developed for eq (3.46) can therefore be applied to a wide range of reduced problems by specifying particular values to the arguments in eq (3.46).

FURTHER SIMPLIFICATION

For most of foundation problems the compressibility of the solid particles $1/K_s$ is insignificant and can be ignored. Furthermore, if the hydraulic conductivity is assumed linear, the matrices in eqs (3.23), (3.27), (3.28) and (3.69) can be rewritten as

$$\mathbf{R} = \int_{\Omega} \mathbf{B}^{\mathrm{T}}\mathbf{m}\bar{\mathbf{N}}\, d\Omega \tag{3.23'}$$

$$\mathbf{S} = \int_{\Omega} \bar{\mathbf{N}}^{\mathrm{T}} \left(\frac{n}{\beta}\right) \bar{\mathbf{N}}\, d\Omega \tag{3.27'}$$

$$\bar{\mathbf{f}} = \int_{\Gamma_e} \bar{\mathbf{N}}^{\mathrm{T}}\, \bar{\varrho}\, dS \tag{3.28'}$$

$$\bar{\mathbf{D}}_{\mathrm{T}} = \mathbf{D}_{\mathrm{T}} + \left(\frac{\beta}{n}\right) \mathbf{m}\mathbf{m}^{\mathrm{T}} \tag{3.69'}$$

If the flow boundary condition is impermeable in the problem under consideration, then eq (3.28') vanishes. Further simplification can be made if the deformation caused by creep and temperature changes are ruled out. In eq (3.27') $\mathbf{S}$ becomes null if both fluid and solid phases are assumed incompressible.

CHAPTER 4

ON THE DEVELOPMENT OF FINITE ELEMENT CODE

The development of an appropriate finite element program written in FORTRAN for the computational implementation of the KGJ three-moduli model is detailed in this Chapter. The design of the modulus I/O ports allow the range of applicability of the finite element program to be broadend by incorporation of additional models in future when such need arises. An abstract data type (ADT) designed for the current problem is introduced to make the program easy to read and maintain. Effort will be focused on a number of special operations, for example, the implementation of different loading paths, selection of incremental steps, etc. Commonly known finite element operations, such as the establishment of element libraries, will be omitted for brevity. The finite element program will be readily usable for computational analyses of mechanical problems encountered in soil engineering practice. A brief description of the user's manual is provided at the top of the program body to faciliate data input, see Appendix C.

4.1. ADT ORDERED LIST OPERATION

An abstract data type (ADT) is a collection of data values together with a set of operations on those data (Helman and Veroff, 1988). This allows control of the interaction between a program and its data structures. It makes programs easier to design, implement, read and modify. Several levels of ADTs were designed before the development of the finite element increment-iteration algorithm. Here the first two levels of ADTs are presented below. Deeper levels are referred to the program context where the operations are stated straight forwardly. The designed operating hierarchy is shown in Figure 4.1.

MAIN ADT INCREMENTAL OPERATIONS

- CALL DIMEN(···)

  Preset variables associated with dynamic dimensioning to prevent the data from causing a potential crash by too large dimension input.

- CALL INPUT(···)

  Read the computational controlling information, the data defining the geometry of the material domain and the physical properties.

- CALL CHECK(···)

  Check the input data and report any errors detected, including the dimensional governing variables.

- CALL DLENG(···)

  Calculate the diagonal addresses for the spars, half triangular, variable width stiffness matrix stored in one dimensional arry.

- CALL LOADPS(···)

Read in the load information and evaluate the consistent nodal forces for each element.

- *Initialization operation.*

- *Perform incrementally the hypoelastic computation*

  Four sub-operations are performed in this operation:

  1. CALL DLTAF(···)

     Initialize the arrays for incremental stresses, incremental displacements, etc. Form and concoct the incremental load combination.

  2. *Perform the iterate operation*

     This performance is composed of several suboperations which will be discribed later.

  3. CALL UPDPSTR(···)

     Update the displacements, energies and the stresses from the newly computed incremental results.

  4. CALL OUTPUT(···)

     Output the computed results. Selections can be made to print out increamental quantities or total quantities. Several output manners can be selected. Nodes where displacements are required to be output as well as Gaussian points where stresses need to be observed can be selected.

- *preserve the final information for future operations*

  Information on the results will be written to a data base for later reference. This operation enables the future continuing operations which can be started and be based on current results. Post data processes, such as graphics can use the data base as well.

INITIALIZATION OPERATION

Initialize the stress, displacement and energy state for all nodes and Gaussian points. The initial state may have three kinds of conditions.

- Both stress and displacements are zero or the displacements at all nodes are zero but the stresses at all element Gaussian points are constant.

- Both displacement field and stress field are non-zero and inherited from a previous computation data base. Some information besides displacements and stresses, e.g. $\mathcal{F}$, ϵ_{vi} at all integration points, will restored. This renders the different loading path requirement realized for certain problems. It also allows a big long-run job to be broken into several consecutive executives, making it easy for one to check the intermediate results and decide whether to continue or abandon the job that has been run.

- The displacements are zero at all element nodes while the stresses are inherited from the previous run. This is designed to meet some special requirements, for instance, if the analysis needs to start from a static earth pressure state where soil weight causes a non-zero stress distribution but the settlement of soil is assumed ceased.

ITERATE OPERATIONS

- CALL STIFFP($\cdots$)

  Evaluate the element stiffness matrices. Two key sub-operations are performed.

  - CALL CNSTTV($\cdots$)

    Evaluate the constitutive matrix $[\mathbf{D}_T]$. This is a module designed for free modification to accomodate new stress-strain constitutive models. Details about the input and output variables are listed in section 4.3.

– *Form the element matrices by calling the element libraries.*

- CALL ASSEMB(···)

  Assemble elemental striffnesses to the global stiffness matrix/matrices.

- CALL BOUND(···)

  Emplace the displacement boundary conditions.

- CALL EQSOLV(···)

  Solve simultaneous equations. Three choices can be made either automatically or manually. They are the Cholisky LDL^T method, the Gauss lower half matrix elemination method and the Gauss-Seidle iteration method. The second solution method will be applied automatically if criterion (2.21) is violated in the computation.

- CALL ESTRSS(···)

  Evaluate incremental effective stresses at all elemenet Gaussian integration points.

- CALL CONVER(···)

  Check if the convergence criterion for iteration is met. If not the iteration will continue. Otherwise the iteration operation will be jumped out.

4.2. LOAD COMBINATION AND INCREMENTS

Generally for a non-linear problem, choosing a suitably small increment of load vector $\mathbf{f}$ will guarantee convergence, and reasonable results will be obtained. On the other hand, if the increments of the load are too small, a large number of increments is inevitable and the computing time may be very long to reach the actual full load. Many solid mechanics problems exhibit high non-linearity within a starting period right after the load is exerted on the material domain. The non-linearity attenuates with the increase

of the load amount when the computation runs towards convergency. Thus a variable incremental load is designed for the purpose of both precise and economic requirements. Denote the total load between 0 and $\mathbf{f}$ by $\mathbf{f_i}$. Then let

$$\mathbf{f}_i = \mathbf{f} \sum \lambda_i \tag{4.1}$$

with

$$\lambda_i = \begin{cases} \lambda_1 & \text{for} \quad \lambda_i \leq \lambda_I \\ \lambda_2 & \text{for} \quad \lambda_I < \lambda_i \leq \lambda_{II} \\ \lambda_3 & \text{for} \quad \lambda_i \geq \lambda_{II} \end{cases} \tag{4.2}$$

where λ_I and λ_{II} are two preset bench marks to control the load increasing amount.

To make the load input flexible, a total load is assumed to be composed of two elementary load cases denoted by $\mathbf{f}_1$ and $\mathbf{f}_2$, respectively. Thus the total load vector can be expressed as

$$\mathbf{f} = \xi_1 \mathbf{f}_1 + \xi_2 \mathbf{f}_2 \tag{4.3}$$

where ξ_1 and ξ_2 are the parameters to combining the load cases.

4.3. CONSTITUTIVE MODULE

Subroutine CNSTTV evaluates the matrix $\mathbf{D}_T$ according to the constitutive formulae coded. It can be modified at the user's discretion by combining the data input through a variable port which provides the following data.

INPUT:

- ENRG0(KGAUS), The engergy $\mathcal{F}_0$ at Gaussian point KGAUS, at begining of current load/time increment.
- ENRG1(KGAUS), The engergy $\mathcal{F}_1$ at Gaussian point KGAUS, of current iteration within current load/time increment.

- EPSV(KGAUS), The volume strain at Gaussian point KGAUS, of current iteration.

- STRS1(4,KGAUS), The stresses $\{\sigma_x,\ \sigma_y,\ \tau_{xy},\ \sigma_z\}$ for plane problems or $\{\sigma_r,\ \sigma_y,\ \tau_{ry}, \sigma_\theta\}$ for axisymmetric problems at Gaussian point KGAUS, of current iteration.

- PROPS(LPROP,14), The material properties defined by 14 parameters for the material group LPROP.

OUTPUT:

- ALFA, A parameter used for certain circumstance. This is not a necessary parameter for all. It serves as a spare output information.

- DMATX(4,4), The stiffness matrix $\mathbf{D}_T$ to be designed.

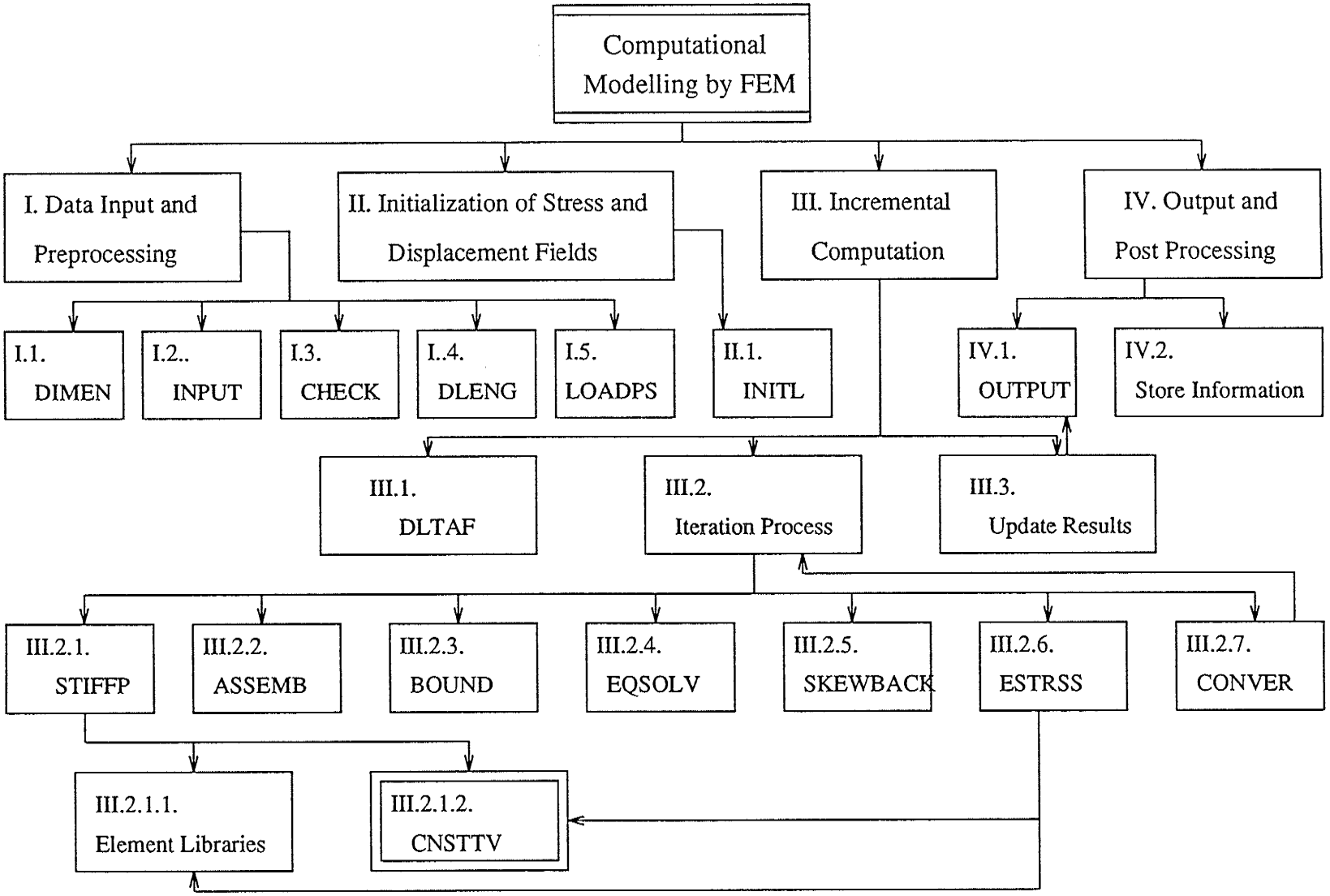


Fig. 4.1. Complete design hierarchy

CHAPTER 5

VERIFICATION OF THE FEM CODE

Subroutines of the standard finite element procedures, that is, the creation of element stiffnesses, assembly of stiffnesses, the solution process etc., have been tested and verified for a number of problems with analytical or theoretical results. In this chapter, attention will only be focused on the examples related to the KGJ model using a series of analytical and experimental solutions that are currently available. Comparison will also be made with predictions from the Modified Cam Clay Model performed by Saadat (1989). The stress system imposed by laboratory triaxial shear specimens and their finite element simulation, shown in Fig. 5.1a and b respectively, will be used in the following examples in this chapter. Programs and input data for both analytical and FEM solutions are attached in Appendices A to C for researchers who wish to check the results or to faciliate their further modelling work.

5.1 EXPERIMENTAL, ANALYTICAL AND FEM RESULTS

EXAMPLE 1. Sand - Bentonite Buffer

First, consider a drained triaxial test with constant p'. The stress system on the triaxial shear specimen is drawn in Fig. 5.1a. Both experimentally measured and analytical results from Yin et al (1989 and 1990) are available to check the FEM prediction. The initial stress and strain conditions are:

$$(\epsilon_s)_i = 0, \quad (\epsilon_v)_i = 0, \quad q_i = 0, \quad p'_i = constant$$

and the loading path requires $dp' = 0$. Note that the subscript i indicates the initial value. For the buffer soil with dry density of 1.5 Mg/m^3 or 1.66 MG/m^3, the three moduli are

for first-time loading:

$$K_f = V_\lambda p', \quad 3G_f = 3DJ_f^2/(J_f^2 + 3DK_f)$$

$$J_f = 13.7cp'\{\exp[V_\lambda \cdot d \cdot (\epsilon_v - \epsilon_{v_0})]/q\}^d \tag{5.1}$$

$$D = \exp[V_\lambda(\epsilon_v - \epsilon_{v_0})]\{1 - Fq\exp[\epsilon_{v_0} - \epsilon_v)]\}^2$$

for unloading/reloading:

$$K_e = 80p', \quad J_e = \infty, \quad G_e = 37.5\exp[V_\lambda(\epsilon_v - \epsilon_{v_0})] \tag{5.2}$$

If the current state were to be unloading/reloading (in which case the sand-bentonite buffer is isotropic, and dq is decoupled from $d\epsilon_v$), $d\epsilon_v = 0$, the only variation in strain energy is due to $qd\epsilon_s$ which is positive due to the shearing strain produced in the same direction as shearing stress. This would mean $\mathcal{F} > \mathcal{F}^*$, which contradicts the assumption of state of unloading/reloading. Thus the current state has to be first-loading and the group of moduli (5.1) must be used.

It is seen in eq (5.1), that all moduli are functions of q and the volumetric strain ϵ_v which on integration can be found as

$$\epsilon_v = \frac{\lambda}{V_i} \ln(p') + \epsilon_{v_0} + \frac{\lambda}{V_i d} \ln \left[1 + \frac{d}{(1+d)c} \left(\frac{q}{p'} \right)^{1+d} \right] \tag{5.3}$$

On combination of eqs (2.1), (5.1) and (5.3), the shear strain can be obtained as a function of shear stress alone:

$$\epsilon_s = \int \frac{1}{3G_f(\epsilon_v(q), q)} dq \tag{5.4}$$

The numerical value of ϵ_s can either be computed by the Runge-Kutta approach for ordinary differential equations or by any other numerical integration scheme. The predictions of the relationships of ϵ_s versus q/p'_i and ϵ_v versus q/p'_i by eqs (5.3) and (5.4) are shown in solid curves through Fig.5.2a, b and Fig.5.3.

With an analytical solution available, the finite element code can then be easily verified. In the finite element computation, four 8-node axisymmetric elements are used to simulate the domain of a laboratory triaxial shearing specimen, see Fig. 5.1b. A quarter of the vertical cross section is discretized because of symmetry of the specimen. The axial pressure σ'_y or σ'_1 and the cell pressure σ'_r or σ'_3 are forced to change in such a way that the mean effective stress is kept constant at $p' = p'_i$. Thus

$$\sigma'_y = 3p' - 2\sigma'_r \tag{5.5}$$

Incremental load must be added so that

$$\Delta\sigma'_y = -2\Delta\sigma'_r \tag{5.6}$$

and the condition of no-tension in soil media (using "compressive stress positive" sign convention) requires

$$(\sigma'_y)_i + \Delta\sigma'_y > 0 \quad \text{and} \quad (\sigma'_r)_i + \Delta\sigma'_r > 0 \tag{5.7}$$

If positive signs are used to define tension, the above condition should be reversed.

A uniformly distributed isotropic initial stress field is set up from input values of $(\sigma_r)_i = (\sigma_y)_i = (\sigma_\theta)_i = p'_i = constant$, which gives $q_i = 0$. Subsequently, the axial loads are increased in accordance with eqs (5.6) and (5.7) from zero until divergent shear strain appears and the specimen fails. Figures 5.2 through 5.4 compare the measured, analytical and the FEM solutions for $\epsilon_s - q/p'_i$, $q/p'_i - \epsilon_v$ and $\epsilon_v - \epsilon_s$ relationships, respectively, where round dots represent the experimentally measured values and dash-triangle lines indicate the FEM results. It is seen that the FEM and the analytical solutions are so close that the dash-triangle lines overlap the solid ones. A series of computations was performed with different values of consolidation pressures. The predicted normalized curves of ϵ_s against q/p'_i are unique provided other initial conditions remain unchanged. The experimental data in these figures had not been used earlier in calibrating the K, G, J functions.

The extreme (failure) state is attained when the shear strain tends to infinity. Fig. 5.3 shows a sudden turn up of the FEM curve when the curve reaches the point where $q/p'_i = 0.49$ and the volume strain $\epsilon_v = 0.003$ for the 1.5 Mg/m^3 buffer soil. The divergency is caused by the zero value of the D_f modulus, referring to eq (2.12), which generates singularity in the fundamental model eq (2.1). The curve of solution (5.3) meets the curve $G = 0$ at $q/p'_i = 0.495$, $\epsilon_v = 0.0031$, as shown in Fig. 5.3. This singularity makes the determinant of the stiffness matrix in eq (2.23) equal to zero, thus leading to divergence in the computation. Physically the numerical divergency means that the soil reaches its critical state as is clearly seen in Fig. 5.2a where the shear strain tends to infinity as q/p'_i moves near its limiting value of 0.5.

Fig. 5.4 illustrates the relationship between ϵ_v and ϵ_s for buffer soils with both dry densities that have been tested. It is seen that at the begining of the shear test, the volume strain increases at almost the same rate as the shear strain does. Later, however,

the rate of shear strain increases faster. The shear strain tends to infinity as the volume strain moves close to 0.003 for 1.5 Mg/m^3 buffer and 0.009 for 1.66 Mg/m^3 buffer. This corresponds to the critical stress state as mentioned in previous paragraph.

One more observation is that the 3-modulus hypoelasticity model could not describe the situation when either p_i' is very small or $q \equiv 0$. Because if $q = 0$, from eqs (2.25) and (2.28) the calculation of parameters A, B and C gives a null stiffness matrix. As it is, the original constitutive relation (2.1) becomes singular when J (which is the function of q) tends to zero. This problem arose at the onset of the computation and was solved by forcing the state at the first loop of iteration of the first increment to the unload-reload case.

The FORTRAN program for predicting the analytical results and the input data for the FEM solution are provided in Appendice A and B.

EXAMPLE 2. Wuhan sand.

The test on Wuhan sand (Yin and Yuan, 1985) that will be used for this comparison was performed at constant mean effective pressure. The model itself was calibrated from drained compression tests with $\sigma_3' = const.$, in contrast with the undrained tests used previously. The expressions of the K, G, J moduli as well as the values of parameters were given ealier in section 2.1. The measured and analytical results for q vs ϵ_s and ϵ_s vs ϵ_v are seen in the references by Yin (1990), and Yin, Saadat and Graham (1988); and are reproduced here in Figures 5.5 and 5.6. The initial conditions used in the computation are

$$p_i' = p_{cons}', \quad (\epsilon_v)_i = p_{cons}'/(R + Sp_{cons}') \tag{5.8}$$

and the testing stress path requires $dp' = 0$, $d\sigma_3' = -dq/3$. Figures 5.5a,b and 5.6 illustrate the comparisons of the experimentally measured, analytical and FEM results

to q vs ϵ_s and ϵ_v vs ϵ_s for the cases of $p'_{cons} = 0.2$ MPa and $p'_{cons} = 0.4$MPa, respectively. The parameters appearing in eqs (2.15) to (2.18) are rewritten in MegaPascals below.

$$R = 6.867 \ \text{(MPa)}, \quad T = 0.0042276 \ (\text{MPa})^{1-m}$$

$$U_0 = 2.483694 \ (\text{MPa})^{m-1}, \ U_{ult} = 3.548134 \ (\text{MPa})^{m-1} \tag{5.9}$$

$$V = 2.81838 \ (\text{MPa})^{m-1}, \ W = 0.281838 \ (\text{MPa})^{m-1}$$

Once again it is seen that the FEM solution matches the analytical solution so well that the dash lines coincide with the solid lines. The program for the analytical solution, and input data for both methods, are given in Appendix A and B. Excellent agreement between the results predicted by the FE code and those obtained by analytical means confirm the validity of the formulations and algorithm used in the code.

5.2 EXPERIMENTAL, CAM CLAY MODEL AND KGJ MODEL RESULTS

EXAMPLE 3. Buffer specimen T915, undrained.

Consider a strain-controlled, undrained shear test, specimen no. T915 from the test series performed by Saadat (1988) on sand bentonite "buffer". At a consolidation pressure p'_{cons}=3 MPa, the specimen was consolidated in 5 increments of cell pressure from $p'_{cons}/3$ to p'_{cons} with load increment ratio 1.32 and then sheared under undrained consitions (CIU). The confining pressure remained $\sigma_3 = constant$ during shearing. At failure the stresses were measured as

$$q_f = 1.24\,\text{MPa} \ \ p'_f = 2.28\,\text{MPa} \tag{5.10}$$

We will here model only the undrained shear part of the test.

The initial stress and strain conditions (i.e., at the end of consolidation) are

$$(\sigma'_1)_i = (\sigma'_3)_i = p'_i = 3 \ \text{MPa}, \ \ q_i = 0$$

$$(\epsilon_v)_i = 0.087, \quad (\epsilon_s)_i = 0$$

The test was conducted on a buffer specimen with initial dry density 1.5 Mg/m^3. By use of the formulas presented in section 2.1, the parameter ϵ_{v_0} is calculated to be 0.0068. All other parameters are the same as given in Table 2.3.

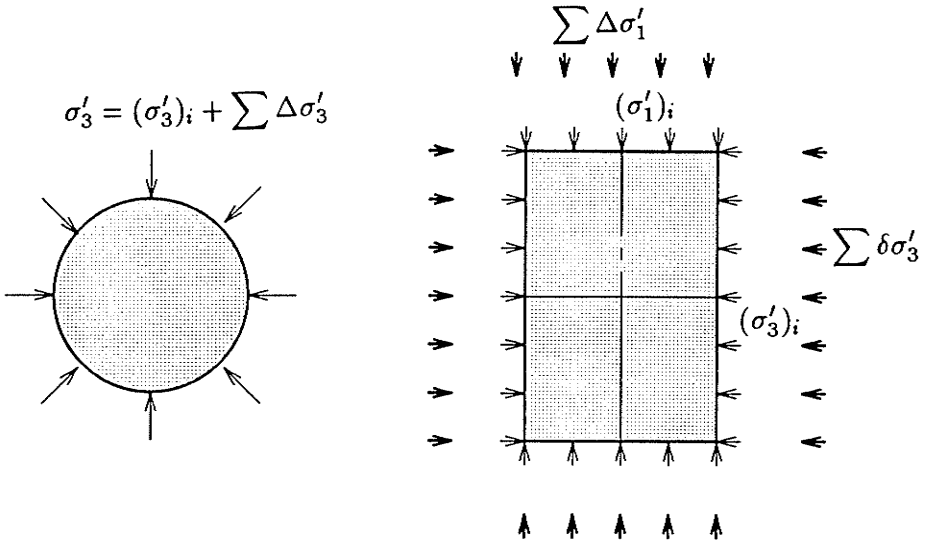
A total stress loading condition is applied. The axial pressure is applied incrementally from 0 until $\sum \Delta\sigma_1$ reaches 1.24 MPa, while $\Delta\sigma_3$ is kept zero throughout. No volume strain change is obtained, in agreement with the undrained tests which are performed at constant volume. Porewater pressure increases generated by the shearing process produce decreases in the effective mean stress. Figure 5.7a shows the computed curve of p' vs q from the end of consolidation untill critical state failure takes place. The measured results at both ends are shown as square dots. The small deviation of the results at failure is because the KGJ parameters selected for the computation were calibrated from the average of a group of tests that did not include T915. As is shown in Figure 5.7b, the finite element result of the normalized q/p_i' vs p'/p_i' relationship falls right in the middle of the calibrating data enclave, and coincides exactly with the curve predicted by the KGJ analytical formula (Yin et al, 1990).

Figures 5.8a and 5.8b compare the measured response of the buffer with values predicted by (a) the Modified Cam Clay model implemented in a finite element program known as CRISP (Britto and Gunn 1987), and (b) by the hypoelastic KGJ finite element modeling from this thesis. The predictions made by both models are similar. It seems that the $q/p_i' - \epsilon_1$ relation predicted by the KGJ model is closer to the experimental results than that predicted by the Modified Cam Clay model. The KGJ q/p' vs ϵ_1 curve passess through the scattered measured dots; and the KGJ q/p_i' vs ϵ_1 prediction matchs the measured data well before failure occurs. This also indicates that the KGJ model can represent the compressive behavior of the buffer in undrained shear reasonably well. However, neither the KGJ nor the Modified Cam Clay model can predict the softening

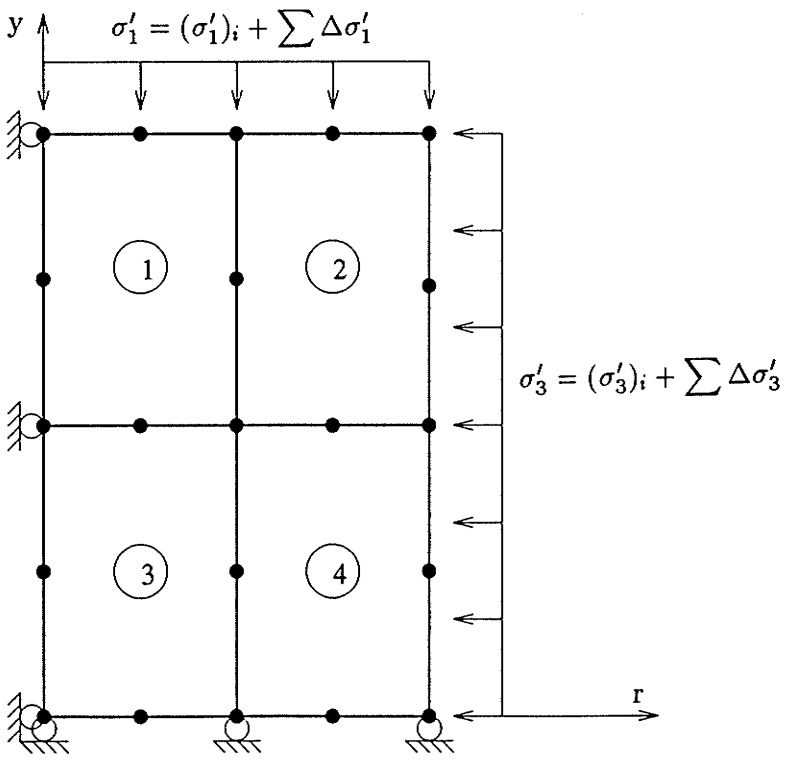
behavior which usually follows shear failure in buffer material.

EXAMPLE 4. Sand-bentonite buffer specimens T910, T918 and T931, drained constant-p' tests.

For the drained condition, the comparison is made for constant-p' test with the specimens numbered T910, T918 and T931. The specimens were consolidated isotropically to 3 MPa pressure and then sheared under drained conditions until the axial deformations accelerated towards failure, while the effective stress p' was held constant. Failure in a specimen is defined as the maximum value obtained by the deviatoric stress. The tests were performed by adding deviator stresses in increments and holding them constant for periods up to 5 days before the next increment was added. Saadat (1989) shows q/q_{max} vs ϵ_s results for strains during 1 day, 2 days, 3 days and 5 days loading. Initial conditions of $p'_i = p'_i, \quad q_i = 0$ have been used in the numerical modelling work. Other parameters used in KGJ model are taken from Table 2.3 for buffer specimens with $r_d =$ 1.5 Mg/m^3. The stress path is determined by eqs (5.5) through (5.7). Fig. 5.9 shows the relationship between normalized shear stress q/q_{max} and shear strain ϵ_s predicted by the KGJ model compared to the 3-day experimental data. The numerically predicted curve represents an acceptable average of the measured curves, suggesting wide applicability of the KGJ model.

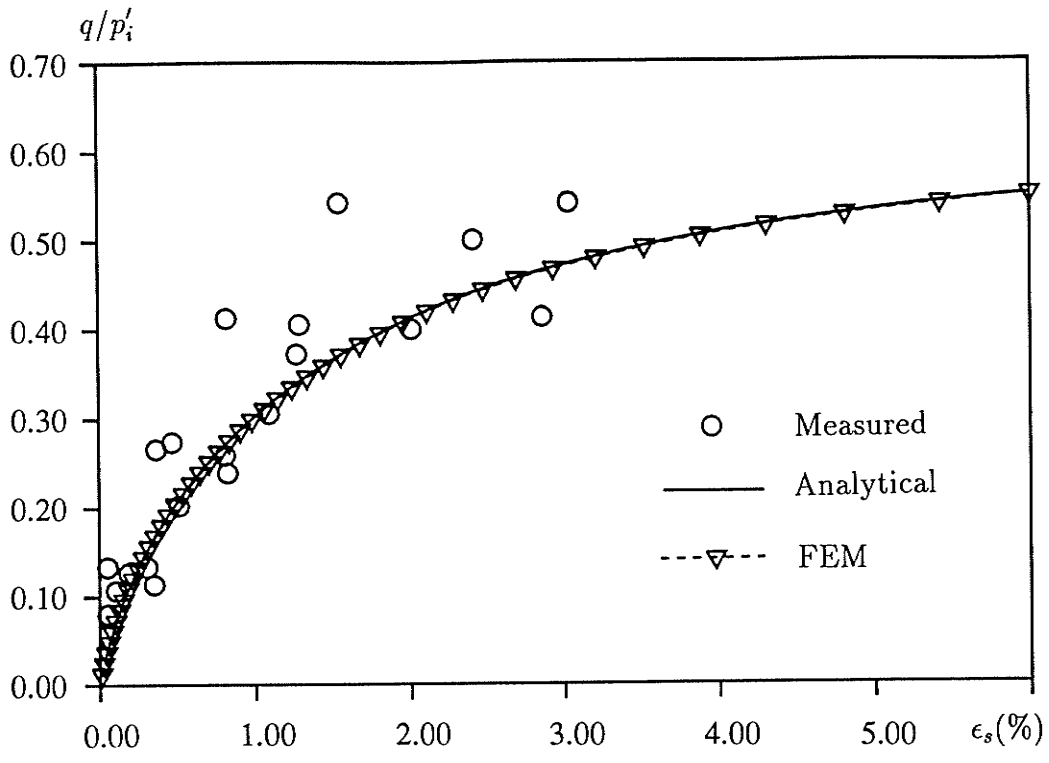


(a) Stress system on the triaxial shear core

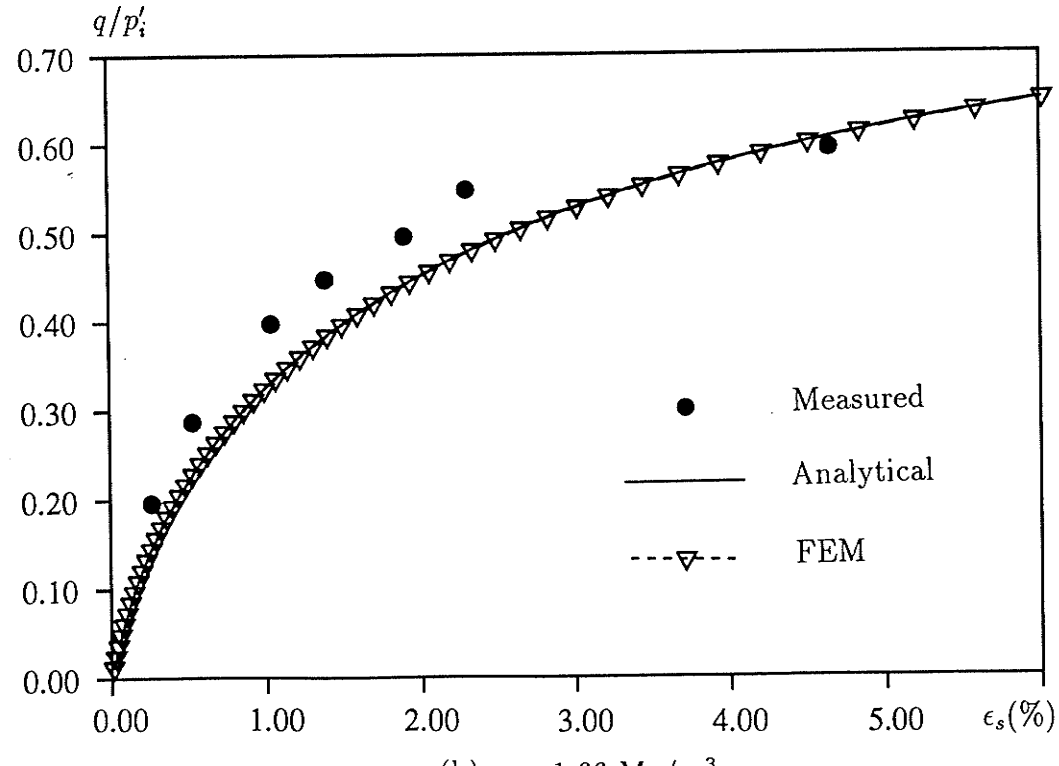


(b) Axisymmtric FE mesh to a quarter of the specimen

Fig. 5.1. Analysis of a laboratory triaxial specimen



(a) r_d =1.5 Mg/m^3



(b) r_d =1.66 Mg/m^3

Fig. 5.2. Comparisons of measured, analytical and FEM q/p_i' vs ϵ_s curves in drained constant-p' test

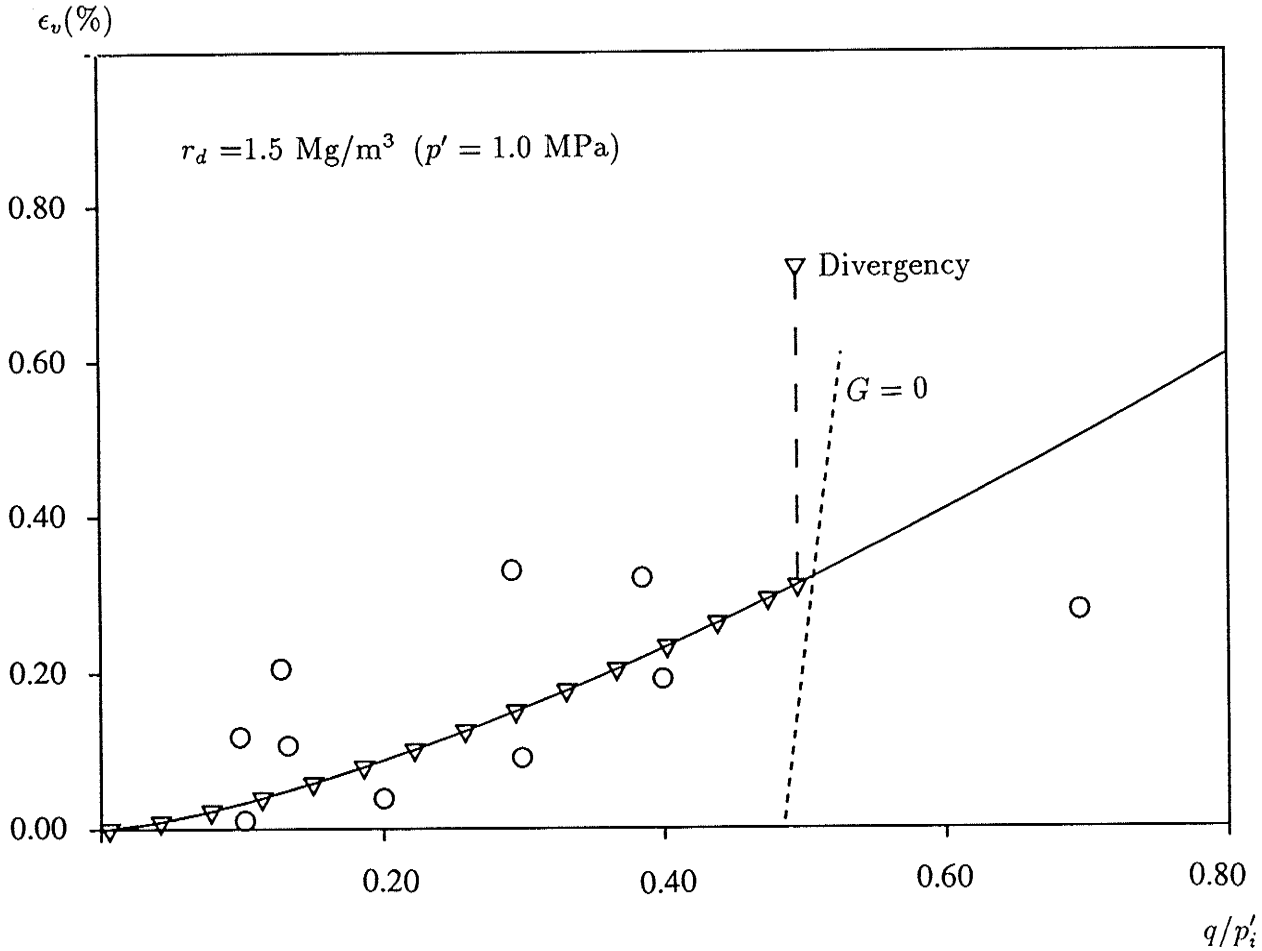


Fig. 5.3. Comparisons of measured, analytical and FEM ϵ_v vs q/p_i' curves in drained constant-p' test

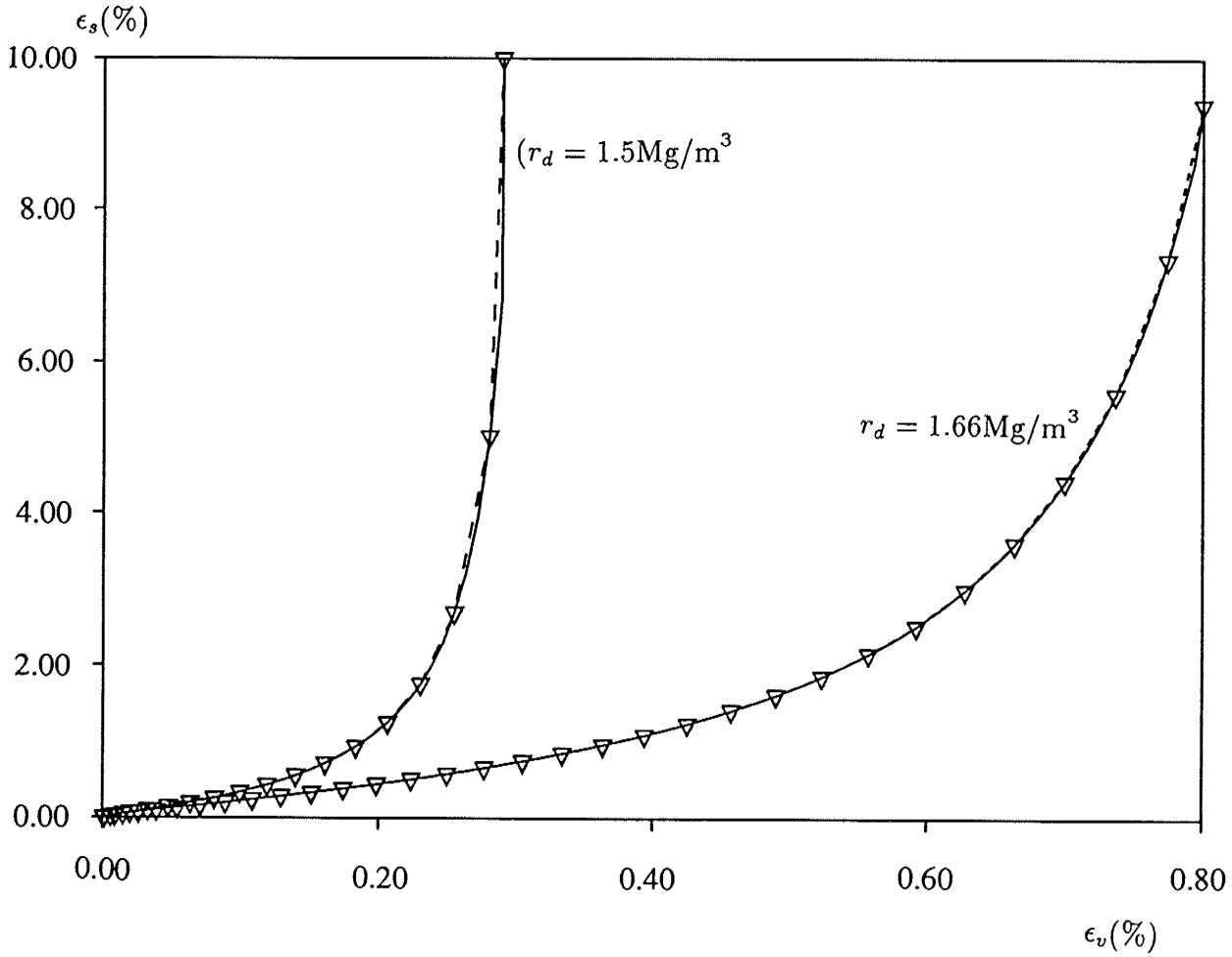
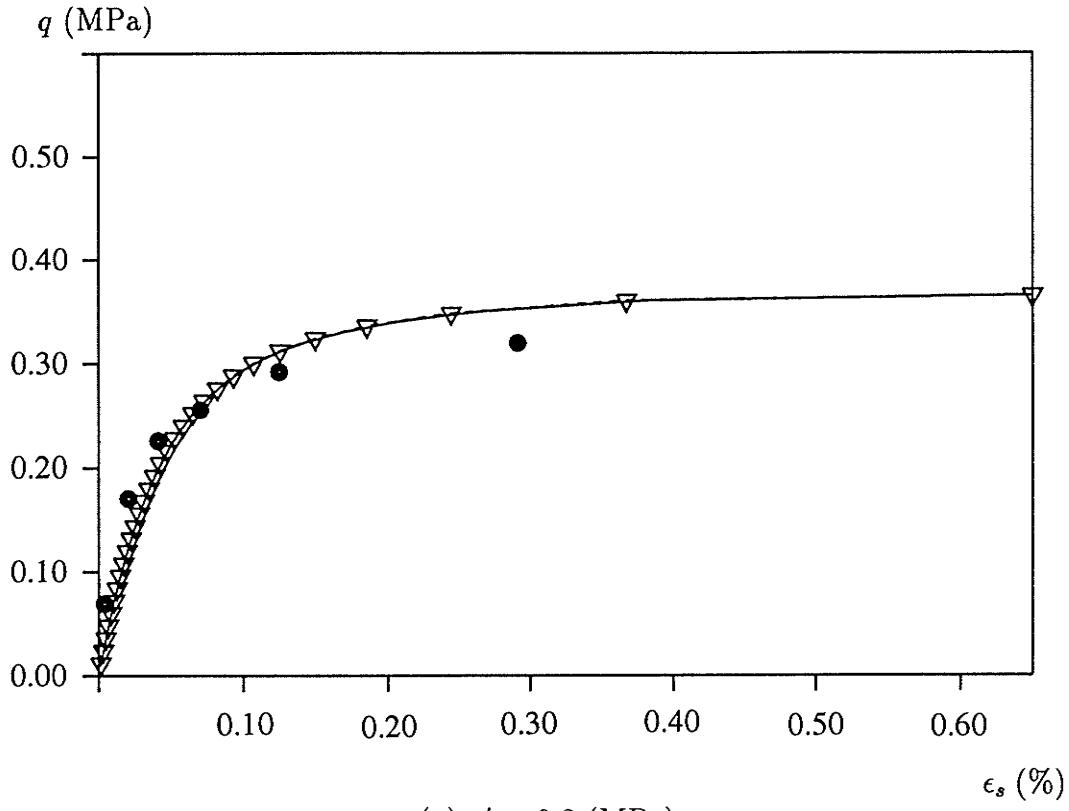
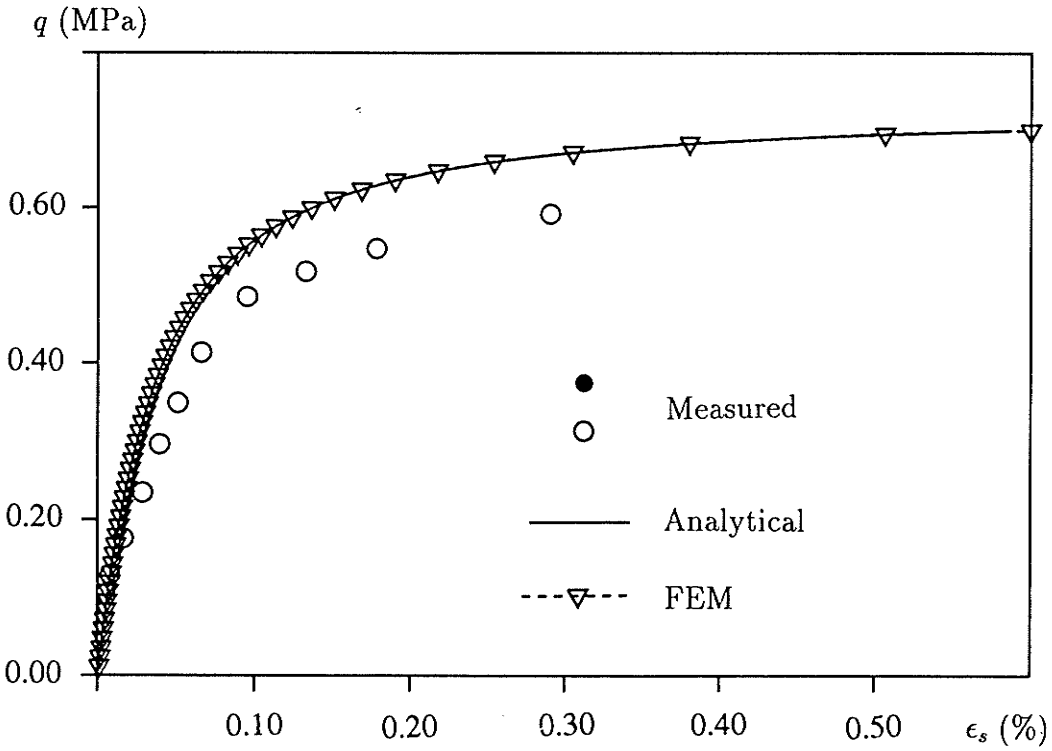


Fig. 5.4. Comparisons of analytical and FEM ϵ_v vs ϵ_s curves



(a) $p_i' = 0.2$ (MPa)



(b) $p_i' = 0.4$ (MPa)

Fig. 5.5 Measured, analytical and FEM q vs ϵ_s results for Wuhan sand

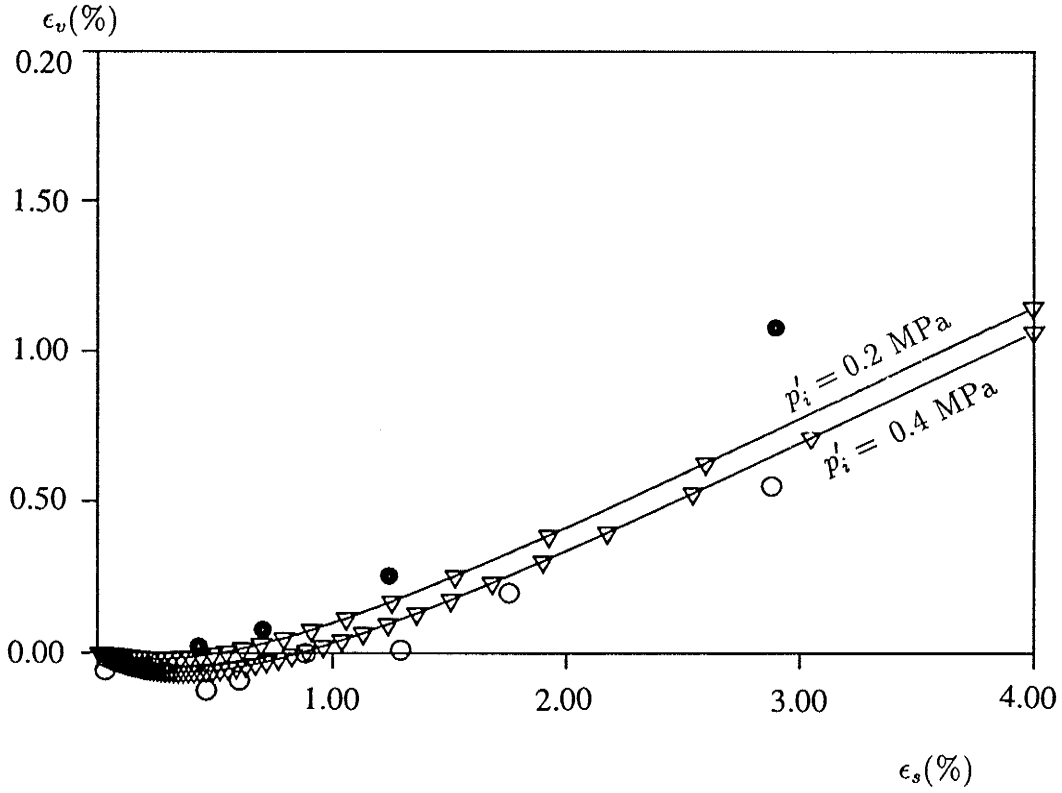
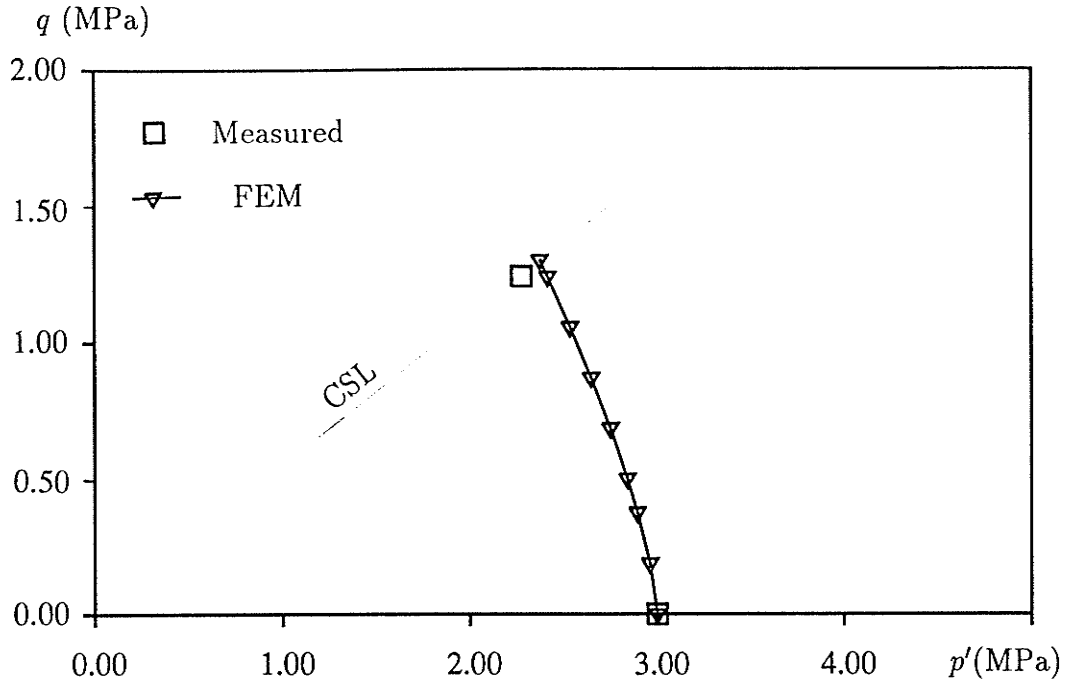
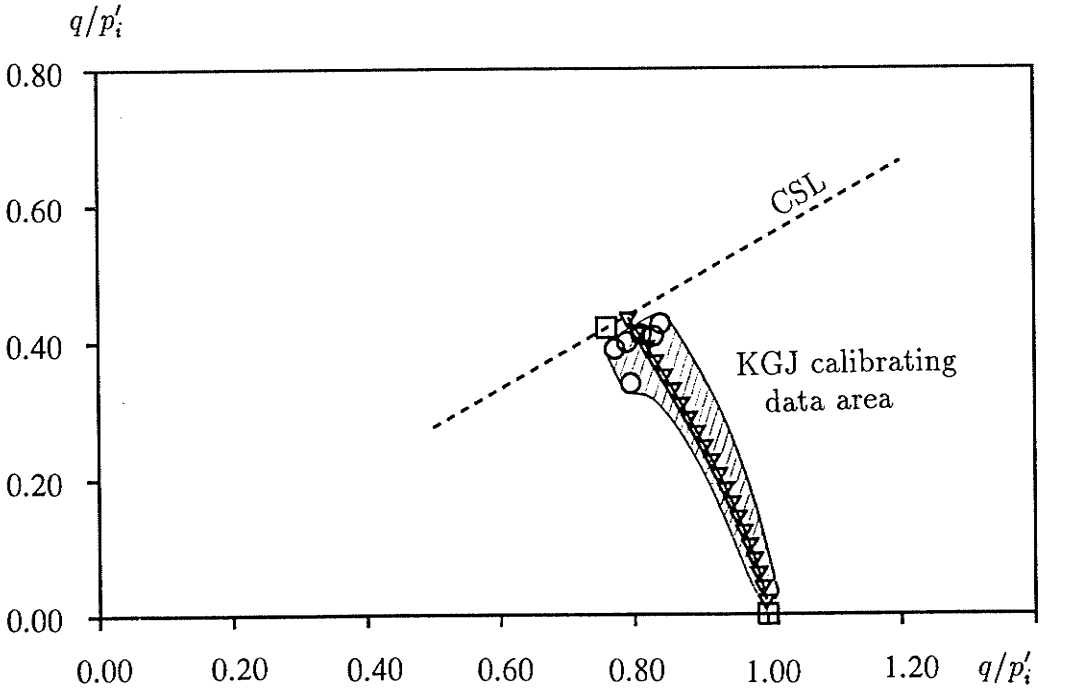


Fig. 5.6. Measured, analytical and FEM ϵ_v vs ϵ_s results for Wuhan sand



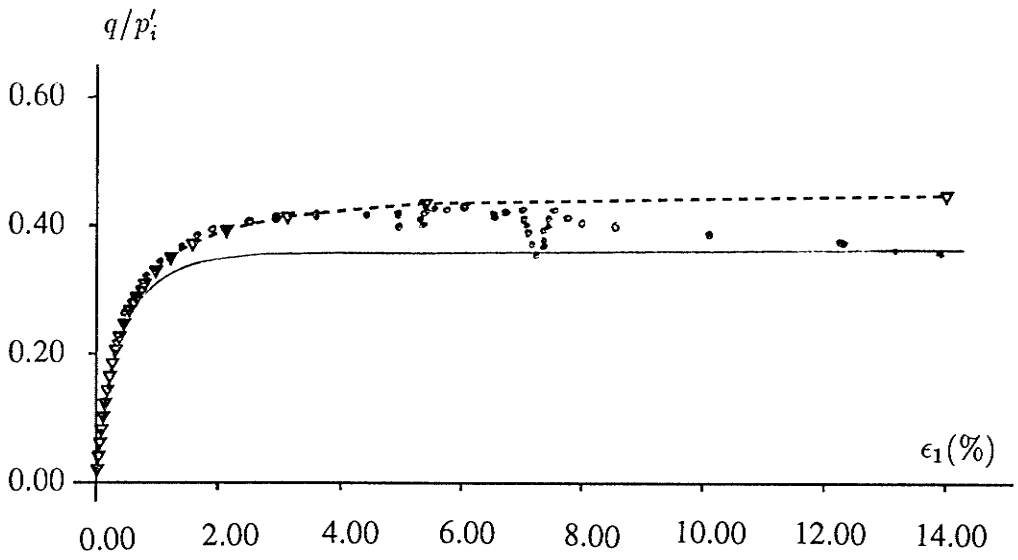
(a) p' vs q results



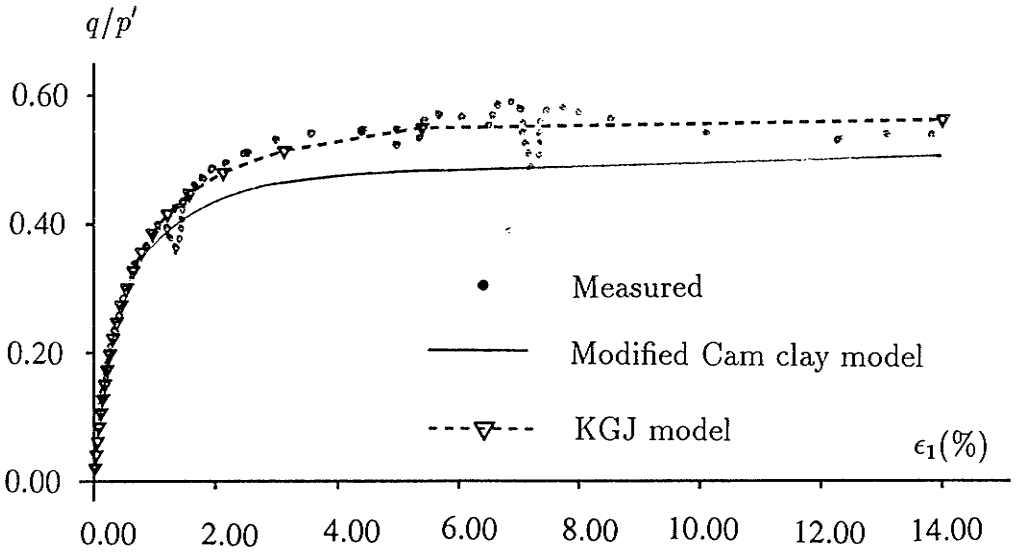
(b) Normalized p'/p'_i vs q/p'_i results.

Shaded area represents the KGJ calibrating data.

Fig. 5.7. Comparison of finite element KGJ and measured results for sepcimen T915



(a)



(b)

Fig. 5.8. Comparison of the Modified Cam clay model and the KGJ model to the test results for specimen T915

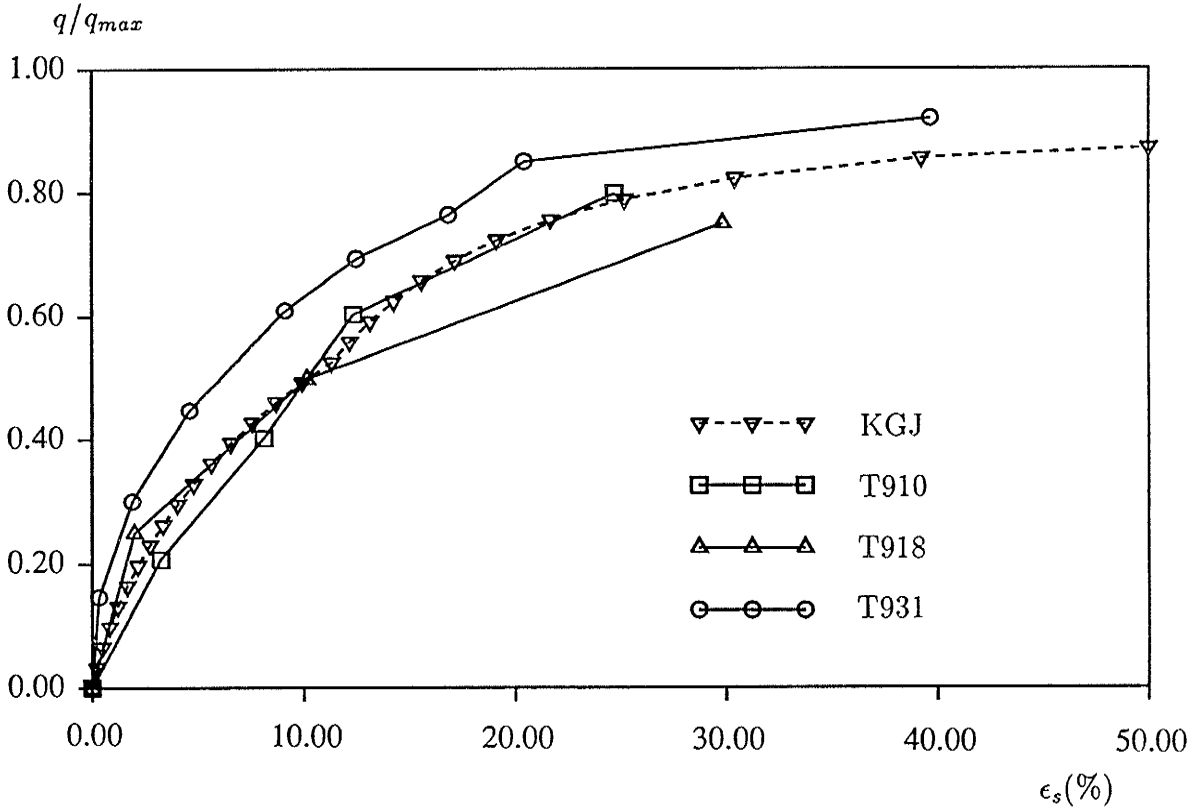


Fig. 5.9. Comparison of the KGJ finite element result to the test results for specimens T910, T931 and T918 (3 days)

CHAPTER 6

MODELLING OF ROCK-BUFFER-CONTAINER PROBLEM

The sand-bentonite buffer modelled in Chapter 5 has been proposed for use in the Canadian nuclear fuel waste management program (Rummery and Rossinger 1983, Figure 6.1). The conditions under which it must function may vary significantly with time. The mechanical behavior of the buffer under various ambient conditions is one of the fundamental concerns of the concept. This chapter presents a preliminary examination of interactions between buffer, rock, container and backfill in the vault environment, and any resulting localized critical states that are likely to occur in the buffer. The KGJ model and the numerical procedures developed in the previous chapters will be applied here to conduct finite element modelling of the vault behavior under several possible loading scenarios. The analyses include (1) conditions where the presence of fissures in the rock mass subject the buffer-container system to sudden high porewater pressures; (2) swelling pressure develops in buffer due to access to groundwater; and (3) the container weight and backfill loading.

6.1. FLUID-INDUCED ROCK-FILL SEPARATION

The disposal vault proposed in the Canadian Nuclear Fuel Waste Management Program consists of a number of mined chambers 500 to 1000 metres below the surface. Recent design concepts for the waste repository regard the eventual flooding of the disposal vault as inevitable. Experimental evidence suggests that the buffer will be fully saturated during a large portion of the design lifetime of the waste containment system (Pusch et al, 1985). Therefore, this study focuses on interactions between saturated buffer, container, rock and backfill caused by the influx of water into the waste disposal vault, and by the development of the swelling pressure in the buffer. The influence of two possible bounding conditions between the buffer and the rock mass will be studied for problems of possible extension of a buffer-rock separation zone, and stress redistribution in the buffer.

The buffer-container system may be subjected to high porewater pressures via cracks or fissures developed in the rock mass. The fluid pressures can act at arbitrary locations on the buffer-rock interface, and result in separation between the relatively impermeable buffer and the rock mass (Figure 6.2). The state of deformation in the system is 3D in general. However, it is here assumed for simplicity that the extent and location of the separation zone is such that the deformation is approximately plane strain in a transverse section of the assembly. Plane strain analysis essentially yields a possible upper bound loading configuration for estimating displacements of the container and the pressures that are generated at the buffer container interface. It is therefore conservative from the design point of view. Additional simplifying assumptions are made below, relating to the mechanical characteristics of the individual components of the disposal vault: (1). The rock mass is treated essentially as an intact rigid medium; (2) The interface between non-separated zone of the buffer and rock mass is assumed to be either completely bounded

or frictionlessly bounded. These model the upper and lower bound conditions. The real problem will have intermediate fixing. (3) The deformation in the waste container is so small compared to that of the buffer soil that it can be modelled as an elastic solid. (4) The boundary between the container and the buffer is assumed to be completely bounded. It is also assumed that the buffer-container geometry and loading is symmetric. With these assumption the finite mesh together with the boundary support conditions is drawn in Fig. 6.3, where only one-half of the cross section needs to be discretized. The elastic modulus and the Poisson's ratio of steel, 2.0×10^5 MPa and 0.25 respectively, are selected to simulate the property of the waste container in this preliminary analysis.

According to the Canadian program (CNFWMP), after vault closure, water pressures in the rock and the pores of the buffer will approach 10 MPa, i.e. 1000m of water head, a value that is high in normal geotechnical engineering terms. Possible hydraulic fracturing of the buffer-rock interface induced by rising groundwater pressures in the rock must be examined. In this computation it is assumed that the 10 MPa uniform fluid pressure could be suddenly applied on the buffer as an external loading in the early stages of borehole response. The extent of initial fluid-induced separation at the buffer-rock interface depends on the hydrogeological conditions of the surrounding rock. In a complete analysis of the problem an iterative technique should be used to determine the extent of the separation zone (Selvadurai et al, 1985). However in the study here, a 120^0 initial angle of separation zone is assumed to be prescribed. Under such a high external pressure under undrained conditions the water phase in the buffer can no longer be assumed incompressible. A standard bulk modulus of water, $\beta = 32 \times 10^4$ lbf/in^2 or 2206.3 MPa, is used (White, 1986). The porosity of buffer is calculated from the specific volume of buffer presented by Saadat (1989). The specific volume of buffer before consolidation falls in the range of 1.51 – 1.80. Most of the specimens with dry density 1.5 Mg/m^3 have a value of approximately 1.75 which corresponds to a porosity of

$n = 43.0\%$. An initial magnitude of effective stresses is assumed as 2.1 MPa to simulate the stress condition of the compacted buffer soil after vault closure.

Figures 6.4a and b show displacement patterns from the compression of the interstitial water for the completely bounded case and the frictionlessly bounded case, respectively. The fluid pressure results in a maximum displacement of the buffer along the horizontal symmetric axis in the order of 1.4 mm and 1.5 mm, respectively, in the two cases. The deformation of the buffer leads to lateral translation of the waste container. In the completely bounded case, translation of the container is 0.9 mm, 0.2 mm greater than the value predicted for the frictionlessly bounded case. Under fully bounded conditions, the soil particles along the non-separated buffer-rock intersection almost stand still. Large difference of the deformation between the particles around the end of the separation zone results in locally high strain which in turn produces high stress concentration there, as is seen in the stress ratio trajectories shown in Figure 6.5a. This strongly suggests a tendency for the separation zone to extend tangentially, because $q/p' = 0.6$ is not possible in buffer. It is expected that full separation will take place between the buffer and rock shortly after the high water pressure is applied through the prescribed crack, since the actions of applying the pressure and separation proceed simultanously.

By contrast, the frictionlessly bounded condition on the buffer-rock mass interface makes the buffer deformation evenly distributed within the buffer soil. Only relatively low mobilized shear resistance is observed in the stress ratio distribution configuration shown in Figure 6.5b. Thus if the bore hole surface is completely smooth, the buffer will be squeezed by the high fluid pressure to the other side of the hole (Fig. 6.4b). Further separation will not take place.

Since the real condition between the buffer and the rock mass is neither fully bounded nor completely frictionless, the real stress state the buffer should be somewhere between the two extreme circumstances predicted above. Nevertheless, to prevent the buffer from

possible shearing stress mobilization, it is suggested that the surface of the borehole in the disposal vault be built as smooth as possible.

Compared to results obtained from the Modified Cam Clay model (Saadat, 1989) for the completely bounded case, the KGJ model gives larger deformations and higher concentrated shear stresses near the end of the separation zone. The difference of the maximum displacements is about 61% larger and the stress ratio is 4 times as high near the end of the separation zone. In fact no stress concentration was reported by Saadat in this region. This may suggest that the KGJ model is either safer or more conservative.

The problem shown in Figures 6.3 to 6.5 came from original work by Saadat (1989). It has been solved here using the same number of elements and the same angle (60^0) for the initial half-opening in which the water pressure is applied. It was felt necessary to examine also some additional half-angles to obtain some sense of how separation might propagate between the buffer and the rock under high water pressure. The results are shown in Figures 6.6a, b, c for half-angles 30^0, 60^0 and 90^0 respectively.

With a small initial opening (30^0 in Fig. 6.6a) high shear strength mobilization occurs at the crack tip, and we could expect further separation. The over stressed region is smaller in Fig. 6.6b (half-angle 60^0), and is not present in Fig. 6.6c (half-angle 90^0). This suggests that high water pressures fed in at one location on the buffer perimeter will not spread indefinitely, and provide full hydraulic connectivity.

6.2. BEHAVIOR UNDER CONTAINER WEIGHT AND BACKFILL LOADING

To simulate the emplacement of the buffer-container-backfill system, in the computation, the buffer is assumed to be in equilibrium under its own weight in a dry borehole and compacted to a certain dry density (swelling pressure) before the container weight and backfill loading are applied. The stiffness of the buffer is clearly much less than

both the container and the surrounding rock. The following simplifying assumptions can therefore be made. (1) The rock is treated as an intact rigid medium. (2) The interface between the buffer and rock mass is conservatively assumed to be completely smooth, i.e., with no frictional forces between them. (3) The waste container is modelled as an elastic solid. (4) The boundary between the container and buffer is assumed to be completely bounded, with full mobilization of frictional stresses. (5) The deformation is axi-symmetric with respect to the central axis of the container. To examine the influence of the compaction on the buffer behavior, two values of compacting conditions will be applied to the buffer soil, producing swelling pressures (initial effective stresses) of 100 kPa and 2.0 MPa, respectively.

The assumption of axi-symmetric conditions can greatly reduce the cost of computation. Fig. 6.7 shows the finite element discretization representing one-half of the vertical cross section of a borehole system. The material properties are for buffer compacted at 95% ASTM modified density, which gives a dry density of 1.67 Mg/m^3 and water content 22.5% (Saadat, 1989). The parameters used in the KGJ model are then selected from Table 2.3. The calculated density of the compacted buffer soil is 2.05 Mg/m^3. The program first subjects the buffer to the in-situ overburden stresses from overlying buffer soil, and then to compaction to produce effective swelling pressures of 0.1 MPa and 2.0 MPa. The stress state of the buffer is then used as the initial state for buffer subjected subsequently to the weight of the container and surcharge pressure from the vault backfill. The initial volume strain, ϵ_{v_i}, is calculated reciprocally. The weight of the container is applied as a uniformly distributed pressure of 277 kPa at the bottom of the container. The backfill surcharge pressure at the buffer surface is assumed to be uniformly distributed and equal to 120 kPa in magnitude. Only the drained condition is considered in the study, since the water compression caused by the relatively small container weight and backfill pressure is trivial under undrained conditions. The magnitude

of the volume change is in the order of 0.01 mm^3 if the water is considered compressible and vanishes if the water compressibility is neglected.

Figures 6.8a and 6.9a depict the resultant vertical displacement contours. The displacement patterns in the buffer are mainly influenced by the presence of the relatively rigid container. The magnitude of deformation is greatly influenced by the compaction and swelling pressure. Loose buffer, compacted to a swelling pressure of 0.1 MPa, produces a maximum vertical displacement as high as 66 mm at the buffer-backfill interface, while the 2.0 MPa swelling pressure compaction makes the buffer so stiff that the maximum vertical displacement is only 3.9 mm. The resultant stress q/p' trajectories within the buffer are shown in Figures 6.8b and 6.9b for the high compaction case and low compaction case respectively. The stress distributions are obviously influenced by the presence of the relatively rigid container, leading to non-uniform stress distribution, particularly around the corners of the container. Higher stress ratios happen right below the container, and stress concentrations are generated around the corners of the container (failure occurs at $q/p' = 0.5$). Again the level of compaction affects the properties of the buffer and influcences the magnitude of the stress ratio significantly. A critical stress state is mobilized around the bottom corner of the container for the loose compaction case (Fig. 6.9b), while the highest stress ratio is only 0.14 within the same location for the denser compaction case. The lowest stress ratios in the buffer always happen along the shaft of the container for both cases, due to the stiffer container and the bounding assumption between the buffer and the container.

The above observation suggests that an appropriate amount of compaction to the buffer before the emplacement of the container is crucial for the buffer-container-backfill system to function normally. Higher compaction, and hence higher swelling pressures in the buffer, will lead to smaller deformations and less tendency of high shearing strengh

mobilization . In contrast, lower compaction may result in potential degradation of the disposal system performance.

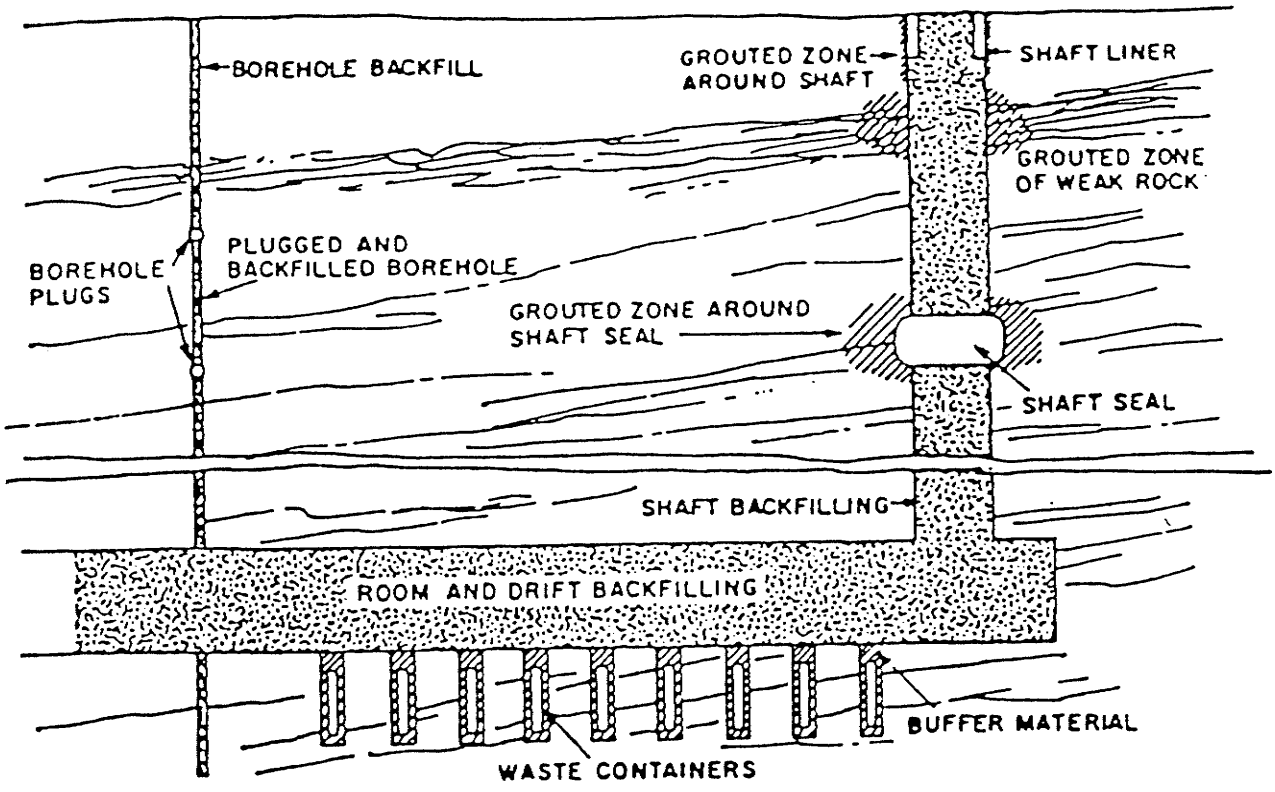
6.3. LONG TERM SWELLING OF THE BUFFER

The time-dependent volume change of the buffer is complex, involving changes in soil fabric structure and redistribution of water within the soil. In the disposal vault, the buffer starts to develop swelling pressure upon access to groundwater, due to the high suction capacity of the compacted bentonite clay in the mixture. The resulting pressure interacts with the surrounding environment. If swelling deformations can take place, the swelling pressure will be reduced. The deformation mainly depends on the density of the emplaced buffer, and the relative stiffnesses of the backfill, the container and the rock. To examine long-term implications of the development of swelling pressures in the buffer, an idealized drained case of swelling and pressure relief in the borehole is assumed here. Under a "no-volume-change" condition, the buffer could develop swelling pressures up to 2.1 MPa (Saadat, 1989). However, interaction between the swelling pressures in the buffer and the backfill leads to upward deformation of the backfill. Subsequent relief of the swelling pressures upon deformation of the backfill results in redistribution of stresses in the borehole. An analysis of the problem is performed under displacement controlled conditions, by allowing uniform swelling displacements from 10 mm up to 70 mm at the top of the borehole in small increments. The same conditions as those used in previous section are assumed for the finite element geometry, the KGJ material parameters, and the simplifying assumptions.

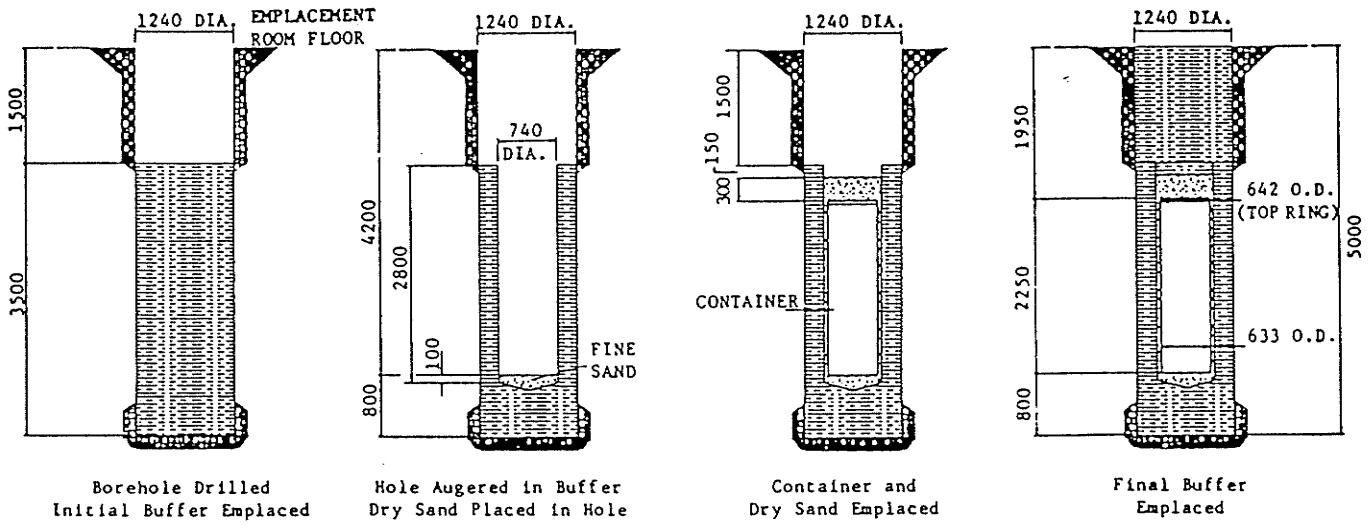
Contours of vertical displacement of the buffer are shown in Fig. 6.10a and the resultant stress ratio q/p' trajectories in the buffer mass are summarized in Fig. 6.10b, respectively, for the 50 mm release case. The results indicate that the waste container

will lift up approximately 15.6 mm if the backfill lifts upwards uniformly by 50 mm. The swelling of the buffer above the container will be essentially uniform in this case. High shear stresses focus around both ends of the container. The highest localized stress ratio is about 0.54 around the corners of the container in the buffer. The stress ratio q/p' reaches 0.5 in the areas right above and below the waste container. This means that the buffer would potentially approach the critical state conditions in continued regions of those areas. Lower stress ratios are seen in the middle of the container especially along the shaft due to the smaller deformations of the container. Figures 6.11a to c show that the plastic yield zone development in the buffer with the release of the swelling pressure by gradually lifting the backfill soil uniformly upwards. Here a plastic zone is defined in the buffer as an area where the stress ratio reaches greater or equal to 0.5. It is found that there is no plastic yield mobilization until the lifting displacement of the backfill comes to 47 mm. The plastic zones start from the top and bottom corners of the buffer and then spread toward the buffer centre with increasing uplift of backfill. After that the plastic zones develop axially upwards or downwards. They are generated simultaneously at both ends of the container and extend symmetrically to the horizontal axis about the container centre. A series of computation reveals that higher compacted buffer soil will require larger release of swelling pressure to start the plastic zone, while smaller uplifts may trigger plastic yield mobilization for loosely compacted buffer.

Figs 6.7 to 6.11 all come from computations using two quadratic elements in the radial direction. Some questions may arise whether this is sufficient to achieve good accuracy. For this reason, additional computations have been done using eight linear elements in the radial direction for the case shown in Fig. 6.10. The results are shown in Fig. 6.12. The stress trajectories and the plastic regions shown in Figs 6.11b and 6.12b are similar, suggesting that two quadratic elements provide acceptable accuracy in this axisymmetric problem.



(a) Schematic diagram of the underground vault for nuclear waste disposal



(b) Borehole emplacement sequence (all dimensions in mm)

Figure 6.1 The components of the Canadian concept for nuclear fuel waste management (after Rosinger and Dixon, 1982; Kjartanson and Gray 1987)

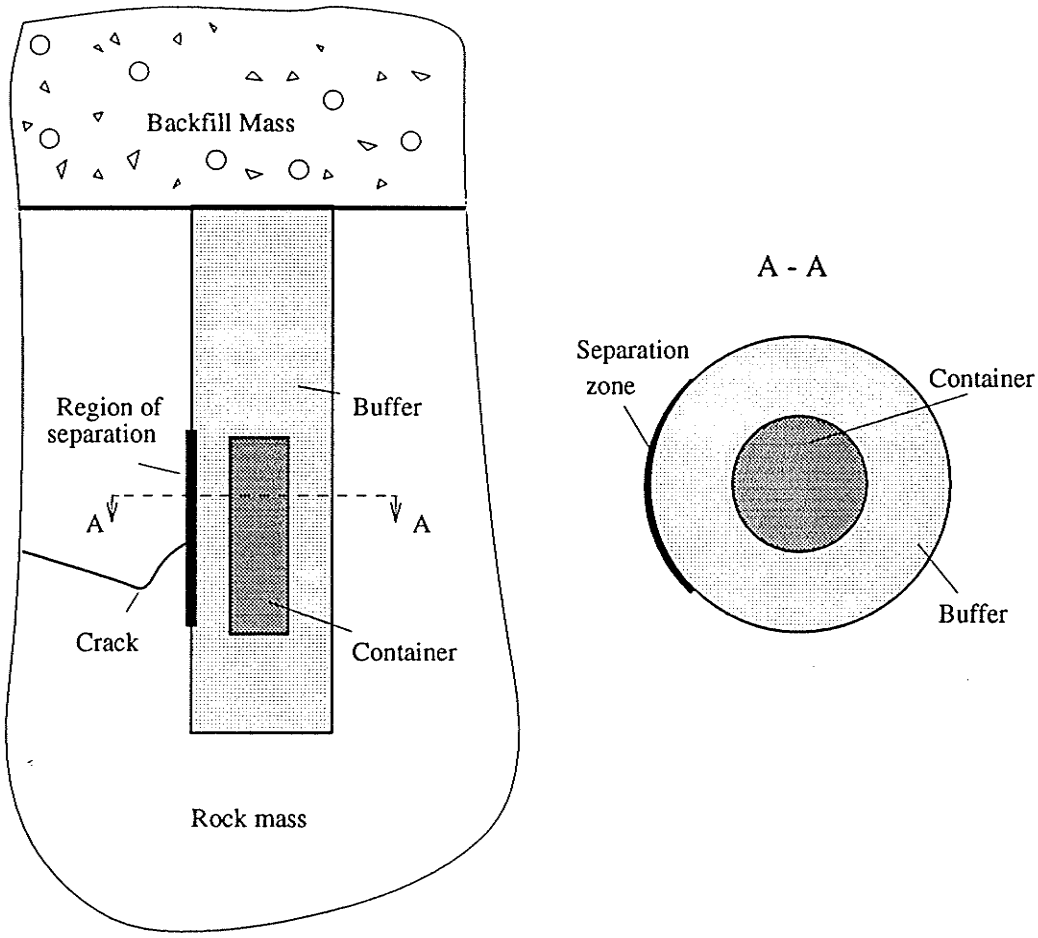
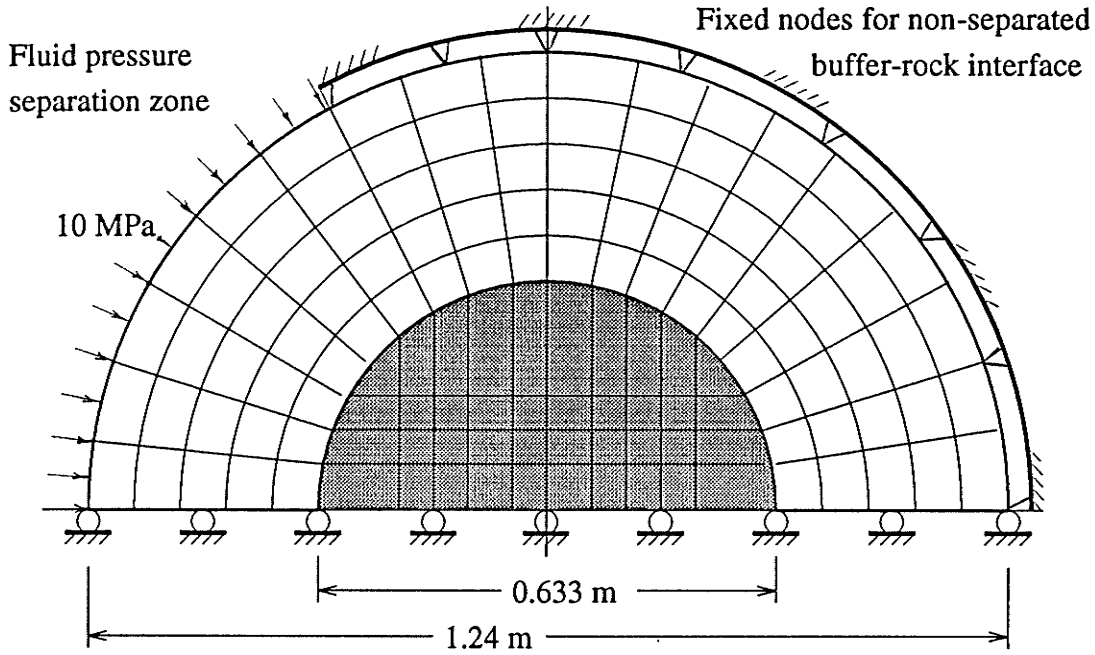
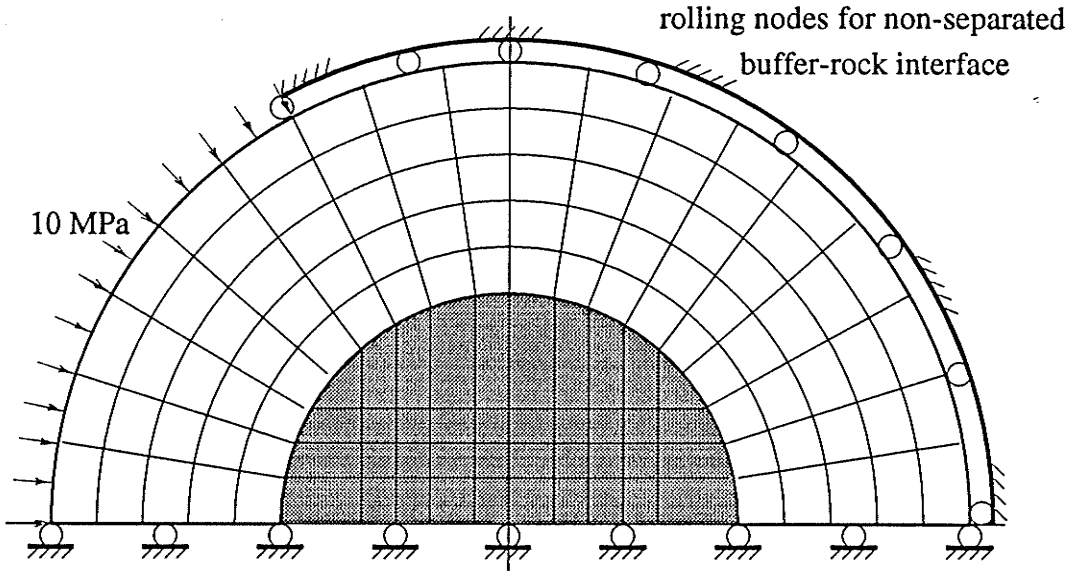


Figure 6.2 Potential fluid-induced separation at the buffer-rock mass interface

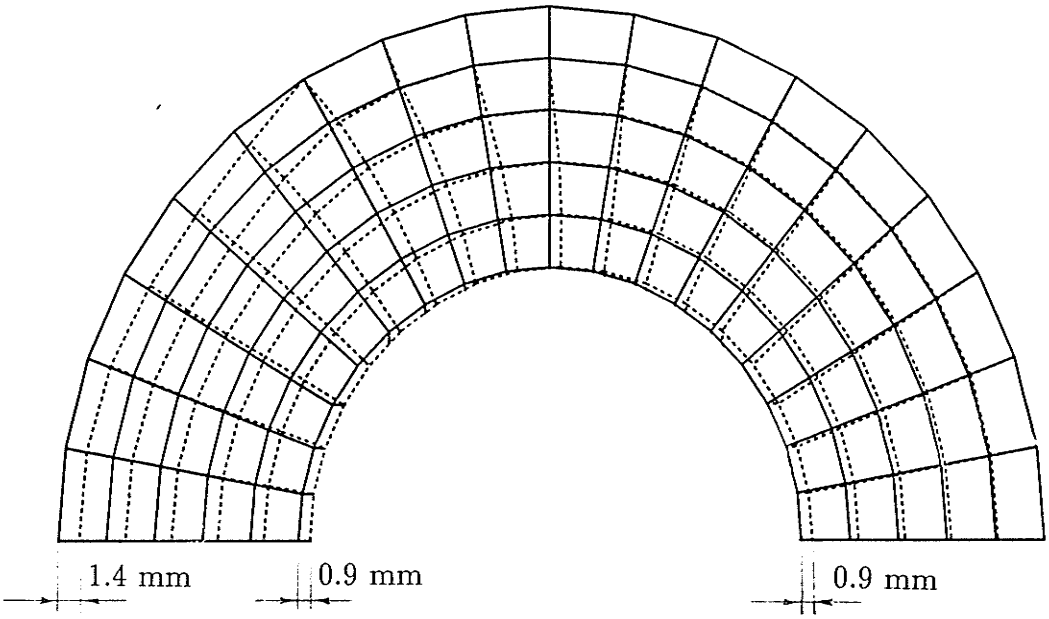


(a) Completely bounded case

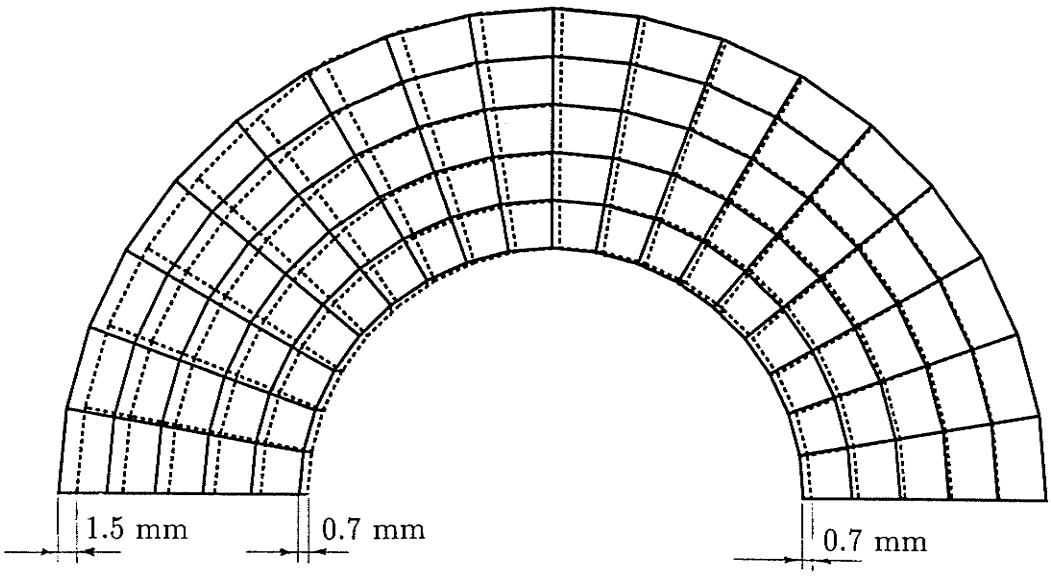


(b) Frictionless bounded case

Figure 6.3 FE mesh for the container-buffer-rock system in plane strain condition

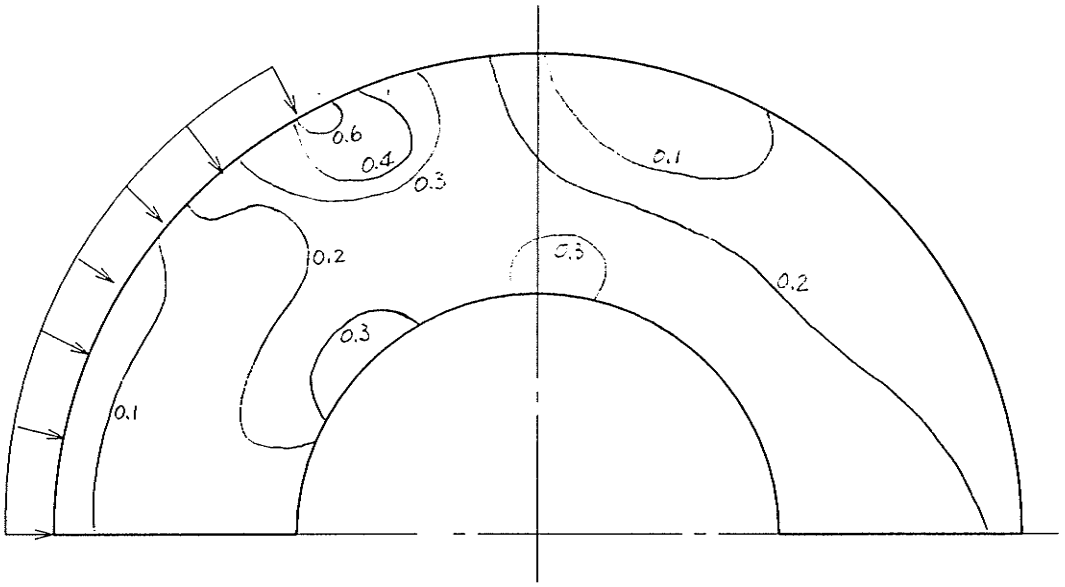


(a) Completely bounded case

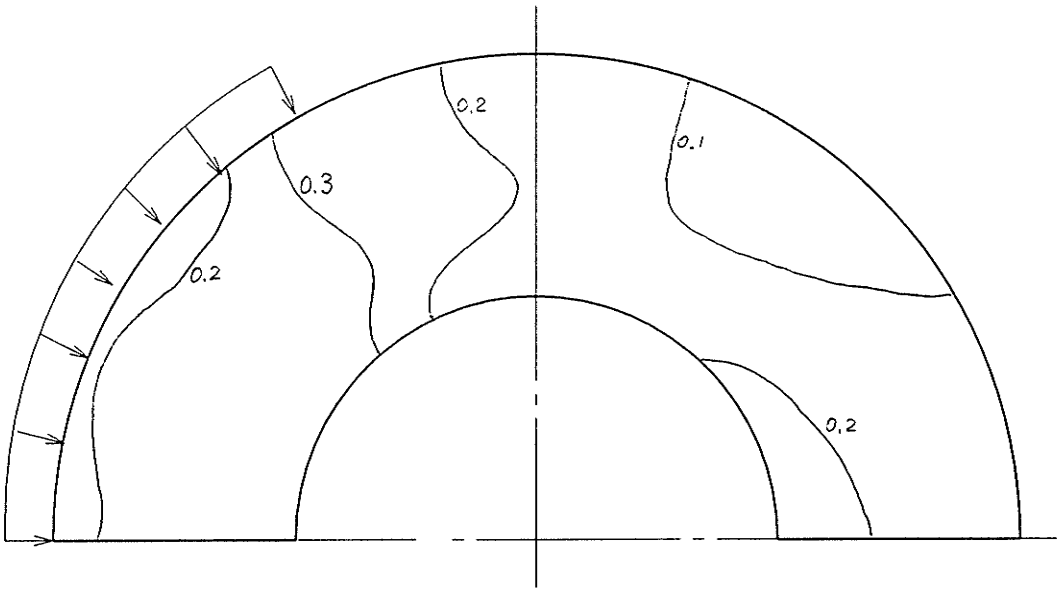


(b) Frictionlessly bounded case

Figure 6.4 Deformation patterns of the buffer due to instant high fluid impaction

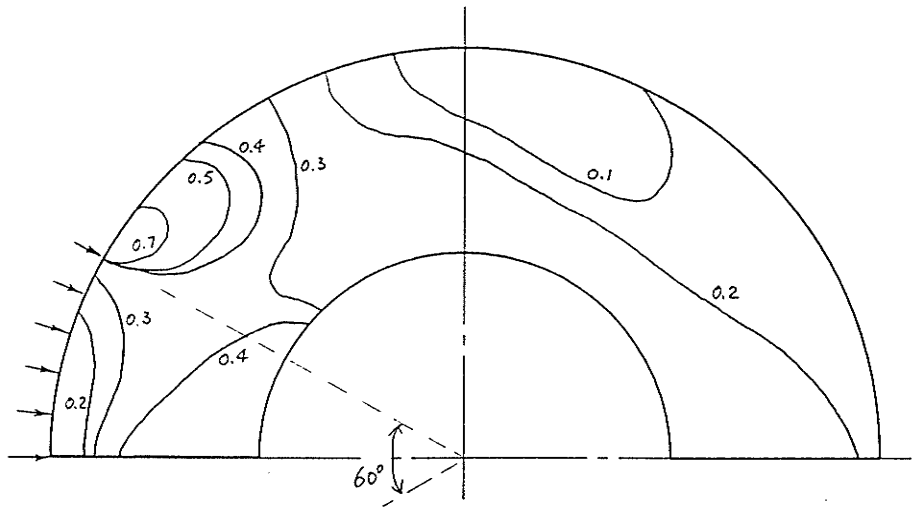


(a) Completely bounded case

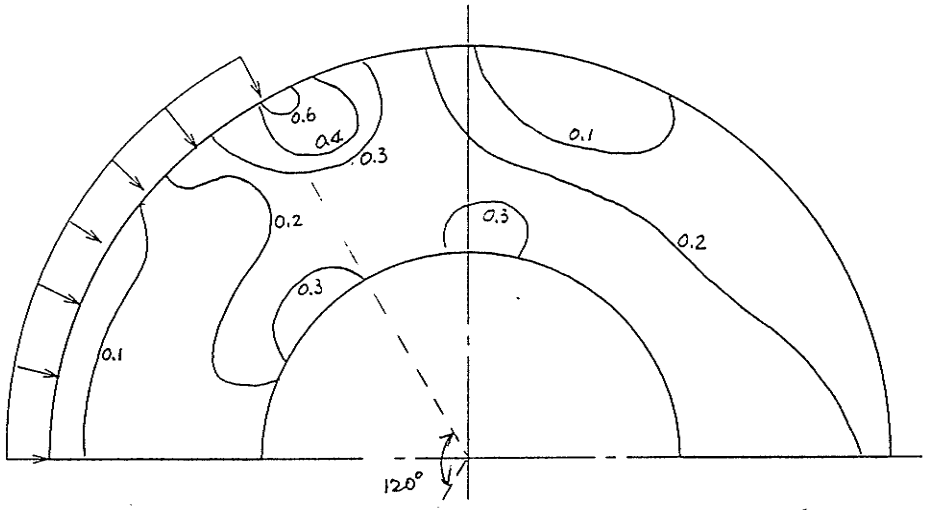


(b) Frictionlessly bounded case

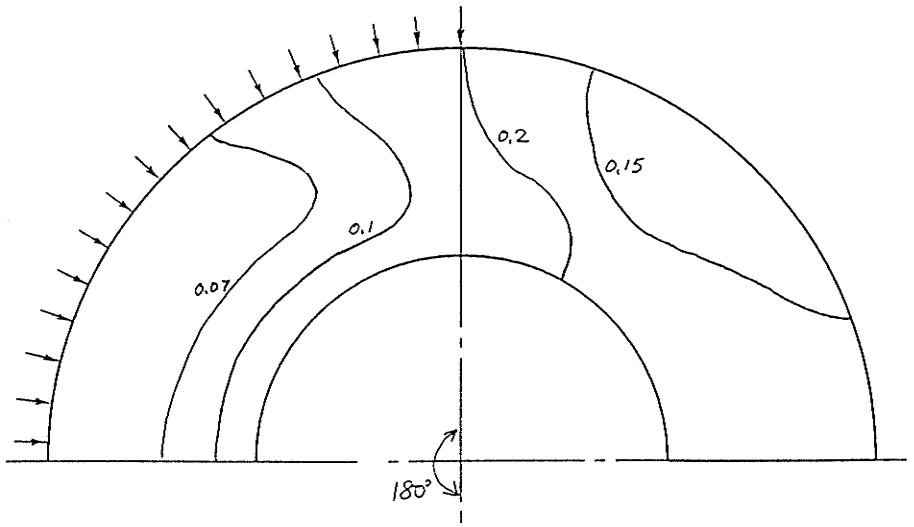
Figure 6.5 Distribution of stress ratio q/p' in the buffer due to impact of instant high fluid pressure



(a) 30^0 half-angle



(b) 60^0 half-angle



(c) 90^0 half-angle

Figure 6.6 q/p' trajectories for the cases of half-angles 30^0, 60^0 and 90^0.

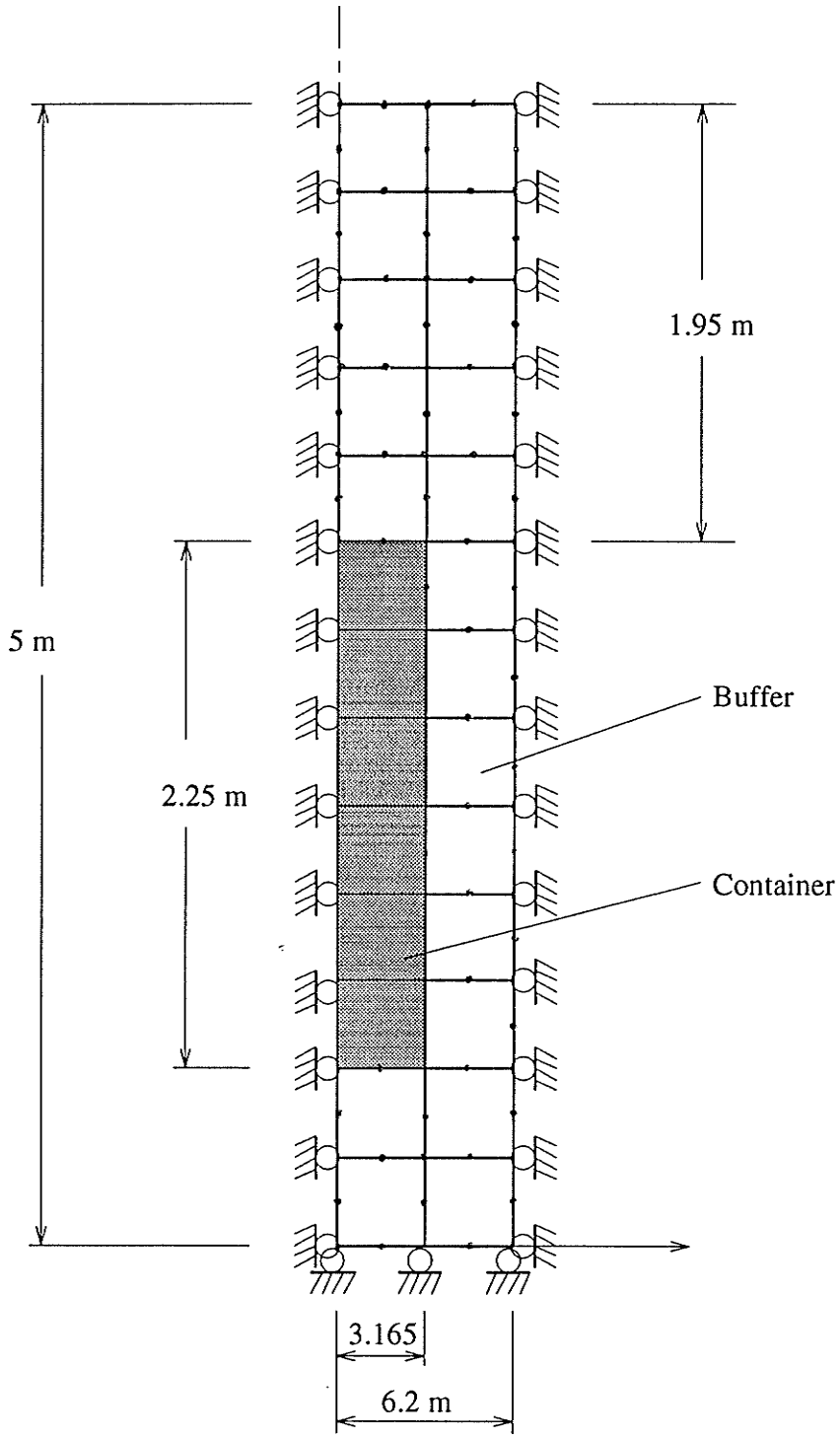
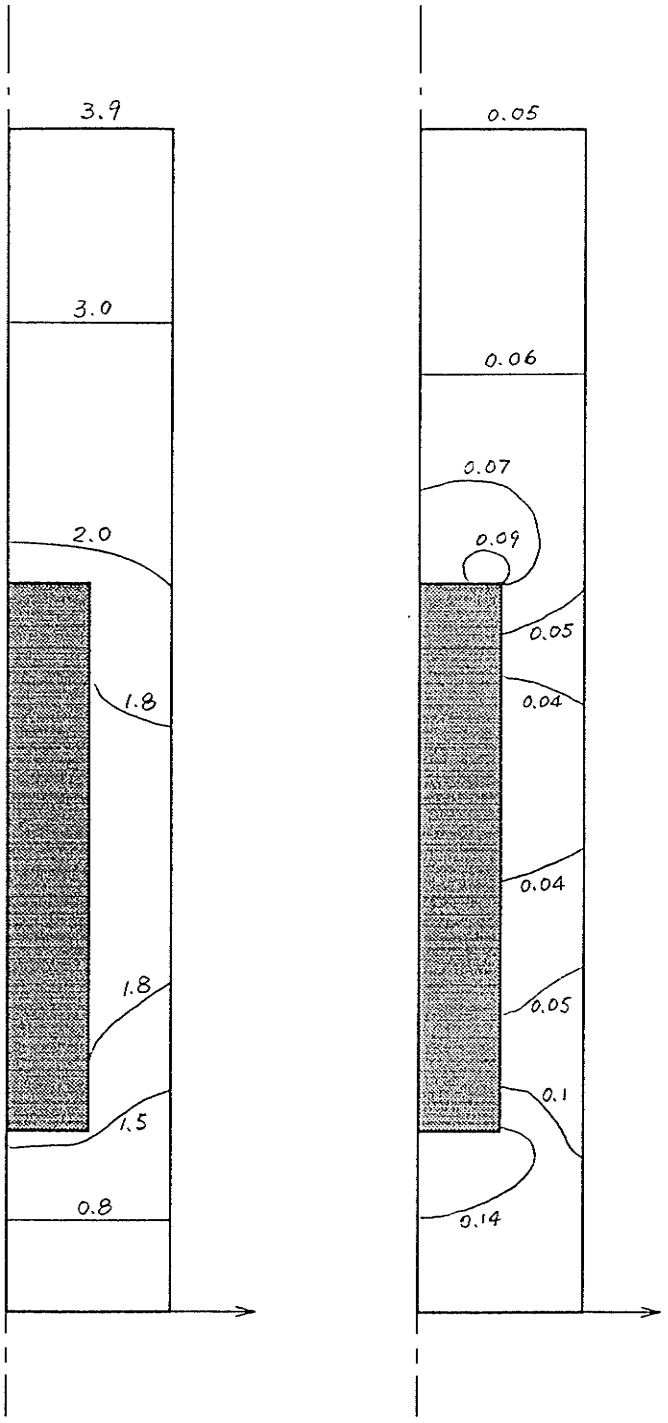
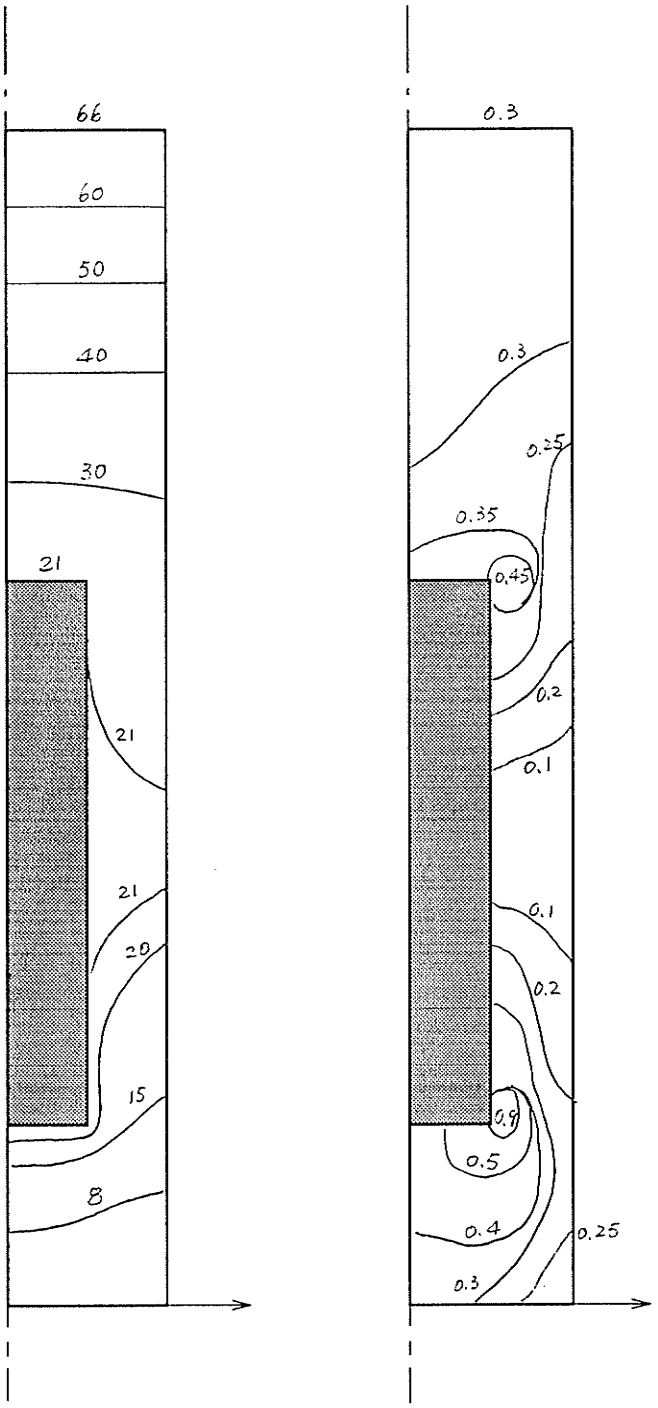


Figure 6.7 FE model of the container-buffer-rock system in axi-symmetric case



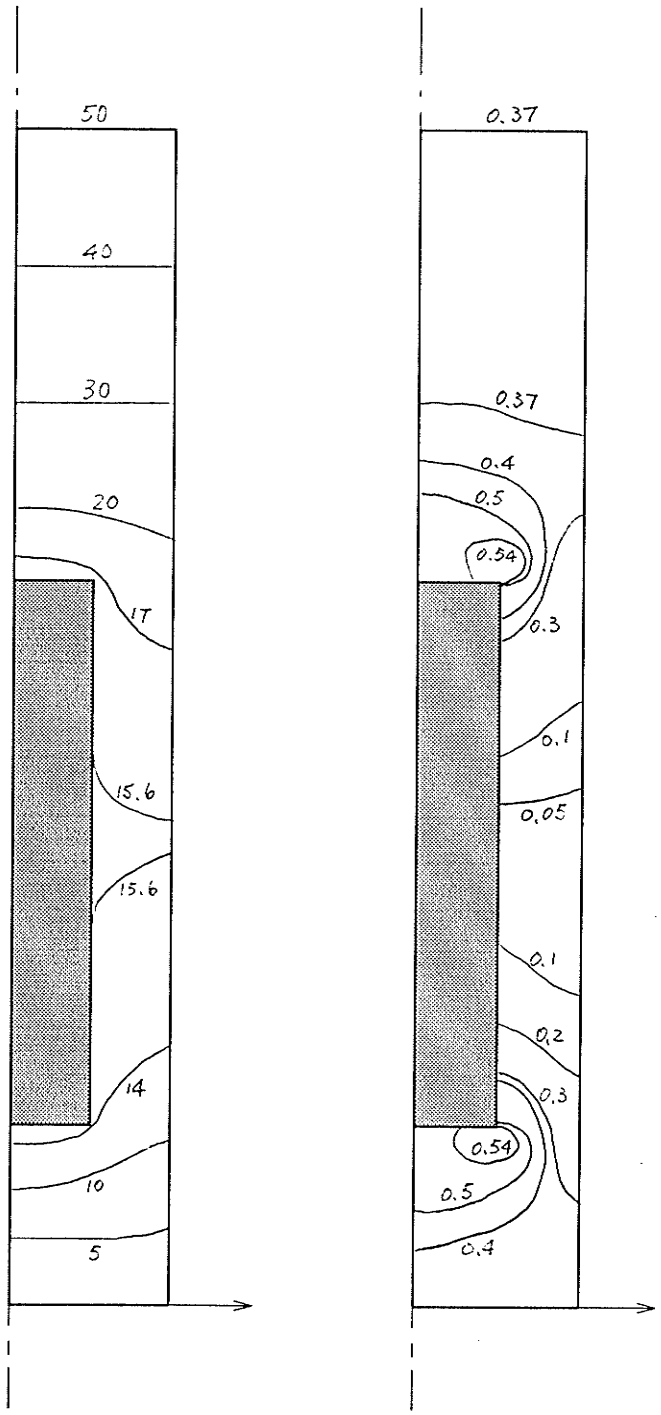
(a) Vertical disp.(mm) (b) q/p'

Figure 6.8. Deformation and stress ratio of the buffer due to its own weight, container weight and backfill surcharge for the case of compaction to $p'_s = 2.0$ MPa



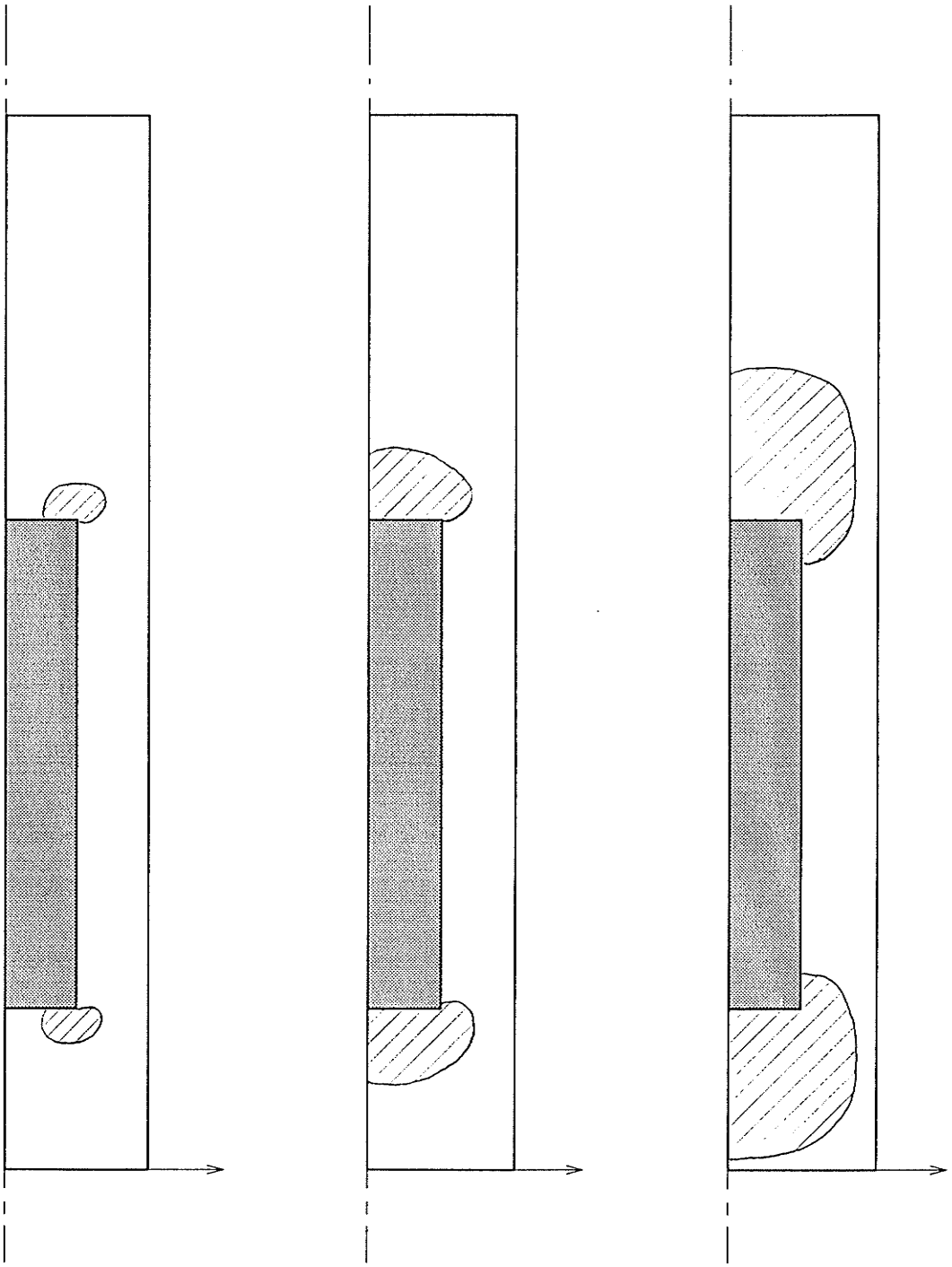
(a) Vertical disp.(mm) (b) q/p'

Figure 6.9. Deformation and stress ratio of the buffer due to its own weight, container weight and backfill surcharge for the case of compaction $p'_s = 0.1$ MPa



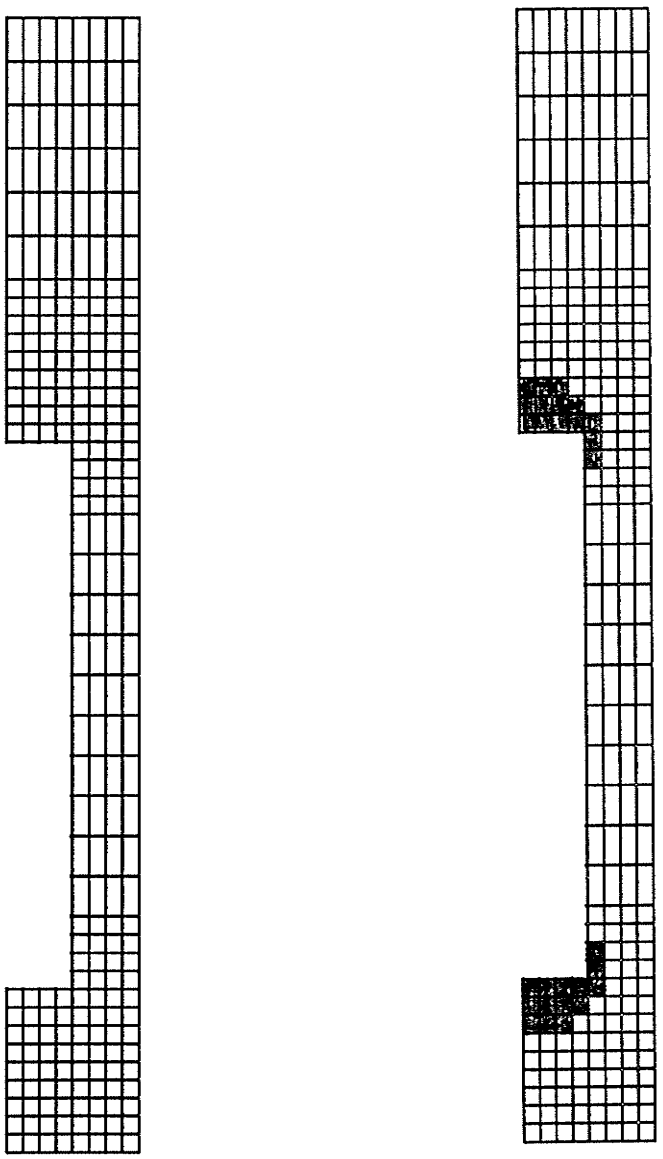
(a) Vertical disp.(mm) (b) q/p'

Figure 6.10 Deformation and stress ratio of the buffer due to partial release of the swelling pressure substantiated by 50 mm lift upwards of the backfill



(a) $\delta = 47$mm (b) $\delta = 50$mm (c) $\delta = 65$mm

Figure 6.11 Plastic yield development in the buffer due to partial release of swelling pressure substantiated by upward lift of the backfill



(a) Refined mesh (b) Plastic zones

Figure 6.12. Plastic yield zones for the case of
50 mm uplift case with refined elements

CHAPTER 7

CONCLUSIONS AND RECOMMENDATIONS

7.1. SUMMARY

The three modulus KGJ hypoelastic model, which is an incremental constitutive relationship expressed in terms of two principal effective stresses and two principal strains was numerically implemented using the finite element method. Much clarification work was done to the original model during the numerical modelling endeavor. Generalization of the KGJ model to multi-axial stress and strain situation was performed. A general symmetric stiffness matrix was obtained which enables the KGJ model to be built into most existing finite element programs without much modification to the original source files. A finite element iterative scheme was formulated to simulate the two-phase nature of the porous media. Four particular forms for saturated soil conditions were deduced to simplify the computational effort.

An appropriate finite element program was developed for implementation of the KGJ hypoelastic model by employing incremental and iterative solution schemes to solve nonlinear boundary value problems associated with soil materials. The computer code was

designed by using the concept of ADT, a data structure whereby the program can be read and maintained easily. A module designated to the constitutive relationship between stress and strain will facilitate modification of the newly-coded model, or extension of the code to additional constitutive models without much effort in changing the computer program. I/O data designed in the code accommodates special requirements such as those for different load path and consecutive computation.

As the KGJ model has only appeared in very recent years, much confirmation and normalization work was performed. The finite element code as well as the model were investigated by analyzing many verification examples where either analytical or experimental results were available. Comparisons of prediction from the KGJ model and the Modified Cam Clay model were made with the experimental results.

The KGJ finite element code was finally applied to several problems in the proposed Canadian program for nuclear fuel waste disposal. The interaction between buffer, container, rock and backfill was modelled under three different vault conditions using the finite element code with KGJ model. The influence of compaction to the buffer strength before the emplacement of the container and backfill was studied.

7.2. CONCLUSIONS

The objectives of the research as outlined in section 1.3 were accomplished satisfactorily. From the results of the research, principal conclusions can be itemized as follows:

(1) The finite element code incorporating hypoelastic constitutive equations provides a flexible and versatile means of solving buffer-structure interaction problems in which anisotropic nonlinearity is prevalent. Solutions obtained by the finite element code can match the analytical results excellently and yield satisfactory agreement with experimental data.

(2) The simple three-modulus (KGJ) hypoelasticity model can be used to account well for the nonlinear, transversely isotropic behavior of the sand-bentonite buffer material. Compared to the findings from laboratory test data, the numerical KGJ model can predict more precisely the stress-strain behavior of the buffer soil than the isotropic elastoplastic Cam Clay model does. However it is noted that hypoelastic models generally require more laboratory calibration than more general elastic plastic models.

(3) Compaction to the buffer soil before emplacement of the waste container and sealing of the vault plays an important role in the mechanical behavior of the buffer under its functioning conditions. Compaction to high density (high swelling pressure) can decrease the possibility of damage from high interstitial fluid pressure through rock mass cracks, upper structure load and large scale swelling pressure release. Low compaction may allow full shear resistance mobilization in sizable regions within the buffer.

(4) The impact on the buffer from sudden high interstitial water pressure does not appear to constitute a threat to the mechanical function of the buffer-canister vault system. The frictional angle between the buffer and the borehole wall is finite but small ($\Phi \approx 15^0$) and will produce more conservative results than the frictionlessly bounded case.

7.3. REMARK FOR THE NEAR FUTURE STUDY

In the author's point of view, there will be no end of further study for any promising research field in science. Thus only research items amenable in the near future are listed below for consideration in connection with this research.

- Incorporation of other hypoelasticity or hyperelasticity models into the code. This work can be conveniently accomplished. For instance, incorporation of the commonly used Duncan-Chang KG nonlinear model may be used to compare the results predicted by Modified Cam Clay model and KGJ hypoelasticity model to the buffer-strucure interaction problems.

- The modelling of the buffer problems is preliminary in nature: this thesis has concentrated on implementing the model and verifying the coding. More detailed modelling and parametric studies remain to be done.

- Broadening the element library to include more element types such as 3D elements and contact elements. Since practical situations are three dimensional and the contact phenomenon is the most common engineering mechanism, this further work can promote the research work more closely toward geotechnical engineering practice.

- To perfect the KGJ model, it would be interesting to compare the predictability of the model to the unloading/reloading conditions. More experinmental data are needed for this performance.

REFERENCES

Britto, A.M. and Gunn, M.J. (1985), *critical State Soil Mechanics via Finite Elements*, Ellis Horwood Ltd., England.

Bowles, J.E. (1977),*Foundation Analysis and Design*, McGraw-Hill, NY, 2nd edn.

Cameron, D.J. (1982), Fuel isolation research for the Canadian nuclear fuel waste management program, Atomic Energy of Canada Ltd. Report AECL-6834.

Cook, R.D. (1989), *Concepts and Applications Finite Element Analysis*, 6th edn, Wiley, NJ.

Craig,R.F. (1983), *Soil Mechanics*, 3rd edn., Van Nostrand Reinhold, UK.

Desai, C.S. and Gioda, G. (1990), *Numerical Methods and Constitutive Modelling in Geomechanics.*

Graham J. and Houlsby, G.T. (1983), Anisotropic Elasticity of a Natural Clay, *Geotechnique*, **33**, 165-180.

Graham, J.(1989), *Elastic-plastic Soil Mechanics with Applications in Geotechnical Engineering*, Lecture notes presented at Univ. of Hong Kong and Univ. of Manitoba.

Gudehus, G. (1977), *Finite Element in Geomechanics*, John Wiley & Sons Ltd.

Helman, P., Veroff, R. and Carrano, F.M. (1991), *Intermediate Problem Solving and Data Structures, Walls and Mirrors*, Benjamin/Cummings Publishing Company, Inc..

Houlsby, G.T.(1981), *A Study of Plasticity Theories and Their Applicability to soils*, PhD thesis, University of Cambridge.

Lewis, R.W. and Schrefler, B.A. (1987), *The Finite Element Method in the Deformation and Consolidation of Porous Media*, John Wiley & Sons Ltd.

Mroz, Z. (1980), On Hypoelasticity and Plasticity Approaches to Constitutive Modelling of Inelastic Behacior of Soil, *I. J. Num. Ana. Methods in Geomechanics*, **4**, 45-55.

Pusch, R., Nilsson, J. and Ramquist, G (1985), Final Report of the Buffer Mass Test, Vol. 1, July, TR 85-11 Stripa Project. SKB, Stockholm, Sweden.

Saadat, F (1989), *Constitutive Modelling of the Behavior of a Sand-bentonite Mixture*, Ph.D Thesis, The University of Manitoba.

Salvadurai, A.P.S. (1979), *Elastic Analysis of Soil-Foundation Interaction*, NY, Elsevier Scientific Publishing Company.

Salvadurai, A.P.S., Lopez, R.S. and Hartley, G.A. (1985), Geomechanical iteraction in a nuclear waste disposal vault, *Proc. of the International Conference on Soil Mechanics and Foundation Engineering*, Vol.3.

Sandhu, R.S. and Wilson, E.L. (1969), Finite Element Analysis of Flow in Saturated Porous Media, *Proc. ASCE*, **95**, EM, 641-652.

Olivier, J.P. (1986), Radioactive waste management in OECD countries national programs and joint activities, Proc. of the 2nd International Conference on Radioactive Waste Management, Winnipeg, Manitoba.

Rummery, T.E. and Rossinger, E.L.J. (1983), The Canadian Nuclear waste management program., *Proc. of Canadian Nuclear Society*, International Conference on Radioactive Waste Management, Winnipeg, Canada.

Truesdell, C. (1955), Hypo-elasticity, *J. of Rational Mechanics and Analysis*, **4**(1), 83-133.

White, F.M. (1986), *Fluid Mechanics*, McGraw Hill, NY.

Wood, D.M.(1990), *Soil Behavior and Critical State soil Mechanics*, Cambridge University Press.

Yin, J.H., Graham, J., Saadat, F. and Azizi, F. (1989), Constitutive Modelling of Soil Behaviour Using Three Modulus Hypoelasticity, *Proc. of the 12the International Conference on Soil Mechanics and Foundation Engineering*, Rio De Janeiro, August, 1989.

Yin, J.H., Saadat, F. and Graham J. (1988), A Three Modulus Hypoelastic Constitutive Model for Sand-bentonite Buffer (RBM), *Proc. 41st Canadian Geotechnical Conference*, Waterloo, Ontario, Oct., 1988.

Yin, J.H., Saadat, F. and Graham, J. (1990), Constitutive Modelling of a Compacted Sand-bentonite Mixture Using Three-modulus Hypoelasticity, *Can. Geotech. J.* **27**, 365-372.

Yin, J.H. (1990), Constitutive Modelling of Time-Dependent Stress-Strain Behavior of Soil, Ph.D. Thesis, The University of Manitoba.

Zinkiewicz, O.C. (1977), *The Finite Element Method*, 3rd ed., McGraw-Hill Co., Maidenhead, Berkshire, England.

APPENDIX A

The following mini-programs and data are used for verification of the finite element modelling code, see Chapter 5.

1. RUNGE-KUTTA SOLUTION OF ORDINARY DIFFERENTIAL EQUATIONS FOR BUFFER SOIL (ANALYTICAL SOLUTION):

```
C
C    bufruku.f
C
C*** This program is used to evaluate the Eps and q by 4th order
C    Runge-Kutta method in ordinary differential equation problem
C
C Input Data:
C-------------------------
C| DLLW, DLUP, DX, PRTDX |
C-------------------------
C          The lower delimit, upper delimit, integral increment, output
C          span
C
C---------------------------
C| RLV, EPSV0, A, RN, E, F |
C---------------------------
C          The KGJ curve fitting parameters.
C
C*********************************************************************
C
C       IMPLICIT REAL*8 (A-H,O-Z)
       DIMENSION EPS(10001),QQ(10001)
       READ(5,*) DLLW, DLUP,DX,PRTDX
       READ(5,*) RLV, EPSV0, A, RN, E, F
       EPS(1)=DLLW
       QQ(1)=DX
       N=(DLUP-DLLW)/DX
       IF (N.GT.10000) THEN
        WRITE(6,*)'THE STEP LENGTH DX TOO SMALL, CAN NOT GO ON'
        WRITE(6,*)'N=',N
        READ(5,*)ANY
```

```
        ENDIF
         CALL RUKU(EPS,QQ,N,DLUP,    RLV,EPSV0,A,RN,E,F)
        STEP=0.0
        DO 50 ISTEP=1,N+1
        STEP=STEP+DX
        IF ((STEP.GE.PRTDX).OR.(EPS(ISTEP).GE.DLUP))
     .   WRITE(6,900) EPS(ISTEP),QQ(ISTEP)
        IF (STEP.GE.PRTDX) STEP=0.0
  50    CONTINUE
 900    FORMAT(2X,F16.8,2X,F16.8)
        STOP
        END
C
C**** Please define your function:
C
        FUNCTION FTN(EPS,Q,   RLV,EPSV0,A,RN,E,F,PEFFC)
C        IMPLICIT REAL*8 (A-H,O-Z)
C
C** The analytical solution of Eps vs q/p' for buffer soil
C
C        PEFFC=3.0
        PEFFC=1.0
        RKV=1.0/80.0
        D=(1.0-RN)/RN
        C=RN*A**(1.0/RN)
        IF (RN.EQ.1.0) THEN
         EPSV=RLV*LOG(PEFFC)+EPSV0
        ELSE
         EPSV=RLV*LOG(PEFFC)+EPSV0+RLV*LOG(1.0
     .       +D/((1.0+D)*C)*(Q/PEFFC)**(1.0+D))/D
        ENDIF
        PCONS=EXP((EPSV-EPSV0)/RLV)
        RK=PEFFC/RLV
        RJ=RK*C*(PCONS/Q)**D
        TD=PCONS*(1.0-F*Q/PCONS)**2/(3.0*E)
        G=TD*RJ*RJ/(RJ*RJ+3.0*TD*RK)
        FTN=3.0*G
        RETURN
        END
C
         SUBROUTINE RUKU(X,Y,N,B,   RLV,EPSV0,A,RN,E,F)
C
```

```
C*************************************************************
C   THIS SUBROUTINE COMPUTES THE SOLUTION OF A DIFFERNTIAL EQUATION
C   Y'=F(X,Y) OVER THE INTERVAL [X(1),B] USING THE CLASSICAL 4-TH
C   ORDER RUNGE-KUTTA METHOD
C   INPUT DATA:
C      X(1)=INDEPENDENT VARIABLE INITIAL VALUE
C      Y(1)=DEPENDENT VARIABLE INITIAL VALUE
C         N=NUMBER OF STEPS
C         B=SOLUTION INTERVAL ENDPOINT (LAST X VALUE)
C   OUPPUT DATA:
C       X[1..N+1] ARRAY OF INDEPENDENT VARIABLE VALUES
C       Y[1..N+1] ARRAY OF DEPENDENT VARIABLE SOLUTION VALUES
C*************************************************************
C
C       IMPLICIT REAL*8 (A-H,O-Z)
        DIMENSION X(N+1), Y(N+1), Z(10001)
        H=(B-X(1))/N
        DO 1 I=2, N+1
         X(I)=X(I-1)+H
         U=X(I-1)
         V=Y(I-1)
         T1=FTN(U,V,   RLV,EPSV0,A,RN,E,F,PCONS)
         U=U+0.5*H
         V=V+0.5*H*T1
         T2=FTN(U,V,   RLV,EPSV0,A,RN,E,F,PCONS)
         V=Y(I-1)+0.5*H*T2
         T3=FTN(U,V,   RLV,EPSV0,A,RN,E,F,PCONS)
         U=U+0.5*H
         V=Y(I-1)+H*T3
         T4=FTN(U,V,   RLV,EPSV0,A,RN,E,F,PCONS)
         Y(I)=Y(I-1)+H*(T1+2.0*T2+2.0*T3+T4)/6.0
         G=FTN(X(I),Y(I),   RLV,EPSV0,A,RN,E,F,PCONS)
C
C  *** to obtain q/p'
C
         Z(I)=Y(I)/PCONS
   1    CONTINUE
        DO 2 I=2, N+1
   2    Y(I)=Z(I)
        RETURN
        END
C  0.0, 0.1, 0.0005, 0.001       {for Epv-q/p', use Runge-Kuta method}
```

```
C  0.073, -0.003, 1.32, 0.71, 0.0086, 2.15

C  0.0, 0.1, 0.0005, 0.001        {for Epv-q/p', use Runge-Kuta method}
C  0.052, -0.039, 1.25, 0.65, 0.0108, 1.83
```

2. EULER INTEGRATION METHOD USED FOR WUHAN SAND (ANALYTICAL SOLUTION):

```
C
C     wuhan.f
C
C***  This program predicts the q vs (Ep)s for Wuhan sand
C***  by euler integration method
C
C***  Condition:  p'=p'cons=constant
C***  The computation starts from QMIN till QMAX, with increment
C***  equal to DQ, and prints out every STEP interval of Q
C
        IMPLICIT REAL*8 (A-H,O-Z)
        OPEN(10,FILE='esq',STATUS='NEW')
        OPEN(11,FILE='esv',STATUS='NEW')
        READ(5,*)QMIN,QMAX,DQ,PCONS,STEP
        R=6.867
        S=39.4
        T=0.0042276
        U0=2.483694
        UULT=3.548134
        RM=0.85
        V=2.81838D-3
        W=0.281838
        Q=QMIN
        DP=0.0
        EPS=0.0
        EPV=0.0
        PEFFC=PCONS
        XG3=PCONS
        STEPS=0.0
        DO 100 WHILE (Q.LT.QMAX)
         Q=Q+DQ
         XG3=XG3-DQ/3.0
```

```
          XG3RM=XG3**RM
          STEPS=STEPS+DQ
          U=Q/XG3RM
          RK=(R+S*PEFFC)**2/R
          RJE=XG3RM*(U-UULT)**2/(U*U-2.0*U*UULT+U0*UULT)/T
          GE=XG3RM*(1.0-W*Q/XG3RM)**2/(3.0*V)
          RJ=3.0*RK*RJE/(3.0*RK-RJE)
          G=3.0*GE*RJ/(3.0*RJ-3.0*GE)
          DEPV=DP/RK+DQ/RJ
          DEPS=DP/RJ+DQ/(3.0*G)
          EPV=EPV+DEPV
          EPS=EPS+DEPS
          IF (STEPS.GE.STEP) THEN
           STEPS=0.0
           WRITE(10,900) EPS,Q
           WRITE(11,900) EPS,-EPV
          ENDIF
  100    CONTINUE
  900    FORMAT(2X,F12.5,2X,F12.5)
         CLOSE(10)
         CLOSE(11)
         STOP
         END

C  0.0, 0.4, 0.005, 0.2, 0.01   {for wuhan sand, by Euler method}
```

APPENDIX B

KGJ DATA FILE USED FOR EXAMPLES 1-4 IN CHAPTER 5.

```
21,9,4,3,8,1,2,  file:k1
0, 210, -1, 20 , 1, 10 ,3
0.005 ,0.0,0.0,   100.0,0.0,0.0,  0.1E-5
1,1, 9,10,11,7,3,2,1,6
2,1,11,12,13,8,5,4,3,7
3,1,17,18,19,15,11,10,9,14
4,1,19,20,21,16,13,12,11,15
1, 0.0, 20.0
2, 2.5, 20.0
3, 5.0, 20.0
4, 7.5, 20.0
5, 10.0, 20.0
6, 0.0, 15.0
7, 5.0, 15.0
8, 10.0, 15.0
9, 0.0, 10.0
10,2.5, 10.0
11,5.0, 10.0
12,7.5, 10.0
13,10.0,10.0
14,0.0, 5.0
15,5.0, 5.0
16,10.0, 5.0
17,0.0,0.0
18,2.5,0.0
19,5.0,0.0
20,7.5,0.0
21,10.0,0.0
1, 10, 0.0, 0.0
6, 10, 0.0, 0.0
9, 10, 0.0, 0.0
14, 10, 0.0, 0.0
17, 11, 0.0, 0.0
18, 01, 0.0, 0.0
19, 01, 0.0, 0.0
20, 01, 0.0, 0.0
```

```
21, 01, 0.0, 0.0
1
0.0,0.0,5130.9E2, 0.0
0.073, 80.0, 0.0068,1.32,0.71,0.0086, 2.15, 37.5, 3.00, 0.087  Line 40
0,0,4,0
12.4, 0.0
1, 3,2,1
0.1,0.0,0.1,0.0,0.1,0.0
2, 5,4,3
0.1,0.0,0.1,0.0,0.1,0.0
-2, 13,8,5
0.1,0.0,0.1,0.0,0.1,0.0
-4, 21,16,13
0.1,0.0,0.1,0.0,0.1,0.0
```

Note: sustitute the following lines under each individual example problem into the corresponding lines in above data file before running the program named 'kg'.

```
Buffer 1.5 Mg/m3      Example 5.1
0, 200, -1, 4, 50, 10 ,1    Line 2
0.01,0.0,0.0,10.0,0.0,0.0,  0.1E-4 Line 3
0.0,0.0,0.0,0.0,
0.073, 80.0, 0.00, 1.32, 0.71 0.0086, 2.15, 37.5, 1.00, 0.00  Line 40
2.0, -1.0  Line 42
Buffer 1.66 Mg/m3     Example 5.1
0, 250, -1, 4, 50, 10 ,1    Line 2
0.01,0.0,0.0,10.0,0.0,0.0,  0.1E-4 Line 3
0.0,0.0,0.0,0.0,
0.052, 80.0, 0.0, 1.25, 0.65, 0.0108, 1.83, 37.5, 1.00, 0.00  Line 40
2.0, -1.0  Line 42

Wuhan sand p'cons=0.2 MPa    Example 5.2
0, 200, -1, 4 , 50, 10 ,1  Line 2
0.01,0.,0.,100.0,0.0,0.0,  0.1E-6   Line 3
0.0,0.0,0.0,0.0,
-9999.0, 80.0, 0.00, 0.0, 0.0, 0.0, 0.0, 40.0, 0.2, 0.00    Line 40
2.0, -1.0  Line 42
Wuhan sand p'cons=0.4 MPa     Example 5.2
0, 300, -1, 4 , 50, 10 ,1  Line 2
0.01,0.2,0.5,100.0,100.0,1000.0,0.1E-6   Line 3
```

```
0.0,0.0,0.0,0.0,
-9999.0, 80.0, 0.00, 0.0, 0.0, 0.0, 0.0, 40.0, 0.4, 0.00    Line 40
2.0, -1.0  Line 42

T915 sample      Example 5.3
0, 210, -1, 20 , 1, 10 ,3   Undrained condition, using    Line 2
0.005 ,0.0,0.0,   100.0,0.0,0.0,  0.1E-5   total stress path  Line3
0.0,0.0,5130.9E2, 0.0
0.073, 80.0, 0.0068,1.32,0.71,0.0086, 2.15, 37.5, 3.00, 0.087  Line 40
12.4, 0.0   Line 42

T910 sample    Example 5.4
0, 130, -1, 4 , 50, 10 ,1    Line 2,
0.02 ,0.0,0.0,   100.0,0.0,0.0,  0.1E-5 Line 3
0.0,0.0,0.0, 0.73
0.073, 80.0, 0.056,1.32,0.71,0.0086, 2.15, 37.5, 1.2, 0.069  Line 40
2.0, -1.0   Line 42
```

APPENDIX C

The FORTRAN program used in the modelling work:

```
C
C    kg.f
C
C//RZHANG$ JOB ',,T=8,',REGION=3500K
C// EXEC WATFIV MAP=NOMAP
C//SYSIN DD *
C$JOB  WATFIV  NOWARN,NOEXT
C*****************************************************************
C
C                INPUT DATA FOR HYPOELASTIC ANALYSIS
C
C-------------------------------------------------
C| NPOIN, NVFIX, NELEM, NTYPE, NNODE, NMATS, NGAUS |
C-------------------------------------------------
C                NPOIN---total number of nodal points
C                NVFIX---total number of nodes with restrained freedom
C                NTYPE---problem type parameter
C                          1, plane stress with thickness=1.0
C                          2, plane strain
C                          3, axial symmetry
C                NNODE---number of nodes per element, 4 or 8 or 9
C                NMATS---total number of different materials
C                NGAUS---order of Gaussian quadrature rule, 2 or 3
C
C-------------------------------------------------
C| NINIT, NINTV, NKEYP, NSTEP, MITER, NALGO, NSELE |
C-------------------------------------------------
C                NINIT---initial state control parameter
C                          0, initialize stress and displacement field
C                          1, restore all results from previous comput-
C                             ation
C                          2, inherit the stress field only, usually
C                             followed from self-weight computation
C                NINTV---number of external load increments
C                NKEYP---number of key points where the displacements
C                        will be output at every NSTEP time intervals
C                         -1, output all displacements and stresses at
```

```
C                              every NSTEP time intervals
C                NSTEP---output selector, output every NSTEP incremental
C                       steps
C                MITER---maximum iterative loops allowed
C                NALGO---algorithm selector, Now it's for the Guassin
C                         point where the values of p',q and epss and
C                         epsv are output
C                NSELE---1, for fully drained problem,
C                       2, for undrained problem, fluid compressable,
C                       3, for undrained triaxial test, incompressible.
C
Cif NKEYP.ne.0, then
C--------------------------
C| NKOUT1, ..., NKOUTnkeyp |
C--------------------------
C                NKOUT1, ..., NKOUTnkeyp---the codes of the key node
C
C-------------------------------------------------
C| DLMD1, DLMD2, DLMD3, TLMD1, TLMD2, TLMD3, CONVR |
C-------------------------------------------------
C     Control parameters for exerting loads during incremental steps.
C                DLMD1---Delta lambda 1
C                DLMD2---Delta lambda 2
C                DLMD3---Delta lambda 3
C                TLMD1---Lambda 1
C                TLMD2---Lambda 2
C                TLMD3---Lambda 3
C                CONVR---Convergent criterion
C       The formulae for counting the number of the load steps,NSTEPS:
C           NSTEPS1=TLMD1/DLMD1,
C           NSTEPS2=TLMD2/DLMD2,
C           NSTEPS3=TLMD3/DLMD3.
C       The load applied will be:
C           dp=DLMD*(PERCT1*pcase1+PERCT2*pcase2)
C
C
C---------------------------------------
C| NUMEL, MATNO, LNODS1, ..., LNODSnnode |
C---------------------------------------
C ...,  ...
C      totally NELEM proups, where
C
```

```
C                4        3           7      6      5
C                +--------+           +------+------+
C                |        |           |             |
C                |        |          8+    (9+)     +4
C                |        |           |             |
C                +--------+           +------+------+
C                1        2           1      2      3
C
C                NUMEL---element number
C                MATNO---material property number
C                LNODS1, ..., LNODSnnode
C                     ---element nodal connection numbers
C
C-------------------------
C| IPOIN, COORD1, COORD2 |
C-------------------------
C...,  ...
C     totally NPOIN groups, where
C                IPOIN---node number
C                COORD1---x (or r) coordinate of the node
C                COORD2---y (or z) coordinate of the node
C
C--------------------------------
C| NOFIX, IFPRE, PRESC1, PRESC2 |
C--------------------------------
C...,  ...
C     totally NVFIX groups, where
C                NOFIX---restrained node number
C                IFPRE---restraint code
C                    10, displacement restrained in x (or r) direction
C                    01, displacement restrained in y (or z) direction
C                    11, displacenent restrained in both directions
C                   note: for skew (oblique) supports, IFPRE<0 and refer
C                         to the restraints in local reference frame
C                PRESC1,PRESC2---the prescribed value of x (or r) and
C                        y (or z) displacement components, respectively
C                   note: for skew supports, PRESC1 is the angle of the
C                         local reference frame counterclockwise from t
C                         global reference frame (in dgree), PRESC2=0.0
C
C---------
C| NUMAT |
```

```
C---------
C-------------------------------
C| PROPS1, PROPS2, ..., PROPS12 |
C-------------------------------
C...,  ...
C     totally NMATS groups, where
C                NUMAT---group number of material properties
C                PROPS1---Young's modulus, for elasticity problem only.
C                         Note: zero value is preserved for hypo-
C                         elastic material.
C                PROPS2---Poisson's ratio, for elasticity problem only
C                PROPS3---1/M=beta/n, the water compressibility, where
C                         beta is the bulk modulus of fluid and n is
C                         the porosity of soil.
C                PROPS4---Density
C                  The rest are for hypoelasticity
C                PROPS5---Vlambda
C                PROPS6---Vkappa
C                PROPS7---ev0
C                PROPS8---a
C                PROPS9---n
C                PROPS10---e
C                PROPS11---f
C                PROPS12---s
C                PROPS13---p'0, the initial static hydraulic pressure,=0
C                PROPS14---ev0, the initial volumetric strain field
C
C***
C-----------------------------
C| NPLOD, IGRAV, NEDGE, NLOAD |
C-----------------------------
C                NPLOD---number of points on which concentrate forces
C                           are loaded
C                IGRAV---gravity loading control parameter:
C                        0, no gravity loads to be considered
C                        1, gravity loading to considered
C                NEDGE---number of element edges on which distributes
C                           loads are to be applied
C                NLOAD---number of load increments
C
C-----------------
C| PERCT1, PERCT2 |
```

```
C-----------------
C  percentages of load case 1 and 2. Note: loads can be discribed by the
C  combination of two basic cases of loads. Thus different percentage of
C  magnitudes may be applied during the incremental steps.
C
Cif NPLOD.ne.0 then
C------------------------
C| LODPT, POINT1, POINT2 |
C------------------------
C...,  ...
C     totally IPLOD groups, where
C                LODPT---node number, >0 for case 1; <0 for case 2
C                POINT1---load component in x (or r) direction
C                POINT2---load component in y (or z) direction
C
Cif IGRAV.ne.0 then
C----------------
C| THETA, GRAVY |
C----------------
C     for load case 1 only
C                THETA---angle of gravity axis measured from the
C                        positive y axis
C                GRAVY---gravity constant, specified as a multiple
C                        of the gravitational acceleration, g. The
C                        unit weight will be GRAVY*PROPS4.
C
Cif NEDGE.ne.0 then
C--------------------------------
C| NEASS, NOPRS1, NOPRS2, NOPRS3 |
C--------------------------------
C-------------------------------------------------------
C| PRESS11, PRESS12, PRESS21, PRESS22, PRESS31, PRESS33 |
C-------------------------------------------------------
C...,  ...
C     totally NEDGE groups, where
C                NEASS---the element number with which the element
C                        edge is associated, >0 for case 1; <0 for case
C                NOPRS1,NOPRS2,NOPRS3---the nodes, in an anticlockwise
C                        sequence, forming the element face on which the
C                        distributed load acts
C                PRESS11,PRESS21,PRESS31---values of normal component
C                        of distributed load at the nodes
```

```
C                PRESS12,PRESS22,PRESS32---values of tangential
C                        component of distributed load at the nodes
C
C
C**********************************************************************
C
C
C         PROGRAM          MASTER kg.f
C
C
C**********************************************************************
C
C        PROGRAM FOR HYPOELASTIC ANALYSIS OF PLANE STRESS,
C        PLANE STRAIN AND AXISYMMETRIC SOLIDS by YIN and GRAHAM's
C        MODEL
C
C**********************************************************************
C
       IMPLICIT REAL*8 (A-H,O-Z)
C       REAL TIME0,SYSTM,TIMEA(2)
       DIMENSION AK(60000),COORD(500,2),DISP0(1000),DISP1(1000),
     .  DLOAD(1000),DTLMD(7),ENRG0(2000),ENRG1(2000),EPSV(2000),
     .  EPSVI(2000),
     .  ID(1000),IFFIX(1000), IJCOL(18),LNODS(500,9),MATN0(500),
     .  NKOUT(50),NOFIX(500),PERCT(2),POSGP(4),PRESC(500,2),
     .  PROPS(10,14),RLDC1(500,18),RLDC2(500,18),STRS0(4,2000),
     .  STRS1(4,2000),STT(60000),TDISP(1000),TLDC1(1000),
     .  TLDC2(1000),WEIGP(4),WPRSC(500,2)
        OPEN(16,file='kgj.res',STATUS='NEW')
        OPEN(17,file='disp.strs',STATUS='OLD')
        OPEN(18,file='qes',STATUS='NEW')
        OPEN(19,file='esv',STATUS='NEW')
        OPEN(21,file='qev',STATUS='NEW')
        OPEN(22,file='qpes',STATUS='NEW')
        REWIND 16
        REWIND 17
        REWIND 18
        REWIND 19
        REWIND 21
        REWIND 22
        CALL PROMPT('STARTING')
C
```

```
C*** Preset variables associated with dynamic dimensioning
C
        CALL   DIMEN(MELEM,MESTF,MEVAB,MGSTF,MKEYP,MPOIN,MTOTG,
     .               MTOTV,MVFIX,NDOFN,NPROP)
C
C*** Read all input data except data of loads
C
        CALL   INPUT(COORD,DTLMD,IFFIX,LNODS,MATNO,MELEM,MEVAB,
     .               MITER,MKEYP,MMATS,MPOIN,MTOTG,MTOTV,MVFIX,
     .               NALGO,NDOFN,NELEM,NEVAB,NGAUS,NGAU2,NINIT,
     .               NINTV,NKEYP,NKOUT,NMATS,NNODE,NOFIX,NPOIN,
     .               NPROP,NSELE,NSTEP,NSTRE,NSTR1,NTOTG,NTOTV,
     .               NTYPE,NVFIX,POSGP,PRESC,PROPS,WEIGP)
C
C*** Calculate the address of stiffness matrix diagonal
C
        CALL   DLENG(ID,   LNODS,MELEM,MTOTV,NELEM,NNODE,NPOIN,
     .               NTOTV)
C
C*** Chech if the dimensional lenghes used exceeds the maximum
C
        CALL  CHECK3(   ID,MELEM,MESTF,MGSTF,MPOIN,MTOTG,MTOTV,
     .               MVFIX,NELEM,NEVAB,NPOIN,NTOTG,NTOTV)
C
C*** Calculate the nodal load vector {db}
C
        CALL  LOADPS(COORD,LNODS,MATNO,MELEM,MMATS,MPOIN,MTOTV,
     .               NDOFN,NELEM,NEVAB,NGAUS,NINIT,NLOAD,NNODE,
     .               NPOIN,NSTRE,NTOTV,NTYPE,PERCT,POSGP,PROPS,
     .               RLDC1,RLDC2,TLDC1,TLDC2,WEIGP)
C
C*** Initialize the stress and displacement fields
C
        CALL   INITL(ENRG0,ENRG1,EPSV ,EPSVI,LINTV,MELEM,MTOTG,
     .               MTOTV,NELEM,NINIT,NTOTL,NTOTG,NTOTV,PROPS,
     .               STRS0,STRS1,TDISP,TLAMD)
C
C*** Loop over each time interval
C
        DO 1000 INTV1=LINTV,NINTV
C
C*** ZERO DELTA{a0} and form Delta{f}
```

```
C
        CALL  DLTAF(DISP0,DLMDB,DLOAD,DTLMD,LOUTI,MTOTV,MVFIX,
     .              NITER,NOFIX,NTOTV,NVFIX,PERCT,PRESC,TLDC1,
     .              TLDC2,TLAMD,WPRSC)
C
C*** Enter into iterate cycle
C
        DO 800 WHILE ((NITER.LE.MITER).AND.(LOUTI.NE.999))
C
C*** Evaluate the element stiffness matrices
C
        CALL  STIFFP(COORD,ENRG0,ENRG1,EPSV, LNODS,MATNO,MESTF,
     .               MEVAB,MMATS,MPOIN,MTOTV,MELEM,MTOTG,NELEM,
     .               NEVAB,NGAUS,NNODE,NSELE,NSTRE,NSTR1,NTYPE,
     .               NTYSV,POSGP,PROPS,STRS1,STT,  WEIGP)
C
C*** Assemble the global stiffness matrix, [K]
C
        CALL  ASSEMB(AK,      ID,IJCOL,LNODS,MELEM,MEVAB,MTOTV,
     .               MPOIN,NELEM,NEVAB,NNODE,NPOIN,NTOTV,  STT)
C
C*** Fit the displacement boundary conditions
C
        CALL   BOUND(AK,   DISP1,DLOAD,ID,   IFFIX,INTV1,LINTV,
     .               MGSTF,MTOTV,MVFIX,NDOFN,NOFIX,NTOTV,NVFIX,
     .               WPRSC)
C
C*** Solve the equations
C
        CALL  EQSOLV(AK,DISP1,ID,NTOTV,MGSTF,NTYSV)
C
C*** Transform back the displacement if there are skew supports
C
        CALL SKEWBACK(DISP1,MTOTV,MVFIX,NOFIX,NTOTV,NVFIX,WPRSC)
C
C*** Calculate element stresses
C
        CALL  ESTRSS(COORD,DISP1,ENRG0,ENRG1,EPSS, EPSV, EPSVI,
     .               LNODS,LPROP,MELEM,MATNO,MMATS,MPOIN,MTOTG,
     .               MTOTV,NALGO,NDOFN,NELEM,NEVAB,NGAUS,NNODE,
     .               NPOIN,NSELE,NSTR1,NSTRE,NTYPE,POSGP,PROPS,
     .               STRS0,STRS1,TDISP,EPS1)
```

```
C
C*** Convergent control
C
        CALL  CONVER(DISP0,DISP1,DTLMD,ERROR,INTV1,LOUTI,MTOTV,
     .               NINIT,NITER,NTOTV)
  800   CONTINUE
C
C*** Update the results
C
        CALL UPDPSTR(DISP1,ENRG0,ENRG1,MTOTG,MTOTV,NSTR1,NTOTG,
     .               NTOTL,NTOTV,STRS0,STRS1,TDISP)
C
C*** Output the results
C
        CALL  OUTPUT(COORD,DISP1,EPSS, EPSV, ERROR,INTV1,LNODS,
     .               MELEM,MKEYP,MPOIN,MTOTG,MTOTV,MVFIX,NALGO,
     .               NDOFN,NELEM,NGAUS,NINTV,NITER,NKEYP,NKOUT,
     .               NNODE,NPOIN,NSTEP,NTOTV,NTOTG,NTYPE,POSGP,
     .               PROPS,STRS1,TDISP,TLAMD,EPS1)
 1000   CONTINUE
C
C*Reserve the time elapsed and the last displacement and stress field
C    for continuing calculation
C
        WRITE(17,*)TLAMD
        WRITE(17,*)(TDISP(I),I=1,NTOTV)
C        WRITE(17,909)(TDISP(I),I=1,NTOTV)
        WRITE(17,*)((STRS1(I,K),I=1,4),K=1,NTOTG)
        WRITE(17,*)(ENRG1(I),I=1,NTOTG)
        WRITE(17,*) (EPSV(I),I=1,NTOTG)
        WRITE(17,*) (0.0,I=1,NTOTG)
C        WRITE(17,*) (EPSVI(I),I=1,NTOTG)
 909    FORMAT(2X,F18.9,2X,F18.9)
        CLOSE (16)
        CLOSE (17)
        CLOSE (18)
        CLOSE (19)
        CLOSE (21)
        CLOSE (22)
        STOP
        END
C
```

```
      SUBROUTINE EQSOLV(A,B,ID,N,LW,NTYSV)
C**********************************************************************
C
C***This subroutine solves the equation
C
C**********************************************************************
       IMPLICIT REAL*8 (A-H,O-Z)
       DIMENSION A(LW),B(N),ID(N),IWRK(2000)
       DIMENSION AA(500500),X(503500)
       IF (N.GT.1000) THEN
        WRITE(6,*)' Stops in EQSOLV, N=',N
        STOP
       ENDIF
      AA(1)=A(1)
      NA1=1
      NAA1=1
      DO 100 IROW=2,N
      NA2=ID(IROW)
      ICL=IROW-(NA2-NA1)
      IF (ICL.GE.1) THEN
       DO 20 IC=1,ICL
 20    AA(NAA1+IC)=0.0
      ENDIF
      DO 30 IC=ICL+1,IROW
 30   AA(NAA1+IC)=A(NA1+IC-ICL)
      NA1=NA2
      NAA1=NAA1+IROW
 100  CONTINUE
      DO 120 IROW=1,N
 120  X(IROW)=B(IROW)
C
      IF (NTYSV.EQ.1) CALL  LDLT(A,B,ID,N,LW,NTYSV)
      IF (NTYSV.EQ.2) CALL  LEQ2S(AA,N,B,1,N,0,IWRK,X,IERR)
      IF (NTYSV.EQ.3) THEN
       DO 200 IROW=1,N
 200   B(IROW)=X(IROW)
       CALL LEQ2S(AA,N,B,1,N,0,IWRK,X,IERR)
      ENDIF
      CALL PROMPT('EQSOLV')
      RETURN
      END
C
```

```
      SUBROUTINE CNSTTV(ALFA ,DMATX,ENRG0,ENRG1,EPSV, IELEM,KGAUS,
     .                  LPROP,MELEM,MTOTG,NSELE,NTYPE,NTYSV,PROPS,
     .                  STRS1)
C***********************************************************************
C
C***This subroutine evaluates the hypoelastic constitutive matrix
C
C***********************************************************************
      IMPLICIT REAL*8  (A-H,O-Z)
      DIMENSION DMATX(4,4),ENRG0(MTOTG),ENRG1(MTOTG),EPSV(MTOTG),
     .          PROPS(10,14),STRS1(4,MTOTG),PORE(4,4),PDMAX(4,4)
C
      NTYSV=1
      IF (PROPS(LPROP,1).LT.0.001) THEN
       SIGX=STRS1(1,KGAUS)
       SIGY=STRS1(2,KGAUS)
       SIGZ=STRS1(4,KGAUS)
       TAU= STRS1(3,KGAUS)
       PEFFC=-(SIGX+SIGY+SIGZ)/3.0
       RLV=PROPS(LPROP,5)
       VKAP=PROPS(LPROP,6)
       EPSV0= PROPS(LPROP,7)
       A=   PROPS(LPROP,8)
       RN=  PROPS(LPROP,9)
       E=   PROPS(LPROP,10)
       F=   PROPS(LPROP,11)
       S=   PROPS(LPROP,12)
      IF (ABS(NSELE).EQ.3) THEN
       PCONS=PROPS(LPROP,13)
      ELSE
       PCONS=DEXP((EPSV(KGAUS)-EPSV0)/RLV)
      ENDIF
       RJ2=((SIGX-SIGY)**2+(SIGY-SIGZ)**2+(SIGZ-SIGX)**2
     .    +6.0*TAU*TAU)/6.0
       QSHER=DSQRT(3.0*RJ2)
       IF ((ENRG1(KGAUS).GE.ENRG0(KGAUS)).AND.(ABS(QSHER).GT.0.00001))
     .  THEN
C*** if first time loading
        IF (RLV.LE.-999.0) THEN
C**  for Wuhan sand
         R=6.867
         S=39.4
```

```
          RM=0.85
          T=0.0042276
          U0=2.483694
          UULT=3.548134
          RM=0.85
          V=2.81838D-3
          W=0.281838
          PCONS=PROPS(LPROP,13)
          XG3=ABS(STRS1(1,KGAUS))
          XG3RM=XG3**RM
          U=QSHER/XG3RM
          RK=(R+S*PEFFC)**2/R
          RJE=XG3RM*(U-UULT)**2/(U*U-2.0*U*UULT+U0*UULT)/T
          GE=XG3RM*(1.0-W*QSHER/XG3RM)**2/(3.0*V)
          RJ=3.0*RK*RJE/(3.0*RK-RJE)
          RG=3.0*GE*RJ/(3.0*RJ-3.0*GE)
         ELSE
C**  for buffer
          RKE=VKAP*PEFFC
          RK=PEFFC/RLV
          DD=PCONS*(1.0-F*QSHER/PCONS)**2/(3.0*E)
          C=RN*A**(1.0/RN)
          D=(1.0-RN)/RN
C          RJ=RKE*C*(PCONS/QSHER)**D
          RJ=RK*C*(PCONS/QSHER)**D
          RG=DD*RJ*RJ/(RJ*RJ+3.0*DD*RK)
         ENDIF
         AA=(4.0*RG*RK/RJ-RK-4.0*RG/3.0)/(3.0*RG*RK/RJ/RJ-1.0)
         HH=2.0*RJ-3.0*RK
         FF=(3.0*RK*RJ-AA*(RJ+3.0*RK))/HH
         ALFA=(DSQRT(FF*FF+8.0*AA*(3.0*RK*RJ-FF*(RJ+3.0*RK))/HH)
     .         -FF)/(2.0*AA)
         BB=FF/ALFA
         IF ((AA.LT.0.0).OR.(RJ*RJ.LE.3.0*RK*RG)) NTYSV=2
        ELSE
C*** if unloading/reloading
         RK=VKAP*PEFFC
         RG=S*PEFFC
         AA=RK+4.0*RG/3.0
         ALFA=1.0
         BB=RK-2.0*RG/3.0
        ENDIF
```

```
      ELSE
       YG=PROPS(LPROP,1)
       PN=PROPS(LPROP,2)
       ALFA=1.0
       AA=YG*(1.0-PN)/((1.0+PN)*(1.0-2.0*PN))
       BB=YG*PN/((1.0+PN)*(1.0-2.0*PN))
      ENDIF
      DO 10 ISTR1=1,4
      DO 10 JSTR1=1,4
   10 DMATX(ISTR1,JSTR1)=0.0
       IF(NTYPE.NE.1) GO TO 4
C
C***  D MATRIX FOR PLANE STRESS CASE
C
      CONST=(AA-BB)/AA
      DMATX(1,1)=CONST*ALFA*ALFA*(AA+BB)
      DMATX(2,2)=CONST*(AA+BB)
      DMATX(3,3)=CONST*ALFA*AA*0.5
      DMATX(1,2)=CONST*ALFA*BB
      DMATX(2,1)=DMATX(1,2)
      GOTO 888
    4  IF(NTYPE.NE.2) GO TO 6
C
C***  D MATRIX FOR PLANE STRAIN CASE
C
      DMATX(1,1)=ALFA*ALFA*AA
      DMATX(2,2)=AA
      DMATX(3,3)=ALFA*(AA-BB)*0.5
      DMATX(1,2)=ALFA*BB
      DMATX(2,1)=DMATX(1,2)
      GOTO 888
    6  IF(NTYPE.NE.3) GO TO 8
C
C***  D MATRIX FOR AXISYMMETRIC CASE
C
      DMATX(1,1)=ALFA*ALFA*AA
      DMATX(2,2)=AA
      DMATX(3,3)=ALFA*(AA-BB)*0.5
      DMATX(4,4)=ALFA*ALFA*AA
      DMATX(1,2)=ALFA*BB
      DMATX(1,4)=ALFA*ALFA*BB
      DMATX(2,4)=ALFA*BB
```

```
      WRITE(6,900)INTV1,NITER,ERROR,TLAMD
  900 FORMAT(//' INCRMT=',I5,' ITRTNS =',I5,' ERROR=',F14.8,
     .          ' Prop. of ld: ',F14.8)
      IF ((NKEYP.EQ.0).AND.(INTV1.NE.NINTV)) RETURN
     IF (INTV1.EQ.NINTV) THEN
      WRITE(6,910)
  910 FORMAT(//1X,13HDISPLACEMENTS/
     . 1X,4HNODE,8X,'X(r)-DISP.',7X,'Y(z)-DISP.',
     . 12X,'Delt u',7X,'Delt v')
      DO 40 IPOIN=1,NPOIN
      NGASH=IPOIN*2
      NGISH=NGASH-2+1
      IF (NKEYP.LE.0) GO TO 20
      DO 30 IKEYP=1,NKEYP
  30  IF (IPOIN.EQ.NKOUT(IKEYP)) GO TO 20
      GO TO 40
  20  WRITE(6,920) IPOIN,(TDISP(IGASH),IGASH=NGISH,NGASH),
     . (DISP1(IGASH),IGASH=NGISH,NGASH)
  40  CONTINUE
  920 FORMAT(1X,I4,4X,2F15.7,8X,2F15.7/)
C
C*** OUTPUT STRESSES
C
      IF((INTV1.EQ.NINTV).AND.(((ABS(PROPS(1,14)+0.104).LT.0.00001).
     .  OR.((ABS(PROPS(1,13)-2.1)).LT.0.00001)).
     .  OR.((ABS(1.0)).EQ.-1.0))) THEN
       WRITE(6,931)
      ELSEIF(NTYPE.NE.3)  THEN
       WRITE(6,930)
      ELSEIF(NTYPE.EQ.3) THEN
       WRITE(6,940)
      ENDIF
  930 FORMAT(//' ACCUMULATED STRESSES AT GUASSIAN POINTS'
     .  /7X,'x',12X,'y',10X,'SIGMA x',6X,'SIGMA y',6X,'TAU xy',
     .  7X,'SIGMA z')
  931 FORMAT(//' RESULTS of p, q, AND q/p AT GAUSSIAN POINTS'
     .  //13X,'x',13X,'y',13X,'p',13X,'q',13X,'q/p')
  940 FORMAT(' ACCUMULATED STRESSES AT GUASSIAN POINTS'
     .  /7X,'r',12X,'z',10X,'SIGMA r',6X,'SIGMA z',6X,'TAU rz',
     .  7X,'SIGMA t')
     ENDIF
       KGAUS=0
```

```
      DO 60 IELEM=1,NELEM
      IF (INTV1.EQ.NINTV) WRITE(6,950) IELEM
  950 FORMAT(2X,'EL.',I4)
      DO 35 INODE=1,NNODE
      LNODE=LNODS(IELEM,INODE)
      NPOSN=(LNODE-1)*NDOFN
      DO 35 IDOFN=1,NDOFN
      NPOSN=NPOSN+1
   35 ELCOD(IDOFN,INODE)=COORD(LNODE,IDOFN)
      Q=0.0
      P=0.0
      PQ=0.0
      KGASP=0
      DO 50 IGAUS=1,NGAUS
      EXISP=POSGP(IGAUS)
      DO 50 JGAUS=1,NGAUS
      ETASP=POSGP(JGAUS)
      KGAUS=KGAUS+1
      KGASP=KGASP+1
      IF ((INTV1.EQ.NINTV).OR.(NALGO.EQ.KGAUS)) THEN
       CALL SFR2(DERIV,ETASP,EXISP,NNODE,SHAPE)
C
C***  CALCULATE COORDINATES OF SAMPLING POINT
C
       DO 2 IDIME=1,2
       GPCOD(IDIME,KGASP)=0.0
       DO 2 INODE=1,NNODE
       GPCOD(IDIME,KGASP)=GPCOD(IDIME,KGASP)+ELCOD(IDIME,INODE)
     .  *SHAPE(INODE)
    2  CONTINUE
       SIGX=STRS1(1,KGAUS)
       SIGY=STRS1(2,KGAUS)
       SIGZ=STRS1(4,KGAUS)
       TAU= STRS1(3,KGAUS)
       PEFFC=-(SIGX+SIGY+SIGZ)/3.0
       RJ2=((SIGX-SIGY)**2+(SIGY-SIGZ)**2+(SIGZ-SIGX)**2
     .     +6.0*TAU*TAU)/6.0
       QSHER=DSQRT(3.0*RJ2)
       IF (KGAUS.EQ.NALGO) THEN
        RLV=PROPS(1,5)
        EPSV0=PROPS(1,7)
        PCONS=DEXP((EPSV(KGAUS)-EPSV0)/RLV)
```

```
          IF (PROPS(1,7).LE.-999.0) THEN
C** Wuhan sand
           WRITE(18,958) EPSS,QSHER
           WRITE(19,958) EPSS,-EPSV(KGAUS)
          ELSE IF (ABS(PROPS(1,14)-0.087).LT.0.001) THEN
C** T915
           WRITE(18,958) EPS1,QSHER/PEFFC
           WRITE(22,958) EPS1,QSHER/PROPS(1,13)
           WRITE(19,958) PEFFC/PROPS(1,13),QSHER/PROPS(1,13)
           WRITE(21,958) PEFFC,QSHER
          ELSE IF ((ABS(PROPS(1,4)-1.2).LT.0.001).OR.(ABS(PROPS(1,4)
     .     -1.93).LT.0.001).OR.(ABS(PROPS(1,4)-0.73).LT.0.001)) THEN
C** T918, T931, T910
           WRITE(18,958) EPSS,QSHER/PROPS(1,4)
          ELSE
C** Buffer sample
C           WRITE(18,958) EPSS,QSHER/PCONS
C           WRITE(21,958) QSHER/PCONS,EPSV(KGAUS)
           WRITE(18,958) EPSS,QSHER/PROPS(1,13)
           WRITE(21,958) QSHER/PROPS(1,13),EPSV(KGAUS)
           WRITE(19,958) EPSV(KGAUS),EPSS
          ENDIF
         ENDIF
         IF((INTV1.EQ.NINTV).AND.(((ABS(PROPS(1,14)+0.104).LT.0.00001).
     .   OR.((ABS(PROPS(1,13)-2.1)).LT.0.00001)))) THEN
C** Print p',q and q/p'
          IF (NNODE.EQ.4) THEN
           P=P+PEFFC
           Q=Q+QSHER
           PQ=PQ+QSHER/PEFFC
          ELSE
           WRITE(6,961)(GPCOD(I,KGASP),I=1,NDOFN),PEFFC,QSHER,QSHER/PEFFC
          ENDIF
         ELSEIF (INTV1.EQ.NINTV) THEN
C** General case
          WRITE(6,960)(GPCOD(I,KGASP),I=1,NDOFN),(STRS1(I,KGAUS),I=1,4)
         ENDIF
        ENDIF
  50    CONTINUE
        IF ((INTV1.EQ.NINTV).OR.(NALGO.EQ.KGAUS)) THEN
         IF (NNODE.EQ.4) THEN
          P=P/4.0
```

```
          Q=Q/4
          PQ=PQ/4.0
          WRITE(6,961)0.0,0.0,P,Q,PQ
         ENDIF
        ENDIF
  60    CONTINUE
  958   FORMAT(1X,2F14.6)
  959   FORMAT(1X,3F14.6)
  960   FORMAT(1X,6E13.3)
  961   FORMAT(5X,5E14.3)
        CALL PROMPT('  OUTPUT')
        RETURN
        END
C
       SUBROUTINE SKEWBACK(DISP1,MTOTV,MVFIX,NOFIX,NTOTV,NVFIX,WPESC)
C*********************************************************************
C
C*** Transform back the displacement from skew supports, if any.
C
C*********************************************************************
       IMPLICIT REAL*8 (A-H,O-Z)
       DIMENSION DISP1(MTOTV),NOFIX(MVFIX),WPESC(MVFIX,2),T(2,2)
C
C*** Transform back from skew support reference frame
C
       PI=3.14159265359D00
       DO 60 IVFIX=1,NVFIX
       IF (NOFIX(IVFIX).LT.0) THEN
        ANGLE=WPESC(IVFIX,1)*PI/180.0D00
        T(1,1)=COS(ANGLE)
        T(1,2)=-SIN(ANGLE)
        T(2,1)=-T(1,2)
        T(2,2)=T(1,1)
        LTOTV=(-NOFIX(IVFIX)-1)*2+1
        B1=DISP1(LTOTV)
        B2=DISP1(LTOTV+1)
        DISP1(LTOTV)=T(1,1)*B1+T(1,2)*B2
        DISP1(LTOTV+1)=T(2,1)*B1+T(2,2)*B2
       ENDIF
  60   CONTINUE
       RETURN
       END
```

```
C
      SUBROUTINE INITL(ENRG0,ENRG1,EPSV ,EPSVI,LINTV,MELEM,MTOTG,
     .                 MTOTV,NELEM,NINIT,NTOTL,NTOTG,NTOTV,PROPS,
     .                 STRS0,STRS1,TDISP,TLAMD)
C*****************************************************************
C
C*** Initialize the stress matrices and form p{S}/p{sg}
C
C*****************************************************************
       IMPLICIT REAL*8 (A-H,O-Z)
       DIMENSION ENRG0(MTOTG),ENRG1(MTOTG),EPSV(MTOTG),EPSVI(MTOTG),
     .           PROPS(10,14),STRS0(4,MTOTG),STRS1(4,MTOTG),
     .           TDISP(MTOTV)
C
C*** Initialise the stress, displacement and time state
C
      LINTV=1
      NPLOT=1
      NTOTL=0
      TLAMD=0.0
      P0=-ABS(PROPS(1,13))
      EPSV0=PROPS(1,14)
      DO 20 ITOTV=1,NTOTV
  20  TDISP(ITOTV)=0.0
      IF (NINIT.EQ.0) THEN
       DO 30 ITOTG=1,NTOTG
       STRS0(1,ITOTG)=P0
       STRS0(2,ITOTG)=P0
       STRS0(3,ITOTG)=0.0
       STRS0(4,ITOTG)=P0
       ENRG0(ITOTG)=0.0
       EPSVI(ITOTG)=EPSV0
  30   EPSV( ITOTG)=EPSV0
      ELSEIF (NINIT.EQ.1) THEN
       READ(17,*)TLAMD
       READ(17,*)(TDISP(I),I=1,NTOTV)
       READ(17,*)((STRS0(I,K),I=1,4),K=1,NTOTG)
       READ(17,*)(ENRG0(I),I=1,NTOTG)
       READ(17,*) (EPSV(I),I=1,NTOTG)
       READ(17,*) (EPSVI(I),I=1,NTOTG)
       REWIND 17
```

```
      ELSEIF (NINIT.EQ.2) THEN
       READ(17,*)(ANYTHING,I=1,NTOTV+1)
       READ(17,*)((STRS0(I,K),I=1,4),K=1,NTOTG)
       IF(ABS(PROPS(1,1)).LT.0.00001) THEN
        DO 44 ITOTG=1,NTOTG
        ENRG0(ITOTG)=0.0
        SIGX=STRS0(1,ITOTG)
        SIGY=STRS0(2,ITOTG)
        SIGZ=STRS0(4,ITOTG)
        PEFFC=-(SIGX+SIGY+SIGZ)/3.0
        EPSV(ITOTG)=PROPS(1,5)*DLOG(PEFFC)+PROPS(1,7)
  44    EPSVI(ITOTG)=EPSV(ITOTG)
       ENDIF
       REWIND 17
      ELSEIF (NINIT.NE.0) THEN
       WRITE(6,*)' No such initial condition coded NINIT=',NINIT
      ENDIF
      DO 40 ITOTG=1,NTOTG
      ENRG1(ITOTG)=ENRG0(ITOTG)
      DO 40 ISTRE=1,4
  40  STRS1(ISTRE,ITOTG)=STRS0(ISTRE,ITOTG)
      CALL PROMPT('INITL  ')
      RETURN
      END
C
      SUBROUTINE ESTRSS(COORD,DISP1,ENRG0,ENRG1,EPSS, EPSV, EPSVI,
     .                  LNODS,LPROP,MELEM,MATNO,MMATS,MPOIN,MTOTG,
     .                  MTOTV,NALGO,NDOFN,NELEM,NEVAB,NGAUS,NNODE,
     .                  NPOIN,NSELE,NSTR1,NSTRE,NTYPE,POSGP,PROPS,
     .                  STRS0,STRS1,TDISP,EPS1)
C*****************************************************************
C
C****  THIS SUBROUTINE CALCULATES THE STRESSES and the unit energy for
C      each element
C
C*****************************************************************
       IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION CARTD(2,9),COORD(MPOIN,2),DISP1(MTOTV),DMATX(4,4),
     .          ETDIS(2,9),ENRG0(MTOTG),ENRG1(MTOTG),EPSV(MTOTG),
     .          EPSVI(MTOTG),DERIV(2,9),ELCOD(2,9),ELDIS(2,9),
     .          GPCOD(2,9),LNODS(MELEM,9),MATNO(MELEM),POSGP(4),
     .          PROPS(10,14),STRAN(4),STRSS(4),STRS0(4,MTOTG),
```

```
     .          STRS1(4,MTOTG),SHAPE(9),TDISP(MTOTV),TSTRN(4)
        KGAUS=0
        DO 90 IELEM=1,NELEM
        LPROP=MATNO(IELEM)
C
C*** COMPUTE COORDINATE AND DISPLACEMENT OF THE ELEMENT NODAL POINTS
C
        DO 30 INODE=1,NNODE
        LNODE=LNODS(IELEM,INODE)
        NPOSN=(LNODE-1)*NDOFN
        DO 30 IDOFN=1,NDOFN
        NPOSN=NPOSN+1
        ELCOD(IDOFN,INODE)=COORD(LNODE,IDOFN)
        ETDIS(IDOFN,INODE)=TDISP(NPOSN)
  30    ELDIS(IDOFN,INODE)=DISP1(NPOSN)
        KGASP=0
        DO 70 IGAUS=1,NGAUS
        EXISP=POSGP(IGAUS)
        DO 70 JGAUS=1,NGAUS
        ETASP=POSGP(JGAUS)
C       WRITE(6,*)'IGAUS,JGAUS,EXISP,ETASP',IGAUS,JGAUS,EXISP,ETASP
        KGAUS=KGAUS+1
        KGASP=KGASP+1
        CALL SFR2(DERIV,ETASP,EXISP,NNODE,SHAPE)
        CALL JACOB2(CARTD,DERIV,DJACB,ELCOD,GPCOD,IELEM,KGASP,
     .              NNODE,SHAPE)
        CALL CNSTTV(ALFA, DMATX,ENRG0,ENRG1,EPSV, IELEM,KGAUS,
     .              LPROP,MELEM,MTOTG,-NSELE,NTYPE,NTYSV,PROPS,
     .              STRS1)
        CALL STRAIN(CARTD,ELDIS,NDOFN,NNODE,NTYPE,STRAN,KGASP,
     .              GPCOD,SHAPE)
        DO 40 ISTRE=1,NSTRE
        STRSS(ISTRE)=0.0
        DO 40 JSTRE=1,NSTRE
 40     STRSS(ISTRE)=STRSS(ISTRE)+DMATX(ISTRE,JSTRE)*STRAN(JSTRE)
        IF (NTYPE.EQ.1) THEN
         STRSS(NSTR1)=0.0
         STRAN(NSTR1)=-DMATX(1,2)*(ALFA*STRAN(1)+STRAN(2))/DMATX(1,1)
        ELSEIF (NTYPE.EQ.2) THEN
         STRSS(NSTR1)=DMATX(1,2)*(ALFA*STRAN(1)+STRAN(2))
        ENDIF
C
```

```
C*** Update the total stresses and the volume strain
C
        DO 50 ISTR1=1,NSTR1
 50     STRS1(ISTR1,KGAUS)=STRS0(ISTR1,KGAUS)+STRSS(ISTR1)
C
C*** Evaluate the unit energy at each Gaussian point
C
        CALL STRAIN(CARTD,ETDIS,NDOFN,NNODE,NTYPE,TSTRN,KGASP,
     .              GPCOD,SHAPE)
        DO 55 ISTR1=1,NSTR1
 55     TSTRN(ISTR1)=TSTRN(ISTR1)+STRAN(ISTR1)
        ENRG1(KGAUS)=0.0
        DO 65 ISTRE=1,NSTR1
 65     ENRG1(KGAUS)=ENRG1(KGAUS)+STRSS(ISTRE)*STRAN(ISTRE)
       IF (1.EQ.-1) THEN
        WRITE(6,*)'STRSS=',(STRSS(I),I=1,NSTR1)
        WRITE(6,*)'STRAN=',(STRAN(I),I=1,NSTRE)
       ENDIF
        EPSV(KGAUS)=EPSVI(KGAUS)-(TSTRN(1)+TSTRN(2)+TSTRN(4))
C       WRITE(6,*)'EPSV = ',EPSV(KGAUS)
        IF (KGAUS.EQ.NALGO) THEN
         RI2=((TSTRN(1)-TSTRN(2))**2.0+(TSTRN(2)-TSTRN(4))**2.0
     .       +(TSTRN(4)-TSTRN(1))**2.0+1.5*TSTRN(3)**2.0)/6.0
         EPSS=DSQRT(4.0*RI2/3.0)
         EPS1=-TSTRN(2)
C       WRITE(6,*)'EPSV(KGAUS),EPSS=',EPSV(KGAUS),EPSS
C       WRITE(6,*)'TSTRN=',(TSTRN(I),I=1,4)
        ENDIF
 70     CONTINUE
 90     CONTINUE
        CALL PROMPT('  ESTRSS ')
        RETURN
        END
C
      SUBROUTINE PROMPT(STRING)
C*******************************************************
C
C***THIS SUBROUTINE TRACES THE FLOW FOR DEBUGGING
C
C*******************************************************
      CHARACTER*6 STRING
C      WRITE(6,*)STRING,' is passed'
```

```
      RETURN
      END
C
      SUBROUTINE INPUT(COORD,DTLMD,IFFIX,LNODS,MATNO,MELEM,MEVAB,
     .                 MITER,MKEYP,MMATS,MPOIN,MTOTG,MTOTV,MVFIX,
     .                 NALGO,NDOFN,NELEM,NEVAB,NGAUS,NGAU2,NINIT,
     .                 NINTV,NKEYP,NKOUT,NMATS,NNODE,NOFIX,NPOIN,
     .                 NPROP,NSELE,NSTEP,NSTRE,NSTR1,NTOTG,NTOTV,
     .                 NTYPE,NVFIX,POSGP,PRESC,PROPS,WEIGP)
C*****************************************************************
C
C****  THIS SUBROUTINE ACCEPTS MOST OF THE INPUT DATA
C
C*****************************************************************
       IMPLICIT REAL*8 (A-H,O-Z)
       DIMENSION COORD(MPOIN,2),DTLMD(7),IFFIX(MTOTV),NKOUT(MKEYP),
     .       LNODS(MELEM,9),MATNO(MELEM),NOFIX(MVFIX),POSGP(4),
     .       PRESC(MVFIX,NDOFN),PROPS(10,14),WEIGP(4)
       TWOPI=6.28318530718D00
C       CHARACTER TITLE*12
C       READ(5,*)  TITLE
C       WRITE(16,920) TITLE
C  920   FORMAT(18A4)
C
C***  READ THE FIRST DATA CARD, AND ECHO IT IMMEDIATELY
C
       READ(5,*) NPOIN,NVFIX,NELEM,NTYPE,NNODE,NMATS,NGAUS
       NSTRE=3
       IF (NTYPE.EQ.3)NSTRE=4
       NEVAB=NDOFN*NNODE
       NSTR1=NSTRE+1
       IF(NTYPE.EQ.3) NSTR1=NSTRE
       NTOTV=NPOIN*NDOFN
       NGAU2=NGAUS*NGAUS
       NTOTG=NELEM*NGAU2
       WRITE(16,901) NPOIN,NELEM,NVFIX,NTYPE,NNODE,NMATS,NGAUS,NEVAB,
     .               NSTRE
  901  FORMAT(//8H NPOIN =,I4,4X,8H NELEM =,I4,4X,8H NVFIX =,I4,4X,
     .          8H NTYPE =,I4,4X,8H NNODE =,I4,//
     .          8H NMATS =,I4,4X,8H NGAUS =,I4,4X,8H NEVAB =,I4,4X,
     .          8H NSTRE =,I4)
       CALL    CHECK1(NDOFN,NELEM,NGAUS,NMATS,NNODE,NPOIN,
```

```
     .                    NSTRE,NTYPE,NVFIX)
C
C*** Input control data for hypoelasticity analysis
C
        READ(5,*) NINIT,NINTV,NKEYP,NSTEP,MITER,NALGO,NSELE
        IF (NKEYP.GT.0) READ(5,*) (NKOUT(IKEYP),IKEYP=1,NKEYP)
        WRITE(16,990) NINIT,NSTEP,NKEYP,MITER,NALGO,NINTV
  990   FORMAT(/' NINIT =',I4,'     NINTV =',I4,'     NKEYP =',I4,4X,
     .          ' NSTEP =',I4,'     MITER =',I4//
     .          ' NALGO =',I4,'     NSELE =',I4)
        IF (NKEYP.GT.0) WRITE(16,991)(NKOUT(I),I=1,NKEYP)
  991   FORMAT(//' KEY OUTPUT POINTS'/10I7)
C
C***  Read the external load incremental control data
C
        READ(5,*)(DTLMD(I),I=1,7)
        WRITE(16,992)(DTLMD(I),I=1,7)
  992   FORMAT(/' Load incremental control data:'/2X,6F12.7
     .         /' Convergent criterion: ',F12.8)
C
C***  READ THE ELEMENT NODAL CONNECTIONS, AND THE PROPERTY NUMBERS.
C
        WRITE(16,902)
  902   FORMAT(//8H ELEMENT,3X,8HPROPERTY,6X,12HNODE NUMBERS)
        DO 2 IELEM=1,NELEM
        READ(5,*) NUMEL,MATNO(NUMEL),(LNODS(NUMEL,INODE),INODE=1,NNODE)
   2    WRITE(16,903)NUMEL,MATNO(NUMEL),
     .  (LNODS(NUMEL,INODE),INODE=1,NNODE)
  903   FORMAT(1X,I5,I9,6X,9I5)
C
C***  ZERO ALL THE NODAL COORDINATS, PRIOR TO READING SOME OF THEM.
C
        DO 4 IPOIN=1,NPOIN
        DO 4 IDIME=1,2
    4   COORD(IPOIN,IDIME)=0.0
C
C***  READ SOME NODAL COORDINATES, FINISHING WITH THE LAST NODE OF ALL.
C
        WRITE(16,904)
  904   FORMAT(//6H  NODE,5X,4HX(r),6X,4HY(z))
    6   READ(5,*) IPOIN,(COORD(IPOIN,IDIME),IDIME=1,2)
C  905   FORMAT(I5,2F10.5)
```

```
        IF (2.EQ.-2) THEN
         X=COORD(IPOIN,1)
         Y=COORD(IPOIN,2)
         COORD(IPOIN,1)=X*0.70710678-Y*0.70710678
         COORD(IPOIN,2)=X*0.70710678+Y*0.70710678
        ENDIF
        IF(IPOIN.NE.NPOIN) GO TO 6
C
C***  INTERPLATE COORDINATES OF MID-SIDE NODES
C
        IF (3.EQ.-3) CALL NODEXY(COORD,LNODS,MELEM,MPOIN,NELEM,NNODE)
        DO 10 IPOIN=1,NPOIN
   10   WRITE(16,906) IPOIN,(COORD(IPOIN,IDIME),IDIME=1,2)
  906   FORMAT(1X,I5,2F10.3)
C
C***  READ THE FIXED VALUES.
C
        WRITE(16,907)
  907   FORMAT(//5H NODE,6X,4HCODE,9X,12HFIXED VALUES)
        DO 7 IVFIX=1,NTOTV
 7      IFFIX(IVFIX)=0
        DO 8 IVFIX=1,NVFIX
      READ(5,*) NOFIX(IVFIX),IFPRE,(PRESC(IVFIX,IDOFN),IDOFN=1,NDOFN)
      WRITE(16,908)NOFIX(IVFIX),IFPRE,(PRESC(IVFIX,IDOFN),IDOFN=1,NDOFN)
        ISIGN=1
        IF (IFPRE.LT.0) ISIGN=-1
        IFPRE=ISIGN*IFPRE
        NLOCA=(NOFIX(IVFIX)-1)*NDOFN
        NOFIX(IVFIX)=ISIGN*NOFIX(IVFIX)
        IFDOF=10**(NDOFN-1)
        DO 8 IDOFN=1,NDOFN
        NGASH=NLOCA+IDOFN
        IF(IFPRE.LT.IFDOF) GO TO 8
        IFFIX(NGASH)=1
        IFPRE=IFPRE-IFDOF
    8   IFDOF=IFDOF/10
  908   FORMAT(1X,I4,5X,I5,5X,2F10.6)
C
C***  READ THE AVAILABLE SELECTION OF ELEMENT PROPERTIES.
C
   16   WRITE(16,910)
  910   FORMAT(//7H NUMBER,12X,18HELEMENT PROPERTIES)
```

```
        DO 18 IMATS=1,NMATS
        READ(5,*) NUMAT
        READ(5,*) (PROPS(NUMAT,IPROP),IPROP=1,NPROP)
   18   WRITE(16,911) NUMAT,(PROPS(NUMAT,IPROP),IPROP=1,NPROP)
  911   FORMAT(2X,I2,(5X,5E13.5,2(/9X,5E13.5)))
C
C***  SET UP GAUSSIAN INTEGRATION CONSTANTS.
C
   50   CALL    GAUSSQ(NGAUS,POSGP,WEIGP)
        CALL    CHECK2(COORD,IFFIX,LNODS,MATNO,MELEM,MKEYP,MPOIN,
     .                 MTOTV,MVFIX,NDOFN,NINIT,NINTV,NKEYP,NKOUT,
     .                 NELEM,NMATS,NNODE,NOFIX,NPOIN,NVFIX)
        CALL PROMPT('INPUT  ')
        RETURN
        END
C
      SUBROUTINE DLTAF(DISP0,DLMDB,DLOAD,DTLMD,LOUTI,MTOTV,MVFIX,
     .                 NITER,NOFIX,NTOTV,NVFIX,PERCT,PRESC,TLDC1,
     .                 TLDC2,TLAMD,WPRSC)
C*****************************************************************
C
C*** This subroutine prepares the basic data for interation by
C    zeroing Delta{a0} and forming Delta{f}
C
C*****************************************************************
       IMPLICIT REAL*8 (A-H,O-Z)
       DIMENSION DISP0(MTOTV),DLOAD(MTOTV),DTLMD(7),NOFIX(MVFIX),
     .  PERCT(2),PRESC(MVFIX,2),TLDC1(MTOTV),TLDC2(MTOTV),
     .  WPRSC(MVFIX,2)
       LOUTI=0
       NITER=1
       DLMDB=DTLMD(1)
       IF ((TLAMD.GE.DTLMD(4)).AND.(TLAMD.LT.DTLMD(5))) DLMDB=DTLMD(2)
       IF ((TLAMD.GE.DTLMD(5)).AND.(TLAMD.LT.DTLMD(6))) DLMDB=DTLMD(3)
       TLAMD=TLAMD+DLMDB
       DO 50 ITOTV=1,NTOTV
       DISP0(ITOTV)=0.0
 50    DLOAD(ITOTV)=DLMDB*(PERCT(1)*TLDC1(ITOTV)+PERCT(2)*TLDC2(ITOTV))
       DO 60 IFIXD=1,NVFIX
       DO 60 IDOFN=1,2
       IF (NOFIX(IFIXD).LT.0) THEN
        WPRSC(IFIXD,IDOFN)=PRESC(IFIXD,IDOFN)
```

```
        ELSE
         WPRSC(IFIXD,IDOFN)=DLMDB*PRESC(IFIXD,IDOFN)
        ENDIF
 60     CONTINUE
        RETURN
        END
C
      SUBROUTINE DIMEN(MELEM,MESTF,MEVAB,MGSTF,MKEYP,MPOIN,MTOTG,
     .                 MTOTV,MVFIX,NDOFN,NPROP)
C*****************************************************************
C
C****  THIS SUBROUTINE PRESETS VARIBLES ASSOCIATED WITH DYNAMIC
C      DIMENSIONING and the maximum dimensions allowed
C       MELEM---Maximum number of membrane elements
C       MESTF---Maximum lengh of membrane element stiffness matrix
C       MEVAB---Maximum number of membrane element variables
C       MGSTF---Maximum lengh of global stiffness matrix
C       MKEYP---Maximum key output nodes
C       MPOIN---Maximum number of structure nodes
C       MTOTG---Maximum number of structure stresses
C       MTOTV---Maximum number of structure degree of freedom
C       MVFIX---Maximum number of prescribed displacement nodes
C       NDOFN---Degree of freedom per node
C       NPROP---Number of membrane element properties of each group
C
C*****************************************************************
        MELEM=500
        MESTF=60000
        MEVAB=2*9
        MGSTF=60000
        MKEYP=50
        MPOIN=500
        MTOTG=MELEM*4
        MTOTV=1000
        MVFIX=500
        NDOFN=2
        NPROP=14
        CALL PROMPT('  DIMEN')
        RETURN
        END
C
        SUBROUTINE DLENG(ID,   LNODS,MELEM,MTOTV,NELEM,NNODE,NPOIN,
```

```
     .                    NTOTV)
C*********************************************************************
C
C*** CALCULATE THE DIAGONAL ADDRESSES FOR THE SPARS,HALF TRIANGULAR,
C*** VARIABLE BAND WIDTH, GLOBAL MATRIX, K
C
C*********************************************************************
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION ID(MTOTV),LNODS(MELEM,9)
      DO 60 ITOTV=2,NTOTV,2
   60 ID(ITOTV)=NPOIN
      DO 110 IELEM=1,NELEM
      MINIJM=NPOIN
      DO 80 INODE=1,NNODE
      NDCOL=LNODS(IELEM,INODE)
      IF(MINIJM.LE.NDCOL) GO TO 80
      MINIJM=NDCOL
   80 CONTINUE
      DO 90 INODE=1,NNODE
      IAA=LNODS(IELEM,INODE)*2
      IF(ID(IAA).LE.MINIJM) GO TO 90
      ID(IAA)=MINIJM
   90 CONTINUE
  110 CONTINUE
      LS=0
      DO 200 INODE=1,NPOIN
      NF=2*INODE
      NFA=NF-2*ID(NF)+1
      ID(NF-1)=LS+NFA
      LS=ID(NF-1)+NFA+1
  200 ID(NF)=LS
      WRITE(16,500) LS
C      WRITE(16,600) (ID(I),I=1,NTOTV)
  500 FORMAT(//' LENGTH OF THE VARIED BAND (HALF) STIFFNESS MATRIX',
     . I5)
C 600  FORMAT(//5X,'ADRESS OF DIAGONAL '/10I7)
      CALL PROMPT('   DLENG')
      RETURN
      END
C
      SUBROUTINE ASSEMB(AK,   ID,   IJCOL,LNODS,MELEM,MEVAB,MTOTV,
     .                  MPOIN,NELEM,NEVAB,NNODE,NPOIN,NTOTV,STT)
```

```
C*********************************************************************
C
C*** ASSEMBLE THE MATRIX, K, the lower half only
C
C*********************************************************************
        IMPLICIT REAL*8 (A-H,O-Z)
       DIMENSION AK(60000),ID(MTOTV),IJCOL(MEVAB),LNODS(MELEM,9),
     .           STT(60000)
       NLAST=ID(NTOTV)
       DO 50 ITOTV=1,NLAST
 50    AK(ITOTV)=0.0
        NTEVA=NEVAB*NEVAB
        NST4=-NTEVA
        DO 90 IELEM=1,NELEM
        DO 80 INODE=1,NNODE
        IP2=INODE*2
        IJCOL(IP2-1)=LNODS(IELEM,INODE)*2-1
  80    IJCOL(IP2)=LNODS(IELEM,INODE)*2
        IAC4=0
        NST4=NST4+NTEVA
        DO 90 IEVAB=1,NEVAB
        I=IJCOL(IEVAB)
        DO 90 JEVAB=1,NEVAB
        IAC4=IAC4+1
        J=IJCOL(JEVAB)
        IF(I.LT.J) GO TO 90
        IJ=ID(I)-I+J
        AK(IJ)=AK(IJ)+STT(NST4+IAC4)
  90    CONTINUE
        DO 200 I=1,NTOTV
        NDIAG=ID(I)
        IF(AK(NDIAG).LT.1.E-8)
     .   WRITE(16,*)'***** MATRIX DIAGONAL ELEMEN TOO SMALL,
     .   MAYBE SOMETHING WRONG'
 200    CONTINUE
C       WRITE(16,*)'K='
C       DO 202 I=1,NTOTV
C          NF=ID(I-1)+1
C          IF (I.EQ.1) NF=1
C          NDIAG=ID(I)
C          WRITE(16,*)I,(AK(IJ),IJ=NF,NDIAG)
C 202   CONTINUE
```

```
        CALL PROMPT('  ASSEMB')
        RETURN
        END
C
         SUBROUTINE LDLT(A,B,ID,N,LW,NTYSV)
C***********************************************************************
C
C**** USING LDLT METHOD TO SOLVE LINEAR EQUATINE
C
C***********************************************************************
        IMPLICIT REAL*8 (A-H,O-Z)
       DIMENSION A(LW),B(N),ID(N)
C        WRITE(*,'(''****LDLT***'')')
       DO 50 I=2,N
       I0=ID(I)-I
       IFSEL=ID(I-1)-I0+1
       DO 30 J=IFSEL,I
       J0=ID(J)-J
       JFSEL=ID(J-1)-J0+1
       MAXIJ=MAX0(IFSEL,JFSEL)
       IJ=I0+J
       WOREL=0
       J1=J-1
       IF(MAXIJ.GT.J1) GOTO 20
       DO 10 K=MAXIJ,J1
       IK=I0+K
       KK=ID(K)
       JK=J0+K
  10   WOREL=WOREL+A(IK)*A(KK)*A(JK)
       A(IJ)=A(IJ)-WOREL
  20   IF(J.EQ.I) GO TO 40
       JJ=ID(J)
       A(IJ)=A(IJ)/A(JJ)
  30   B(I)=B(I)-A(IJ)*B(J)
  40   II=ID(I)
  50   IF(A(II).LT.1.E-8) GO TO 70
       DO 55 I=1,N
       II=ID(I)
       B(I)=B(I)/A(II)
  55   CONTINUE
       J=N+1
       DO 60 K=2,N
```

```
      J=J-1
      J0=ID(J)-J
      JFSEL=ID(J-1)-J0+1
      J1=J-1
      DO 60 I=JFSEL,J1
      IF(JFSEL.EQ.J) GO TO 60
      JI=J0+I
      B(I)=B(I)-A(JI)*B(J)
   60 CONTINUE
      GO TO 200
   70 WRITE(6,80)
      WRITE(16,80)
   80 FORMAT(4X,'MATRIX [K] IS NOT POSITIVE')
      NTYSV=3
      RETURN
  200 CALL PROMPT('     LDLT')
      RETURN
      END
C
      SUBROUTINE CHECK3(   ID,MELEM,MESTF,MGSTF,MPOIN,MTOTG,MTOTV,
     .                  MVFIX,NELEM,NEVAB,NPOIN,NTOTG,NTOTV)
C*****************************************************************
C
C*** This subroutine checks the dimensions
C
C*****************************************************************
      DIMENSION ID(MTOTV)
      NGSTF=ID(NTOTV)
      NESTF=NELEM*NEVAB*NEVAB
      IF (NELEM.GT.MELEM) WRITE(6,*)' ****ERROR****: NELEM.GT.MELEM'
      IF (NESTF.GT.MESTF) WRITE(6,*)' ****ERROR****: NESTF.GT.MESTF'
      IF (NGSTF.GT.MGSTF) WRITE(6,*)' ****ERROR****: NGSTF.GT.MGSTF',
     .            NGSTF,MGSTF
      IF (NPOIN.GT.MPOIN) WRITE(6,*)' ****ERROR****: NPOIN.GT.MPOIN'
      IF (NTOTG.GT.MTOTG) WRITE(6,*)' ****ERROR****: NTOTG.GT.MTOTG'
      IF (NTOTV.GT.MTOTV) WRITE(6,*)' ****ERROR****: NTOTV.GT.MTOTV'
      CALL PROMPT('CHECK3   ')
      RETURN
      END
C
      SUBROUTINE BOUND(AK,   DISP1,DLOAD,ID,   IFFIX,INTV1,LINTV,
     .                 MGSTF,MTOTV,MVFIX,NDOFN,NOFIX,NTOTV,NVFIX,
```

```
     .                    WPRSC)
C*********************************************************************
C
C***SATISFY THE WPRSCRIBED DISPLACEMENT CONDITIONS
C
C*********************************************************************
        IMPLICIT REAL*8 (A-H,O-Z)
       DIMENSION AK(MGSTF),DISP1(MTOTV),DLOAD(MTOTV),ID(MTOTV),
     .         IFFIX(MTOTV),NOFIX(MVFIX),WPRSC(MVFIX,NDOFN),T(2,2)
       IF (1.EQ.-1) THEN
        WRITE(6,*)'BEFORE BOUND'
        WRITE(6,*)'F='
        WRITE(6,*)(DLOAD(I),I=1,NTOTV)
       ENDIF
       DO 10 ITOTV=1,NTOTV
  10   DISP1(ITOTV)=DLOAD(ITOTV)
C
C*** For skew (oblique) support condition
C
       PI=3.14159265359D00
       DO 60 IVFIX=1,NVFIX
       IF (NOFIX(IVFIX).LT.0) THEN
        ANGLE=WPRSC(IVFIX,1)*PI/180.0D00
        T(1,1)=DCOS(ANGLE)
        T(1,2)=-DSIN(ANGLE)
        T(2,1)=-T(1,2)
        T(2,2)=T(1,1)
C
C** Evaluate [T]t*[K]
C
        NLOCA=(-NOFIX(IVFIX)-1)*NDOFN
        NLENG=ID(NLOCA+2)-ID(NLOCA+1)
        IF (NLOCA.GE.1) THEN
         IDD1=ID(NLOCA)
        ELSE
         IDD1=0
        ENDIF
        IDD2=ID(NLOCA+1)
        IF (NLOCA.EQ.0) IDD1=0
        DO 40 ILENG=1,NLENG
        IF (ILENG.LT.NLENG) THEN
         AKUP=AK(IDD1+ILENG)
```

```
          AKDW=AK(IDD2+ILENG)
          AK(IDD1+ILENG)=T(1,1)*AKUP+T(2,1)*AKDW
          AK(IDD2+ILENG)=T(1,2)*AKUP+T(2,2)*AKDW
         ELSE
          AKUP=AKDW
          AKDW=AK(IDD2+NLENG)
          AKIDD1=T(1,1)*AKUP+T(2,1)*AKDW
          AK(IDD2+NLENG)=T(1,2)*AKUP+T(2,2)*AKDW
         ENDIF
  40     CONTINUE
         IF (1.EQ.-1) THEN
          WRITE(6,*)'TK='
          DO 201 I=1,NTOTV
          IF (I.GT.1) THEN
           NF=ID(I-1)+1
          ELSE
           NF=1
          ENDIF
          NDIAG=ID(I)
          WRITE(6,*)I,(AK(IJ),IJ=NF,NDIAG)
 201      CONTINUE
         ENDIF
C
C** Evaluate [K']=([T]t[K])*[T]
C
         LTOTV=NLOCA+1
         DO 50 ITOTV=LTOTV,NTOTV
         IF (ITOTV.GT.1) THEN
          MLENG=ID(ITOTV)-ID(ITOTV-1)
         ELSE
          MLENG=ID(ITOTV)
         ENDIF
         NLENG=ITOTV-LTOTV+1
         IF (MLENG.LT.NLENG) GOTO 50
         IJ1=ID(ITOTV)-NLENG+1
         AKLF=AK(IJ1)
         AKRT=AK(IJ1+1)
         IF (NLENG.EQ.1) THEN
          AKRT=AKIDD1
          AK(IJ1)=AKLF*T(1,1)+AKRT*T(2,1)
         ELSE
          AK(IJ1)=AKLF*T(1,1)+AKRT*T(2,1)
```

```
          AK(IJ1+1)=AKLF*T(1,2)+AKRT*T(2,2)
         ENDIF
  50     CONTINUE
         IF (1.EQ.-1) THEN
          WRITE(6,*)'TKT='
          DO 202 I=1,NTOTV
          NF=ID(I-1)+1
          IF (I.EQ.1) NF=1
          NDIAG=ID(I)
          WRITE(6,*)I,(AK(IJ),IJ=NF,NDIAG)
 202      CONTINUE
         ENDIF
C
C** Evaluate {b'}=[T]t*{b}
C
         B1=DISP1(LTOTV)
         B2=DISP1(LTOTV+1)
         DISP1(LTOTV)=T(1,1)*B1+T(2,1)*B2
         DISP1(LTOTV+1)=T(1,2)*B1+T(2,2)*B2
        ENDIF
  60    CONTINUE
C
C*** Impose the prescribed displacement conditions
C
       DO 100 IVFIX=1,NVFIX
       NLOCA=(IABS(NOFIX(IVFIX))-1)*NDOFN
       DO 100 IDOFN=1,NDOFN
       NLO1=NLOCA+IDOFN
       PRESU=WPRSC(IVFIX,IDOFN)
       IF (NOFIX(IVFIX).LT.0) PRESU=0.0
       IF (IFFIX(NLO1).LT.1) GOTO 100
C
C** UNIFY THE ROW OF K AND THE RELATED FORCE VECTOR
C
  75   IDD=ID(NLO1)
       AK(IDD)=1.0
       DISP1(NLO1)=PRESU
       M2=IDD-1
       IF (M2.LT.1) GO TO 85
       M1=ID(NLO1-1)+1
       MM=NLO1-(M2-M1)-2
       DO 80 IAK=M1,M2
```

```
       MM=MM+1
       DISP1(MM)=DISP1(MM)-AK(IAK)*PRESU
 80    AK(IAK)=0.0
C
C** UNIFY THE COLLUM OF K AND THE RELATED FORCE VECTOR
C
 85    M3=NLO1+1
       IF (M3.GT.NTOTV) GO TO 100
       DO 90 IAK=M3,NTOTV
       IF ((ID(IAK)-ID(IAK-1)).LE.(IAK-NLO1)) GO TO 90
       M4=ID(IAK)-IAK+NLO1
       DISP1(IAK)=DISP1(IAK)-AK(M4)*PRESU
       AK(M4)=0.0
 90    CONTINUE
 100   CONTINUE
      IF (1.EQ.-1) THEN
       WRITE(6,*)'After bound, K='
       N1=1
       DO 101 IR=1,NTOTV
       N2=ID(IR)
       WRITE(6,*)(AK(I),I=N1,N2)
       N1=N2+1
 101   CONTINUE
       WRITE(6,*)'F=',(DISP1(I),I=1,NTOTV)
      ENDIF
       CALL PROMPT('  BOUND ')
       RETURN
       END
C
      SUBROUTINE CONVER(DISP0,DISP1,DTLMD,ERROR,INTV1,LOUTI,MTOTV,
     .                  NINIT,NITER,NTOTV)
C*****************************************************************
C
C*** This subroutine controls the convergent speed
C
C*****************************************************************
        IMPLICIT REAL*8 (A-H,O-Z)
        DIMENSION DISP0(MTOTV),DISP1(MTOTV),DTLMD(7)
        ERROR=0.0
        CNVER=DTLMD(7)
        DO 50 ITOTV=1,NTOTV
        ERROR=ERROR+(DISP0(ITOTV)-DISP1(ITOTV))**2.0
```

```
        DISP0(ITOTV)=DISP1(ITOTV)
   50   CONTINUE
        ERROR=DSQRT(ERROR)
        CALL PROMPT('  CONVER')
        IF((ERROR.LE.CNVER))THEN
C        IF((ERROR.LE.CNVER).OR.((INTV1.EQ.1).AND.(NINIT.NE.1)))THEN
C          WRITE(6,*)'NINIT=',NINIT,'ERROR,CNVER=',ERROR,CNVER
         LOUTI=999
         RETURN
        ENDIF
        NITER=NITER+1
        RETURN
        END
C
      SUBROUTINE UPDPSTR(DISP1,ENRG0,ENRG1,MTOTG,MTOTV,NSTR1,NTOTG,
     .                   NTOTL,NTOTV,STRS0,STRS1,TDISP)
C*****************************************************************
C
C*** This program resets the displacements, energy and stresses
C    initial state for next increment, and counts the number of
C    iteration times
C
C*****************************************************************
        IMPLICIT REAL*8 (A-H,O-Z)
        DIMENSION DISP1(MTOTV),ENRG0(MTOTG),ENRG1(MTOTG),
     .            STRS0(4,MTOTG),STRS1(4,MTOTG),TDISP(MTOTV)
        NTOTL=NTOTL+NITER
        DO 50 JTOTG=1,NTOTG
        ENRG0(JTOTG)=ENRG1(JTOTG)
        DO 50 ISTR1=1,NSTR1
   50   STRS0(ISTR1,JTOTG)=STRS1(ISTR1,JTOTG)
        DO 60 ITOTV=1,NTOTV
   60   TDISP(ITOTV)=TDISP(ITOTV)+DISP1(ITOTV)
        CALL PROMPT(' UPDPSTR')
        RETURN
        END
C
      SUBROUTINE LOADPS(COORD,LNODS,MATNO,MELEM,MMATS,MPOIN,MTOTV,
     .                  NDOFN,NELEM,NEVAB,NGAUS,NINIT,NLOAD,NNODE,
     .                  NPOIN,NSTRE,NTOTV,NTYPE,PERCT,POSGP,PROPS,
     .                  RLDC1,RLDC2,TLDC1,TLDC2,WEIGP)
C*****************************************************************
```

```
C
C**** THIS SUBROUTINE EVALUATES THE CONSISTANT NODAL FORCES FOR EACH
C     ELEMENT
C
C*********************************************************************
        IMPLICIT REAL*8 (A-H,O-Z)
        DIMENSION CARTD(2,9),COORD(MPOIN,2),DERIV(2,9),DGASH(2),
     .          ELCOD(2,9),LNODS(MELEM,9),MATNO(MELEM),NOPRS(4),
     .          PERCT(2),PGASH(2),POINT(2),POSGP(4),PRESS(4,2),
     .          PROPS(10,14),RLDC1(MELEM,18),RLDC2(MELEM,18),
     .          SHAPE(9),TLDC1(MTOTV),TLDC2(MTOTV),WEIGP(4),GPCOD(2,9)
C        CHARACTER TITLE*12
        TWOPI=6.28318530718D00
        DO 10 IELEM=1,NELEM
        DO 10 IEVAB=1,NEVAB
   10   RLDC1(IELEM,IEVAB)=0.0
        DO 20 ITOTV=1,NTOTV
        TLDC1(ITOTV)=0.0
   20   TLDC2(ITOTV)=0.0
C        READ(5,*) TITLE
C        FORMAT(18A4)
C        WRITE(16,910) TITLE
C  910   FORMAT(2X,18A4)
C
C***  READ DATA CONTROLLING LOADING TYPES TO BE INPUTTED
C
        READ(5,*) NPLOD,IGRAV,NEDGE,NLOAD
        WRITE(16,920) NPLOD,IGRAV,NEDGE,NLOAD
  920   FORMAT(//' NPLOD=',I4,'  IEGRA=',I4,'  NEDGE=',I4,'  NLOAD',I4)
C
C*** Read the load percentages
C
        NLOAD=2
        READ(5,*)(PERCT(I),I=1,NLOAD)
        WRITE(16,930)(PERCT(I),I=1,NLOAD)
  930   FORMAT(//2X,'LOAD PERCENTAGES:',(20(/2X,5F15.5)))
C
C***  READ NODAL POINT LOADS
C
        IF(NPLOD.EQ.0) GO TO 70
        WRITE(16,940)
  940   FORMAT(/' Inputed nodal point loads')
```

```
        DO 60 IPLOD=1,NPLOD
        READ(5,*) LODP,(POINT(IDOFN),IDOFN=1,2)
        WRITE(16,941) LODP,(POINT(IDOFN),IDOFN=1,2)
        LODPT=IABS(LODP)
  941   FORMAT(/I5,2F15.5)
C
C***  ASSOCIATE THE NODAL POINT LOADS WITH AN ELEMENT
C
        DO 30 IELEM=1,NELEM
        DO 30 INODE=1,NNODE
        NLOCA=LNODS(IELEM,INODE)
        MNODE=INODE
   30   IF(LODPT.EQ.NLOCA) GO TO 40
   40   DO 50 IDOFN=1,2
        NGASH=(MNODE-1)*2+IDOFN
        IF (LODP.GT.0) THEN
         RLDC1(IELEM,NGASH)=POINT(IDOFN)
         ELSE
         RLDC2(IELEM,NGASH)=POINT(IDOFN)
        ENDIF
   50   CONTINUE
   60   CONTINUE
   70   CONTINUE
        IF(IGRAV.EQ.0) GO TO 120
C
C*** GRAVITY LOADING SECTION
C
C
C***  READ GRAVITY ANGLE AND GRAVITATIONAL CONSTANT
C
        READ(5,*) THETA,GRAVY
        WRITE(16,950) THETA,GRAVY
  950   FORMAT(2X,16H GRAVITY ANGLE =,F10.3,19H GRAVITY CONSTANT =,
     .    F14.6)
        THETA=THETA/57.2957695477D00
C
C***  LOOP OVER EACH ELEMENT
C
        DO 110 IELEM=1,NELEM
C
C***  SET UP PRELIMINARY CONSTANTS
C
```

```
        LPROP=MATNO(IELEM)
        THICK=1.0
        DENSE=PROPS(LPROP,4)
        IF(DENSE.EQ.0.0) GO TO 110
        GXCOM=DENSE*GRAVY*DSIN(THETA)
        GYCOM=-DENSE*GRAVY*DCOS(THETA)
        IF (DABS(DSIN(THETA)).LT.1.0E-5) GXCOM=0.0
        IF (DABS(DCOS(THETA)).LT.1.0E-5) GYCOM=0.0
C
C***  COMPUTE COORDINATES OF THE ELEMENT NODAL POINTS
C
        DO 80 INODE=1,NNODE
        LNODE=LNODS(IELEM,INODE)
        DO 80 IDIME=1,2
   80   ELCOD(IDIME,INODE)=COORD(LNODE,IDIME)
C
C***  ENTER LOOPS FOR AERA NUMERICAL INTEGRATION
C
        KGASP=0
        DO 100 IGAUS=1,NGAUS
        DO 100 JGAUS=1,NGAUS
        EXISP=POSGP(IGAUS)
        ETASP=POSGP(JGAUS)
C
C***  COMPUTE THE SURFACE FUNCTIONS AT SAMPLING POINTS AND ELEMENTAL
C     VOLUME
C
        CALL        SFR2(DERIV,ETASP,EXISP,NNODE,SHAPE)
        KGASP=KGASP+1
        CALL        JACOB2(CARTD,DERIV,DJACB,ELCOD,GPCOD,IELEM,KGASP,
     .  NNODE,SHAPE)
        DVOLU=DJACB*WEIGP(IGAUS)*WEIGP(JGAUS)
        IF(THICK.NE.0.0) DVOLU=DVOLU*THICK
        IF(NTYPE.EQ.3) DVOLU=DVOLU*TWOPI*GPCOD(1,KGASP)
C
C***  CALCULATE LOADS AND ASSOCIATE WITH ELEMENT NODAL POINTS
C
        DO 90 INODE=1,NNODE
        NGASH=(INODE-1)*2+1
        MGASH=(INODE-1)*2+2
        RLDC1(IELEM,NGASH)=RLDC1(IELEM,NGASH)+GXCOM*SHAPE(INODE)*DVOLU
   90   RLDC1(IELEM,MGASH)=RLDC1(IELEM,MGASH)+GYCOM*SHAPE(INODE)*DVOLU
```

```
  100   CONTINUE
  110   CONTINUE
  120   CONTINUE
        IF(NEDGE.EQ.0) GO TO 220
C
C***  DISTRIBUTED EDGE LOADS SECTION
C
        WRITE(16,960) NEDGE
  960   FORMAT(//1X,21HNO. OF LOADED EDGES =,I5)
       WRITE(16,970)
  970   FORMAT(//1X,38HLIST OF LOADED EDGES AND APPLIED LOADS/)
        NODEG=3
        NCODE=NNODE
        IF(NNODE.EQ.4) NODEG=2
        IF(NNODE.EQ.9) NCODE=8
C
C***  LOOP OVER EACH LOADED EDGE
C
        DO 210 IEDGE=1,NEDGE
C
C***  READ DATA LOCATING THE LOADED EDGE AND APPLIED LOAD
C
        READ(5,*) NEAS,(NOPRS(IODEG),IODEG=1,NODEG)
        NEASS=IABS(NEAS)
        WRITE(16,980) NEAS,(NOPRS(IODEG),IODEG=1,NODEG)
  980   FORMAT(I10,5X,3I5)
        READ(5,*)  ((PRESS(IODEG,IDOFN),IDOFN=1,2),IODEG=1,NODEG)
        WRITE(16,990) ((PRESS(IODEG,IDOFN),IDOFN=1,2),IODEG=1,NODEG)
  990   FORMAT(2X,2F10.3,3X,2F10.3,3X,2F10.3/)
        ETASP=-1.0
C
C***  CALCULATE THE COORDINATES OF THE NODES OF THE ELEMENT EDGE
C
        LPROP=MATNO(NEASS)
C        THICK=PROPS(LPROP,3)
        THICK=1.0
        DO 130 IODEG=1,NODEG
        LNODE=NOPRS(IODEG)
        DO 130 IDIME=1,2
  130   ELCOD(IDIME,IODEG)=COORD(LNODE,IDIME)
C
C***  ENTER LOOP FOR LINEAR NUMERICAL INTEGRATION
```

```
C
        DO 200 IGAUS=1,NGAUS
        EXISP=POSGP(IGAUS)
C
C***  EVALUATE THE SHAPE FUNCTIONS AT THE SAMPLING POINTS
C
        CALL    SFR2(DERIV,ETASP,EXISP,NNODE,SHAPE)
C
C***  CALCULATE COMPONENTS OF THE EQUIVALENT NODAL LOADS
C
        DO 140 IDOFN=1,2
        PGASH(IDOFN)=0.0
        DGASH(IDOFN)=0.0
        DO 140 IODEG=1,NODEG
        PGASH(IDOFN)=PGASH(IDOFN)+PRESS(IODEG,IDOFN)*SHAPE(IODEG)
  140   DGASH(IDOFN)=DGASH(IDOFN)+ELCOD(IDOFN,IODEG)*DERIV(1,IODEG)
        DVOLU=WEIGP(IGAUS)
        IF (THICK.GT.0.0) DVOLU=DVOLU*THICK
        PXCOM=DGASH(1)*PGASH(2)-DGASH(2)*PGASH(1)
        PYCOM=DGASH(1)*PGASH(1)+DGASH(2)*PGASH(2)
        IF(NTYPE.NE.3) GO TO 160
        RADUS=0.0
        DO 150 IODEG=1,NODEG
  150   RADUS=RADUS+SHAPE(IODEG)*ELCOD(1,IODEG)
        DVOLU=DVOLU*TWOPI*RADUS
  160   CONTINUE
C
C***  CALCULATE THE EQUIVALENT NODAL EDGE LOADS WITH AN ELEMENT
C
        DO 170 INODE=1,NNODE
        NLOCA=LNODS(NEASS,INODE)
        MNODE=INODE
  170   IF(NLOCA.EQ.NOPRS(1)) GO TO 180
  180   JNODE=MNODE+NODEG-1
        KOUNT=0
        DO 200 KNODE=MNODE,JNODE
        KOUNT=KOUNT+1
        NGASH=(KNODE-1)*NDOFN+1
        MGASH=(KNODE-1)*NDOFN+2
        IF(KNODE.GT.NCODE) NGASH=1
        IF(KNODE.GT.NCODE) MGASH=2
        IF (NEAS.GT.0) THEN
```

```
          RLDC1(NEASS,NGASH)=RLDC1(NEASS,NGASH)+SHAPE(KOUNT)*PXCOM*DVOLU
          RLDC1(NEASS,MGASH)=RLDC1(NEASS,MGASH)+SHAPE(KOUNT)*PYCOM*DVOLU
         ELSE
          RLDC2(NEASS,NGASH)=RLDC2(NEASS,NGASH)+SHAPE(KOUNT)*PXCOM*DVOLU
          RLDC2(NEASS,MGASH)=RLDC2(NEASS,MGASH)+SHAPE(KOUNT)*PYCOM*DVOLU
         ENDIF
  200    CONTINUE
  210    CONTINUE
  220    CONTINUE
        DO 230 IELEM=1,NELEM
        DO 230 INODE=1,NNODE
        NLOCA=LNODS(IELEM,INODE)
        DO 230 IDOFN=1,NDOFN
        NGASH=(INODE-1)*NDOFN+IDOFN
        ITOTV=(NLOCA-1)*NDOFN+IDOFN
        TLDC1(ITOTV)=TLDC1(ITOTV)+RLDC1(IELEM,NGASH)
        TLDC2(ITOTV)=TLDC2(ITOTV)+RLDC2(IELEM,NGASH)
  230   CONTINUE
        IF (4.EQ.4) THEN
         WRITE(16,1000)
  1000   FORMAT(//' TOTAL NODAL FORCES'//' Load case 1:')
         DO 290 IPOIN=1,NPOIN
         ITOT1=(IPOIN-1)*2+1
         ITOT2=ITOT1+NDOFN-1
  290    WRITE(16,1100) IPOIN, (TLDC1(ITOTV),ITOTV=ITOT1,ITOT2)
         WRITE(16,*)' Load case 2:'
         DO 300 IPOIN=1,NPOIN
         ITOT1=(IPOIN-1)*2+1
         ITOT2=ITOT1+NDOFN-1
  300    WRITE(16,1100) IPOIN, (TLDC2(ITOTV),ITOTV=ITOT1,ITOT2)
 1100    FORMAT(8X,I5,4X,F16.5,2X,F16.5)
        ENDIF
        CALL PROMPT(' LOADPS  ')
        RETURN
        END
C
         SUBROUTINE CHECK1(NDOFN,NELEM,NGAUS,NMATS,NNODE,NPOIN,
     . NSTRE,NTYPE,NVFIX)
C*****************************************************************
C
C****  THIS SUBROUTINE CHECKS THE MAIN CONTROL DATA.
C
```

```
C*********************************************************************
        IMPLICIT REAL*8 (A-H,O-Z)
        DIMENSION NEROR(24)
        DO 100 IEROR=1,12
 100    NEROR(IEROR)=0
C
C***  CREATE THE DIAGNOSTIC MESSAGES
C
        IF(NPOIN.LE.0) NEROR(1)=1
        IF(NVFIX.LT.2.OR.NVFIX.GT.NPOIN) NEROR(3)=1
        IF(NTYPE.LT.1.OR.NTYPE.GT.3) NEROR(5)=1
        IF(NNODE.LT.4.OR.NNODE.GT.9) NEROR(6)=1
        IF(NDOFN.LT.2.OR.NDOFN.GT.5) NEROR(7)=1
        IF(NMATS.LT.1.OR.NMATS.GT.NELEM) NEROR(8)=1
        IF(NGAUS.LT.1.OR.NGAUS.GT.3) NEROR(10)=1
        IF(NSTRE.LT.3.OR.NSTRE.GT.5) NEROR(12)=1
C
C***  EITHER RETURN, OR ELSE PRINT ERRORS DIAGNOSED
C
        KEROR=0
        DO 20 IEROR=1,12
        IF(NEROR(IEROR).EQ.0) GO TO 20
        KEROR=1
        WRITE(6,900) IEROR
  900   FORMAT(//31H *** DIAGNOSIS BY CHECK1, ERROR,I3)
        GO TO (1,2,3,4,5,6,7,8,9,10,11,12), IEROR
 1      WRITE(6,*)'The specified total number of node points, NPOIN,
     . in the structure is less than of equal to zero.'
        GO TO 20
 2      WRITE(6,*)'The possible maximum total number of node points
     . in the structure is less than the specified total, NPOIN.'
        GO TO 20
 3      WRITE(6,*)'The number of restrained nodal points is less than
     . 2 or greater than NPOIN (for plane problems at least a points
     . must be restrained to eliminate rigid body motion).'
        GOTO 20
 4      WRITE(6,*)'The total number of load increments is less than 1.'
        GO TO 20
 5      WRITE(6,*)'The problem type parameter, NTYPE, is not specified
     . as either 1,2 or 3.'
        GO TO 20
 6      WRITE(6,*)'The number of nodes/element is less than 4 or greater
```

```
     . than 9.'
        GO TO 20
 7      WRITE(6,*)'The number of degrees of freedom per node is not
     . equal to 2 or 3.'
        GO TO 20
 8      WRITE(6,*)'The total number of different materials is less than
     . or',' equal to zero or greater than the total number of elements
     . in the structure.'
        GO TO 20
 9      GO TO 20
 10     WRITE(6,*)'The number of Gaussian integration points in each
     . direction is not equal to either 2 or 3.'
        GO TO 20
 11     WRITE(6,*)'The parameter specifying the nonlinear solution
     . algorithm to be employed is outside the permissible range.'
        GO TO 20
 12     WRITE(6,*)'The size of the stress matrix is less than 3 or
     . greater than 5.'
   20   CONTINUE
        IF(KEROR.EQ.0) RETURN
C
C***  OTHERWISE ECHO ALL THE REMAINING DATA WITHOUT FURTHER COMMENT
C
        CALL    ECHO
        CALL PROMPT('  CHECK1')
        RETURN
        END
C
        SUBROUTINE ECHO
C*********************************************************************
C
C****  IF DATA ERRORS HAVE BEEN DETECTED BY SUBROUTINES CHAECK1 OR
C      CHECK2,THIS SUBROUTINE READS AND WRITES THE REMAINING DATA CARDS
C
C*********************************************************************
        IMPLICIT REAL*8 (A-H,O-Z)
C        DIMENSION NTITL(80)
C        CHARACTER NTITL*80
        WRITE(6,900)
  900 FORMAT(//50H NOW FOLLOWS A LISTING OF POST-DISASTER DATA CARDS/)
C        READ(5,*) NTITL
C  905   FORMAT(80A1)
```

```
C        WRITE(6,910) NTITL
C  910   FORMAT(20X/80A1)
       RETURN
       END
C
      SUBROUTINE CHECK2(COORD,IFFIX,LNODS,MATNO,MELEM,MKEYP,MPOIN,
     .                  MTOTV,MVFIX,NDOFN,NINIT,NINTV,NKEYP,NKOUT,
     .                  NELEM,NMATS,NNODE,NOFIX,NPOIN,NVFIX)
C*****************************************************************
C
C****  THIS SUBROUTINE CHECKS THE REMAINDER OF THE INPUT DATA
C
C*****************************************************************
       IMPLICIT REAL*8 (A-H,O-Z)
       DIMENSION COORD(MPOIN,2),IFFIX(MTOTV),LNODS(MELEM,9),
     .           MATNO(MELEM),NEROR(40),NOFIX(MVFIX),NKOUT(MKEYP)
C
C***  CHECK AGAINST TWO IDENTICAL NONZERO NODAL COORDINATES
C
       DO 5 IEROR=13,26
    5  NEROR(IEROR)=0
       DO 40 IPOIN=2,NPOIN
       KPOIN=IPOIN-1
       DO 30 JPOIN=1,KPOIN
       DO 20 IDIME=1,2
       IF(COORD(IPOIN,IDIME).NE.COORD(JPOIN,IDIME)) GO TO 30
   20  CONTINUE
       NEROR(13)=NEROR(13)+1
   30  CONTINUE
   40  CONTINUE
C
C***  CHEK THE LIST OF ELEMENT PROPERTY NUMBERS
C
       DO 50 IELEM=1,NELEM
   50  IF(MATNO(IELEM).LE.0.OR.MATNO(IELEM).GT.NMATS) NEROR(14)=
     .  NEROR(14)+1
C
C***  CHECK FOR IMPOSSIBLE NODE NUMBERS
C
       DO 70 IELEM=1,NELEM
       DO 60 INODE=1,NNODE
       IF(LNODS(IELEM,INODE).EQ.0) NEROR(15)=NEROR(15)+1
```

```
   60   IF(LNODS(IELEM,INODE).LT.0.OR.LNODS(IELEM,INODE).GT.NPOIN)
     .    NEROR(16)=NEROR(16)+1
   70   CONTINUE
C
C***  CHECK FOR ANY REPETITION OF A NODE NUMBER WITHIN AN ELEMENT
C
        DO 140 IPOIN=1,NPOIN
        KSTAR=0
        DO 100 IELEM=1,NELEM
        KZERO=0
        DO 90 INODE=1,NNODE
        IF(LNODS(IELEM,INODE).NE.IPOIN) GO TO 90
        KZERO=KZERO+1
        IF(KZERO.GT.1) NEROR(17)=NEROR(17)+1
C
C***  SEEK FIRST, LAST AND INTERMEDIATES OF NODE IPOIN
C
        KSTAR=IELEM
   90   CONTINUE
  100   CONTINUE
        IF(KSTAR.EQ.0) GO TO 110
        GO TO 140
C
C*** CHECK THAT COORDINATES FOR AN UNUSED NODE HAVE NOT BEEN SPECIFIED
C
  110   WRITE(6,900) IPOIN
  900   FORMAT(/15H CHECK WHY NODE,I4,14H NEVER APPEARS)
        NEROR(18)=NEROR(18)+1
        SIGMA=0.0
        DO 120 IDIME=1,2
  120   SIGMA=SIGMA+DABS(COORD(IPOIN,IDIME))
        IF(SIGMA.NE.0.0) NEROR(19)=NEROR(19)+1
C
C*** CHECK THAT AN UNUSED NODE NUMBER IS NOT A RESTRAINED NODE
C
        DO 130 IVFIX=1,NVFIX
  130   IF(IABS(NOFIX(IVFIX)).EQ.IPOIN) NEROR(20)=NEROR(20)+1
  140   CONTINUE
C
C*** CONTINUE CHECHING THE DATA FOR THE FIXED VALUES
C
        DO 170 IVFIX=1,NVFIX
```

```
        IF (IABS(NOFIX(IVFIX)).GT.NPOIN) NEROR(22)=
     .   NEROR(22)+1
        KOUNT=0
        NLOCA=(IABS(NOFIX(IVFIX))-1)*NDOFN
        DO 160 IDOFN=1,NDOFN
        NLOCA=NLOCA+1
  160   IF(IFFIX(NLOCA).GT.0) KOUNT=1
        IF(KOUNT.EQ.0) NEROR(23)=NEROR(23)+1
        KVFIX=IVFIX-1
        IF (KVFIX.EQ.0) KVFIX=1
        DO 170 JVFIX=1,KVFIX
  170   IF(IVFIX.NE.1.AND.IABS(NOFIX(IVFIX)).EQ.
     .    IABS(NOFIX(JVFIX))) NEROR(24)=
     .   NEROR(24)+1
C
        IF ((NINIT.LT.0).OR.(NINIT.GT.2)) NEROR(25)=NEROR(25)+1
        IF (NKEYP.GT.MKEYP) NEROR(25)=NEROR(25)+1
        IF (NKEYP.LE.0) GO TO 177
        DO 175 IKEYP=1,NKEYP
  175   IF ((NKOUT(IKEYP).LT.1).OR.(NKOUT(IKEYP).GT.NPOIN))
     .  NEROR(26)=NEROR(26)+1
C
  177   KEROR=0
        DO 180 IEROR=13,26
        IF(NEROR(IEROR).EQ.0) GO TO 180
        KEROR=1
        WRITE(6,910) IEROR,NEROR(IEROR)
  910   FORMAT(//31H *** DIAGNOSIS BY CHECK2, ERROR,I3,6X,
     .  18H ASSOCIATED NUMBER,I5)
  180   CONTINUE
        IF(KEROR.NE.0) GO TO 200
C
C*** RETURN ALL NODAL CONNECTION NUMBERS TO POSITIVE VALUES
C
        DO 190 IELEM=1,NELEM
        DO 190 INODE=1,NNODE
  190   LNODS(IELEM,INODE)=IABS(LNODS(IELEM,INODE))
        RETURN
  200   CALL    ECHO
        CALL PROMPT('  CHECK2')
        RETURN
        END
```

```
C
        SUBROUTINE GAUSSQ(NGAUS,POSGP,WEIGP)
C*********************************************************************
C
C*** THIS SUBROUTINE SETS UP THE GAUSS-LEGENDRE INTERATION CONSTAINTS
C
C*********************************************************************
        IMPLICIT REAL*8 (A-H,O-Z)
        DIMENSION POSGP(4),WEIGP(4)
        IF (NGAUS.GT.1) GOTO 1
        POSGP(1)=0.0
        WEIGP(1)=2.0
        RETURN
  1     IF(NGAUS.GT.2) GO TO 4
        POSGP(1)=-DSQRT(1.0D00/3.00D00)
        WEIGP(1)=1.0
        GO TO 6
    4   POSGP(1)=-0.774596669241D00
        POSGP(2)=0.0
        WEIGP(1)=0.555555555555555556D00
        WEIGP(2)=0.888888888888888889D00
    6   KGAUS=NGAUS/2
        DO 8 IGASH=1,KGAUS
        JGASH=NGAUS+1-IGASH
        POSGP(JGASH)=-POSGP(IGASH)
        WEIGP(JGASH)=WEIGP(IGASH)
    8   CONTINUE
        CALL PROMPT(' GAUSSQ ')
        RETURN
        END
C
       SUBROUTINE NODEXY(COORD,LNODS,MELEM,MPOIN,NELEM,NNODE)
C*********************************************************************
C
C*** THIS SUBROUTINE INTERPOLATES THE MIDE SIDE NODES OF STRAIGHT
C    SIDES OF ELEMEMTS AND THE CENTRAL NODE OF 9 NODED ELEMENTS
C
C*********************************************************************
        IMPLICIT REAL*8 (A-H,O-Z)
        DIMENSION COORD(MPOIN,2),LNODS(MELEM,9)
        IF(NNODE.EQ.4) RETURN
C
```

```
C*** LOOP OVER EACH ELEMENT
C
        DO 30 IELEM=1,NELEM
C
C*** LOOP OVER EACH ELEMENT EDGE
C
        NNOD1=9
        IF(NNODE.EQ.8) NNOD1=7
        DO 20 INODE=1,NNOD1,2
        IF(INODE.EQ.9) GO TO 50
C
C*** COMPUTE THE NODE NUMBER OF THE FIRST NODE
C
        NODST=LNODS(IELEM,INODE)
        IGASH=INODE+2
        IF(IGASH.GT.8) IGASH=1
C
C*** COMPUTE THE NODE NUMBER OF THE LAST NODE
C
        NODFN=LNODS(IELEM,IGASH)
        MIDPT=INODE+1
C
C*** COMPUTE THE NODE NUMBER OF THE INTERMEDIATE NODE
C
        NODMD=LNODS(IELEM,MIDPT)
        TOTAL=DABS(COORD(NODMD,1))+DABS(COORD(NODMD,2))
C
C*** IF THE COORDINATES OF THE INTERMIDATE NODE ARE BOTH ZREO
C    INTERPOLATE BY A STRAIGHT LINE
C
        IF(TOTAL.GT.0.0) GO TO 20
        KOUNT=1
   10 COORD(NODMD,KOUNT)=(COORD(NODST,KOUNT)+COORD(NODFN,KOUNT))/2.0
        KOUNT=KOUNT+1
        IF(KOUNT.EQ.2) GO TO 10
   20   CONTINUE
        GO TO 30
   50   LNODE=LNODS(IELEM,INODE)
        TOTAL=DABS(COORD(LNODE,1))+DABS(COORD(LNODE,2))
        IF(TOTAL.GT.0.0) GO TO 30
        LNOD1=LNODS(IELEM,1)
        LNOD3=LNODS(IELEM,3)
```

```
      LNOD5=LNODS(IELEM,5)
      LNOD7=LNODS(IELEM,7)
      KOUNT=1
40    COORD(LNODE,KOUNT)=(COORD(LNOD1,KOUNT)+COORD(LNOD3,KOUNT)
     .   +COORD(LNOD5,KOUNT)+COORD(LNOD7,KOUNT))/4.0
      KOUNT=KOUNT+1
      IF(KOUNT.EQ.2) GO TO 40
30    CONTINUE
      CALL PROMPT('  NODEXY')
      RETURN
      END
C
      SUBROUTINE SFR2(DERIV,ETASP,EXISP,NNODE,SHAPE)
C*********************************************************************
C
C*** THIS SUBROUTINE EVALUATES SHAPE FUNCTIONS AND THEIR DERIVATIVES
C    FOR LINEAR, QUADRATIC LAGRANGIAN AND SERENDIPITY
C    ISOPARAMETRIC 2-D ELEMEMTS
C
C*********************************************************************
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION DERIV(2,9),SHAPE(9)
      S=EXISP
      T=ETASP
      IF(NNODE.GT.4) GO TO 10
      ST=S*T
C
C*** SHAPE FUNCTIONS FOR 4 NODED ELEMENT
C
      SHAPE(1)=(1-T-S+ST)*0.25
      SHAPE(2)=(1-T+S-ST)*0.25
      SHAPE(3)=(1+T+S+ST)*0.25
      SHAPE(4)=(1+T-S-ST)*0.25
C
C*** SHAPE FUNCTION DERIVATIVES
C
      DERIV(1,1)=(-1+T)*0.25
      DERIV(1,2)=(+1-T)*0.25
      DERIV(1,3)=(+1+T)*0.25
      DERIV(1,4)=(-1-T)*0.25
      DERIV(2,1)=(-1+S)*0.25
      DERIV(2,2)=(-1-S)*0.25
```

```
      DERIV(2,3)=(+1+S)*0.25
      DERIV(2,4)=(+1-S)*0.25
      RETURN
   10 IF(NNODE.GT.8) GO TO 30
      S2=S*2.0
      T2=T*2.0
      SS=S*S
      TT=T*T
      ST=S*T
      SST=S*S*T
      STT=S*T*T
      ST2=S*T*2.0
C
C*** SHAPE FUNCTIONS FOR 8 NODED ELEMENT
C
      SHAPE(1)=(-1.0+ST+SS+TT-SST-STT)*0.25
      SHAPE(2)=(1.0-T-SS+SST)*0.5
      SHAPE(3)=(-1.0-ST+SS+TT-SST+STT)*0.25
      SHAPE(4)=(1.0+S-TT-STT)*0.5
      SHAPE(5)=(-1.0+ST+SS+TT+SST+STT)*0.25
      SHAPE(6)=(1.0+T-SS-SST)*0.5
      SHAPE(7)=(-1.0-ST+SS+TT+SST-STT)*0.25
      SHAPE(8)=(1.0-S-TT+STT)*0.5
C
C*** SHAPE FUNCTION DERIVATIVES
C
      DERIV(1,1)=(T+S2-ST2-TT)*0.25
      DERIV(1,2)=(-S+ST)
      DERIV(1,3)=(-T+S2-ST2+TT)*0.25
      DERIV(1,4)=(1.0-TT)*0.5
      DERIV(1,5)=(T+S2+ST2+TT)*0.25
      DERIV(1,6)=(-S-ST)
      DERIV(1,7)=(-T+S2+ST2-TT)*0.25
      DERIV(1,8)=(-1.0+TT)*0.5
      DERIV(2,1)=(S+T2-SS-ST2)*0.25
      DERIV(2,2)=(-1.0+SS)*0.5
      DERIV(2,3)=(-S+T2-SS+ST2)*0.25
      DERIV(2,4)=(-T-ST)
      DERIV(2,5)=(S+T2+SS+ST2)*0.25
      DERIV(2,6)=(1.0-SS)*0.5
      DERIV(2,7)=(-S+T2+SS-ST2)*0.25
      DERIV(2,8)=(-T+ST)
```

```
      RETURN
   30 CONTINUE
      SS=S*S
      ST=S*T
      TT=T*T
      S1=S+1.0
      T1=T+1.0
      S2=S*2.0
      T2=T*2.0
      S9=S-1.0
      T9=T-1.0
C
C*** SHAPE FUNCTIONS FOR 9 NODED ELEMENT
C
      SHAPE(1)=0.25*S9*ST*T9
      SHAPE(2)=0.5*(1.0-SS)*T*T9
      SHAPE(3)=0.25*S1*ST*T9
      SHAPE(4)=0.5*S*S1*(1.0-TT)
      SHAPE(5)=0.25*S1*ST*T1
      SHAPE(6)=0.5*(1.0-SS)*T*T1
      SHAPE(7)=0.25*S9*ST*T1
      SHAPE(8)=0.5*S*S9*(1.0-TT)
      SHAPE(9)=(1.0-SS)*(1.0-TT)
C
C*** SHAPE FUNCTION DERIVATIVES
C
      DERIV(1,1)=0.25*T*T9*(-1.0+S2)
      DERIV(1,2)=-ST*T9
      DERIV(1,3)=0.25*(1.0+S2)*T*T9
      DERIV(1,4)=0.5*(1.0+S2)*(1.0-TT)
      DERIV(1,5)=0.25*(1.0+S2)*T*T1
      DERIV(1,6)=-ST*T1
      DERIV(1,7)=0.25*(-1.0+S2)*T*T1
      DERIV(1,8)=0.5*(-1.0+S2)*(1.0-TT)
      DERIV(1,9)=-S2*(1.0-TT)
      DERIV(2,1)=0.25*(-1.0+T2)*S*S9
      DERIV(2,2)=0.5*(1.0-SS)*(-1.0+T2)
      DERIV(2,3)=0.25*S*S1*(-1.0+T2)
      DERIV(2,4)=-ST*S1
      DERIV(2,5)=0.25*S*S1*(1.0+T2)
      DERIV(2,6)=0.5*(1.0-SS)*(1.0+T2)
      DERIV(2,7)=0.25*S*S9*(1.0+T2)
```

```
      DERIV(2,8)=-ST*S9
      DERIV(2,9)=-T2*(1.0-SS)
   20 CONTINUE
      RETURN
      END
C
      SUBROUTINE JACOB2(CARTD,DERIV,DJACB,ELCOD,GPCOD,IELEM,KGASP,
     . NNODE,SHAPE)
C*****************************************************************
C
C**** THIS SUBROUTINE EVALUATES THE JACOBIAN MATRIX AND THE CARTISIAN
C     SHAPE FUNCTION DERIVATIVES
C
C*****************************************************************
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION CARTD(2,9),DERIV(2,9),ELCOD(2,9),GPCOD(2,9),SHAPE(9),
     .          XJACI(2,2),XJACM(2,2)
C
C***  CALCULATE COORDINATES OF SAMPLING POINT
C
      DO 2 IDIME=1,2
      GPCOD(IDIME,KGASP)=0.0
      DO 2 INODE=1,NNODE
      GPCOD(IDIME,KGASP)=GPCOD(IDIME,KGASP)+ELCOD(IDIME,INODE)
     .  *SHAPE(INODE)
    2 CONTINUE
C
C***  CREATE JACOBIAN MATRIX XJACM
C
      DO 4 IDIME=1,2
      DO 4 JDIME=1,2
      XJACM(IDIME,JDIME)=0.0
      DO 4 INODE=1,NNODE
      XJACM(IDIME,JDIME)=XJACM(IDIME,JDIME)+DERIV(IDIME,INODE)*
     .  ELCOD(JDIME,INODE)
    4 CONTINUE
C
C***  CALCULATE DETERMINANT AND REVERSE OF JACOBIAN MATRIX
C
      DJACB=XJACM(1,1)*XJACM(2,2)-XJACM(1,2)*XJACM(2,1)
      IF(DJACB) 6,6,8
    6 WRITE(6,600) IELEM
```

```
        STOP
    8   CONTINUE
        XJACI(1,1)=XJACM(2,2)/DJACB
        XJACI(2,2)=XJACM(1,1)/DJACB
        XJACI(1,2)=-XJACM(1,2)/DJACB
        XJACI(2,1)=-XJACM(2,1)/DJACB
C
C***  CALCULATE CARTESIAN DERIVATIVES
C
        DO 10 IDIME=1,2
        DO 10 INODE=1,NNODE
        CARTD(IDIME,INODE)=0.0
        DO 10 JDIME=1,2
        CARTD(IDIME,INODE)=CARTD(IDIME,INODE)+XJACI(IDIME,JDIME)*
     .   DERIV(JDIME,INODE)
   10   CONTINUE
  600   FORMAT(//,36H PROGRAM HALTED IN SUBROTINE JACOB2,/,11X,
     .    22H ZERO OR NEGATIVE AERA,/, 10X,16H ELEMENT NUMBER ,I5)
        RETURN
        END
C
        SUBROUTINE BMATPS(BMATX,CARTD,NNODE,SHAPE,GPCOD,NTYPE,KGASP)
C*******************************************************************
C
C***  THIS SUBROUTINE EVALUATES THE STRAIN-DISPLACEMENT MATRIX
C
C*******************************************************************
        IMPLICIT REAL*8 (A-H,O-Z)
        DIMENSION BMATX(4,18),CARTD(2,9),SHAPE(9),GPCOD(2,9)
        NGASH=0
        DO 10 INODE=1,NNODE
        MGASH=NGASH+1
        NGASH=MGASH+1
        BMATX(1,MGASH)=CARTD(1,INODE)
        BMATX(1,NGASH)=0.0
        BMATX(2,MGASH)=0.0
        BMATX(2,NGASH)=CARTD(2,INODE)
        BMATX(3,MGASH)=CARTD(2,INODE)
        BMATX(3,NGASH)=CARTD(1,INODE)
        IF(NTYPE.NE.3) GO TO 10
        BMATX(4,MGASH)=SHAPE(INODE)/GPCOD(1,KGASP)
        BMATX(4,NGASH)=0.0
```

```
   10   CONTINUE
        RETURN
        END
C
        SUBROUTINE DBE(BMATX,DBMAT,DMATX,MEVAB,NEVAB,NSTRE,NSTR1)
C*****************************************************************
C
C****  THIS SUBROUTINE MULTIPLIES THE D-MATRIX BY THE B-MATRIX
C
C*****************************************************************
        IMPLICIT REAL*8 (A-H,O-Z)
        DIMENSION BMATX(4,18),DBMAT(4,18),DMATX(4,4)
        DO 2 ISTRE=1,NSTRE
        DO 2 IEVAB=1,NEVAB
        DBMAT(ISTRE,IEVAB)=0.0
        DO 2 JSTRE=1,NSTRE
        DBMAT(ISTRE,IEVAB)=DBMAT(ISTRE,IEVAB)+
     .  DMATX(ISTRE,JSTRE)*BMATX(JSTRE,IEVAB)
    2   CONTINUE
        RETURN
        END
C
      SUBROUTINE STIFFP(COORD,ENRG0,ENRG1,EPSV, LNODS,MATNO,MESTF,
     .                  MEVAB,MMATS,MPOIN,MTOTV,MELEM,MTOTG,NELEM,
     .                  NEVAB,NGAUS,NNODE,NSELE,NSTRE,NSTR1,NTYPE,
     .                  NTYSV,POSGP,PROPS,STRS1,STT,  WEIGP)
C*****************************************************************
C
C****  THIS SUBROUTINE EVALUATES THE STIFFNESS MATRIX FOR EACH
C      ELEMENT IN TURN
C
C*****************************************************************
        IMPLICIT REAL*8 (A-H,O-Z)
        DIMENSION BMATX(4,18),CARTD(2,9),COORD(MPOIN,2),DBMAT(4,18),
     .            DERIV(2,9),DMATX(4,4),ENRG0(MTOTG),ENRG1(MTOTG),
     .            EPSV(MTOTG),ESTIF(18,18),GPCOD(2,9),ELCOD(2,9),
     .            LNODS(MELEM,9),MATNO(MELEM),POSGP(4),PROPS(10,14),
     .            SHAPE(9),STRS1(4,MTOTG),STT(MESTF),WEIGP(4)
        IF (1.EQ.-1) THEN
         WRITE(6,*)'COORD='
         DO 2 IPOIN=1,22
  2      WRITE(6,*)(COORD(IPOIN,I),I=1,2)
```

```
         WRITE(6,*)'POSGP,WEIGP=',(POSGP(I),I=1,4),(WEIGP(I),I=1,4)
        ENDIF
        TWOPI=6.28318530718D00
        NST4=0
        NTEVA=NEVAB*NEVAB
        KGAUS=0
C
C*** LOOP OVER EACH ELEMET
C
        DO 70 IELEM=1,NELEM
        LPROP=MATNO(IELEM)
C
C*** EVALUATE THE COORDINATES OF THE ELEMENT NODAL POINTS
C
        DO 10 INODE=1,NNODE
        LNODE=LNODS(IELEM,INODE)
        IPOSN=(LNODE-1)*2
        DO 10 IDIME=1,2
        IPOSN=IPOSN+1
   10   ELCOD(IDIME,INODE)=COORD(LNODE,IDIME)
        THICK=1.0
C
C*** INITIALIZE THE ELEMENT STIFFNESS MATRIX
C
        DO 20 IEVAB=1,NEVAB
        DO 20 JEVAB=1,NEVAB
   20   ESTIF(IEVAB,JEVAB)=0.0
        KGASP=0
C
C*** ENTER LOOPS FOR AREA NUMERICAL INTEGRATION
C
        DO 50 IGAUS=1,NGAUS
        EXISP=POSGP(IGAUS)
        DO 50 JGAUS=1,NGAUS
        ETASP=POSGP(JGAUS)
        KGASP=KGASP+1
        KGAUS=KGAUS+1
C
C*** EVALUATE THE SHAPE FUNCTIONS, ELEMENTAL VOLUME, ETC.
C
        CALL    SFR2(DERIV,ETASP,EXISP,NNODE,SHAPE)
        CALL    JACOB2(CARTD,DERIV,DJACB,ELCOD,GPCOD,IELEM,KGASP,
```

```
     .                  NNODE,SHAPE)
        CALL    CNSTTV(ALFA, DMATX,ENRG0,ENRG1,EPSV, IELEM,KGAUS,
     .                 LPROP,MELEM,MTOTG,NSELE,NTYPE,NTYSV,PROPS,
     .                 STRS1)
        DVOLU=DJACB*WEIGP(IGAUS)*WEIGP(JGAUS)
        IF(NTYPE.EQ.3) DVOLU=DVOLU*TWOPI*GPCOD(1,KGASP)
        IF(THICK.NE.0.0) DVOLU=DVOLU*THICK
C
C*** EVALUATE THE B,D AND DB MATRICES
C
        CALL    BMATPS(BMATX,CARTD,NNODE,SHAPE,GPCOD,NTYPE,KGASP)
        CALL    DBE(BMATX,DBMAT,DMATX,MEVAB,NEVAB,NSTRE,NSTR1)
C
C*** CALCULATE THE ELEMENT STIFFNESSES
C
       IF (1.EQ.-1) THEN
        WRITE(6,*)'IELEM,KGAUS=',IELEM,KGAUS
        WRITE(6,*)'DMATX=',((DMATX(I,J),J=1,4),I=1,4)
        WRITE(6,*)'BMATX=',((BMATX(I,J),I=1,NSTRE),J=1,NEVAB)
        WRITE(6,*)'DBMAT=',((DBMAT(I,J),J=1,NEVAB),I=1,4)
       ENDIF
        DO 30 IEVAB=1,NEVAB
        DO 30 JEVAB=IEVAB,NEVAB
        DO 30 ISTRE=1,NSTRE
   30   ESTIF(IEVAB,JEVAB)=ESTIF(IEVAB,JEVAB)+BMATX(ISTRE,IEVAB)*
     .   DBMAT(ISTRE,JEVAB)*DVOLU
   50   CONTINUE
C
C*** CONSTRUCT THE LOWER TRIANGLE OF THE STIFFNESS MATRIX
C
        DO 60 IEVAB=1,NEVAB
        DO 60 JEVAB=IEVAB,NEVAB
   60   ESTIF(JEVAB,IEVAB)=ESTIF(IEVAB,JEVAB)
C
C*** STORE THE STIFFNESS MATRIX, STRESS MATRIX,AND SAMPLING POINT
C    COORDINATES FOR EACH ELEMENT ON DISC FILE
C
        IAC4=0
        DO 65 IEVAB=1,NEVAB
        DO 65 JEVAB=1,NEVAB
        IAC4=IAC4+1
 65     STT(NST4+IAC4)=ESTIF(IEVAB,JEVAB)
```

```
        NST4=NST4+NTEVA
       IF (1.EQ.-1) THEN
        WRITE(6,*)'ELEMET STIFFNESS--',IELEM
        DO 66 I=1,NEVAB
 66     WRITE(6,*)(ESTIF(I,J),J=1,NEVAB)
       ENDIF
   70   CONTINUE
        CALL PROMPT('  STIFFP ')
        RETURN
        END
C
        SUBROUTINE STRAIN(CARTD,ELDIS,NDOFN,NNODE,NTYPE,STRAN,KGASP,
     .                    GPCOD,SHAPE)
C*******************************************************************
C
C****  This subroutine evaluates total incremental strain
C
C*******************************************************************
        IMPLICIT REAL*8 (A-H,O-Z)
        DIMENSION AGASH(2,2),CARTD(2,9),ELDIS(2,9),
     .            STRAN(4),GPCOD(2,9),SHAPE(9)
C       WRITE(6,*)'KGASP,ELDIS=',KGASP,((ELDIS(I,J),I=1,2),J=1,NNODE)
        DO 20 IDOFN=1,NDOFN
        DO 20 JDOFN=1,NDOFN
        BGASH=0.0
        DO 10 INODE=1,NNODE
   10   BGASH=BGASH+CARTD(JDOFN,INODE)*ELDIS(IDOFN,INODE)
   20   AGASH(IDOFN,JDOFN)=BGASH
C
        STRAN(1)=AGASH(1,1)
        STRAN(2)=AGASH(2,2)
        STRAN(3)=AGASH(1,2)+AGASH(2,1)
        STRAN(4)=0.0
        IF (NTYPE.EQ.3) THEN
         DO 30 INODE=1,NNODE
   30    STRAN(4)=STRAN(4)+ELDIS(1,INODE)*SHAPE(INODE)/GPCOD(1,KGASP)
        ENDIF
C        CALL PROMPT('STRAIN')
        RETURN
        END
C
      SUBROUTINE GAUSSJ(A,B,N,M,IPIV,INDXR,INDXC)
```

```
C*****************************************************************
C
C*** GET INVERSE MATRIX, when N=M and [B]=[I], otherwise solve
C    equations by Gauss elimination algorithm
C
C*****************************************************************
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION A(500,500),B(N,1),IPIV(N),INDXR(N),INDXC(N)
      IF (-1.EQ.1) THEN
       WRITE(6,*)'AA='
       DO 140 IROW=1,N
       WRITE(6,*)'IROW=',IROW
 140   WRITE(6,900)(A(IROW,J),J=1,N)
 900   FORMAT(1X,5(1X,F14.5))
       WRITE(6,*)'B='
       WRITE(6,900)(B(I,1),I=1,N)
      ENDIF
      DO 11 J=1,N
   11 IPIV(J)=0
      DO 22 I=1,N
      BIG=0.
       DO 13 J=1,N
        IF(IPIV(J).NE.1) THEN
         DO 12 K=1,N
         IF(IPIV(K).EQ.0) THEN
          IF (DABS(A(J,K)).GE.BIG) THEN
          BIG=DABS(A(J,K))
          IROW=J
          ICOL=K
         ENDIF
        ELSE IF(IPIV(K).GT.1) THEN
        WRITE (6,78)
   78   FORMAT(/,'SINGULAR MATRIX')
        STOP
        ENDIF
   12   CONTINUE
       ENDIF
   13  CONTINUE
       IPIV(ICOL)=IPIV(ICOL)+1
       IF(IROW.NE.ICOL) THEN
       DO 14 L=1,N
       DUM=A(IROW,L)
```

```
      A(IROW,L)=A(ICOL,L)
      A(ICOL,L)=DUM
14    CONTINUE
      DO 15 L=1,M
        DUM=B(IROW,L)
       B(IROW,L)=B(ICOL,L)
       B(ICOL,L)=DUM
15     CONTINUE
       ENDIF
        INDXR(I)=IROW
        INDXC(I)=ICOL
        IF(A(ICOL,ICOL).EQ.0.) THEN
        WRITE(6,78)
        STOP
        ENDIF
        PIVINV=1./A(ICOL,ICOL)
        A(ICOL,ICOL)=1.
        DO 16 L=1,N
        A(ICOL,L)=A(ICOL,L)*PIVINV
16     CONTINUE
        DO 17 L=1,M
        B(ICOL,L)=B(ICOL,L)*PIVINV
17      CONTINUE
        DO 21 LL=1,N
          IF(LL.NE.ICOL) THEN
          DUM=A(LL,ICOL)
           A(LL,ICOL)=0.
            DO 18 L=1,N
            A(LL,L)=A(LL,L)-A(ICOL,L)*DUM
18        CONTINUE
          DO 19 L=1,M
          B(LL,L)=B(LL,L)-B(ICOL,L)*DUM
19        CONTINUE
          ENDIF
21       CONTINUE
22        CONTINUE
           DO 24 LI=1,N
            L=N-LI+1
            IF(INDXR(L).NE.INDXC(L)) THEN
            DO 23 K=1,N
            DUM=A(K,INDXR(L))
            A(K,INDXR(L))=A(K,INDXC(L))
```

```
            A(K,INDXC(L))=DUM
   23       CONTINUE
           ENDIF
   24    CONTINUE
         RETURN
         END
C
      SUBROUTINE SEID(N,X0,X,A,B,EPS,U)
C
C ***********************************************************************
C *** This subroutine computes the solution for a system of N equations
C     in N unknowns using the Gauss Seidel iteration method
C     INPUT:
C        N---Number of equations and unkowns
C        X0[1..N]---Initial solution values
C        A[1..N,1..N]---Matrix coefficients
C        B[1..N]---Right hand side vector
C        EPS---Error bound
C        U[1..N]---work array
C     OUTPUT:
C        X[1..N]---Solution values
C***********************************************************************
C
       DIMENSION A(500,500),B(N),X0(N),X(N),U(N)
C** Initialization
       K=1
       M=1
       DO 1 I=1,N
       U(I)=X0(I)
       B(I)=B(I)/A(I,I)
       DO 5 J=1,N
       IF (I.NE.J) A(I,J)=A(I,J)/A(I,I)
  5    CONTINUE
       A(I,I)=0.0
  1    CONTINUE
       DO 200 WHILE (M.LT.N+1)
C** Compute solution values X
       DO 2 I=1,N
       X(I)=0.0
       DO 3 J=1,N
       X(I)=X(I)-A(I,J)*U(J)
  3    CONTINUE
```

```
      X(I)=X(I)+B(I)
      U(I)=X(I)
  2   CONTINUE
C** Test if solution values X are close
      M=1
      DO 50 WHILE ((ABS(X(M)-X0(M)).LT.EPS).AND.(M.LT.N+1))
  50  M=M+1
C** If not then reset the initial X0 and continue iteration process
      IF (M.LT.N+1) THEN
        DO 4 I=1,N
        X0(I)=X(I)
   4    CONTINUE
C** Otherwise solution values X are good enough to cease iteration
        K=K+1
       ENDIF
  200  CONTINUE
       RETURN
       END
     SUBROUTINE LEQ2S  (A,N,B,M,IB,IJOB,ICHNG,DET,IER)
C ***********************************************************************
C   PURPOSE             - LINEAR EQUATION SOLUTION - INDEFINITE MATRIX
C                           - SYMMETRIC STORAGE MODE - HIGH ACCURACY
C                           SOLUTION
C
C   ARGUMENTS    A      - THE COEFFICIENT MATRIX OF THE EQUATION
C                           AX = B, WHERE A IS ASSUMED TO BE AN N BY N
C                           SYMMETRIC MATRIX. A IS STORED IN SYMMETRIC
C                           STORAGE MODE AND THEREFORE HAS DIMENSION
C                           N*(N+1)/2. (INPUT)
C                N      - ORDER OF A AND THE NUMBER OF ROWS IN B.
C                           (INPUT)
C                B      - INPUT/OUTPUT MATRIX OF DIMENSION N BY M.
C                         ON INPUT, B CONTAINS THE M RIGHT HAND SIDES
C                           OF THE EQUATION AX = B.
C                         ON OUTPUT, THE SOLUTION MATRIX X REPLACES B.
C                           IF IJOB = 1, B IS NOT USED.
C                M      - NUMBER OF RIGHT HAND SIDES (COLUMNS IN B).
C                           (INPUT)
C                IB     - ROW DIMENSION OF MATRIX B EXACTLY AS
C                           SPECIFIED IN THE DIMENSION STATEMENT IN THE
C                           CALLING PROGRAM. (INPUT)
C                IJOB   - INPUT OPTION PARAMETER. IJOB = I IMPLIES:
```

```
C                              I = 0, FACTOR THE MATRIX A AND SOLVE THE
C                                EQUATION AX = B.
C                              I = 1, FACTOR THE MATRIX A. THE FACTORIZED
C                                FORM OF A IS STORED IN THE FIRST
C                                N*(N+1)/2 LOCATIONS OF DET.
C                              I = 2, SOLVE THE EQUATION AX = B. THIS
C                                OPTION IMPLIES THAT MATRIX A HAS ALREADY
C                                BEEN FACTORED BY LEQ2S USING
C                                IJOB = 0 OR 1. IN THIS CASE, THE
C                                INFORMATION CONTAINED IN DET AND ICHNG
C                                MUST HAVE BEEN SAVED FOR REUSE IN THE
C                                CALL TO LEQ2S.
C                  ICHNG  - WORK AREA OF LENGTH 2N.
C                  DET    - WORK AREA OF LENGTH N*(N+1)/2+3N.
C                  IER    - ERROR PARAMETER. (OUTPUT)
C                           TERMINAL ERROR
C                             IER = 129 INDICATES THAT MATRIX A IS
C                               ALGORITHMICALLY SINGULAR. (SEE THE
C                               CHAPTER L PRELUDE)
C                             IER = 130 INDICATES THAT ITERATIVE
C                               IMPROVEMENT FAILED TO CONVERGE. THE
C                               MATRIX IS TOO ILL-CONDITIONED.
C
C   REQD. IMSL ROUTINES - SINGLE/LEQ1S,UERTST,UGETIO
C                       - DOUBLE/LEQ1S,UERTST,UGETIO,VXADD,VXMUL,
C                           VXSTO
C
C-----------------------------------------------------------------------
C
      INTEGER             ICHNG(1)
      DOUBLE PRECISION    A(1),DET(1),B(IB,M),ZERO,XNORM,DXNORM
      DOUBLE PRECISION    ACCXT(2)
      DATA                ZERO/0.D0/,ITMAX/60/
      IER = 0
      NL = (N*(N+1))/2
      NLP1 = NL+1
      NLPN = NL+N
      NLP2N = NLPN+N
      IF (IJOB .EQ. 2) GO TO 10
      DO 5 I = 1,NL
         DET(I) = A(I)
    5 CONTINUE
```

```
      CALL LEQ1S (DET,N,B,M,IB,1,ICHNG,DET(NLP1),IER)
      IF (IER .NE. 0) GO TO 9000
      IF (IJOB .EQ. 1) GO TO 9005
   10 DO 55 J = 1,M
         DO 15 I = 1,N
            DET(NLPN+I) = B(I,J)
   15    CONTINUE
         CALL LEQ1S (DET,N,DET(NLPN+1),1,IB,2,ICHNG,DET(NLP1),IER)
         XNORM = ZERO
         DO 20 I = 1,N
            XNORM = DMAX1(XNORM, DABS(DET(NLPN+I)))
   20    CONTINUE
         IF (XNORM .EQ. ZERO) GO TO 55
         DO 40 ITER = 1,ITMAX
            L = 1
            DO 30 I = 1,N
               K = L
               ACCXT(1) = 0.0D0
               ACCXT(2) = 0.0D0
               CALL VXADD(B(I,J),ACCXT)
               INC = 1
               DO 25 JJ = 1,N
                  CALL VXMUL(-A(K),DET(NLPN+JJ),ACCXT)
                  IF (JJ .GE. I) INC = JJ
                  K = K+INC
   25          CONTINUE
               CALL VXSTO(ACCXT,DET(NLP2N+I))
               L = L+I
   30       CONTINUE
            CALL LEQ1S (DET,N,DET(NLP2N+1),1,IB,2,ICHNG,DET(NLP1),IER)
            DXNORM = ZERO
            DO 35 I = 1,N
               DET(NLPN+I) = DET(NLPN+I)+DET(NLP2N+I)
               DXNORM = DMAX1(DXNORM, DABS(DET(NLP2N+I)))
   35       CONTINUE
            DXNORM = DXNORM+XNORM
            IF (DXNORM .EQ. XNORM) GO TO 45
   40    CONTINUE
         IER = 130
   45    DO 50 I = 1,N
            B(I,J) = DET(NLPN+I)
   50    CONTINUE
```

```
         IF (IER .NE. 0) GO TO 9000
   55 CONTINUE
      GO TO 9005
 9000 CONTINUE
      CALL UERTST(IER,6HLEQ2S )
 9005 RETURN
      END
C
      SUBROUTINE LEQ1S  (A,N,B,M,IB,IJOB,ICHNG,DET,IER)
C*********************************************************************
C
C   PURPOSE             - LINEAR EQUATION SOLUTION - INDEFINITE
C                           MATRIX - SYMMETRIC STORAGE MODE - SPACE
C                           ECONOMIZER SOLUTION
C
C---------------------------------------------------------------------
C
      DIMENSION           A(1),B(IB,1),ICHNG(1),DET(N)
      DOUBLE PRECISION    A,B,SAVE,X0,X1,DET,ALPHA,ZERO,ONE,TEMP,TEMP1,
     *                    RN
      DATA                ZERO/0.D0/,ONE/1.D0/
      DATA                ALPHA/.6403882032022076D0/
      IER = 0
      N0 = N-1
      N1 = N+1
      IF (IJOB.EQ.2) GO TO 125
      RN = 16.0D0*N
      LL = 1
      DO 10 I=1,N
         DET(I) = ZERO
         IC = 1
         L = LL
         DO 5 J=1,N
            TEMP = DABS(A(L))
            IF (TEMP.GT.DET(I)) DET(I) = TEMP
            IF (J.GE.I) IC = J
            L = L+IC
    5    CONTINUE
         DET(I) = DET(I)*RN
         LL = LL+I
   10 CONTINUE
      IDET = 1
```

```
      I = 1
      L = 0
   15 LL = L
      IX = L+I
      IXI = IX+I
      IX1 = IXI+1
      X0 = ZERO
      DO 20 J=I,N
         L = L+J
         X1 = DABS(A(L))
         IF (X1.LE.X0) GO TO 20
         X0 = X1
         K = J
   20 CONTINUE
      X1 = X0
      IR = K
      IS = K
      ICHNG(I) = K
      I1 = I+1
      IF (I.GT.N0) GO TO 75
      J = 1
      LX1 = IXI
      DO 35 L=I1,N
         LX = LX1
         DO 30 KK=1,J
            TEMP = DABS(A(LX))
            IF (TEMP.LE.X0) GO TO 25
            X0 = TEMP
            IR = L
            IS = KK
   25       LX = LX+1
   30    CONTINUE
         LX1 = LX1+L
         J = J+1
   35 CONTINUE
      IF (IS.NE.K) IS = IS+I-1
      INX = LL
      II = I
      KK = K
      ASSIGN 75 TO IBACK
      IF (X1.GE.X0*ALPHA) GO TO 40
      ASSIGN 70 TO IBACK
```

```
      ICHNG(I) = IS
      KK = IS
   40 IF (KK.EQ.II) GO TO IBACK, (75,70,95)
      TEMP = DET(II)
      DET(II) = DET(KK)
      DET(KK) = TEMP
      K1 = KK+1
      JK = (K1*KK)/2
      KX = JK-KK
      IF (K1.GT.N) GO TO 50
      JI = JK+II
      JK = JK+KK
      DO 45 J=K1,N
         TEMP = A(JK)
         A(JK) = A(JI)
         A(JI) = TEMP
         JK = JK+J
         JI = JI+J
   45 CONTINUE
   50 K0 = KK-1
      II1 = II+1
      IF (II1.GT.K0) GO TO 60
      JK = KX+II1
      JI = INX+II+II
      DO 55 J=II1,K0
         TEMP = A(JI)
         A(JI) = A(JK)
         A(JK) = TEMP
         JK = JK+1
         JI = JI+J
   55 CONTINUE
   60 INXJ = INX+II
      KNXJ = KX+KK
      TEMP = A(INXJ)
      A(INXJ) = A(KNXJ)
      A(KNXJ) = TEMP
      IF (II.EQ.1) GO TO IBACK, (75,70,95)
      IM1 = II-1
      DO 65 J=1,IM1
         INXJ = INX+J
         KNXJ = KX+J
         TEMP = A(INXJ)
```

```
      A(INXJ) = A(KNXJ)
      A(KNXJ) = TEMP
65 CONTINUE
   GO TO IBACK, (75,70,95)
70 ICHNG(I1) = IR
   ASSIGN 95 TO IBACK
   KK = IR
   II = I1
   INX = INX+I
   GO TO 40
75 TEMP = DET(I)+A(IX)
   IF (TEMP.EQ.DET(I)) GO TO 200
   IF (I1.GT.N) GO TO 90
   JX = IX
   L = JX+I
   TEMP = ONE/A(IX)
   DO 85 J=I1,N
      INXJ = JX+I
      SAVE = A(INXJ)
      A(INXJ) = SAVE*TEMP
      KX = L
      DO 80 K=I1,J
         INXJ = JX+K
         A(INXJ) = A(INXJ)-SAVE*A(KX)
         KX = KX+K
80    CONTINUE
      JX = JX+J
85 CONTINUE
90 ICHNG(N+I) = 1
   I = I1
   L = IX
   GO TO 120
95 TEMP = A(IX)*A(IX1)
   SAVE = DABS(TEMP)+A(IXI)*A(IXI)
   DET(IDET) = TEMP-A(IXI)*A(IXI)
   IF (SAVE.EQ.ZERO) SAVE = ONE
   SAVE = N*SAVE
   TEMP = SAVE+DABS(DET(IDET))
   IF (TEMP.EQ.SAVE) GO TO 200
   I2 = I+2
   IF (I2.GT.N) GO TO 115
   JX = IX+I1
```

```
      L = JX
      TEMP1 = ONE/DET(IDET)
      IDET = IDET+1
      J0 = I1
      II = L+I
      DO 110 J=I2,N
         JXI = JX+I
         JXI1 = JX+I1
         SAVE = A(JXI)
         TEMP = A(JXI1)
         IF (I2.GT.J0) GO TO 105
         KX = II
         DO 100 K=I2,J0
            JXK = JX+K
            A(JXK) = A(JXK)-A(KX)*SAVE-A(KX+1)*TEMP
            KX = KX+K
100      CONTINUE
105      A(JXI) = (A(IX1)*SAVE-A(IXI)*TEMP)*TEMP1
         A(JXI1) = (A(IX)*TEMP-A(IXI)*SAVE)*TEMP1
         JX = JX+J
         A(JX) = A(JX)-A(JXI)*SAVE-A(JXI1)*TEMP
         J0 = J
110   CONTINUE
115   ICHNG(N+I) = 2
      ICHNG(N+I1) = 0
      L = IX+I1
      I = I2
120   IF (I.LE.N) GO TO 15
125   IF (IJOB.EQ.1) GO TO 9005
      DO 195 JC=1,M
         IF (N .EQ. 1) GO TO 145
         DO 130 I=1,N
            SAVE = B(I,JC)
            J = ICHNG(I)
            B(I,JC) = B(J,JC)
            B(J,JC) = SAVE
130      CONTINUE
         L = 1
         DO 143 I=1,N0
            JX = L+I
            I1 = I+1
            L = L+I1
```

```
            SAVE = B(I,JC)
            IF (SAVE .EQ. ZERO) GO TO 143
            IF (ICHNG(N+I).NE.2) GO TO 135
            JX = JX+I1
            I1 = I1+1
            IF (I1 .GT. N) GO TO 143
135         DO 140 J=I1,N
               B(J,JC) = B(J,JC)-A(JX)*SAVE
               JX = JX+J
140         CONTINUE
143      CONTINUE
145      L = 0
         IDET = 1
         DO 165 I=1,N
            L = L+I
            IF (ICHNG(N+I)-1) 150,155,160
150         B(I,JC) = (B(I,JC)*A(L-I)-SAVE*A(L-1))/DET(IDET)
            IDET = IDET+1
            GO TO 165
155         B(I,JC) = B(I,JC)/A(L)
            GO TO 165
160         SAVE = B(I,JC)
            B(I,JC) = (SAVE*A(L+I+1)-B(I+1,JC)*A(L+I))/DET(IDET)
165      CONTINUE
         NN = NO
         IF (ICHNG(N+N).EQ.0) NN = NN-1
         IF (NN.LT.1) GO TO 185
         NN1 = NN+1
         L = (NN*NN1)/2+NN
         DO 180 K=1,NN
            I = NN1-K
            I1 = I+1
            JX = L
            L = L-I1
            SAVE = B(I,JC)
            IF (ICHNG(N+I).NE.2) GO TO 170
            JX = JX+I1
            I1 = I1+1
170         DO 175 J=I1,N
               SAVE = SAVE-A(JX)*B(J,JC)
               JX = JX+J
175         CONTINUE
```

```
            B(I,JC) = SAVE
  180     CONTINUE
  185     DO 190 K=1,N
            I = N1-K
            SAVE = B(I,JC)
            J = ICHNG(I)
            B(I,JC) = B(J,JC)
            B(J,JC) = SAVE
  190     CONTINUE
  195 CONTINUE
      GO TO 9005
  200 IER = 129
      CALL UERTST (IER,6HLEQ1S )
 9005 RETURN
      END
C
      SUBROUTINE UERTST (IER,NAME)
C-----------------------------------------------------------------------
C
C   PURPOSE             - PRINT A MESSAGE REFLECTING AN ERROR CONDITION
C
C   ARGUMENTS    IER    - ERROR PARAMETER. (INPUT)
C                            IER = I+J WHERE
C                              I = 128 IMPLIES TERMINAL ERROR MESSAGE,
C                              I =  64 IMPLIES WARNING WITH FIX MESSAGE,
C                              I =  32 IMPLIES WARNING MESSAGE.
C                              J = ERROR CODE RELEVANT TO CALLING
C                                  ROUTINE.
C                NAME   - A CHARACTER STRING OF LENGTH SIX PROVIDING
C                            THE NAME OF THE CALLING ROUTINE. (INPUT)
C
C-----------------------------------------------------------------------
C
      INTEGER             IER
      INTEGER             NAME(1)
      INTEGER             I,IEQ,IEQDF,IOUNIT,LEVEL,LEVOLD,NAMEQ(6),
     *                    NAMSET(6),NAMUPK(6),NIN,NMTB
      DATA                NAMSET/1HU,1HE,1HR,1HS,1HE,1HT/
      DATA                NAMEQ/6*1H /
      DATA                LEVEL/4/,IEQDF/0/,IEQ/1H=/
      CALL USPKD (NAME,6,NAMUPK,NMTB)
      CALL UGETIO(1,NIN,IOUNIT)
```

```
      DO 60 I=1,6
   60 NAMEQ(I) = NAMUPK(I)
   65 RETURN
      END
C
      SUBROUTINE UGETIO(IOPT,NIN,NOUT)
C-----------------------------------------------------------------------
C
C   PURPOSE             - TO RETRIEVE CURRENT VALUES AND TO SET NEW
C                           VALUES FOR INPUT AND OUTPUT UNIT
C                           IDENTIFIERS.
C
C   ARGUMENTS    IOPT   - OPTION PARAMETER. (INPUT)
C                           IF IOPT=1, THE CURRENT INPUT AND OUTPUT
C                           UNIT IDENTIFIER VALUES ARE RETURNED IN NIN
C                           AND NOUT, RESPECTIVELY.
C                           IF IOPT=2, THE INTERNAL VALUE OF NIN IS
C                           RESET FOR SUBSEQUENT USE.
C                           IF IOPT=3, THE INTERNAL VALUE OF NOUT IS
C                           RESET FOR SUBSEQUENT USE.
C                NIN    - INPUT UNIT IDENTIFIER.
C                           OUTPUT IF IOPT=1, INPUT IF IOPT=2.
C                NOUT   - OUTPUT UNIT IDENTIFIER.
C                           OUTPUT IF IOPT=1, INPUT IF IOPT=3.
C
C-----------------------------------------------------------------------
C
      INTEGER             IOPT,NIN,NOUT
      INTEGER             NIND,NOUTD
      DATA                NIND/5/,NOUTD/6/
      IF (IOPT.EQ.3) GO TO 10
      IF (IOPT.EQ.2) GO TO 5
      IF (IOPT.NE.1) GO TO 9005
      NIN = NIND
      NOUT = NOUTD
      GO TO 9005
    5 NIND = NIN
      GO TO 9005
   10 NOUTD = NOUT
 9005 RETURN
      END
C
```

```
      SUBROUTINE VXADD(A,ACC)
C-----------------------------------------------------------------------
C
C   PURPOSE             - EXTENDED PRECISION ADD
C
C   ARGUMENTS    A      - DOUBLE PRECISION NUMBER TO BE ADDED TO THE
C                            ACCUMULATOR. (INPUT)
C                ACC    - ACCUMULATOR. (INPUT AND OUTPUT)
C                            ACC IS A DOUBLE PRECISION VECTOR OF LENGTH
C                            2. ON OUTPUT, ACC CONTAINS THE SUM OF
C                            INPUT ACC AND A.
C
C-----------------------------------------------------------------------
C
      DOUBLE PRECISION   A,ACC(2)
      DOUBLE PRECISION   X,Y,Z,ZZ
      X = ACC(1)
      Y = A
      IF (DABS(ACC(1)).GE.DABS(A)) GO TO 1
      X = A
      Y = ACC(1)
    1 Z = X+Y
      ZZ = (X-Z)+Y
      ZZ = ZZ+ACC(2)
      ACC(1) = Z+ZZ
      ACC(2) = (Z-ACC(1))+ZZ
      RETURN
      END
C
      SUBROUTINE VXMUL (A,B,ACC)
C-----------------------------------------------------------------------
C
C   PURPOSE             - EXTENDED PRECISION MULTIPLY
C
C   ARGUMENTS    A      - INPUT DOUBLE PRECISION NUMBER
C                B      - INPUT DOUBLE PRECISION NUMBER
C                ACC    - ACCUMULATOR. (INPUT AND OUTPUT)
C                            ACC IS A DOUBLE PRECISION VECTOR OF LENGTH
C                            2.  ON OUTPUT, ACC CONTAINS THE SUM OF
C                            INPUT ACC AND A*B.
C
C-----------------------------------------------------------------------
```

```
C
      DOUBLE PRECISION   A,B,ACC(2)
      DOUBLE PRECISION   X,HA,TA,HB,TB
      INTEGER            IX(2),I
      LOGICAL*1          LX(8),LI(4)
      EQUIVALENCE        (X,LX(1),IX(1)),(I,LI(1))
      DATA               I/0/
      X = A
      LI(4) = LX(5)
      IX(2) = 0
      I = (I/16)*16
      LX(5) = LI(4)
      HA=X
      TA=A-HA
      X = B
      LI(4) = LX(5)
      IX(2) = 0
      I = (I/16)*16
      LX(5) = LI(4)
      HB = X
      TB = B-HB
      X = TA*TB
      CALL VXADD(X,ACC)
      X = HA*TB
      CALL VXADD(X,ACC)
      X = TA*HB
      CALL VXADD(X,ACC)
      X = HA*HB
      CALL VXADD(X,ACC)
      RETURN
      END
C
      SUBROUTINE VXSTO (ACC,D)
C-----------------------------------------------------------------------
C
C   PURPOSE             - DOUBLE PRECISION STORE.
C
C   ARGUMENTS    ACC    - ACCUMULATOR. (INPUT)
C                            ACC IS A DOUBLE PRECISION VECTOR OF LENGTH
C                            2. ACC IS ASSUMED TO BE THE RESULT OF
C                            CALLING VXADD OR VXMUL TO PERFORM EXTENDED
C                            PRECISION OPERATIONS.
```

```
C                D      - DOUBLE PRECISION SCALAR. (OUTPUT)
C                           ON OUTPUT, D CONTAINS A DOUBLE PRECISION
C                           APPROXIMATION TO THE VALUE OF THE EXTENDED
C                           PRECISION ACCUMULATOR.
C
C-----------------------------------------------------------------------
C
      DOUBLE PRECISION   ACC(2),D
      D = ACC(1)+ACC(2)
      RETURN
      END
C
      SUBROUTINE USPKD  (PACKED,NCHARS,UNPAKD,NCHMTB)
C
C-----------------------------------------------------------------------
C
C   PURPOSE             - NUCLEUS CALLED BY IMSL ROUTINES THAT HAVE
C                           CHARACTER STRING ARGUMENTS
C
C   ARGUMENTS    PACKED - CHARACTER STRING TO BE UNPACKED.(INPUT)
C                NCHARS - LENGTH OF PACKED. (INPUT)  SEE REMARKS.
C                UNPAKD - INTEGER ARRAY TO RECEIVE THE UNPACKED
C                         REPRESENTATION OF THE STRING. (OUTPUT)
C                NCHMTB - NCHARS MINUS TRAILING BLANKS. (OUTPUT)
C
C-----------------------------------------------------------------------
      INTEGER             NC,NCHARS,NCHMTB
      INTEGER*1           UNPAKD(1),PACKED(1),LBYTE,LBLANK
      INTEGER*2           IBYTE,IBLANK
      EQUIVALENCE (LBYTE,IBYTE)
      DATA                LBLANK /1H /
      DATA                IBYTE /1H /
      DATA                IBLANK /1H /
      NCHMTB = 0
      IF(NCHARS.LE.0) RETURN
      NC = MIN0 (129,NCHARS)
      NWORDS = NC*4
      J = 1
      DO 110 I = 1,NWORDS,4
      UNPAKD(I) = PACKED(J)
      UNPAKD(I+1) = LBLANK
      UNPAKD(I+2) = LBLANK
```

```
      UNPAKD(I+3) = LBLANK
  110 J = J+1
      DO 200 N = 1,NWORDS,4
         NN = NWORDS - N - 2
         LBYTE = UNPAKD(NN)
         IF(IBYTE .NE. IBLANK) GO TO 210
  200 CONTINUE
      NN = 0
  210 NCHMTB = (NN + 3) / 4
      RETURN
      END
C
```