ELLIS HORWOOD SERIES IN CIVIL ENGINEERING

Series Editors: Professor R. T. Severn and Dr. R. Sellin, Department of Civil Engineering, University of Bristol

Bhatt, P	**Programming the Matrix Analysis of Skeletal Structures**
Blockley, D.I.,	**The Nature of Structural Design and Safety**
Britto, A.M. & Gunn, M.J.	**Critical State Soil Mechanics via Finite Elements**
Bljuger, F.	**Design of Precast Concrete Structures**
Calladine, C.R.	**Plasticity for Engineers**
Carmichael, D.G.	**Structural Modelling and Optimization**
Carmichael, D.G.	**Engineering Queues in Construction and Mining**
Cyras, A.A.	**Mathematical Models for the Analysis and Optimisation of Elastoplastic Systems**
Dowling, A.P. & Ffowcs-Williams, J.E.	**Sound and Sources of Sound**
Edwards, A.D. & Baker, G.	**Prestressed Concrete**
Farkas, J.	**Optimum Design of Metal Structures**
Graves-Smith, T.R.	**Linear Analysis of Frameworks**
Hendry, A.W., Sinha, B.A. & Davies, S.R.	**Introduction to Load Bearing Brickwork Design**
Heyman, J.	**The Masonry Arch**
Holmes, M. & Martin, L. H.	**Analysis and Design of Structural Connections: Reinforced Concrete and Steel**
Irons, B. & Ahmad, S.	**Techniques of Finite Elements**
Irons, B. & Shrive, N.G.	**Finite Element Primer**
Jordaan, I.J.	**Probability for Engineering Decisions: A Bayesian Approach**
Kwiecinski, M.,	**Plastic Design of Reinforced Slab-beam Structures**
Lencastre, A.	**Handbook of General Hydraulics**
May, J.O.	**Roofs and Roofing**
Megaw, T.M. & Bartlett, J.	**Tunnels: Planning, Design, Construction**
Melchers, R.E.	**Structural Reliability Analysis and Prediction**
Mrazik, A., Skaloud, M. & Tochacek, M.	**Plastic Design of Steel Structures**
Pavlovic, M.	**Thin Plates and Shells: Theory and Applications**
Shaw, E.	**Engineering Hydrology**
Spillers, W.R.	**Introduction to Structures**
Szabo, K. & Kollar, L.	**Structural Design of Cable-suspended Roofs**
White, R.G. & Walker, J.G.	**Noise and Vibration**

CRITICAL STATE SOIL MECHANICS VIA FINITE ELEMENTS

A. M. BRITTO, B.Sc., Ph.D.
Department of Engineering
University of Cambridge

and

M. J. GUNN, M.A., Dip. Comp. Sci.
Department of Civil Engineering
University of Surrey

ELLIS HORWOOD LIMITED
Publishers · Chichester

Halsted Press: a division of
JOHN WILEY & SONS
New York · Chichester · Brisbane · Toronto

First published in 1987 by
ELLIS HORWOOD LIMITED
Market Cross House, Cooper Street,
Chichester, West Sussex, PO19 1EB, England
The publisher's colophon is reproduced from James Gillison's drawing of the ancient Market Cross, Chichester.

Distributors:
Australia and New Zealand:
JACARANDA WILEY LIMITED
GPO Box 859, Brisbane, Queensland 4001, Australia
Canada:
JOHN WILEY & SONS CANADA LIMITED
22 Worcester Road, Rexdale, Ontario, Canada
Europe and Africa:
JOHN WILEY & SONS LIMITED
Baffins Lane, Chichester, West Sussex, England
North and South America and the rest of the world:
Halsted Press: a division of
JOHN WILEY & SONS
605 Third Avenue, New York, NY 10158, USA

British Library Cataloguing in Publication Data
Britto, A.M.
Critical state soil mechanics via finite elements. –
(Ellis Horwood series in civil engineering)
1. Soil mechanics – Data processing
2. Finite element method – Data processing
I. Title II. Gunn, M.J.
624.1'5136'01515353 TA710

Library of Congress Card No. 86-33791

ISBN 0-85312-937-1 (Ellis Horwood Limited)
ISBN 0-470-20816-3 (Halsted Press)

Printed in Great Britain by Unwin Bros. of Woking

Table of Contents

Preface

Engineers have to predict the behaviour of various materials when they are loaded by mechanical forces. Geotechnical engineers are no different to other engineers in this respect: they have to predict the behaviour of soil whereas other engineers deal with steel, concrete, wood, plastics or fluids. In describing the behaviour of materials, engineers use a number of conceptual 'models' which are simplifications of real behaviour. Examples of these models include linear elastic solids, perfectly plastic solids and viscous fluids. If we compare the behaviour of each engineering material with the appropriate conceptual model, then we will always find some differences in detail. However, the important point is that the conceptual model is often sufficiently accurate for the purposes of engineering analysis and design. Associated with each of the examples listed above there is a collection of standard solutions to commonly occurring problems to which the engineer can refer (i.e. the theories of elasticity, plasticity and fluid mechanics).

Soil behaviour conforms less to the models of material behaviour that we have mentioned so far than do most engineering materials. This is because soil is a two-phase material consisting of solid particles and water. Its response to being loaded is inherently more complex than the response of steel or concrete, for example. Another complicating factor arises because the distribution of soil properties in a typical deposit (such as stiffness and strength) is non-uniform. In particular, soil properties always vary with the depth below the ground surface and this will usually have to be taken into account in engineering design.

Terzaghi's effective stress principle was the first conceptual model which successfully accounted for the two-phase nature of soil. We believe that the theories known as 'Critical State Soil Mechanics' represent a similar step forward in describing, understanding and predicting soil behaviour. This book describes the critical state theories and contains an 8000-line FORTRAN computer program written by the authors. This program, known by the acronym CRISP (CRItical State Program), uses the finite element technique and allows predictions to be made of ground deformations using critical state theories. It differs from most finite element programs used in geotechnics in that it is possible to predict the development of deformations with time. When used in this way the program enforces continuity of water flow through the soil as well as equilibrium of total stresses. Since both critical state soil mechanics and the finite element technique have been developed over the last 30 years, we set out below a brief account of the development and characteristic features of each area.

During the 1940s and 1950s, Cambridge University Engineering Department was at the centre of research into the use of the theory of plasticity for the design of steel structures. Part of this research programme involved the full-scale testing of steel portal frames, and the late Professor K.H. Roscoe (who was then a lecturer in soil mechanics in the department) was asked to assist with the design of the foundations. One question which Roscoe was asked to answer was: what would be the angular rotation of a concrete footing embedded in the ground when the portal frame applied an increasing moment to it? It was obvious that none of the existing calculations or theories in soil mechanics could answer this question. The theories that were then available dealt either with the maximum loads which bodies of soil could carry (i.e. ultimate strength theories) or with the prediction of settlements assuming that soil is a linear elastic material. What was needed was a theory which could describe the complete stress–strain behaviour of soil from small strains (when elasticity might be an appropriate description) to larger strains near failure.

Although Roscoe was certainly not the only person to realise the importance of devising an adequate constitutive model for soil, he was unique in the methodical way he devoted the next 17 years to establishing a large research group which had this as a major objective. During this period a number of publications described the progress towards this aim. Roscoe, *et al.* (1958) set out the importance of the concept of the critical void ratio line in describing the behaviour of soils. Roscoe and Schofield (1963) present a complete constitutive model which is successful in reproducing many important aspects of soil behaviour. This model material was given the name 'Cam-clay' by Schofield in 1965 and the book *Critical State Soil Mechanics* (Schofield and Wroth, 1968) elaborated in some detail the behaviour of the model material Cam-clay and compared this with the observed behaviour of real soils.

Schofield and Wroth approach soil mechanics from a completely different direction to most accounts of the subject. They start off with an introduction to some of the fundamental ideas of continuum mechanics and the theories of

elasticity and plasticity. Subsequently these ideas are combined with a small number of assumptions to produce a complete elasto-plastic constitutive model of soil behaviour (i.e. Cam-clay). Critical state soil mechanics includes many ideas developed by others (e.g. Coulomb, Terzaghi, Rendulic, Hvorslev) but its strength is the way that it combines in one theory aspects of soil behaviour previously treated in an unconnected fashion. Critical state soil mechanics is now being taught on an increasing number of undergraduate and postgraduate courses in geotechnical engineering. The major contribution that it currently makes to engineering practice comes from the possibility of interpreting and predicting basic soil properties. For example, from the results of a series of undrained triaxial tests on a particular soil it is possible to predict how the same soil would behave in drained triaxial tests (and vice versa). The critical state soil parameters can then be used to arrive at a rational choice of the traditional soil properties (angle of friction, undrained shear strength) that are used in geotechnical design.

Proceeding along the lines described above, however, is only to use part of the potential of critical state soil mechanics. Simply reinterpreting basic soil properties does not allow (for example) the solution of Roscoe's original problem of the response of the buried footing. To solve problems such as this it is necessary to develop a calculation procedure which keeps track of the stress–strain behaviour of many small elements of soil surrounding the footing, simultaneously ensuring that the strain and stress state of each small element is compatible with and in equilibrium with its neighbours. The finite element method furnishes the basic technique which makes this possible.

The finite element method was introduced during the 1950s as a computer-based technique for the stress analysis of continuous structures. During the 1960s the method was extended to non-structural problems such as heat and fluid flow. The finite element method has grown to be the most popular technique for predicting the behaviour of deformable bodies in civil, mechanical and aeronautical engineering. Its popularity is mostly due to the fact that it is available to engineers as general-purpose computer programs. In principle all the engineer has to do is to describe the geometry of the problem at hand togeth with details of materail properties and the boundary conditions (e.g. externa loads) for the analysis. Thus in geotechnical engineering the same computer program can be used to predict the behaviour of an excavation, foundation or slope. Until the last few years, relatively few engineers have had access to finite element programs because mainframe or minicomputers were required for their operation. Now, however, the continuing fall in the price of computing equipment and the development of more powerful microcomputers will soon put the use of finite element techniques within the scope of the majority of civil engineers.

Although the availability of finite element programs greatly extends the analytical power available to engineers, there are attendant dangers. Usually the engineer using a program has not participated in the programming. This division of engineering activity between program writers and users can lead to mistakes in engineering analysis and perhaps engineering failures. This is because there is

considerable scope for making errors when using a program, either because of a lack of understanding of the underlying principles or because of a simple mistake in preparing the input data for the program. There is also the possibility of a mistake (or 'bug') in the program itself. We believe that the best ways to avoid these possible problems are to improve the education of engineers and to make available to them the source listing of programs.

CRISP was developed over a number of years by research workers in the Cambridge University Engineering Department Soil Mechanics Group, starting in 1975. Since 1977 the authors have been responsible for the development of the program, but it is appropriate that we should acknowledge the early contribution of Mark Zytynski and the later influence and contributions of other members of the group (John Carter, Nimal Seneviratne, Chris Szalwinski and Scott Sloan). Brian Simpson (at Cambridge) and David Naylor (at Swansea) were pioneers in implementing critical state models in finite element programs. Their conclusions have also guided us.

We have revised, rewritten, and omitted many parts of the program for its publication here, and in doing this we have been guided by the following principles:

(a) the program incorporates the critical state description of soil behaviour in a fashion which is as close to the classical presentation of those theories as possible. Thus the reader can check the output of the program with hand calculations such as those presented in Chapter 2 and other texts on critical state soil mechanics. It is possible to think of the program as a testing apparatus in a numerical laboratory where soil structures made of Cam-clay can be tested. (The program also contains elastic descriptions of soil behaviour which might be used: (i) in preliminary analyses; (ii) in conjunction with critical state analyses to assess the importance of non-linearity; (iii) to provide useful results in their own right – for example a consolidation analysis essentially generalises Terzaghi's one-dimensional theory to two dimensions and allows a study of the effects of anisotropic permeabilities);
(b) we have included those features appropriate for geotechnical engineering analysis which are (generally) not present in other published programs;
(c) we have written and documented the program so that it is possible to incorporate new soil models, element types and analysis options.

Our intention has been to produce a book which is self-contained in relation to the basic theories of continuum mechanics, critical state soil mechanics and finite element techniques as they relate to CRISP. The book contains a number of comments and some general advice as to when critical state theories might be expected to give good (or not so good) results. However, we have not included any comparisons of the data of soil tests with the predictions of critical state theories. Nor have we attempted to give a comprehensive account of the range of geotechnical problems to which finite elements can be applied. The application

of advanced analysis techniques such as those described in this book is an area where experience is still being accumulated. We refer readers to journals such as *Géotechnique*.

The authors are grateful to Neil Taylor and Ryan Phillips who read the drafts of several chapters and made many useful comments. Computing facilities were provided by the Universities of Cambridge and Surrey. The typescript of the book was produced using the GCAL text-processing program written by Dr. P. Hazel of the Cambridge University Computing Service, who always provided quick assistance with hardware and software problems.

The authors' work on CRISP was supported by various research contracts, in particular from the Transport and Road Research Laboratory and British Gas. In this connection we would like to thank Myles O'Reilly of the former organisation and Malcolm Howe of the latter. Peter Wroth was responsible for initiating the project and supervised it in the initial stages. Andrew Schofield took over this responsibility, and the authors are particularly grateful to him for his continued encouragement. It was his idea that the program should be made available beyond the environs of Cambridge.

Finally we must thank those who have used the program (either in their academic research or in their profession as engineers). They have discussed their analyses with us, have let us know about the program's shortcomings and have told us what they would like it to do. We have learned a lot from them. Thanks are also due to Robert Mair, Mike Davies, Marcio Almeida, Osamu Kusakabe, Ken Brady, Geoff Leach, Rick Woods, Mark Randolph, Goksel Kutmen, Nobuo Takagi, C.Y. Ah-Teck, David Wood, Sarah Springham, R.K.W. Lung, Trish Hensley, Hans Vaziri, Muni ram Budhu, K.S. Ravindran, Kevin Stone, Guy Houlsby, Shandri Nageswaran, Mr. Kwok, H.L. Goh, Dave Airey, John Mawditt, Ian Pyrah, Robin Andrews, Dickie Bassett, chrysanthi Savvidou, Steve Moore, . . . to name a few.

Arul Britto,
(network address: amb2@uk.ac.cambridge.phoenix)

Mike Gunn.
(network address: mjg1@uk.ac.cambridge.phoenix
or gunn@uk.ac.surrey.syse)

The computer programs described in this book are available on magnetic tape for mini-computers, and on floppy disk for IBM PC-compatible micro-computers. The software may be purchased from:

Ellis Horwood Ltd.,
Market Cross House,
Cooper Street,
Chichester PO19 1EB,
West Sussex.

1

Mechanics

1.1 COMPUTATIONAL MECHANICS

Engineers now routinely use computer programs to predict the behaviour of buildings, bridges, mechanical components and volumes of soil when they are subjected to loads. As argued in the preface, it is important that engineers should understand the fundamental assumptions that are made in these analyses so they may appreciate and interpret the significance of the computer's numerical results. 'Computational mechanics' is the collective name given to the various theories and techniques which are involved. Mechanics is one of the oldest branches of natural science. (Archimedes (287–212 BC), who was concerned with the equilibrium of levers and the buoyancy of submersed objects, is usua regarded as being the first theoretician in the field.) Some scientists anu engineers use the term today to describe the particular subject area of physics which deals with the laws governing the behaviour of 'rigid' bodies. Here, however, mechanics is regarded as encompassing such areas as continuum mechanics, the mechanics of materials, the strength of materials, the mechanics of deformable solids and the theories of elasticity and plasticity as well as the more traditional area covering the equilibrium or motion of rigid bodies.

In all branches of engineering the finite element method is becoming increasingly popular as a method of solving the systems of partial differential equations which describe various physical phenomena. These equations may describe the deformation of solid bodies, the flow of fluids or almost any effect which can be described by the laws of classical physics. The finite element method is advancing on two fronts: firstly it is replacing traditional methods of

analysis and secondly it is opening up new fields for analysis that were previously regarded as intractable. The reasons for the popularity of the method can easily be identified. A typical finite element program provides a general analytical tool which is capable of being applied to a wide range of geometrical configurations involving a spatial variation of material properties. Also the conceptual subdivision of a continuum into finite elements has a strong appeal to most engineers. Of course the advance of finite element analysis is closely connected to the increasing availability of digital computers for engineering analysis.

The traditional equations of continuum mechanics needs some modification when applied to soils. Some of these modifications are straightforward in nature: for example, the sign convention for stresses and strains. For most engineering materials, tensile stresses and strains are taken to be positive. Soil mechanics (and this book) uses the opposite sign convention (i.e. compressive stresses and strains are positive). For the sake of completeness, and to avoid any possible confusion, the next section sets out the basic definitions and equations for an elastic material using this sign convention. Other modifications to the equations of continuum mechanics require rather more thought. The final section of this chapter considers the modifications which are necessary to take account of the two-phase nature of soil. Again the basic soil stress–strain behaviour is taken as elastic. We must emphasise before passing on that this assumption of elasticity is not always adequate. Soil behaviour is markedly non-linear. Chapter 2 explains how this behaviour can be explained within the framework of work-hardening plasticity.

1.2 CONTINUUM MECHANICS

1.2.1 Stresses and equilibrium

Figs. 1.1 to 1.4 show the essential ideas of the equilibrium of bodies and stresses which are assumed in this book.

Fig. 1.1 shows a body of material that is acted on by a number of forces. If the body is in equilibrium then six equations of equilibrium can be written which relate the forces acting on the body to one another. Three of these equations state that the sums of all the forces in three mutually orthogonal directions are zero. The other three equations state that the sums of the moments of the forces about three orthogonal axes are also zero. If the body is not in statical equilibrium then these equations can be replaced by the appropriate forms of Newton's second law of motion.

Fig. 1.2 shows a planar cut across a similar body of material. Since the part of the body on either side of the cut must be in equilibrium there must be internal forces acting in the body (i.e. across the plane) to maintain the state of equilibrium. Using the equations of equilibrium described above, six resultants equivalent to this system of forces (three forces and three couples) can be found. Considering the forces transmitted across a small area δA inscribed on this plane, it is possible to define a measure of the local intensity of the internal force

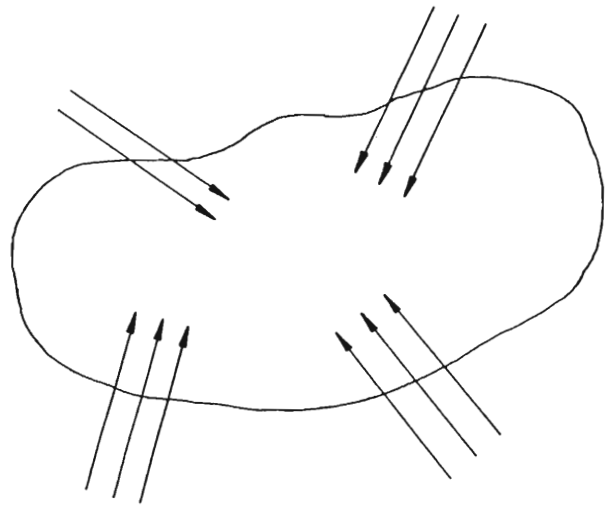


Fig. 1.1 – Forces acting on a body

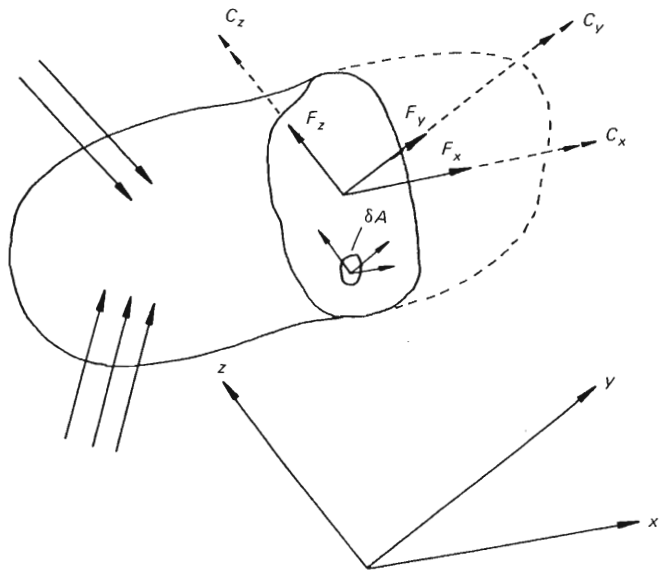


Fig. 1.2 – Internal forces acting in a body

system. These are, of course, the internal stresses acting in the material. Taking the plane to be perpendicular to the x axis, internal stresses are obtained:

$$\sigma_x = \underset{\delta A \to 0}{\text{Limit}} (-\delta F_x/\delta A),$$

$$\tau_{xy} = \underset{\delta A \to 0}{\text{Limit}} (-\delta F_y/\delta A),$$

$$\tau_{xz} = \underset{\delta A \to 0}{\text{Limit}} (-\delta F_z/\delta A).$$

The reader should note that while six force resultants were necessary to describe the interaction of the two parts of the body, only three stresses are needed to describe the local intensity of forces at one particular point on the surface. This is because the force distribution is considered to be essentially continuous, and as the small area δA shrinks in size the force distribution over the area approaches a constant value. The couples arise from integrating the stresses over the cutting plane.

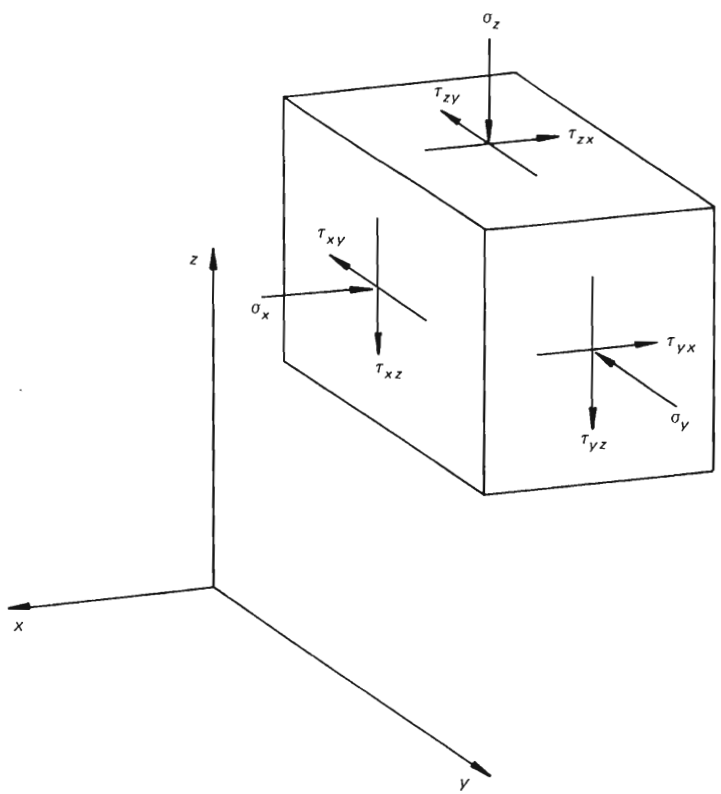


Fig. 1.3 – Definition of stress components

To completely define the state of stress at a point in the material it is necessary to consider the internal forces acting on three mutually perpendicular planes through the point. Thus stress components σ_y, τ_{yx} and τ_{yz} act on a plane perpendicular to the y axis and stress components σ_z, τ_{zx} and τ_{zy} act on a plane perpendicular to the z axis. Considering the equilibrium of an infinitesimal cube of material (Fig. 1.3):

$$\tau_{xy} = \tau_{yx},$$

$$\tau_{yz} = \tau_{zy},$$

$$\tau_{zx} = \tau_{xz}.$$

Hence there are six independent components of stress at a point in the material.

Usually the state of stress in a body is not constant but varies from point to point. Considering the equilibrium of an infinitesimal cube of material in a varying stress field (Fig. 1.4), the following equations are obtained:

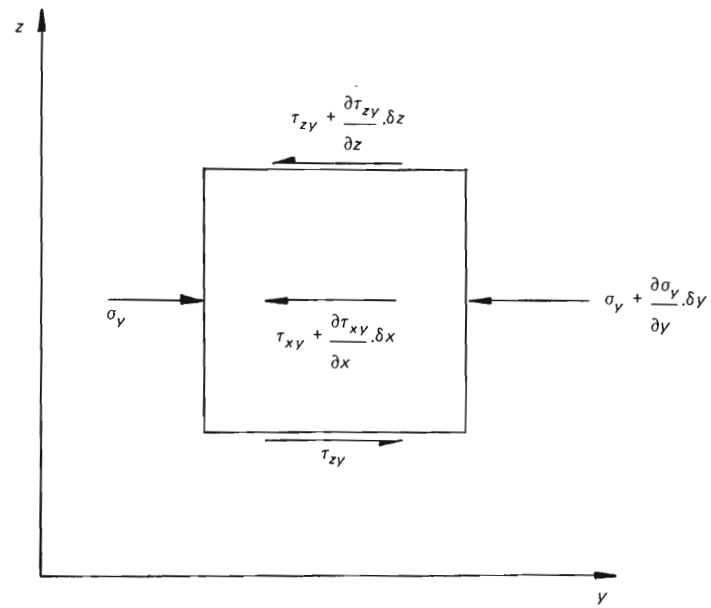


Fig. 1.4 – Stresses acting in a varying stress field (only stresses appearing in the equilibrium equation for the y direction are shown)

$$\frac{\partial \sigma_x}{\partial x} + \frac{\partial \tau_{yx}}{\partial y} + \frac{\partial \tau_{zx}}{\partial z} = w_x, \tag{1.1}$$

$$\frac{\partial \tau_{xy}}{\partial x} + \frac{\partial \sigma_y}{\partial y} + \frac{\partial \tau_{zy}}{\partial z} = w_y, \tag{1.2}$$

$$\frac{\partial \tau_{xz}}{\partial x} + \frac{\partial \tau_{yz}}{\partial y} + \frac{\partial \sigma_z}{\partial z} = w_z, \tag{1.3}$$

where w_x, w_y and w_z are the body forces per unit volume in the directions of the x, y and z axis respectively. If the y axis points vertically upwards then the body forces corresponding to the self-weight of the soil are $w_x = 0$, $w_y = -\gamma$ and $w_z = 0$, where γ is the soil's unit weight.

1.2.2 Displacements and strains (compatibility)

When a material is strained, a typical point with co-ordinates (x, y, z) moves to a new position $(x + d_x, y + d_y, z + d_z)$. Except for the case when the body is given a rigid-body translation the *displacements* d_x, d_y and d_z will vary across the body (i.e. they will each be functions of x, y and z).

Fig. 1.5 shows three infinitesimal fibres of length δx, δy and δz in a material and their new locations following straining. The direct strains ϵ_x, ϵ_y and ϵ_z and the engineering shear strains γ_{xy}, γ_{yz} and γ_{zx} are given by

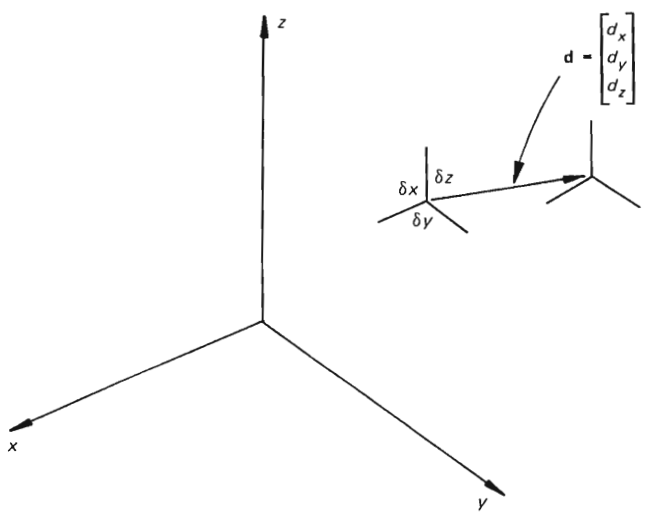


Fig. 1.5 – Definition of displacements

$$\epsilon_x = -\frac{\partial d_x}{\partial x}, \tag{1.4}$$

$$\epsilon_y = -\frac{\partial d_y}{\partial y}, \tag{1.5}$$

$$\epsilon_z = -\frac{\partial d_z}{\partial z}, \tag{1.6}$$

$$\gamma_{xy} = -\frac{\partial d_y}{\partial x} - \frac{\partial d_x}{\partial y}, \tag{1.7}$$

$$\gamma_{yz} = -\frac{\partial d_z}{\partial y} - \frac{\partial d_y}{\partial z}, \tag{1.8}$$

$$\gamma_{zx} = -\frac{\partial d_x}{\partial z} - \frac{\partial d_z}{\partial x}. \tag{1.9}$$

Most texts on continuum mechanics or elasticity use the symbols u, v and w for displacements. We use d_x, d_y and d_z to avoid confusion with the normal soil mechanics convention of u for pore water pressure and v for artificial seepage velocity. Note that a side effect of reversing the normal sign convection for strains is that a positive shear strain γ_{xy} corresponds to an increase in the angle between two fibres initially aligned with the x and y axes (see Fig. 1.6).

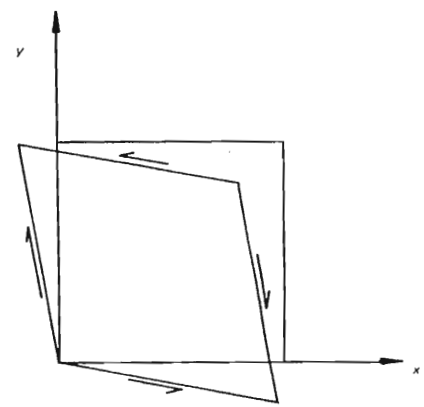


Fig. 1.6 – Positive shear strain for our sign convention

1.2.3 Elastic stress–strain relations

If elastic material is stressed in the x direction by a direct stress σ_x then it experiences strains:

$$\epsilon_x = \sigma_x/E,$$

$$\epsilon_y = -\nu\sigma_x/E,$$

$$\epsilon_z = -\nu\sigma_x/E,$$

where E is Young's modulus (or modulus of elasticity) of the material and ν is Poisson's ratio. A shear stress τ_{xy} gives rise to a shear strain:

$$\gamma_{xy} = \tau_{xy}\, 2(1+\nu)/E.$$

The effects of three direct stresses and three shear stresses can be superposed to give the generalised form of Hooke's Law:

$$\begin{aligned}
\epsilon_x &= \sigma_x/E - \nu\sigma_y/E - \nu\sigma_z/E,\\
\epsilon_y &= -\nu\sigma_x/E + \sigma_y/E - \nu\sigma_z/E,\\
\epsilon_z &= -\nu\sigma_x/E - \nu\sigma_y/E + \sigma_z/E,\\
\gamma_{xy} &= \tau_{xy}\, 2(1+\nu)/E,\\
\gamma_{yz} &= \tau_{yz}\, 2(1+\nu)/E,\\
\gamma_{zx} &= \tau_{zx}\, 2(1+\nu)/E.
\end{aligned}$$

These equations can be written in matrix form:

$$\begin{bmatrix} \epsilon_x \\ \epsilon_y \\ \epsilon_z \\ \gamma_{xy} \\ \gamma_{yz} \\ \gamma_{zx} \end{bmatrix} = \begin{bmatrix} 1/E & -\nu/E & -\nu/E & 0 & 0 & 0 \\ -\nu/E & 1/E & -\nu/E & 0 & 0 & 0 \\ -\nu/E & -\nu/E & 1/E & 0 & 0 & 0 \\ 0 & 0 & 0 & 1/G & 0 & 0 \\ 0 & 0 & 0 & 0 & 1/G & 0 \\ 0 & 0 & 0 & 0 & 0 & 1/G \end{bmatrix} \begin{bmatrix} \sigma_x \\ \sigma_y \\ \sigma_z \\ \tau_{xy} \\ \tau_{yz} \\ \tau_{zx} \end{bmatrix} \tag{1.10}$$

where G (which is equal to $E/(2(1+\nu))$) is the elastic shear modulus. These relations can be inverted to give stresses in terms of strains:

$$\begin{bmatrix} \sigma_x \\ \sigma_y \\ \sigma_z \\ \tau_{xy} \\ \tau_{yz} \\ \tau_{zx} \end{bmatrix} = \begin{bmatrix} 1-\nu & \nu & \nu & 0 & 0 & 0 \\ \nu & 1-\nu & \nu & 0 & 0 & 0 \\ \nu & \nu & 1-\nu & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.5-\nu & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.5-\nu & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.5-\nu \end{bmatrix} \begin{bmatrix} \epsilon_x \\ \epsilon_y \\ \epsilon_z \\ \gamma_{xy} \\ \gamma_{yz} \\ \gamma_{zx} \end{bmatrix}. \tag{1.11}$$

where

$$A = \frac{E}{(1-2\nu)(1+\nu)}.$$

This relation is often written in matrix notation:

$$\sigma = \mathbf{D}\epsilon. \tag{1.12}$$

It is sometimes more convenient to write these equations using the elastic parameters G (defined above) and K (the elastic bulk modulus). In fact it can be argued (see the next section) that it is preferable to use these parameters when defining the elastic properties of soil. K is the elastic modulus which appears in the equation relating volumetric strain to change in mean normal stress:

$$(\sigma_x + \sigma_y + \sigma_z)/3 = K(\epsilon_x + \epsilon_y + \epsilon_z)$$

where

$$K = \frac{E}{3(1-2\nu)}.$$

The D matrix can be written:

$$\begin{bmatrix} D_1 & D_2 & D_2 & 0 & 0 & 0 \\ D_2 & D_1 & D_2 & 0 & 0 & 0 \\ D_2 & D_2 & D_1 & 0 & 0 & 0 \\ 0 & 0 & 0 & D_3 & 0 & 0 \\ 0 & 0 & 0 & 0 & D_3 & 0 \\ 0 & 0 & 0 & 0 & 0 & D_3 \end{bmatrix},$$

where

$$\begin{aligned}
D_1 &= K + (4/3)G,\\
D_2 &= K - (2/3)G,\\
D_3 &= G.
\end{aligned}$$

1.3 SOIL MECHANICS

1.3.1 Effective stresses

Saturated soil is a two-phase continuum consisting of solid particles and water in the pores. Terzaghi showed that the definition of effective stresses allows a rational treatment of the stress–strain behaviour. Effective stresses are defined by the equations

$$\begin{aligned}
\sigma'_x &= \sigma_x - u,\\
\sigma'_y &= \sigma_y - u,\\
\sigma'_z &= \sigma_z - u,\\
\tau'_{xy} &= \tau_{xy},
\end{aligned}$$

$$\tau'_{yz} = \tau_{yz},$$

$$\tau'_{zx} = \tau_{zx},$$

where u is the pore water pressure.

Terzaghi's principle of effective stress states that all measurable effects of a change in stress in soils (such as compression, distortion, or a change in shearing resistance) are due to changes in effective stresses. Thus changing the pore water pressure and normal total stresses by equal amounts produces no strains.

One consequence of Terzaghi's principle is that when soil (either dry or saturated) is to be described by elastic stress–strain relations, the equations must refer to effective (rather than total) stresses. Thus it is appropriate to write

$$\sigma' = \mathbf{D}'\epsilon \tag{1.13}$$

where the matrix D' contains elastic moduli E' and ν' rather than E and ν. The significance of these 'effective stress parameters' (i.e. E' and ν') will be discussed further below. In geotechnical problems we are frequently interested in strains caused by changes in effective stresses and so we rewrite (1.13) as

$$\delta\sigma' = \mathrm{D}'\delta\epsilon. \tag{1.14}$$

$\delta\sigma'$ and $\delta\epsilon$ represent incremental changes in effective stresses and strains.

1.3.2 A physical interpretation of effective stress

A physical interpretation of soil effective stresses will be useful in thinking about soil behaviour and in particular the role of effective stresses as defined above. A good mental picture of soil structure is a collection of approximately spherical solid particles surrounded by water.† When loads are applied to the soil, the loads are transferred internally through the soil partly by the solid phase and partly by the water. Loads transferred by the solid phase are transferred between the particles via their points of contact. If a plane is constructed through a typical contact point (Fig. 1.7) then equilibrium of forces across the plane gives

$$A\sigma = A_{\mathrm{w}}u + A_{\mathrm{s}}\sigma_{\mathrm{c}}$$

where

A is the area of the plane

A_{w} is the area of the plane across which the force is transmitted by the water

A_{s} is the area of the plane across which the force is transmitted by the particle contact

† Of course, neither clays nor most sands are really like this. The point is that the simplified 'mental picture' is capable of yielding results which are appropriate to real soil behaviour. It is not necessary to refine the mental picture to include factors such as actual particle shape.

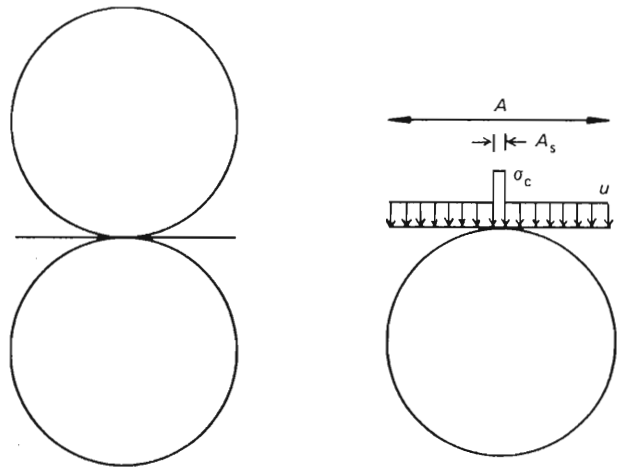


Fig. 1.7 – Forces acting on a plane through particle contact point

σ is the total stress acting normal to the plane

u is the pore water pressure

σ_{c} is the average contact stress between the two particles.

Now $A_{\mathrm{w}} \gg A_{\mathrm{s}}$, i.e. A_{w} is approximately equal to A, so it is possible to write

$$(A_{\mathrm{s}}/A)\sigma_{\mathrm{c}} = \sigma - u$$

Thus effective stresses can be regarded as the contact forces between soil particles averaged over the *whole* area of the soil.

1.3.3 Elastic constants for dry soil

How do we make use of the appropriate 'effective stress moduli' when soil is loaded? We shall answer this question by first considering the (relatively) simple case of dry soil with air in the pore space. The important point to appreciate is that the effective stress elastic moduli for soil describe the elastic properties of an assemblage of soil particles rather than the elastic moduli of the material which makes up the solid phase of the soil. Consider a cylindrical sample of dry soil in a triaxial apparatus. Again we think of the soil as being a collection of roughly spherical particles, now with elastic properties.† If an all-round total pressure is applied to the sample, then the strains can be calculated from (1.14).

† Even when elastic modelling of soil is appropriate (for example: the calculation of small deformations of over-consolidated soils) this mental picture is not quite accurate. However, it turns out again that the conclusions we draw from this model are appropriate to real soil behaviour.

(In this case the effective stresses are the same as the imposed total stresses since the pore water pressure is zero.) The shear strains are zero and the volumetric strain can be calculated from

$$\delta V/V = \delta\sigma/K'. \tag{1.15}$$

An examination of the collection of elastic soil particles would reveal some flattening of the contact points between the particles, but apart from this they would not change in shape very much. A very small change in volume of the particles would be accompanied by a larger change in volume of the void space (see Fig. 1.8). Thus the elastic bulk modulus K' is measuring the bulk stiffness of the collection of particles rather than the stiffness of the material which constitutes those particles. In other words the soil is more 'squashy' than if there were no voids present.

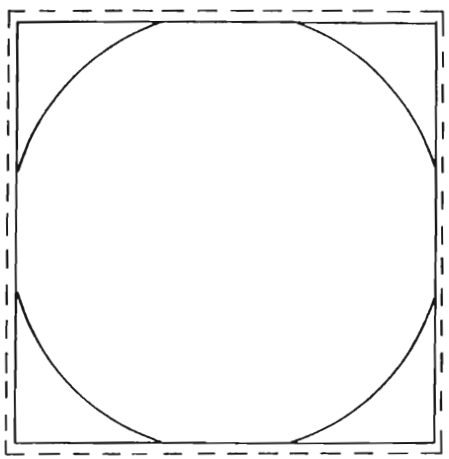


Fig. 1.8 – Consider a collection of spherical particles in a 'simple cubic' packing (where each sphere is in contact with six neighbours). A 1% direct strain in three directions corresponds to a volumetric strain of 3% in the 'unit cell' and thus the overall soil mass. If the particles are rigid, apart from flattening of the contact points, then approximately 0.02% of this strain is due to a change in volume of the solid particles and the remaining 2.98% is due to a change in volume of the void space

1.3.4 Elastic constants for saturated soil

Now consider a specimen of saturated soil in a triaxial apparatus. The pore water pressure is initially at atmospheric pressure and the drainage tap is turned off before the soil is loaded. An all-round total pressure $\delta\sigma$ is now applied to the triaxial sample.

If V is the volume of the soil and V_s and V_w the volumes of the solid and water phases, then

$$V = V_s + V_w. \tag{1.16}$$

Resulting from the change in all-round pressure the soil decreases in volume by δV. This overall decrease in volume consists of decreases in the solid and water phases δV_s and δV_w respectively. Clearly:

$$\delta V = \delta V_s + \delta V_w. \tag{1.17}$$

Note that the normal assumption is that saturated soil is incompressible when drainage is not allowed. Here, however, we are attempting an accurate analysis of the very small changes in volume which take place. These are given by

$$\delta V/V = (1/K_u)\,\delta\sigma, \tag{1.18}$$

$$\delta V_w/V = (1/K_w)\delta u, \tag{1.19}$$

$$\delta V_s/V = (1/K_s)\delta u, \tag{1.20}$$

where K_u, K_w and K_s are the elastic bulk moduli of the soil composite and the two phases (i.e. water and solid) respectively. Equations (1.18) and (1.19) are definitions of K_u and K_w. Equation (1.20) perhaps needs some comment: the volumetric compression of the solid particles is caused by the increase in pore water pressure (see Fig. 1.9). The change in effective stress $\delta\sigma'$ must be consistent with the two equations

$$\delta\sigma = \delta\sigma' + \delta u, \tag{1.21}$$

$$\delta V/V = (1/K')\delta\sigma'. \tag{1.22}$$

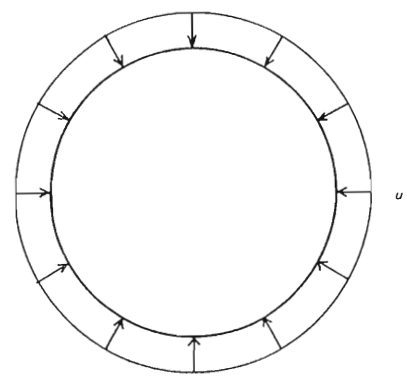


Fig. 1.9 – Change in volume of the solid particles is due mainly to the change in pore water pressure which acts on them

Equations (1.17) to (1.22) can be regarded as six equations in six unknowns (δV, δV_w, δV_s, $\delta\sigma'$, δu and K_u). Manipulation of the equations gives

$$K_u = K' + K_w \frac{V}{V_w} \frac{1}{(K_w/K_s)(V_s/V_w) + 1}. \tag{1.23}$$

Since the elastic bulk modulus of the grains is about 30 times as large as that of water, (1.23) can be written:

$$K_u = K' + K_w(V/V_w). \tag{1.23a}$$

A further simplification follows the observation that K' is much smaller than K_w:

$$K_u = K_w(V/V_w). \tag{1.23b}$$

Here it is convenient to introduce the normal soil mechanics definition of the voids ratio:

$$e = V_w/V_s$$

and therefore:

$$K_u = (1 + 1/e)K_w. \tag{1.23c}$$

Thus the bulk compressibility of saturated soil is effectively due to the bulk compressibility of the water phase alone (but taking account of the fact that the water only occupies a certain fraction of the soil volume). The approximations that we have made in obtaining this result are equivalent to taking $\delta V = \delta V_w$, $\delta V_s = 0$, $\delta\sigma' = 0$ and $\delta u = \delta\sigma$ in equations (1.17) to (1.22). Therefore the undrained loading produces no change in the effective stresses: the external load is carried by the pore water pressure.

Now suppose that the drainage tap is opened. The difference in pressure between the pore water in the sample and the water outside causes water to flow out of the sample. The rate at which this outflow takes place is controlled by the pore size of the soil, but eventually the pore water pressure in the sample returns to atmospheric pressure. The change in the effective stress is now equal to the change in the total stress ($\delta\sigma' = \delta\sigma$) and the volumetric strain can be calculated from

$$\delta V/V = \delta\sigma/K'.$$

This equation is identical to (1.15), which gave the volumetric strain for dry soil. When calculating the long-term soil strains we must clearly use the effective stress elastic properties. Soil is a rather special kind of material when examined from the viewpoint of traditional continuum mechanics. This is because the elastic volumetric 'strain' associated with the definition of effective bulk modulus is due to the disappearance of some water from a small element of soil rather than a change in volume of the individual components which make up the soil.

This example demonstrates the difference between two modes of soil behaviour which geotechnical engineers often identify. *Drained* deformation takes place when the soil is strained slowly and the water in the soil pores escapes as the water pressures return to their original (perhaps hydrostatic) values. In *undrained* deformation the straining takes place sufficiently quickly so that the water does not have the time to flow out of the pores, i.e. the soil

behaves essentially as an incompressible material. So far we have been looking at the volumetric behaviour of soil and we have identified two elastic bulk moduli: K_u appropriate for undrained behaviour and K' appropriate for drained behaviour. In Table 1.1 we summarise the relationships between the full set of eight elastic moduli which describe isotropic behaviour.

Table 1.1

Elastic constant	
E'	(Regarded here as an independent parameter)
ν'	(Regarded here as an independent parameter)
K'	$= E'/(3(1-2\nu'))$
G'	$= E'/(2(1+\nu'))$
E_u	$= 1.5E'/(1+\nu')$ (see text)
ν_u	0.5
K_u	Infinite
G_u	$= G'$ (see text)

Since the pore water has no shear stiffness it cannot make a contribution to the elastic shear stiffness of the soil. Thus the symbol $G(= G' = G_u)$ is usually used for shear modulus. Note that this implies $E'/(2(1+\nu')) = E_u/3$, and this equation is used to obtain the relationship between E' and E_u quoted in Table 1.1.

It should now be possible to appreciate the comment in section 1.2 that K and G are elastic properties more appropriate for the description of soil behaviour (more appropriate than E and ν, that is). G remains the same for drained and undrained behaviour, and the effective bulk modulus K' allows the calculation of drained volumetric strains. If partially drained behaviour is considered (that is before pore pressure equilibrium is finally reached) then G is again appropriate for the calculation of shear strains and some value of K between K' and infinity could be assumed for the calculation of volumetric strains.

1.3.5 Flow of water through soils

The rate of flow of water through soil is controlled by two factors, firstly the size of the pores and secondly the gradient of water pressure which is tending to cause the flow. These two factors are encompassed in Darcy's Law:

$$v = k\,i$$

where

v is the 'artificial' velocity of the water (i.e. the flow rate divided by the whole cross-sectional area through the soil)

k is the soil permeability (independent of flow rate for a wide range of velocities)

i is the hydraulic gradient.

The definition of hydraulic gradient is shown in Fig. 1.10. Note that the position of the datum shown in the figure is arbitrary – only the gradient of hydraulic head appears in Darcy's Law. In this book the term 'excess pore pressure' is defined as the hydraulic head divided by the bulk density of water:

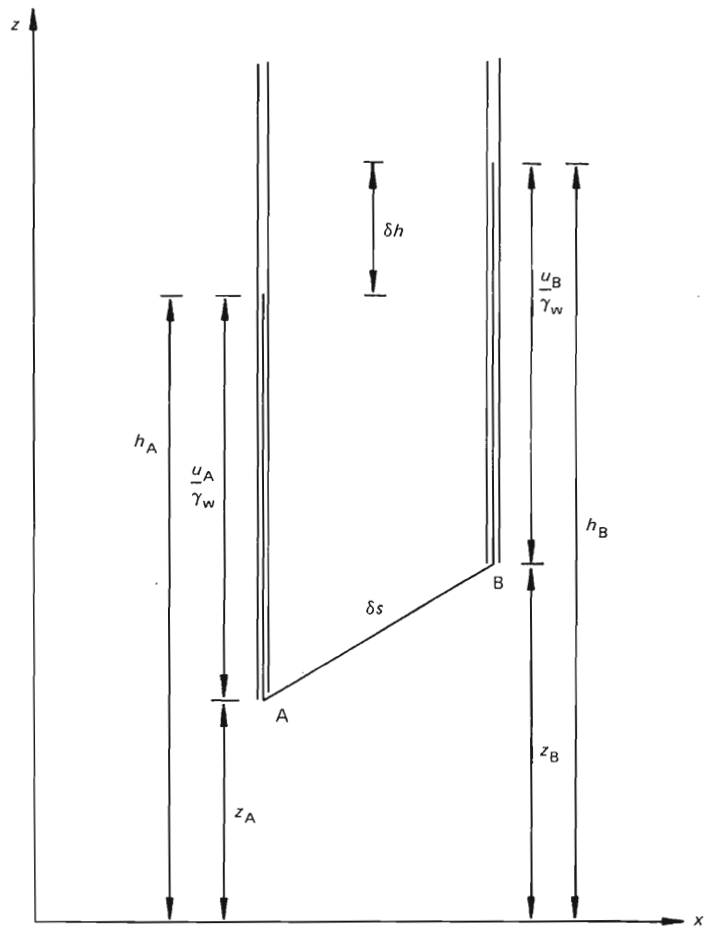


Fig. 1.10 – Hydraulic gradient $= -\delta h/\delta s$, a positive gradient causing flow from A to B

$$\bar{u} = h/\gamma_w; \tag{1.24}$$

thus it is always possible to calculate the actual pore pressure from an excess pore pressure (and vice versa) by an equation of the form

$$\bar{u} = u + z\gamma_w \tag{1.25}$$

where z is the height of the point at which the pore pressure is being measured above the arbitrary datum. The reader should note that our definition of hydraulic head is the standard one. The definition of excess pore pressure, however, differs from that given in some texts on soil mechanics. This differen arises because it is normal to consider steady seepage problems (where po pressures do not change with time) separately from consolidation problems (where pore pressures vary with time). In the former case, hydraulic head is the basic variable used in solving the problem whereas excess pore pressures are used in the latter case. For the purposes of our finite element formulation we need to link together these two quantities and this is done via (1.24). Consider an analysis of a consolidation problem with under-drainage (as in section 3.6.4). Using the present definition of excess pore pressure, the final state of steady seepage downwards has a linear variation of excess pore pressure. In contrast it would often be assumed that the excess pore pressure is the time dependent component of the pore pressure which eventually decays to zero. The point to note is that both definitions of excess pore pressure satisfy the basic differential equation derived by Terzaghi (Terzaghi and Frohlich, 1936):

$$\frac{\partial \bar{u}}{\partial t} = c_v \frac{\partial^2 \bar{u}}{\partial z^2},$$

where c_v is the coefficient of consolidation.

Geotechnical engineers often need to predict the distribution of pore pressures in a mass of soil under the condition of steady seepage. The basic equation which must be satisfied at all points within the soil is obtained by considering the flow of water into and out of an infinitesimal element of so (Fig. 1.11) (under conditions of steady seepage there must be no volum change):

$$\frac{\partial v_x}{\partial x} + \frac{\partial v_y}{\partial y} + \frac{\partial v_z}{\partial z} = 0. \tag{1.26}$$

The permeability of the soil may be different in the directions of the three co-ordinate axes, and the general form of Darcy's Law is

$$v_x = -\frac{k_x}{\gamma_w}\frac{\partial \bar{u}}{\partial x}, \tag{1.27}$$

$$v_y = -\frac{k_y}{\gamma_w}\frac{\partial \bar{u}}{\partial y}, \tag{1.28}$$

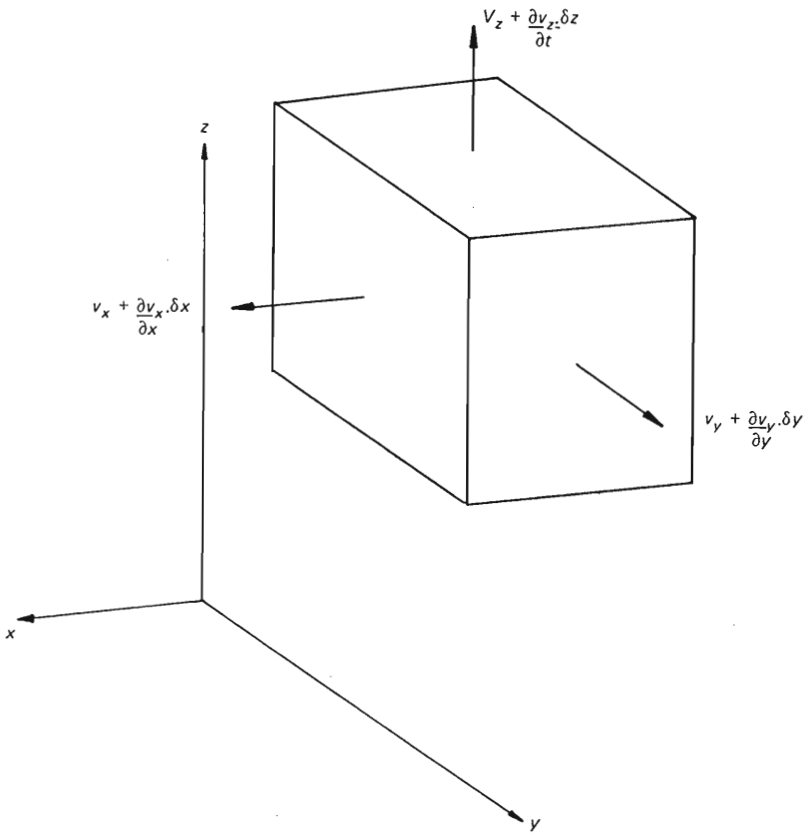


Fig. 1.11 – flow out of a small element of soil

$$v_z = -\frac{k_z}{\gamma_w}\frac{\partial \bar{u}}{\partial z}. \tag{1.29}$$

Substituting these relations into the equation of continuity (1.26):

$$k_x \frac{\partial^2 \bar{u}}{\partial x^2} + k_y \frac{\partial^2 \bar{u}}{\partial y^2} + k_z \frac{\partial^2 \bar{u}}{\partial z^2} = 0. \tag{1.30}$$

For the same permeability in all directions (i.e. $k_x = k_y = k_z$) this equation reduces to Laplaces equation which governs a number of other physical phenomena (e.g. the flow of electricity in a conducting medium and the stresses in an elastic bar under a torsional load).

These equations may be extended to the case of time dependent flow of water in soil. The basic equation now becomes

$$\frac{k_x}{\gamma_w}\frac{\partial^2 \bar{u}}{\partial x^2} + \frac{k_y}{\gamma_w}\frac{\partial^2 \bar{u}}{\partial y^2} + \frac{k_z}{\gamma_w}\frac{\partial^2 \bar{u}}{\partial z^2} + \frac{\partial v}{\partial t} = 0 \tag{1.31}$$

where the last term in this equation is equal to the rate of volumetric strain of a soil element.

This equation together with the equations of differential equilibrium, the equations defining effective stresses and the effective stress–strain relations are known as Biot's equations of consolidation (Biot, 1941). The one-dimensional form of these equations is precisely equivalent to Terzaghi's one-dimensional consolidation theory.

2

Critical State Soil Mechanics

2.1 INTRODUCTION

The theories of soil behaviour known as 'critical state soil mechanics' were developed from the application of the theory of plasticity to soil mechanics. It is possible to appreciate and use many of the ideas of critical state soil mechanics without making much reference to the theory of plasticity. Indeed there is a tendency to teach critical state soil mechanics in this way because many degree courses in civil engineering do not find room for a proper account of plasticity theory. We regret this. In our view a real appreciation of critical state soil mechanics requires a knowledge of plasticity theory. To understand how soil deformations can be predicted (for example by a finite element program such as CRISP) using the theories of critical state soil mechanics, familiarity with plasticity theory is essential. Hence the first few sections of this chapter are devoted to an explanation of some of the fundamental ideas in this theory.

Fig. 2.1 shows the stress–strain curve obtained from testing a bar of metal in a tension test. Initially the relation between stress and strain is linear (OA in the figure). If the bar is unloaded from any point on OA then the stress–strain relation for the material follows the same path but in the reverse direction to the origin. If the bar is loaded beyond A then subsequent unloading is also reversible, even though part of the stress–strain relation is non-linear. However, there is a point B beyond which unloading is *not* reversible: this is called the *yield point* of the material. When the bar is loaded up to the point C and then unloaded, the path CD is followed. OD represents a permanent strain which remains on unloading. This permanent strain is known as the *plastic* strain experienced by the metal.

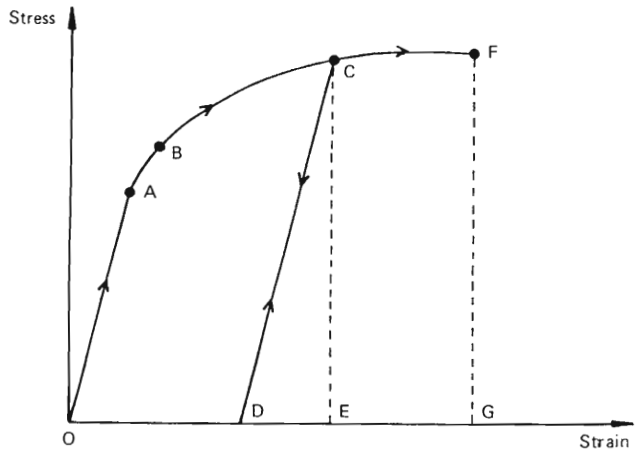


Fig. 2.1 – Stress–strain curve typical of many metals

Up to point B the behaviour of the bar is regarded as being elastic. It is the reversibility rather than the linearity which is the important feature of behaviour which distinguishes between elastic and plastic straining of a material. However, points A and B can often be regarded as being coincident for practical purposes. When the material is in a state represented by the point C, the total strain OE is made up of the plastic strain OD and an elastic strain DE which is completely recovered on unloading. The slope of the elastic unloading line CD is usually very close to the initial elastic loading portion OA.

Reloading the metal from the point D results in the line DC being followed until the point C is reached which is the new yield point of the material. Further loading follows a continuation of the original stress–strain curve until the maximum stress is reached (point F) when the bar fails. The stress at the point F (i.e. FG in the figure) is the strength of the metal in direct tension. This is often called the ultimate tensile strength or (UTS).

Suppose that two similar bars of the same metal are tested. The first has gone through a stress cycle OCD, but the second has not. The first bar has a higher yield point than the second and thus the material seems to be harder. The process of raising the yield point is called 'hardening' the material. The amount that the yield stress is raised is often linked to either the plastic strain or the mechanical work that is done on the material. Thus the terms 'strain-hardening' and 'work-hardening' are often used to describe this kind of behaviour.

The type of behaviour described above is typical of an alloy of aluminium such as duralumin. Other metals (and soils) display plastic behaviour which is broadly similar to that described above, but the behaviour shows some differences in detail. Some of these differences are shown in Fig. 2.2. Fig. 2.2(a) shows the phenomenon of an upper yield point which is displayed by low-carbon steels. Fig. 2.2(b) shows that, when a material is unloaded from tensile

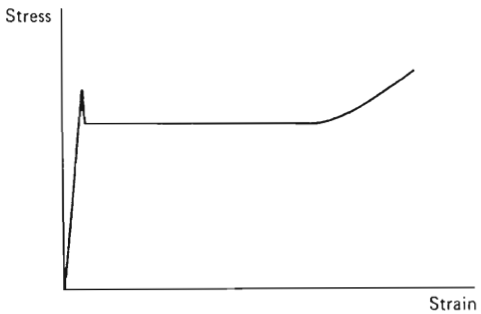


Fig. 2.2(b) – The Bauschinger effect ($\sigma_{Y_1} > \sigma_{Y_2}$)

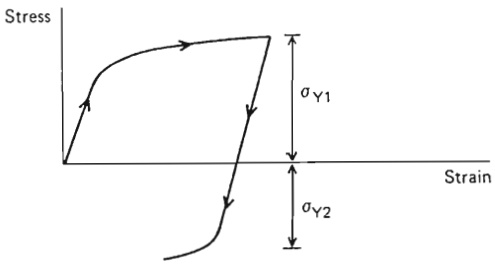


Fig. 2.2(a) – Upper and lower yield points for a mild steel

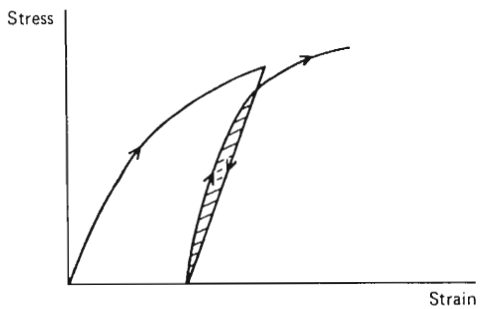


Fig. 2.2(c) – Anelastic behaviour: the shaded area represents an amount of energy dissipated during the 'elastic' hysteresis loop

yielding, it can yield in compression at a lower stress than if it were reloaded in tension. This is known as the Bauschinger effect. Fig. 2.2(c) shows the phenomenon of anelasticity or elastic hysteresis. A material which has been subject to elastic unloading and is then reloaded does not always follow exactly the same stress–strain path. The shaded area within the 'hysteresis loop' of the stress–strain curve represents an amount of energy which is dissipated during straining.

2.2 IDEALISATIONS OF PLASTIC BEHAVIOUR

Plasticity is a very useful feature of the behaviour of metals for a number of reasons. Firstly a large amount of plastic straining before failure (known as ductility) signals the imminent collapse of a structure before catastrophic failure occurs. Secondly the ability to deform metals plastically under high stresses is the basis of many manufacturing processes such as rolling, drawing, machining or pressing in dies. Thirdly the complete description of the strength of metals within the mathematical theory of plasticity allows buildings and mechanical engineering components to be designed to provide a factor of safety against overall collapse (rather than designing to prevent some local part of the structure from becoming overstressed).

The plastic behaviour of soils allows a rational treatment of bearing capacities of foundations and the failure of slopes, excavations and tunnels. It also allows complete description of the stress–strain behaviour of soils so that soil deformations can be predicted right up to failure. Admittedly the behaviour of soil is more complex than is accounted for by current elasto-plastic models of behaviour. However, attempts to produce new mathematical descriptions of soil behaviour invariably use the framework of elasto-plasticity.

In order to predict the behaviour of engineering structures when plastic behaviour is involved, the first step is to choose an appropriate idealisation of plasticity. In such an idealisation the main features of the behaviour are identified and included in the description, but aspects of secondary importance are ignored. Fig. 2.3(a) shows the idealisation known as elastic–perfectly plastic. Here the first part of the stress–strain curve is linear and elastic until the material yields. The material then continues to deform at a constant yield stress. In the terminology of plasticity the material exhibits no strain-hardening. Fig. 2.3(b) shows the simplest way of incorporating strain-hardening into an idealisation. When the material yields, the stress–strain curve is still linear but at a reduced slope. This type of behaviour is referred to as elastic–linear-strain-hardening plastic. Sometimes (when only collapse loads are to be considered in a calculation) it is convenient to idealise the behaviour as rigid-plastic (see Fig. 2.3(c)).

The idealisations of plastic behaviour which have just been described will sometimes be suitable to describe the behaviour of soil. (Indeed the rigid-plastic idealisation underlies most stability calculations in soil mechanics.) However, soil exhibits a rather more complex behaviour than metals, and the main aim of this chapter is to describe a more appropriate idealisation.

To completely describe the stress–strain relations for an elasto-plastic material, four different types of statement are required.

(a) A yield function for the material. This generalises the concept of the yield stress described above to two- and three-dimensional stress states.
(b) A relationship between the directions of the principal plastic strain increments and the principal stresses.

(c) A flow rule for the material. This specifies the relative magnitudes of the incremental plastic strains when the material is yielding.

(d) A hardening law for the material. This is a relationship between the amount a material hardens and the plastic strain the material undergoes or the work that is done on the material when it is yielding.

Each kind of statement is considered in more detail in sections 2.3 and 2.4.

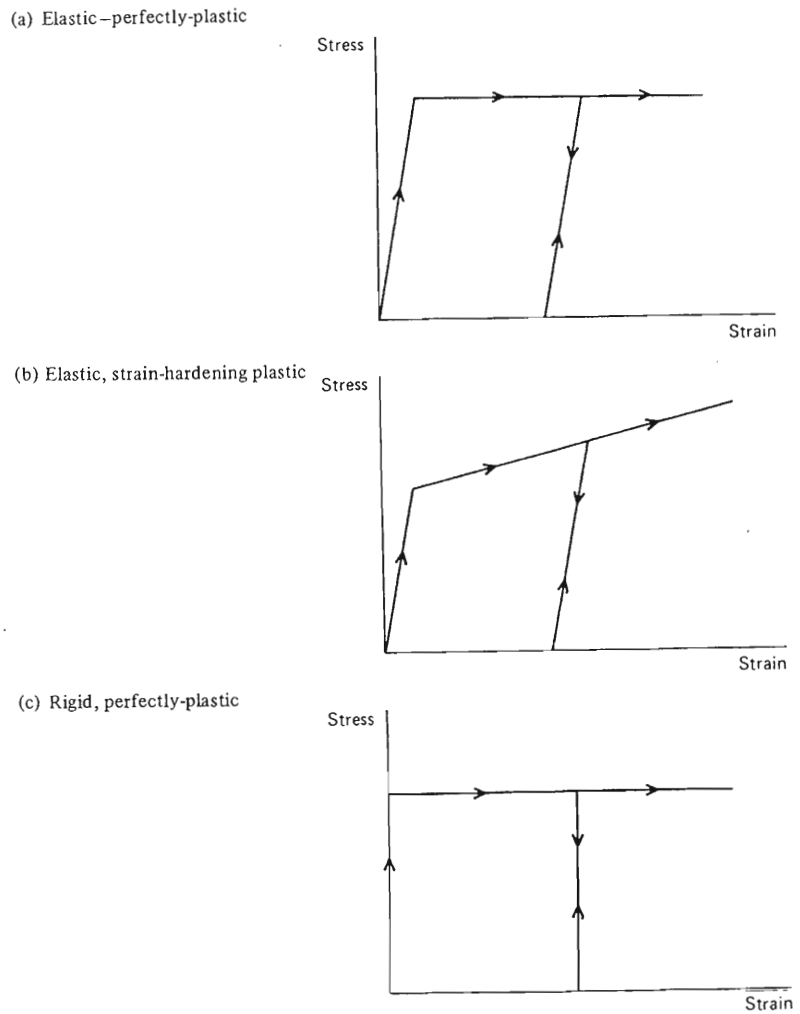


Fig. 2.3 – Idealisations of plastic behaviour

2.3 YIELD FUNCTIONS

So far the discussion of plastic behaviour has been limited to the case of uniaxial straining – only one stress has been involved in describing the loading applied to the material. When a material is subjected to two- or three-dimensional states of stress, then whether the material is elastic or plastic will in general depend on all the stress components acting (which number six in the fully three-dimensional case). When material behaviour is isotropic (same properties in all directions), then it is only necessary to consider the values of the principal stresses (σ_a, σ_b, and σ_c).

2.3.1 Yield functions for metals

For the case of metals, two criteria for 'elastic breakdown' are due to Tresca and von Mises. Tresca's criterion states that plastic yielding starts when the maximum shear stress reaches a certain value k. This happens when the principal stresses satisfy the following equation:

$$\text{Max}\,(|\sigma_a - \sigma_b|,\quad |\sigma_b - \sigma_c|,\quad |\sigma_c - \sigma_a|) = 2k. \tag{2.1}$$

This equation can be represented in principal stress space as the surface of a prism with a hexagonal cross-section, centred on the hydrostatic ($\sigma_a = \sigma_b = \sigma_c$) axis (see Fig. 2.4). When the stress state of an element of material is represented as a point inside this surface, the material behaviour is elastic. When the stress state is described by a point on the surface, then the material is yielding. (Stress states outside the surface are impossible to attain.)

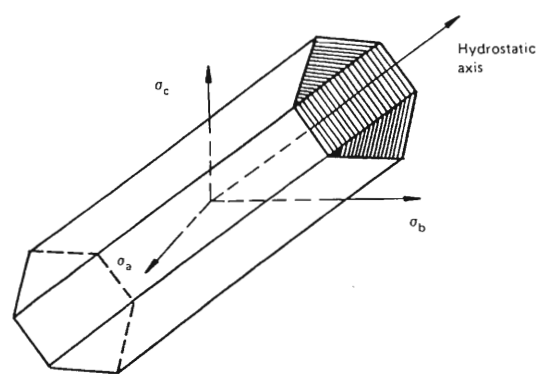


Fig. 2.4 – The Tresca yield surface

von Mises' criterion states that plastic yielding starts when the following equation is satisfied:

$$(\sigma_a - \sigma_b)^2 + (\sigma_b - \sigma_c)^2 + (\sigma_c - \sigma_a)^2 = 2\,\sigma_Y^2. \tag{2.2}$$

This criterion is equivalent to plastic yielding starting when the elastic strain energy due to shearing reaches a critical value. Here σ_Y is the yield stress in uniaxial tension. (Considering the stress state in uniaxial tension we see that Tresca's $k = 0.5\sigma_Y$.) In principal stress space, (2.2) is equivalent to a cylindrical surface (Fig. 2.5) which coincides with the Tresca surface on the edges (i.e. where $\sigma_a = \sigma_b$ or $\sigma_b = \sigma_c$ or $\sigma_c = \sigma_a$).

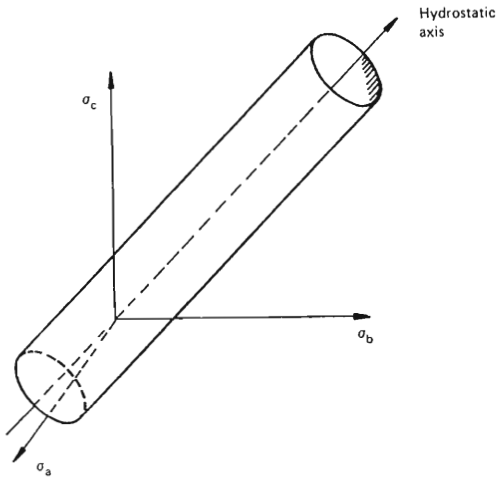


Fig. 2.5 – The von Mises yield surface

In general a yield function for an isotropic material is written:

$$f(\sigma_a, \sigma_b, \sigma_c) = 0,$$

this equation representing a surface in three-dimensional stress space. The Tresca and von Mises yield criteria are two examples of the more general form. It is conventional to write the yield function in such a way that if one substitutes into the function the current stress state, then a negative value of the function indicates that behaviour is elastic (inside the yield surface). A zero value of the function indicates that yielding is taking place, and by convention positive values are not allowed.

2.3.2 Some yield functions suggested for soils

Now we turn to a yield surface perhaps more appropriate for soils. In 1773 the French engineer Coulomb (Coulomb 1773) introduced in his analysis of the thrust acting on a retaining wall the failure condition for soil (usually called the Mohr–Coulomb criterion) which is still in wide use:

$$\tau = c + \sigma \tan \phi.$$

Today, geotechnical engineers prefer to write this equation in terms of effective stresses:

$$\tau = c' + \sigma' \tan \phi'. \tag{2.3}$$

Although this equation is normally interpreted in terms of a Mohr's circle plot, we can instead represent this failure criterion in the three-dimensional stress space that we have been using to describe the yielding of metals. This is achieved by rewriting the equation:

$$\sigma_1' - \sigma_3' = \sin \phi' (\sigma_1' + \sigma_3' + 2c' \cot \phi')$$

where σ_1' and σ_3' are the major and minor principal effective stresses respectively. Taking account of the six possible permutations of the magnitudes of σ_a', σ_b' and σ_c' (i.e. $\sigma_a' > \sigma_b' > \sigma_c'$, $\sigma_a' > \sigma_c' > \sigma_b'$, etc.) six planes are generated in (σ_a', σ_b', σ_c') space. Thus the Mohr–Coulomb yield criterion is equivalent to the irregular hexagonal pyramid in principal effective stress space shown in Fig. 2.6. In fact the Mohr–Coulomb criterion represents an incomplete picture of the yielding of soils. Firstly, soils show evidence of volumetric yielding under isotropic stress changes where Mohr–Coulomb suggests elastic behaviour. Secondly, if one follows the normal approach of calculating plastic strains when yielding (as used for metals and described in section 2.4.2), then the predictions of expansive volumetric strains are unrealistic.

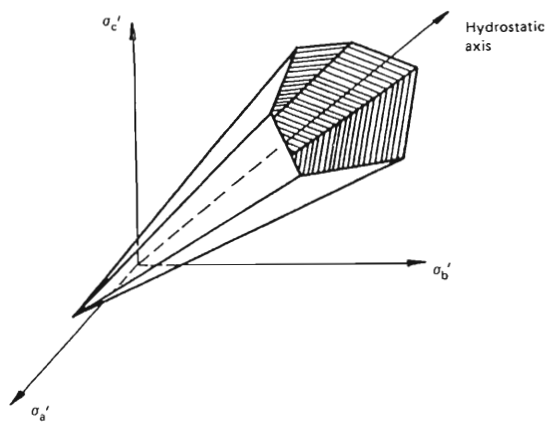


Fig. 2.6 – The Mohr–Coulomb yield surface

We conclude this section on yield surfaces with the yield surface proposed by Drucker & Prager (1952). For some metal plasticity calculations, von Mises is more convenient than Tresca, and so Drucker and Prager believed it might be useful to 'round-off' the Mohr–Coulomb yield surface to give the conical surface for soils shown in Fig. 2.7. This has all the drawbacks of the Mohr–Coulomb yield surface and gives a worse fit to the data of soil failure. As a yield surface

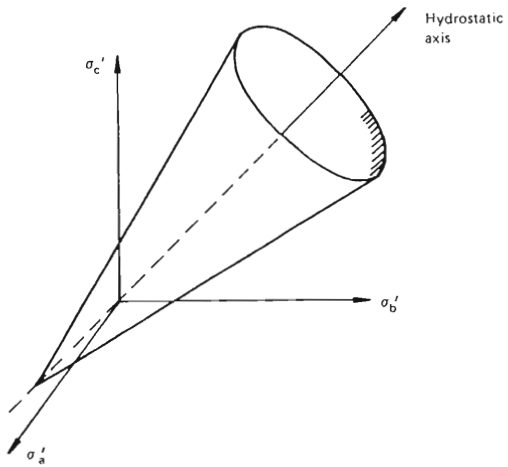


Fig. 2.7 – The Drucker–Prager yield surface

for soils, it does not have much in its favour, and we include it partly to 'complete the set' and partly because the conical shape reappears in the Cam-clay model, not as a yield surface, but as the 'critical state cone' (see Chapter 5).

2.3.3 The hardening law

The hardening law generalises the concept of the uniaxial yield stress being increased by strain-hardening to more general stress states. Hardening a material can result in the yield surface either being enlarged or being translated in stress space (or perhaps some combination of the two). These two possibilities are illustrated in Fig. 2.8. The former is normally called 'isotropic-hardening' and the latter 'kinematic-hardening'. The kinematic-hardening assumption can describe behaviour such as the Bauschinger effect described earlier. Although the assumption of isotropic-hardening is less realistic for many materials, it is more often used because it is simpler to describe mathematically. If the loading applied to the material is monotonic, then the assumption of isotropic-hardening will be adequate (because the 'opposite' side of the yield surface is not encountered). The hardening law is incorporated into the yield surface equation by writing

$$f(\sigma, \mathbf{h}) = 0, \tag{2.4}$$

where $\mathbf{h}$ is a vector of hardening parameters. The hardening parameters will define the size of the yield locus and there will be some prescribed relationship between the hardening parameters and the components of the plastic strain (for a strain-hardening material). In the simplest case, there may be just one hardening parameter, say h_1, which may be the same as the yield stress in uniaxial tension, for example. One particular value of h_1 will be relevant for a

yield locus of a certain size, and after strain-harding there will be a larger yield locus associated with a larger value of h_1.

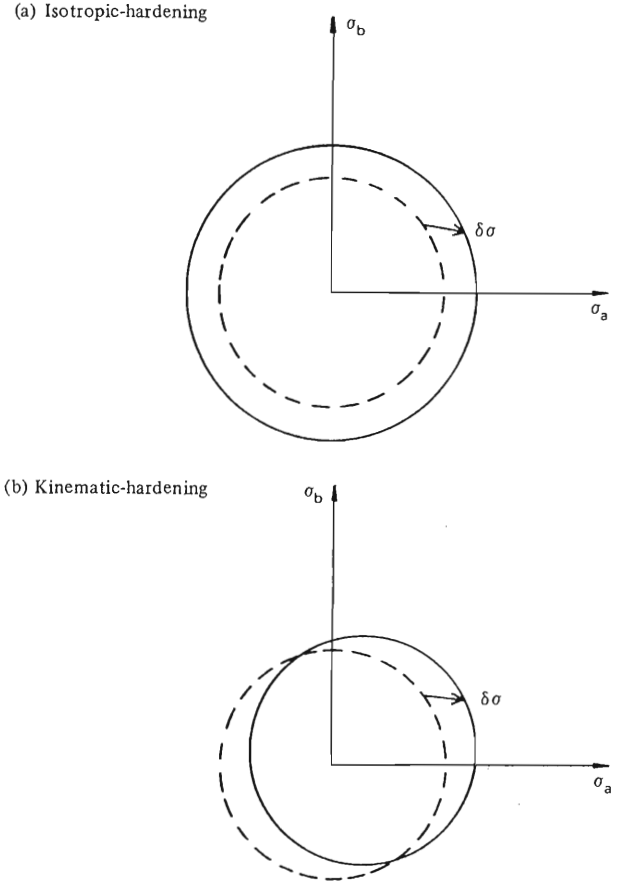


Fig. 2.8 – Two methods of describing hardening

2.4 PLASTIC STRAINS

2.4.1 Co-incidence of principal axes

Consider a cube of material which is subjected to principal stresses σ_a, σ_b, and σ_c (Fig. 2.9(a)). A small incremental shear stress $\delta\tau$ is now applied to four faces of the cube. If the cube deforms elastically then the incremental strains are as shown in Fig. 2.9(b). If the cube deforms plastically then the incremental strains

are as shown in Fig. 2.9(c). In elastic behaviour the directions of the principal strain increments coincide with the directions of the principal stress increments. In plastic behaviour the directions of the principal strain increments coincide with the directions of the principal stresses (*not* the principal stress increments). This *coaxiality* of the principal strain increments and the principal stresses is associated with plastic theories describing isotropic material behaviour.

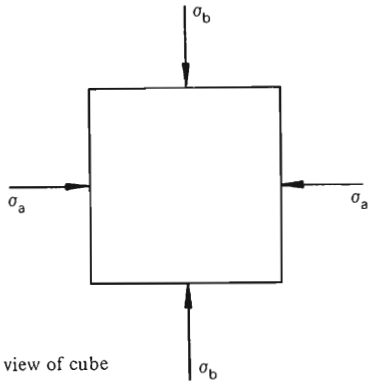


Fig. 2.9(a) – Side view of cube

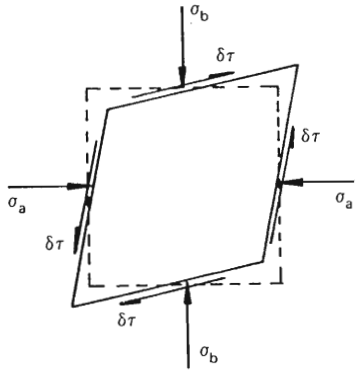


Fig. 2.9(b) – Elastic response to an increment of shear stress

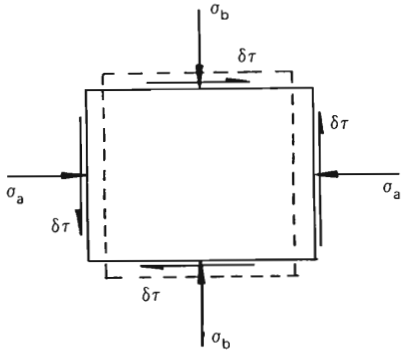


Fig. 2.9(c) – Plastic response to an increment of shear stress

Fig. 2.9 – Elastic and plastic response of a cube subjected to shear (only the side view is shown; σ_c is the out-of-plane stress)

2.4.2 Flow rules

The flow rule for a plastic material gives the ratios of the plastic strain increments when the material is yielding in a particular stress state. Thus a flow rule describes the relative sizes of individual strain increments, but not their absolute sizes. The flow rule is given mathematical expression by the following equation:

$$\delta\epsilon^p = \delta m \frac{\partial g}{\partial \sigma}. \tag{2.5}$$

In this equation, δm is known as the plastic multiplier (the reader should note that many writers use the symbol $d\lambda$ instead of δm: this usage is not applied here to avoid confusion with the use of λ in critical state soil mechanics). The function g is known as the plastic potential.

The use of a potential function is a natural way of describing a vector quantity which depends only on the location of a point in space. A potential function is a scalar function of position, and taking the partial derivatives of the potential with respect to the co-ordinate axes, a uniquely defined direction is obtained.

The plastic potential $g(\sigma_a, \sigma_b, \sigma_c) = 0$ defines a surface in principal stress space. If vectors representing plastic strain increments are plotted in stress space, then the strain increment vectors are normal to the potential surface (Fig. 2.10).

The form of the plastic potential function for a material could be determined by performing many careful experiments. However, for many materials, the yield function and the plastic potential appear to be the same: $g(\sigma_a, \sigma_b, \sigma_c) = f(\sigma_a, \sigma_b, \sigma_c)$. When $g = f$ it is often said that the condition of 'normality' holds (this is because vectors of plastic strain increment are normal to the yield locus). Alternatively this situation is sometimes described as being one of 'associated' flow, in contrast to the case when g is not equal to f and there is said to be 'non-associated' flow.

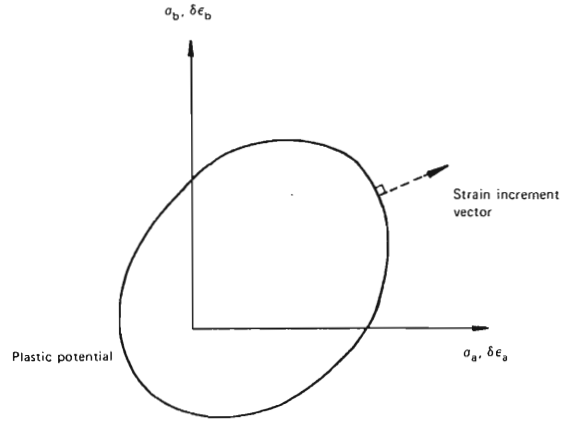


Fig. 2.10 – The plastic potential

Hill (1950) discusses the plastic deformation of metal crystal grains and comments 'It is likely, therefore, that there is a relation, from a statistical average over possible orientations of the grains in a polycrystal, between the

plastic potential g and the function $f(\sigma)$ defining the yield locus. It is not yet known what this should be, theoretically, for any particular metal.

'It seems, however, that the simple relation $g = f$ has an especial place in the mathematical theory of plasticity, for as will be shown later, certain variational principles and uniqueness theorems can then be formulated.'

Although normality ($g = f$) appears to be true for metals, we shall see that there has been some discussion (even controversy) as to whether it can be applied to soils.

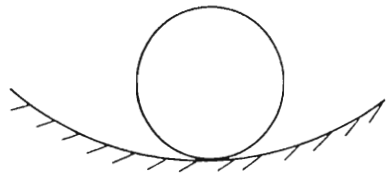
2.4.3 Drucker's stability postulate

Drucker (1950, 1951) introduced a 'postulate of stability' which helps in understanding the physical significance of normality. The concept of stability is a familiar one in the consideration of engineering systems. Consider, for example, the case of a sphere resting on a (possibly non-flat) surface (Fig. 2.11). If the surface is concave upwards and the sphere is subjected to a small perturbing force then the response is stable (when the force is removed, the sphere returns to its original position). If, however, the surface is convex upwards, then the response is unstable. A flat surface gives a response which is 'neutral' in terms of stability. Note that in each case the sphere is initially in equilibrium; however, the stability of the equilibrium is different in each case.

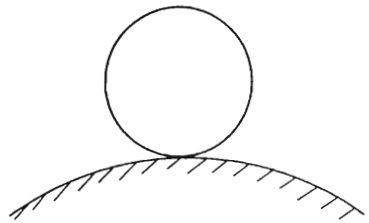
Drucker considers a system which is in equilibrium in some stress state σ and which is then loaded by a small extra increment of load $\delta\sigma$. Drucker regards the incremental stress $\delta\sigma$ as being due to an external agency (i.e. external to the 'system' he is considering). Subsequently $\delta\sigma$ is removed. A stable system is one which absorbs work from the external agency, whereas an unstable system releases work. If the external agency is incapable of absorbing work from the system (for example, if it is supplied by a dead load placed on the system) then the system collapses. Schofield and Wroth (1968) illustrate these concepts in relation to the loads acting on a triaxial test system for soil, and the reader is referred there for a more detailed account. For our purposes it is sufficient to note that Drucker's definition of the stability of equilibrium corresponds to that in use in other branches of engineering (e.g. buckling theory in structures). As engineers we would always prefer to be dealing with stable systems which are capable of absorbing work if we subject them to small disturbing loads.

The plastic work done in a small increment of deformation is approximately $\sigma\delta\epsilon^{\mathrm{p}} + (\delta\sigma\delta\epsilon^{\mathrm{p}})/2$.† Drucker shows that his definition of stability corresponds to a value of $\delta\sigma\delta\epsilon^{\mathrm{p}}$ greater than or equal to zero, so Drucker is concerned with the sign of the second-order work term. In terms of a uniaxial test, stable deformation is equivalent to strain-hardening behaviour, whereas unstable deformation corresponds to strain-softening behaviour (see Fig. 2.12).

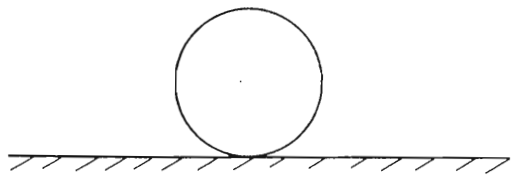
† A consequence of the definitions of stresses and strains given in Chapter 1 is that the mechanical work done (per unit volume of material) is equal to the scalar product of the vectors of stress and incremental strain components.



(a) Stable equilibrium



(b) Unstable equilibrium



(c) Neutral equilibrium

Fig. 2.11 – Stability of equilibrium

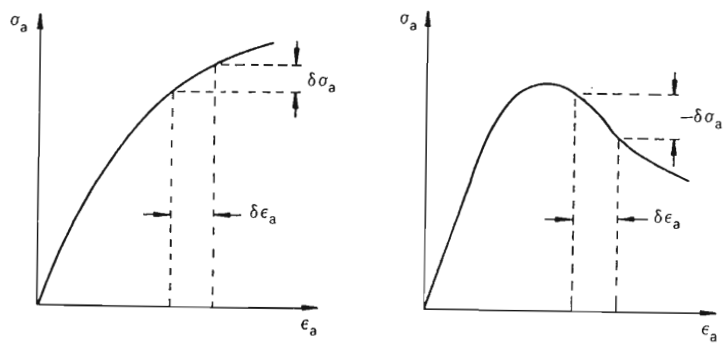


Fig. 2.12 – Stable and unstable responses in a tension test

Why is Drucker's postulate equivalent to normality? Consider a small increment of stress $\delta\sigma$ applied to a plastic material which results in hardening, i.e. a new yield locus is established (Fig. 2.13). In fact this hardening could be

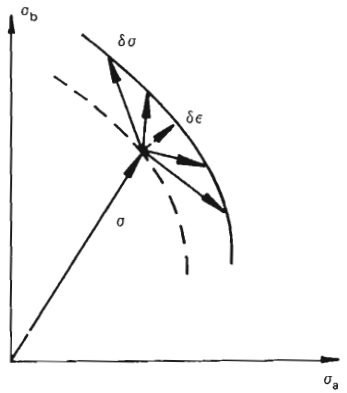


Fig. 2.13 – Drucker's stability postulate

caused by several increments $\delta\sigma$, all starting from the same stress state (and directed outwards from the initial yield locus). The only possible direction of the plastic strain increment vector (satisfying Drucker's postulate) is that normal to the current yield locus. This is because it would otherwise be possible to find a possible $\delta\sigma$ which made an angle of greater than 90° with $\delta\epsilon$.

Drucker introduced his postulate in the context of metal plasticity where strain-hardening behaviour is the norm and systems are generally stable. Some have criticised the application of his postulate to situations (e.g. soils) where strain-softening can occur. We follow Palmer (1973) in asserting that the postulate is basically a classification of material response. In section 2.7.2 we shall examine the implications of Drucker's postulate for soil behaviour.

2.4.4 Frictional systems and plasticity theory

Systems with frictional interfaces have a certain similarity with perfectly-plastic solids. Consider the simple case of a rigid block resting on a plane subject to a horizontal force F and a vertical force N (Fig. 2.14(a)). When $F < \mu N$ there is no movement and the line $F = \mu N$ could be identified as a yield locus for the system. However, if one plots the incremental 'plastic' displacements for this system, it appears that normality does not apply (Fig. 2.14(b)). Drucker (1954) considers some cases of systems made of frictional blocks and concludes that they must be excluded from his definition of stable plastic systems.

Now soil strength is often described by a drained angle of friction. Hence the question immediately arises: is it legitimate to describe soil as a plastic material to which one can apply the principle of normality? Clearly the actual behaviour of a particulate medium such as clay or sand is much more complex than that of a block sliding on a plane. A possible answer to this question could come from performing tests on samples of soil and measuring the plastic strains. If the Mohr–Coulomb surface is taken as an appropriate yield surface (to which normality can be applied) then yielding should be accompanied by a constant

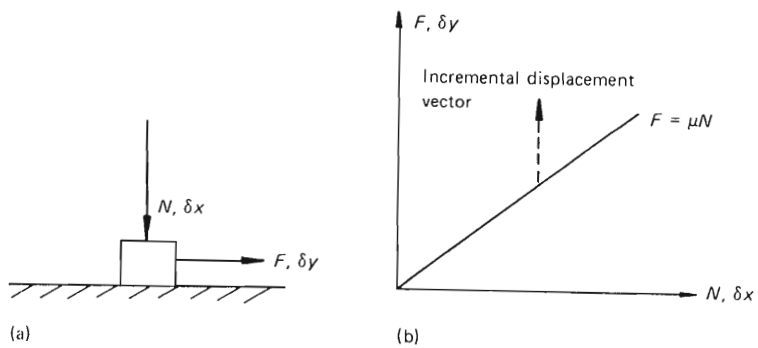


Fig. 2.14 – Lack of normality in a simple system with friction

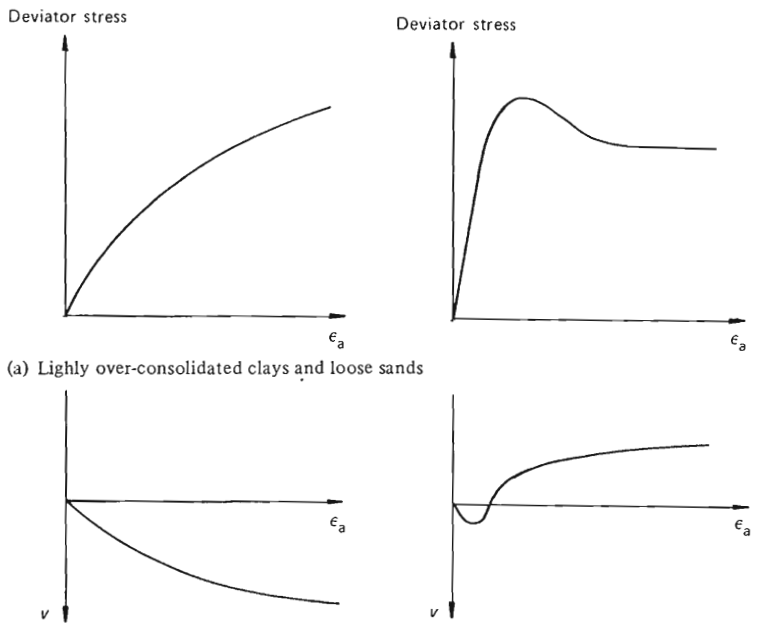


Fig. 2.15 – Typical stress–strain and volumetric strain response of soils when sheared in a triaxial apparatus

rate of negative volumetric strain (i.e. expansion of 'dilation'). In fact soils sometimes compress when they are sheared; sometimes they dilate; and sometimes they deform at constant volume. A typical pattern of behaviour for loose sands or drained tests on lightly over-consolidated clay would be compression during the first part of the test followed by eventual deformation at constant volume (Fig. 2.15(a)). In contrast, dense to medium-dense sands and heavily

over-consolidated clays tend to dilate initially and deform at constant volume later in the test (Fig. 2.15(b)). Therefore, at first sight, it seems that normality cannot be applied to soils. We shall show, however, that this more complex volumetric behaviour of soils can be described by a plastic theory of soil deformation that uses the normality principle.

2.5 CAM-CLAY

Cam-clay is the name given to an elasto-plastic model of soil behaviour. Thus Cam-clay is not a real soil in the sense that one cannot find deposits of it at some location in the ground. However, the Cam-clay equations can be used to describe many real soils if appropriate material parameters are chosen.

This section provides a complete description of Cam-clay. It is intended both as an introduction and as a ready-reference section to contain all the basic equations and definitions. First the symbolic notation used in describing Cam-clay is reviewed. Then the assumptions governing the relationships between volume and applied (isotropic) pressure are described. The critical state concept is then covered. Next the equations which govern plastic yielding are given. Later sections of the chapter show how the Cam-clay equations can be used to predict soil strengths and strains in triaxial tests. For the time being we omit one of the most interesting aspects of Cam-clay: its theoretical derivation. Thus our initial account of Cam-clay is descriptive, and equations are introduced without an attempt at justification. This comes in section 2.7.1.

2.5.1 Critical state parameters

Three parameters, p', q and V, describe the state of a sample of soil during a triaxial test. The parameters are defined:

$$p' = \frac{\sigma'_a + 2\sigma'_r}{3} = \frac{\sigma_a + 2\sigma_r}{3} - u,$$

$$q = \sigma'_a - \sigma'_r = \sigma_a - \sigma_r,$$

V is the specific volume, i.e. the volume of soil containing unit volume of solid material. (N.B. $V = 1 + e$, where e is the voids ratio.)

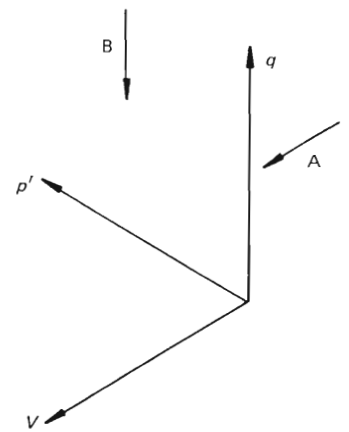
p' is often called the mean normal effective pressure, and q the 'deviator stress'. The reader should note that these three parameters will vary during a test.† The progress of a soil sample during a triaxial test can be represented by a series of points describing a line in a three-dimensional space with axes p', V and q. Different types of test (drained, undrained, compression, extension and so on)

† Unfortunately nearly every book dealing with critical state soil mechanics uses a slightly different notation for the same set of parameters. Schofield and Wroth (1968) use p, q and v. Atkinson and Bransby (1978) use p', q' and v. We use the same notation as Wood (1984).

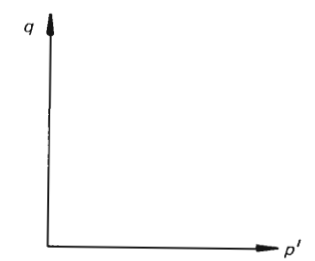
lead to different test paths in this '(p', V, q) space'. Critical state soil mechanics gives us a set of rules for calculating test paths in (p', V, q) space: usually two of (p', V, q) are determined by the type of test and there is a simple procedure for determining the third.

We shall also describe the progress of tests with reference to (p', q) and (p', V) plots. These simply correspond to two orthogonal views of (p', V, q) space (Fig. 2.16). The reader should also note that in the (p', V) plots, the p' axis does not correspond to $V = 0$: instead the V axis is started at a convenient value to illustrate the part of the (p', V) plot which is of interest.

(a) Three-dimensional (p', V, q) space



(b) (p', q) plot (view in direction A)



(c) (p', V) plot (view in direction B)

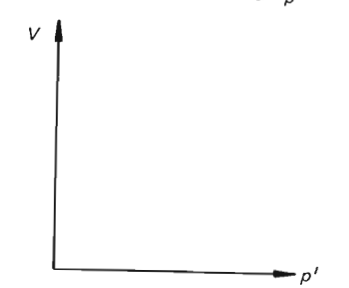


Fig. 2.16 – Two orthogonal views of (p', V, q) space

There are also four parameters which are soil constants: M, Γ, κ and λ. These are introduced below. They describe the fundamental properties of soil with a given mineralogy. Other parameters are defined in terms of the seven already mentioned; for example the stress ratio $\eta = q/p'$.

Corresponding to the stress parameters p' and q are strain parameters v (volumetric strain) and ϵ (deviator strain):

$$v = \epsilon_a + 2\epsilon_r, \tag{2.6}$$

$$\epsilon = \tfrac{2}{3}(\epsilon_a - \epsilon_r). \tag{2.7}$$

v and ϵ describe the strains from the start of the test: we shall often make use of the symbols δv and $\delta\epsilon$ (for strain increments) where

$$\delta v = \delta\epsilon_a + 2\delta\epsilon_r, \tag{2.8}$$

$$\delta\epsilon = \tfrac{2}{3}(\delta\epsilon_a - \delta\epsilon_r). \tag{2.9}$$

The reason for the factor of 2/3 that appears in the definition of shear strain ϵ is so that the work done by a small increment of straining is equal to $p'\delta v + q\delta\epsilon$. Thus the stress and strain parameters correspond to one another in that multiplication leads to the correct evaluation of work done in deformation: the situation is the same as for stress and strain parameters σ_x and ϵ_x, etc. (section 2.4.3). The reader may care to confirm that $p'\delta v + q\delta\epsilon = \sigma'_a\delta\epsilon_a + 2\sigma'_r\delta\epsilon_r$. The formula for work done is valid for drained, partially drained or undrained deformation; see Schofield and Wroth (1968, section 5.6).

2.5.2 Volume–pressure relations

If a sample of soil is subjected to isotropic compression (and swelling) tests, then it follows paths in (p', V) plots as shown in Fig. 2.17. This is basically similar to the more familiar (σ'_v, e) plots obtained from oedometer tests. In critical state theory the virgin compression, swelling and recompression lines are assumed to be straight in ($\ln(p')$, V) plots with slopes $-\lambda$ and $-\kappa$ respectively, as shown in Fig. 2.18. The equation of the isotropic virgin compression line (often called the isotropic normal consolidation line) is

$$V = N - \lambda \ln(p') \tag{2.10}$$

where N is a constant for a particular soil. N is the value of V when $\ln(p') = 0$, i.e. $p' = 1$: clearly the value of N depends on the units which are used to measure pressure. The units adopted here are kN/m^2, sometimes called kPa (kilopascals). Although N is a soil constant, it is related to those already defined ($N = \Gamma + \lambda - \kappa$): this is demonstrated below. The equation of a swelling or recompression line is given by

$$V = V_\kappa - \kappa \ln(p'). \tag{2.11}$$

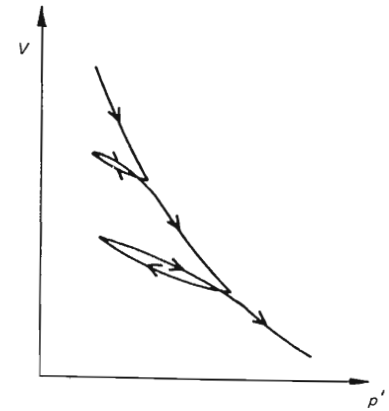


Fig. 2.17 – Typical (p', V) plot of isotropic compression, swelling and recompression

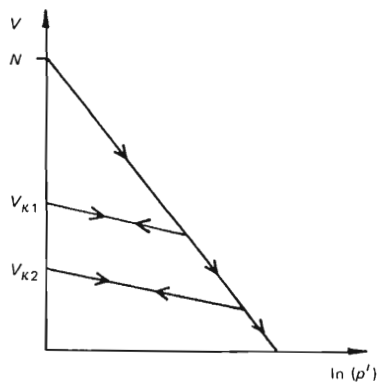


Fig. 2.18 – Idealised ($\ln p'$, V) plots in critical state theory

When moving up or down one of these 'κ-lines' the soil is over-consolidated. Equation (2.11) is sometimes written as

$$V_\kappa = V + \kappa \ln(p'). \tag{2.12}$$

The value of V_κ depends upon which κ-line the soil is on, but it stays constant while the soil is moving up or down the same line.

It is convenient here to introduce the parameter V_λ. The definition of V_λ is similar to that of V_κ:

$$V_\lambda = V + \lambda \ln(p'). \tag{2.13}$$

We have already encountered one particular λ-line, the isotropic normal consolidation line, when $V_\lambda = N$. Note that if V and p' are specified, then V_κ and V_λ can always be determined using (2.12) and (2.13). Conversely, if V_κ and

V_λ are known then it is always possible to deduce V and p' (see Fig. 2.19). Thus V_κ and V_λ can be regarded as a set of parameters describing the soil, which are an alternative to V and p'.

It is worth noting that for very large effective pressures, (2.10) predicts values of V less than 1 (a physical impossibility). Clearly this equation represents an approximation to soil behaviour which is valid in the range of stresses of engineering interest.

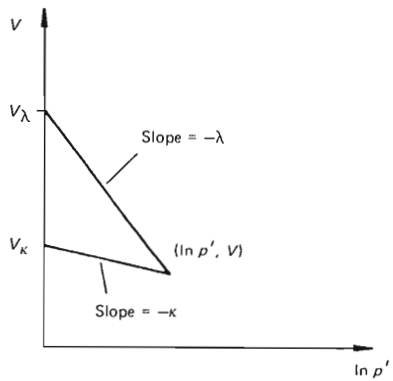


Fig. 2.19 – Each point in a ($\ln p'$, V) plot is uniquely associated with a pair of values (V_κ, V_λ) (and vice versa)

2.5.3 Critical state line

When soil samples are sheared they approach the Critical State Line (CSL).† The equations of the CSL are

$$q = Mp', \tag{2.14}$$

$$V = \Gamma - \lambda \ln (p'). \tag{2.15}$$

M and Γ are constants for a particular soil. They determine the slope of the CSL in a (p', q) plot and the location of the CSL in the (p', V) plot, respectively.‡ Figs. 2.20(a) and 2.20(b) show the CSL in (p', q) and (p', V) plots. Note that (2.15) is the equation of a λ-line with $V_\lambda = \Gamma$. The critical state line represents the final state of soil samples in triaxial tests when it is possible to continue to shear the sample with no change in imposed stresses or volume of the soil. Hence, at the critical state:

† Strictly speaking this statement is true only when the effective stress path obeys the relationship $\delta q/\delta p' > M$ or $\delta q/\delta p' < -M$. However, this condition applies in all normal triaxial tests where one is shearing the sample to failure.

‡ Of course, many people pronounce M as the capital English (rather than Greek) letter. The reason for this (at first perhaps surprising) convention is that M represents a frictional constant for Cam-clay, and 'μ' is used widely in mechanics to signify a coefficient of friction.

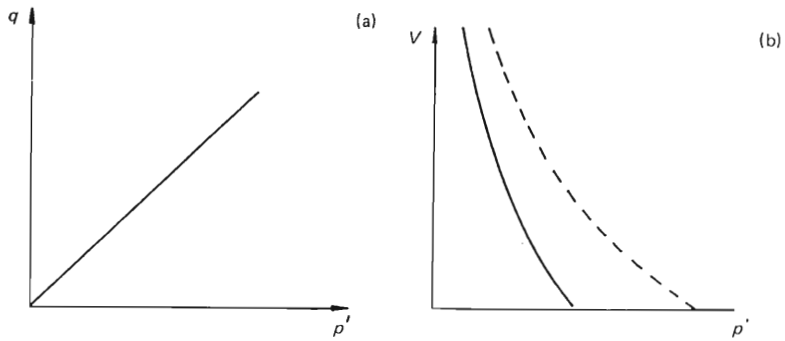


Fig. 2.20 – The critical state line in (a) (p', q) plot and (b) (p', V) plot (isotropic normal compression line is shown dashed in (b))

$$\frac{\delta v}{\delta \epsilon} = 0; \quad \frac{\delta q}{\delta \epsilon} = 0; \quad \frac{\delta p'}{\delta \epsilon} = 0.$$

(2.14) and (2.15) describe a curved line in three-dimensional (p', V, q) space (Fig. 2.21).

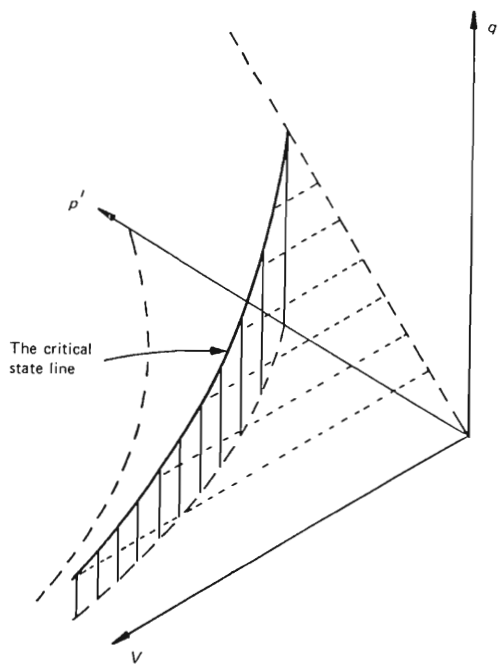


Fig. 2.21 – The critical state line in (p', V, q) space is given by the intersection of of two planes: $q = Mp'$ and a curved vertical plane $V = \Gamma - \lambda \ln (p')$

2.5.4 Yielding of Cam-clay

First consider the $(\ln(p'), V)$ plot in Fig. 2.18 rotated anti-clockwise through an angle of 90° (Fig. 2.22). This picture is basically the same as that for a linear work-hardening metal (Fig. 2.3(b)). However, a significant difference is apparent when comparing soils with metals. With soils we are seeing elasto-plastic behaviour associated with volumetric strains. The von Mises and Tresca yield functions for metal suggest that one can hydrostatically compress metals indefinitely without yielding taking place.

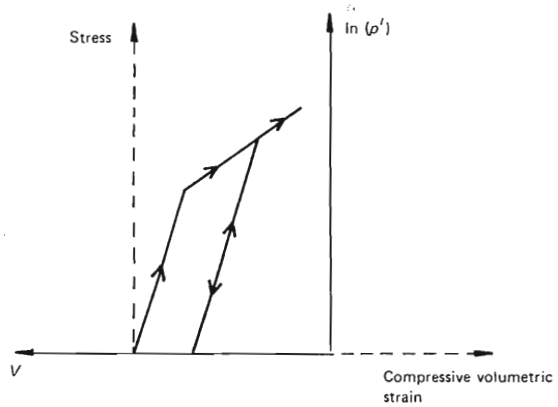


Fig. 2.22 – Volumetric straining of soils viewed as strain-hardening plastic behaviour

The next part of our description of the yielding of soils considers the effect of shearing a sample. Suppose that the state of the soil can initially be represented by the point A in a (p', V) plot (Fig. 2.23). The deviator stress, q, is now increased while p' and V remain constant. Subsequently we shall see that this is what happens to an over-consolidated sample in an undrained triaxial test. As the test proceeds, the state of the sample can be represented by a point in the three-dimensional (p', V, q) space which lies directly above the original point (Fig. 2.24). The sample yields at a point such as B when the value of q is given by the following equation:

$$q = \frac{Mp'}{(\lambda - \kappa)} (\Gamma + \lambda - \kappa - V - \lambda \ln(p')). \tag{2.16}$$

(2.16) describes a surface in (p', V, q) space. Fig. 2.25 shows an isometric view of this surface. When the state of a specimen of soil can be represented by a point below the surface, then soil behaviour is elastic. Soil states on the surface indicate yielding, and it is impossible for soil samples to exist in states equivalent to points above the surface. For this reason the surface is known as the Stable State Boundary Surface (SSBS). Another way of writing (2.16) is

$$V_\lambda = \Gamma + (\lambda - \kappa)(1 - \eta/M). \tag{2.17}$$

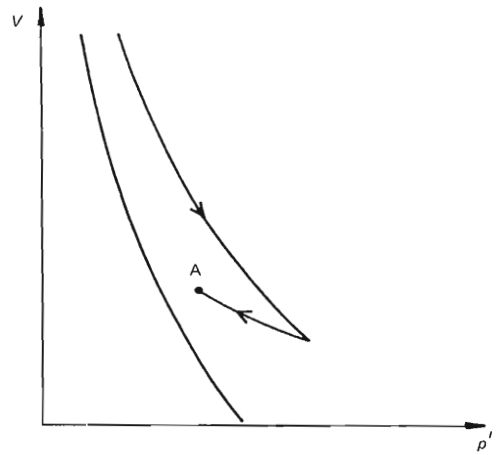


Fig. 2.23 – Preparation of a soil sample by isotropic normal consolidation and then swelling

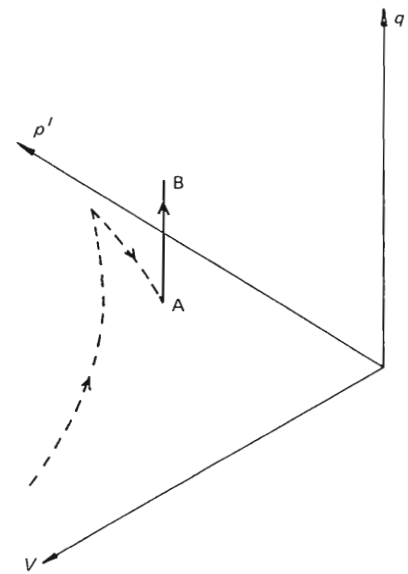


Fig. 2.24 – The yielding of a sample in (p', V, q) space. Sample preparation follows the dashed lines which lie in the $q = 0$ plane. Progress towards yielding is then along the vertical path AB which is parallel to the q axis.

(2.17) is probably the most useful form of the equation. Note that when η is set to zero we recover the equation of the isotropic normal consolidation line (EF in Fig. 2.25). If (2.14) is substituted into (2.17) then (2.15) is obtained. On the

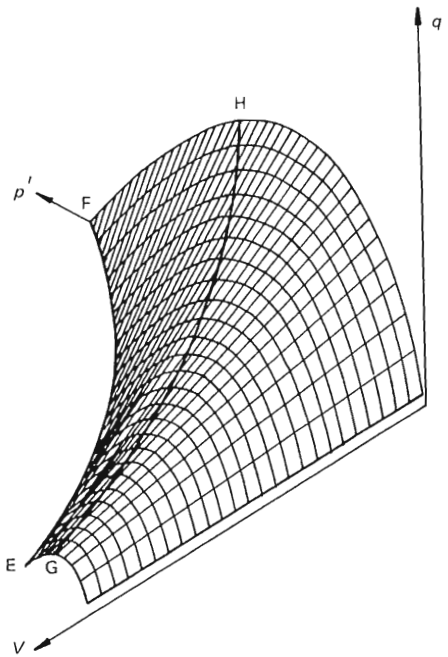


Fig. 2.25 – The stable state boundary surface in (p', V, q) space

other hand, if (2.15) is substituted into (2.17) then (2.14) is obtained. This demonstrates that the CSL lies on the SSBS (GH in Fig. 2.25).

Although either (2.16) or (2.17) describes the combination of stresses that causes yielding, neither is the equation of a yield surface in the sense introduced in section 2.3.3. The reason for this is that V appears in both equations. The equation of a yield surface should be in terms of the current stresses together with a hardening parameter to fix the size. V is unable to fulfil the role of a hardening parameter because it changes for elastic stress increments inside a yield locus.

Elastic straining underneath the SSBS corresponds to movement along a κ-line, with a corresponding change in V. Thus when an elastic sample is brought to the point of yield it must simultaneously lie both on the κ-line and on the SSBS. Therefore the intersection of the SSBS with the κ-line equation gives the current yield surface:

$$q = M p' \ln (p_c'/p'). \tag{2.18}$$

The form of this yield function is shown in Fig. 2.26. As we have mentioned above, elastic straining is governed by the κ-line equation, and thus in terms of (p', V, q) space the state of the material must remain on an 'elastic wall' (Fig. 2.27). The 'point' of the yield locus lies on the isotropic normal consolidation line. p_c' is the isotropic pre-consolidation pressure for a soil sample lying on a particular κ-line (Fig. 2.28).

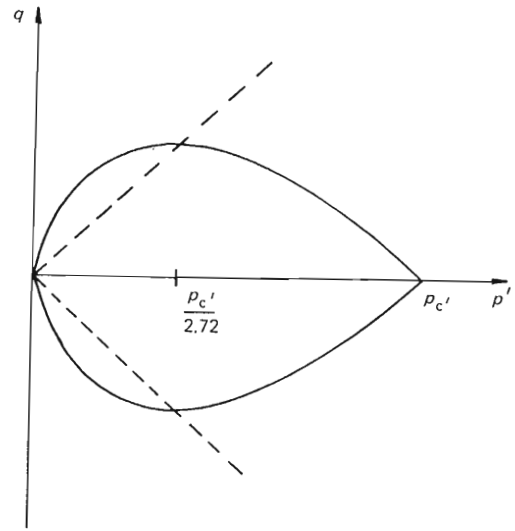


Fig. 2.26 – The Cam-clay yield locus (the yield locus is assumed to be symmetric about the p' axis)

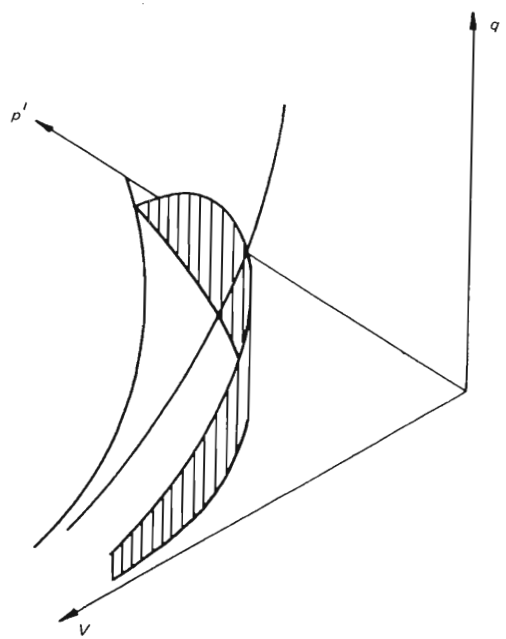


Fig. 2.27 – Isometric view of an elastic wall

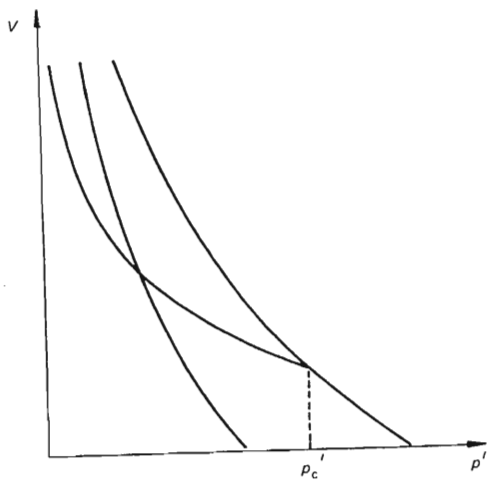


Fig. 2.28 – The size of the Cam-clay yield locus is determined by p_c', the isotropic consolidation pressure

2.5.5 Strains

Total volumetric and shear strains can be expressed as the sum of elastic and plastic components:

$$v = v^e + v^p, \tag{2.19}$$

$$\epsilon = \epsilon^e + \epsilon^p, \tag{2.20}$$

and a similar pair of equations is valid for incremental strains:

$$\delta v = \delta v^e + \delta v^p, \tag{2.21}$$

$$\delta \epsilon = \delta \epsilon^e + \delta \epsilon^p. \tag{2.22}$$

Cam-clay corresponds to the following assumptions about elastic and plastic strains:

Elastic strains

δv^e is calculated from the κ-line equation

$$\delta \epsilon^e = 0.$$

Plastic strains

$$\delta v^p = \delta V_\kappa / V$$

$\delta \epsilon^p$ is calculated from the flow rule: $\delta v^p / \delta \epsilon^p = M - \eta$.

2.6 TRIAXIAL TESTS ON CAM-CLAY

The equations of the previous section can be used to predict stress paths, shear strengths and strains in triaxial tests.

2.6.1 Preparing the sample

In each of the following examples the triaxial test sample is prepared by isotropic normal consolidation to $p' = p_c'$, followed by swelling to $p' = p_0'$. Fig. 2.29 shows the path followed by the specimen in a (p', V) plot. The value of V at the start of the test, V_0, can be calculated from the equations of the isotropic NCL and the κ-line as follows:

$$V_c = N - \lambda \ln (p_c'),$$

$$V_\kappa = V_c + \kappa \ln (p_c') = V_0 + \kappa \ln (p_0');$$

hence

$$V_0 = N - \lambda \ln (p_c') + \kappa \ln (p_c'/p_0'). \tag{2.23}$$

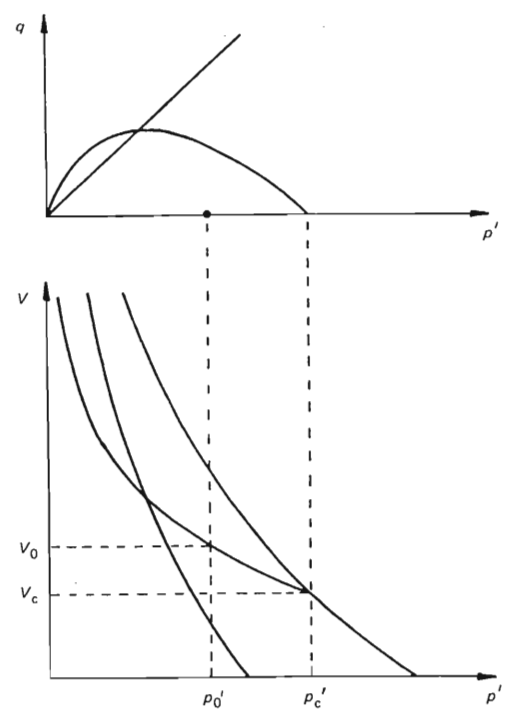


Fig. 2.29 – Preparing the sample by isotropic normal consolidation and swelling establishes the initial yield locus of size p_c'

In a (p', q) plot, this establishes the initial stress state as inside a yield locus which intersects the p' axis at $p' = p_c'$ (Fig. 2.29). In fact this sample preparation procedure has been described previously (but without the equations): see Fig. 2.24 for the view in (p', V, q) space.

2.6.2 Drained compression tests

In a standard drained compression test the cell pressure σ_r remains constant and the axial stress σ_a is increased. In this example it is assumed that the pore pressure is maintained at a back pressure of zero (i.e. atmospheric). Thus the Effective Stress Path (ESP) always corresponds with the Total Stress Path (TSP) (since $p' = p$), and the ESP can be determined by considering the total stresses acting on the soil sample. On the other hand, if a constant back pressure were maintained, then there would always be a constant horizontal offset u between the total and effective stress paths. The initial state of the soil in a (p', q) plot is $(p_0', 0)$. At a later point in the test, $\sigma_r = p_0'$ and $\sigma_a = p_0' + x$ (say), so the soil sample can now be represented by the point $((p_0' + x/3), x)$. Thus the ESP for the test is a line of slope 3 starting from $(p_0', 0)$ (see Fig. 2.30). During the initial part of the test, before the ESP intersects the current yield locus at B (see the (p', q) plot in Fig. 2.31), the soil behaviour is elastic. After point B the soil is yielding and each stress state on BF is associated with a new (enlarged) yield locus. Finally, the soil fails when the ESP intersects the CSL (point F in Fig. 2.31). Note that the yield locus at failure, intersecting the p' axis at H, corresponds to the κ-line intersecting the isotropic NCL at point H in the (p', V) plot. If one knows the critical state parameters for the soil then it is straightforward to calculate the value of p' and q at failure from the intersection of the ESP and the CSL:

$$q = 3p' - 3p_0'$$

$$q = \mathrm{M}\, p',$$

giving $p' = 3p_0'/(3 - \mathrm{M})$ and $q = 3\mathrm{M}p_0'/(3 - \mathrm{M})$.

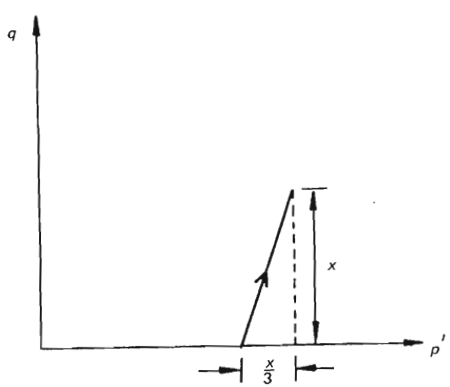


Fig. 2.30 – Drained ESP for a compression test

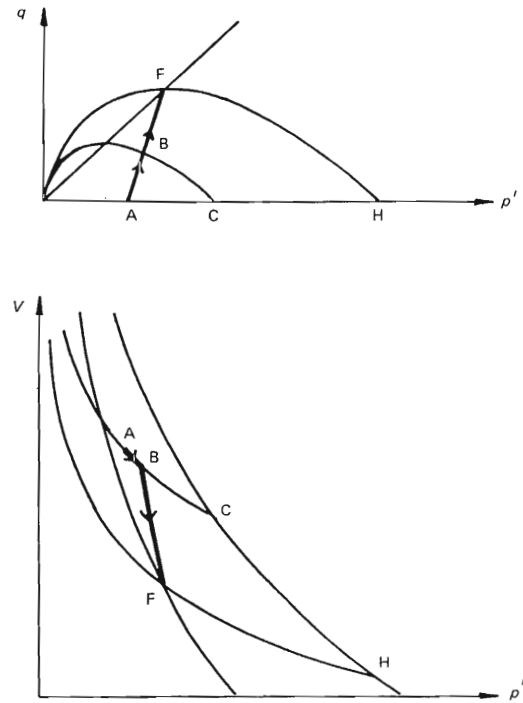


Fig. 2.31 – Drained compression test on Cam-clay (over-consolidation ratio $R = 2$)

In the (p', V) plot in Fig. 2.31 the soil follows the κ-line while it is elastic (until point B) and then changes direction to move to failure on the CSL at point F. Each κ-line that the soil crosses corresponds to a yield locus in t' (p', q) plot, although Fig. 2.31 only shows the first and last of these. Since t value of p' at failure is known, the value of V can be found from (2.15). Hence the volumetric strain to failure can be calculated as $(V - V_0)/V_0$.

Now consider a test on a sample which has a higher over-consolidation ratio $(R = p_c'/p_0')$ so that its initial state A in the (p', V) plot is on the left-hand side of the CSL in a (p', V) plot. The progress of this sample in a drained compression test is shown in Fig. 2.32. Note that although the ESP appears to intersect the CSL in the (p', q) plot before yielding, in fact it is missing the CSL in the three-dimensional (p', V, q) space, as is made clear by examination of the test path in the (p', V) plot (Fig. 2.32). After yielding, the state of the sample moves back down the ESP to point F on the CSL. This is accompanied by the yield loci 'shrinking' rather than 'growing', as was the case for the sample considered earlier.

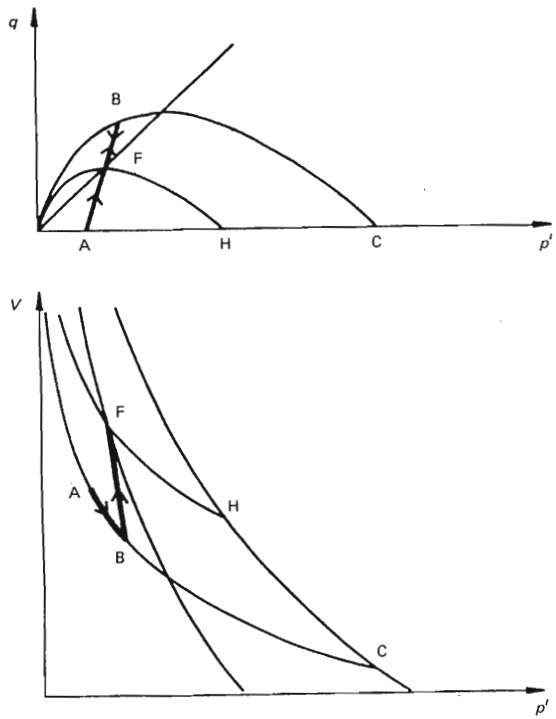


Fig. 2.32 – Drained compression test on a sample of Cam-clay with $R = 7$

2.6.3 Calculation of strains in drained tests

In this section the procedures for calculating the strains in a drained triaxial test are set out as a series of itemised steps. Although we have so far only considered the standard compression test, these steps can be used to calculate the strains in other kinds of test. Basically, the strains are calculated for a number of increments of stress once the sample has yielded. Although there are a few situations where it is possible to obtain an analytical expression for the stress–strain curve (e.g. constant p' tests), in general the procedure described here will be required.

1. Establish starting values of p', q, V and p'_c.
2. Calculate values of p' and q when yielding starts. This involves finding the intersection point of the drained effective stress path (ESP) and the current yield locus. In general one has to solve a non-linear equation because of the nature of the yield function. However, this can be done fairly quickly by hand by substituting a few values of p' into both the yield locus and the ESP equation until the values of q are close.

3. Calculate the (elastic) volumetric strain up to this point. Since elastic shear strains are zero, $\delta\epsilon_a = \delta\epsilon_r$, and hence $\delta\epsilon_a = \delta v/3$.
4. Divide the ESP between the point of first yielding and the intersection with the CSL into a number of equal increments (say n). Then repeat the following steps for values of i from 1 to n.
5. Calculate the volumetric strain in increment i from the values of p' and q at the start and end of the increment. (Values of V can be obtained from the equation of the SSBS.)
6. Calculate the elastic volumetric strain for this increment from the κ-line equation.
7. Calculate the plastic volumetric strain for this increment by subtracting the elastic strain calculated in 6. from the strain calculated in 5.
8. Calculate the shear strain for this increment from the plastic volumetric strain and the Cam-clay flow rule (use values of p' and q corresponding to the start of the increment).
9. Use the shear strain obtained in 8. and the volumetric strain obtained in 5. together with the basic definitions of these strains to calculate $\delta\epsilon_a$ and $\delta\epsilon_r$.
10. Add $\delta\epsilon_a$ to values calculated for previous increments to obtain a point on the q versus ϵ_a plot.

Fig. 2.33 contrasts the behaviour of the two samples that were considered in the previous section. The first strain-hardened after yielding (q increased) and exhibited compressive plastic volumetric strains. The second strain-softened (q decreased) and exhibited expansive volumetric strains. Note the similarity of these results with the experimental behaviour shown in Fig. 2.15.

2.6.4 Undrained compression tests

Now we consider the behaviour of a sample of Cam-clay in an undrained compression test. The total stress path for this test is identical to the total stress path for the drained case (because the total stress path is specified by the total stresses applied to the soil). During the whole of the undrained test, the specific volume must remain constant since no water is allowed to flow into or out of the soil. Although the total volumetric strain must be zero, elastic and plastic components of the strain can be non-zero as long as

$$v^p + v^e = 0. \tag{2.24}$$

Before the sample yields, the plastic volumetric strain v^p must be zero and therefore the elastic volumetric strain must also be zero. If the elastic volumetric strain is zero then there can be no change in p'. In other words, the effective stress path in the (p', q) plot must be parallel to the q axis. Thus in the three-dimensional (p', V, q) space, the test path will be vertical before yield takes place. When the sample does yield, equal (and opposite in sign) values of v^p and

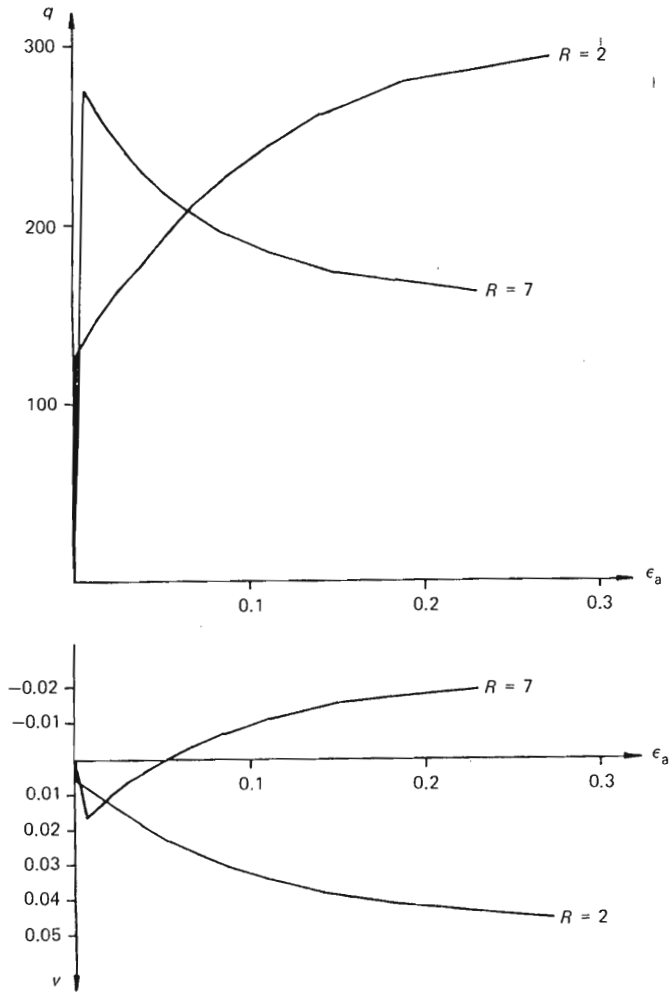


Fig. 2.33 – Stress–strain response for drained tests

v^e are possible and the test path now follows the constant V cross-section of the SSBS until the sample reaches the CSL (Fig. 2.34).

The final point of the test (and hence the soil's undrained strength) can be calculated by substituting the value of V_0 from (2.23) into the critical state line equation (2.15). Thus:

$$p'_f = \exp((\Gamma - V_0)/\lambda), \tag{2.25}$$

and

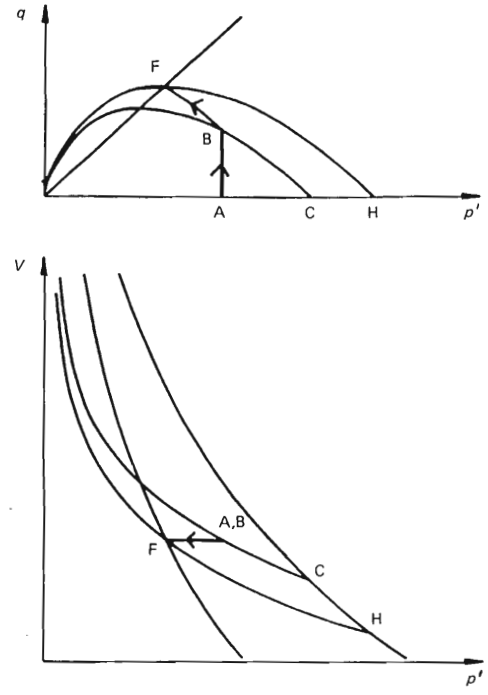


Fig. 2.34 – Undrained compression test on Cam-clay ($R = 1.5$)

$$c_u = (1/2)q_f = (1/2)Mp'_f = (1/2)M \exp((\Gamma - V_0)/\lambda). \tag{2.26}$$

The pore pressure at the end of the test is given by

$$u_f = p'_0 + q_f/3 - p'_f, \tag{2.27}$$

whereas the pore pressure at yield is given by

$$u_y = q_y/3. \tag{2.28}$$

Now consider a soil sample which is more heavily over-consolidated, starting at a point in a (p', V) plot such as that shown in Fig. 2.35. This sample also has an initial ESP which is vertical (following the same argument as before). Again, on yielding, the sample moves over the constant V cross-section of the SSBS until the CSL is reached. Initially the sample appears to strain-harden (q increases) but towards the end of the test it strain-softens (q decreases). However, both the strain-hardening and the strain-softening are associated with a decrease in the size of the yield locus.

Note that the isometric view of the SSBS shown in Fig. 2.25 was made up of constant V-lines and constant p'-lines. Each constant V-line includes (the yielding) part of the undrained ESP for samples starting at that value of V.

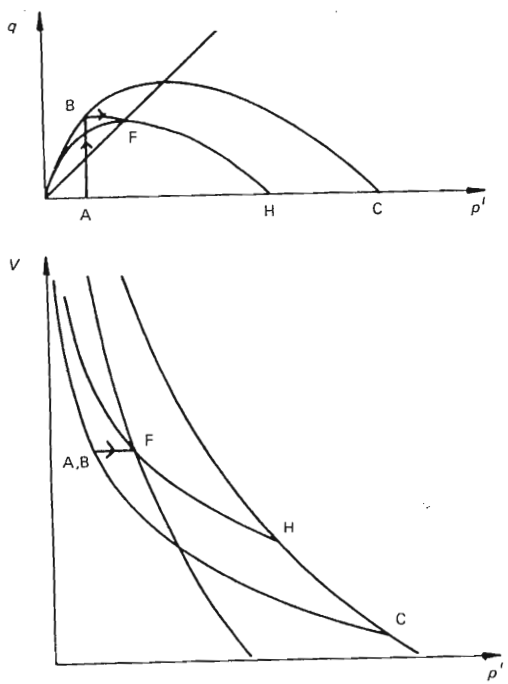


Fig. 2.35 – Undrained compression test on Cam-clay ($R = 8$)

Suppose that a sample is initially normally consolidated to a pressure p'_e. Then the initial volume is given by

$$V_e = \Gamma + \lambda - \kappa - \lambda \ln (p'_e).$$

Substituting this value of V_e into the equation of the SSBS (2.16), the following equation is obtained:

$$q = \frac{Mp'}{(1 - \kappa/\lambda)} \ln (p'_e/p'). \tag{2.29}$$

(2.29) is the equation of the undrained ESP for a sample initially normally consolidated to a pressure p'_e. Over-consolidated samples at the same initial volume $V_0 = V_e$ have vertical ESPs until they intersect this line, after which they follow the same route to the critical state (Fig. 2.36).

Note that the undrained ESP has the same basic equation as the yield locus (2.18), except that M in (2.18) has been replaced by $M/(1 - \kappa/\lambda)$ in (2.29) and p'_c in (2.18) has been replaced by p'_e. In fact the role of p'_e or p'_c is to fix the size of the undrained locus or yield locus respectively, and so the effect of the factor $1/(1 - \kappa/\lambda)$ is to 'stretch' the yield locus in the direction of the q axis to

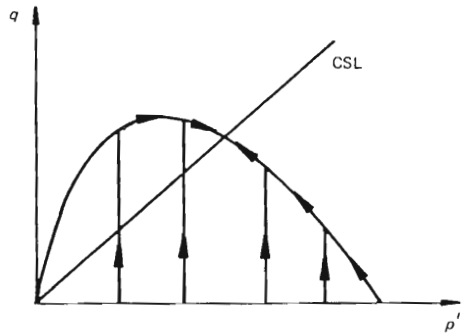


Fig. 2.36 – The undrained surface for Cam-clay

form the undrained ESP equation. The fact $1 - \kappa/\lambda$ occurs often when undrained tests are considered and so some writers (notably Wroth, 1984) have used the symbol Λ for this ratio.

The Cam-clay model gives an elegant account of the effect of over-consolidation on undrained shear strength. Consider a specimen which is normally consolidated to p'_c, then allowed to swell back to an isotropic pressure of p'_0, giving an over-consolidation ratio, $R = p'_c/p'_0$. Then from (2.23), the initial volume V_0 is given by

$$V_0 = \Gamma + \lambda - \kappa - \lambda \ln (p'_c) + \kappa \ln (R).$$

V_0 will remain the same during the test and so we can set this expression equal to $\Gamma - \lambda \ln (p'_f)$, where p'_f is the value of p' at the end of the test. Hence (after some manipulation):

$$p'_f = p'_0 R^{\Lambda} \exp (-\Lambda)$$

and the undrained shear strength c_u is given by

$$c_u = (1/2)q_f = (1/2)Mp'_f = (1/2)M p'_0 R^{\Lambda} \exp (-\Lambda). \tag{2.30}$$

(In fact we have just taken (2.26) one stage further by substituting in the appropriate value of V_0.) When $R = 1$, (2.30) gives the shear strength for a normally consolidated sample, so the effect of over-consolidation is expressed in the factor R^{Λ}. The experimental data of Ladd *et al.* (1977) support this basic relationship (see also Wroth, 1984).

It is also possible to obtain an expression for Skempton's pore pressure parameter as a function of the over-consolidation ratio. Substituting (2.29) and $q_f = Mp'_f$ into (2.27), the following equation for Skempton's pore pressure parameter, A, at failure is then obtained:

$$A_f = \frac{1}{3} - \frac{1}{M} + \frac{R^{-\Lambda}}{M} \exp (\Lambda). \tag{2.31}$$

2.6.5 Calculation of strains in undrained tests

1. Establish starting values of p', q, V and p'_c.
2. Calculate the value of q when yielding starts from the equation of the current yield locus. (The undrained ESP is vertical inside the yield locus.)
3. Note that both elastic shear strains are zero (by definition) and elastic volumetric strains are zero (because undrained). Hence ϵ_a and ϵ_r are also zero.
4. Divide the horizontal distance between the initial point and the critical state line in the (p', V) plot into a number of equal increments (say n). Then repeat the following steps for values of i from 1 to n.
5. Calculate the values of q at the end of the increment from the equation of the SSBS.
6. Calculate the elastic volumetric strain for this increment from the κ-line equation.
7. The plastic volumetric strain for this increment is equal to minus the elastic strain calculated in 6. (The overall volumetric strain increment is zero because of undrained behaviour.)
8. Calculate the shear strain for this increment from the plastic volumetric strain and the Cam-clay flow rule (use values of p' and q corresponding to the start of the increment).
9. Use the shear strain obtained in 8. to calculate $\delta\epsilon_a$ and $\delta\epsilon_r$ (using the fact that the volumetric strain is zero).
10. Add $\delta\epsilon_a$ to values calculated for previous increments to obtain a point on the q versus ϵ_a plot.

Fig. 2.37 shows plots of q and pore pressure versus ϵ_a for the two tests considered earlier. Note that although the pore pressure increases linearly with q during the initial (elastic) part of each test, following yield the behaviour is different, with the first specimen tending to generate positive pore pressures and the second negative pore pressures. The first test exhibits q increasing before failure, while the second ends with q decreasing.

2.6.6 Other types of triaxial test

The calculations described above for compression tests can easily be extended to other types of triaxial test (e.g. extension, constant p', etc.). In drained tests, one simply has a total stress path (equivalent to the ESP) inclined at some other angle in the (p', q) plot, and it is a matter of simple geometry to calculate the intersection of the ESP with the CSL and the current yield locus. In undrained tests, although the total stress path will differ, the effective stress path remains the same. The calculation of the pore pressure in a test is again just a geometric exercise.

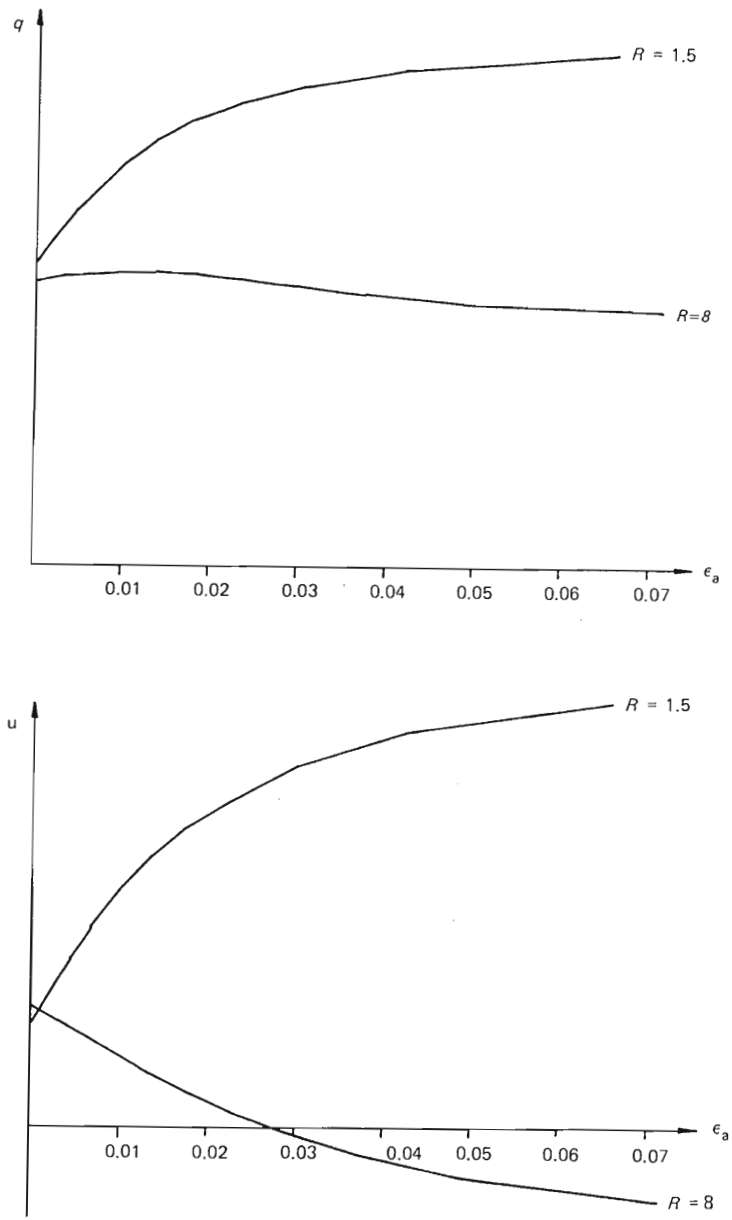


Fig. 2.37 – Stress–strain response for undrained tests

2.7 COMMENTS ON CAM-CLAY

The general concept of using a hardening plasticity model to describe the stress–strain behaviour of soils was first proposed by Drucker *et al.* (1957). Essentially, Drucker *et al.* suggested putting a spherical 'cap' on the 'Drucker–Prager cone'. The cap could be enlarged (accompanied by a smaller enlargement of the cone) by hydrostatic loading of the soil. Their paper speculates about what happens to the cap on elastic unloading and during triaxial tests, but makes no firm proposals. The paper expresses doubts as to whether normality should be applied to the 'frictional' yielding on the cone (compare section 2.4.4). In constructing the critical state models, the Cambridge group took up some of the proposals of Drucker *et al.*, and discarded others. In doing so they managed to produce a model of soil behaviour which is 'simple' in the sense that the model is derived from a small number of basic assumptions, yet the model manages to reproduce for the first time an appropriate description of volumetric response under shear. What really sets the critical state models apart from other attempts to formulate elasto-plastic models for soils is the critical state line in the (p', V) plot. This allows a consistent and realistic treatment of both drained and undrained tests. Although the Cam-clay model was at first just proposed for stress ratios less than M, Schofield and Wroth (1968) extend the proposal for stress ratios greater than M (as we have done earlier in this chapter). However, they advance good reasons why the predictions of the model may not be as good in this region (see section 2.7.4).

2.7.1 Derivation of Cam-clay

Cam-clay is based on the following assumptions.

(a) The isotropic normal consolidation line has an equation:

$$V = \Gamma + \lambda - \kappa - \lambda \ln (p'),$$

and isotropic swelling and recompression lines have equations:

$$V = V_\kappa - \kappa \ln (p'). \qquad (2.11\ bis)$$

Γ, λ and κ are soil constants.

(b) Elastic volumetric strains for Cam-clay are given by the κ-line equation. Elastic shear strains are zero (this is equivalent to taking an infinite value for the elastic shear modulus, G).

(c) When Cam-clay is yielding, the plastic work done is given by $Mp'\delta\epsilon^p$. Thus:

$$p'\delta v^p + q\delta\epsilon^p = Mp'\delta\epsilon^p. \qquad (2.32)$$

(d) (2.32) represents a flow rule. Normality can be applied to this relation to give the equation of the Cam-clay yield locus.

(e) The size of the Cam-clay yield locus is fixed by specifing that the intersection of the yield locus with the p' axis corresponds to the isotropic normal consolidation line.

Although there are experimental data supporting (c) (Roscoe *et al.*, 1963), there is also a strong physical intuition about the nature of the deformation of soil underlying (2.32). According to Schofield and Wroth (1968):

'Consider a random aggregate of irregular "solid" particles of diverse sizes which tear, rub, scratch, chip and even bounce against each other during the process of continuous deformation. If the motion were viewed at close range we could see a stochastic process of random movements, but we keep our distance and see a continuous flow. At close range we would expect to find many complicated causes of power dissipation and some damage to particles; however, we stand back from the small details and loosely describe the whole process of power dissipation as "friction", neglecting the possibilities of degradation or of orientation of particles.'

(2.32) is rearranged:

$$\frac{\delta v^p}{\delta \epsilon^p} = M - \frac{q}{p'}. \qquad (2.33)$$

From the condition of normality, the direction of the incremental plastic strain vector specified by this equation must intersect the yield locus at a right angle. Hence:

$$\frac{\delta \epsilon^p}{\delta v^p} \cdot \frac{\delta q}{\delta p'} = -1. \qquad (2.34)$$

Combining (2.33) and (2.34), and taking the limit as $\delta p'$ and $\delta q \to 0$, a differential equation is obtained:

$$\frac{dq}{dp'} = -M + \frac{q}{p'}. \qquad (2.35)$$

(2.35) is integrated to obtain the equation of the yield locus. What follows is just mathematical manipulation: substitute $\eta = q/p'$ and use the relation

$$\frac{d\eta}{dp'} = \left(p' \frac{dq}{dp'} - q\right) \Big/ (p'^2) \qquad (2.36)$$

to substitute for dq/dp', to obtain an equation $p'(d\eta/dp') = -M$ which can be directly integrated. ((2.36) comes from the standard rule for differentiating a quotient.) The resulting equation is $\eta = q/p' = -M \ln (p') + c$, where c is a constant of integration. The constant of integration is determined using (e) above; thus when $q/p' = 0$, $p' = p'_c$, and the Cam-clay yield locus is arrived at:

$$q = M p' \ln (p'_c/p'). \qquad (2.18\ bis)$$

The equation of the SSBS is obtained as follows: consider a sample of Cam-clay

which is yielding; then the current values of p' and q must satisfy the equation of the yield locus. The current value of specific volume, V, is given by

$$V = \Gamma + \lambda - \kappa - \lambda \ln (p'_c) + \kappa \ln (p'_c/p'). \tag{2.37}$$

This equation follows exactly the same reasoning as in section 2.6.1. The next step is to eliminate p'_c between (2.18) and (2.37), and the result is the equation of the SSBS.

$$q = \frac{Mp'}{(\lambda - \kappa)} (\Gamma + \lambda - \kappa - V - \lambda \ln (p')), \tag{2.16 \textit{bis}}$$

or alternatively (the preferred form):

$$V_\lambda = \Gamma + (\lambda - \kappa)(1 - \eta/M). \tag{2.17 \textit{bis}}$$

Note that the equations of the critical state line have not been used anywhere in the derivation of any of the equations of this section. The assumptions can basically be boiled down to two statements:

1. The work done in plastic deformation is $Mp'\delta\epsilon^p$, which gives the flow rule and by integration the yield locus.
2. Elastic strains inside the yield locus correspond to movement on a κ-line. The size of the yield locus is fixed by the isotropic normal consolidation pressure p'_c (given a convenient visual interpretation as the yield locus 'sitting on top of' a κ-line in (p', V, q) space).

From the point of view of the theory of plasticity, 1. is the yield function and 2. is the hardening law. Both assumptions can be varied to produce slightly different (but basically similar) models.

When the rules for calculating strains (from plasticity theory) are applied to triaxial samples of Cam-clay, the samples end up in a condition defined by the critical state line equations, deforming at constant volume with no change in stress. This point is sometimes disguised by the way that critical state soil mechanics is taught, where the equations of the critical state line are described first (and therefore appear to be basic assumptions in the theory). Although this is probably the best way of explaining the theory to initiates, it has the unfortunate side-effect of hiding the small number of assumptions which are actually needed to produce a sophisticated description of soil behaviour.

Of course in practice the critical state line was 'discovered' first (Roscoe *et al.*, 1958). From the present point of view it can be regarded as a theoretical consequence of the Cam-clay assumptions (Roscoe and Schofield, 1963).

2.7.2 The Cam-clay flow rule

Cam-clay resolves the dilemma (mentioned in section 2.4.4) about whether the principle of normality can be applied to soils. In Cam-clay, normality is applied, but not to what was previously regarded as the appropriate yield surface (i.e.

Mohr–Coulomb or Drucker–Prager). Cam-clay separates the yield surface from the failure criterion: it is to the yield surface (i.e. (2.18)) that normality must be applied.

Fig. 2.38 shows the Cam-clay yield locus with superimposed incremental strain vectors. When yielding takes place with $\eta < M$ then there are compressive volumetric strains (in drained tests) or there is a tendency to generate positive pore pressures. When yielding takes place with $\eta > M$ then there are dilative volumetric strains (in drained tests) or there is a tendency to generate negative pore pressures (in undrained tests). In the (p', V) plot, these two different kinds of behaviour are associated with soil samples which yield above and below (or to the right and the left of) the CSL respectively. The former kind of behaviou. is termed 'wet' (because the positive pore pressures cause the water to flow out of the soil), whereas the latter kind of behaviour is termed 'dry' (because the negative pore pressures result in water being sucked into the soil). Thus yielding is either 'on the wet side of critical' or 'on the dry side of critical'.

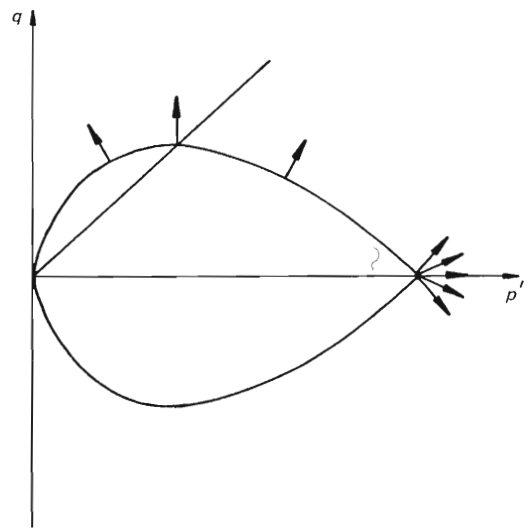


Fig. 2.38 – The Cam-clay flow rule

We can go further in distinguishing between wet and dry types of behaviour in the light of Drucker's postulate. Because the yield locus always shrinks on the dry side and enlarges on the wet side, the second-order work term $\delta\sigma\delta\epsilon^p$ is always negative on the dry side (corresponding to unstable behaviour) and is always positive on the wet side (corresponding to stable behaviour). In situations where the soil is continually sheared in the same direction, the wet side behaviour corresponds to strain-hardening and the dry side behaviour corresponds to strain-softening (perhaps preceded by some strain-hardening).

Critical state soil mechanics gives a good qualitative account of how deformation proceeds in both 'wet' and 'dry' clays. Suppose that one particular zone in a 'wet' clay has strained more than neighbouring zones. This zone will have strain-hardened more than the surrounding soil and will thus be stronger. Further deformation takes place *around* this hardened zone and there is a tendency for the soil to deform in a uniform, homogeneous fashion. On the other hand, if a zone in 'dry' soil has deformed more than the surrounding soil, it will be weaker than the surrounding material. Further deformation will tend to be concentrated in this weakened zone, which will continue to strain-soften. The latter behaviour describes quite well the progressive formation of rupture surfaces in soil. Henkel (1956) made measurements of water contents close to a slip surface consistent with the behaviour described above.

There is often a good match between experimental data for 'wet' clays and Cam-clay (or modified Cam-clay) theory. On the 'dry' side, the match is not so good and the data of failure are better described by Hvorslev's equation (Schofield and Wroth, 1968). Atkinson and Bransby (1978) suggest that soils that hit the Hvorslev surface continue yielding until they reach the critical state. Although some soils follow this pattern, there are others which do not. Both approaches give the same undrained shear strength on the dry side, which tends to overpredict observed strengths for some soils. Although the Hvorslev equation may be useful in some contexts, our experience is that it does not have any advantages over Cam-clay when used with finite elements.

2.7.3 Modified Cam-clay

Although Cam-clay makes a significant step forward in the modelling of soil behaviour, there are some aspects of stress–strain modelling where it is deficient. Of course, it is not alone in this respect. Every theoretical description of material behaviour will have some successes in matching reality and some failures. The overall utility of a particular material idealisation will rest primarily with whether it successfully models those aspects of material response which are pertinent for the problem at hand.

Modified Cam-clay (Burland, 1965; Roscoe and Burland, 1968) addresses two particular dissatisfactions with the original Cam-clay model: the point on the yield locus and the predicted value of K_0 (the coefficient of earth pressure at rest). The objection to the point is to a certain degree aesthetic (it does not look right) and to a certain degree based on experimental evidence (the shear strains predicted by Cam-clay are too high at low stress ratios). In fact there is no theoretical objection to yield surfaces with slope discontinuities: Koiter (1953) shows that the plastic strain increment vector at such a point must lie within the 'fan' of possible directions (e.g. see Fig. 2.38 for the condition on the Cam-clay point). As we shall see in Chapter 5, Cam-clay predicts a value of $K_0 = 1$ for a normally consolidated soil where measured values are normally in the range 0.5 to 0.7.

Modified Cam-clay changes the assumption for dissipated work in Cam-clay (i.e. (2.32)) to

$$p'\delta v^{\mathrm{p}} + q\delta\epsilon^{\mathrm{p}} = p'\sqrt{\{\delta v^{\mathrm{p}2} + (\mathrm{M}\delta\epsilon^{\mathrm{p}})^2\}}, \tag{2.38}$$

and this changes the flow rule to

$$\frac{\delta v^{\mathrm{p}}}{\delta\epsilon^{\mathrm{p}}} = \frac{\mathrm{M}^2 - \eta^2}{2\eta} \tag{2.39}$$

(compared with (2.33)).

As before, the flow rule can be integrated to give the modified Cam-clay yield locus:

$$q^2 + \mathrm{M}^2 p'^2 = \mathrm{M}^2 p' p'_{\mathrm{c}} \tag{2.40}$$

which is shown in Fig. 2.39. The modified Cam-clay yield locus is elliptical in shape: this is the main difference between modified Cam-clay and Cam-clay. Because of this different shape of the yield locus the vertical distance between the isotropic NCL and the CSL becomes $(\lambda - \kappa)\ln(2)$ rather than $\lambda - \kappa$.

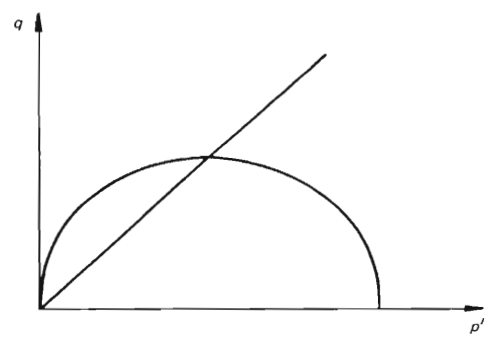


Fig. 2.39 – The modified Cam-clay yield locus is elliptical

For the sake of completeness we summarise the equations for modified Cam-clay in the same order as we presented them for Cam-clay in section 2.5.

(a) Volume–pressure relations: the equation of the isotropic NCL is the same as before:

$$V = N - \lambda \ln(p'), \tag{2.10 \textit{bis}}$$

but $N = \Gamma + (\lambda - \kappa)\ln(2)$. The definitions of V_λ and V_κ are the same as before (equations (2.12) and (2.13)).

(b) Critical state line: the equations are the same as for Cam-clay (i.e. equations (2.14) and (2.15)).

(c) Yielding: the equation of the SSBS is now

$$V_\lambda = \Gamma + (\lambda - \kappa)\{\ln(2) - \ln(1 + (\eta/\mathrm{M})^2)\}. \tag{2.41}$$

(d) Strains: the same assumptions as for Cam-clay, with the exception of the flow rule which is given by (2.39).

The rules for calculating strains given in sections 2.6.3 and 2.6.5 can be used to calculate the strains in triaxial tests, provided that the appropriate equations for the SSBS and the flow rule are used.

The established view is that there is not much difference between Cam-clay and modified Cam-clay for the purposes of making engineering predictions of behaviour. Broadly speaking this is true, but sometimes the difference can be more than would be expected. This is basically because of the way material parameters are chosen: a matter which is discussed in Chapter 5.

2.7.4 Cam-clay: out of date?

Since Cam-clay was proposed in 1963, many deficiencies have been pointed out, and many modifications proposed. It is therefore relevant to ask: is Cam-clay out of date? We believe that it is not, and that Cam-clay (or modified Cam-clay for that matter) will come to be regarded in much the same way as, for example, the Mohr–Coulomb failure criterion. We mean this in the sense that Cam-clay describes certain aspects of soil behaviour extremely well. Starting from a small set of material parameters there are powerful and (relatively) simple calculations that can be made. On the other hand, we do not claim that it provides a universal explanation of all geotechnical phenomena.

Laboratory tests on real soils demonstrate aspects of soil behaviour which are not predicted by the critical state theories. For example, a soil with a high clay fraction which undergoes large relative shear displacements usually exhibits a residual shear strength much lower than the critical state (Skempton, 1985). Recent laboratory tests using internal strain measuring devices have shown that a very wide range of soils has highly non-linear stiffnesses at low strain levels (Jardine *et al.*, 1984). Some normally consolidated natural clays fail in undrained tests well before the critical state is reached. On the other hand, there has been success in using the Cam-clay models in geotechnical predicting, particularly where lightly over-consolidated clay is involved, e.g. embankments and oil tanks on soft foundations. In most geotechnical problems there will be one or two features of the basic soil behaviour which will determine (along with the loads in the system) the overall response. These features may or may not be those included in the critical state framework.

Although Cam-clay can be regarded as deficient in some respects, most attempts to refine theoretical predictions of soil behaviour make use of the concepts of critical state soil mechanics, rather than abandoning them completely. Perhaps the major area of the development of new constitutive equations for soils has been that of cyclic loading, relevant to dynamic loading in earthquakes or on offshore structures in the oil industry. Under the action of cyclic stresses, pore pressure in soil tends to build up a certain cumulative amount in each cycle. If one uses the Cam-clay (or modified Cam-clay) model in these circumstances, then the pore pressure increases in the first cycle, but after that remains constant. This problem can obviously be circumvented by abandoning the assumption of elasticity beneath the SSBS, and this route has

been followed by many. Mroz (e.g. Mroz and Norris, 1982) has proposed models with smaller yield loci 'nested' inside a larger yield locus. Dafalias (e.g. Dafalias and Herrmann, 1982) has proposed a 'bounding surface' model where the amount of plastic behaviour associated with a stress point inside the bounding surface depends on the distance to an image point on the surface. Another model with plasticity inside the traditional yield locus is suggested by Pender (1982). More recent models along these lines include a 'continuous plasticity' model proposed by Naylor (1985) and the 'spread work function' of Dean (1985). Some of these models have the promise of describing better anisotropic yielding and dry-side behaviour.

However, we should point out that all these models are more complicated than Cam-clay. If one of them is going to supplant Cam-clay then the extra work involved in doing calculations must be offset both by a better conceptual picture and by better numerical predictions.

3

Analysis of Consolidation using Finite Elements

3.1 INTRODUCTION

In Chapter 1 we presented the underlying assumptions and basic equations of Biot's consolidation theory. The system of partial differential equations that was obtained described the relationship between total and effective stresses, excess pore pressures, strains and artificial seepage velocities at *one point* in a body of soil. These equations were obtained by applying physical balance laws (describing equilibrium of stresses and continuity of volumetric strain with water flow) to infinitesimally small elements of soil. This chapter shows how the finite element method can be used to solve a particular *boundary value problem* where some combination of loads and drainage boundary conditions acts on a finite volume of soil.

Mathematically the solution of a particular problem is equivalent to finding some mathematical functions which define the time dependent distribution of displacements and excess pore pressure which satisfy the governing differential equations at all points in the 'domain' of the problem. These distributions must also satisfy some conditions on the boundary of the problem domain. For the excess pore pressure these boundary conditions will be either prescribed values of excess pore pressure or prescribed artificial velocities of water flow. The boundary conditions in the case of the stresses will be either prescribed displacements or prescribed distributions of stress. Traditional engineering mathematics is largely concerned with solving problems of this type. Establishing a solution to a particular problem involves a lot of mathematical manipulation, and so an engineer will normally make use of a 'standard' solution from a book or

academic journal. Although there are not many published solutions for consolidation problems, there are many standard solutions for the related equations of elastic stress analysis and steady seepage (see, for example, Timoshenko and Goodier (1970) or Poulos and Davis (1974) for stress analysis and Harr (1962) for seepage).

Whereas these mathematical or 'analytical' solutions are *exact* solutions of the relevant equations, the finite element method provides *approximate* solutions of the same systems of equations. The mathematical techniques used in obtaining these approximate solutions are not covered in most engineering courses and so we introduce them in this chapter. In our view, successful use of the finite element technique is dependent on engineering judgement rather than knowledge of the mathematics. Indeed we agree with Irons and his co-authors (Irons and Ahmad, 1980; Irons and Shrive, 1983) that the teaching of finite elements is becoming much too mathematical. The reasons for this trend are understandable: the finite element method for elastic stress analysis was originally developed on a largely intuitive basis. It is only recently that the underlying mathematics has come to be understood. It is possible to identify three stages in how finite element techniques for stress analysis have been formulated and interpreted over the last three decades:

(a) the method was regarded as an extension of matrix methods for the computerised analysis of structural frames. This method requires a 'stiffness matrix' describing the stiffness properties of one part of the structure. The only difference between a computer program for matrix analysis and one for finite element analysis is that the latter uses stiffness matrices which describe the stiffness of parts of a continuum. These matrices were calculated using structural theorems such as the principle of virtual work or Castigliano's theorem;

(b) the method was recognised as an application of the calculus of variations. In this classical method of engineering analysis the solution to a system of differential equations is obtained by converting the problem into an equivalent one of minimising a 'functional'. For example, solving a problem of elastic stress analysis is equivalent to minimising the total potential energy of the system;

(c) the method was recognised as a particular application of Galerkin's weighted residual method. Weighted residual methods obtain approximate solutions to systems of differential equations by arranging for the (hopefully small) error in the solution to be distributed in some manner throughout the continuum.

The important point to emphasise about these three different approaches is that each interpretation or formulation leads to an identical set of algebraic equations to be solved on the computer. Clearly it is largely a matter of taste how one sets up these equations. Weighted residual methods are now in fashion and while they are used in the following material we would prefer to be able to use a more

direct approach (like the virtual work principle for stress analysis). This is because we believe that the earlier approach is easier to understand and develops an 'engineering' approach to finite elements rather than relying on the mathematics. Unfortunately, there seems to be no direct counterpart of virtual work in fluid mechanics problems. However, when we come to transform the continuity equation in the standard way (to get it in a form suitable for computer solution), we notice a strong similarity with virtual work. Indeed, we can regard the continuity equation as being equivalent to a virtual power (or work) equation.

Since the scope of this chapter is wide, we now summarise the material and explain its arrangement:

1. The next section covers the mathematical preliminaries (numerical integration, interpolation polynomials, the approximate solution of differential equations and Green's theorem in the plane).
2. Section 3.3 presents the fundamentals of the 'displacement' (or 'stiffness') method of finite element analysis via a simple example using linear elastic springs.
3. Section 3.4 covers the virtual work principle. We include this section because we are aware that many engineers find the principle rather obscure. However, it turns out to be a very versatile and powerful tool in formulating finite element methods as well as the theory of structures.
4. Section 3.5 describes the basic theory for formulating the stiffness matrices of 'displacement' finite elements. This subject matter occupies several chapters in other books on finite elements (where the reader is referred for a more detailed treatment).
5. Section 3.6 completes the chapter with a derivation of the finite element equations for consolidation analysis, a FORTRAN program implementing these equations and some examples of its use.

3.2 MATHEMATICAL AND NUMERICAL PRELIMINARIES

3.2.1 Numerical integration

When there is a need to calculate an integral in a computer program, two approaches are possible. The first approach is to take the expression to be integrated and to 'integrate the expression by hand'. The resulting formula is then coded directly in the computer program. The second approach is to perform the integration within the computer program using the techniques of 'numerical integration'. In the latter approach the integral is calculated as the weighted sum of values of the function of some points in the interval. The basic technique of numerical integration will be illustrated below by considering the calculation of areas under curves. Finite element programs usually use numerical integration to calculate the coefficients of element stiffness matrices: this is done in CRISP and also in the program in section 3.6.

First we consider the calculation of the area shown in Fig. 3.1. Mathematically we write this integration:

$$A = \int_0^{10} f(x)\,dx$$

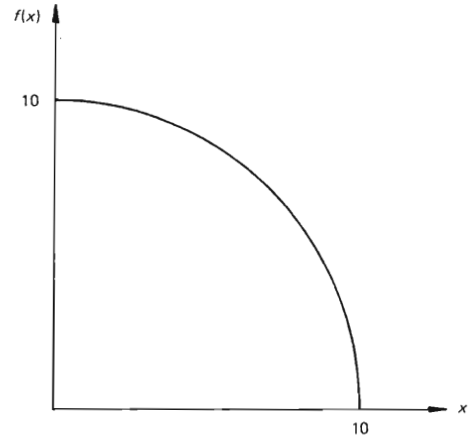


Fig. 3.1 – The function $f(x) = \sqrt{(100 - x^2)}$

where $f(x) = \sqrt{(100 - x^2)}$. Of course this is simply one quarter of the area of a circle of radius ten units ($\pi 10^2/4 = 78.54$). This example is convenient for illustrative purposes because none of the methods considered gives the exact answer. Thus the example will give a rough idea of the accuracy of the different methods. The three methods are known as the trapezoidal rule, Simpson's rule, and two-point Gaussian integration. When using one of these methods the interval between the limits of integration is split into a number of strips, as shown in Fig. 3.2. Each method then applies a different formula or 'rule' calculate the area, A, of a typical strip which starts at $x = x_1$ and ends at $x = x_2$ ($x_2 = x_1 + h$).

Trapezoidal:

$$A = \int_{x_1}^{x_2} f(x)\,dx = (h/2)\,f(x_1) + (h/2)\,f(x_2).$$

Simpson:

$$A = \int_{x_1}^{x_2} f(x)\,dx = (h/6)\,f(x_1) + (2h/3)\,f((x_1 + x_2)/2) + (h/6)\,f(x_2).$$

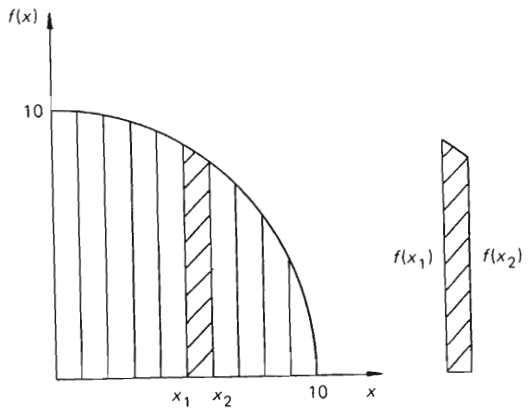


Fig. 3.2 – Separate strips for numerical integration

Two-point Gaussian integration:

$$A = \int_{x_1}^{x_2} f(x)\,dx = (h/2)\,f(x_1 + h(1 - 1/\sqrt{3})/2) + (h/2)\,f(x_2 - h(1 - 1/\sqrt{3})/2).$$

The presentation of these rules is simplified by the adoption of a co-ordinate system which is local to each strip. The local co-ordinate ξ is given by the expression

$$\xi = (2x - (x_1 + x_2))/(x_2 - x_1);$$

thus $\xi = -1$ when $x = x_1$ and $\xi = 1$ when $x = x_2$ (at the midpoint of the strip $\xi = 0$). The local and global systems are shown in Fig. 3.3. The integration rules are now written as follows.

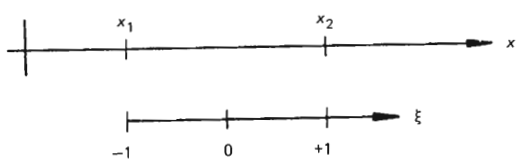


Fig. 3.3 – Local co-ordinate system adopted for numerical integration

Trapezoidal:

$$A = \frac{h}{2}\int_{-1}^{+1} f(\xi)\,d\xi = (h/2)\,f(-1) + (h/2)\,f(+1).$$

Simpson:

$$A = \frac{h}{2}\int_{-1}^{+1} f(\xi)\,d\xi = (h/6)\,f(-1) + (2h/3)\,f(0) + (h/6)\,f(+1).$$

Two-point Gaussian integration:

$$A = \frac{h}{2}\int_{-1}^{+1} f(\xi)\,d\xi = (h/2)\,f(-1/\sqrt{3}) + (h/2)\,f(+1/\sqrt{3}).$$

(The $h/2$ terms come from changing the integration variable, and are equal to $dx/d\xi$.)

Table 3.1 shows the result of applying these three rules with different numbers of strips.

Table 3.1

Number of strips	Trapezoidal rule	Simpson's rule	Two-point Gauss rule
1	50.00	74.40	79.61
2	68.30	77.09	78.91
4	74.89	78.03	78.67
8	77.25	78.36	78.59
16	78.08	78.48	78.56

The integration rules used in finite element programs are usually based on Gauss rules because they give superior accuracy for a given number of function evaluations. Another example of the calculation of the area under a curve will help explain the superiority of the Gauss rules. Consider the integral

$$\int_2^6 (5 + x - (3/4)x^2 + (1/8)x^3)\,dx.$$

Fig. 3.4 shows the area equivalent to this integral. First we write this integral in terms of local co-ordinates:

$$2\int_{-1}^{+1} (5 + 2\xi + 3\xi^2 + \xi^3)\,d\xi.$$

Integrating analytically we obtain $A = 24$. Applying the trapezoidal rule (using one strip) gives the area as 32. The geometric interpretation of the trapezoidal rule is quite straightforward: the cubic curve is approximated as a straight line and the integral is equal to the area of the trapezium. The principle underlying Simpson's rule is similar: only now the curve is approximated as the quadratic curve which passes through values of the function at the two end points and the

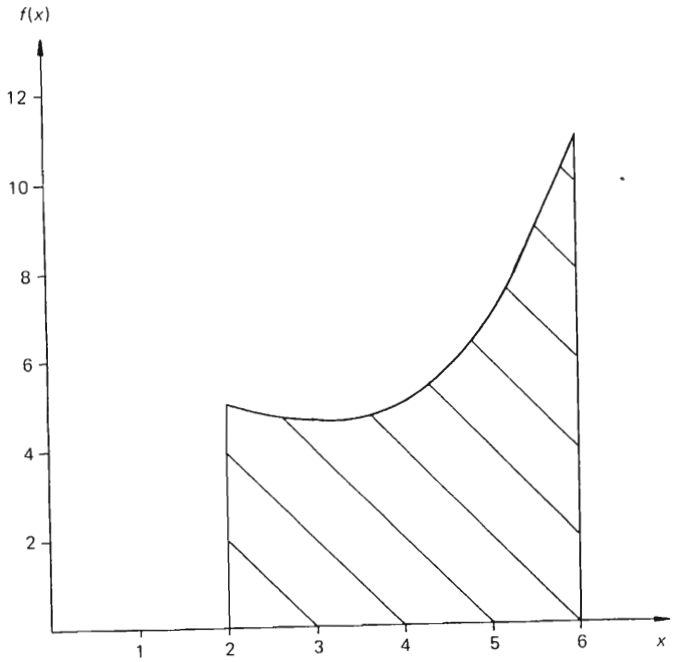


Fig. 3.4 – $\int_2^6 (5 + x - (3/4)x^2 + (1/8)x^3)\,dx$

midpoint of the interval. Applying Simpson's rule, the area is evaluated as 24. At first sight this result is surprising: we have assumed a quadratic approximation to a cubic curve, yet the exact answer has been obtained for the area beneath the curve. In fact this result is not fortuitous – it has happened because we have been wise (or perhaps lucky) in the choice of points to 'sample' the function. This prompts the question: is there a way of choosing the sampling or integration points to achieve optimum accuracy? As we have implied above, the answer is 'yes', and it is the Gauss rules which represent that optimum choice. Using the two-point Gauss rule on the above example, the exact answer (24) is again obtained. In general a Gauss rule with n integration points exactly integrates a polynomial including terms up to the power $2n - 1$.

3.2.2 Interpolation polynomials (shape functions)

Underlying the derivation of the integration rules described in the previous section is the concept of the interpolation polynomial. If one knows the values of a function at (say) three separate points in some interval, then it is possible to fit a quadratic curve to the three points.

Consider the general quadratic

$$f = c_0 + c_1 x + c_2 x^2. \tag{3.1}$$

The three coefficients c_0, c_1 and c_2 are uniquely determined by the three values of the function, and can be obtained by substituting into (3.1) three times and solving the resulting equations. In fact one can write down the quadratic straight away as

$$f = f_1 \frac{(x_3 - x)(x_2 - x)}{(x_3 - x_1)(x_2 - x_1)} + f_2 \frac{(x_3 - x)(x_1 - x)}{(x_3 - x_2)(x_1 - x_2)} + f_3 \frac{(x_2 - x)(x_1 - x)}{(x_2 - x_3)(x_1 - x_3)}. \tag{3.2}$$

It is possible to see by substituting $x = x_1$ etc. that this must be the corr equation of the quadratic. An expression in this form is known as a Langragian interpolation polynomial, and the idea can clearly be extended to any number of points. Expressions of this form arise quite often in finite element theory where the notation

$$f = f_1 N_1 + f_2 N_2 + f_3 N_3$$

is often adopted and each of the N_i is referred to as a 'shape function'.

3.2.3 Approximate solution of differential equations

The problem of steady seepage is used to demonstrate the basic technique. The problem we choose to solve is that of radial seepage away from a borehole which contains water under a pressure which is maintained at a constant value. As shown in Chapter 1 the solution of seepage problems is equivalent to solving the partial differential equation known as Laplace's equation subject to the appropriate boundary conditions. In the case of cylindrical radial symmetry this equation can be written:

$$\frac{d^2\bar{u}}{dr^2} + \frac{1}{r}\frac{d\bar{u}}{dr} = 0.$$

The problem to which a solution is sought is: what is the distribution of excess pore pressure in the soil if the internal boundary is maintained at an excess pore pressure of 10 kPa and the external boundary is maintained at zero excess pore pressure (Fig. 3.5)? The exact solution can be obtained by integrating analytically:

$$\bar{u} = \frac{10 \ln (16/r)}{\ln (16)}.$$

To find an approximate solution of the problem, the distribution of excess pore pressure is represented by a quadratic equation:

$$\bar{u} = c_0 + c_1 r + c_2 r^2.$$

As above, this is conveniently written:

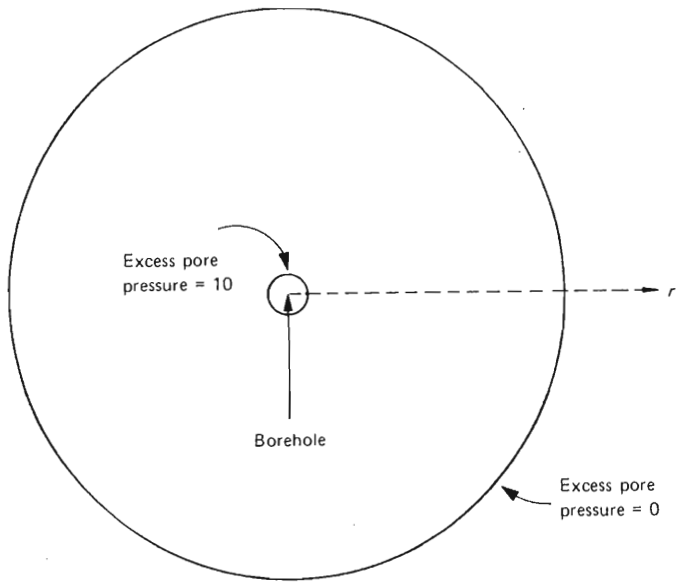


Fig. 3.5 – Cylindrical steady seepage from a borehole

$$\bar{u} = \bar{u}_i \frac{(r_0 - r)(r_c - r)}{(r_0 - r_i)(r_c - r_i)} + \bar{u}_c \frac{(r_0 - r)(r_i - r)}{(r_0 - r_c)(r_i - r_c)} + \bar{u}_0 \frac{(r_c - r)(r_i - r)}{(r_c - r_0)(r_i - r_0)}.$$

Adopting the appropriate values (i.e. $\bar{u}_i = 10$, $\bar{u}_0 = 0$, $r_i = 1$, $r_c = 8.5$ and $r_0 = 16$):

$$\bar{u} = \frac{10(16 - r)(8.5 - r)}{112.5} + \frac{\bar{u}_c(r - 1)(16 - r)}{56.25}. \tag{3.3}$$

Since the excess pore pressures on the two boundaries are known, there is effectively one value of excess pore pressure (taken for convenience at the mid-point between the internal and external boundaries) which defines the variation throughout the soil. How can a value be assigned to this single unknown to furnish a 'good' approximate solution to the problem, bearing in mind that it will not be possible to obtain the exact (logarithmic) solution?

The method to be described for doing this belongs to a group of methods known as *weighted residual methods*. The basic procedure is to take an expression for the unknown pore pressure (such as (3.3) above) and to substitute it into the differential equation. For each value of r the approximating function will not satisfy the differential equation exactly, but there will be an error or

residual: $R(r)$. A weighted residual method makes this error as small as possible by applying the condition

$$\int_V W R \, \mathrm{d(vol)} = 0, \tag{3.4}$$

where W is a weighting function: different weighted residual methods make use of different weighting functions.

According to Crandall (1956), Courant was the first to classify the different methods of obtaining approximate solutions to differential equations as 'weighted residual methods'. We shall make use of the method proposed in 1915 by Galerkin (Galerkin, 1915), who suggested that the weighting functions W should be the same as the interpolation (or shape) functions. Thus we write the distribution of excess pore pressure as

$$\bar{u} = N_i \bar{u}_i + N_c \bar{u}_c + N_0 \bar{u}_0, \tag{3.5}$$

then the weighting function is taken as

$$W = N_i W_i + N_c W_c + N_0 W_0, \tag{3.6}$$

where W_i, W_c and W_0 are arbitrary scalars.

The weighted residual equation is:

$$\int_V W \left[\frac{\partial^2 \bar{u}}{\partial r^2} + \frac{1}{r} \frac{\partial \bar{u}}{\partial r} \right] r \, \mathrm{d}r = 0, \tag{3.7}$$

which can be written as

$$\int_V W \frac{\mathrm{d}}{\mathrm{d}r} \left[r \frac{\mathrm{d}\bar{u}}{\mathrm{d}r} \right] \mathrm{d}r = 0. \tag{3.8}$$

(3.8) is now integrated by parts:

$$\left[W r \frac{\mathrm{d}\bar{u}}{\mathrm{d}r} \right]_{r_i}^{r_0} - \int_V \frac{\mathrm{d}W}{\mathrm{d}r} \frac{\mathrm{d}\bar{u}}{\mathrm{d}r} \cdot r \, \mathrm{d}r = 0. \tag{3.9}$$

We now make the substitutions (3.5) and (3.6) for $\bar{u}$ and W.

In general if there are n unknown coefficients to be determined we can obtain n equations by letting each of the W_j be 1 in turn (while all other W_j are zero). Here, there is just one unknown and so we just substitute $W = N_c$ and $\bar{u}$ from (3.3). After a certain amount of (lengthy) manipulation we obtain the solution

$$\bar{u}_c = 190/68.$$

Fig. 3.6 shows the comparison between the exact and approximate solutions. There are two ways of obtaining a more accurate solution. The first is to include more terms in the polynomial: this is the classical approach in engineering analysis. The second way is to split the interval into a number of sub-intervals,

with lower-order polynomials in each: this is the modern or finite element approach. (In fact we introduce finite elements below in a more direct physical way.)

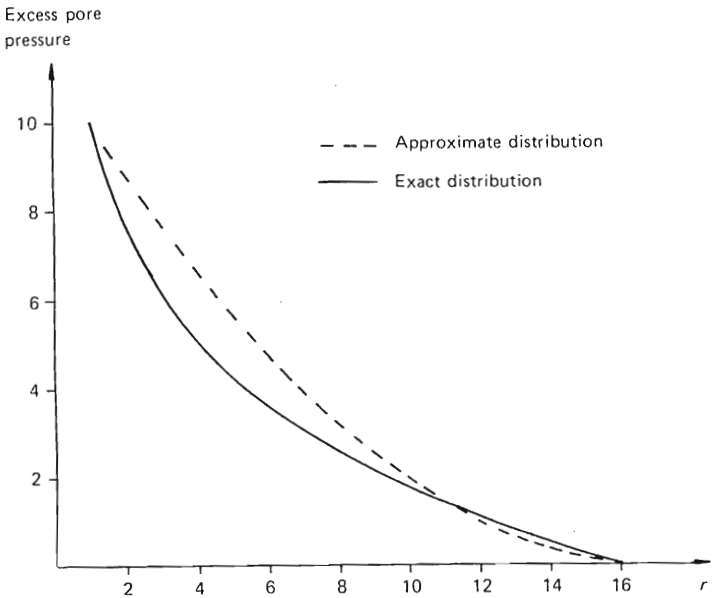


Fig. 3.6 – Comparison of approximate and exact solutions for cylindrical seepage

3.2.4 Zienkiewicz–Green theorem

When we come to do the integration by parts described above in a two-dimensional problem we make use of the following standard results . standard results are

$$\int_A f \frac{\partial g}{\partial x} \, dx\, dy = - \int_A g \frac{\partial f}{\partial x} \, dx\, dy + \int_S f g \, n_x \, dS, \tag{3.10}$$

and

$$\int_A f \frac{\partial g}{\partial y} \, dx\, dy = - \int_A g \frac{\partial f}{\partial y} \, dx\, dy + \int_S f g \, n_y \, dS, \tag{3.11}$$

where n_x and n_y are the direction cosines of the outward normal **n** to the closed curve S surrounding the area A.

These results are proved in Zienkiewicz (1977), in the form given above. We also make use of the three-dimensional version of this theorem, which is basically Gauss's divergence theorem with the extra ingredient of integration by parts. Zienkiewicz refers to (3.10) and (3.11) as Green's theorem, but the writers

of mathematical texts use this name for another result. We believe that these formulae have no generally accepted name, and therefore we will call them the 'Zienkiewicz–Green' theorem.

3.3 THE DISPLACEMENT METHOD

3.3.1 General procedure

This section explains the basic steps of the displacement method of finite element analysis. This will be done by considering a simple example where the 'finite elements' are linear elastic springs.

Consider the system of interconnected springs shown in Fig. 3.7. The springs are assumed to be weightless and are interconnected at *nodes* which are the points labelled 1, 2, 3 and 4. Weights can be hung from the nodal points and the question which must be answered is: what are the vertical displacements of the nodes and the tensions in the springs? The problem is 'statically indeterminate' in the terminology of structural mechanics, that is: it is not possible to calculate the forces in the springs from the equilibrium equations alone.

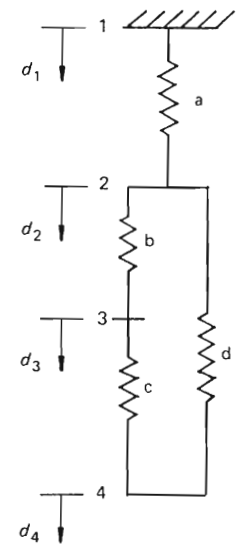


Fig. 3.7 – System of interconnected springs

In order to find the spring tensions it is necessary to take into account the stiffnesses of the individual springs k_a, k_b, k_c and k_d relating the tension in each spring to its elongation:

$$T_a = k_a \, e_a,$$

$$T_b = k_b \, e_b,$$

$$T_c = k_c\, e_c,$$

$$T_d = k_d\, e_d.$$

To arrive at a solution for this problem, the three fundamental principles of structural mechanics (compatibility, material behaviour and equilibrium) are applied in turn. What distinguishes the *displacement* method from other solution methods is first the choice of basic unknowns (i.e. displacements) and second the order in which the three principles are applied.

Compatiblity: the basic unknowns are defined as the displacements of the nodal points (see Fig. 3.7). The equations of compatibility are

$$e_a = d_2 - d_1,$$

$$e_b = d_3 - d_2,$$

$$e_c = d_4 - d_3,$$

$$e_d = d_4 - d_2.$$

Material behaviour: using the definitions of spring stiffnesses detailed above:

$$T_a = k_a(d_2 - d_1),$$

$$T_b = k_b(d_3 - d_2),$$

$$T_c = k_c(d_4 - d_3),$$

$$T_d = k_d(d_4 - d_2).$$

Equilibrium: considering the forces acting at node 2 (see Fig. 3.8):

$$T_a = T_b + T_d + W_2,$$

i.e.

$$k_a(d_2 - d_1) = k_b(d_3 - d_2) + k_d(d_4 - d_2) + W_2$$

and rearranging this equation:

$$-k_a d_1 + (k_a + k_b + k_d)d_2 - k_b d_3 - k_d d_4 = W_2.$$

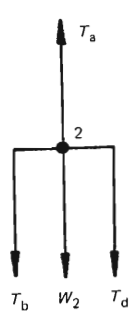


Fig. 3.8 – Forces acting on node 2

Similar equations can be written for the other nodes, giving four linear simultaneous equations in d_1, d_2, d_3 and d_4 which can be expressed in matrix form:

$$\begin{bmatrix} k_a & -k_a & & \\ -k_a & k_a + k_b + k_d & -k_b & -k_d \\ & -k_b & k_b + k_c & -k_c \\ & -k_d & -k_c & k_c + k_d \end{bmatrix} \begin{bmatrix} d_1 \\ d_2 \\ d_3 \\ d_4 \end{bmatrix} = \begin{bmatrix} W_1 \\ W_2 \\ W_3 \\ W_4 \end{bmatrix}.$$

The square matrix is called the *global stiffness matrix* for the collection of springs. The equation can be written in matrix notation:

$$\mathbf{Kd} = \mathbf{W}.$$

General rules for determining the coefficients in the global stiffness matrix for a general arrangement of springs can be stated:

Rule 1: the diagonal term for node i is made up of the sum of all the individual spring stiffnesses that are connected to node i.

Rule 2: the off-diagonal terms (i, j) and (j, i) contain the stiffness of the spring connecting node i and node j multiplied by -1.

An equivalent statement is that the global stiffness matrix consists of the sum of matrices of the following form: (where k_e is the stiffness of one particular spring)

$$\begin{array}{c} \\ i \\ j \end{array} \begin{array}{c} \begin{array}{cc} i & j \end{array} \\ \begin{bmatrix} k_e & -k_e \\ -k_e & k_e \end{bmatrix} \end{array}.$$

One of these matrices is added into the global stiffness matrix for each spring in the collection. The node numbers i and j indicate where the terms must be added (or 'assembled') into the global matrix. These matrices are called the *element stiffness matrices* of each spring, relating nodal displacements to the forces exerted on each spring at nodal points.

$$\begin{bmatrix} k_e & -k_e \\ -k_e & k_e \end{bmatrix} \begin{bmatrix} d_i \\ d_j \end{bmatrix} = \begin{bmatrix} F_i \\ F_j \end{bmatrix}.$$

The forces acting on each nodal point taken to be positive downwards; thus

$$F_i = -T_e \quad \text{and} \quad F_j = T_e.$$

3.3.2 Solving the equations

To demonstrate the method of solving these equations, the following values are adopted: $k_a = k_b = k_c = 20$, $k_d = 10$, $W_2 = W_4 = 0$ and $W_3 = 1$. Thus the equations which must be solved are

$$\begin{matrix}(1)\\(2)\\(3)\\(4)\end{matrix}\quad \begin{bmatrix} 20 & -20 & 0 & 0 \\ -20 & 50 & -20 & -10 \\ 0 & -20 & 40 & -20 \\ 0 & -10 & -20 & 30 \end{bmatrix} \begin{bmatrix} d_1 \\ d_2 \\ d_3 \\ d_4 \end{bmatrix} = \begin{bmatrix} W_1 \\ 0 \\ 1 \\ 0 \end{bmatrix},$$

subject to the boundary condition $d_1 = 0$. These equations are solved using the process known as *Gaussian Elimination*. The first stage of this process (known as forward elimination) is based on the observation that adding an arbitrary multiple of one equation to any other equation does not change the solution of the set of equations.

First, however, it is necessary to deal with the boundary condition $d_1 = 0$. There are many alternative methods for doing this, but one of the simplest (which is adopted here) is to add a large number (say 10^6) to the diagonal term of equation (1). This forces this equation to yield a solution of $d_1 = 0$. Physically the addition of this large number can be interpreted as the connection of node 1 to earth with a very stiff spring (with a stiffness of 10^6).

$$\begin{bmatrix} 10^6 & -20 & 0 & 0 \\ -20 & 50 & -20 & -10 \\ 0 & -20 & 40 & -20 \\ 0 & -10 & -20 & 30 \end{bmatrix} \begin{bmatrix} d_1 \\ d_2 \\ d_3 \\ d_4 \end{bmatrix} = \begin{bmatrix} W_1 \\ 0 \\ 1 \\ 0 \end{bmatrix}.$$

Now the process of forward elimination is started:

(a) Multiples of the first equation are added to the following equations so that the coefficients of d_1 in these equations become zero. (This process is called *eliminating* d_1 from the following equations.) In this particular example, only equation (2) needs to be modified according to the following rule:

(new equation (2)) = (old equation (2)) + $(20/10^6)$ × (equation 1)):

$$\begin{bmatrix} 10^6 & -20 & 0 & 0 \\ 0 & 50 & -20 & -10 \\ 0 & -20 & 40 & -20 \\ 0 & -10 & -20 & 30 \end{bmatrix} \begin{bmatrix} d_1 \\ d_2 \\ d_3 \\ d_4 \end{bmatrix} = \begin{bmatrix} W_1 \\ 0 \\ 1 \\ 0 \end{bmatrix}.$$

(b) Multiples of equation (2) are now added to equations (3) and (4) to eliminate coefficients of d_2 from those equations:

(new equation (3)) = (old equation (3)) + (20/50) × (equation (2))
(new equation (4)) = (old equation (4)) + (10/50) × (equation (2)).

$$\begin{bmatrix} 10^6 & -20 & 0 & 0 \\ 0 & 50 & -20 & -10 \\ 0 & 0 & 32 & -24 \\ 0 & 0 & -24 & 28 \end{bmatrix} \begin{bmatrix} d_1 \\ d_2 \\ d_3 \\ d_4 \end{bmatrix} = \begin{bmatrix} W_1 \\ 0 \\ 1 \\ 0 \end{bmatrix}.$$

The general method being adopted is now apparent: in step (a), terms in column 1 under the diagonal become zero whereas in step (b), terms in column 2 under the diagonal became zero. The matrix is gradually being converted into 'upper triangular' form.

(c) Eliminate coefficient of d_3 from equation (4):

(new equation (4)) = (old equation (4)) + (24/32) × (equation (3)).

$$\begin{bmatrix} 10^6 & -20 & 0 & 0 \\ 0 & 50 & -20 & -10 \\ 0 & 0 & 32 & -24 \\ 0 & 0 & 0 & 10 \end{bmatrix} \begin{bmatrix} d_1 \\ d_2 \\ d_3 \\ d_4 \end{bmatrix} = \begin{bmatrix} W_1 \\ 0 \\ 1 \\ 3/4 \end{bmatrix}.$$

Forward elimination is now complete. Now the process of *back-substitution* is started.

(d) Solve for d_4 from the last equation:

$$d_4 = 3/40.$$

(e) Solve for d_3 from the third equation:

$$32d_3 - 24(3/40) = 1.$$

$$d_3 = 7/80.$$

(f) Solve for d_2 from the second equation:

$$50d_2 - 20(7/80) - 10(3/40) = 0.$$

$$d_2 = 1/20.$$

(g) Solve for d_1 from the first equation:

$$10^6 d_1 - 20d_2 = W_1.$$

$$d_1 = 0 \text{ (very nearly)}.$$

From the nodal displacements it is now possible to calculate the spring elongations and tensions.

3.3.3 A computer program for the displacement method

Listed below is a FORTRAN program which can be used to analyse collections of springs similar to the one considered above.

The basic steps of this short program are highlighted by the comments in the listing. The identical steps are present in the finite element program for consolidation analysis presented later in this chapter, and in CRISP. To use the program it is necessary to present it with input data describing the problem to be analysed. The input data must be prepared according to the following scheme:

Data record	Contents				No. of records
A	NN	NS	NF	NL	1
B	N1	N2	AK		NS
C	NOD	FIX			NF
D	NOD	W			NL

where in record A, NN is the number of nodes, NS is the number of springs, NF is the number of nodes with prescribed displacements, and NL is the number of loaded nodes. In records of type B, N1 and N2 are the node numbers at either end of a spring and AK is its stiffness. In records of type C, NOD is the node number which is given a prescribed displacement with a value FIX. In records of type D, NOD is the node which is loaded with a load W.

An example data file follows the program listing.

```
      DIMENSION ST(12,12),RHS(12)
      WRITE(6,100)
  100 FORMAT(16HOSPRINGS PROGRAM)
C*************************************INITIALISE**********************
      DO 6 J=1,12
      DO 4 I=1,12
    4 ST(I,J)=0.
    6 RHS(J)=0.
C*************************************READ DATA***********************
      READ(5,101) NN,NS,NF,NL
  101 FORMAT(4I5)
      IF(NN.LT.0) STOP
      WRITE(6,102) NN,NS,NF,NL
  102 FORMAT(11HONODES.....,I5/11H SPRINGS...,I5/
     1       11H FIXES.....,I5/11H LOADS.....,I5)
C*************************************ASSEMBLE************************
      DO 10 N=1,NS
      READ(5,103) N1,N2,AK
  103 FORMAT(2I5,F10.0)
      WRITE(6,104) N1,N2,AK
  104 FORMAT(2I5,F10.3)
      ST(N1,N1)=ST(N1,N1)+AK
      ST(N2,N2)=ST(N2,N2)+AK
      ST(N1,N2)=ST(N1,N2)-AK
   10 ST(N2,N1)=ST(N2,N1)-AK
C*************************************FIX NODES***********************
      DO 14 I=1,NF
      READ(5,105) NOD,FIX
  105 FORMAT(I5,F10.0)
      WRITE(6,106) NOD,FIX
  106 FORMAT(1X,I5,F10.3)
```



```
      ST(NOD,NOD)=ST(NOD,NOD)+1.0E6
   14 RHS(NOD)=RHS(NOD)+1.0E6*FIX
C*************************************LOADS ON NODES******************
      DO 18 I=1,NL
      READ(5,107) NOD,W
  107 FORMAT(I5,F10.0)
      WRITE(6,106) NOD,W
   18 RHS(NOD)=RHS(NOD)+W
C*************************************FORWARD ELIMINATION**************
      NN1=NN-1
      DO 30 IQ=1,NN1
      I1=IQ+1
      DO 26 I=I1,NN
      DO 22 J=IQ,NN
   22 ST(I,J)=ST(I,J)-ST(IQ,I)*ST(IQ,J)/ST(IQ,IQ)
   26 RHS(I)=RHS(I)-ST(IQ,I)*RHS(IQ)/ST(IQ,IQ)
   30 CONTINUE
C*************************************BACK SUBSTITUTE******************
      RHS(NN)=RHS(NN)/ST(NN,NN)
      DO 60 II=1,NN1
      IQ=NN-II
      I1=IQ+1
      DO 58 I=I1,NN
   58 RHS(IQ)=RHS(IQ)-ST(IQ,I)*RHS(I)
   60 RHS(IQ)=RHS(IQ)/ST(IQ,IQ)
C*************************************PRINT DISPLACEMENTS**************
      WRITE(6,109) (RHS(I),I=1,NN)
  109 FORMAT(14HODISPLACEMENTS/(1X,10E12.4))
      STOP
      END
```

Below are the data which describe the example worked through above:

A	4	4	1	1
B	1	2	20.0	
B	2	3	20.0	
B	3	4	20.0	
B	2	4	10.0	
C	1	0.0		
D	3	1.0		

Running the program with these data produces the following output:

```
SPRINGS PROGRAM
NODES.....    4
SPRINGS....   4
FIXES.....    1
LOADS.....    1
   1     2      20.000
   2     3      20.000
   3     4      20.000
```

```
 2     4        10.000
 1              0.000
 3              1.000
DISPLACEMENTS
  0.1000E-05   0.5000E-01   0.8750E-01   0.7500E-01
```

and the reader can see that the printed displacements correspond to those calculated in the example.

3.4 VIRTUAL WORK

The general procedure described above can be used to analyse problems where the properties of the individual elements are more complicated than those of the elastic springs described above. The overall approach of assembling the stiffnesses of individual elements into a global stiffness matrix and solving the linear simultaneous equations remains precisely the same as that described above. This holds true regardless of whether the finite elements represent volumes of solid material (i.e. a continuum) or discrete members in a structural framework.

In formulating stiffness matrices for continuum elements, use will be made of the principle of virtual work. The principle of virtual work will be used to determine the equivalent nodal loads which are in equilibrium with internal stresses in the finite elements. Since the virtual work principle is regarded as difficult and/or obscure by many engineers, this section discusses the derivation of the principle for a plane truss and a continuum.

3.4.1 Virtual work for a truss

Fig. 3.9 shows a plane truss consisting of a collection of pin-ended bars. The description of the bars as 'pin-ended' means that an individual bar cannot transmit a moment to other bars via the joints at its ends. The joints in the truss are numbered from 1 to n (if there are n joints) so that loads applied to one joint can be distinguished from loads applied to other joints by the use of numerical subscripts. In the following, one particular joint will be considered and it will be referred to as joint i for the sake of generality. Considering the forces acting on joint i and resolving horizontally and vertically:

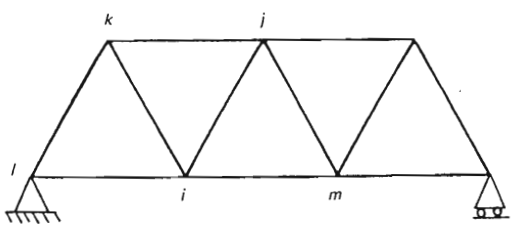


Fig. 3.9 – Pin-jointed truss

$$H_i + T_{ij}\cos\alpha_{ij} + T_{ik}\cos\alpha_{ik} + T_{il}\cos\alpha_{il} + T_{im}\cos\alpha_{im} = 0,$$

$$V_i + T_{ij}\sin\alpha_{ij} + T_{ik}\sin\alpha_{ik} + T_{il}\sin\alpha_{il} + T_{im}\sin\alpha_{im} = 0,$$

where H_i and V_i are the external horizontal and vertical loads acting on the joint and T_{ij} is the tension in the member connecting joint i to joint j which is inclined at an angle α_{ij} to the horizontal. There are n pairs of equations similar to this one (one pair for each joint). The number of terms in each equation depends on the number of bars connecting each joint to other joints in the truss. Here it has been assumed that joint i is connected to four joints: j, k, l and m.

Each equation is now multiplied by a (different) arbitrary number, thus:

$$h_i(H_i + T_{ij}\cos\alpha_{ij} + T_{ik}\cos\alpha_{ik} + T_{il}\cos\alpha_{il} + T_{im}\cos\alpha_{im}) = 0,$$

$$v_i(V_i + T_{ij}\sin\alpha_{ij} + T_{ik}\sin\alpha_{ik} + T_{il}\sin\alpha_{il} + T_{im}\sin\alpha_{im}) = 0.$$

All the equations are now added together:

$$\sum_{\text{joints}} (hH + vV) + \text{a large number of terms} = 0.$$

Examining the form of the 'large number of terms' it can be seen that the following four terms appear owing to the existence of the bar connecting joint i to joint j:

$$\ldots h_i T_{ij}\cos\alpha_{ij} + v_i T_{ij}\sin\alpha_{ij} + h_j T_{ji}\cos\alpha_{ji} + v_j T_{ji}\cos\alpha_{ji} \ldots .$$

Now $T_{ij} = T_{ji}$ = the tension in the bar connecting joint i to joint j. However, $\cos\alpha_{ij} = -\cos\alpha_{ji}$ and $\sin\alpha_{ij} = -\sin\alpha_{ji}$ (see Fig. 3.10). The following series of definitions are now made:

$$e_{ij} = h_j\cos\alpha_{ij} + v_j\sin\alpha_{ij} - h_i\cos\alpha_{ij} - v_i\sin\alpha_{ij}$$

and the equation now becomes

$$\sum_{\text{joints}} (hH + vV) = \sum_{\text{bars}} eT.$$

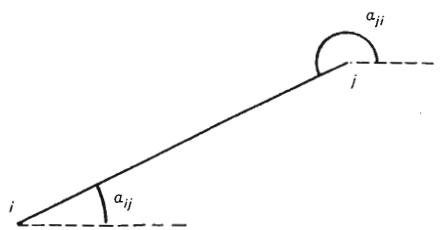


Fig. 3.10 – Geometric relationship used in proof of virtual work for a plane truss ($\cos\alpha_{ij} = -\cos\alpha_{ji}$; $\sin\alpha_{ij} = -\sin\alpha_{ji}$)

If in this equation h_i is set to one and all the other hs and vs are set to zero, then the original equilibrium equation for forces in the horizontal direction at joint i

is recovered. Any of the original equilibrium equations can be recovered in a similar fashion by setting the appropriate h or v to one and the others to zero.

The equation which has just been derived is in fact the principle of virtual work for a pin-jointed truss (or, strictly speaking, the principal of virtual displacements). In deriving this principle, however, no reference has been made to displacements or work quantities: only the principle of equilibrium has been used. Suppose that the 'arbitrary numbers' h_i and v_i are now taken to be horizontal and vertical displacements of joint i. Then the quantity e_{ij} which was defined above turns out to be precisely the extension of bar ij due to these displacements. This result can be obtained by using Pythagoras's theorem to calculate the length of the bar before and after straining and taking the limit of the difference when deflections are small. (Alternative methods are using a 'displacement diagram' or simply resolving the joint displacements along the direction of the bar.) The equations giving bar extensions in terms of joint displacements are the equations of compatibility for the truss.

Alternatively the equations of compatibility could have been used instead of the equilibrium equations as the starting point in the derivation of the principle. The arbitrary numbers which multiply these equations are identified as forces, and by selecting particular force system it is possible to recover the original compatibility equations (or some combination of them). Deriving the principle in this way leads to what is strictly called 'the principle of virtual forces'. It is normal to refer to both these principles as the principle of virtual work. The essential point to note is that either the set of forces in equilibrium or the set of compatible displacements may be 'arbitrary', 'imaginary' or 'virtual' (these are the terms that are commonly used in this context).

The principle of virtual work is being increasingly used in the theory of structures to obtain solutions to redundant frameworks and structures. It is replacing the more traditional energy theorems mainly because the analyst only has to remember one basic principle rather than a series of different theorems (which all depend on virtual work for their proof). The aspect of the principle which leads to many regarding it as obscure is the introduction of the word 'work'. Although it is natural to introduce this term in relation to the product of a force and a displacement, it inevitably leads to some confusion as to what this 'imaginary work' actually represents in practice. In fact, as has been shown above, the principle merely represents statements of equilibrium and compatibility. The fact that both types of statement can be obtained from one single equation is the result of the 'duality' present in definitions of the force and displacement systems. This can be seen in the case of the plane truss in the fact that the $\cos\alpha_{ij}$ and $\sin\alpha_{ij}$ factors occur in both equilibrium and compatibility equations. The proof of the virtual work principle involves transferring these factors from forces to displacements.

3.4.2 Virtual work for a continuum

The starting point is the differential equations of equilibrium for a two-dimensional continuum:

$$\frac{\partial\sigma_x}{\partial x} + \frac{\partial\tau_{yx}}{\partial y} = w_x, \tag{3.12}$$

$$\frac{\partial\tau_{xy}}{\partial x} + \frac{\partial\sigma_y}{\partial y} = w_y. \tag{3.13}$$

These equations are multiplied by arbitrary scalar functions h and v, added together and integrated over the area of the continuum:

$$\int_A \left[h\left[\frac{\partial\sigma_x}{\partial x} + \frac{\partial\tau_{yx}}{\partial y} - w_x\right] + v\left[\frac{\partial\tau_{yx}}{\partial x} + \frac{\partial\sigma_y}{\partial y} - w_y\right]\right] \mathrm{d}(\text{area}) = 0.$$

As in the case of the truss, h and v will subsequently be identified with horizontal and vertical displacements, but initially they are regarded as arbitrary functions (of x and y).

Terms involving the derivatives of stresses are now integrated using the Zienkiewicz–Green theorem. This will be demonstrated by considering the integration of the first term in the equation above:

$$\int_A h\frac{\partial\sigma_x}{\partial x}\,\mathrm{d}x\,\mathrm{d}y = -\int_A \frac{\partial h}{\partial x}\sigma_x\,\mathrm{d}x\,\mathrm{d}y + \int_S h\sigma_x n_x\,\mathrm{d}S.$$

When the arbitrary scalar function h is identified as the horizontal displacement d_x, the term $(\partial h/\partial x)$ is recognised as $-\epsilon_x$. Performing similar integrations for all terms of this type, the principle of virtual work for a continuum is obtained:

$$\int \boldsymbol{\epsilon}^{\mathrm{T}}\boldsymbol{\sigma}\,\mathrm{d}(\text{vol}) = \int \mathbf{d}^{\mathrm{T}}\boldsymbol{\tau}\,\mathrm{d}(\text{area}) + \int \mathbf{d}^{\mathrm{T}}\mathbf{w}\,\mathrm{d}(\text{vol}). \tag{3.14}$$

In this equation, $\boldsymbol{\tau}$ is a vector with components $\tau_x = n_x\sigma_x + n_y\tau_{xy}$ and $\tau_y = n_x\tau_{xy} + n_y\sigma_y$. These are called 'tractions', and the term $\int \mathbf{d}^{\mathrm{T}}\boldsymbol{\tau}\,\mathrm{d}(\text{area})$ represents the work done by these tractions on the boundary of the continuum. A simple transformation shows that this is equivalent to the work done by the direct and shear stresses acting on the inclined boundary.

In order to emphasise in the virtual work principle that the strains are not necessarily caused by the stresses (but can be arbitrary as long as they are compatible), it is common to denote the virtual strains and displacements by a superposed *: $\overset{*}{\boldsymbol{\epsilon}}$ and $\overset{*}{\mathbf{d}}$.

The purpose of this exposition (and the introductory case for the plane truss) was to demonstrate that in both cases the principle of virtual work is derived directly from the equations of equilibrium (or the equations of compatibility). The reason that the virtual work principle is employed in structural analysis (rather than the equilibrium equations) is mainly one of convenience.

However, we draw the reader's attention to the fact that the derivation of the principle of virtual work for a continuum followed a very similar course to the procedure for applying Galerkin's weighted residual method to the seepage problem in section 3.2.3. Indeed, we could have referred to the arbitrary scalar functions h and v as weighting functions, and it is possible to regard (3.14) as a

weighted residual statement. The identification of (3.14) as both the virtual work principle and a weighted residual statement leads to a physical interpretation of what is happening when an approximate solution is obtained using this equation. Substituting an approximate stress distribution into the equilibrium equations (3.12) and (3.13) gives a residual term which corresponds to an error in the body force. Satisfaction of (3.14) ensures that the integral of the work done by the (erroneous) body force is locally zero (over an area associated with each node in a finite element mesh). Alternatively the statement can be regarded as one of local equilibrium, in which the resultants of internal stresses, body forces and boundary stresses balance at the nodal points.

3.5 DISPLACEMENT FINITE ELEMENTS

3.5.1 The basic formula

In this section the account of the displacement method is taken one step further by considering some 'finite elements' which are rather more complicated than the springs considered in section 3.3. As mentioned previously, the general solution procedure remains the same regardless of the type of element employed.

First the basic technique for obtaining the stiffness matrix for a finite element based on an assumed displacement field is presented. The technique is then illustrated by deriving element stiffness matrices first for a pin-ended bar and second for a triangular element to be used in the analysis of plane strain problems.

The notation used follows that established by Zienkiewicz in his series of texts on the finite element method (1967, 1971, 1977). The first step is to express the displacement inside the finite element as a function of the displacements of nodal points and position within the element. This relationship is written in matrix notation:

$$\mathbf{d} = \mathbf{N}\mathbf{a}_e, \tag{3.15}$$

where

$$\mathbf{d} = \begin{bmatrix} d_x \\ d_y \end{bmatrix}$$

and $\mathbf{a}_e$ is a vector listing all the nodal displacements associated with an element. The matrix $\mathbf{N}$ contains the 'shape functions' for the element. The form of these functions for different types of element is discussed below.

The equations of compatibility are now used to obtain the strains inside the element in terms of the nodal displacements. This relationship is normally written in matrix notation:

$$\epsilon = \mathbf{B}\mathbf{a}_e. \tag{3.16}$$

The matrix $\mathbf{B}$ is sometimes referred to as the 'strain matrix', but is more often simply referred to as the '*B* matrix'.

The next step is to use the elastic stress–strain relation for the material ($\sigma = \mathbf{D}\epsilon$) to express the stresses inside the elements in terms of the nodal displacements:

$$\sigma = \mathbf{D}\mathbf{B}\mathbf{a}_e. \tag{3.17}$$

The principle of virtual work is now used to find the nodal forces ($\mathbf{F}_e$) which are in equilibrium with this state of internal stress. These nodal forces do not represent actual concentrated forces in the body: rather they represent *resultants* in much the same way as engineers use the concepts of an axial force, shear force and bending moment to describe the state of stress in a beam. A set of virtual nodal displacements applied to the element accompanies a set of virtual strains within the element according to the relation

$$\overset{*}{\epsilon} = \mathbf{B}\overset{*}{\mathbf{a}}_e. \tag{3.18}$$

The principle of virtual work gives

$$\overset{*}{\mathbf{a}}_e^{\mathrm{T}}\mathbf{F}_e = \int_V \overset{*}{\epsilon}^{\mathrm{T}}\sigma \,\mathrm{d}(\mathrm{vol}). \tag{3.19}$$

Substituting for σ and $\overset{*}{\epsilon}$ using (3.17) and (3.18) we obtain

$$\overset{*}{\mathbf{a}}_e^{\mathrm{T}}\mathbf{F}_e = \overset{*}{\mathbf{a}}_e^{\mathrm{T}}\int_V (\mathbf{B}^{\mathrm{T}}\mathbf{D}\mathbf{B})\,\mathrm{d}(\mathrm{vol})\,\mathbf{a}_e,$$

and $\overset{*}{\mathbf{a}}_e^{\mathrm{T}}$ can be cancelled to give

$$\begin{aligned}\mathbf{F}_e &= \int_V (\mathbf{B}^{\mathrm{T}}\mathbf{D}\mathbf{B})\,\mathrm{d}(\mathrm{vol})\,\mathbf{a}_e \\ &= \mathbf{K}\mathbf{a}_e,\end{aligned} \tag{3.20}$$

where

$$K = \int_V (\mathbf{B}^{\mathrm{T}}\mathbf{D}\mathbf{B})\,\mathrm{d}(\mathrm{vol})$$

is the element of stiffness matrix.

The equivalent nodal forces $\mathbf{F}_e$ balance loads due to self-weight and boundary stresses – taking into account overall equilibrium, the resulting equation is

$$\begin{aligned}\int_V (\mathbf{B}^{\mathrm{T}}\mathbf{D}\mathbf{B})\,\mathrm{d}(\mathrm{vol})\,\mathbf{a}_e = &\int_V \mathbf{N}^{\mathrm{T}}\mathrm{w}\,\mathrm{d}(\mathrm{vol}) \\ &+ \int_S \mathbf{N}^{\mathrm{T}}\tau\,\mathrm{d}(\mathrm{area}),\end{aligned} \tag{3.21}$$

where

$$\tau = \begin{bmatrix} \sigma_n \\ \tau_{nt} \end{bmatrix}$$

represents normal and shear stresses acting on an element boundary. Although these equations have been developed for a single element, we could equally well have considered a whole mesh of elements in deriving them. Of course, one has to use the **N** and **B** matrices for each element in turn when performing an integration over the whole mesh.

3.5.2 Example: a plane truss element

Fig. 3.11 shows one member of a plane truss of length L, inclined at an angle α to the x axis. The nodal degrees of freedom are the displacements in the x and y directions at the two ends of the element, d_{x1}, d_{y1}, d_{x2} and d_{y2}.

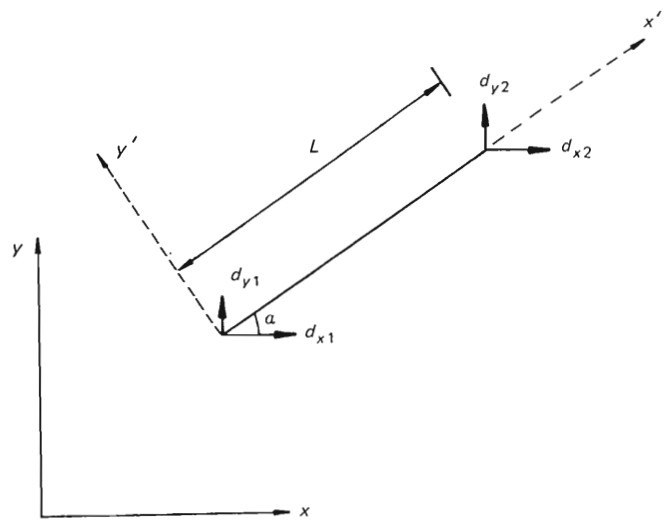


Fig. 3.11 – Nodal degrees of freedom for plane truss element

In calculating the strain in this element, we are only interested in the displacements along the direction of the element and so we define an axis system local to the element, (x', y'), with the x' axis coincident with the direction of the member. The displacement a distance x' along the element is given by

$$d'_x = (1 - x'/L)\, d'_{x1} + (x'/L)\, d'_{x2}. \tag{3.22}$$

To obtain the element stiffness matrix we need to obtain this expression in terms of degrees of freedom d_{x1}, d_{y1}, d_{x2} and d_{y2}. This is achieved by noting that

$$d'_x = d_x \cos\alpha + d_y \sin\alpha \tag{3.23}$$

(which follows from a simple consideration of geometry).

Making this substitution we obtain

$$[d'_x] = \begin{bmatrix} (1 - x'/L)\cos\alpha & (1 - x'/L)\sin\alpha \\ (x'/L)\cos\alpha & (x'/L)\sin\alpha \end{bmatrix} \begin{bmatrix} d_{x1} \\ d_{y1} \\ d_{x2} \\ d_{y2} \end{bmatrix} \tag{3.24}$$

which is the same form as (3.15) above. The **B** matrix is obtained by differentiating this equation:

$$\epsilon'_x = -\frac{\mathrm{d}(d'_x)}{\mathrm{d}x'}$$

and is given by

$$[-C/L \quad -S/L \quad C/L \quad S/L],$$

where $C = \cos\alpha$ and $S = \sin\alpha$.

The **D** matrix here simply reduces to Young's modulus, E ($\sigma'_x = E\,\epsilon'_x$).

$$\int_V (\mathbf{B}^{\mathrm{T}}\mathbf{D}\mathbf{B})\,\mathrm{d}(\mathrm{vol}) = \frac{AE}{L}\begin{bmatrix} C^2 & CS & -C^2 & -CS \\ CS & S^2 & -CS & -S^2 \\ -C^2 & -CS & C^2 & CS \\ -CS & -S^2 & CS & S^2 \end{bmatrix}.$$

The stiffness matrix of this element is normally obtained using a direct equilibrium approach. We have applied the general form (3.20).

3.5.3 Example: constant strain triangle

Fig. 3.12 shows the simplest triangular finite element for continuum analysis. The nodal degrees of freedom are the displacements at the vertices of the triangle, d_{x1}, d_{y1}, d_{x2}, d_{y2}, d_{x3} and d_{y3}. The displacement at some point in the element is assumed to have a linear variation:

$$d_x = c_0 + c_1 x + c_2 y,$$

$$d_y = c_3 + c_4 x + c_5 y.$$

The coefficients c_0, c_1, etc. are found by substituting the co-ordinates of the three nodal points into these expressions. Solving the resulting sets of simultaneous equations we obtain

$$d_x = \frac{x}{h} d_{x1} + \frac{y}{h} d_{x2} + \left(1 - \frac{x}{h} - \frac{y}{h}\right) d_{x3},$$

$$d_y = \frac{x}{h} d_{y1} + \frac{y}{h} d_{y2} + \left(1 - \frac{x}{h} - \frac{y}{h}\right) d_{y3}.$$

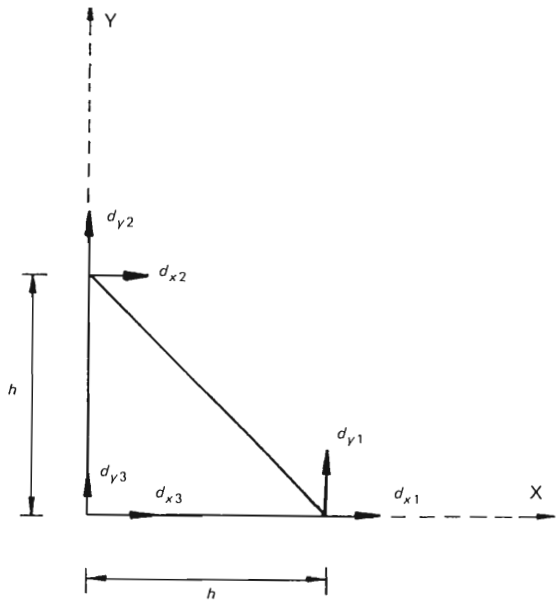


Fig. 3.12 – Constant strain triangle

Thus the shape function matrix **N** is given by

$$\begin{bmatrix} x/h & 0 & y/h & 0 & (1-x/h-y/h) & 0 \\ 0 & x/h & 0 & y/h & 0 & (1-x/h-y/h) \end{bmatrix}.$$

Applying the normal definitions of strains (1.4), (1.5) and (1.7), the **B** matrix is given by

$$\begin{bmatrix} -1/h & 0 & 0 & 0 & 1/h & 0 \\ 0 & 0 & 0 & -1/h & 0 & 1/h \\ 0 & -1/h & -1/h & 0 & 1/h & 1/h \end{bmatrix}.$$

For a plane strain problem the **D** matrix relating σ to ϵ is given by

$$\frac{E}{(1-2\nu)(1+\nu)} \begin{bmatrix} 1-\nu & \nu & 0 \\ \nu & 1-\nu & 0 \\ 0 & 0 & 0.5-\nu \end{bmatrix}.$$

Calculating the element stiffness matrix is a simple matter of calculating the matrix product $\mathbf{B}^T\mathbf{DB}$ times the area of the element ($h^2/2$), since the terms of all these matrices are constant.

The resulting matrix is

$$\begin{bmatrix} a & 0 & 0 & \nu & -a & -\nu \\ 0 & b & b & 0 & -b & -b \\ 0 & b & b & 0 & -b & -b \\ \nu & 0 & 0 & a & -\nu & -a \\ -a & -b & -b & -\nu & c & 1/2 \\ -\nu & -b & -b & -a & 1/2 & c \end{bmatrix} \frac{E}{2(1+\nu)(1-2\nu)}$$

where $a = 1-\nu$, $b = 0.5-\nu$ and $c = 1.5-2\nu$.

Note that the terms of this matrix are independent of the dimensions of the element – a property of all element stiffness matrices for plane strain and plane stress analysis that can be expected on physical grounds.

3.5.4 Higher-order elements

The second element presented in the previous section, usually known as the CST (the Constant Strain Triangle), was the first element formulated for continuum analysis (Turner *et al.*, 1956). Although it has the virtue of simplicity it is currently not regarded as a good choice of element for general use in analyses. This is because a large number of CST elements are required to obtain a sufficiently accurate representation of non-constant stress fields. Irons and Ahmad (1980) demonstrate a number of cases where this element gives poor results, even with apparently fine meshes.

Elements with a higher-order variation of displacement (and hence strain) have the advantage that fewer elements are needed to obtain a sufficiently accurate solution to problems. However, a higher-order element is more difficult to program, more difficult for a program user to understand and uses more computer resources than lower-order elements. Despite these disadvantages it is generally accepted that the balance of advantage in terms of both computational efficiency and ease of use favours the higher-order elements. The element usually used for plane strain analyses by CRISP is the linear strain triangle (Fig. 3.13). Whereas the constant strain triangle has a displacement field which has a linear variation in all directions:

$$d_x = c_0 + c_1 x + c_2 y,$$

$$d_y = c_3 + c_4 x + c_5 y,$$

the linear strain triangle (or LST) has a displacement field which has a quadratic variation in each direction:

$$d_x = c_0 + c_1 x + c_2 y + c_3 x^2 + c_4 xy + c_5 y^2,$$

$$d_y = c_6 + c_7 x + c_8 y + c_9 x^2 + c_{10} xy + c_{11} y^2.$$

It is convenient to express the shape functions for higher-order elements (such as the LST) in terms of co-ordinate systems which are local to the element

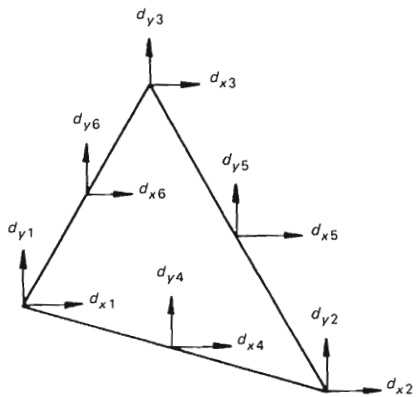


Fig. 3.13 – Nodal degrees of freedom for linear strain triangle

concerned. The remainder of this section explains the basic techniques for doing this.

First consider the bar element presented in section 3.5.2. To simplify the demonstration of a local co-ordinate system, we align the element with the global x axis. If the co-ordinates of the ends of the bar are $(x_1, 0)$ and $(x_2, 0)$ then the shape functions for axial displacement of a point $(x, 0)$ on the bar are

$$d_x = (x_2 - x)/(x_2 - x_1)\, d_{x1} + (x - x_1)/(x_2 - x_1)\, d_{x2}.$$

We now introduce the same local co-ordinate system that was adopted in section 3.2.2 for the numerical integration rules:

$$\xi = (2x - (x_1 + x_2))/(x_2 - x_1).$$

Thus $\xi = -1$ when $x = x_1$ and $\xi = 1$ when $x = x_2$ (at the midpoint of the element $\xi = 0$). The transformation from the global co-ordinate system to the local one involves a linear stretch and a translation.

In terms of the local co-ordinate, the shape functions are

$$d_x = 0.5\,(1 - \xi)\, d_{x1} + 0.5\,(1 + \xi)\, d_{x2}.$$

The same functions are used to transform the local co-ordinate to the global one:

$$x = 0.5\,(1 - \xi)\, x_1 + 0.5\,(1 + \xi)\, x_2$$

(the functions are the same because for the bar the displacement varies linearly along the element and the axis transformation is also linear). It should now be apparent that the advantage of using the local co-ordinate system is that the shape functions of all linear bar elements are now given by the same expression. The use of local co-ordinates requires some small modifications to the way the stiffness matrix is calculated. Before discussing these changes, the two most common forms of local co-ordinates for two-dimensional elements are described.

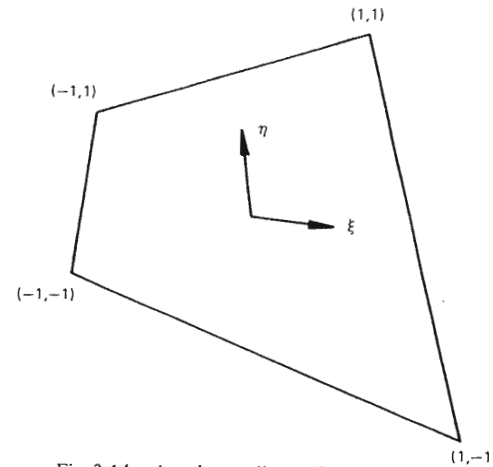


Fig. 3.14 – Local co-ordinates for quadrilateral elements

Fig. 3.14 shows the system of local co-ordinates (ξ, η) appropriate for rectangular or quadrilateral elements. The local co-ordinates of the vertices of the rectangle or quadrilateral are (1, 1), (−1, 1), (−1, −1) and (1, −1). Transformation from local to global co-ordinates is described by the equations

$$x = 0.25(1 + \xi)(1 + \eta)x_1 + 0.25(1 - \xi)(1 + \eta)x_2 + 0.25(1 - \xi)(1 - \eta)x_3 + 0.25(1 + \xi)(1 - \eta)x_4,$$

$$y = 0.25(1 + \xi)(1 + \eta)y_1 + 0.25(1 - \xi)(1 + \eta)y_2 + 0.25(1 - \xi)(1 - \eta)y_3 + 0.25(1 + \xi)(1 - \eta)y_4.$$

The similarity between this system and the one-dimensional system should be apparent.

Fig. 3.15 shows the system of local co-ordinates appropriate for triangular elements. A point within a triangle is defined by three co-ordinates (L_1, L_2, L_3). Only two of these co-ordinates are independent since

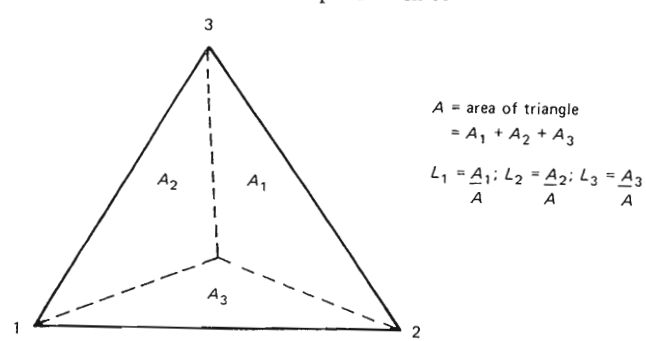


Fig. 3.15 – Triangular co-ordinates

$$A_1 + A_2 + A_3 = A$$

and hence

$$L_1 + L_2 + L_3 = 1.$$

The advantage of using three co-ordinates for triangular elements is that expressions for the shape functions are symmetrical with respect to the nodes. Transformation from local to global co-ordinates is given by the equations

$$x = L_1 x_1 + L_2 x_2 + L_3 x_3,$$

$$y = L_1 y_1 + L_2 y_2 + L_3 y_3.$$

The elements provided in CRISP are triangular (see Fig. 4.1). Triangular elements possess the (probably small) theoretical advantage over quadrilaterals that they give the same variation in displacement in all directions over the element. This is because the shape functions contain complete polynomial expansions of x and y, and unlike quadrilaterals do not have extra 'junk' terms. In some situations, analyses with triangular elements have succeeded where quadrilateral elements have come to grief (e.g. recent work on computing elastic–perfectly-plastic collapse loads (Sloan and Randolph, 1982). The shape functions for the CST element are the triangular co-ordinates, i.e. $N_1 = L_1$, $N_2 = L_2$ and $N_3 = L_3$. The shape functions for the LST element are $N_1 = (2L_1 - 1)L_1$, $N_2 = (2L_2 - 1)L_2$, $N_3 = (2L_3 - 1)L_3$, $N_4 = 4L_1L_2$, $N_5 = 4L_2L_3$ and $N_6 = 4L_3L_1$. Shape functions for higher-order elements can be obtained by a simple recurrence relation. While it is convenient to formulate the triangular elements in terms of triangular co-ordinates, it is necessary at some point to change to the (ξ, η) local co-ordinates, when the substitutions $L_1 = \xi$, $L_2 = \eta$ and $L_3 = 1 - \xi - \eta$ are made.

It is straightforward to calculate the derivatives of these functions with respect to the local co-ordinates. Integrating functions within the triangular and quadrilateral areas is also straightforward in terms of the local co-ordinates. However, in calculating the stiffness matrix it is necessary to obtain derivatives with respect to the global co-ordinates (i.e. when calculating terms in the B matrix). The *Jacobian* matrix is used to transform between derivatives with respect to local and global co-ordinates (see for example the text by Maxwell, 1954). The Jacobian matrix arises from the chain rule of partial differentiation:

$$\frac{\partial f}{\partial \xi} = \frac{\partial f}{\partial x}\frac{\partial x}{\partial \xi} + \frac{\partial f}{\partial y}\frac{\partial y}{\partial \xi},$$

$$\frac{\partial f}{\partial \eta} = \frac{\partial f}{\partial x}\frac{\partial x}{\partial \eta} + \frac{\partial f}{\partial y}\frac{\partial y}{\partial \eta},$$

and can be written:

$$\begin{bmatrix} \dfrac{\partial f}{\partial \xi} \\ \dfrac{\partial f}{\partial \eta} \end{bmatrix} = \mathbf{J} \begin{bmatrix} \dfrac{\partial f}{\partial x} \\ \dfrac{\partial f}{\partial y} \end{bmatrix}$$

where the Jacobian matrix $\mathbf{J}$ is given by

$$\begin{bmatrix} \dfrac{\partial x}{\partial \xi} & \dfrac{\partial y}{\partial \xi} \\ \dfrac{\partial x}{\partial \eta} & \dfrac{\partial y}{\partial \eta} \end{bmatrix}.$$

In forming the terms of the $\mathbf{B}$ matrix, the Jacobian matrix of the inverse relation is required (i.e. local → global rather than global → local). It is computationally easier to calculate $\mathbf{J}$ and then the terms of $\mathbf{J}^{-1}$ and this course is pursued in CRISP (see Chapter 7). The other standard result which is used in integrating the terms of the stiffness matrix is

$$\int f \, dx \, dy = \int f \det(J) \, d\xi \, d\eta$$

where $\det(J)$ is the determinant of the matrix J.

As indicated earlier, numerical integration is used to calculate the terms of the element stiffness matrices. For two-dimensional numerical integration there are 'integration points' within each element where the terms of the matrix product $\mathbf{B}^T\mathbf{D}\mathbf{B}$ are calculated.

3.5.5 One-dimensional quadratic element

As an example of a higher-order element we show how the stiffness matrix of a one-dimensional element with a quadratic variation of displacement (and, hence, a linear distribution of strain) can be calculated. This element can be regarded a a three-noded bar element. Alternatively, it could be regarded as suitable for a (rather simple) analysis of layers of soil where there is no straining in either of the horizontal directions. The element is shown in Fig. 3.16. The terms of the $\mathbf{N}$ matrix are given by

$$[0.5\xi(\xi - 1) \quad 0.5\xi(\xi + 1) \quad (1 - \xi^2)].$$

The transformation between local and global co-ordinates when forming derivatives is quite simple in the one-dimensional case:

$$\epsilon_x = -\frac{\partial d_x}{\partial x} = -\frac{\partial d_x}{\partial \xi}\frac{d\xi}{dx}.$$

Hence the terms of the $\mathbf{B}$ matrix are

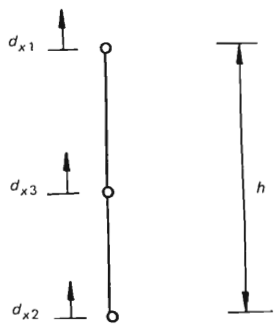


Fig. 3.16 – One-dimensional quadratic element

$$\frac{2}{h}\left[(0.5-\xi) \quad -(0.5+\xi) \quad 2\xi\right]$$

(noting that $d\xi/dx = 2/h$).

Now since $\epsilon_y = \epsilon_z = 0$, $\sigma_x = (E(1-\nu)/((1+\nu)(1-2\nu)))\epsilon_x$. Thus **D** is in this case a square matrix containing precisely one term (which is usually known as the one-dimensional modulus). The stiffness matrix for this element is therefore given by the integral of the following matrix over the volume of the element:

$$\frac{4D}{h^2}\begin{bmatrix} (0.5-\xi)(0.5-\xi) & -(0.5-\xi)(0.5+\xi) & 2\xi(0.5-\xi) \\ -(0.5+\xi)(0.5-\xi) & (0.5+\xi)(0.5+\xi) & -2\xi(0.5+\xi) \\ 2\xi(0.5-\xi) & -2\xi(0.5+\xi) & 4\xi\xi \end{bmatrix}.$$

To perform this integration, each term in the matrix is integrated between limits $\xi = -1$ and $\xi = 1$, and each resulting term is multiplied by $h/2$ (the equivalent of det (**J**) in this case). The matrix resulting from this process is

$$\frac{DA}{3h}\begin{bmatrix} 7 & 1 & -8 \\ 1 & 7 & -8 \\ -8 & -8 & 16 \end{bmatrix}$$

where A is the area of the column of soil.

3.5.6 Approximation and accuracy in the displacement method

Engineers sometimes regard displacement finite elements as being connected only at the nodal points in a mesh. This is not a good conceptual picture of how finite elements behave. Straining displacement elements results in a deformation pattern similar to that shown in Fig. 3.17(a) rather than Fig. 3.17(b) (i.e. no gaps open up between element sides). This is because the displacement shape functions are chosen so that there is continuity of displacements between

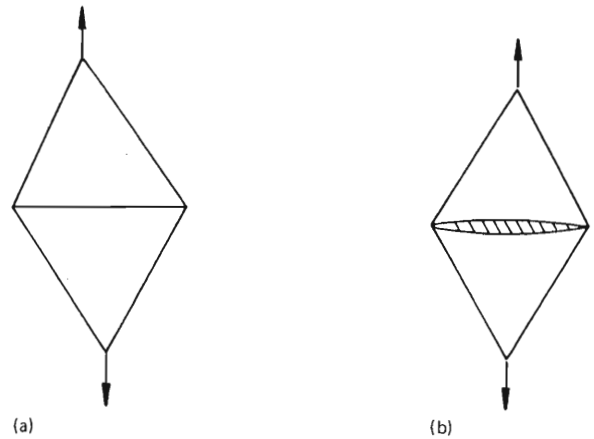


Fig. 3.17 – Displacement finite elements deform as in (a) not (b) (displacements are compatible: no gaps open up)

elements. On the other hand, although the strains will be continuous within elements, there will usually be a discontinuity of strains between adjacent elements.

The stress field in an element will be continuous – but may not satisfy the differential equations of equilibrium. Except for very simple problems, stresses on either side of element boundaries will not be equal. Equilibrium is satisfied, however, in an average sense through the equilibrium equations at nodal points where the resultant forces equivalent to internal stress fields balance resultant forces equivalent to external tractions and body forces.

The extent to which local stresses appear not to be in equilibrium gives some idea of the accuracy of the solution.

3.6 FINITE ELEMENTS FOR CONSOLIDATION ANALYSIS

3.6.1 The basic equations

In this section the basic matrix equations for consolidation analysis by finite elements are derived. The starting point is the differential equations of equilibrium and compatibility that were described in the first chapter. The equations will be developed for a two-dimensional analysis. To formulate a three-dimensional analysis it is merely necessary to add the extra terms for variation in the z direction. For the sake of completeness we repeat the equilibrium equations:

$$\frac{\partial \sigma_x}{\partial x} + \frac{\partial \tau_{yx}}{\partial y} = w_x, \qquad (3.12\ bis)$$

$$\frac{\partial \tau_{xy}}{\partial x} + \frac{\partial \sigma_y}{\partial y} = w_y. \qquad (3.13\ bis)$$

The two-dimensional differential equation of continuity is

$$\frac{k_x}{\gamma_w} \cdot \frac{\partial^2 \bar{u}}{\partial x^2} + \frac{k_y}{\gamma_w} \cdot \frac{\partial^2 \bar{u}}{\partial y^2} + \frac{\partial v}{\partial t} = 0. \qquad (3.25)$$

To obtain the finite element matrix equations we can apply Galerkin's weighted residual method to the equilibrium equations and to the continuity equation in turn. In section 3.3 it was shown that for the equilibrium equations the resulting equation was equivalent to the principle of virtual work. We now show that performing the same kind of operation on the continuity equation yields another 'virtual principle'. The first step is to multiply the continuity equation by an arbitrary scalar which can vary with x and y. We identify this scalar with an imaginary or virtual pore pressure. Thus (3.25) is replaced by

$$\int_V \overset{*}{u} \cdot \left[\frac{k_x}{\gamma_w} \frac{\partial^2 \bar{u}}{\partial x^2} + \frac{k_y}{\gamma_w} \frac{\partial^2 \bar{u}}{\partial y^2} + \frac{\partial v}{\partial t} \right] \mathrm{d}\,(\mathrm{vol}) = 0. \qquad (3.26)$$

Zienkiewicz–Green theorem is now applied to this equation:

$$- \int_V \left[\frac{k_x}{\gamma_w} \frac{\partial \overset{*}{u}}{\partial x} \frac{\partial \bar{u}}{\partial x} + \frac{k_y}{\gamma_w} \frac{\partial \overset{*}{u}}{\partial y} \frac{\partial \bar{u}}{\partial y} \right] \mathrm{d}\,(\mathrm{vol})$$

$$- \int_S \overset{*}{u} v_n \,\mathrm{d}\,(\mathrm{area}) + \int_V \overset{*}{u} \frac{\partial v}{\partial t} \,\mathrm{d}\,(\mathrm{vol}) = 0 \qquad (3.27)$$

(where v_n is the artificial seepage velocity normal to the boundary). It is this equation that could be regarded as the 'principle of virtual power' and could form the starting point for obtaining the finite element equations, in much the same way as the principle of virtual work can be used to obtain the finite element equations for stress analysis.

We now introduce the finite element discretisation of the problem. The displacements are assumed to vary over a finite element mesh according to

$$\mathbf{d} = \mathbf{N}\mathbf{a}, \qquad (3.28)$$

and the excess pore pressures are assumed to vary over the same mesh according to

$$\bar{u} = \bar{\mathbf{N}}\mathbf{b}. \qquad (3.29)$$

Note that different shape functions are indicated for displacement (matrix $\mathbf{N}$) and excess pore pressure (matrix $\bar{\mathbf{N}}$). For example the displacement may vary in a quadratic fashion and pore pressure in a linear fashion over one element. The virtual excess pore pressure is assumed to vary according to the same shape functions as the excess pore pressures:

$$\overset{*}{u} = \bar{\mathbf{N}}\overset{*}{\mathbf{b}}. \qquad (3.30)$$

As usual the strains are given by

$$\epsilon = \mathbf{B}\mathbf{a}, \qquad (3.31)$$

and the gradient of the excess pore pressure is given by

$$\begin{bmatrix} \dfrac{\partial \bar{u}}{\partial x} \\ \dfrac{\partial \bar{u}}{\partial y} \end{bmatrix} = \mathbf{E}\,\mathbf{b}, \qquad (3.32)$$

where the terms of the $\mathbf{E}$ matrix are obtained by differentiating $\bar{\mathbf{N}}$. A vector $\mathbf{m}$ is defined:

$$\mathbf{m} = \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}, \qquad (3.33)$$

such that

$$\sigma = \sigma' + \mathbf{m}u, \qquad (3.34)$$

and

$$v = \mathbf{m}^{\mathrm{T}}\epsilon. \qquad (3.35)$$

Substituting into (3.27) we have

$$\overset{*}{\mathbf{b}}^{\mathrm{T}} \int_V \bar{\mathbf{N}}^{\mathrm{T}}\mathbf{m}^{\mathrm{T}}\mathbf{B} \,\mathrm{d}\,(\mathrm{vol}) \frac{\mathrm{d}(\mathbf{a})}{\mathrm{d}t} - \overset{*}{\mathbf{b}}^{\mathrm{T}} \int_V \mathbf{E}^{\mathrm{T}}\mathbf{k}\mathbf{E}/\gamma_w \,\mathrm{d}\,(\mathrm{vol})\,\mathbf{b}$$

$$= \overset{*}{\mathbf{b}}^{\mathrm{T}} \int_S \bar{\mathbf{N}}^{\mathrm{T}} v_n \,\mathrm{d}\,(\mathrm{area}) \qquad (3.36)$$

where k is a permeability matrix:

$$\begin{bmatrix} k_x & 0 \\ 0 & k_y \end{bmatrix}.$$

The virtual pore pressure can be cancelled from this equation, and making the substitutions

$$\mathbf{L} = \int_V \mathbf{B}^{\mathrm{T}}\mathbf{m}\bar{\mathbf{N}} \,\mathrm{d}\,(\mathrm{vol}) \quad \text{and} \quad \Phi = \int_V \mathbf{E}^{\mathrm{T}}\mathbf{k}\mathbf{E}/\gamma_w \,\mathrm{d}\,(\mathrm{vol})$$

we obtain

$$\mathbf{L}^{\mathrm{T}} \frac{\mathrm{d}(\mathbf{a})}{\mathrm{d}t} - \Phi\,\mathbf{b} = \int_S \bar{\mathbf{N}}^{\mathrm{T}} v_n \,\mathrm{d}\,(\mathrm{area}).$$

This is a first-order differential equation which we integrate with respect to time, from time t to time $t + \Delta t$:

$$\int_t^{t+\Delta t} \mathbf{L}^{\mathrm{T}} \frac{\mathrm{d}(\mathbf{a})}{\mathrm{d}t}\,\mathrm{d}t - \Phi \int_t^{t+\Delta t} \mathbf{b}\,\mathrm{d}t = \int_t^{t+\Delta t} \int_S \bar{\mathbf{N}}^{\mathrm{T}} v_{\mathrm{n}}\,\mathrm{d}\,(\text{area})\,\mathrm{d}t. \tag{3.37}$$

In performing this integration we make the approximation

$$\int_t^{t+\Delta t} \mathbf{b}\,\mathrm{d}t = \{(1-\theta)\,\mathbf{b}_1 + \theta\,\mathbf{b}_2 \quad \Delta t\}$$

where $\mathbf{b}_1 = \mathbf{b}(t)$ and $\mathbf{b}_2 = \mathbf{b}(t + \Delta t)$. The value of θ defines the way that $\mathbf{b}$ varies during the time interval; for example, $\theta = \frac{1}{2}$ corresponds to a linear variation and the trapezoidal integration rule.

A similar approximation is made for the integration of v_{n}, and after substitution (3.37) becomes

$$\mathbf{L}^{\mathrm{T}}\,[\mathbf{a}]_t^{t+\Delta t} - \Phi\{(1-\theta)\,\mathbf{b}_1 + \theta\,\mathbf{b}_2\}\,\Delta t = \int \mathbf{N}^{\mathrm{T}}\,\{(1-\theta)\,v_{\mathrm{n}1} + \theta\,v_{\mathrm{n}2}\}\,\Delta t\,\mathrm{d}\,(\text{area}). \tag{3.38}$$

Booker and Small (1975) consider the stability of integration schemes using different values of θ and show that for stability, $\theta \geqslant \frac{1}{2}$. We have adopted a value of $\theta = 1$. Making that substitution in (3.38), and defining $\Delta\mathbf{a} = \mathbf{a}(t + \Delta t) - \mathbf{a}(t)$ and $\Delta\mathbf{b} = \mathbf{b}_2 - \mathbf{b}_1$, we arrive at

$$\mathbf{L}^{\mathrm{T}}\,\Delta\mathbf{a} - \Phi\,\Delta t \cdot \Delta\mathbf{b} = \Phi\,\Delta t \cdot \mathbf{b}_1 + \int_S \mathbf{N}^{\mathrm{T}} v_{\mathrm{n}2}\,\Delta t\,\mathrm{d}\,(\text{area}). \tag{3.39}$$

Now we turn to the equilibrium equations. Rather than start from the differential form we make direct use of the incremental form of virtual work:

$$\int \overset{*}{\boldsymbol{\epsilon}}^{\mathrm{T}} \Delta\boldsymbol{\sigma}\,\mathrm{d}\,(\text{vol}) = \int \overset{*}{\mathbf{d}}^{\mathrm{T}}\,\Delta\boldsymbol{\tau}\,\mathrm{d}\,(\text{area}) + \int \overset{*}{\mathbf{d}}^{\mathrm{T}}\,\Delta\mathbf{w}\,\mathrm{d}\,(\text{vol}). \tag{3.40}$$

Previously we have used the virtual work principle for total stresses. That the incremental form is valid follows from the principle of superposition for linear elastic systems. In fact the incremental form is also valid for non-linear systems, as can be shown by writing the equations in terms of total stresses and subtracting. Now:

$$\Delta\sigma = \Delta\sigma' + \mathbf{m}\,\Delta u,$$

and therefore (noting that $\Delta u = \Delta\bar{u}$)

$$\Delta\sigma = \Delta\sigma' + \mathbf{m}\,\Delta\bar{u}.$$

Using this relation, and making the usual finite element substitutions:

$$\Delta\epsilon = B\,\Delta\mathbf{a},$$
$$\overset{*}{\mathbf{d}} = N\,\overset{*}{\mathbf{a}},$$
$$\Delta\bar{u} = \bar{N}\,\Delta\mathbf{b},$$

we obtain

$$\overset{*}{\mathbf{a}}^{\mathrm{T}} \int_v \mathbf{B}^{\mathrm{T}}\mathbf{D}\mathbf{B}\,\mathrm{d}\,(\text{vol})\,\Delta\mathbf{a} + \overset{*}{\mathbf{a}}^{\mathrm{T}} \int (\mathbf{B}^{\mathrm{T}}\mathbf{m}\bar{\mathbf{N}})\,\mathrm{d}\,(\text{vol})\,.\,\Delta\mathbf{b} = \overset{*}{\mathbf{a}}^{\mathrm{T}} \int_S \mathbf{N}^{\mathrm{T}}\,.\,\Delta\boldsymbol{\tau}\,\mathrm{d}\,(\text{area}). \tag{3.41}$$

$\overset{*}{\mathbf{a}}^{\mathrm{T}}$ can be cancelled, and using the notation already established:

$$\mathbf{K}\,\Delta\mathbf{a} + \mathbf{L}\,\Delta\mathbf{b} = \int_S \mathbf{N}^{\mathrm{T}}\,\Delta\boldsymbol{\tau}\,\mathrm{d}\,(\text{area}) \tag{3.42}$$

where

$$\mathbf{K} = \int [\mathbf{B}^{\mathrm{T}}\mathbf{D}\mathbf{B}]\,\mathrm{d}\,(\text{vol}).$$

Equations (3.39) and (3.42) can be used to establish a solution at time $t + \Delta t$ from the solution at time t. Thus the solution can be 'marched forward' in time from $t = 0$. In summarising, the equations can be written:

$$\begin{bmatrix} \mathbf{K} & \mathbf{L} \\ \mathbf{L}^{\mathrm{T}} & -\Phi\Delta t \end{bmatrix} \begin{bmatrix} \Delta\mathbf{a} \\ \Delta\mathbf{b} \end{bmatrix} = \begin{bmatrix} \Delta\mathbf{r}_1 \\ \Delta\mathbf{r}_2 \end{bmatrix}. \tag{3.43}$$

It is normal to refer to the square matrix in (3.43) as a stiffness matrix, even though it multiplies a vector of mixed displacement and pore pressure variables. The first equation in (3.43) represents approximate satisfaction of the equilibrium equations and the second equation approximate satisfaction of the continuity equation. The right-hand-side term $\Delta\mathbf{r}_1$ consists of the normal finite element incremental load terms. The right-hand-side term $\Delta\mathbf{r}_2$ consists of a load term corresponding to a prescribed seepage on the boundary:

$$\int_S \bar{\mathbf{N}} v_{\mathrm{n}2}\,\mathrm{d}\,(\text{area})$$

and an additional term ($\Phi\Delta t\,.\,\mathbf{b}_1$) which is calculated as the solution proceeds.

3.6.2 A finite element program for consolidation analysis

This section shows how the matrix equations derived in the previous section are implemented in a computer program. We call this program 'TINY'. The name is appropriate because the program has been set up to solve problems with a maximum of six elements. Of course, it would not be difficult to lift this restriction by making some modifications to the program. The program performs

a one-dimensional consolidation analysis using the element shown in Fig. 3.18. The basic displacement element is the three-noded one in section 3.5.5, here supplemented by a linear variation in excess pore pressure. Here is a 'subroutine hierarchy' for TINY, showing the order in which the various subroutines are called and their relation to one another:

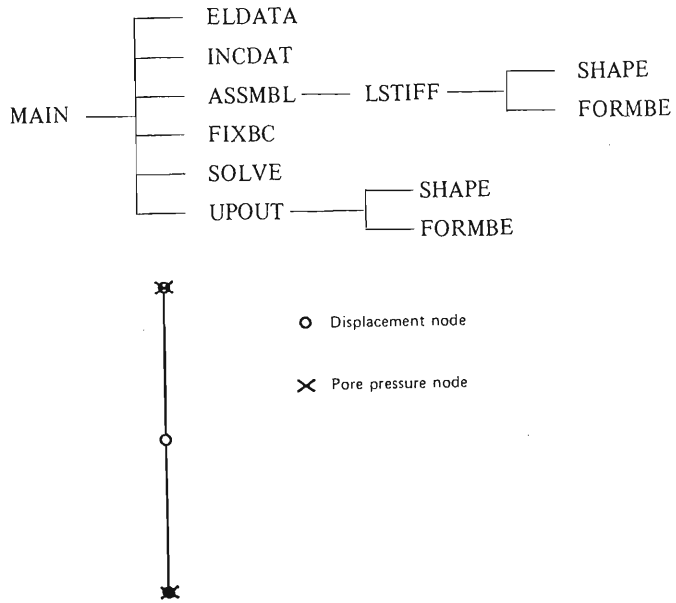


Fig. 3.18 – Finite element used by TINY for one-dimensional consolidation

The overall sequence of operations performed by the program is the same as for the 'springs' program described earlier. However, routines ASSMBL, FIXBC, SOLVE and UPOUT are now in a loop and are executed for every time step or increment of the analysis. Routine ASSMBL assembles the global 'stiffness' matrices using the 5×5 element matrices. The main controlling routine is below, and is followed by cross-referenced explanations, a style we shall use for all other routines:

Routine MAIN

```
      CHARACTER*80 TITLE                                                  MAIN   1
      COMMON /DAT/ GP(2),W(2),LIN(3)                                      MAIN   2
      COMMON GAMMAW,H(6),YM(6),POISS(6),PERM(6),DTIME(40)                 MAIN   3
      COMMON SHFND(3),DSD(3),CARDSD(3),SHFNP(2),DSP(2),CARDSP(2)          MAIN   4
      COMMON B(3),E(2),DB(3),ES(5,5),FI(2,2),UAXS(2),ERHS(5)              MAIN   5
      COMMON ST(20,20),RHS(20),DISPA(20),VARINT(2,2,6),BC(4),XI,BTIME     MAIN   6
      COMMON NCONN(3,6),NW(13),IBC(4),NINC,NDF,NEL,NG,INC,NE              MAIN   7
      COMMON L5,L6                                                        MAIN   8
      L5=5                                                                MAIN   9
```



```
      L6=1                                                                MAIN  10
      NG=2                                                                MAIN  11
      READ(L5,100) TITLE                                                  MAIN  12
  100 FORMAT(A)                                                           MAIN  13
      WRITE(L6,200) TITLE                                                 MAIN  14
  200 FORMAT(1X,20(4H****)/1X,A/1X,20(4H****))                            MAIN  15
      CALL ELDATA                                                         MAIN  16
      READ(L5,*) NOINCB                                                   MAIN  17
      WRITE(L6,201) NOINCB                                                MAIN  18
  201 FORMAT(30H NUMBER OF INCREMENT BLOCKS = ,I5)                        MAIN  19
      CALL ZEROR1(DISPA,20)                                               MAIN  20
      CALL ZEROR3(VARINT,2,2,6)                                           MAIN  21
      DO 80 INCB=1,NOINCB                                                 MAIN  22
      WRITE(L6,202) INCB                                                  MAIN   2
  202 FORMAT(1X,20(4H====)/16H INCREMENT BLOCK,I5/1X,20(4H====))          MAIN   2
      CALL INCDAT                                                         MAIN  25
      DO 70 I=1,NINC                                                      MAIN  26
      INC=I                                                               MAIN  27
      WRITE(L6,203) INC                                                   MAIN  28
  203 FORMAT(1X,20(4H++++)/10H INCREMENT,I5/1X,20(4H++++))                MAIN  29
      CALL ASSMBL                                                         MAIN  30
      CALL FIXBC                                                          MAIN  31
      CALL SOLVE                                                          MAIN  32
      CALL UPOUT                                                          MAIN  33
   70 CONTINUE                                                            MAIN  34
   80 CONTINUE                                                            MAIN  35
      STOP                                                                MAIN  36
      END                                                                 MAIN  37
```

MAIN 12–15 : read and write title for analysis.
MAIN 16 : subroutine ELDATA reads the element properties (geometry and material).
MAIN 17–19 : no. of increment blocks.
MAIN 20 : initialise cumulative displacement/excess pore pressure array.
MAIN 21 : initialise stresses at integration points.
MAIN 22 : loop on all increment blocks.
MAIN 25 : read no. of increments and the time steps for each increment in the increment block.
MAIN 30 : calculate element stiffness matrix and assemble into global stiffness matrix.
MAIN 31 : apply boundary conditions.
MAIN 32 : solve for unknowns (displacement/excess pore pressure).
MAIN 33 : print out results.
MAIN 34 : end of increment loop.
MAIN 35 : end of increment block loop.

Routine **ELDATA** reads the user's data describing the element properties. The program assumes that the elements are in order, starting from the top (or bottom) of the layer and numbers the nodes accordingly. Nodes at the mid-points of elements have only one (displacement) degree of freedom (**d.o.f.**), while nodes at the ends of elements have two d.o.f. (displacement plus excess pore pressure). The position of a d.o.f. in the vector RHS (which initially holds the load terms, and after solution the nodal incremental displacements and

incremental excess pore pressures) is called the global variable number (g.v.n.). The displacement d.o.f. at node 1 has a g.v.n. of 1; the pressure d.o.f. at node 1 has a g.v.n. of 2; the displacement d.o.f. at node 2 has a g.v.n. of 3; the displacement d.o.f. at node 3 has a g.v.n. of 4; the excess pore pressure d.o.f. at node 3 has a g.v.n. of 5; the displacement d.o.f. at node 4 has a g.v.n. of 6; and so on. Array NW is set up so that NW(I) gives the first g.v.n. associated with node I.

Subroutine ELDATA

```
      SUBROUTINE ELDATA                                                  ELDT   1
      COMMON /DAT/ GP(2),W(2),LIN(3)                                     ELDT   2
      COMMON GAMMAW,H(6),YM(6),POISS(6),PERM(6),DTIME(40)                ELDT   3
      COMMON SHFND(3),DSD(3),CARDSD(3),SHFNP(2),DSP(2),CARDSP(2)         ELDT   4
      COMMON B(3),E(2),DB(3),ES(5,5),FI(2,2),UAXS(2),ERHS(5)             ELDT   5
      COMMON ST(20,20),RHS(20),DISPA(20),VARINT(2,2,6),BC(4),XI,BTIME    ELDT   6
      COMMON NCONN(3,6),NW(13),IBC(4),NINC,NDF,NEL,NG,INC,NE             ELDT   7
      COMMON L5,L6                                                       ELDT   8
      READ(L5,*) GAMMAW                                                  ELDT   9
      WRITE(L6,200) GAMMAW                                               ELDT  10
  200 FORMAT(10H GAMMAW = ,E15.5)                                        ELDT  11
      READ(L5,*) NEL                                                     ELDT  12
      WRITE(L6,201) NEL                                                  ELDT  13
  201 FORMAT(22H NUMBER OF ELEMENTS = ,I5)                               ELDT  14
      DO 10 N=1,NEL                                                      ELDT  15
      READ(L5,*) H(N),YM(N),POISS(N),PERM(N)                             ELDT  16
      WRITE(L6,202) N,H(N),YM(N),POISS(N),PERM(N)                        ELDT  17
  202 FORMAT(1X,I5,4E15.5)                                               ELDT  18
   10 CONTINUE                                                           ELDT  19
      I=1                                                                ELDT  20
      NW(1)=1                                                            ELDT  21
      DO 20 N=1,NEL                                                      ELDT  22
      NCONN(1,N)=2*N-1                                                   ELDT  23
      NCONN(2,N)=2*N+1                                                   ELDT  24
      NCONN(3,N)=2*N                                                     ELDT  25
      NW(I+1)=NW(I)+2                                                    ELDT  26
      NW(I+2)=NW(I+1)+1                                                  ELDT  27
   20 I=I+2                                                              ELDT  28
      NDF=2+3*NEL                                                        ELDT  29
      RETURN                                                             ELDT  30
      END                                                                ELDT  31
```

ELDT 9–11 : read and write unit weight of water.
ELDT 12–14 : read and write no. of elements ($\leqslant 6$).
ELDT 15–19 : read height (or thickness), Young's modulus and Poisson's ratio and permeability.
ELDT 22 : loop on all elements.
ELDT 23–25 : set up element–nodal connectivity list (list of nodes connected to each element).
ELDT 26–27 : set up g.v.n. for the first variable of each node.
ELDT 29 : total no. of d.o.f. (variables).

Routine **INCDAT** reads the data describing the loads etc. associated with each analysis increment. For ease of data preparation, increments are grouped together into increment blocks.

Subroutine INCDAT

```
      SUBROUTINE INCDAT                                                  INCD   1
      COMMON /DAT/ GP(2),W(2),LIN(3)                                     INCD   2
      COMMON GAMMAW,H(6),YM(6),POISS(6),PERM(6),DTIME(40)                INCD   3
      COMMON SHFND(3),DSD(3),CARDSD(3),SHFNP(2),DSP(2),CARDSP(2)         INCD   4
      COMMON B(3),E(2),DB(3),ES(5,5),FI(2,2),UAXS(2),ERHS(5)             INCD   5
      COMMON ST(20,20),RHS(20),DISPA(20),VARINT(2,2,6),BC(4),XI,BTIME    INCD   6
      COMMON NCONN(3,6),NW(13),IBC(4),NINC,NDF,NEL,NG,INC,NE             INCD   7
      COMMON L5,L6                                                       INCD   8
      READ(L5,*) NINC                                                    INCD   9
      WRITE(L6,200) NINC                                                 INCD  10
  200 FORMAT(38H NUMBER OF INCREMENTS IN THIS BLOCK = ,I5)               INCD  11
      READ(L5,*) (DTIME(I),I=1,NINC)                                     INCD  12
      WRITE(L6,201) (DTIME(I),I=1,NINC)                                  INCD  13
  201 FORMAT(16H TIME INCREMENTS/(1X,8E15.5))                            INCD  14
      BTIME=0.0                                                          INCD  15
      DO 10 N=1,NINC                                                     INCD  16
   10 BTIME=BTIME+DTIME(N)                                               INCD  17
      WRITE(L6,203) BTIME                                                INCD  18
  203 FORMAT(33H TOTAL TIME FOR INCREMENT BLOCK =,E15.5)                 INCD  19
      READ(L5,*) IBC,BC                                                  INCD  20
      WRITE(L6,202) IBC,BC                                               INCD  21
  202 FORMAT(20H BOUNDARY CONDITIONS/1X,I7,3I15/1X,4E15.5)               INCD  22
      RETURN                                                             INCD  23
      END                                                                INCD  24
```

INCD 19–11 : read and write no. of increments in the increment block ($\leqslant 40$).
INCD 12–14 : read and write the time steps for each increment.
INCD 16–17 : calculate the total time step for increment block.
INCD 20–22 : read prescribed boundary conditions and fixity codes for first and last nodes.

Routine **ASSMBL** calls LSTIFF which calculates the 'stiffness' matrix for each element and assembles it into the global matrix. It uses the array NW to decide *where* to put the stiffness terms (NW = Node Where) in the global matrix. Array LIN(3) contains the number of d.o.f. associated with each element node (2, 2, 1), and so helps ASSMBL to decide how many rows/columns to slot in. (LIN = eLement INformation, a mini version of the LINFO array in CRISP.) The 'element right-hand-side' terms (ERHS) are slotted in too.

Subroutine ASSMBL

```
      SUBROUTINE ASSMBL                                                  ASML   1
      COMMON /DAT/ GP(2),W(2),LIN(3)                                     ASML   2
      COMMON GAMMAW,H(6),YM(6),POISS(6),PERM(6),DTIME(40)                ASML   3
      COMMON SHFND(3),DSD(3),CARDSD(3),SHFNP(2),DSP(2),CARDSP(2)         ASML   4
      COMMON B(3),E(2),DB(3),ES(5,5),FI(2,2),UAXS(2),ERHS(5)             ASML   5
      COMMON ST(20,20),RHS(20),DISPA(20),VARINT(2,2,6),BC(4),XI,BTIME    ASML   6
      COMMON NCONN(3,6),NW(13),IBC(4),NINC,NDF,NEL,NG,INC,NE             ASML   7
      COMMON L5,L6                                                       ASML   8
      CALL ZEROR2(ST,20,20)                                              ASML   9
      CALL ZEROR1(RHS,20)                                                ASML  10
      DO 60 N=1,NEL                                                      ASML  11
      NE=N                                                               ASML  12
      CALL LSTIFF                                                        ASML  13
```

```
      DO 50 I=1,3                                                    ASML  14
      NODI=NCONN(I,NE)                                               ASML  15
      IN=LIN(I)                                                      ASML  16
      IK=NW(NODI)-1                                                  ASML  17
      IL=2*(I-1)                                                     ASML  18
      DO 50 II=1,IN                                                  ASML  19
      IK=IK+1                                                        ASML  20
      IL=IL+1                                                        ASML  21
      DO 48 J=1,3                                                    ASML  22
      NODJ=NCONN(J,NE)                                               ASML  23
      JN=LIN(J)                                                      ASML  24
      JK=NW(NODJ)-1                                                  ASML  25
      JL=2*(J-1)                                                     ASML  26
      DO 48 JJ=1,JN                                                  ASML  27
      JK=JK+1                                                        ASML  28
      JL=JL+1                                                        ASML  29
   48 ST(IK,JK)=ST(IK,JK)+ES(IL,JL)                                  ASML  30
   50 RHS(IK)=RHS(IK)+ERHS(IL)                                       ASML  31
   60 CONTINUE                                                       ASML  32
      RETURN                                                         ASML  33
      END                                                            ASML  34
```

ASML 9 : initialise global stiffness matrix.
ASML 10 : initialise RHS load vector.
ASML 11 : loop on all elements.
ASML 13 : calculate element stiffness matrix.
ASML 14 : slot element stiffness matrix in global matrix (loop on all rows).
ASML 15 : node no.
ASML 16 : no. of d.o.f. of node.
ASML 17 : global variable number of first d.o.f. of node (= IK + 1).
ASML 18 : index of the first variable of node (= IL + 1).
ASML 19 : loop on all variables of node.
ASML 20 : global variable number.
ASML 21 : local variable number.
ASML 22 : loop on all columns.
ASML 23 : node no.
ASML 24 : no. of d.o.f. of node.
ASML 25 : global variable number of first d.o.f. of node (= JK + 1).
ASML 26 : index of the first variable of node (= JL + 1).
ASML 27 : loop on all variables of node.
ASML 28 : global variable number.
ASML 29 : local variable number.
ASML 30 : slot element stiffness matrix into global matrix.
ASML 31 : assemble element RHS terms into global RHS (load) array.
ASML 32 : end of element loop.

Routine **LSTIFF** calculates the element 'stiffness' matrix for element NE. The loop from statement 12 to statement 30 calculates the component parts of the stiffness matrix using two-point Gaussian numerical integration. The terms of the various matrix products are calculated NG times (NG = no. of Gauss points = 2) and are summed. The $\mathbf{K}$ and $\mathbf{L}^T$ terms go straight into ES, but the

terms of Φ are stored in a separate matrix FI which is used to produce the pore pressure 'loads' from the marching process ($\Phi\Delta t\,.\,\mathbf{b}_1$). Routines SHAPE and FORMBE are used to calculate the terms of the $\mathbf{N}$, $\bar{\mathbf{N}}$, $\mathbf{B}$ and $\mathbf{E}$ matrices at each integration point.

Subroutine LSTIFF

```
      SUBROUTINE LSTIFF                                              LSTF   1
      COMMON /DAT/ GP(2),W(2),LIN(3)                                 LSTF   2
      COMMON GAMMAW,H(6),YM(6),POISS(6),PERM(6),DTIME(40)            LSTF   3
      COMMON SHFND(3),DSD(3),CARDSD(3),SHFNP(2),DSP(2),CARDSP(2)     LSTF   4
      COMMON B(3),E(2),DB(3),ES(5,5),FI(2,2),UAXS(2),ERHS(5)         LSTF   5
      COMMON ST(20,20),RHS(20),DISPA(20),VARINT(2,2,6),BC(4),XI,BTIME LSTF  6
      COMMON NCONN(3,6),NW(13),IBC(4),NINC,NDF,NEL,NG,INC,NE         LSTF   7
      COMMON L5,L6                                                   LSTF   8
      CALL ZEROR2(ES,5,5)                                            LSTF   9
      CALL ZEROR2(FI,2,2)                                            LSTF  10
      CALL ZEROR1(ERHS,5)                                            LSTF  11
      DO 40 IG=1,NG                                                  LSTF  12
      XI=GP(IG)                                                      LSTF  13
      WF=W(IG)*H(NE)/2.0                                             LSTF  14
      CALL SHAPE                                                     LSTF  15
      CALL FORMBE                                                    LSTF  16
      DO 18 J=1,3                                                    LSTF  17
   18 DB(J)=YM(NE)*(1.0-POISS(NE))/((1.0-2.0*POISS(NE))*(1.0+POISS(NE)))LSTF 18
     1  *B(J)                                                        LSTF  19
      DO 20 J=1,3                                                    LSTF  20
      DO 20 I=1,3                                                    LSTF  21
   20 ES(2*I-1,2*J-1)=ES(2*I-1,2*J-1)+WF*B(I)*DB(J)                  LSTF  22
      DO 22 I=1,3                                                    LSTF  23
      DO 22 J=1,2                                                    LSTF  24
      ES(2*I-1,2*J)=ES(2*I-1,2*J)+WF*B(I)*SHFNP(J)                   LSTF  25
   22 ES(2*J,2*I-1)=ES(2*I-1,2*J)                                    LSTF  26
      DO 24 I=1,2                                                    LSTF  27
      DO 24 J=1,2                                                    LSTF  28
   24 FI(I,J)=FI(I,J)+E(I)*E(J)*PERM(NE)*WF/GAMMAW                   LSTF  29
   40 CONTINUE                                                       LSTF  30
      DO 50 I=1,2                                                    LSTF  31
      DO 50 J=1,2                                                    LSTF  32
   50 ES(2*I,2*J)=-DTIME(INC)*FI(I,J)                                LSTF  33
      DO 54 I=1,2                                                    LSTF  34
      N=NCONN(I,NE)                                                  LSTF  35
      K=NW(N)+1                                                      LSTF  36
   54 UAXS(I)=DISPA(K)                                               LSTF  37
      DO 58 I=1,2                                                    LSTF  38
      DO 58 J=1,2                                                    LSTF  39
   58 ERHS(2*I)=ERHS(2*I)+FI(I,J)*UAXS(J)*DTIME(INC)                 LSTF  40
      IF(NE.NE.1) GOTO 70                                            LSTF  41
      IF(IBC(1).EQ.0) ERHS(1)=BC(1)*DTIME(INC)/BTIME                 LSTF  42
      IF(IBC(3).EQ.0) ERHS(2)=ERHS(2)-DTIME(INC)*BC(3)/GAMMAW        LSTF  43
   70 IF(NE.NE.NEL) RETURN                                           LSTF  44
      IF(IBC(2).EQ.0) ERHS(3)=-BC(2)*DTIME(INC)/BTIME                LSTF  45
      IF(IBC(4).EQ.0) ERHS(4)=ERHS(4)-DTIME(INC)*BC(4)/GAMMAW        LSTF  46
      RETURN                                                         LSTF  47
      END                                                            LSTF  48
```

LSTF 9–11 : initialise element stiffness matrix, flow matrix and element load array.
LSTF 12 : loop on all integration points.

LSTF 13 : local co-ordinate of integration point.
LSTF 14 : weighting factor X Jacobian.
LSTF 15 : calculate shape functions and derivatives for displacement and excess pore pressures.
LSTF 16 : form **B** and **E** matrices.
LSTF 17–19 : calculate **DB** matrix.
LSTF 20–22 : calculate displacement part of stiffness matrix, $\mathbf{B}^T\mathbf{DB}$.
LSTF 23–26 : calculate link matrix.
LSTF 27–29 : calculate flow matrix Φ.
LSTF 30 : end of integration point loop.
LSTF 31–33 : multiply Φ by time step.
LSTF 34–37 : current excess pore pressure (value at the end of previous increment).
LSTF 38–40 : calculate RHS pore pressure terms.
LSTF 41 : skip if not first element.†
LSTF 42 : add loads proportional to the time step for this increment.
LSTF 43 : add flow term to the RHS.
LSTF 44 : skip if not last element.†
LSTF 45 : add loads proportional to the time step for this increment.
LSTF 46 : add flow term to RHS.

Routine **SHAPE** calculates the shape functions and their derivatives (with respect to both local and 'Cartesian' axes) for displacements and excess pore pressures at the point with local co-ordinate ξ(XI) within an element.

Subroutine SHAPE

```
      SUBROUTINE SHAPE                                                  SHAP   1
      COMMON /DAT/ GP(2),W(2),LIN(3)                                    SHAP   2
      COMMON GAMMAW,H(6),YM(6),POISS(6),PERM(6),DTIME(40)               SHAP   3
      COMMON SHFND(3),DSD(3),CARDSD(3),SHFNP(2),DSP(2),CARDSP(2)        SHAP   4
      COMMON B(3),E(2),DB(3),ES(5,5),FI(2,2),UAXS(2),ERHS(5)            SHAP   5
      COMMON ST(20,20),RHS(20),DISPA(20),VARINT(2,2,6),BC(4),XI,BTIME   SHAP   6
      COMMON NCONN(3,6),NW(13),IBC(4),NINC,NDF,NEL,NG,INC,NE            SHAP   7
      COMMON L5,L6                                                      SHAP   8
      SHFND(1)=XI*(XI-1.0)/2.0                                          SHAP   9
      SHFND(2)=XI*(XI+1.0)/2.0                                          SHAP  10
      SHFND(3)=1.0-XI*XI                                                SHAP  11
      DSD(1)=XI-0.5                                                     SHAP  12
      DSD(2)=XI+0.5                                                     SHAP  13
      DSD(3)=-2.0*XI                                                    SHAP  14
      CARDSD(1)=DSD(1)*2.0/H(NE)                                        SHAP  15
      CARDSD(2)=DSD(2)*2.0/H(NE)                                        SHAP  16
      CARDSD(3)=DSD(3)*2.0/H(NE)                                        SHAP  17
      SHFNP(1)=(1.0-XI)/2.0                                             SHAP  18
      SHFNP(2)=(1.0+XI)/2.0                                             SHAP  19
      DSP(1)=-0.5                                                       SHAP  20
      DSP(2)=0.5                                                        SHAP  21
      CARDSP(1)=DSP(1)*2.0/H(NE)                                        SHAP  22
      CARDSP(2)=DSP(2)*2.0/H(NE)                                        SHAP  23
      RETURN                                                            SHAP  24
      END                                                               SHAP  25
```

† The boundary conditions are applied only to the first and last nodes.

SHAP 9–11 : calculate displacement shape functions.
SHAP 12–14 : calculate local derivatives of displacement shape functions.
SHAP 15–17 : calculate Cartesian derivatives of displacement shape functions.
SHAP 18–19 : calculate excess pore pressure shape functions.
SHAP 20–21 : calculate local derivatives of excess pore pressure shape functions.
SHAP 22–23 : calculate Cartesian derivatives of exess pore pressure shape functions.

Routine **FORMBE** computes values for the B and E matrices, using the values just calculated by SHAPE.

Subroutine FORMBE

```
      SUBROUTINE FORMBE                                                 FRMB   1
      COMMON /DAT/ GP(2),W(2),LIN(3)                                    FRMB   2
      COMMON GAMMAW,H(6),YM(6),POISS(6),PERM(6),DTIME(40)               FRMB   3
      COMMON SHFND(3),DSD(3),CARDSD(3),SHFNP(2),DSP(2),CARDSP(2)        FRMB   4
      COMMON B(3),E(2),DB(3),ES(5,5),FI(2,2),UAXS(2),ERHS(5)            FRMB   5
      COMMON ST(20,20),RHS(20),DISPA(20),VARINT(2,2,6),BC(4),XI,BTIME   FRMB   6
      COMMON NCONN(3,6),NW(13),IBC(4),NINC,NDF,NEL,NG,INC,NE            FRMB   7
      COMMON L5,L6                                                      FRMB   8
      B(1)=-CARDSD(1)                                                   FRMB   9
      B(2)=-CARDSD(2)                                                   FRMB  10
      B(3)=-CARDSD(3)                                                   FRMB  11
      E(1)=CARDSP(1)                                                    FRMB  12
      E(2)=CARDSP(2)                                                    FRMB  13
      RETURN                                                            FRMB  14
      END                                                               FRMB  15
```

FRMB 9–11 : calculate B matrix.
FRMB 12–13 : calculate E matrix.

Routine **FIXBC** 'fixes' the values of variables corresponding to the boundary conditions on the top and bottom of the layer.

Subroutine FIXBC

```
      SUBROUTINE FIXBC                                                  FXBC   1
      COMMON /DAT/ GP(2),W(2),LIN(3)                                    FXBC   2
      COMMON GAMMAW,H(6),YM(6),POISS(6),PERM(6),DTIME(40)               FXBC   3
      COMMON SHFND(3),DSD(3),CARDSD(3),SHFNP(2),DSP(2),CARDSP(2)        FXBC   4
      COMMON B(3),E(2),DB(3),ES(5,5),FI(2,2),UAXS(2),ERHS(5)            FXBC   5
      COMMON ST(20,20),RHS(20),DISPA(20),VARINT(2,2,6),BC(4),XI,BTIME   FXBC   6
      COMMON NCONN(3,6),NW(13),IBC(4),NINC,NDF,NEL,NG,INC,NE            FXBC   7
      COMMON L5,L6                                                      FXBC   8
      DO 20 I=1,4                                                       FXBC   9
      IF(IBC(I).EQ.0) GOTO 20                                           FXBC  10
      IF(I.EQ.1) N=1                                                    FXBC  11
      IF(I.EQ.2) N=NDF-1                                                FXBC  12
      IF(I.EQ.3) N=2                                                    FXBC  13
      IF(I.EQ.4) N=NDF                                                  FXBC  14
      IF(IBC(I).NE.1) GOTO 10                                           FXBC  15
      ST(N,N)=ST(N,N)+1.0E18                                            FXBC  16
      RHS(N)=RHS(N)+1.0E18*BC(I)*DTIME(INC)/BTIME                       FXBC  17
      GOTO 20                                                           FXBC  18
```

```
 10 IF(IBC(I).NE.2) GOTO 18                                              FXBC  19
    ST(N,N)=ST(N,N)+1.0E18                                               FXBC  20
    RHS(N)=RHS(N)+1.0E18*(BC(I)-DISPA(N))                                FXBC  21
    IBC(I)=1                                                             FXBC  22
    BC(I)=0.0                                                            FXBC  23
    GOTO 20                                                              FXBC  24
 18 WRITE(L6,200) I,IBC(I)                                               FXBC  25
200 FORMAT(32H ILLEGAL BOUNDARY CONDITION CODE,2I5)                      FXBC  26
    STOP                                                                 FXBC  27
 20 CONTINUE                                                             FXBC  28
    RETURN                                                               FXBC  29
    END                                                                  FXBC  30
```

FXBC 9 : loop on all variables with possible prescribed boundary conditions.
FXBC 10 : skip if variable is not prescribed (indicated by 0).
FXBC 11–14 : corresponding global variable number.
FXBC 15 : skip if the incremental value is not prescribed.
FXBC 16 : add large value to the diagonal term.
FXBC 17 : adjust RHS to yield prescribed value.
FXBC 19 : skip if the cumulative value is not prescribed (only applicable to excess pore pressure).
FXBC 20 : add large number to diagonal term (the pivot).
FXBC 21 : adjust RHS to yield prescribed value.
FXBC 25 : inadmissible boundary condition code.

Routine **SOLVE** solves the global matrix equations using Gaussian elimination.

Subroutine SOLVE

```
    SUBROUTINE SOLVE                                                     SOLV   1
    COMMON /DAT/ GP(2),W(2),LIN(3)                                       SOLV   2
    COMMON GAMMAW,H(6),YM(6),POISS(6),PERM(6),DTIME(40)                  SOLV   3
    COMMON SHFND(3),DSD(3),CARDSD(3),SHFNP(2),DSP(2),CARDSP(2)           SOLV   4
    COMMON B(3),E(2),DB(3),ES(5,5),FI(2,2),UAXS(2),ERHS(5)               SOLV   5
    COMMON ST(20,20),RHS(20),DISPA(20),VARINT(2,2,6),BC(4),XI,BTIME      SOLV   6
    COMMON NCONN(3,6),NW(13),IBC(4),NINC,NDF,NEL,NG,INC,NE               SOLV   7
    COMMON L5,L6                                                         SOLV   8
    NDF1=NDF-1                                                           SOLV   9
    DO 30 IQ=1,NDF1                                                      SOLV  10
    I1=IQ+1                                                              SOLV  11
    DO 26 I=I1,NDF                                                       SOLV  12
    DO 22 J=IQ,NDF                                                       SOLV  13
 22 ST(I,J)=ST(I,J)-ST(IQ,I)*ST(IQ,J)/ST(IQ,IQ)                          SOLV  14
 26 RHS(I)=RHS(I)-ST(IQ,I)*RHS(IQ)/ST(IQ,IQ)                            SOLV  15
 30 CONTINUE                                                             SOLV  16
    RHS(NDF)=RHS(NDF)/ST(NDF,NDF)                                        SOLV  17
    DO 60 II=1,NDF1                                                      SOLV  18
    IQ=NDF-II                                                            SOLV  19
    I1=IQ+1                                                              SOLV  20
    DO 58 I=I1,NDF                                                       SOLV  21
 58 RHS(IQ)=RHS(IQ)-ST(IQ,I)*RHS(I)                                      SOLV  22
 60 RHS(IQ)=RHS(IQ)/ST(IQ,IQ)                                            SOLV  23
    RETURN                                                               SOLV  24
    END                                                                  SOLV  25
```


SOLV 10–16 : Gaussian elimination to reduce global stiffness matrix to triangular form.
SOLV 18–23 : back-substitution to yield the unknown values. RHS contains the solved incremental values of displacement/excess pore pressure.

Routine **UPOUT** updates total displacements and excess pore pressures and prints out effective stresses and pore pressures at integration points.

Subroutine UPOUT

```
    SUBROUTINE UPOUT                                                     UOUT   1
    COMMON /DAT/ GP(2),W(2),LIN(3)                                       UOUT   2
    COMMON GAMMAW,H(6),YM(6),POISS(6),PERM(6),DTIME(40)                  UOUT   3
    COMMON SHFND(3),DSD(3),CARDSD(3),SHFNP(2),DSP(2),CARDSP(2)           UOUT   4
    COMMON B(3),E(2),DB(3),ES(5,5),FI(2,2),UAXS(2),ERHS(5)               UOUT   5
    COMMON ST(20,20),RHS(20),DISPA(20),VARINT(2,2,6),BC(4),XI,BTIME      UOUT   6
    COMMON NCONN(3,6),NW(13),IBC(4),NINC,NDF,NEL,NG,INC,NE               UOUT   7
    COMMON L5,L6                                                         UOUT   8
    DO 10 N=1,NDF                                                        UOUT   9
 10 DISPA(N)=DISPA(N)+RHS(N)                                             UOUT  10
    WRITE(L6,200)                                                        UOUT  11
200 FORMAT(40H DISPLACEMENTS AND EXCESS PORE PRESSURES/                  UOUT  12
   1  54H   NODE     INCREMENTAL VALUES         ABSOLUTE VALUES/         UOUT  13
   2  58H           ----------------------   ----------------------)     UOUT  14
    NN=1+2*NEL                                                           UOUT  15
    DO 20 N=1,NN                                                         UOUT  16
    K1=NW(N)                                                             UOUT  17
    IF(2*(N/2).NE.N) WRITE(L6,201) N,RHS(K1),RHS(K1+1),                  UOUT  18
   1  DISPA(K1),DISPA(K1+1)                                             UOUT  19
    IF(2*(N/2).EQ.N) WRITE(L6,202) N,RHS(K1),DISPA(K1)                   UOUT  20
201 FORMAT(1X,I5,4E13.3)                                                 UOUT  21
202 FORMAT(1X,I5,E13.3,13X,E13.3)                                        UOUT  22
 20 CONTINUE                                                             UOUT  23
    WRITE(L6,203)                                                        UOUT  24
203 FORMAT(38H EFFECTIVE STRESSES AND PORE PRESSURES/                    UOUT  25
   1  38H   ELEM  I.P.  EFF STRESS   PORE PRESS)                        UOUT  26
    DO 30 N=1,NEL                                                        UOUT  27
    NE=N                                                                 UOUT  28
    DO 30 IG=1,NG                                                        UOUT  29
    XI=GP(IG)                                                            UOUT  30
    CALL SHAPE                                                           UOUT  31
    CALL FORMBE                                                          UOUT  32
    N1=NCONN(1,NE)                                                       UOUT  33
    N2=NCONN(2,NE)                                                       UOUT  34
    N3=NCONN(3,NE)                                                       UOUT  35
    K1=NW(N1)                                                            UOUT  36
    K2=NW(N2)                                                            UOUT  37
    K3=NW(N3)                                                            UOUT  38
    VARINT(1,IG,NE)=VARINT(1,IG,NE)+YM(NE)*(1.0-POISS(NE))               UOUT  39
   1  /((1.0-2.0*POISS(NE))*(1.0+POISS(NE)))*(B(1)*RHS(K1)              UOUT  40
   2  +B(2)*RHS(K2)+B(3)*RHS(K3))                                       UOUT  41
    VARINT(2,IG,NE)=VARINT(2,IG,NE)+SHFNP(1)*RHS(K1+1)+SHFNP(2)          UOUT  42
   1  *RHS(K2+1)                                                        UOUT  43
    WRITE(L6,204) NE,IG,VARINT(1,IG,NE),VARINT(2,IG,NE)                  UOUT  44
204 FORMAT(1X,2I5,2E13.3)                                                UOUT  45
 30 CONTINUE                                                             UOUT  46
    RETURN                                                               UOUT  47
    END                                                                  UOUT  48
```

UOUT 9–10 : calculate cumulative values of displacement/excess pore pressure.
UOUT 16–23 : print out incremental and cumulative values of displacements/ excess pore pressures.
UOUT 27 : loop on all elements to print effective stress and pore pressures at integration points.
UOUT 28 : NE – element no.
UOUT 29 : loop on all integration points.
UOUT 30 : local co-ordinate of integration point.
UOUT 31 : calculate shape functions and derivatives.
UOUT 32 : calculate **B** and **E** matrices.
UOUT 33–35 : nodes of element.
UOUT 36–38 : g.v.n. of first variable of all nodes.
UOUT 39–41 : calculate incremental effective stress.
UOUT 42–43 : calculate incremental excess pore pressure.
UOUT 44 : print out stresses.
UOUT 46 : end of integration point and element loop.

These three subroutines zero real arrays with one, two and three subscripts respectively.

Subroutine ZERO

```
      SUBROUTINE ZEROR1(A,N)                                    ZERO   1
      DIMENSION A(N)                                            ZERO   2
      DO 10 I=1,N                                               ZERO   3
   10 A(I)=0.0                                                  ZERO   4
      RETURN                                                    ZERO   5
      END                                                       ZERO   6
      SUBROUTINE ZEROR2(A,M,N)                                  ZERO   7
      DIMENSION A(M,N)                                          ZERO   8
      DO 10 J=1,N                                               ZERO   9
      DO 10 I=1,M                                               ZERO  10
   10 A(I,J)=0.0                                                ZERO  11
      RETURN                                                    ZERO  12
      END                                                       ZERO  13
      SUBROUTINE ZEROR3(A,L,M,N)                                ZERO  14
      DIMENSION A(L,M,N)                                        ZERO  15
      DO 10 K=1,N                                               ZERO  16
      DO 10 J=1,M                                               ZERO  17
      DO 10 I=1,L                                               ZERO  18
   10 A(I,J,K)=0.0                                              ZERO  19
      RETURN                                                    ZERO  20
      END                                                       ZERO  21
```

ZERO 3–4 : zero a one-dimensional REAL array.
ZERO 9–11 : zero a two-dimensional REAL array.
ZERO 16–19 : zero a three-dimensional REAL array.

The **BLOCK DATA** subprogram initialises integration point co-ordinates and weights. It also initialises the element information vector LIN.


```
      BLOCK DATA                                                BDAT   1
      COMMON /DAT/ GP(2),W(2),LIN(3)                            BDAT   2
      DATA GP(1),GP(2),W(1),W(2)/-0.57735,.57735,1.0,1.0/       BDAT   3
      DATA LIN(1),LIN(2),LIN(3)/2,2,1/                          BDAT   4
      END                                                       BDAT   5
```

Arrays in common

GP	– Gauss point co-ordinates
W	– Weights
LIN	– Element type data
H†	– Height of elements
YM†	– Young's modulus
POISS†	– Poisson's ratio
PERM†	– Permeability
SHFND	– Displacement shape functions
DSD	– Derivatives of shape functions w.r.t. local co-ordinate
CARDSD	– Cartesian derivatives of shape functions
SHFNP	– Pore pressure shape functions
DSP	– Local derivatives of excess pore pressures
CARDSP	– Cartesian derivatives of excess pore pressure shape functions
B	– Strain–displacement matrix
E	– **E** matrix
DB	– **D** × **B**
ES	– Element stiffness matrix
FI	– Flow matrix – Φ
UAXS	– Excess pore pressures
ERHS	– Element Right-Hand-Side terms
ST†	– Global stiffness matrix
RHS†	– Global RHS
DISPA†	– Global displacement/pore pressure array
VARINT†	– Stresses at integration points
BC	– Boundary conditions
NCONN†	– Element–nodal connectivity
NW†	– Global variable number of first d.o.f. of each node
IBC	– Code for boundary conditions

Variables in common

NINC	– Number of increments
NDF	– Total number of d.o.f. (variables)
NEL	– Number of elements
NG	– Number of integration points
INC	– Current increment
NE	– Current element

† These arrays have been set up for a maximum of 6 elements.

GAMMAW – Unit weight of water
XI – Local co-ordinate
BTIME – Total time step for increment block

3.6.3 Input specification for TINY

Data record	Contents	No. of records
A	TITLE	1
B	GAMMAW	1
C	NEL	1
D	H YM POISS PERM	NEL
E	NOINCB	1
F	NINC	NOINCB
G	DTIME(1) . . . DTIME (NINC)	NOINCB
H	IBC(1) . . . IBC(4) BC(1) . . . BC(4)	NOINCB

where

TITLE – Title for analysis
GAMMAW – Unit weight of water
NEL – Number of elements
H – Height of element
YM – Young's modulus
POISS – Poisson's ratio
PERM – Permeability
NOINCB – Number of increment blocks
NINC – Number of increments in increment block
DTIME(I) – Time step for Ith increment in block
IBC(1) – 0 – Displacement d.o.f. at node 1 has applied stress boundary condition = BC(1) (compression +ve)
1 – Displacement d.o.f. at node 1 is prescribed with incremental value equal to BC(1) (applied at constant rate over time of increment block)
2 – Displacement d.o.f. at node 1 is prescribed to have an absolute value of BC(1) during the first increment of block and then kept steady at this value
IBC(2) – Boundary condition for displacement d.o.f. at last node (same conventions as above)
IBC(3) – 0 – Excess pore pressure d.o.f. at node 1 has prescribed artificial seepage velocity of BC(3) (flow in +ve)
1 – Excess pore pressure d.o.f. at node 1 is prescribed with incremental value equal to BC(3) (applied at constant rate over time of increment block)

2 – Excess pore pressure d.o.f. at node 1 is prescribed to have an absolute value of BC(3) during the first increment of block and is then kept steady at this value
IBC(4) – Boundary condition for excess pore pressure d.o.f. at last node (same conventions as above)

In the examples that follow, node 1 is considered to be at the top of the layer, and the last node at the bottom (however, the program is oblivious to this difference, and would produce identical results if the opposite convention were used).

3.6.4 Consolidation analyses

This section illustrates the use of the TINY program in section 3.6.2, and explains why the choice of time steps for analyses can require some care. The program is used in analysing the following two problems:

1. One-dimensional Terzaghi consolidation.
2. Under-drainage.

The first problem is a layer of thickness 20 m subjected to a vertical pressure. This generates a uniform excess pore pressure throughout the layer. Then drainage is allowed from both the top and the bottom surfaces. Because of symmetry, only the upper half is considered in the analysis (see Fig. 3.19). The mesh is modelled by six elements, with thinner elements adjacent to the top drainage surface. This is because of the rapid change in pore pressures near this boundary. The following material properties are assumed for the layer, which is isotropic and homogeneous.

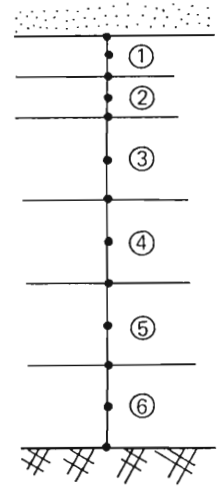


Fig. 3.19 – Finite elements to model Terzaghi one-dimensional consolidation

$$E = 1000 \text{ kPa} \quad \nu = 0.25 \quad k = 10^{-9} \text{ m/sec.}$$

The applied pressure is 10 kPa. The base of the layer is restrained and impermeable (corresponding to a pore pressure boundary condition of zero flow). The first increment block consists of a single increment in which the vertical pressure of 10 kPa is applied and the base is restrained. This causes a uniform excess pore pressure of 10 kPa to develop in the layer.

At this stage, two points need clarification: the pore pressure boundary condition and the selection of time steps in the subsequent increment. For integration in time, $\theta = 1$. Hence the solution is unconditionally stable for any size of time steps (Booker and Small, 1975). However, this does not necessarily imply that any size of time step is permissible. For the above example, taking the unit weight of water is 10 kN/m^3, $c_v = 1.2 \times 10^{-7}$ m^2/sec. It is possible to solve the one-dimensional consolidation problem approximately using parabolic isochrones (Schofield and Wroth, 1968).

Fig. 3.20 illustrates the isochrone moving in from the boundary up to the point denoted by A. Points below A have not yet experienced any change in pore pressure due to the draining boundary. It can be shown that the time taken, t, for this is given by

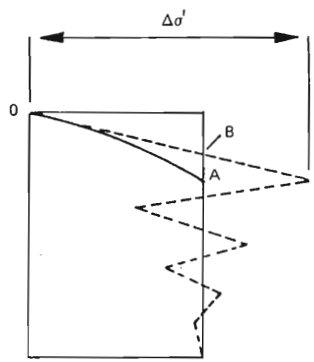


Fig. 3.20 – Pore pressure distribution after first time step of analysis with short time step

$$l = \sqrt{(12 c_v t)}, \tag{3.44}$$

where l is the distance to point A from the boundary. If l is the normal distance of the first pore pressure node from the boundary then t specifies the minimum time step that can be specified. This can be explained in a simple manner. The element chosen allows for a linear variation of pore pressure. If a time step $t_1 < t$ is chosen then the drainage would have taken place up to a point (say) B. An attempt by the analysis to model this situation closely would generate a pore pressure at A equal to $\Delta\sigma'$ which is greater than the applied vertical pressure. In order to compensate for this error, a smaller pore pressure is generated in the next node. This results in the zigzag distribution, shown in Fig. 3.20. A similar

limit on the minimum time step has been arrived at by others (Pyrah, 1980; Vermeer and Verrujit, 1981). Substituting the values for the chosen mesh, $t = 6.9 \times 10^5$. Based on this, a time step of 10^6 is chosen for the first increment. It is quite common to use a log scale for time in the plot of settlement (or degree of consolidation) against time. As time passes, dissipation takes place at a reduced rate. It is logical to use progressively larger time steps in the finite element analysis. The usual practice is to select a fixed number of time steps (say 4 or 5) within a log cycle. The following are examples of such a scheme:

```
0.1   |   1 2 2 5   | 10 10 20 50   | 100 100 200 500 |
total time      10              100                 1000

  1   |   1  2  6 | 10  20   60 | 100  200   600 |
total time      10            100            1000
```

For this problem it is also possible to make a simple estimate of the total time required for the dissipation of the pore pressures. Using the relationship between degree of consolidation and time factor, the time for 90% consolidation is calculated to be 0.7×10^9. Using the above data on the smallest possible time step and the total time, the following time steps were chosen for the analysis:

```
1.  | 1.E6  1.E6  2.E6  6.E6 | 1.E7  2.E7  6.E7 | 1.E8  2.E8  6.E8 |

    | 1.E9
```

Now we come to the question of pore pressure boundary conditions. We have found that the best technique is not to apply both loads and pore pressure boundary conditions in the same increment. The load was applied in the first increment. The appropriate pore pressure boundary condition (drainage from top surface) is then applied in the second increment. In order to fix the absolute excess pore pressure, a fixity code of 2 is used. There is more discussion of the use of fixity codes 1 and 2 for excess pore pressures in section 9.2. The input data for the analysis are as follows:

```
Record
  A  | ***EXAMPLE 1 *** TERZAGHI 1-D CONSOLIDATION ***
  B  | 10.
  C  | 6
  D  | 1.  1.E3  0.25  1.E-9
  D  | 1.  1.E3  0.25  1.E-9
  D  | 2.  1.E3  0.25  1.E-9
  D  | 2.  1.E3  0.25  1.E-9
  D  | 2.  1.E3  0.25  1.E-9
  D  | 2.  1.E3  0.25  1.E-9
  E  | 2
  F  | 1
  G  | 1.
```

```
H | 0 1 0 0   10. 0. 0. 0.
F | 11
G | 1.E6 1.E6 2.E6 6.E6 1.E7 2.E7 6.E7
G | 1.E8 2.E8 6.E8 1.E9
H | 0 1 2 0  0. 0. 0. 0.
```

Fig. 3.21 shows the computed isochrones compared with the theoretical solution (based on Fourier series). The comparison is good, bearing in mind the number of time steps and elements used in the analysis. Fig. 3.22 shows the degree of consolidation plotted against $\sqrt{T_v}$. Again the comparison is reasonably good.

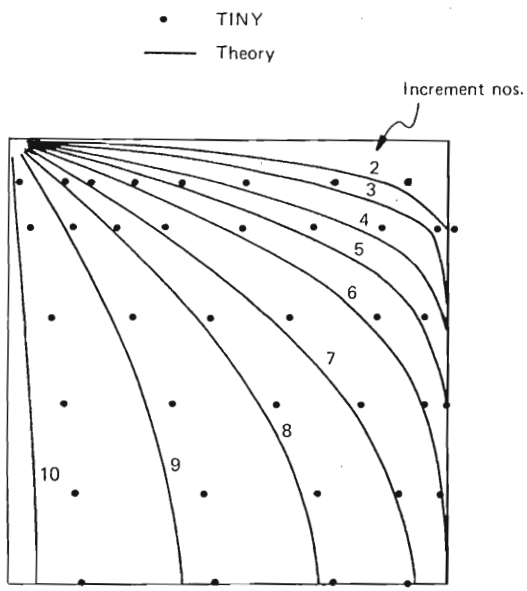


Fig. 3.21 – Excess pore pressure isochrones for Terzaghi one-dimensional consolidation

In order to demonstrate the discussion on the time step, a separate analysis with a lower time step of 10^3 was carried out, and Fig. 3.23 shows the pore pressure distribution at the end of that increment.

The input data for the next example are as follows:

Record

```
A | ***EXAMPLE 2 *** UNDER DRAINAGE ***
B | 10.
C | 6
D | 2.  1.E3 0.25 1.E-9
D | 2.  1.E3 0.25 1.E-9
D | 2.  1.E3 0.25 1.E-9
```

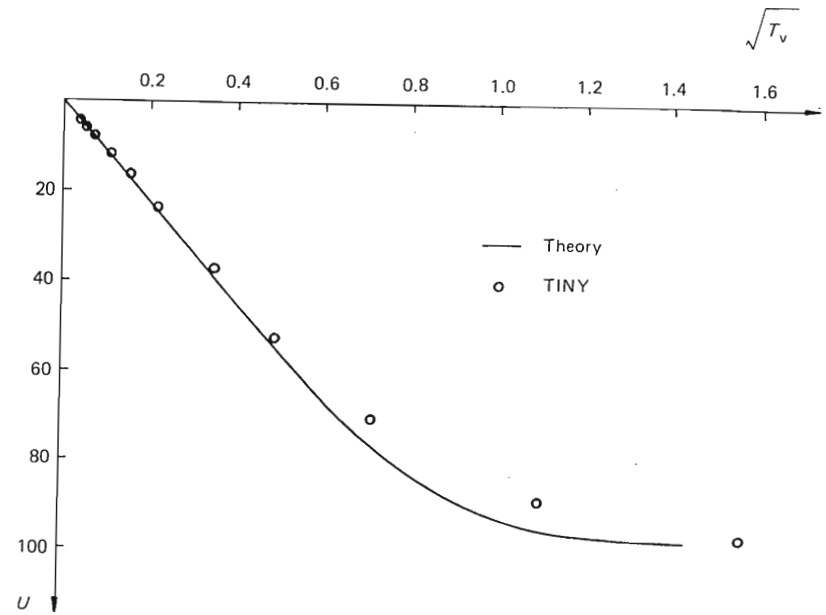



Fig. 3.22 – Plot of degree of consolidation against $\sqrt{T_v}$ for Terzaghi one-dimensional consolidation

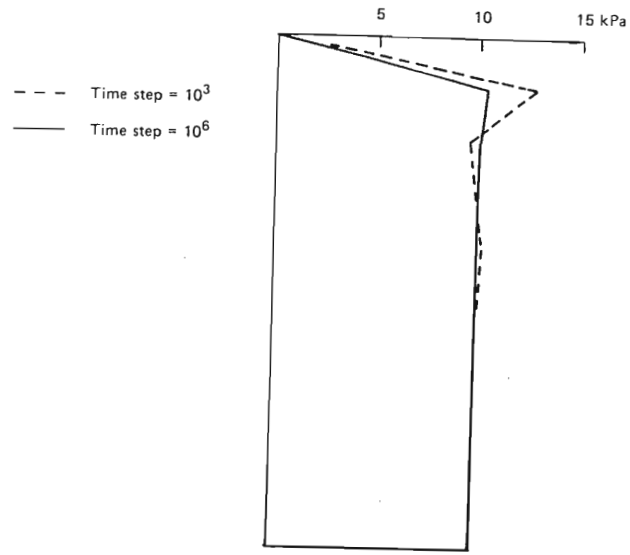


Fig. 3.23 – Comparison of pore pressure distributions for different time steps

```
D |  2.  1.E3 0.25 1.E-9
D |  1.  1.E3 0.25 1.E-9
D |  1.  1.E3 0.25 1.E-9
E |  1
F |  11
G |  1.E6  1.E6  2.E6  6.E6  1.E7  2.E7  6.E7
G |  1.E8  2.E8  6.E8  1.E9
H |  0  1  1  2  0.  0.  0.  −10.
```

In this example, the drainage boundary (with rapidly changing pore pressures) is at the bottom and elements are thinner towards it. However, both top and bottom are drainage boundaries. The base of the layer is restrained and is maintained at an excess pore pressure of −10 kPa. The top surface is maintained at 0. The resultant isochrones are plotted and compared against the theoretical solution in Fig. 3.24. Again the comparison is good. Fig. 3.25 shows the plot of degree of consolidation against the $\sqrt{T_v}$. It is worth mentioning that the plot of degree of consolidation against $\sqrt{T_v}$ is the same for the dissipation of both rectangular and triangular pore pressure distributions. In fact it is the same (Taylor, 1948) for any linear distribution of pore pressure.

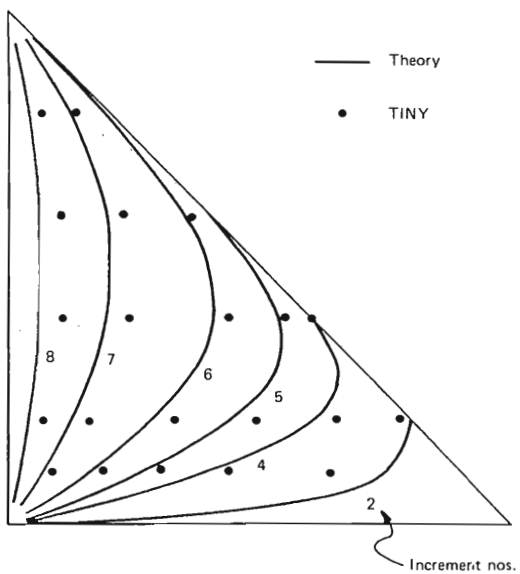


Fig. 3.24 – Excess pore pressure isochrones for under-drainage

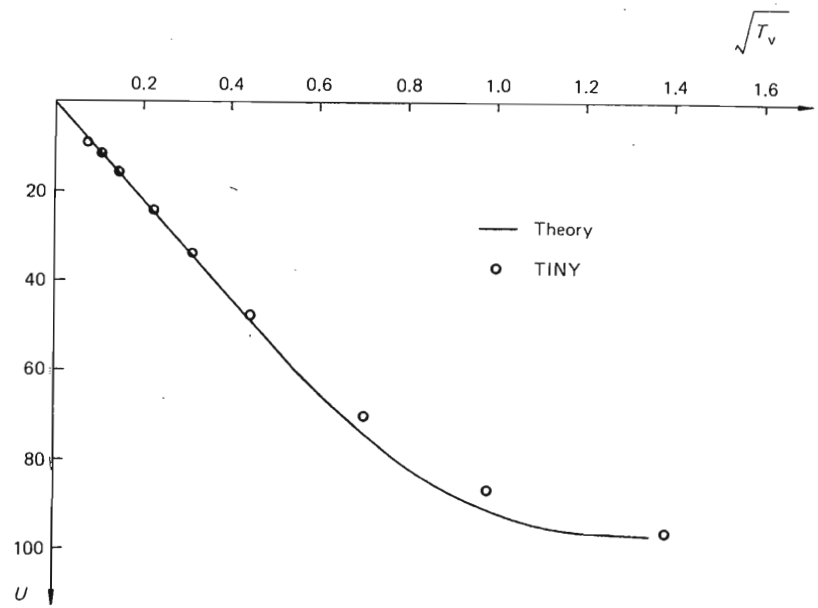


Fig. 3.25 – Plot of degree of consolidation against $\sqrt{T_v}$ for under-drainage problem

4

Introduction to CRISP

4.1 INTRODUCTION

This chapter introduces CRISP (CRItical State Program). The size of problem which CRISP can tackle is limited only by the amount of memory and processing power of the computer concerned. CRISP has been mounted on many different makes of computer, with only minor modifications. We explain the programming strategy which has made this possible. Finally we explain the basic structure of the program and document the main controlling routines.

4.1.1 Summary of facilities

(a) Types of analysis:
Undrained, drained or fully-coupled (Biot) consolidation analysis of two-dimensional plane strain or axisymmetric (with axisymmetric loading) solid bodies.

(b) Soil models:
Anisotropic elasticity, inhomogeneous elasticity (properties varying with depth), critical state soil models (Cam-clay, modified Cam-clay).

(c) Element types:
Linear strain triangle and cubic strain triangle (with extra pore pressure degrees of freedom for a consolidation analysis).

(d) Non-linear techniques:
Incremental (tangent stiffness) approach. Options for updating nodal co-ordinates with progress of analysis. $\theta = 1$ for integration in time.

(e) Boundary conditions:
Element sides can be given prescribed incremental values of displacements or excess pore pressures. Loading applied as nodal loads or pressure loading on element sides. Automatic calculation of loads simulating excavation or construction when elements are removed or added.

(f) Miscellaneous:
Stop–restart facility allows analysis to be continued from a previous run.

4.2 CRISP: HOW IT'S DONE (AND WHY)

4.2.1 Element types

The library of elements consists of the triangular elements shown in Fig. 4.1. The basic element is the six-noded linear strain triangle (LST – element type 2). This element and the higher-order Cubic Strain Triangle (CuST – element type 6) can be used for drained or undrained analysis. The corresponding elements for consolidation analyses are element types 3 and 7 respectively. These element types have additional degrees of freedom (d.o.f.), namely excess pore pressures. The pore pressure nodes are deployed such that the strains and pore pressures have the same order of variation across an element.

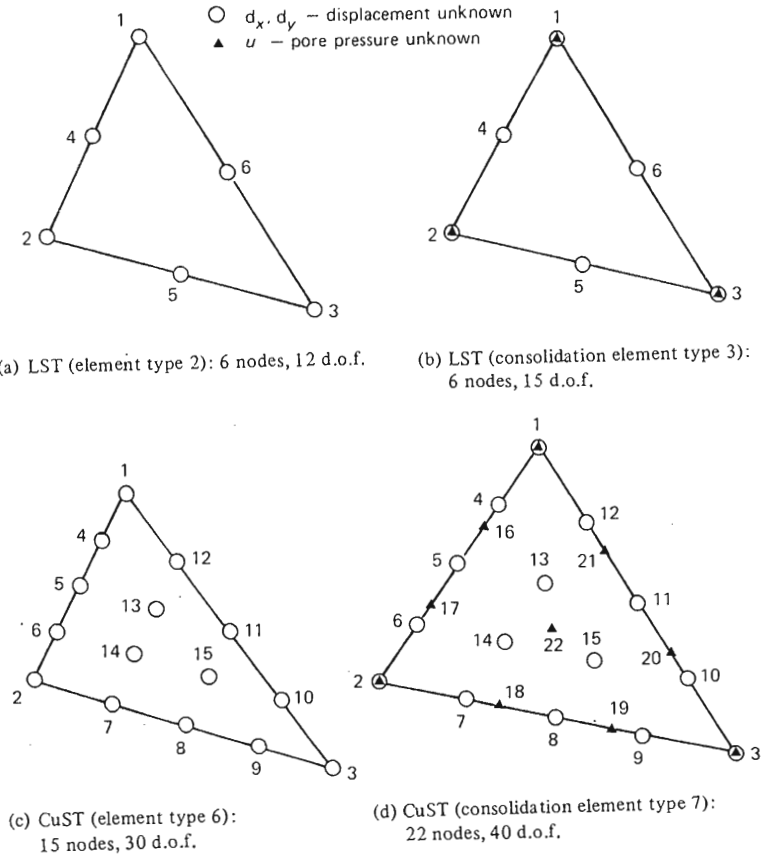
The higher-order triangular elements have two attractions.

1. Fewer elements are needed for the analysis of most problems, making the data preparation less arduous.
2. Under undrained conditions the constraint of no volume change leads to 'locking' of finite element meshes when low-order elements are used. Recent research (Sloan and Randolph, 1982) has shown that these problems can be avoided by using higher-order elements (at least LST for plane strain and CuST for axisymmetric plane strain).

On the other hand, there are occasions where the use of a lower-order element (i.e. LST rather than CuST) can be advantageous: for example, situations where the mesh has irregular boundaries or contains several zones of soil with different properties. Indiscriminate use of higher-order elements in these circumstances can lead to unnecessarily expensive analyses.

Elements of type 2 can be mixed with elements of type 3, and so can type 6 with type 7. This may be useful in carrying out a consolidation analysis where part of the mesh behaves in a drained manner.

Using the higher-order elements is just as straightforward as the lower-order ones because the program user only has to specify the co-ordinates of the nodes at the vertices of triangular elements. Edge and interior nodes are then calculated



(a) LST (element type 2): 6 nodes, 12 d.o.f.

(b) LST (consolidation element type 3): 6 nodes, 15 d.o.f.

(c) CuST (element type 6): 15 nodes, 30 d.o.f.

(d) CuST (consolidation element type 7): 22 nodes, 40 d.o.f.

Fig. 4.1 – Different types of element

by interpolation, assuming the element has straight sides. However, elements with curved boundaries can be used if the appropriate side node coordinates are specified.

The program has been designed so that new element types can be added with relatively little effort. In particular the incorporation of elements like the three-noded bar element or the eight-noded quadrilateral element is not difficult. These two element types could be mixed with LST elements in a mesh. The only restriction on different element types being mixed together is that they should have the same number of nodes along the sides (edges).

In numbering the vertex nodes and the elements in the mesh, gaps in the numbering are allowed for; this permits the user to alter some part of the mesh without having to re-number the mesh completely. The additional nodes along element sides and any inner nodes are assigned numbers by the program.

4.2.2 Solution techniques

When describing finite element techniques in Chapter 3, it was assumed that soil response is linear and elastic. The causes of non-linear response can be identified as being either geometric non-linearity or material non-linearity. Geometric non-linearity arises when large deformations of the structure mean that the equilibrium equations (based on the undeformed geometry) are no longer sufficiently accurate. Material non-linearity arises when the stress–strain relation for the material is non-linear (e.g. the Cam-clay relations described in Chapter 2). In general, non-linearity of a system may be due to geometric non-linearity, material non-linearity, or both together. Carter (1977; Carter *et al.*, 1977) examined the importance of non-linear geometric effects in geotechnical analysis. His general conclusion was that the 'linear' assumption of small strains and small displacements is usually satisfactory in the solution of geotechnical problems. In the majority of cases the normal infinitesimal strain assumption leads to an overestimation of deformations compared to the use of finite deformation theory (and hence is pessimistic). Thus in most geotechnical analyses, non-linearity arising from material behaviour is of more importance than non-linearity from geometrical effects.

The small-displacement, small-strain approach is used throughout in this book (and in CRISP). Hence we are able to avoid the extra complexity of using the strain and stress tensors appropriate to large deformations and strains. The program does, however, contain the option of updating the co-ordinates of nodal points as the analysis proceeds. In fact this is equivalent to a first approximation to an updated Lagrangian formulation (see, for example, Cook, 1981, Chapter 13).

There are a number of techniques for analysing non-linear problems using finite elements. CRISP uses the incremental or tangent stiffness approach: the user divides the total load acting into a number of small increments (say 50 or 100 in a typical analysis) and the program applies each of these incremental loads in turn. During each increment the stiffness properties appropriate for the current stress levels are used in the calculations. If only a few increments are used, this method produces a solution which tends to drift away from the true or exact solution. This means a stiffer response results for a strain-hardening model and the displacements are always under-predicted. In mathematical terms we are integrating a differential equation using Euler's method.

This approach is in contrast to that adopted in the elasto-plastic programs used in the analysis of mechanical engineering components or steel structures (see, for example, Owen and Hinton, 1980). In these applications it is usual to use a larger size of increments (say 10 in a complete analysis) and to correct for the error described above by performing iterations within each increment until convergence to the non-linear load–displacement curve is obtained. Experience with this technique with critical state models has been rather mixed. Some claim to have applied the technique with no particular difficulty (e.g. Zienkiewicz *et al.*, 1975; Potts, 1981), but our experience, in common with that

of Naylor (1975), is that sometimes there can be problems with convergence, and that sometimes the known (analytical) solution cannot be recovered from the numerical procedure. Perhaps this is not surprising: in structural mechanics problems the zone of plastic behaviour is often restricted to a small part of the structure, whereas in geotechnical problems the zone of plastic deformation frequently occupies the majority or even the whole mesh.

Clearly there must be some limitation on the maximum increment size when using an incremental scheme. Some advice on this is included in Chapter 9. The use of an incremental scheme fits in quite well with the scheme for consolidation analysis that we have adopted, an incremental time-marching technique with $\theta = 1$, as described in Chapter 3.

4.2.3 Excavation, construction and increment blocks

A finite element program intended for geotechnical analysis should be capable of analysing problems where soil is excavated or soil structures (e.g. embankments) are constructed. This is not a standard feature found in finite element programs in other branches of engineering. CRISP allows elements to be removed to simulate excavation and elements to be added to simulate construction. The implied loadings for both these cases are automatically calculated by the program.

When performing a non-linear analysis involving excavation or construction, the requirement for relatively small applied loads in each increment still applies. The obvious way of achieving this is the removal or addition of a large number of layers of 'thin' elements. Unfortunately the result is an unacceptable rise in the solution cost (due to the large number of elements), and possible numerical conditioning problems associated with elements that have large aspect ratios. CRISP circumvents this problem by allowing the effect of element removal or addition to be spread over several increments in an 'increment block'. An increment block is just a series of ordinary increments grouped together in the input data for the program. Element stiffnesses are always added or removed in the first increment of a block, but the associated loads are distributed over all the increments in the block. Clearly this procedure introduces an extra degree of approximation in modelling, but it has been found to be satisfactory in practice. Increment blocks can also be used for the purpose of distributing applied boundary loads or prescribed displacements over several increments, achieving a certain economy in data preparation.

4.2.4 Equilibrium check

The program incorporates an equilibrium check to ensure that equilibrium is satisfied at the end of each increment. In this equilibrium check the stresses in the elements currently in the mesh are integrated over the volume to calculate the equivalent nodal loads and these are then compared with the external loadings. The difference is then expressed as a percentage of the applied loading, and is called the error in equilibrium or the out-of-balance load. This form of equilibrium check is essential in any analysis using iterative methods or the

load increments are sufficiently small, there is no stress correction at the end of each increment. This means that the stresses calculated at the end of each increment should be consistent with the applied loading. Hence, in theory, an initial stress approach. In CRISP, because of the implicit assumption that the equilibrium check is not necessary, but in fact it is useful in giving an indication of any numerical problems that may arise during the course of an analysis.

4.2.5 Stop–restart facility

Non-linear finite element analysis tends to be a time-consuming business (for both the computer and the program user). Getting the size of the load increments right usually involves re-running the program several times and examining the computer output. So that the user does not have to continually rerun the analysis from the start each time, a stop–restart facility is provided. The program can be requested to store analysis results on a permanent magnetic storage medium (i.e. magnetic disk or magnetic tape) and the computer run can be restarted.

Two versions of the stop–restart facility are available. In the first, the results of every increment are saved; in the second, results from the last increment only are stored. To use the first version one must be able to run a job with two magnetic tapes or have access to large amounts of disk space (probably more than 10 megabytes). For the second a more modest amount of disk space will suffice (say 100 kilobytes).

The stop–restart facility also makes possible the production of graphical displays of the results. A 'post-processing' program is used to read information from the stop–restart file. Usually this program will use calls to a local graphics library to produce plots and graphical displays on the devices that the user has access to. Now that CRISP is being mounted on many different computers, there is a tendency to write programs using graphics libraries which are more generally available, e.g. GINO from the CAD (Computer Aided Design) centre, Madingley Road, Cambridge.

4.2.6 Frontal solver

CRISP solves the linear simultaneous stiffness equations using the frontal solution method. In essence this is just Gaussian elimination as encountered in Chapter 3, but programmed in such a way as to minimise operations on zero terms and to use minimum computer memory for the stiffness matrix. Our version is based on the model program by Irons (1970), modified for variable numbers of degrees of freedom at nodal points. The frontal technique starts from the observation that in Gaussian elimination one can start eliminating variables before the global matrix is fully assembled.

We introduced the frontal method into our program because it was the only way of running reasonably-sized meshes for consolidation analysis on a machine with a fixed core store limit. Now that virtual store operating systems are widespread, there is an argument that all this complicated programming is not really

necessary. Perhaps it is not, but you have to be prepared to wait longer for your results.

4.3 CRISP PORTABILITY AND PROGRAMMING TECHNIQUES

The following two sections explain why CRISP has proved to be such a portable program, and the technique which allows it to handle problems of an arbitrary size is described.

4.3.1 Portability

CRISP is written in ANSI (American National Standards Institute) standard FORTRAN. Because of this the program has been mounted on many different manufacturers' computers with relatively little effort. We suspect that most engineers writing FORTRAN programs have not heard of the standard, and we therefore set out why it is important.

The FORTRAN programming language was originally developed to run on the IBM704 computer in the mid-fifties. Its success led to its adoption by other computer manufacturers, who wrote compilers to translate FORTRAN statements into the machine language of their own computers. Since the original FORTRAN language contained restrictions owing to the hardware limitations of the IBM704, there was a natural move to extend the language on the other computers, thereby offering a more powerful programming language (and a more saleable product). Unfortunately, the consequence of this was that a FORTRAN program written for one computer would be unlikely to run on another computer without some modification.

To overcome these problems, ANSI produced a standard definition of FORTRAN in 1966. This language is sometimes called FORTRAN IV, but should more properly be called ANSI (1966) Standard FORTRAN. (FORTRAN IV is the name of the IBM implementation.) Although computer manufacturers made sure that their compilers accepted the standardised language, they did not remove the various extensions. Most engineers engaged in programming would make use of the manufacturer's FORTRAN reference manual, and so non-portable programming practices persisted. This is perhaps understandable, because FORTRAN 66 still lacked some facilities which make programming (and using programs) easier.

In 1978, ANSI produced a new standard, FORTRAN 77, and at the time of writing another new standard, FORTRAN 8X, is under discussion. CRISP conforms to the 77 standard, and, apart from a couple of exceptions mentioned below, to the 66 standard too. Indeed, the majority of CRISP even avoids some FORTRAN 66 constructs which have been known to cause problems on some computers. In doing this we have followed the advice of Larmouth (1973a, 1973b) and Day (1978). The program has also been passed through the PFORT verifier (Ryder, 1974). Only those readers who have not had to convert FORTRAN programs from running on one machine to running on another will find all our precautions pedantic.

We take advantage of only two features of FORTRAN 77 not present in the 1966 standard. The first is the list-directed READ statement (often referred to as the free-format READ). The second is the use of CHARACTER variables to store textual information.

If FORTRAN 77 had been fully supported on the Cambridge University Computing Service IBM installation before 1984, the reader would probably see other FORTRAN 77 statements, such as the block IF construction, in our program. (This certainly makes programs more readable and is a definite advance on FORTRAN 66.) Readers who intend to modify CRISP for their own purposes, or who are going to write their own programs, are advised to use a textbook as their main reference, rather than the manufacturer's manual. Katzan (1978) completely covers the 77 standard, including the syntax diagrams from the standard. However, its succinct style makes it suitable for experienced FORTRAN programmers. A text that is more suitable for relatively inexperienced programmers is Monro (1982). Of course, the really dedicated will read the standard from ANSI (1978).

4.3.2 Pseudo-dynamic dimensioning

Finite element programs written in FORTRAN make use of REAL and INTEGER arrays to store the data which they manipulate. Some of these arrays will always be the same size each time the program is run (for example an array storing an element stiffness matrix). The size of other arrays (for example the global stiffness matrix) will depend on the data for the current problem.

The simplest approach is to dimension these 'variable length' arrays to a size which appears reasonable. In fact this was done in the TINY program in Chapter 3, where the arrays were set up to solve a problem with a maximum of six elements. However, this approach has two drawbacks. Firstly, a user of the program will inevitably want to run a problem which requires larger arrays, resulting in a lot of program changes. Secondly, for much of the time a lot of space in the arrays will be unused.

CRISP uses a technique known as 'pseudo-dynamic dimensioning' to avoid these pitfalls. To understand this technique, a brief account of how FORTRAN implementations allocate storage for arrays will be useful.

If an array is declared in a subroutine by a statement such as

```
DIMENSION XYZ (2, 50)
```

(and the array is not a dummy argument of the subroutine), then 100 contiguous storage locations (associated with the subroutine) are reserved. Alternatively, an array may be declared as being in a common area of storage (using the COMMON statement), which is not associated with any particular subroutine. The TINY program in Chapter 3 used this technique to allow its various subroutines to access the same arrays. When an array is passed as an actual argument to another subroutine, it is the address of the first memory

location that is transferred. An array in a subroutine may be given variable dimensions, e.g.

```
DIMENSION XYZ (NDIM, NN)
```

provided that the array and its dimensions are dummy arguments of the subroutine. Thus one improvement over using fixed dimensions in each subroutine is to have fixed dimensions in the main program, and to pass the arrays to subroutines as variably dimensioned arrays. Changes to the program now require amending the main program only. However, the basic disadvantage of having to continually edit the program and of wasted space still remain.

CRISP overcomes these remaining disadvantages by arranging that all the variably dimensioned arrays are allocated as part of one long array.

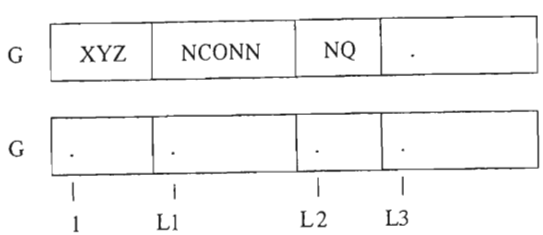
Suppose that the following arrays have to be allocated store:

XYZ (NDIM, NN), NCONN(NTPE, NEL), NQ(NN)

where

XYZ	– co-ordinates of nodes
NCONN	– list of nodes associated with each element
NQ	– no. of d.o.f. of each node
NDIM	– no. of spatial dimensions for analysis (2 for 2-D)
NTPE	– no. of nodes associated with each element
NN	– total no. of nodes in mesh
NEL	– total no. of elements in mesh.

The arrays are allocated in the same order as above to a single array G:



where

```
L1 =  1 + NDIM * NN
L2 = L1 + NTPE * NEL
L3 = L2 + NN
```

and

G(1) is the first storage location of array XYZ
G(L1) is the first storage location of array NCONN
G(L2) is the first storage location of array NQ
G(L3) is the first storage location of next array
(if no futher arrays are present then L3 − 1 serves an index to the amount of array G which has been used).

The arrays are passed to a subroutine SUB1 as follows:

```
CALL SUB1 (G(1), G(L1), G(L2), . . . . . NDIM,NN,NTPE, NEL)
```

In the subroutine the arrays appear as dummy arguments and are dimensioned:

```
SUBROUTINE SUB1 (XYZ,NCONN,NQ, . . . . . NDIM,NN,NTPE,NEL)
DIMENSION XYZ (NDIM,NN), NCONN(NTPE,NEL),NQ(NN)
```

A disadvantage of this technique is the long argument lists that result, as the indexes for the numerous arrays have to be passed from the main routine to other routines. Instead of declaring some arrays in the few routines that use them, they have to be passed through the intermediate routines which do not require them. This to a certain extent gives a complex look to the program. However, the benefits more than offset this minor irritation.

CRISP extends this technique to arrays which would appear to have fixed dimensions (e.g. NDIM in the above example). The aim is to make future program modifications relatively straightforward.

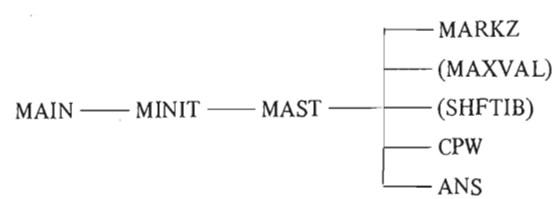
Some arrays of fixed size are used in the program and usually reside in named COMMON blocks. Arrays which provide the indexes and numerical integration data are initialised in a block data routine. Therefore these arrays cannot be allocated store pseudo-dynamically. This would mean that if new element types are introduced, the sizes of these arrays have to be altered in all routines which reference these arrays. The way round this is to allocate sizes which include some spare space. This means additional element types can be included without having to change the sizes of these arrays every time.

Other fixed-length arrays are mainly linked with the number of nodes with fixities and the number of nodes with externally applied loads (the loads in fact are stored in terms of pressure loads (both normal and shear components) applied at nodes along element sides). The required space is dependent on the number of nodes (and element sides) which lie along the mesh boundary. The sizes of these arrays have been arbitrarily fixed; however, a count is kept of the number of entries made, and error/warning messages are printed when array sizes are exceeded and clear messages of what has to be done to remedy the situation are printed.

4.4. CRISP

4.4.1 CRISP organisation

The relationship between the main controlling routines of CRISP is shown below:



The **MAIN** program is the only routine that needs to be changed if a larger version of CRISP is required. For this reason it is kept as short as possible. Many users will keep several versions of this routine (e.g. small, medium and large), and will link in whichever is appropriate for the problem at hand. Routine **MINIT** contains machine independent initialisations, e.g. unit numbers for files. **MAST** is the main controlling routine, and its main business is pseudo-dynamic dimensioning.

The rest of the program logically falls into three parts, identified by routines MARKZ, CPW and ANS, each being called by MAST in turn. (MAXVAL and SHFTIB shown above for completeness just carry out housekeeping associated with the dynamic arrays.)

MARKZ is the part of the program that deals with the geometry of the user's mesh. MARKZ tries to make the time-consuming business of drawing up (or modifying) a mesh easier, by allowing gaps in the numbering systems for elements and nodes, automatically generating midside (and where appropriate internal) node numbers and co-ordinates. Basically, it is all housekeeping.

CPW is the part of the program that deals with *in situ* stresses and material parameters. This is an important part of the problem definition which the user must attempt to get right, and try to understand the consequences (not everything is independent).

ANS is the part of the program that performs the analysis. ANS reads the loads and other boundary conditions and applies the principles of mechanics. Sometimes ANS will seem to produce bizarre results, but this will be because of the way that the user has set up the problem. Remember that ANS has a strong preference for stable systems.

4.4.2 The program

Routine MAIN

```
C=====================================================================MAIN   1
C      CRISP PROGRAM                                                  MAIN   2
C=====================================================================MAIN   3
C--------USE THE FOLLOWING STATEMENT AFTER CONVERTING PROGRAM TO DOUBLE MAIN   4
C--------PRECISION.  ARRAY G ALWAYS USES ONE NUMERIC STORAGE UNIT      MAIN   5
CC     REAL G                                                         MAIN   6
       COMMON /GVAR/ G(55000)                                         MAIN   7
C                                                                     MAIN   8
       LG=55000                                                       MAIN   9
       CALL MINIT(G,LG)                                               MAIN  10
       STOP                                                           MAIN  11
       END                                                            MAIN  12
```

Main 9 : set up size of working array G.

Routine MINIT

```
       SUBROUTINE MINIT(G,LG)                                         MNIT   1
C*********************************************************************MNIT   2
C      ROUTINE SETS UP DEVICE NUMBERS AND SOME CONSTANTS              MNIT   3
C      ALSO SETS UP FILES FOR FORTRAN 77                              MNIT   4
C*********************************************************************MNIT   5
C--------USE THE FOLLOWING STATEMENT AFTER CONVERTING PROGRAM TO DOUBLE MNIT   6
```



```
C--------PRECISION.  ARRAY G ALWAYS USES ONE NUMERIC STORAGE UNIT      MNIT   7
CC     REAL G                                                         MNIT   8
       DIMENSION G(LG)                                                MNIT   9
       COMMON /DEVICE/ IR1,IR4,IR5,IW2,IW4,IW6,IW7,IW8,IW9            MNIT  10
       COMMON /PARS  / PYI,ALAR,ASMVL,ZERO                            MNIT  11
       COMMON /PRECSN/ NP                                             MNIT  12
C-----------------------------------------------------------------------MNIT  13
CC     OPEN(1,FILE='CRISPOLD',FORM='UNFORMATTED')                     MNIT  14
CC     OPEN(2,FILE='CRISPNEW',FORM='UNFORMATTED')                     MNIT  15
CC     OPEN(5,FILE='CRISPDAT')                                        MNIT  16
CC     OPEN(6,FILE='CRISPOUT')                                        MNIT  17
CC     OPEN(8,FILE='PLOTDATA',FORM='UNFORMATTED')                     MNIT  18
CC     OPEN(7,FILE='CRISPSOL',FORM='UNFORMATTED')                     MNIT  19
C-----------------------------------------------------------------------MNIT  20
C      DEVICE NUMBERS R - READ ; W - WRITE .                          MNIT  21
C                                                                     MNIT  22
C     DEVICE                                                          MNIT  23
C       1      -  STOP/RESTART READ FILE (CONTAINS PREVIOUS RESULTS)  MNIT  24
C       2      -  STOP/RESTART WRITE FILE (CONTAINS CURRENT RESULTS)  MNIT  25
C       4      -  NOT USED IN THIS VERSION                            MNIT  26
C       5      -  INPUT DATA FILE (READ)                              MNIT  27
C       6      -  OUTPUT FILE (WRITE)                                 MNIT  28
C       7      -  OUT OF CORE SOLVER FILE (WRITE/READ)                MNIT  29
C       8      -  PLOT DATA FILE (WRITE) - INFO TO CREATE A PLOT OF MESH MNIT  30
C       9      -  NOT USED IN THIS VERSION                            MNIT  31
C-----------------------------------------------------------------------MNIT  32
       IR1=1                                                          MNIT  33
       IR4=4                                                          MNIT  34
       IR5=5                                                          MNIT  35
       IW2=2                                                          MNIT  36
       IW4=4                                                          MNIT  37
       IW6=6                                                          MNIT  38
       IW7=7                                                          MNIT  39
       IW8=8                                                          MNIT  40
       IW9=9                                                          MNIT  41
C-----------------------------------------------------------------------MNIT  42
C      NP = 1 FOR SINGLE PRECISION;  NP = 2 FOR DOUBLE PRECISION      MNIT  43
C-----------------------------------------------------------------------MNIT  44
       NP=1                                                           MNIT  45
CC     NP=2                                                           MNIT  46
C-----------------------------------------------------------------------MNIT  47
C      SET SOME CONSTANTS                                             MNIT  48
C-----------------------------------------------------------------------MNIT  49
       PYI=4.*ATAN(1.)                                                MNIT  50
       ALAR=1.E+17                                                    MNIT  51
       ASMVL=1.E-20                                                   MNIT  52
       ZERO=0.                                                        MNIT  53
C                                                                     MNIT  54
       WRITE(IW6,900)                                                 MNIT  55
C                                                                     MNIT  56
       CALL MAST(G,LG)                                                MNIT  57
C                                                                     MNIT  58
       RETURN                                                         MNIT  59
  900 FORMAT(1H1,120(1H*)//                                           MNIT  60
     1 11H CRISP (S1)//                                               MNIT  61
     2 33H PROGRAM LAST MODIFIED ON 27/9/85                           MNIT  62
     3 )                                                              MNIT  63
       END                                                            MNIT  64
```

MNIT 33–41 : set device numbers.
MNIT 50–53 : set some constants.
MNIT 55 : print version no. of program and date.
MNIT 57 : master-control routine.

Subroutine MAST

```
      SUBROUTINE MAST(G,LG)                                                  MAST   1
C*************************************************************************MAST   2
C     ROUTINE TO SET-UP ARRAY SIZES FOR GEOMETRY AND MAIN PART OF       MAST   3
C     THE PROGRAM.  REAL ARRAYS ARE ALLOCATED ON THE LEFT HAND SIDE     MAST   4
C     OF ARRAY G AND INTEGER ARRAYS ON THE RIGHT                        MAST   5
C*************************************************************************MAST   6
      REAL LL                                                                MAST   7
C--------USE THE FOLLOWING STATEMENT AFTER CONVERTING PROGRAM TO DOUBLE MAST   8
C--------PRECISION.  ARRAY G ALWAYS USES ONE NUMERIC STORAGE UNIT       MAST   9
CC    REAL G                                                                 MAST  10
      CHARACTER*80 TITLE                                                     MAST  11
      DIMENSION G(LG)                                                        MAST  12
      DIMENSION NAD(11),KLT(11),NTY(10),PR(10,10),PDISLD(3,5),               MAST  13
     1 PRES(3,5),V(5),FXYZ(3),CIP(3),LL(4)                                   MAST  14
      COMMON /LABEL / TITLE                                                  MAST  15
      COMMON /DEVICE/ IR1,IR4,IR5,IW2,IW4,IW6,IW7,IW8,IW9                    MAST  16
      COMMON /ELINF / LINFO(50,15)                                           MAST  17
      COMMON /PARS  / PYI,ALAR,ASMVL,ZERO                                    MAST  18
      COMMON /DEBUGS/ ID1,ID2,ID3,ID4,ID5,ID6,ID7,ID8,ID9,ID10               MAST  19
      COMMON /OUT   / IBC,IRAC,NVOS,NVOF,NMOS,NMOF,NELOS,NELOF,ISR           MAST  20
      COMMON /PRECSN/ NP                                                     MAST  21
      DATA NAD(1),NAD(2),NAD(3),NAD(4),NAD(5),NAD(6),NAD(7),                 MAST  22
     1 NAD(8),NAD(9),NAD(10),NAD(11)/                                        MAST  23
     2 1,3,3,4,4,12,19,12,12,6,6/                                            MAST  24
C-------------------------------------------------------------------------MAST  25
      READ(IR5,901)TITLE                                                     MAST  26
      WRITE(IW6,903)TITLE                                                    MAST  27
      LINK1=1                                                                MAST  28
CC    READ(IR5,*)LINK1                                                       MAST  29
CC    WRITE(IW6,906)LINK1                                                    MAST  30
C                                                                            MAST  31
      READ(IR5,*)NVTX,NEL,MXNDV,MXTYP,NDIM,IPLOT                             MAST  32
      WRITE(IW6,904)NVTX,NEL,MXNDV,MXTYP,NDIM,IPLOT                          MAST  33
      READ(IR5,*)NUMAX,MUMAX                                                 MAST  34
      WRITE(IW6,907)NUMAX,MUMAX                                              MAST  35
      IF(NUMAX.EQ.0)NUMAX=NVTX                                               MAST  36
      IF(MUMAX.EQ.0)MUMAX=NEL                                                MAST  37
C-------------------------------------------------------------------------MAST  38
C     NVRS  - NUMBER OF STRESS PARAMETERS                                MAST  39
C     NVRN  - NUMBER OF STRAIN AND STRESS COMPONENTS                     MAST  40
C     NDZ   - INDEX FOR MID-SIDE (EDGE) NODE NUMBERS                     MAST  41
C     NPL   - LENGTH OF ARRAYS NP1,NP2                                   MAST  42
C     NMATZ - MAXIMUM ADMISSIBLE MATERIAL ZONE NUMBER                    MAST  43
C     LTZ   - LARGEST ADMISSIBLE ELEMENT TYPE NUMBER                     MAST  44
C     INXL  - INDEX TO NO. OF D.O.F. OF FIRST NODE OF ELEMENT            MAST  45
C     IFR   - LIST OF NODES IN FRONT (SEE ROUTINES SFWZ,FRONTZ)          MAST  46
C     IFRZ  - SIZE OF ARRAY IFR                                          MAST  47
C-------------------------------------------------------------------------MAST  48
      NVRS=7                                                                 MAST  49
      NVRN=4                                                                 MAST  50
      IF(NDIM.NE.3)GOTO 10                                                   MAST  51
      NVRS=9                                                                 MAST  52
      NVRN=6                                                                 MAST  53
   10 NDZ=750                                                                MAST  54
      NPL=21                                                                 MAST  55
      NMATZ=10                                                               MAST  56
      LTZ=7                                                                  MAST  57
      IFRZ=300                                                               MAST  58
      INXL=20                                                                MAST  59
C-------------------------------------------------------------------------MAST  60
C     NAD   - ESTIMATE OF ADDITIONAL NODES PER ELEMENT FOR               MAST  61
C             DIFFERENT ELEMENT TYPES                                    MAST  62
C     NDEAD - TOTAL NUMBER OF ADDITIONAL NODES (AN ESTIMATE)             MAST  63
C     NDSD  - NO. OF DISPLACEMENT NODES ALONG EDGE (EXCLUDE END NODES)   MAST  64
```



```
C     NEDG  - NUMBER OF ELEMENT EDGES + 1                                MAST  65
C     NTPE  - MAXIMUM NUMBER OF NODES IN ANY ELEMENT                     MAST  66
C-------------------------------------------------------------------------MAST  67
C----------TEST FOR MXTYP < LTZ                                          MAST  68
      IF(MXTYP.GT.0.AND.MXTYP.LE.LTZ)GOTO 20                                 MAST  69
      WRITE(IW6,935)MXTYP                                                    MAST  70
  935 FORMAT(/1X,30HINADMISSIBLE VALUE FOR MXTYP =,I5,2X,                    MAST  71
     1 14H(ROUTINE MAST))                                                    MAST  72
      STOP                                                                   MAST  73
C                                                                            MAST  74
   20 CONTINUE                                                               MAST  75
C----------MAXM NO. OF EDGES (SIDES) IN AN ELEMENT (2-D)                 MAST  76
      NEDZ=MXNDV                                                             MAST  77
      NDSD=LINFO(7,MXTYP)                                                    MAST  78
      NEDG=LINFO(3,MXTYP)+1                                                  MAST  79
      NDEAD=NAD(MXTYP)*NEL                                                   MAST  80
      NTPE=LINFO(1,MXTYP)                                                    MAST  81
C-------------------------------------------------------------------------MAST  82
C     ESTIMATE THE TOTAL NUMBER OF NODES - NNE                           MAST  83
C-------------------------------------------------------------------------MAST  84
      NNE=NVTX+NDEAD                                                         MAST  85
      NNE1=NNE+1                                                             MAST  86
      IF(NVTX.GT.NDZ)NDZ=NVTX                                                MAST  87
C-------------------------------------------------------------------------MAST  88
C     NDZ+1 IS THE STARTING POINT FOR NODE NUMBERING                     MAST  89
C     FOR ADDITIONAL NODES                                               MAST  90
C     NNU   - ESTIMATE OF MAXIMUM VALUE OF USER NODE NUMBER              MAST  91
C-------------------------------------------------------------------------MAST  92
      NNU=NDZ+NDEAD                                                          MAST  93
C-------------------------------------------------------------------------MAST  94
C     SIZE OF ARRAY ITAB (SEE ROUTINES MIDSID, MIDPOR)                   MAST  95
C-------------------------------------------------------------------------MAST  96
      LDIM=NDSD+3                                                            MAST  97
      LTAB=NEDG*NEL                                                          MAST  98
C=========================================================================MAST  99
C----------INDEXES FOR ARRAYS FOR USE IN GEOMETRY PART OF PROGRAM        MAST 100
C*************************************************************************MAST 101
C     THE FOLLOWING ARRAYS ARE DYNAMICALLY ALLOCATED STORE IN            MAST 102
C     ARRAY G FOR GEOMETRY PART OF THE PROGRAM. REAL ARRAYS ARE          MAST 103
C     ALLOCATED AT THE BEGINNING OF ARRAY G WITH ARRAY INDEX             MAST 104
C     INCREASING. INTEGER ARRAYS ARE ALLOCATED TO THE END OF             MAST 105
C     ARRAY G WITH ARRAY INDEX DECREASING. THIS LEAVES A GAP             MAST 106
C     BETWEEN THE REAL AND INTEGER ARRAYS WHICH IS USED AS A             MAST 107
C     BUFFER (FOR SOLUTION) IN THE MAIN PART OF THE PROGRAM.             MAST 108
C                                                                        MAST 109
C    G(1) -  G(L1-1) = NODAL COORDINATES .................XYZ(NDIM,NNE) MAST 110
C   G(M1) -    G(LG) = ELEMENT-NODAL CONNECTIVITY.......NCONN(NTPE,NEL) MAST 111
C   G(M2) -  G(M1-1) = MATERIAL PROPERTY NUMBER.................MAT(NEL) MAST 112
C   G(M3) -  G(M2-1) = ELEMENT TYPE NUMBER.....................LTYP(NEL) MAST 113
C   G(M4) -  G(M3-1) = USER ELEMENT NUMBERS...................MRELVV(NEL) MAST 114
C   G(M5) -  G(M4-1) = PROGRAM ELEMENT NUMBERS..............MREL(MUMAX) MAST 115
C   G(M6) -  G(M5-1) = USER NODE NUMBERS.....................NRELVV(NNE) MAST 116
C   G(M7) -  G(M6-1) = PROGRAM NODE NUMBERS...................NREL(NNU) MAST 117
C   G(M8) -  G(M7-1) = INDEX OF FIRST D.O.F. OF NODES.........NW(NNE+1) MAST 118
C   G(M9) -  G(M8-1) = NO. OF D.O.F. OF EACH NODE..............NQ(NNE) MAST 119
C  G(M10) -  G(M9-1) = TABLE OF ELEMENT EDGES...........ITAB(LDIM,LTAB) MAST 120
C  G(M11) - G(M10-1) = USER ELEMENT NOS. IN FRONTAL ORDER.....MFRU(NEL) MAST 121
C  G(M12) - G(M11-1) = ELEMENT NO. IN FRONTAL ORDER.........MFRN(MUMAX) MAST 122
C  G(M13) - G(M12-1) = FRONTAL DESTINATION OF NODES..........NDEST(NNE) MAST 123
C  G(M14) - G(M13-1) = NODE NOS. OF ELEMENT..................NLST(NTPE) MAST 124
C  G(M15) - G(M14-1) = LIST OF NODES (AND D.O.F.) IN FRONT....IFR(IFRZ) MAST 125
C  G(M16) - G(M15-1) = INDEX OF ONE END OF ELEMENT EDGE........NP1(NPL) MAST 126
C  G(M17) - G(M16-1) = INDEX OF OTHER END OF ELEMENT EDGE......NP2(NPL) MAST 127
C                                                                        MAST 128
C     IN THE ABOVE                                                       MAST 129
C                                                                        MAST 130
```

```
C     LDIM   - MAXIMUM NUMBER OF (DISPLACEMENT) NODES ALONG EDGE + 3    MAST 131
C     LTAB   - TOTAL NUMBER OF ELEMENT EDGES (ESTIMATE)                 MAST 132
C     MUMAX  - MAXIMUM VALUE OF USER ELEMENT NUMBER                     MAST 133
C              (THIS NEED NOT BE EQUAL TO THE TOTAL NO. OF ELEMENTS)    MAST 134
C     NTPE   - MAXIMUM NO. OF NODES IN ANY ELEMENT IN MESH              MAST 135
C     NDIM   - NO. OF DIMENSIONS TO PROBLEM (2 OR 3)                    MAST 136
C     NEL    - TOTAL NUMBER OF ELEMENTS IN MESH                         MAST 137
C     NNE    - TOTAL NUMBER OF NODES IN MESH (ESTIMATE)                 MAST 138
C     NNU    - ESTIMATE OF MAXIMUM VALUE OF USER NODE NUMBER            MAST 139
C     NPL    - LENGTH OF ARRAYS NP1,NP2                                 MAST 140
C*****************************************************************     MAST 141
      L1=1+NNE*NDIM*NP                                                  MAST 142
      LZ=L1                                                             MAST 143
      M1=LG-NTPE*NEL+1                                                  MAST 144
      M2=M1-NEL                                                         MAST 145
      M3=M2-NEL                                                         MAST 146
      M4=M3-NEL                                                         MAST 147
      M5=M4-MUMAX                                                       MAST 148
      M6=M5-NNE                                                         MAST 149
      M7=M6-NNU                                                         MAST 150
      M8=M7-NNE-1                                                       MAST 151
      M9=M8-NNE                                                         MAST 152
      M10=M9-LTAB*LDIM                                                  MAST 153
      M11=M10-NEL                                                       MAST 154
      M12=M11-MUMAX                                                     MAST 155
      M13=M12-NNE                                                       MAST 156
      M14=M13-NTPE                                                      MAST 157
      M15=M14-IFRZ                                                      MAST 158
      M16=M15-NPL                                                       MAST 159
      M17=M16-NPL                                                       MAST 160
      MZ=M17                                                            MAST 161
      IF(MZ.GT.LZ)GO TO 40                                              MAST 162
      MORE=LZ-MZ+1                                                      MAST 163
      WRITE(IW6,908)MORE                                                MAST 164
      STOP                                                              MAST 165
C                                                                       MAST 166
   40 KSTO=LG-MZ+LZ-1                                                   MAST 167
      WRITE(IW6,910)KSTO,LG                                             MAST 168
C                                                                       MAST 169
      CALL MARKZ(NVTX,NEL,NUMAX,MUMAX,NTPE,MXNDV,NNE,NNE1,NN,           MAST 170
     1   NNU,NNZ,LTAB,LDIM,NDIM,NDF,NDZ,IFRZ,MCORE,MAXNFZ,              MAST 171
     2   NPL,LTZ,KLT,NMATZ,INXL,IPLOT,                                  MAST 172
     3   G(1),G(M1),G(M2),G(M3),G(M4),G(M5),G(M6),G(M7),G(M8),          MAST 173
     4   G(M9),G(M10),G(M11),G(M12),G(M13),G(M14),G(M15),               MAST 174
     5   G(M16),G(M17),ND,NCORET,MDZ)                                   MAST 175
      IF(ID8.EQ.0)GOTO 45                                               MAST 176
      WRITE(IW6,925)NNE,NNU,LDIM,LTAB,NTPE,IFRZ,NPL                     MAST 177
  925 FORMAT(/1X,6HNNE = ,I5,3X,6HNNU = ,I5,3X,7HLDIM = ,3X,            MAST 178
     1 7HLTAB = ,I5,3X,7HNTPE = ,I5,3X,7HIFRZ = ,I5,3X,6HNPL = ,I5)     MAST 179
      WRITE(IW6,920)(G(JK),JK=1,L1)                                     MAST 180
      WRITE(IW6,930)(G(JK),JK=M17,LG)                                   MAST 181
  920 FORMAT(//1X,4HREAL/(1X,10F10.2/))                                 MAST 182
  930 FORMAT(//1X,7HINTEGER/(1X,20I6/))                                 MAST 183
   45 CONTINUE                                                          MAST 184
C                                                                       MAST 185
      CALL MAXVAL(IW6,KLT,LTZ,NDIM,NVRN,NDMX,NPMX,NIP,NS,NB,NL,         MAST 186
     1   NPT,NSP,NPR,NMT,MDFE,KES,NVPN,LV,MXEN,MXLD,MXFXT)              MAST 187
C=================================================================     MAST 188
C----------THE FOLLOWING INDEXES FOR ARRAYS ARE FOR USE IN THE          MAST 189
C----------MAIN (ANALYSIS) PART OF THE PROGRAM                          MAST 190
C-----------------------------------------------------------------     MAST 191
C    G(1) -  G(L1-1) = COORDINATES OF NODES.............XYZ(NDIM,NN)    MAST 192
C   G(L1) -  G(L2-1) = INCREMENTAL DISPLACEMENTS.........DI(NDF)       MAST 193
C   G(L2) -  G(L3-1) =  CUMULATIVE DISPLACEMENTS.........DA(NDF)       MAST 194
C   G(L3) -  G(L4-1) = STRESS PARS AT GAUSS POINTS. VARINT(NVRS,NIP,NEL) MAST 195
C   G(L4) -  G(L5-1) = INCREMENTAL NODAL LOADS.....P(NDF)              MAST 196
```



```
C   G(L5) -  G(L6-1) =  CUMULATIVE NODAL LOADS.....PT(NDF)            MAST 197
C   G(L6) -  G(L7-1) = NODAL LOADS FOR INCREMENTAL BLOCK.....PIB(NDF) MAST 198
C   G(L7) -  G(L8-1) = REACTIONS TO EARTH.....REAC(NDF)               MAST 199
C   G(L8) -  G(L9-1) = OUT OF BALANCE LOADS.....PCOR(NDF)             MAST 200
C   G(L9) - G(L10-1) = TOTAL EQUILIBRIUM LOADS.....PEQT(NDF)          MAST 201
C  G(L10) - G(L11-1) = INCREMENTAL POINT LOADS.....XYFT(NDF)          MAST 202
C  G(L11) - G(L12-1) = POINT LOADS FOR INCREMENTAL BLOCK.....XYFIB(NDF) MAST 203
C  G(L12) - G(L13-1) = STRAIN PARS AT GAUSS POINTS....STR(NVRN,NIP,NEL) MAST 204
C  G(L13) - G(L14-1) = EXCAVATION LOADS FOR INCR BLOCK.....PEXIB(NDF) MAST 205
C  G(L14) - G(L15-1) = EXCAVATION LOADS FOR INCREMENT.....PEXI(NDF)   MAST 206
C  G(L15) - G(LS1-1) = INSITU EQUILIBRIUM POINT LOADS.....PCONI(NDF)  MAST 207
C  G(LS1) - G(LS2-1) = D (STRESS - STRAIN ) MATRIX.....D(NS,NS)       MAST 208
C  G(LS2) - G(LS3-1) = DISP. NODE COORDS. OF ELEMENT...ELCOD(NDIM,NDMX) MAST 209
C  G(LS3) - G(LS4-1) = DERIVATIVES OF SHAPE FUNS(LOCAL)...DS(NDIM,NDMX) MAST 210
C  G(LS4) - G(LS5-1) = SHAPE FUNCTIONS.....SHFN(NDMX)                 MAST 211
C  G(LS5) - G(LS6-1) = CARTESIAN DERIV. OF SHAPE FUNS..CARTD(NDIM,NDMX) MAST 212
C  G(LS6) - G(LS7-1) = STRAIN - DISPLACEMENT MATRIX.....B(NS,NB)      MAST 213
C  G(LS7) - G(LS8-1) = D * B MATRIX.....DB(NS,NB)                     MAST 214
C  G(LS8) - G(LS9-1) = ELEMENT FORCE MATRIX.....FT(NDIM,NDMX)         MAST 215
C  G(LS9) -G(LS10-1) = ELEMENT STIFFNESS MATRIX.....SS(NB,NB)         MAST 216
C G(LS10) - G(LC1-1) = UPPER TRIANGULAR ELEMENT STIFF MATRIX....ES(KES) MAST 217
C  G(LC1) - G(LC2-1) = P.P.NODE COORDS OF ELEMENT.....ELCODP(NDIM,NPMX) MAST 218
C  G(LC2) - G(LC3-1) = PORE PRESSURE GRADIENTS.....E(NDIM,NPMX)       MAST 219
C  G(LC3) - G(LC4-1) = PERMEABILITY * POREPRES GRADIENTS..PE(NDIM,NPMX) MAST 220
C  G(LC4) - G(LC5-1) = AN ARRAY FOR LINK MATRIX.....RN(NB)            MAST 221
C  G(LC5) - G(LC6-1) = AN ARRAY FOR LINK MATRIX.....AA(NPMX)          MAST 222
C  G(LC6) - G(LC7-1) = FLOW MATRIX.....ETE(NPMX,NPMX)                 MAST 223
C  G(LC7) - G(LC8-1) = LINK MATRIX.....RLT(NB,NPMX)                   MAST 224
C                                                                     MAST 225
C   WHERE                                                            MAST 226
C     KES   - MAXM SIZE OF UPPER TRIANGULAR ELEMENT STIFFNESS MATRIX  MAST 227
C     NB    - SIZE OF STIFFNESS MATRIX SS ( = NDIM * NDMX )           MAST 228
C     NDF   - TOTAL NO. OF D.O.F. IN PROBLEM                          MAST 229
C     NDIM  - DIMENSION OF PROBLEM (2 OR 3)                           MAST 230
C     NDMX  - MAXM NO. OF DISP. NODES IN ANY ELEMENT IN MESH          MAST 231
C     NEL   - TOTAL NO. OF ELEMENTS IN MESH                           MAST 232
C     NIP   - MAXM NO. OF INTEGRATION POINTS IN ANY ELEMENT IN MESH   MAST 233
C     NN    - TOTAL NO. OF NODES IN MESH                              MAST 234
C     NPMX  - MAXM NO. OF PORE-PRESSURE NODES IN ANY ELEMENT IN MESH  MAST 235
C     NS    - SIZE OF D - MATRIX ( = NO. OF STRESS/STRAIN COMPONENTS) MAST 236
C     NVRN  - NO. OF STRAIN (AND STRESS) COMPONENTS (NVRN = NS)       MAST 237
C     NVRS  - NO. OF STRESS COMPONENTS PLUS PARAMETERS (U,P,Q ETC.)   MAST 238
C-----------------------------------------------------------------    MAST 239
C----------INDEXES FOR REAL ARRAYS - LEFT HAND SIDE                   MAST 240
      L1=1+NDIM*NN*NP                                                 MAST 241
      L2=L1+NDF*NP                                                    MAST 242
      L3=L2+NDF*NP                                                    MAST 243
      L4=L3+NVRS*NIP*NEL*NP                                           MAST 244
      L5=L4+NDF*NP                                                    MAST 245
      L6=L5+NDF*NP                                                    MAST 246
      L7=L6+NDF*NP                                                    MAST 247
      L8=L7+NDF*NP                                                    MAST 248
      L9=L8+NDF*NP                                                    MAST 249
      L10=L9+NDF*NP                                                   MAST 250
      L11=L10+NDF*NP                                                  MAST 251
      L12=L11+NDF*NP                                                  MAST 252
      L13=L12+NDF*NP                                                  MAST 253
      L14=L13+NDF*NP                                                  MAST 254
      L15=L14+NDF*NP                                                  MAST 255
      LS1=L15+NVRN*NIP*NEL*NP                                         MAST 256
      LS2=LS1+NS*NS*NP                                                MAST 257
      LS3=LS2+NDIM*NDMX*NP                                            MAST 258
      LS4=LS3+NDIM*NDMX*NP                                            MAST 259
      LS5=LS4+NDMX*NP                                                 MAST 260
      LS6=LS5+NDIM*NDMX*NP                                            MAST 261
      LS7=LS6+NS*NB*NP                                                MAST 262
```

```
      LS8=LS7+NS*NB*NP                                                  MAST 263
      LS9=LS8+NDIM*NDMX*NP                                              MAST 264
      LS10=LS9+NB*NB*NP                                                 MAST 265
      LC1=LS10+KES*NP                                                   MAST 266
      LC2=LC1+NDIM*NPMX*NP                                              MAST 267
      LC3=LC2+NDIM*NPMX*NP                                              MAST 268
      LC4=LC3+NDIM*NPMX*NP                                              MAST 269
      LC5=LC4+NB*NP                                                     MAST 270
      LC6=LC5+NPMX*NP                                                   MAST 271
      LC7=LC6+NPMX*NPMX*NP                                              MAST 272
      LC8=LC7+NB*NPMX*NP                                                MAST 273
      LZ=LC8                                                            MAST 274
C-----------------------------------------------------------------------MAST 275
C   G(N1) -    G(LG) = ELEMENT-NODAL CONNECTIVITY.......NCONN(NTPE,NEL) MAST 276
C   G(N2) -  G(N1-1) = MATERIAL PROPERTY NUMBER.................MAT(NEL) MAST 277
C   G(N3) -  G(N2-1) = ELEMENT TYPE NUMBER.....................LTYP(NEL) MAST 278
C   G(N4) -  G(N3-1) = USER ELEMENT NUMBERS...................MRELVV(NEL) MAST 279
C   G(N5) -  G(N4-1) = PROGRAM ELEMENT NUMBERS................MREL(NUMAX) MAST 280
C   G(N6) -  G(N5-1) = USER NODE NUMBERS.....................NRELVV(NN) MAST 281
C   G(N7) -  G(N6-1) = PROGRAM NODE NUMBERS...................NREL(NNZ) MAST 282
C   G(N8) -  G(N7-1) = INDEX OF FIRST D.O.F. OF NODES.........NW(NNOD1) MAST 283
C   G(N9) -  G(N8-1) = NO. OF D.O.F. OF EACH NODE...............NQ(NN) MAST 284
C  G(N10) - G(N11-1) = INDICATOR OF ELEMENT CHANGES............JEL(NEL) MAST 285
C  G(N11) - G(N12-1) = INDICTORS OF RESTRIANED VARIABLES......IDFX(NDF) MAST 286
C  G(N12) - G(N11-1) = FRONTAL DESTINATION OF NODES...........NDEST(NN) MAST 287
C  G(N13) - G(N12-1) = INDEX OF ONE END OF ELEMENT EDGE.......NP1(NPL) MAST 288
C  G(N14) - G(N13-1) = INDEX OF OTHER END OF ELEMENT EDGE.....NP2(NPL) MAST 289
C  G(NS1) - G(N14-1) = LIST OF NODES (AND D.O.F.) IN FRONT....IFR(IFRZ) MAST 290
C  G(NS2) - G(NS1-1) = DESTINATION IN FRONT OF ELEMENT D.O.F..NDL(MDFE) MAST 291
C  G(NS3) - G(NS2-1) = INDEX TO POREPRESSURE NODES OF ELEMENT.NWL(NPMX) MAST 292
C  G(NS4) - G(NS3-1) = STRESS STATE INDICATOR FOR MODEL5..NMOD(NIP,NEL) MAST 293
C                      (NOT USED IN THIS VERSION)                      MAST 294
C   WHERE                                                              MAST 295
C                                                                      MAST 296
C     IFRZ   - LENGTH OF ARRAY IFR                                     MAST 297
C     MDFE   - MAXM NO. OF D.O.F. IN ANY ELEMENT IN MESH               MAST 298
C     MUMAX  - MAXM VALUE OF USER ELEMENT NUMBER                       MAST 299
C     NNZ    - MAXM VALUE OF USER NODE NUMBER                          MAST 300
C     NNOD1  - NN + 1                                                  MAST 301
C-----------------------------------------------------------------------MAST 302
C----------INDEXES FOR INTEGER ARRAYS - RIGHT HAND SIDE                MAST 303
      NNOD1=NN+1                                                        MAST 304
      N1=LG-NTPE*NEL+1                                                  MAST 305
      N2=M1-NEL                                                         MAST 306
      N3=N2-NEL                                                         MAST 307
      N4=N3-NEL                                                         MAST 308
      N5=N4-MUMAX                                                       MAST 309
      N6=N5-NN                                                          MAST 310
      N7=N6-NNZ                                                         MAST 311
      N8=N7-NNOD1                                                       MAST 312
      N9=N8-NN                                                          MAST 313
      N10=N9-NEL                                                        MAST 314
      N11=N10-NDF                                                       MAST 315
      N12=N11-NN                                                        MAST 316
      N13=N12-NPL                                                       MAST 317
      N14=N13-NPL                                                       MAST 318
      NS1=N14-IFRZ                                                      MAST 319
      NS2=NS1-MDFE                                                      MAST 320
      NS3=NS2-NPMX                                                      MAST 321
      NS4=NS3-NIP*NEL                                                   MAST 322
      NZ=NS4                                                            MAST 323
C-----------------------------------------------------------------------MAST 324
C     CALCULATE SIZE OF WORKING REGION                                  MAST 325
C-----------------------------------------------------------------------MAST 326
      NWORK=NZ-LZ                                                       MAST 327
      KVARS=LG+LZ-NZ                                                    MAST 328
```



```
      NCORET=NCORET*NP                                                  MAST 329
      MCORE=MCORE*NP                                                    MAST 330
      CALL SHFTIB(IW6,G(N7),G(M7),NNZ)                                  MAST 349
      CALL SHFTIB(IW6,G(N8),G(M8),NNOD1)                                MAST 350
      CALL SHFTIB(IW6,G(N13),G(M16),NPL)                                MAST 351
      CALL SHFTIB(IW6,G(N14),G(M17),NPL)                                MAST 352
C                                                                       MAST 353
      CALL CPW(NN,NEL,NDF,NNOD1,NTPE,NIP,NVRS,NVRN,NDIM,               MAST 354
     1   MUMAX,NDZ,IFRZ,NNZ,NDMX,NPMX,NS,NB,NL,NPR,NMT,                MAST 355
     2   NPT,NSP,NPL,MDFE,KES,NVPN,INXL,MXEN,MXLD,MXFXT,               MAST 356
     2   LV,MCORE,LINK1,NVTX,ND,MDZ,NEDZ,                              MAST 357
     3   G(1),G(L1),G(L2),G(L3),G(L4),G(L5),G(L6),G(L7),G(L8),         MAST 358
     3   G(L9),G(L10),G(L11),G(L12),G(L13),G(L14),G(L15),              MAST 359
     4   G(LS1),G(LS2),G(LS3),G(LS4),G(LS5),G(LS6),G(LS7),             MAST 360
     5   G(LS8),G(LS9),G(LS10),G(LC1),G(LC2),G(LC3),G(LC4),            MAST 361
     6   G(LC5),G(LC6),G(LC7),G(N1),G(N2),G(N3),G(N4),                 MAST 362
     6   G(N5),G(N6),G(N7),G(N8),G(N9),G(N10),G(N11),                  MAST 363
     7   G(N12),G(N13),G(N14),G(NS1),G(NS2),G(NS3),G(NS4),             MAST 364
     8   CIP,LL,V,FXYZ,PR,PDISLD,PRES,NTY,G(LZ),NWORK,                 MAST 365
     9   NOIB,TTIME,TGRAV,IUPD,ICOR,IDCHK,INCT)                        MAST 366
C                                                                       MAST 367
      CALL ANS(NN,NEL,NDF,NNOD1,NTPE,NIP,NVRS,NVRN,NDIM,               MAST 368
     1   MUMAX,NDZ,IFRZ,NNZ,NDMX,NPMX,NS,NB,NL,NPR,NMT,                MAST 369
     2   NPT,NSP,NPL,MDFE,KES,NVPN,INXL,MXEN,MXLD,MXFXT,               MAST 370
     2   LV,NVTX,ND,                                                   MAST 371
     3   G(1),G(L1),G(L2),G(L3),G(L4),G(L5),G(L6),G(L7),G(L8),         MAST 372
     3   G(L9),G(L10),G(L11),G(L12),G(L13),G(L14),G(L15),              MAST 373
     4   G(LS1),G(LS2),G(LS3),G(LS4),G(LS5),G(LS6),G(LS7),             MAST 374
     5   G(LS8),G(LS9),G(LS10),G(LC1),G(LC2),G(LC3),G(LC4),            MAST 375
     6   G(LC5),G(LC6),G(LC7),G(N1),G(N2),G(N3),G(N4),                 MAST 376
     6   G(N5),G(N6),G(N7),G(N8),G(N9),G(N10),G(N11),                  MAST 377
     7   G(N12),G(N13),G(N14),G(NS1),G(NS2),G(NS3),G(NS4),             MAST 378
     8   CIP,LL,V,FXYZ,PR,PDISLD,PRES,NTY,G(LZ),NWORK,                 MAST 379
     9   NOIB,TTIME,TGRAV,IUPD,ICOR,IBC,IDCHK,INCT)                    MAST 380
      RETURN                                                            MAST 381
  901 FORMAT(A)                                                         MAST 382
  903 FORMAT(/1X,A)                                                     MAST 383
  904 FORMAT(//                                                         MAST 384
     1 10X,46HTOTAL NUMBER OF VERTEX NODES.................=,I8/       MAST 385
     2 10X,46HTOTAL NUMBER OF ELEMENTS.....................=,I8/       MAST 386
     5 10X,46HMAXIMUM NUMBER OF VERTEX NODES IN AN ELEMENT.=,I8/       MAST 387
     6 10X,46HELEMENT TYPE WITH MAXIMUM NUMBER OF NODES....=,I8/       MAST 388
     8 10X,46HNUMBER OF DIMENSIONS IN PROBLEM..............=,I8/       MAST 389
     9 10X,46HPLOTTING CODE................................=,I8//)     MAST 390
CC906 FORMAT(/1X,14HLINK NUMBER = ,I6)                                  MAST 391
  907 FORMAT(                                                           MAST 392
     1 10X,46HMAXIMUM VALUE OF VERTEX NODE NUMBER..........=,I8/       MAST 393
     2 10X,46HMAXIMUM VALUE OF ELEMENT NUMBER..............=,I8//)     MAST 394
  908 FORMAT(/1X,28HINCREASE SIZE OF ARRAY G BY ,I8,13H FOR GEOMETRY,   MAST 395
     1 1X,30HPART OF PROGRAM (ROUTINE MAST)/)                           MAST 396
      INCORE=NCORET-MCORE                                               MAST 331
      NBUFF=NWORK-MCORE                                                 MAST 332
      WRITE(IW6,915)LG,KVARS,NWORK,MCORE,NBUFF,INCORE                   MAST 333
C----------ADDITIONAL ARRAYS CREATED IN ROUTINES UPARAL/UPOUT           MAST 334
      MOUT=13*NIP*NEL+5*NEL                                             MAST 335
      MINM=MOUT                                                         MAST 336
      IF(MINM.GT.MCORE)MINM=MCORE                                       MAST 337
      IF(NWORK.GT.MINM)GOTO 50                                          MAST 338
      INCLG=MCORE-NWORK                                                 MAST 339
      WRITE(IW6,912)INCLG                                               MAST 340
      STOP                                                              MAST 341
   50 CONTINUE                                                          MAST 342
      IF(NWORK.GE.NCORET)WRITE(IW6,940)                                 MAST 343
      IF(NWORK.LT.NCORET)WRITE(IW6,950)                                 MAST 344
C-----------------------------------------------------------------------MAST 345
C     SHIFT NRELVV,NREL,NW,NP1,NP2 TO NEW LOCATION                      MAST 346
```

```
C-----------------------------------------------------------------------MAST 347
      CALL SHFTIB(IW6,G(N6),G(M6),NN)                                  MAST 348
  910 FORMAT(47H ARRAY STORE - USED IN GEOMETRY PART OF PROGRAM,       MAST 397
     1 I7,2X,17HOUT OF ALLOCATED ,I7//120(1H*))                        MAST 398
  912 FORMAT(/10X,42HTO PROVIDE MINIMUM CORE TO SOLVE EQUATIONS/       MAST 399
     1 10X,29HINCREASE SIZE OF ARRAY G BY =,I10,2X,14H(ROUTINE MAST)// MAST 400
     1 1X,120(1H*))                                                    MAST 401
  915 FORMAT(//1X,120(1H*)//                                           MAST 402
     1 10X,51HTOTAL ALLOCATION OF STORE FOR G.....=,I10/               MAST 403
     2 10X,51HSTORE FOR MAIN ARRAYS.....=,I10/                         MAST 404
     3 10X,51HWORKING REGION LEFT FOR SOLVING EQUATIONS.....=,I10/     MAST 405
     4 10X,51HMINIMUM CORE TO SOLVE EQUATIONS.....=,I10/               MAST 406
     5 10X,51HAMOUNT OF STORE LEFT FOR BUFFER.....=,I10/               MAST 407
     6 10X,51HSIZE OF BUFFER FOR IN-CORE SOLUTION.....=,I10/           MAST 408
     7 10X,51H(BUFFER SIZE TO STORE ALL THE REDUCED COEFFICIENTS)/)    MAST 409
  940 FORMAT(/10X,28HEQUATIONS ARE SOLVED IN-CORE//1X,120(1H*))        MAST 410
  950 FORMAT(/10X,32HEQUATIONS ARE SOLVED OUT-OF-CORE//1X,120(1H*))    MAST 411
      END                                                              MAST 412
```

MAST 26–27 : read title of analysis.

MAST 32–37 : read and write information on the geometry of the mesh.
NVTX – the total number of vertex nodes in mesh.
NEL – the number of elements in mesh.
NDIM – the number of dimensions to problem.

MAST 49–59 : parameters which govern the size of principle (main) arrays and which depend on the type of problem being analysed (i.e. whether 2-D or 3-D) are set up.

MAST 80 : calculate NDEAD, which is an estimate of the (total) no. of additional nodes in the mesh (this includes both displacement and pore pressure nodes) – the latter only for consolidation elements. This estimate is intended to be more than the actual no. of additional nodes.

MAST 85–93 : estimate of total no. of nodes (NNU); this includes the vertex nodes.

MAST 142–163 : set up indexes, allocating store to various arrays in G for use in the geomtry part of the program.

MAST 170–175 : geometry part of the program. Calculate nodal co-ordinates of additional nodes and number them, starting with 751. Calculate total no. of d.o.f. in mesh.

MAST 186–187 : set up maximum size of arrays and maximum values of some parameters.

MAST 241–274 : re-define indexes for various REAL arrays at the beginning of array G for use in the main part of the program.

MAST 304–323 : set up indexes for various INTEGER arrays at the end of array G for use in the main part of the program.

MAST 327–341 : calculate size of the working area and determine whether there is enough core store for solving equations either in-core or out-of-core.

MAST 348–352 : shift the INTEGER arrays evaluated in the geometry part of the program, to new position, for use in the rest of the program.

MAST 354–366 : Routine CPW reads the control data and sets up the *in situ* stresses.

MAST 368–380 : analysis (main) part of the program. Routine ANS is a control routine which sets up and delegates tasks to other control routines to carry out the analysis.

4.5 CRISP SUBROUTINE HIERARCHY

Fig. 4.2 shows all the subroutines in CRISP, arranged to show the structure of the program.

4.6 ADDING NEW FEATURES

Many institutions and individuals around the world have versions of CRISP which differ in some respects from the version presented here. Every time we modified the program, we stored in a computer file the actual editing instructions (together with an explanation of their purpose). We also updated the program version number and date of last modification. Unfortunately, we did not keep a record of the version given to all those who passed through our office or who wrote in. Inevitably, they changed the version number, so confusion reigns.

The book version will presumably become the most widely distributed, so we call it version S (or CRISP-S). 'S' is for standard: originally it was S for small, but really that is not appropriate. Extending the program is not an endeavour to be lightly undertaken, but the explanations in the book are designed to assist. Adding a new soil model is likely to be a popular extension: see Appendix D for details.

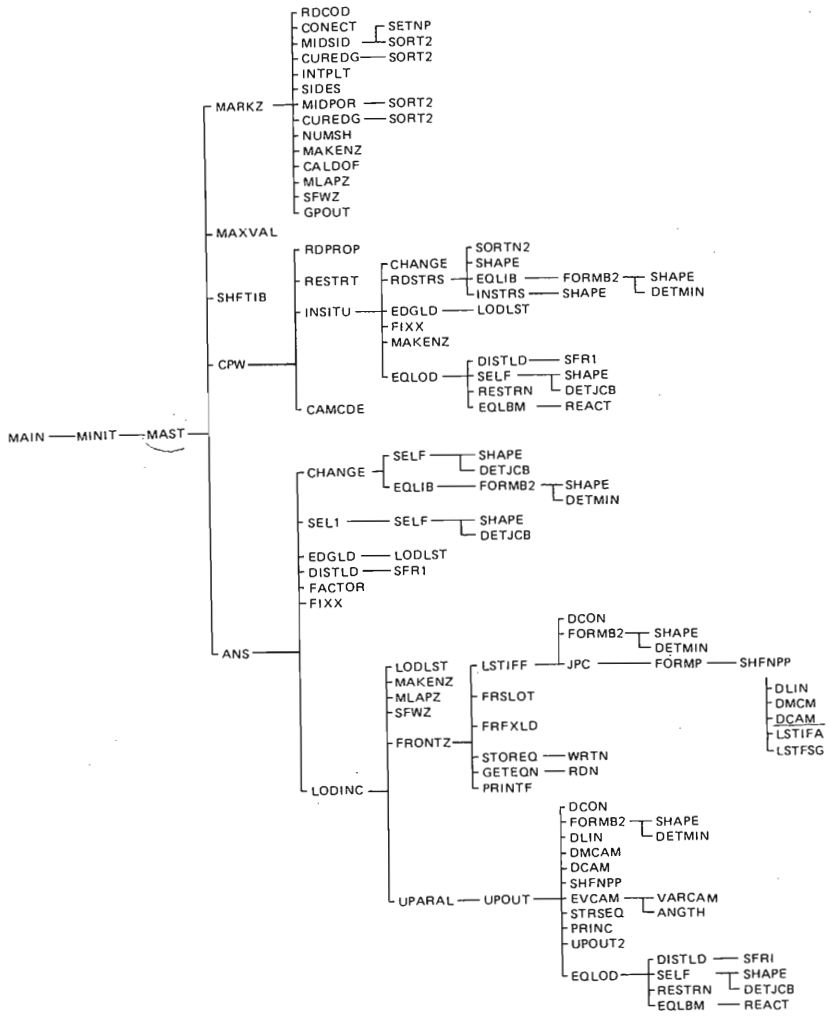


Fig. 4.2 – Subroutine hierarchy for CRISP–S

5

Cam-clay in Finite Element Analysis

5.1 INTRODUCTION

Chapter 2 described the critical state soil models entirely in relation to the standard triaxial test for soils. Thus it was possible to describe the effective stress state of a soil sample by just two stress parameters (p' and q). The reader may have wondered (and indeed we did not attempt to explain) why these two parameters were chosen. In fact the definitions of p' and q that were given in Chapter 2 were simplified versions of the full definitions for general three-dimensional stress states that we present in section 5.2. To extend the models to more general two- and three-dimensional stress states, some additional assumptions are necessary. These are also covered in section 5.2.

How the incremental stress–strain relations are actually implemented in CRISP is described in section 5.3.

When performing a finite element analysis using one of the critical state models, a necessary preliminary is to define the *in situ* stress state. We describe how this is done and also give guidance on how the critical state parameters M, Γ, λ and κ should be selected in sections 5.4 and 5.5.

5.2 GENERALISING CAM-CLAY

5.2.1 Three-dimensional stress states

To generalise the Cam-clay model to two- and three-dimensional stress states, we replace the definitions of p' and q given in Chapter 2 by

$$p' = (\sigma_x' + \sigma_y' + \sigma_z')/3, \tag{5.1}$$

$$q = (1/\sqrt{2})\sqrt{\{(\sigma_x - \sigma_y)^2 + (\sigma_y - \sigma_z)^2 + (\sigma_z - \sigma_x)^2 + 6\tau_{xy}^2 + 6\tau_{yz}^2 + 6\tau_{zx}^2\}}. \tag{5.2}$$

Note that these definitions reduce to those of Chapter 2 for triaxial stress conditions. p' and q are invariants of the effective stress tensor: for a given three-dimensional stress state, p' and q will always have the same values regardless of the orientation of the reference axes (x, y, z).† Another set of invariants of the stress tensor are the principal stresses, and p' and q can be regarded as describing the position of a point in principal stress space. The co-ordinates of a point $(\sigma_a', \sigma_b', \sigma_c')$ can be decomposed into a distance along the hydrostatic axis and a distance from the hydrostatic axis (Fig. 5.1). $(\sqrt{3})p'$ is equivalent to the distance along the hydrostatic axis, and $(\sqrt{2}/\sqrt{3})q$ is equivalent to the perpendicular distance from the hydrostatic axis.

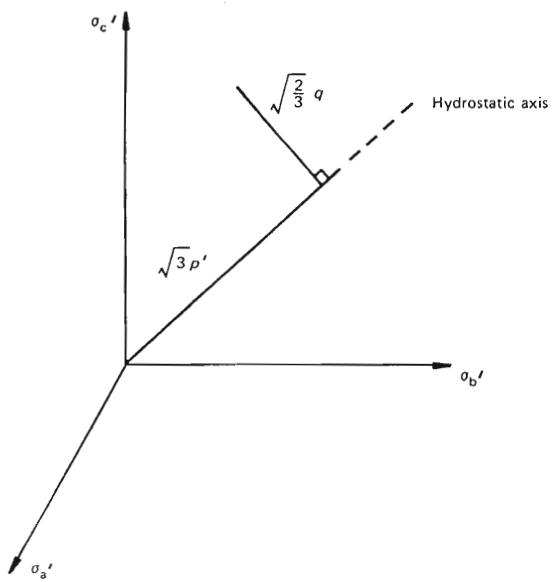


Fig. 5.1 – The significance of p' and q in principal stress space

† Quantities describing the state of a material at a point are often described by the mathematical entities of scalar, vector or tensor. An example of a scalar is pore pressure; and example of a vector is a force; an example of a tensor is stress. The difference between these entities is the transformation law that is necessary to calculate the entity in a co-ordinate system (x', y', z'), given values in an inclined co-ordinate system (x, y, z). A brief, yet fairly complete, account of all the relevant mathematics is given in Chapter 3 and Appendix A of the text by Richards (1977). Readers without the time or stamina to pursue the mathematics of tensors should not be intimidated. To perform a two-dimensional transformation of stresses, one can use the Mohr's circle construction. Engineers who understand Mohr's circles already know 90% of what there is to know about tensors. The rest is just notation.

In Chapter 2 we were limited to the triaxial plane in principal stress space (this is the plane including the σ_a and hydrostatic axes on which $\sigma_b = \sigma_c$ (= σ_r in triaxial tests)). The Cam-clay models are generalised to the whole of principal stress space by rotating the yield loci and CSL to give the result shown in Fig. 5.2. Mathematically this rotation is achieved by using (5.1) as the definition of p' and (5.2) as the definition of q for all the Cam-clay (or modified Cam-clay) relationships described in Chapter 2.

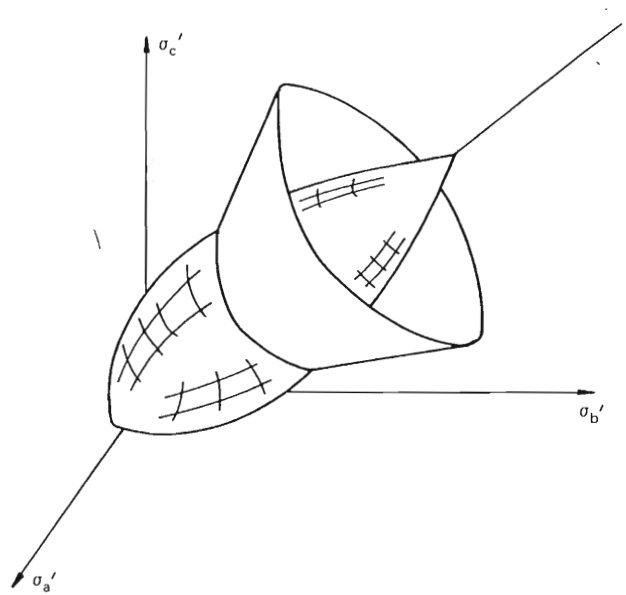


Fig. 5.2 – The Cam-clay yield locus in principal effective stress space

Thus the CSL in a (p', q) plot becomes the 'critical state cone' in principal stress space. Obviously, there is a similarity with the Drucker–Prager cone of section 2.3.2, but of course the critical sate cone is a locus of failure points, not an elasto-plastic yield surface.

The generalisation of Cam-clay in this way follows the simplest and most mathematically convenient approach. Most of the experimental evidence is that the Mohr–Coulomb surface (Fig. 2.6) would be a better generalisation than the Drucker–Prager cone (Fig. 2.7) (Bishop, 1966). However, the adoption of the Mohr–Coulomb criterion means that the critical state parameter M is dependent on the value of the intermediate principal stress. The use of the simpler approach means that it is always possible to compare directly finite element calculations with an equivalent triaxial test. (But see section 5.4.2.)

5.2.2 The 'other' elastic property

The assumption made for the Cam-clay models in Chapter 2 about elastic behaviour (volumetric strains given by the κ-line equation, zero shear strains) causes a small difficulty in implementing the models in a finite element program. The assumption of zero shear strains implies an infinite value of the shear modulus (G). The most straightforward way of circumventing this difficulty is to allow the program to calculate realistic elastic shear strains inside the yield locus. In calculating the terms of the D matrix for Cam-clay under the yield locus, the effective stress bulk modulus is calculated as

$$K' = \frac{Vp'}{\kappa}. \tag{5.3}$$

(This equation is obtained by differentiating the equation of the κ-line.) The second independent elastic property is chosen by using either an assumed constant value of ν' or an assumed constant value of G. The pros and cons of each option are discussed in section 5.4. The addition of the extra elastic strains makes very little difference to the predictions of the Cam-clay models. In triaxial tests, drained and undrained stress paths (and therefore soil strengths and pore pressures) are unchanged: the only difference is in the strain predictions and this just involves calculating the extra strain component and adding it to those already determined.

5.3 THE INCREMENTAL STRESS–STRAIN RELATIONS

In order to perform non-linear finite element analysis using elasto-plastic models of soil behaviour, it is necessary to compute the modulus matrix $\mathbf{D}_{ep}$ relating an increment of strain to an increment of stress:

$$\Delta\sigma = \mathbf{D}_{ep}\,\Delta\epsilon. \tag{5.4}$$

Starting from the yield function $f(\sigma, \mathbf{h}) = 0$, and the plastic potential $g(\sigma, \mathbf{h}) = 0$, there is a piece of standard manipulation to obtain a formula for D_{ep} (e.g. Zienkiewicz, 1977):

$$\mathbf{D}_{ep} = \left[1 - \frac{\mathbf{D}_E\, aa^T}{a^T\mathbf{D}_E a - c^T Ha}\right] \mathbf{D}_E, \tag{5.5}$$

where $a = \partial g/\partial\sigma = \partial f/\partial\sigma$, $c = \partial f/\partial\mathbf{h}$ and H is a matrix relating changes in hardening parameters to changes in the incremental plastic strain: $\mathrm{d}\mathbf{h} = Hd\epsilon^p$.

We have used the symbol $\mathbf{D}_E$ above to emphasise that this refers to the elastic **D** matrix. The term '**D** matrix' has passed into common (finite element) usage in much the same way as has the term '**B** matrix', following the notation established by Zienkiewicz (1967, 1971, 1977). It is quite common to use the term '**D** matrix' to refer to different matrices (i.e. sometimes $\mathbf{D}_E$ and sometimes $\mathbf{D}_{ep}$). The reader must learn to spot which is intended by the context.

We now list the routines which calculate the terms of the **D** matrices in CRISP. Although our main intention is to demonstrate how (5.5) is implemented in CRISP, it is convenient to start with the two elastic models. Comparison of the elastic and elasto-plastic routines shows clearly the extra steps necessary for the latter.

In the rest of the book routines have been introduced to the reader in the same order as they are called in the program. The **D**-matrix routines are an exception to this, and so the reader may wish to pass over them quickly on a first reading. It is possible, however, to make use of these routines independently of the rest of the program. We explain why this might be appropriate in the last section of this chapter.

5.3.1 Routine DCON

Routine **DCON** calculates the **D** matrix for anisotropic elasticity. The anisotropic elastic properties relate strains to changes in stress via the following equations:

$$\begin{aligned}
\epsilon_x &= \frac{1}{E_h}\sigma_x - \frac{\nu_{vh}}{E_v}\sigma_y - \frac{\nu_{hh}}{E_h}\sigma_z,\\
\epsilon_y &= -\frac{\nu_{hv}}{E_h}\sigma_x + \frac{1}{E_v}\sigma_y - \frac{\nu_{hv}}{E_h}\sigma_z,\\
\epsilon_z &= -\frac{\nu_{hh}}{E_h}\sigma_x - \frac{\nu_{vh}}{E_v}\sigma_y + \frac{1}{E_h}\sigma_z,\\
\gamma_{xy} &= \frac{1}{G_{hv}}\tau_{xy}.
\end{aligned} \tag{5.6}$$

We have used suffixes 'h' (for horizontal) and 'v' (for vertical) to clarify how this model would be used in a geotechnical analysis. Section 9.2 contains a discussion of the significance of the various elastic parameters. The inverse form of (5.6) is inserted into the **D** matrix following Zienkiewicz (1977). The array PR contains the material properties as specified by the user in the data.

Routine DCON

```
      SUBROUTINE DCON(I,IET,NEL,NDIM,NS,NPR,NMT,MAT,PR,D,BK)            DCON   1
C*************************************************************************DCON   2
C     CALCULATES STRESS-STRAIN MATRIX FOR ANISOTROPIC ELASTICITY          DCON   3
C*************************************************************************DCON   4
      DIMENSION MAT(NEL),D(NS,NS),PR(NPR,NMT)                           DCON   5
C                                                                       DCON   6
      KM=MAT(I)                                                         DCON   7
      AN=PR(1,KM)/PR(2,KM)                                              DCON   8
      A=PR(2,KM)/((1.0+PR(3,KM))*(1.0-PR(3,KM)-2.0*AN*PR(4,KM)*         DCON   9
     1 PR(4,KM)))                                                       DCON  10
      D(1,1)=A*AN*(1.0-AN*PR(4,KM)*PR(4,KM))                            DCON  11
      D(1,2)=A*AN*PR(4,KM)*(1.0+PR(3,KM))                               DCON  12
      D(1,3)=A*AN*(PR(3,KM)+AN*PR(4,KM)*PR(4,KM))                       DCON  13
      D(2,1)=D(1,2)                                                     DCON  14
      D(2,2)=A*(1.0-PR(3,KM)*PR(3,KM))                                  DCON  15
```

```
      D(2,3)=D(1,2)                                                    DCON  16
      D(3,1)=D(1,3)                                                    DCON  17
      D(3,2)=D(2,3)                                                    DCON  18
      D(3,3)=D(1,1)                                                    DCON  19
      D(4,4)=PR(5,KM)                                                  DCON  20
      BK=(D(2,2)+2.*D(2,1))/3.                                         DCON  21
      IF(NDIM.EQ.2)GOTO 5                                              DCON  22
      D(5,5)=PR(5,KM)                                                  DCON  23
      D(6,6)=PR(5,KM)                                                  DCON  24
    5 IF(IET.EQ.0) GO TO 20                                            DCON  25
C                                                                      DCON  26
      DO 10 J=1,3                                                      DCON  27
      DO 10 JJ=1,3                                                     DCON  28
   10 D(JJ,J)=D(JJ,J)+PR(7,KM)*BK                                      DCON  29
   20 RETURN                                                           DCON  30
      END                                                              DCON  31
```

DCON 7 : material **zone** number.
DCON 8 : ratio $E_h/E_v = n$.
DCON 9–10 : $E_v'/[(1+\nu_h)(1-\nu_h-2n\nu_{vh}^2)]$.
DCON 11–21 : calculate components of elastic **D** matrix for 2-D.
DCON 23–24 : calculate additional components for 3-D.
DCON 27–29 : add K_w term $(\alpha K')$ for **drained/undrained** analysis during assembly of stiffness matrix (i.e. if IET $\neq 0$).

5.3.2 Routine DLIN

Routine **DLIN** calculates the **D** matrix when there is a linear variation of elastic properties with depth.

The elastic Young's modulus is given by the equation

$$E = E_0 + m(y_0 - y). \tag{5.7}$$

where

E_0 – Young's modulus at a depth y_0.

m – rate of increase in modulus with depth.

Routine DLIN

```
      SUBROUTINE DLIN(IP,I,IET,NEL,NDIM,NDN,NS,NPR,NMT,                DLIN   1
     1 ELCOD,SHFN,MAT,D,PR,INDX,BK)                                    DLIN   2
C*********************************************************************DLIN   3
C     CALCULATES STRESS-STRAIN MATRIX FOR LINEAR ELASTIC               DLIN   4
C     BEHAVIOUR WHEN ELASTIC PROPERTIES VARY LINEARLY WITH DEPTH       DLIN   5
C*********************************************************************DLIN   6
      DIMENSION ELCOD(NDIM,NDN),SHFN(NDN),D(NS,NS)                     DLIN   7
      DIMENSION MAT(NEL),PR(NPR,NMT)                                   DLIN   8
      COMMON /PARS  / PYI,ALAR,ASMVL,ZERO                              DLIN   9
C                                                                      DLIN  10
      KM=MAT(I)                                                        DLIN  11
CC    IPA=IP+INDX                                                      DLIN  12
      YY=ZERO                                                          DLIN  13
      DO 5 IN=1,NDN                                                    DLIN  14
    5 YY=YY+SHFN(IN)*ELCOD(2,IN)                                       DLIN  15
      E=PR(1,KM)+PR(3,KM)*(PR(2,KM)-YY)                                DLIN  16
      G=E/(2.*(1.+PR(4,KM)))                                           DLIN  17
      A=E/((1.+PR(4,KM))*(1.-2.*PR(4,KM)))                             DLIN  18
```



```
      BK=E/(3.*(1.-2.*PR(4,KM)))                                       DLIN  19
      D(1,1)=A*(1.-PR(4,KM))                                           DLIN  20
      D(1,2)=A*PR(4,KM)                                                DLIN  21
      D(1,3)=D(1,2)                                                    DLIN  22
      D(2,1)=D(1,2)                                                    DLIN  23
      D(2,2)=D(1,1)                                                    DLIN  24
      D(2,3)=D(1,3)                                                    DLIN  25
      D(3,1)=D(1,3)                                                    DLIN  26
      D(3,2)=D(2,3)                                                    DLIN  27
      D(3,3)=D(1,1)                                                    DLIN  28
      D(4,4)=G                                                         DLIN  29
      IF(NDIM.EQ.2)GOTO 8                                              DLIN  30
      D(5,5)=G                                                         DLIN  31
      D(6,6)=G                                                         DLIN  32
    8 IF(IET.EQ.0)RETURN                                               DLIN  33
      DO 10 J=1,3                                                      DLIN  34
      DO 10 JJ=1,3                                                     DLIN  35
   10 D(JJ,J)=D(JJ,J)+PR(7,KM)*BK                                      DLIN  36
      RETURN                                                           DLIN  37
      END                                                              DLIN  38
```

DLIN 11 : material **zone** number.
DLIN 13–15 : y co-ordinate (or z in axisymmetric problems) of integration point.
DLIN 16 : calculate value of Young's modulus at integration point.
DLIN 17 : calculate shear modulus.
DLIN 18–29 : calculate elastic **D** matrix for 2-D.
DLIN 31–32 : calculate additional components for 3-D.
DLIN 34–36 : add K_w term $(\alpha K')$ for **drained/undrained** analysis during assembly of stiffness matrix (i.e. if IET $\neq 0$).

5.3.3 Routine DCAM

Routine **DCAM** calculates the **D** matrix for Cam-clay. The array VARINT gives the values of VARiables at INTegration points. The first index of this array gives seven variables for two-dimensional analysis: σ_x', σ_y', σ_z', τ_{xy}, u, e (voids ratio) and p_c'. These variables will, in general, be varying over the whole finite element mesh.

Routine DCAM

```
      SUBROUTINE DCAM(IP,I,IET,NEL,NIP,NVRS,NDIM,NS,NPR,NMT,           DCAM   1
     1 VARINT,MAT,D,PR,ITP,BK)                                         DCAM   2
C*********************************************************************DCAM   3
C     CALCULATES  STRESS-STRAIN  MATRIX  FOR  CAM-CLAY                 DCAM   4
C*********************************************************************DCAM   5
      DIMENSION VARINT(NVRS,NIP,NEL),D(NS,NS),MAT(NEL)                 DCAM   6
      DIMENSION S(6),A(6),B(6),PR(NPR,NMT)                             DCAM   7
C                                                                      DCAM   8
      KM=MAT(I)                                                        DCAM   9
      SX=VARINT(1,IP,I)                                                DCAM  10
      SY=VARINT(2,IP,I)                                                DCAM  11
      SZ=VARINT(3,IP,I)                                                DCAM  12
      TXY=VARINT(4,IP,I)                                               DCAM  13
      E=VARINT(NS+2,IP,I)                                              DCAM  14
      PC=ABS(VARINT(NS+3,IP,I))                                        DCAM  15
      P=(SX+SY+SZ)/3.                                                  DCAM  16
      Q2=SX*(SX-SY)+SY*(SY-SZ)+SZ*(SZ-SX)+3.*TXY*TXY                   DCAM  17
```

```
      IF(NDIM.EQ.2)GOTO 10                                              DCAM  18
C                                                                       DCAM  19
      TYZ=VARINT(5,IP,I)                                                DCAM  20
      TZX=VARINT(6,IP,I)                                                DCAM  21
      Q2=Q2+3.*TYZ*TYZ+3.*TZX*TZX                                       DCAM  22
   10 Q=SQRT(Q2)                                                        DCAM  23
      PY=P*EXP(Q/(PR(4,KM)*P))                                          DCAM  24
      BK=(1.+E)*P/PR(1,KM)                                              DCAM  25
C-----------------------------------------------------------------------DCAM  26
C     CALCULATE  ELASTIC  STRESS-STRAIN  MATRIX                         DCAM  27
C-----------------------------------------------------------------------DCAM  28
      G=PR(5,KM)                                                        DCAM  29
      IF(G.LT.1.) G=BK*1.5*(1.-2.*PR(5,KM))/(1.+PR(5,KM))               DCAM  30
      AL=(3.*BK+4.*G)/3.                                                DCAM  31
      DL=(3.*BK-2.*G)/3.                                                DCAM  32
C                                                                       DCAM  33
      CALL ZEROR2(D,NS,NS)                                              DCAM  34
      D(1,1)=AL                                                         DCAM  35
      D(2,1)=DL                                                         DCAM  36
      D(3,1)=DL                                                         DCAM  37
      D(1,2)=DL                                                         DCAM  38
      D(2,2)=AL                                                         DCAM  39
      D(3,2)=DL                                                         DCAM  40
      D(1,3)=DL                                                         DCAM  41
      D(2,3)=DL                                                         DCAM  42
      D(3,3)=AL                                                         DCAM  43
      D(4,4)=G                                                          DCAM  44
      IF(NDIM.EQ.2)GOTO 11                                              DCAM  45
      D(5,5)=G                                                          DCAM  46
      D(6,6)=G                                                          DCAM  47
C                                                                       DCAM  48
   11 IF(PY.LT.0.99*PC) GO TO 50                                        DCAM  49
C-----------------------------------------------------------------------DCAM  50
C     CALCULATE  PLASTIC  STRESS-STRAIN  MATRIX  IF  CURRENT            DCAM  51
C     POINT  ON  YIELD  LOCUS  AND  SET  PC  NEGATIVE                   DCAM  52
C-----------------------------------------------------------------------DCAM  53
      VARINT(NS+3,IP,I)=-ABS(VARINT(NS+3,IP,I))                         DCAM  54
      S(1)=SX-P                                                         DCAM  55
      S(2)=SY-P                                                         DCAM  56
      S(3)=SZ-P                                                         DCAM  57
      S(4)=2.*TXY                                                       DCAM  58
      IF(NDIM.EQ.2)GOTO 12                                              DCAM  59
      S(5)=2.*TYZ                                                       DCAM  60
      S(6)=2.*TZX                                                       DCAM  61
   12 BB=(1.-Q/(PR(4,KM)*P))/(3.*P)                                     DCAM  62
      ITP=0                                                             DCAM  63
      IF(Q.LT.1.0E-5) GOTO 15                                           DCAM  64
      QMP=Q/(PR(4,KM)*P)                                                DCAM  65
      IF(QMP.LT.0.01) GOTO 14                                           DCAM  66
      C=1.5/(Q*PR(4,KM)*P)                                              DCAM  67
      GOTO 16                                                           DCAM  68
C-----------------------------------------------------------------------DCAM  69
C     Q/MP IS SMALL.USE FITTED CURVE TO CALCULATE C VALUE               DCAM  70
C-----------------------------------------------------------------------DCAM  71
   14 CA=153.0302/(PR(4,KM)**2*PC**2)                                   DCAM  72
      C=(-2.98*(100.*QMP)**3+3.98*(100.*QMP)**2)*CA                     DCAM  73
      ITP=1                                                             DCAM  74
      GOTO 16                                                           DCAM  75
C-----------------------------------------------------------------------DCAM  76
C     Q/MP IS TOO SMALL.USE C VALUE FOR ZERO Q/MP                       DCAM  77
C-----------------------------------------------------------------------DCAM  78
   15 C=0.                                                              DCAM  79
      ITP=1                                                             DCAM  80
   16 A(1)=BB+C*S(1)                                                    DCAM  81
      A(2)=BB+C*S(2)                                                    DCAM  82
      A(3)=BB+C*S(3)                                                    DCAM  83
```



```
      A(4)=C*S(4)                                                       DCAM  84
      IF(NDIM.EQ.2)GOTO 18                                              DCAM  85
      A(5)=C*S(5)                                                       DCAM  86
      A(6)=C*S(6)                                                       DCAM  87
C                                                                       DCAM  88
   18 DO 20 J=1,3                                                       DCAM  89
      B(J)=0.                                                           DCAM  90
      DO 20 JJ=1,3                                                      DCAM  91
   20 B(J)=B(J)+D(J,JJ)*A(JJ)                                           DCAM  92
      B(4)=D(4,4)*A(4)                                                  DCAM  93
      IF(NDIM.EQ.2)GOTO 25                                              DCAM  94
      B(5)=D(5,5)*A(5)                                                  DCAM  95
      B(6)=D(6,6)*A(6)                                                  DCAM  96
C                                                                       DCAM  97
   25 XI=(PR(2,KM)-PR(1,KM))/(1.+E)                                     DCAM  9
      AA=3.*BB/XI                                                       DCAM  99
      AB=0.                                                             DCAM 100
C                                                                       DCAM 101
      DO 30 J=1,NS                                                      DCAM 102
   30 AB=AB+A(J)*B(J)                                                   DCAM 103
      BETA=AA+AB                                                        DCAM 104
      DO 40 J=1,NS                                                      DCAM 105
      DO 40 JJ=1,NS                                                     DCAM 106
   40 D(JJ,J)=D(JJ,J)-B(JJ)*B(J)/BETA                                   DCAM 107
   50 IF(IET.EQ.0) GOTO 80                                              DCAM 108
C                                                                       DCAM 109
      DO 60 J=1,3                                                       DCAM 110
      DO 60 JJ=1,3                                                      DCAM 111
   60 D(JJ,J)=D(JJ,J)+PR(7,KM)*BK                                       DCAM 112
   80 RETURN                                                            DCAM 113
      END                                                               DCAM 114
```

DCAM 9 : material **zone** number.
DCAM 10–13 : effective stress components for 2-D.
DCAM 14 : voids ratio (e).
DCAM 15 : size of current yield locus (p_c').
DCAM 16 : mean normal effective stress (p').
DCAM 17 : q^2.
DCAM 20–21 : additional shear stress components (3-D).
DCAM 22 : q^2 for 3-D.
DCAM 23 : q.
DCAM 24 : size of yield locus passing through stress state (not the same as current yield locus).
DCAM 25 : calculate bulk modulus of soil.
DCAM 29 : shear modulus (or Poisson's ratio if < 1).
DCAM 30 : calculate shear modulus G.
DCAM 31–32 : elastic constants.
DCAM 34 : zero **D** matrix.
DCAM 35–44 : elastic **D** matrix (2-D).
DCAM 46–47 : additional components of elastic **D** matrix for 3-D.
DCAM 49 : skip if elastic.
DCAM 54 : make p_c' negative to indicate yielding.
DCAM 55–58 : calculate deviatoric stresses for 2-D.
DCAM 60–61 : additional components for 3-D.
DCAM 62 : calculate constant part of flow matrix a.

DCAM 63–66 : check if stress state is close to tip along p' axis.
DCAM 67–68 : if not, skip after calculating C.
DCAM 72–75 : calculate C using curve fitting if close to tip.
DCAM 81–84 : calculate flow matrix a for 2-D.
DCAM 86–87 : calculate additional components of flow matrix a for 3-D.
DCAM 89–93 : calculate $b = \mathbf{D} \cdot a$ for 2-D.
DCAM 95–96 : calculate $b = \mathbf{D} \cdot a$ for 3-D.
DCAM 98–99 : calculate hardening parameter $c^{\mathrm{T}}Ha$.
DCAM 102–104 : calculate $a^{\mathrm{T}}\mathbf{D}_{\mathrm{E}}a - c^{\mathrm{T}}Ha$.
DCAM 105–107 : calculate $\mathbf{D}_{\mathrm{ep}}$ matrix.
DCAM 110–112 : add K_{w} term for **drained/undrained** analysis during assembly of element stiffness matrix (i.e. only if IET $\neq$ 0).

5.3.4 Routine DMCAM

Routine **DMCAM** calculates the **D** matrix for modified Cam-clay.

Routine DMCAM

```
      SUBROUTINE DMCAM(IP,I,IET,NEL,NIP,NVRS,NDIM,NS,NPR,NMT,          DMCM   1
     1 VARINT,MAT,D,PR,BK)                                             DMCM   2
C*************************************************************************DMCM   3
C     CALCULATES  STRESS-STRAIN  MATRIX  FOR  MODIFIED CAM-CLAY         DMCM   4
C*************************************************************************DMCM   5
      DIMENSION VARINT(NVRS,NIP,NEL),D(NS,NS),MAT(NEL)                  DMCM   6
      DIMENSION S(6),A(6),B(6),PR(NPR,NMT)                              DMCM   7
                                                                        DMCM   8
C                                                                       DMCM   9
      KM=MAT(I)                                                         DMCM  10
      SX=VARINT(1,IP,I)                                                 DMCM  11
      SY=VARINT(2,IP,I)                                                 DMCM  12
      SZ=VARINT(3,IP,I)                                                 DMCM  13
      TXY=VARINT(4,IP,I)                                                DMCM  14
      E=VARINT(NS+2,IP,I)                                               DMCM  15
      PC=ABS(VARINT(NS+3,IP,I))                                         DMCM  16
      P=(SX+SY+SZ)/3.                                                   DMCM  17
      Q2=SX*(SX-SY)+SY*(SY-SZ)+SZ*(SZ-SX)+3.*TXY*TXY                    DMCM  18
      IF(NDIM.EQ.2)GOTO 10                                              DMCM  19
C                                                                       DMCM  20
      TYZ=VARINT(5,IP,I)                                                DMCM  21
      TZX=VARINT(6,IP,I)                                                DMCM  22
      Q2=Q2+3.*TYZ*TYZ+3.*TZX*TZX                                       DMCM  23
   10 Q=SQRT(Q2)                                                        DMCM  24
      PY=P+Q*Q/(P*PR(4,KM)*PR(4,KM))                                    DMCM  25
      BK=(1.+E)*P/PR(1,KM)                                              DMCM  26
C-------------------------------------------------------------------------DMCM  27
C     CALCULATE  ELASTIC  STRESS-STRAIN  MATRIX                         DMCM  28
C-------------------------------------------------------------------------DMCM  29
      G=PR(5,KM)                                                        DMCM  30
      IF(G.LT.1.) G=BK*1.5*(1.-2.*PR(5,KM))/(1.+PR(5,KM))               DMCM  31
      AL=(3.*BK+4.*G)/3.                                                DMCM  32
      DL=(3.*BK-2.*G)/3.                                                DMCM  33
C                                                                       DMCM  34
      CALL ZEROR2(D,NS,NS)                                              DMCM  35
      D(1,1)=AL                                                         DMCM  36
      D(2,1)=DL                                                         DMCM  37
      D(3,1)=DL                                                         DMCM  38
      D(1,2)=DL                                                         DMCM  39
      D(2,2)=AL                                                         DMCM  40
      D(3,2)=DL                                                         DMCM  41
      D(1,3)=DL                                                         DMCM  42
      D(2,3)=DL                                                         DMCM  43
      D(3,3)=AL                                                         DMCM  44
      D(4,4)=G                                                          DMCM  45
      IF(NDIM.EQ.2)GOTO 12                                              DMCM  46
      D(5,5)=G                                                          DMCM  47
      D(6,6)=G                                                          DMCM  48
C                                                                       DMCM  49
   12 IF(PY.LT.0.99*PC) GO TO 50                                        DMCM  50
C-------------------------------------------------------------------------DMCM  51
C     CALCULATE  PLASTIC  STRESS-STRAIN  MATRIX  IF  CURRENT            DMCM  52
C     POINT  ON  YIELD  LOCUS  AND  SET  PC  NEGATIVE                   DMCM  53
C-------------------------------------------------------------------------DMCM  54
      VARINT(NS+3,IP,I)=-ABS(VARINT(NS+3,IP,I))                         DMCM  55
      PCS=.5*PC                                                         DMCM  56
      PB=P/PCS                                                          DMCM  57
      S(1)=SX-P                                                         DMCM  58
      S(2)=SY-P                                                         DMCM  59
      S(3)=SZ-P                                                         DMCM  60
      S(4)=2.*TXY                                                       DMCM  61
      IF(NDIM.EQ.2)GOTO 16                                              DMCM  62
      S(5)=2.*TYZ                                                       DMCM  63
      S(6)=2.*TZX                                                       DMCM  64
   16 BB=-2.*(1.-PB)/(3.*PCS)                                           DMCM  65
      C=3./(PCS*PCS*PR(4,KM)*PR(4,KM))                                  DMCM  66
      A(1)=BB+C*S(1)                                                    DMCM  67
      A(2)=BB+C*S(2)                                                    DMCM  68
      A(3)=BB+C*S(3)                                                    DMCM  69
      A(4)=C*S(4)                                                       DMCM  70
      IF(NDIM.EQ.2)GOTO 18                                              DMCM  71
      A(5)=C*S(5)                                                       DMCM  72
      A(6)=C*S(6)                                                       DMCM  73
C                                                                       DMCM  74
   18 DO 20 J=1,3                                                       DMCM  75
      B(J)=0.                                                           DMCM  76
      DO 20 JJ=1,3                                                      DMCM  77
   20 B(J)=B(J)+D(J,JJ)*A(JJ)                                           DMCM  78
      B(4)=D(4,4)*A(4)                                                  DMCM  79
      IF(NDIM.EQ.2)GOTO 25                                              DMCM  80
      B(5)=D(5,5)*A(5)                                                  DMCM  81
      B(6)=D(6,6)*A(6)                                                  DMCM  82
C                                                                       DMCM  83
   25 XI=(PR(2,KM)-PR(1,KM))/(1.+E)                                     DMCM  84
      AA=-4.*PB*(1.-PB)/(PCS*XI)                                        DMCM  85
      AB=0.                                                             DMCM  86
C                                                                       DMCM  87
      DO 30 J=1,NS                                                      DMCM  88
   30 AB=AB+A(J)*B(J)                                                   DMCM  89
      BETA=AA+AB                                                        DMCM  90
      DO 40 J=1,NS                                                      DMCM  91
      DO 40 JJ=1,NS                                                     DMCM  92
   40 D(JJ,J)=D(JJ,J)-B(JJ)*B(J)/BETA                                   DMCM  93
   50 IF(IET.EQ.0) GOTO 80                                              DMCM  94
C                                                                       DMCM  95
      DO 60 J=1,3                                                       DMCM  96
      DO 60 JJ=1,3                                                      DMCM  97
   60 D(JJ,J)=D(JJ,J)+PR(7,KM)*BK                                       DMCM  98
   80 CONTINUE                                                          DMCM  99
CC    WRITE(6,801)I,IP,D                                                DMCM 100
CC801 FORMAT(/1X,4HI = ,I5,2X,5HIP = ,I5,3X,1HD/(1X,9E14.5))            DMCM 101
      RETURN                                                            DMCM 102
      END
```

DMCM 9 : material **zone** number.
DMCM 10–13 : effective stress components for 2-D.

DMCM 14 : voids ratio (e).
DMCM 15 : size of yield locus (p_c').
DMCM 16 : mean normal effective stress (p').
DMCM 17 : q^2.
DMCM 20–21 : additional shear stress components (3-D).
DMCM 22 : q^2 for 3-D.
DMCM 23 : q.
DMCM 24 : size of yield locus passing through stress state (not the same as current yield locus).
DMCM 25 : calculate bulk modulus of soil.
DMCM 29 : shear modulus (or Poisson's ratio if < 1).
DMCM 30 : calculate shear modulus G.
DMCM 31–32 : elastic constants.
DMCM 34 : zero D matrix.
DMCM 35–44 : elastic D matrix (2-D).
DMCM 46–47 : additional components of elastic D matrix for 3-D.
DMCM 49 : skip if elastic.
DMCM 54 : make p_c' negative to indicate yielding.
DMCM 57–60 : calculate deviatoric stresses for 2-D.
DMCM 62–63 : additional components for 3-D.
DMCM 64 : calculate constant part of flow matrix a.
DMCM 65 : calculate C.
DMCM 66–69 : calculate flow matrix a for 2-D.
DMCM 71–72 : calculate additional components of flow matrix a for 3-D.
DMCM 74–78 : calculate $b = D \,.\, a$ for 2-D.
DMCM 80–81 : calculate $b = D \,.\, a$ for 3-D.
DMCM 83–84 : calculate hardening parameter $c^{\mathrm{T}}Ha$.
DMCM 87–88 : calculate $a^{\mathrm{T}}\mathbf{D}_{\mathrm{E}}a - c^{\mathrm{T}}Ha$.
DMCM 90–92 : calculate D_{ep} matrix.
DMCM 95–97 : add K_{w} term for **drained/undrained** analysis during assembly of element stiffness matrix (i.e. only if IET $\neq 0$).

5.4 DETERMINING THE CAM-CLAY PARAMETERS

5.4.1 Introduction

The critical state soil parameters can all be determined from the normal range of laboratory tests that are performed on a soil. The approach to the selection of parameters will depend on the problem to which the program is to be applied. In general the information should be obtained from high-quality laboratory tests. This is particularly so when the program is to be used to predict behaviour in a field situation. In these circumstances, advanced *in situ* testing techniques (e.g. Wroth, 1984) are desirable in addition to high-quality laboratory tests on 'undisturbed samples'.

Of course, sometimes high-quality data will not be available, and the analyst must develop a feel for the range of possible parameter values and the influence

of the variation of each. In practical and research applications it is quite common to perform 'parametric studies', where one performs analyses with different parameter values to study the influence of each.

Some soil tests give information which is not independently specified within the critical state framework (but depends on other CSSM parameters and the *in situ* stresses). One example is the undrained shear strength. In these circumstances there will usually be some discrepancy between data from different sources. Some of this will be due to the quality of the data, and some will be due to the fact that despite their sophistication, the critical state models are simplified idealisations of real soil behaviour. The analyst needs to obtain a 'best fit' between all the available data and the critical state parameters, bearing in mind the reliability of each piece of data. Indeed one of the strengths of the critical state theories is this ability to review data from different types of soil test (Wroth, 1984).

5.4.2 The frictional constant M

Triaxial tests (drained and undrained with pore pressure measurement) on isotropically consolidated samples can be used to obtain the frictional constant M. A number of tests need to be carried out with different consolidation pressures. It is necessary to continue these tests to large strains to ensure that the samples are close to the critical state. For the undrained tests the pore pressures should be monitored to see that they are not still changing at the end of the test. If they are, then the samples have not reached the critical state and these results would lead to M being underestimated.

If one obtains the principal effective stresses at failure, then the drained angle of friction ϕ' can be obtained from the geometry of a Mohr's circle plot: $\sigma_a'/\sigma_r' = (1 + \sin\phi')/(1 - \sin\phi')$. Combining this relation with the definitions of p' and q, M (the value of q/p' at failure) is given by

$$\mathrm{M} = \frac{6 \sin\phi'}{3 - \sin\phi'} \,. \tag{5.8}$$

Of course, it is not necessary to go through the intermediate step of calculating ϕ': we have introduced this to make the relationship of M and ϕ' explicit. Alternatively, by plotting the q/p' values at failure, the slope of the best-fitting straight line is taken as M. If one is testing field samples, a fair amount of scatter is to be expected and some 'engineering judgement' is needed here.

As noted in section 5.2.1, the influence of the intermediate principal stress on the soil strength is usually better described by the Mohr–Coulomb equation than by the critical state cone. Sometimes the value of M is adjusted slightly to take this into account (e.g. a lower value is chosen which will match the soil strength in plane strain better when used in the finite element analysis).

5.4.3 Slopes of the normal consolidation and swelling lines (λ and κ)

These parameters can be obtained from oedometer tests or from triaxial tests on

samples either isotropically or with K_0 normally consolidated. From the theoretical point of view, one expects to obtain equal values of λ from any constant η compression test. Thus one would expect to get the same value of λ from an isotropic compression test and a K_0 compression test. Because the value of K_0 changes on one-dimensional unloading (see section 5.5), an oedometer capable of horizontal stress measurement is required it κ is to be determined from one-dimensional unloading rather than isotropic unloading.

It is standard practice to plot the results of one-dimensional compression tests in terms of e (voids ratio) against $\log_{10}\sigma'_v$, where σ'_v is the effective vertical stress. The slope C_c of the normally consolidated line is known as the 'compression index'.

$$\lambda = C_c/2.303. \tag{5.9}$$

($2.303 = \ln(10.)$ Alternatively λ can be directly determined from the slope of the compression line in a $(\ln(p'), e)$ plot. Often κ is simply estimated from λ, as indicated at the end of this section.

One sometimes finds that the compression line in $(\ln(p'), e)$ space is curved rather than linear. Under these circumstances one has to choose the slope appropriate to the stress level believed to be relevant in the problem to be analysed. (Note that this will also affect the estimation of Γ or e_{cs} discussed below.)

It is interesting to note that Butterfield (1979) re-plotted the results discussed above in $\ln(V)$–$\ln(p')$ space and obtained linear plots. In fact it is difficult to decide on the basis of the available data whether Butterfield's proposal or the traditional approach is better. From a theoretical point of view, linear relations in $(\ln(e), \ln(p'))$ plots would be preferable, eliminating the possibility of negative values of e at high stress levels. This would tidy up one corner of critical state theory, but for practical purposes the traditional relations appear to be quite satisfactory.

In fact one can re-formulate the critical state models to incorporate Butterfield's suggestion (or any other hardening law). This would involve some changes to the finite element program (but not major ones).

κ-lines are usually found to be even more curved than λ-lines. In Chapter 2 we pointed out that although the assumed form of elasticity is adequate for many purposes, there are situations (e.g. cyclic loading) where the κ-line assumption is not adequate. κ values are often chosen in the range of one-fifth to one-third of λ. The data usually indicate a lower (stiffer) value on immediate unloading and a higher value at later stages of unloading.

5.4.4 Location of CSL in $(e, \ln(p'))$ plot ($e_{cs} = \Gamma - 1$)

e_{cs} is defined as the voids ratio on the critical state line for a value of $p' = 1$.

Note that the parameter describing the location of the CSL in Chapter 2 (Γ) was a specific volume, whereas the parameter required here (e_{cs}) is a voids ratio. Since specific volumes can always be converted into voids ratios (and vice versa) using the relation $V = 1 + e$, this should not lead to any confusion.

Following on from the determination of M above, the reader might expect that e_{cs} would be determined by measuring the moisture contents of several triaxial tests at failure. This is rarely done, however, basically because of the difficulty in obtaining sufficiently accurate data. In fact once λ and κ have been determined, a value of moisture content at any point on the stable state boundary surface will suffice to fix a value of Γ, using either (2.17) for Cam-clay or (2.41) for modified Cam-clay. It is common in fact to determine e_{cs} in this way from consolidation data. A side-effect of this procedure is that different values of e_{cs} (or Γ) are obtained for the Cam-clay and modified Cam-clay models. This is in contrast to the conventional assessment of the differences between Cam-clay and modified Cam-clay when it is assumed that the critical states coincide for the two models.

Fig. 5.3 shows the normal assumption which is made: the CSSM parameters M, λ, κ and Γ (or e_{cs}) are assumed to be identical for Cam-clay and modified Cam-clay. In this case the difference between the two models shows up as different isotropic normal consolidation lines.

Fig. 5.4 shows the result of following the procedure outlined above. Here the value of Γ has been obtained from a moisture content (i.e. value of e or V) on the isotropic normal consolidation line. This gives different values of Γ for Cam-clay and modified Cam-clay and thus two different positions of the critical state

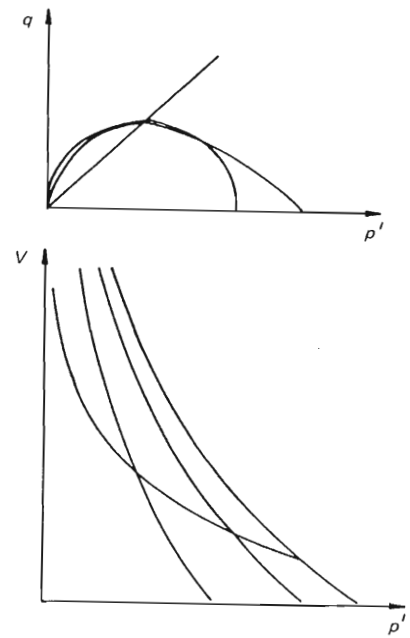


Fig. 5.3 – When comparing Cam-clay and modified Cam-clay it is conventionally assumed that the models coincide at the critical state. Hence the isotropic normal consolidation lines are different

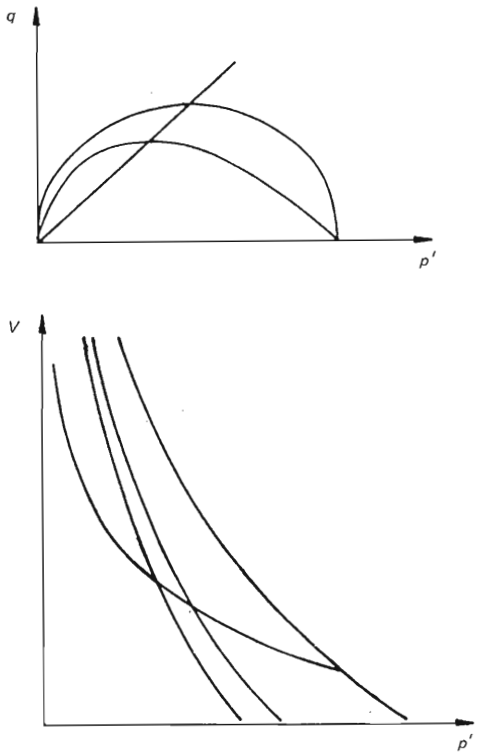


Fig. 5.4 – If the critical state parameter Γ is calculated from the moisture content of an isotropically normally consolidated sample, then Cam-clay and modified Cam-clay have different CSLs in the (p', V) plot. Hence modified Cam-clay gives higher undrained shear strengths than Cam-clay

line in the ($\ln(p')$, V) plot. One practical consequence of this approach is that the undrained shear strength of a soil (with the same moisture content) is now 28% greater for modified Cam-clay compared to Cam-clay. It was this fact that we were referring to in Chapter 2 when we commented that the difference between Cam-clay and modified Cam-clay is often greater than is sometimes suggested. The figure of 28% here is based on soil parameters with $\lambda = 5\kappa$: the ratio of shear strengths is obtained by substituting into (2.26) the two different values of Γ. For other soil parameters, the ratio can be calculated as 1.36 raised to the power Λ, where 1.36 is half the base of natural logarithms and $\Lambda = 1 - \kappa/\lambda$, as in Chapter 2.

If the values of Γ are obtained from moisture contents from an oedometer test then neither the CSL nor the isotropic NCL will coincide for Cam-clay and modified Cam-clay. In this case the discrepancy between predictions of undrained shear strength will remain, but will not be so large as above.

5.4.5 ν' or G

As indicated above, CRISP allows the user to specify either a constant value of ν' or a constant value of G. Now K' varies with p' (as indicated by (5.3)), and it can be shown that if G is also allowed to vary with p', then the soil is not truly elastic. This is because elastic stress cycles are not necessarily reversible (Zytynski *et al.*, 1978). Thus it would appear to be preferable from a theoretical point of view to assume a constant value of G. The question is: what value of G? Experimental evidence indicates that G does vary with stress level. Attempts to correlate G with other data suggest a stronger relation with p', than p'_c or c_u.

It is, therefore, usually more convenient to specify a value of ν' which means that G varies in the same way as K'. This is particularly so when analysing a problem where there is a significant variation in stress level in the soil. The question now is: what value of ν'? There are two ways of arriving at a value of ν'. The first is from data of K_0 versus OCR, and the second from strain measurements in triaxial tests. The first is the more usual, and gives a value of about 0.3 for many soils: this is related to the consideration of *in situ* stresses discussed below. The second method tends to give lower values of ν' (e.g. 0.12 for London clay (Wroth, 1972)). At first, this kind of discrepancy may seem to throw doubt on whether it is possible to assign realistic elastic parameters. Techniques for accurate measurement of strains recently developed at Imperial College reveal a more complex non-linear behaviour in this 'elastic' region of behaviour (Jardine *et al.*, 1984).

It is worth pointing out here that the main strength of the Cam-clay models is in the calculation of plastic strains during yielding, as opposed to the elastic strains which are calculated for over-consolidated behaviour. Thus for many problems the exact assumption made for elastic properties is of only secondary importance. On the other hand, there will certainly be some problems where the assumptions made here are deficient, and the user should consider incorporating some new material idealisation within the yield locus in the program. As we have indicated in Chapter 2, this is an area of continuing research.

5.4.6 Horizontal and vertical permeabilities

Although permeabilities are not 'Cam-clay parameters', they are considered here for completeness. For layered soils it is well known that the horizontal permeability is greater than the vertical permeability. The same is true for any samples anisotropically (for example K_0) consolidated. In the laboratory, the permeability can be determined from oedometer tests. To determine the horizontal permeability, a specially modified oedometer with radial drainage is required. Oedometers with external radial drainage may be preferable. The measuring of permeabilities either in the field or in the laboratory is well documented and will not be discussed here. The vertical permeability can also be estimated from the coefficient of consolidation (c_v) and the coefficient of compressibility (m_v) from the expression

$$k_v = c_v m_v \gamma_w. \tag{5.10}$$

(In terms of the CSSM parameters, $m_v = \lambda/(p'V)$, from differentiating the λ-line equation, but since the value of the horizontal stress is not necessarily known, direct use of (5.10) is more convenient.)

In CRISP the permeability is assumed to be constant throughout the analysis. Experimental evidence shows that permeability varies with stress level. As the voids ratio increases, the pore water can flow more easily, and it is realistic to expect the permeability to increase with increase in voids ratio. Such a relationship can be readily incorporated into the program if sufficient data to support this are available for the particular soil being modelled (Almeida, 1984).

5.5 *IN SITU* STRESSES

5.5.1 Introduction

In section 5.4, various means of obtaining the Cam-clay parameters (i.e. soil constants) were described. In this section we discuss how to determine the stress parameters, which vary from point to point in the soil. These are the *in situ* distribution of σ_v', σ_h', u_0 and p_c' for the entire region of the analysis. The parameter p_c' is only needed for those zones of the mesh where the Cam-clay models are used. CRISP uses this information to calculate the initial values of voids ratio (e) over those zones.

The reason that these *in situ* stresses are required is that in an elasto-plastic analysis the stiffness matrix of a finite element will be dependent on the stress state within the element. In general the stress state will vary across an element, and the stiffness terms are calculated by integrating expressions dependent on these varying stresses over the volume of each element. CRISP integrates these expressions numerically by 'sampling' the stresses at particular points within the element and then using standard numerical integration rules for *triangular* areas.

For Cam-clays it is important to try to establish the *in situ* stress state as accurately as possible. This is because the displacements predicted by an analysis are quite sensitive to the relative amounts of elastic (over-consolidated)/plastic straining that take place.

5.5.2 How *in situ* stresses are set up

The *in situ* stresses in the ground are produced by the loadings which the geological history of a site imposes on each small element of soil. Many natural soils are deposited as mineral particles from water or the atmosphere. As a deposit of soil is progressively built up in a series of layers, each small element of soil is subjected to a steadily increasing vertical effective stress. Soil in this condition is normally consolidated, because each element has never been subjected to a greater stress. The erosion of upper layers of the soil will lead to unloading of the remaining soil, which therefore becomes over-consolidated. An alternative reason for over-consolidation is the raising of the water table (Parry, 1970). The water table may fall again, or new layers of soil may be deposited, and so an element of soil may go through several cycles of loading, unloading and reloading. In all cases the assumption is made that the soil loading and unloading is one dimensional, i.e. no shear stresses develop on vertical or horizontal planes. In other words, the principal stress directions are vertical and horizontal, and the horizontal stresses are equal.

There are some situations where this description will not be appropriate. The recent engineering history of a site (e.g. excavation, compaction or construction) will also affect the *in situ* stresses of soil elements near to the engineering activity. Residual soils which are formed by the *in situ* weathering of rocks do not conform to this picture: they invariably behave as if over-consolidated.

The calculation of the vertical effective stress is straightforward. The vertical total stress at any depth is calculated as the bulk density of the soil multiplied by the depth. (A more sophisticated approach is to take into account the variation of bulk density with depth, but this is usually not necessary.) From the position of the water table, the pore water pressure is calculated, and hence the vertical effective stress ($\sigma_v' = \sigma_v - u$). The calculation of the horizontal effective stress is not so straightforward. The coefficient of earth pressure at rest ($K_0 = \sigma_h'/\sigma_v'$) depends on the stress history of the soil. We describe below some methods of estimating K_0, but it is worth pointing out at the start that from both the practical and the theoretical points of view, these methods are not entirely satisfactory. A prudent engineer would supplement these estimates by *in situ* measurements of the horizontal effective stress using, for example, the self-boring pressure meter (Wroth, 1984).

5.5.3 Two approaches for *in situ* stresses

In an elastic analysis of soil (and sometimes in an elastic–perfectly-plastic analysis) it is quite common to set K_0 as $\nu'/(1-\nu')$. This is consistent with the condition of zero lateral strain inherent in one-dimensional elastic compression, but unfortunately measured laboratory values of ν' are not consistent with the usual values of K_0 believed appropriate for the field.

Of course this elastic assumption is *not* to be used for analyses using the critical state models, where one-dimensional compression involves plastic yielding. When using the Cam-clay models there are basically two approaches for determining the *in situ* stresses:

(i) an analysis is performed (either using CRISP or by hand) in which a soil column is subjected to the stress history which is believed has been applied to the soil deposit in practice. This approach has the merit of being theoretically consistent with subsequent analysis but it suffers from the disadvantage that Cam-clay and, to a lesser extent, modified Cam-clay are not very successful in predicting values of K_{nc} (the coefficient of earth pressure at rest for normally consolidated soil):

(ii) a rather more empirical method is used, based on the data accumulated by Wroth (1975).

We concentrate on the second approach, which is rather more practical and easy to use. However, in the final section we briefly compare Wroth's method with the so-called 'consistent' approach.

5.5.4 Wroth's method

In Wroth's method the value of K_{nc} is taken as

$$K_{nc} = 1 - \sin \phi', \tag{5.11}$$

a simplified version of Jaky's relation (1944). Although there is some theoretical analysis underlying this equation, examination of Jaky's paper reveals that the relation is deduced for the stress state at the centre of an embankment, where there is no necessity for there to be a condition of zero lateral strain. Hence (5.11) must be regarded as an empirical relation. However, there is evidence that (5.11) gives K_{nc} values which match data from laboratory tests (Wroth, 1972).

Wroth (1975) proposes two alternative relationships between K_0, K_{nc} and OCR (OCR $= \sigma'_{vm}/\sigma'_v$ where σ'_{vm} is the maximum vertical effective stress experienced by a soil element):

$$K_0 = \text{OCR}\, K_{nc} - \frac{\nu'}{1-\nu'}(\text{OCR} - 1), \tag{5.12}$$

and

$$m\left[\frac{3(1-K_{nc})}{1+2K_{nc}} - \frac{3(1-K_0)}{1+2K_0}\right] = \ln\left[\frac{\text{OCR}(1+2K_{nc})}{1+2K_0}\right]. \tag{5.13}$$

(5.12) is obtained by considering elastic unloading from the normally consolidated state, and gives a good fit to the existing data for a number of soils up to an OCR of about 5. The values of ν' necessary to fit the observed data were determined by Wroth to be in the range 0.254 to 0.371 for eight different soils. (5.13) was proposed as valid up to higher values of OCR and was obtained from the observation that an unloading plot of q/p' versus $\ln p'$ is a straight-line relationship (Fig. 5.5). m is an empirical constant which Wroth shows is linearly related to the Plasticity Index (PI) for a number of soils. Wroth (1976) suggests the following equation for estimating m (where direct measurements are not available):

$$m = 0.022875\ \text{PI} + 1.22, \tag{5.14}$$

where PI is in per cent.

Wroth's method requires a knowledge of the OCR for soil at each depth. The standard procedure for obtaining the value of OCR is to test samples of clay in an oedometer and carry out one-dimensional consolidation with small load increments (Bjerrum, 1973), using the method of Casagrande (1936) to determine the vertical pre-consolidation pressure. Samples taken at frequent intervals of depth should give the variation of σ'_{vm} with depth. Hence the over-consolidation ratio (OCR) with depth can be determined.

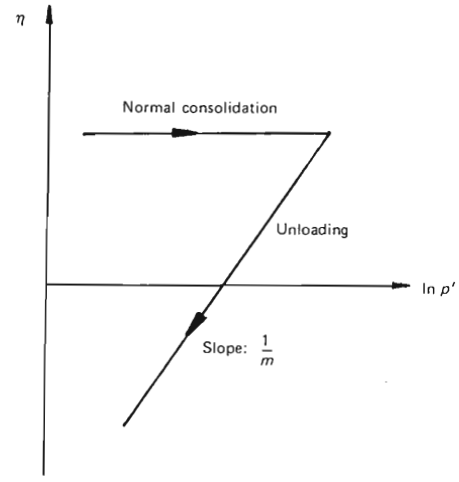


Fig. 5.5 – The relation between η and $\ln(p')$ observed for many soils on one-dimensional loading and unloading

(5.13) is a non-linear equation and must be solved iteratively to obtain values of K_0. This process is possible (if a little tedious) by hand, using a pocket calculator. As Wroth (1975) points out, (5.13) is only valid for the first unloading from the normally consolidated condition, as the data show that reloading does not follow the original unloading stress path. In practice, however, (5.13) is used irrespective of unloading/reloading cycles that may have taken place. (Only rarely does one know the details of the soil's previous stress history, and in any case some additional empirical relations would be necessary to specify what happens on reloading.)

The basic steps in calculating *in situ* stresses using Wroth's method can be summarised as follows.

1. Calculate σ'_v from the bulk density of the soil and the position of the water table.
2. Calculate σ'_{vm} from an oedometer test. (If no oedometer tests are performed for soil at this particular depth, then interpolate between neighbouring values of σ'_{vm}.)
3. Use (5.11) (Jaky's relation) to calculate K_{nc} and hence the horizontal effective stress acting when the maximum vertical effective stress (σ'_{vm}) was present.
4. Calculate values of p' and q corresponding to the maximum stresses found in 3. Substitute these values into the equation of the yield locus (either (2.18) or (2.40) depending on whether Cam-clay or modified Cam-clay is to be used in the subsequent analysis) to calculate the value of p'_c.

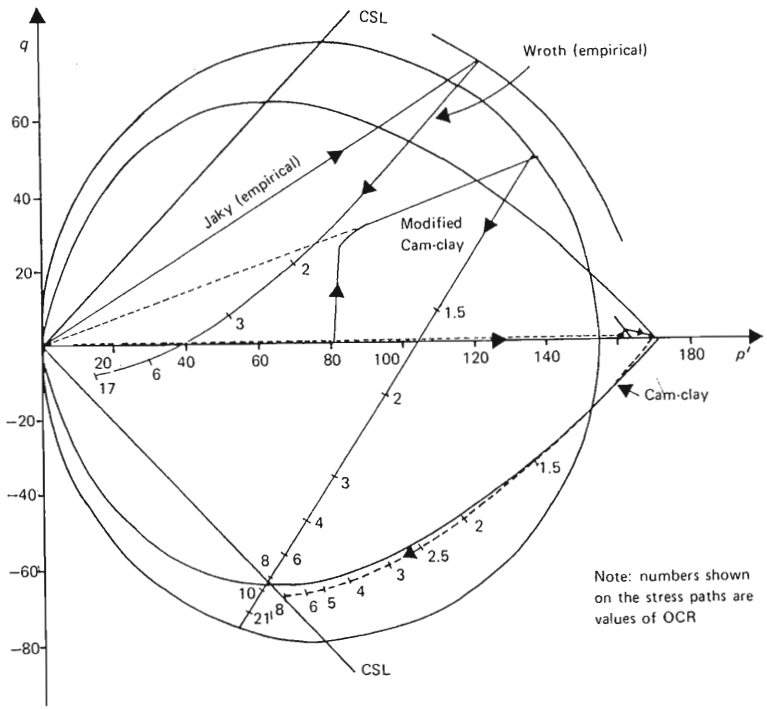


Fig. 5.6 – Different assumptions for loading and unloading Cam-clay and modified Cam-clay one-dimensionally

5. Use either (5.12) or (5.13) to calculate the value of K_0 from K_{nc} and OCR. Hence the *in situ* horizontal effective stress $\sigma'_h = K_0 \sigma'_v$.

5.5.5 Different approaches compared

Fig. 5.6 illustrates the effect of following three different approaches for estimating the *in situ* stresses. In each case the soil is loaded to the same effective vertical stress and then unloaded.

The upper stress path is obtained using Wroth's method with (5.13) for unloading. Although this stress path is shown to establish a modified Cam-clay yield locus, exactly the same stress path is obtained if one is going to use Cam-clay in the subsequent analysis.

The modified Cam-clay and Cam-clay stress paths were obtained from a CRISP analysis. For modified Cam-clay, one can calculate the value of K_{nc} using the theory for calculating strains in Chapter 2 (supplemented by the extra elastic shear strains). However, it is more straightforward to use CRISP to find the theoretical K_{nc}. The analysis was started at from a point on the isotropic normal consolidation line at half the final maximum effective vertical stress. After the initial part of the stress path, which is almost vertical, the stress path bends round and follows the constant η-line corresponding to K_{nc}. The unloading line is straight, and in fact is the same as would be obtained using (5.12). The slope of the unloading line is given by $3(1-\nu')/(1+\nu')$. The value of ν' used here was 0.2, which is slightly lower than the range suggested by Wroth (1975). If a value of ν' of $\frac{1}{3}$ is used, then the unloading part of the stress path has exactly half the slope of the one shown in Fig. 5.6.

The same basic procedure was followed for Cam-clay, producing the lower stress path shown in Fig. 5.6. The analysis was started quite close to σ'_{vm}, because for most values of the CSSM parameters, $K_{nc} = 1$ (the incorporation of elastic shear strains via ν' does not affect this standard result described by Schofield and Wroth (1968)). The unloading part of the stress path involves expansive elastic volumetric strains and compressive plastic volumetric strains, giving an overall volumetric strain which is expansive. When the OCR is equal to 8, the soil is close to a state of passive failure at the critical state.

The *in situ* stresses obtained by using the Cam-clay models directly lead to higher values of K_0 (for a given OCR). This is particularly the case for Cam-clay. On the other hand, it is possible to take account of information describing the complete stress history of the soil (including unloading/reloading cycles) where this is available. A side-effect of using Wroth's method for high values of OCR is that the initial stress state in an analysis is near the origin of the (p', q) plot, well over on the dry side of the critical state. In the subsequent analysis there will be quite a lot of elastic shearing before the soil yields. In an undrained analysis, yielding will take place in a region of stress space (i.e. on the dry side of critical) where the predictions of the Cam-clay models are known to be not very satisfactory. In contrast, using the 'consistent' approach the soil would tend to yield nearer the critical state. Thus the response would be closer to elastic–perfectly-plastic for medium to high over-consolidation ratios.

Clearly the actual response of soil in an analysis depends on the stress history assumed before the start of the analysis. If the soil is over-consolidated then the predictions of soil deformations in the early part of the analysis will be quite sensitive to the assumed unloading relation. On the other hand, if the analysis approaches failure then the main factor which influences the results will be the value of σ'_{vm}. We can compare this situation to that for steel structures where plastic collapse loads are independent of initial (residual) stresses. Collapse loads in geotechnical engineering do depend on the initial stresses, but not necessarily on every detail of the stress history. It is likely that many useful calculations can be carried out with relatively crude estimations of the *in situ* stresses, but we must admit that there has not been much work (that we are aware of) where the effect of different assumptions has been systematically studied.

5.5.6 Final comments on *in situ* stresses

Although CRISP was used to produce the results discussed in the previous section, it is not necessary to perform a complete finite element analysis. Use of

the **D**-matrix routines listed earlier in this chapter is possible: calculating incremental stress changes for imposed one-dimensional incremental strains (updating the current stresses as one proceeds).

A further empirical relation between K_0 and OCR is due to Parry (1982):

$$K_0 = K_{nc}\,(\mathrm{OCR})^{\phi'}.$$

(ϕ' is in radians.) This equation gives values of K_0 similar to (5.13) and its manipulation is slightly more straightforward.

In this discussion of *in situ* stresses, we have failed to mention the experimental evidence that seems to show a yield locus centred on the η-line corresponding to K_{nc} rather than $\eta = 0$. This observation can be incorporated into the critical state framework to produce an anisotropic Cam-clay model (Ohta and Wroth, 1976). This model would yield much earlier on passive stress paths where the isotropic models we have described continue to shear elastically. Yielding on active stress paths will not be much affected, however. We expect that this explains why satisfactory predictions are often produced using Cam-clay where there is positive loading (e.g. under embankments), but unloading problems often show too much elastic behaviour.

No matter how sophisticated the theoretical model, the problem of deciding what has happened to the soil at a particular site still remains. We believe that the simple one-dimensional loading and unloading idealisation of stress history may be appropriate to fewer cases than are commonly supposed. For example, Dalton and Hawkins (1982) measured different values of σ'_h in different directions in the ground using the self-boring pressure meter at an apparently undisturbed site. (Up to 50% variation in σ'_h was detected.) Despite the careful allowance that was made for instrumentation errors, these findings have not been accepted by most geotechnical engineers. We prefer to believe the experimental information, even if it does not fit in with our preconceived notions of what has happened to the ground in the past.

6

Geometry of the Finite Element Mesh

6.1 INTRODUCTION

Chapter 4 described how the program and the input data can be logically divided into three distinct parts: (i) mesh geometry; (ii) material properties and *in situ* stresses; (iii) analysis.

This chapter deals with part (i). **MARKZ** is the master control routine for the geometry part of the program, and is called by routine MAST as described in Chapter 4.

The subroutine hierarchy (Fig. 6.1) shows the routine MARKZ delegating tasks to various routines. A brief explanation of each subroutine listed in this chapter is given below.

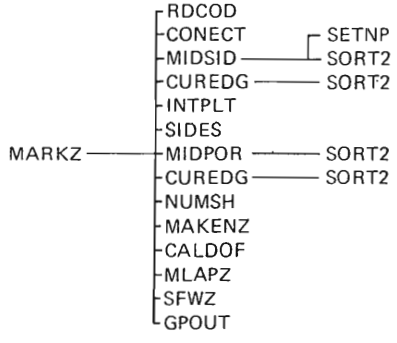


Fig. 6.1 – Subroutine hierarchy for geometry part of program

MARKZ – Control routine for geometry part of program; delegates tasks to other routines.

RDCOD – Reads the node numbers and nodal co-ordinates of *vertex* nodes.

CONECT – Reads the element number, element type number and material zone numbers and the vertex nodes associated with *each* element.

MIDSID – Calculates co-ordinates of additional *displacement* nodes (nodes along element sides and element interiors). These nodes are also numbered.

SORT2 – Returns the lower of two node numbers.

SETNP – Sets up indexes for element sides for different *types* of element.

CUREDG – If the element sides are curved then the nodal co-ordinates calculated by MIDSID for the nodes along element sides, assuming the sides are straight, will be incorrect. This routine allows the user to specify the correct co-ordinates, which replace the co-ordinates calculated by the program.

INTPLT – If a plot of the mesh is required then the overall dimensions of the mesh, in order to calculate the scale, are written to a Plot Data (PD) file.

SIDES – Information to draw element sides are written to PD file.

MIDPOR – Calculates co-ordinates of additional *pore pressure* nodes (nodes along element sides and interiors). The nodes are also assigned numbers.

NUMSH – Information to number the nodes and elements are written to PD file for plotting.

MAKENZ – Calculates the degrees of freedom (d.o.f.) of each node.

CALDOF – Assigns unique global variable numbers to each variable.

MLAPZ – Relevant to the frontal method. Marks last appearance of nodes.

SFWZ – Calculates the maximum frontwidth and the amount of store required for solving the equations.

GPOUT – Prints out nodal co-ordinates and list of nodes associated with each element.

BDATA1 – Block data routine element type dependent parameters and integration schemes.

SHFTIB – Shifts a region to a different part of the global array G.

MAXVA1 – Sets maximum values and sizes of some arrays.

6.2 GEOMETRY PART OF THE PROGRAM

MARKZ delegates tasks to other routines.

Routine MARKZ

```
      SUBROUTINE MARKZ(NVTX,NEL,NUMAX,MUMAX,MXND,MXNDV,NNE,NNE1,      MARK   1
     1 NN,NNU,NNZ,LTAB,LDIM,NDIM,NDF,NDZ,IFRZ,MCORE,MNFZ,            MARK   2
     2 NPL,LTZ,KLT,NMATZ,INXL,IPLOT,                                 MARK   3
     3 XYZ,NCONN,MAT,LTYP,MRELVV,MREL,NRELVV,NREL,NW,NQ,              MARK   4
     4 ITAB,MFRU,MFRN,NDEST,NLST,IFR,NP1,NP2,ND,NCORET,MDZ)           MARK   5
C*************************************************************************MARK   6
C     MASTER CONTROL ROUTINE FOR GEOMETRY PART OF THE PROGRAM.          MARK   7
C     READS INPUT DATA (COORDINATES AND ELEMENT-NODAL                   MARK   8
C     CONNECTIVITY ) AND SETS UP ADDITIONAL ARRAYS.                     MARK   9
C*************************************************************************MARK  10
      CHARACTER*80 TITLE                                               MARK  11
      DIMENSION XYZ(NDIM,NNE),NCONN(MXND,NEL),MAT(NEL),                MARK  12
     1 LTYP(NEL),MRELVV(NEL),MREL(MUMAX),NRELVV(NNE),                  MARK  13
     2 NREL(NNU),NW(NNE1),NQ(NNE),ITAB(LTAB,LDIM),MFRU(NEL),MFRN(MUMAX),MARK  14
     3 NDEST(NNE),NLST(MXND),IFR(IFRZ),NP1(NPL),NP2(NPL),KLT(LTZ)     MARK  15
      COMMON /DEVICE/ IR1,IR4,IR5,IW2,IW4,IW6,IW7,IW8,IW9              MARK  16
      COMMON /DEBUGS/ ID1,ID2,ID3,ID4,ID5,ID6,ID7,ID8,ID9,ID10         MARK  17
      COMMON /LABEL / TITLE                                            MARK  18
      READ(IR5,*)ID1,ID2,ID3,ID4,ID5,ID6,ID7,ID8,ID9,ID10             MARK  19
C-----------------------------------------------------------------------MARK  20
C     NSDZ - MAXIMUM NUMBER OF DISPLACEMENT NODES ALONG EDGE           MARK  21
C     NSPZ - MAXUMUM NUMBER OF PORE-PRESSURE NODES ALONG EDGE          MARK  22
C            (EXCLUDING END NODES)                                     MARK  23
C-----------------------------------------------------------------------MARK  24
      READ(IR5,*)NSDZ,NSPZ,NDCUR,NPCUR                                 MARK  25
      WRITE(IW6,902)NSDZ,NSPZ,NDCUR,NPCUR                              MARK  26
C-----------------------------------------------------------------------MARK  27
C     READ VERTEX NODE COORDINATES                                     MARK  28
C-----------------------------------------------------------------------MARK  29
      CALL RDCOD(IR5,IW6,NNE,NDIM,NNU,NVTX,NUMAX,XYZ,NRELVV,NREL)      MARK  30
C-----------------------------------------------------------------------MARK  31
C     READ ELEMENT-NODAL CONNECTIVITY                                  MARK  32
C-----------------------------------------------------------------------MARK  33
      CALL CONECT(IR5,IW6,MXND,NEL,MUMAX,NNE,NNU,MXNDV,NCONN,          MARK  34
     1   MAT,LTYP,MRELVV,MREL,NRELVV,NREL,MFRU,MFRN,NLST,              MARK  35
     1   LTZ,KLT,NMATZ,NVTX,NUMAX)                                     MARK  36
      IF(ID1.EQ.1)WRITE(IW6,801)NCONN                                  MARK  37
C-----------------------------------------------------------------------MARK  38
C     CALCULATE COORDINATES OF ADDITIONAL NODES                        MARK  39
C-----------------------------------------------------------------------MARK  40
      CALL MIDSID(IW6,MXND,NEL,LTAB,LDIM,NNU,NDIM,NNE,NPL,             MARK  41
     1   XYZ,NCONN,LTYP,MRELVV,NRELVV,NREL,ITAB,                       MARK  42
     1   NP1,NP2,ND,NN,KRD,NVTX,NDZ,MDZ)                               MARK  43
C-----------------------------------------------------------------------MARK  44
C     READ COORDINATES OF DISPLACEMENT NODES ALONG CURVED SIDES        MARK  45
C     NDCUR - NUMBER OF ELEMENT SIDES (WITH DISPLACEMENT NODES)        MARK  46
C             THAT ARE CURVED.                                         MARK  47
C-----------------------------------------------------------------------MARK  48
      IF(NDCUR.EQ.0)GOTO 10                                            MARK  49
      CALL CUREDG(IR5,IW6,MXND,NEL,NDIM,NNE,LTAB,LDIM,MUMAX,NNU,NPL,   MARK  50
     1   XYZ,NCONN,LTYP,MREL,NREL,ITAB,NP1,NP2,NDCUR,1,NSDZ)           MARK  51
   10 CONTINUE                                                         MARK  52
C-----------------------------------------------------------------------MARK  53
C     WRITE TITLE AND DIMENSIONS OF MESH TO A PLOT FILE                MARK  54
C-----------------------------------------------------------------------MARK  55
      IF(IPLOT.NE.0)WRITE(IW8)TITLE                                    MARK  56
      IF(IPLOT.NE.0)CALL INTPLT(IW6,IW8,NDIM,NNE,XYZ,ND)               MARK  57
C-----------------------------------------------------------------------MARK  58
C     PLOT ELEMENT SIDES                                               MARK  59
C-----------------------------------------------------------------------MARK  60
      CALL SIDES(IW6,IW8,LTAB,LDIM,NDIM,NNE,MXND,NEL,XYZ,              MARK  61
     1   NCONN,ITAB)                                                   MARK  62
C-----------------------------------------------------------------------MARK  63
C     CALCULATE COORDINATES OF ADDITIONAL PORE-PRESSURE NODES          MARK  64
C-----------------------------------------------------------------------MARK  65
      CALL MIDPOR(IW6,MXND,NEL,LTAB,LDIM,NNU,NDIM,NNE,NPL,             MARK  66
     1   XYZ,NCONN,LTYP,MRELVV,NRELVV,NREL,ITAB,                       MARK  67
     1   NP1,NP2,NN,KRD,NNZ)                                           MARK  68
```

```
C-----------------------------------------------------------------------MARK  69
C     READ COORDINATES OF PORE-PRESSURE NODES ALONG CURVED SIDES        MARK  70
C     NPCUR - NUMBER OF ELEMENT SIDES (WITH PORE PRESSURES NODES)       MARK  71
C             THAT ARE CURVED.                                          MARK  72
C-----------------------------------------------------------------------MARK  73
      IF(NPCUR.EQ.0)GO TO 20                                            MARK  74
      CALL CUREDG(IR5,IW6,MXND,NEL,NDIM,NNE,LTAB,LDIM,MUMAX,NNU,NPL,    MARK  75
     1   XYZ,NCONN,LTYP,MREL,NREL,ITAB,NP1,NP2,NPCUR,2,NSPZ)            MARK  76
   20 CONTINUE                                                          MARK  77
      NN1=NN+1                                                          MARK  78
      IF(ID7.EQ.0)GOTO 22                                               MARK  79
      WRITE(IW6,801)NCONN                                               MARK  80
  801 FORMAT(/1X,5HNCONN/(1X,20I5))                                     MARK  81
      WRITE(IW6,802)MREL                                                MARK  82
  802 FORMAT(/1X,4HMREL/(1X,20I5))                                      MARK  83
      WRITE(IW6,803)MRELVV                                              MARK  84
  803 FORMAT(/1X,6HMRELVV/(1X,20I5))                                    MARK  85
      WRITE(IW6,804)NREL                                                MARK  86
  804 FORMAT(/1X,4HNREL/(1X,20I5))                                      MARK  87
      WRITE(IW6,805)NRELVV                                              MARK  88
  805 FORMAT(/1X,6HNRELVV/(1X,20I5))                                    MARK  89
      WRITE(IW6,806)LTYP                                                MARK  90
  806 FORMAT(/1X,4HLTYP/(1X,20I5))                                      MARK  91
      WRITE(IW6,807)MAT                                                 MARK  92
  807 FORMAT(/1X,3HMAT/(1X,20I5))                                       MARK  93
   22 CONTINUE                                                          MARK  94
C-----------------------------------------------------------------------MARK  95
C     NUMBER THE MESH                                                   MARK  96
C-----------------------------------------------------------------------MARK  97
      CALL NUMSH(IW6,IW8,NDIM,NNE,MXND,NEL,MUMAX,NNU,                   MARK  98
     1   XYZ,NCONN,LTYP,MREL,NREL,NDZ,IPLOT)                            MARK  99
C-----------------------------------------------------------------------MARK 100
C     CALCULATE NUMBER OF DEGREES OF FREEDOM FOR EACH NODE              MARK 101
C-----------------------------------------------------------------------MARK 102
      CALL MAKENZ(MXND,NEL,NN,NCONN,LTYP,NQ,INXL)                       MARK 103
      IF(ID7.EQ.1)WRITE(IW6,809)NQ                                      MARK 104
  809 FORMAT(/1X,2HNQ/(1X,20I5/))                                       MARK 105
C-----------------------------------------------------------------------MARK 106
C     GENERATE GLOBAL NUMBERS FOR ALL D.O.F.                            MARK 107
C-----------------------------------------------------------------------MARK 108
      CALL CALDOF(IW6,NN,NN1,NDF,NW,NQ)                                 MARK 109
C-----------------------------------------------------------------------MARK 110
C     MARK LAST APPEARANCE OF ALL NODES                                 MARK 111
C-----------------------------------------------------------------------MARK 112
      CALL MLAPZ(MXND,NEL,NN,NCONN,LTYP,NQ)                             MARK 113
C-----------------------------------------------------------------------MARK 114
C     CALCULATE MAXIMUM FRONTWIDTH AND MINIMUM STORE FOR SOLUTION       MARK 115
C-----------------------------------------------------------------------MARK 116
      CALL SFWZ(MNFZ,MXND,NEL,NN,MUMAX,NNU,IFRZ,NCONN,                  MARK 117
     1   LTYP,MREL,NREL,NQ,NDEST,IFR,-1,MCORE,NCORET)                   MARK 118
C-----------------------------------------------------------------------MARK 119
C     PRINT OUT ARRAYS                                                  MARK 120
C-----------------------------------------------------------------------MARK 121
      CALL GPOUT(IW6,MXND,NEL,MUMAX,NN,NN1,NDF,                         MARK 122
     1   NCONN,MAT,LTYP,MRELVV,MREL,NRELVV,NW,NQ,NLST)                  MARK 123
C                                                                       MARK 124
      RETURN                                                            MARK 125
  902 FORMAT(/                                                          MARK 126
     1 10X,46HMAX NUMBER OF DISPLACEMENT NODES ALONG EDGE..=,I8/        MARK 127
     2 10X,46HMAX NUMBER OF PORE-PRESSURE NODES ALONG EDGE.=,I8/        MARK 128
     3 10X,46HNUMBER OF CURVED EDGES (DISPLACEMENT).....=,I8/           MARK 129
     4 10X,46HNUMBER OF CURVED EDGES (PORE-PRESSURE).....=,I8/          MARK 130
     5 /120(1H*)/)                                                      MARK 131
      END                                                               MARK 132
```


MARK 19 : read debug option – a set of 10 flags to print out various arrays used or calculated in the geometry part of the program.

MARK 25–26 : read and write details of curved sides – only relevant if there are any in the mesh. The normal option is straight-edged elements (then all values are set to zero).

MARK 30 : read vertex node co-ordinates (the user needs to specify only co-ordinates of the vertex nodes at this stage of the analysis, irrespective of the 'order' of the elements being used). If the elements are straight-edged then these are the only co-ordinates to be specified by the user. Any additional nodes (depending on the 'order' of the element) will be calculated by the program.

MARK 34–37 : read the nodal connectivity list (vertex nodes associated with each element). Also read the element type and material zone number for each element.

MARK 41–43 : calculate the additional (displacement) node co-ordinates.

MARK 50–51 : if some element edges are curved, then read the nodal co-ordinates of displacement nodes along all edges that are curved.

MARK 56–57 : write title of analysis to Plot Data (PD) file. Calculate the overall dimensions of the finite element mesh and write these to PD file only if a plot is required.

MARK 61–62 : write to PD file information (co-ordinates of nodes at either end of all element edges) necessary to draw the mesh. If the element edges are curved, write the co-ordinates of the intermediate nodes as well.

MARK 66–68 : calculate co-ordinates of additional pore pressure nodes (if any).

MARK 75 : if the element edges are curved, then read the nodal co-ordinates of pore pressure nodes (if any) along all edges that are curved. For example, if the element type is 2 and the element edges are curved then there is one additional displacement node along each edge. Therefore it is only necessary to specify the co-ordinates of the displacement node (there are no additional pore pressure nodes along the edge for element type 2).

MARK 79–93 : print out arrays for debugging (only if the debug flag ID7 is set to 1).

MARK 98–99 : write relevant information to the PD file to number the mesh and close the PD file.

MARK 103–105 : calculate the d.o.f. (no. of variables) for each node. This is necessary if different elements sharing a node have different d.o.f. (e.g. elements of type 2 and 3 sharing an edge) – print array NQ for debugging.

MARK 109 : calculate the total no. of d.o.f. (variables) in the mesh. All d.o.f. are assigned a unique global variable number (g.v.n.). An array NW(NN+1) is set up which gives the g.v.n. of the first d.o.f. (variable) of all nodes. All d.o.f. of a node are given consecutive numbers. For example, if the g.v.n. of the first d.o.f. of node 53 is NW(53) = 131, then if node 53 has 3 d.o.f., the global variable numbers are 131, 132 and 133 respectively, and for the next node, 54, NW(54) = 134.

MARK 113 : mark last appearance of all nodes.

MARK 117–118 : pre-front routine. Calculate maximum front size and the store required to solve equations.

MARK 122–123 : print out arrays from geometry part of the program.

The geometry part of the input data consists of the type of elements being used in the mesh, the co-ordinates of all vertex nodes and the list of elements and the nodes associated with each. This scheme is illustrated by means of a simple example (Fig. 6.2).

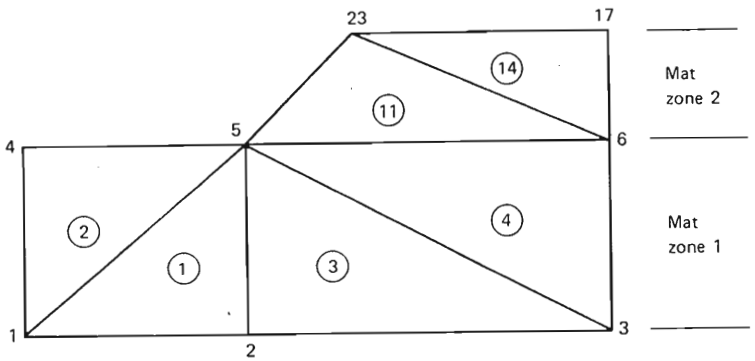


Fig. 6.2 – Example problem: six LST elements of type 2

There are six elements in the mesh: NEL = 6. Each is a six-noded Linear Strain Triangle (LST), and the element numbers are shown circled. The vertex nodes are in the range 1 to 23 and the total number of vertex nodes is represented by NVTX = 8.

NEL – Number of ELements in mesh
NVTX – Number of VerTeX nodes in mesh

In considering the mesh, one has to identify different zones of material behaviour. Each zone is identified by a number, and all elements which are within that zone are given the same number. At this stage it is sufficient to differentiate between the different zones. The question of what type of soil behaviour each zone represents is considered in Chapter 7. In the input data,

the material zone number is denoted by IMAT. Each element is also identified by an element type number (see Fig. 4.1).

Note that only the vertex nodes have to be numbered by the user. This eases the problem of data preparation as the program numbers all other nodes and calculates their co-ordinates. To differentiate between the vertex nodes and the other nodes, the additional nodes are numbered, starting with 751. The program allows gaps in both element and vertex node numbering. When dealing with large finite element meshes (which is the case, most of the time) the meshes may have to be modified a number of times and this allows the renumbering to be carried out without too much difficulty.

There are two sets of node and element numbers. One set is assigned by the user. The program sets up its own node and element numbers, which are strictly for use within the program for reasons of efficiency. The maintenance of these two sets of numbers requires two arrays:

for node numbers – NREL and NRELVV
for element numbers – MREL and MRELVV

These are 'cross-reference arrays'. The sizes of these arrays will depend on the maximum values of element and node numbers specified by the user and will vary from problem to problem. In the above example the maximum element number is 14, i.e. MUMAX = 14, and the maximum vertex node number is 23, i.e. NUMAX = 23. There are no limits set on the maximum number of elements and nodes in any mesh. These are only constrained by the amount of memory available on any particular computer. NDIM represents the number of dimensions in the problem. NDIM = 2 for all two-dimensional plane strain and axisymmetric problems.

MUMAX – MAXimum value of User eleMent number
NUMAX – MAXimum value of User vertex Node number

For the above example the element chosen was the six-noded linear strain triangle. The analysis is of the undrained type, and referring to the list of different element types (see Fig. 4.1), this element is type 2. For example, if elements of type 2 and 3 are mixed in a mesh, then MXTYP = 3. For the present example, the element type with the greatest number of nodes is 2; hence MXTYP = 2. Again the *maximum* number of vertex nodes in any element in the mesh is 3; therefore MXNDV = 3. The nodal co-ordinates are input with one line of data per vertex node.

MXTYP – element TYPe with MaXimum number of nodes or d.o.f.
MXNDV – MaXimum number of Vertex NoDes

User node number	x co-ordinate	y co-ordinate
1	0.0	0.0
2	20.0	0.0
3	40.0	0.0
4	0.0	16.0
5	20.0	16.0
6	40.0	16.0
23	30.0	21.0
17	40.0	21.0

The user node numbers are entered in an array NRELVV(NN), e.g.

NRELVV(1) = 1, NRELVV(2) = 2, . . . NRELVV(7) = 23,
NRELVV(8) = 17.

The last two are the seventh and eighth nodes in the list. The co-ordinates are entered in XYZ(NDIM,NN). Note that the indexes to array XYZ are the same as for array NRELVV.

XYZ(1, 1) = 0. XYZ(2, 1) = 0,

the x and y co-ordinates of the first node in the list.

XYZ(1, 7) = 30. XYZ(2, 7) = 21,

the x and y co-ordinates of the seventh node in the list.

Here NN is the total number of nodes in the mesh. At this stage the exact value of NN is not known. An estimate (NNE) is made, first assuming for example that there are three additional nodes in each element. For six elements it is 18. The actual number will be less because most of the nodes are shared between elements.

NNE = NVTX + NDEAD
= 8 + 18
= 26

NDEAD – ADditional number of NoDEs estimated by the program.

The indexes to array NRELVV are referred to as the *program* node numbers. Array NRELVV gives the 'user' node number for a given 'program' node number. The cross-reference array NREL is set up to do just the opposite: given a 'user' node number, it specifies the 'program' node number.

NRELVV(7) = 23
NREL(23) = 7

The above tasks are carried out by routine **RDCOD**.

Routine RDCOD

```
      SUBROUTINE RDCOD(IR5,IW6,NNE,NDIM,NNU,NVTX,NUMAX,XYZ,NRELVV,NREL) RDCD   1
C*************************************************************************RDCD   2
C     ROUTINE TO READ THE COORDINATES OF VERTEX NODES                     RDCD   3
C*************************************************************************RDCD   4
      DIMENSION XYZ(NDIM,NNE),NRELVV(NNE),NREL(NNU)                       RDCD   5
C                                                                         RDCD   6
      WRITE(IW6,900)                                                      RDCD   7
      WRITE(IW6,901)                                                      RDCD   8
C-------------------------------------------------------------------------RDCD   9
C     INITIALISE NREL, NRELVV                                             RDCD  10
C-------------------------------------------------------------------------RDCD  11
      CALL ZEROI1(NRELVV,NNE)                                             RDCD  12
      CALL ZEROI1(NREL,NNU)                                               RDCD  13
C-------------------------------------------------------------------------RDCD  14
C     READ ALL VERTEX NODE COORDINATES                                    RDCD  15
C-------------------------------------------------------------------------RDCD  16
      DO 10 J=1,NVTX                                                      RDCD  17
      READ(IR5,*)K,(XYZ(ID,J),ID=1,NDIM)                                  RDCD  18
      WRITE(IW6,906)K,(XYZ(ID,J),ID=1,NDIM)                               RDCD  19
      NRELVV(J)=K                                                         RDCD  20
   10 NREL(K)=J                                                           RDCD  21
      RETURN                                                              RDCD  22
  900 FORMAT(//10X,28HCO-ORDINATES OF VERTEX NODES)                       RDCD  23
  901 FORMAT(/3X,4HNODE,5X,1HX,9X,1HY,9X,1HZ/)                            RDCD  24
  906 FORMAT(1X,I5,3F10.3)                                                RDCD  25
      END                                                                 RDCD  26
```

RDCD 7–8 : write output header.
RDCD 12–13 : zero arrays NRELVV and NREL. Array NRELVV stores the node numbers (*user* nos.) in the same sequence as they are read. The sequence in which these are read are the *program* node numbers. NREL is the cross-reference array.
RDCD 17 : loop to read all vertex node co-ordinates.
RDCD 18–19 : read and write the node number and co-ordinates.
RDCD 20 : enter user node number in array NRELVV.
RDCD 21 : enter program node number in array NREL.

6.3 NODAL CONNECTIVITY

The next input data are the node numbers which are associated with each element. This link between nodes and elements is referred to as element–nodal *connectivity*. The data are as follows:

Element number	Node 1	Node 2	Node 3	Node 4	Node 5	Node 6
1	1	2	5	–	–	–
2	5	4	1	–	–	–
3	5	2	3	–	–	–
4	5	3	6	–	–	–
11	5	6	23	–	–	–
14	6	17	23	–	–	–

Each element has to be assigned a material zone number (IMAT) and element type number (ITYP) (see Fig. 4.1 for different element types). Therefore the input data are as follows:

Element no. KEL	Element type no. ITYP	Material zone no. IMAT	Node 1 NLST(1)	Node 2 NLST(2)	Node 3 NLST(3)
1	2	1	1	2	5
2	2	1	5	4	1
3	2	1	5	2	3
4	2	1	5	3	6
11	2	2	5	6	23
14	2	2	6	17	23

As in the case of the nodes, the element numbers (KEL) are entered in array MRELVV(NEL) as they are read, e.g.

MRELVV(1) = 1, MRELVV(2) = 2, ... MRELVV(5) = 11,
MRELVV(6) = 14.

The sixth element in the list has the number 14. A cross-reference array MREL(MUMAX) is then set up. For the above example:

MREL(1) = 1
MREL(2) = 2
...
...
MREL(11) = 5
MREL(14) = 6.

This gives the 'program' element numbers for 'user' element numbers. Element type number (ITYP) and the material zone number (IMAT) are entered in arrays LTYP(NEL) and MAT(NEL) respectively. The indexes to these arrays are the same as for the arrays MRELVV. These indexes are the position of the elements in the input data list. From the input data, it can be seen that elements 1, 2, 3 and 4 belong to material zone 1, and elements 11 and 14 to material zone 2 (Fig. 6.2).

The numbers marked inside each element near the vertex nodes (in Fig. 6.3) are the indexes to the array NLST and NCONN(NTPE,NEL). These indexes define the *local* node numbering, and in the rest of the book they will be referred to as the indexes to array NCONN. The indexes can begin at any node, but then should follow an *anti-clockwise* ordering. Specifying the nodes in clockwise order results in a negative value for the area of the element and will cause the program to stop at a later stage.

Array NLST(MXNDV) is a temporary array for storing nodes associated with each element as they are read. Array NCONN is the nodal connectivity array.

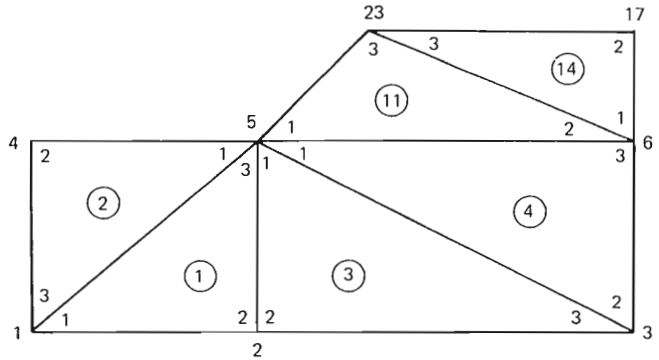


Fig. 6.3 – Indexes to array NCONN

The only difference between NLST and NCONN is that NCONN contains the 'program' node numbers.

The contents of array NCONN appear as

Element	Index to NCONN	Node 1	Node 2	Node 3	Node 4	Node 5	Node 6
1	1	1	2	5	0	0	0
2	2	5	4	1	0	0	0
3	3	5	2	3	0	0	0
4	4	5	3	6	0	0	0
11	5	5	6	7	0	0	0
14	6	6	8	7	0	0	0

NCONN(1,5) = 5, NCONN(2,5) = 6 and NCONN(3,5) = 7. These are the first, second and third nodes associated with the fifth element (which has the number 11) in the list; note that the locations 4, 5 and 6 are empty. They have zero values at this stage and will be replaced by the edge (side) node numbers when they are assigned by the program later.

The nodes that define the variation of displacements are also used to define the element geometry, which is the well known isoparametric formulation. The nodes are referred to as 'displacement' nodes in the rest of the book. The lower-order elements have a linear variation of strain across the element (element types 2 and 3). For undrained and drained problems, the displacements are the only unknowns.

For coupled consolidation analysis there are additional excess pore pressure variables; appropriate element types (3 and 7) will be referred to as *consolidation* elements. The pore pressure nodes are positioned such that the variation of excess pore pressure is of the same order as the variation in strain. For example, for the cubic strain triangle, nodes 16 to 21 are pore pressure nodes (see Fig. 4.1). For 'consolidation' elements the vertex nodes have both

displacements and excess pore pressures as variables. The type 3 'consolidation' element does not have 'additional' pore pressure nodes. It has three displacement nodes and no *additional* pore pressure nodes (see Fig. 6.4). Thus, in general, different nodes will have different d.o.f. (variables).

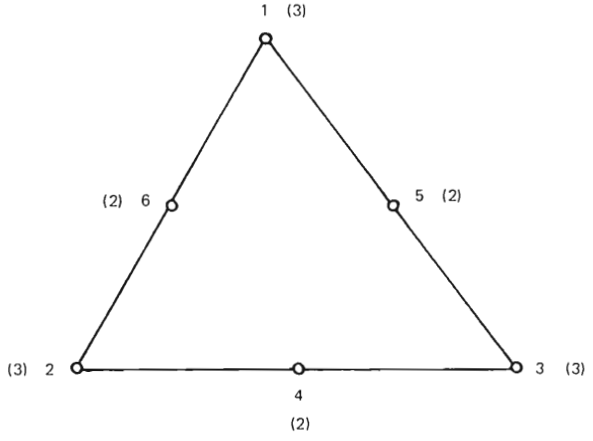


Fig. 6.4 – Different d.o.f. at different nodes

The order in which each element is listed in the input data is the program element number by default and it is in fact the order in which the element stiffness terms will be assembled in the FRONTAL method of solution. However, the sequence in which the elements were input by the user may not always be the optimum sequence for the frontal method.

At present a number of 'stand-alone' programs and techniques which can optimise the element numbering for the frontal method are available (Akin and Pardue, 1975; Razzaque, 1980; Sloan and Randolph, 1983). These programs only need the element–nodal connectivity list as input data. Therefore the option of specifying an alternative order of the elements, which is less costly for the frontal method, has been included. Hence there are two sets of element numbers. The first is the arbitrary element numbering specified by the user; the second is the element numbering sequence which is better for the frontal method. The user has to be aware of the importance of having an efficient element numbering. Just as efficient node numbering is very desirable for band solvers, efficient element numbering is very desirable for the frontal method to keep the cost of computation and core-store equipment down.

In the input data after all the co-ordinates of the vertex nodes have been specified, a parameter IRNFR is specified. If this parameter is set to 1 then the user will specify an alternative element numbering sequence starting from the next data record. It is followed by the element–nodal connectivity list as

described before. If IRNFR is set to 0, an alternative element numbering will not be provided by the user, and the elements will be assembled in the order they are specified in the input data. For the above example, an improved frontal sequence could be

```
IRNFR
  1
Element numbers
  2  1  3  4  11  14
```

If the alternative element order is not available then IRNFR is set equal to 0 and the element–nodal connectivity list follows immediately. The list is unchanged whether IRNFR = 0 or 1. The alternative numbering, if specified, is read into an array MFRU(NEL) in routine **CONECT**.

$$\text{MRFRU}(1) = 2 \quad \text{MFRU}(2) = 1 \quad \ldots \quad \text{MFRU}(5) = 11 \quad \text{MFRU}(6) = 14$$

A cross-reference array MFRN(MUMAX) is set up at the same time.

$$\text{MFRN}(1) = 2 \quad \text{MFRN}(2) = 1 \quad \text{MFRN}(11) = 5 \quad \text{MFRN}(14) = 6$$

These two arrays are then used to set up the arrays MRELVV(NEL) and MREL(MUMAX) while the element–nodal connectivity list is being read. It should be noted that the contents of MRELVV(NEL) and MREL(MUMAX) are different, depending on whether the optimum element sequence for the frontal method has been specified or not. The contents of these arrays are as follows (note the difference when an alternative element numbering is not provided):

$$\text{MRELVV}(1) = 2 \quad \text{MRELVV}(2) = 1 \quad \text{MRELVV}(3) = 3$$
$$\text{MRELVV}(4) = 4 \quad \text{MRELVV}(5) = 11 \quad \text{MRELVV}(6) = 14$$

$$\text{MREL}(1) = 2 \quad \text{MREL}(2) = 1 \quad \text{MREL}(3) = 3$$
$$\text{MREL}(4) = 4 \quad \text{MREL}(11) = 5 \quad \text{MREL}(14) = 6$$

Routine CONECT

```
      SUBROUTINE CONECT(IR5,IW6,MXND,NEL,MUMAX,NNE,NNU,MXNDV,NCONN,     CNCT   1
     1 MAT,LTYP,MRELVV,MREL,NRELVV,NREL,MFRU,MFRN,NLST,                 CNCT   2
     2 LTZ,KLT,NMATZ,NVTX,NUMAX)                                       CNCT   3
C*****************************************************************    CNCT   4
C     SUBROUTINE TO READ ELEMENT-NODAL CONNECTIVITY                    CNCT   5
C*****************************************************************    CNCT   6
      DIMENSION NCONN(MXND,NEL),MAT(NEL),LTYP(NEL),MRELVV(NEL),        CNCT   7
     1 MREL(MUMAX),NRELVV(NNE),NREL(NNU),MFRU(NEL),MFRN(MUMAX),        CNCT   8
     1 NLST(MXNDV),KLT(LTZ)                                            CNCT   9
      COMMON /DEBUGS/ ID1,ID2,ID3,ID4,ID5,ID6,ID7,ID8,ID9,ID10         CNCT  10
      COMMON /ELINF / LINFO(50,15)                                     CNCT  11
C                                                                      CNCT  12
      READ(IR5,*)IRNFR                                                 CNCT  13
      WRITE(IW6,901)IRNFR                                              CNCT  14
      IF(IRNFR.NE.1)GO TO 30                                           CNCT  15
C-----------------------------------------------------------------     CNCT  16
C     READ OPTIMUM FRONTAL ORDER OF ELEMENTS                           CNCT  17
C-----------------------------------------------------------------     CNCT  18
      WRITE(IW6,902)                                                   CNCT  19
```

```
      READ(IR5,*)(MFRU(IL),IL=1,NEL)                                    CNCT  20
      WRITE(IW6,904)(MFRU(IL),IL=1,NEL)                                 CNCT  21
C                                                                       CNCT  22
      CALL ZEROI1(MFRN,MUMAX)                                           CNCT  23
C                                                                       CNCT  24
      DO 20 IM=1,NEL                                                    CNCT  25
      LU=MFRU(IM)                                                       CNCT  26
   20 MFRN(LU)=IM                                                       CNCT  27
C                                                                       CNCT  28
      IF(ID6.EQ.1)WRITE(IW6,930)MFRN                                    CNCT  29
C                                                                       CNCT  30
   30 CALL ZEROI2(NCONN,MXND,NEL)                                       CNCT  31
      CALL ZEROI1(LTYP,NEL)                                             CNCT  32
      CALL ZEROI1(MAT,NEL)                                              CNCT  33
      CALL ZEROI1(MREL,MUMAX)                                           CNCT  34
C                                                                       CNCT  35
      WRITE(IW6,906)                                                    CNCT  36
C                                                                       CNCT  37
      DO 100 IL=1,NEL                                                   CNCT  38
C-----------------------------------------------------------------------CNCT  39
C     READ ELEMENT NUMBER, TYPE NUMBER, MATERIAL ZONE NUMBER AND        CNCT  40
C     VERTEX NODE NUMBERS                                               CNCT  41
C-----------------------------------------------------------------------CNCT  42
      READ(IR5,*)KEL,ITYP,IMAT,NLST                                     CNCT  43
      WRITE(IW6,909)KEL,ITYP,IMAT,NLST                                  CNCT  44
C                                                                       CNCT  45
      NVN=LINFO(2,ITYP)                                                 CNCT  46
C                                                                       CNCT  47
      MNW=IL                                                            CNCT  48
      IF(IRNFR.EQ.1)MNW=MFRN(KEL)                                       CNCT  49
C                                                                       CNCT  50
      MRELVV(MNW)=KEL                                                   CNCT  51
      LTYP(MNW)=ITYP                                                    CNCT  52
      MAT(MNW)=IMAT                                                     CNCT  53
      MREL(KEL)=MNW                                                     CNCT  54
C                                                                       CNCT  55
      DO 95 IK=1,NVN                                                    CNCT  56
      NUS=NLST(IK)                                                      CNCT  57
      NPR=NREL(NUS)                                                     CNCT  58
   95 NCONN(IK,MNW)=NPR                                                 CNCT  59
C                                                                       CNCT  60
  100 CONTINUE                                                          CNCT  61
      IF(ID5.EQ.0)GOTO 105                                              CNCT  62
      WRITE(IW6,991)NCONN                                               CNCT  63
  991 FORMAT(/1X,5HNCONN/(1X,20I5))                                     CNCT  64
      WRITE(IW6,992)MREL                                                CNCT  65
  992 FORMAT(/1X,4HMREL/(1X,20I5))                                      CNCT  66
      WRITE(IW6,993)MRELVV                                              CNCT  67
  993 FORMAT(/1X,6HMRELVV/(1X,20I5))                                    CNCT  68
  105 CONTINUE                                                          CNCT  69
C                                                                       CNCT  70
      CALL ZEROI1(KLT,LTZ)                                              CNCT  71
C                                                                       CNCT  72
      DO 150 IL=1,NEL                                                   CNCT  73
      LT=LTYP(IL)                                                       CNCT  74
  150 KLT(LT)=KLT(LT)+1                                                 CNCT  75
      RETURN                                                            CNCT  76
  901 FORMAT(/1X,7HIRNFR =,I5)                                          CNCT  77
  902 FORMAT(/1X,36HOPTIMISED SOLUTION ORDER OF ELEMENTS/)              CNCT  78
  904 FORMAT(1X,20I5)                                                   CNCT  79
  906 FORMAT(/1X,46HELEMENT TYPE  MAT    1     2     3     4     5,     CNCT  80
     1 18H     6     7     8/)                                          CNCT  81
  909 FORMAT(I5,2X,2I5,15I6)                                            CNCT  82
  930 FORMAT(/1X,4HMFRN/(1X,20I5))                                      CNCT  83
      END                                                               CNCT  84
```


CNCT 13–14 : read and write the code IRNFR, which indicates that the user will specify an alternative optimum frontal numbering of elements (otherwise the user-specified element sequence will be used as the sequence for frontal assembly).

CNCT 15 : skip if alternative element numbers for frontal method are not specified.

CNCT 19–21 : read and write the alternative element numbering sequence which is more efficient for the frontal method.

CNCT 23 : zero cross-reference of frontal element numbering sequence array MFRN.

CNCT 25 : loop on all elements (in frontal sequence).

CNCT 26 : new element number.

CNCT 27 : form cross-reference array MFRN.

CNCT 29 : debugging option – print out array MFRN.

CNCT 31–34 : zero arrays NCONN (element–nodal connectivity array), LTYP (element type array), MAT (element material zone array) and MREL (cross-reference array of element number).

CNCT 38 : loop on all elements.

CNCT 43–44 : read and write element number (KEL), element type number (ITYP), element material zone number (IMAT), list of vertex nodes associated with the element (NLST).

CNCT 46 : NVN – the number of vertex nodes in element.

CNCT 49 : if alternative frontal element numbering is available, obtain number from array MFRN or use ascending order of element number sequence.

CNCT 51–53 : store element number, element type number and element material zone number.

CNCT 54 : enter in cross-reference array.

CNCT 56 : loop on all vertex nodes of element.

CNCT 59 : enter node number in connectivity array NCONN.

CNCT 61 : end of element loop.

CNCT 63–68 : print out arrays NCONN, MREL and MRELVV for debugging.

CNCT 71 : zero array KLT – counter of elements of each type.

CNCT 73–75 : count the number of elements of each type.

All the element types provided in this program have additional nodes along the element sides (edges). The next stage of the program is to assign numbers to these nodes and calculate the co-ordinates by linear interpolation from the co-ordinates of nodes at either end of the element sides. The number of displacement nodes along the sides depends on the *order* of the element. The lower-order elements presented here are the *linear strain* elements, which have one node at the midpoint of the side (hence the name 'midside' node).

The elements are considered in the sequence they appeared in the input data. Each side of the element is considered in turn in the anti-clockwise order. An entry is made as soon as the nodes along an edge have been numbered and its

co-ordinate calculated. The following procedure avoids the possibility of nodes being given two different numbers (i.e. being numbered twice) when they are common to two or more elements. Each side is identified by a unique code IHASH = N1 * 10000 + N2, where N1 = lower-node and N2 = higher-node numbers. Whenever a new edge is encountered, 1 is entered against this code in a hash table. The procedure is to consult this entry to see whether nodes along a particular edge have already been numbered.

The terms 'edge' and 'side' are used interchangeably here. The term 'edge' is preferred because when extended to three-dimensional elements it remains unambiguous whereas 'side' would mean the element 'face'. Since this book mainly deals with two-dimensional programming aspects, both words have the same meaning.

Once nodes along the sides have been numbered, nodes within element interiors, if present, are numbered. This procedure is repeated for all the elements. The hash table ITAB contains the information in code form for all the element sides which is later used in routine SIDES to create a PD file. CRISP does not have any plotting routines, and hence no plots are produced. However, it creates the data necessary to produce the plot. A separate program is then needed to plot the mesh. (A suitable program is included in Appendix B.)

6.4 NUMBERING THE ADDITIONAL DISPLACEMENT NODES

For each element, as nodes along the sides and element interiors are numbered, they are entered in the array NCONN after the vertex node numbers. At the same time, cross-reference arrays NREL and NRELVV are also updated.

The routine **MIDSID** calculates the numbers and co-ordinates of the additional (displacement) nodes and also sets up the information necessary to plot the mesh. CRISP writes the information necessary to plot the mesh to a PD file. The program uses a simple technique to scane the element sides. For example, if node 53 is connected to nodes 23, 28, 70, 5, 99 and 123 then only the sides to nodes 70, 99 and 123 are written to the PD file. This process begins with the node with the lowest number (usually 1) and then continued in the ascending order.

The entries made in array ITAB for each element side (IHASH represents the code identifying an element side) are always linked to the smaller of the two nodes at either end. Since IHASH has a unique value for a given set of two nodes, the array ITAB needs only to be scanned in the region allocated to the smaller node for the existence of IHASH. If found, this indicates that the co-ordinates of the displacement nodes along its side have already been calculated. Each node is allocated a certain region. Regions of fixed size are allocated for different nodes. The region allocated to a node is scanned and all non-zero entries are compared with IHASH. A zero entry terminates the scan. If IHASH is not found, the location is used to enter the code for the element side, and the co-ordinates of nodes along its side are calculated. This technique is known as 'hashing' (Day, 1972).

Routine MIDSID

```
      SUBROUTINE MIDSID(IW6,MXND,NEL,LTAB,LDIM,NNU,NDIM,NNE,NPL,        MSID   1
     1 XYZ,NCONN,LTYP,MRELVV,NRELVV,NREL,ITAB,                          MSID   2
     2 NP1,NP2,ND,NN,KRD,NVTX,NDZ,MDZ)                                  MSID   3
C*************************************************************************MSID   4
C     GENERATES  MID-SIDE  NODES ALONG EDGE                              MSID   5
C*************************************************************************MSID   6
      DIMENSION XYZ(NDIM,NNE),NCONN(MXND,NEL),LTYP(NEL),                 MSID   7
     1 MRELVV(NEL),NRELVV(NNE),NREL(NNU),ITAB(LTAB,LDIM),               MSID   8
     1 NP1(NPL),NP2(NPL),KNDX(3)                                        MSID   9
      COMMON /DEBUGS/ ID1,ID2,ID3,ID4,ID5,ID6,ID7,ID8,ID9,ID10          MSID  10
      COMMON /ELINF/ LINFO(50,15)                                       MSID  11
      DATA KNDX(1),KNDX(2),KNDX(3)/8,11,5/                              MSID  12
C                                                                       MSID  13
      MDZ=0                                                             MSID  14
      KR=NDZ                                                            MSID  15
      K=NVTX                                                            MSID  16
      LDIM1=LDIM-1                                                      MSID  17
C                                                                       MSID  18
      CALL SETNP(NP1,NP2,NPL)                                           MSID  19
C                                                                       MSID  20
      WRITE(IW6,900)                                                    MSID  21
      DO 10 J=1,LDIM                                                    MSID  22
      DO 10 I=1,LTAB                                                    MSID  23
   10 ITAB(I,J)=0                                                       MSID  24
C                                                                       MSID  25
      DO 100 NE=1,NEL                                                   MSID  26
      MUS=MRELVV(NE)                                                    MSID  27
      LT=LTYP(NE)                                                       MSID  28
      NVN=LINFO(2,LT)                                                   MSID  29
      NEDG=LINFO(3,LT)                                                  MSID  30
      NDSD=LINFO(7,LT)                                                  MSID  31
      INDED=LINFO(14,LT)                                                MSID  32
C                                                                       MSID  33
      DO 26 IS=1,NEDG                                                   MSID  34
CC    WRITE(IW6,950)NE,IS                                               MSID  35
CC950 FORMAT(1X,9HELEMENT =,I5,2X,6HSIDE =,I5)                          MSID  36
      NL=(IS-1)*NDSD+NVN                                                MSID  37
      INDS=INDED+IS                                                     MSID  38
      IN1=NP1(INDS)                                                     MSID  39
      IN2=NP2(INDS)                                                     MSID  40
      N1=NCONN(IN1,NE)                                                  MSID  41
      N2=NCONN(IN2,NE)                                                  MSID  42
C                                                                       MSID  43
      CALL SORT2(N1,N2,I1,I2)                                           MSID  44
      IHASH=10000*I1+I2                                                 MSID   1
      IT=5*I1                                                           MSID   4
      GOTO 18                                                           MSID  47
C                                                                       MSID  48
   16 IT=IT+1                                                           MSID  49
   18 IF(IT.GT.LTAB) IT=1                                               MSID  50
      IF(ITAB(IT,1).EQ.IHASH) GOTO 24                                   MSID  51
      IF(ITAB(IT,1).NE.0) GOTO 16                                       MSID  52
C                                                                       MSID  53
      MDZ=MDZ+1                                                         MSID  54
      DO 22 IDSD=1,NDSD                                                 MSID  55
C-------------------------------------------------------------------------MSID  56
C     CALCULATE CO-ORDINATES OF NODES ALONG THE EDGE                    MSID  57
C-------------------------------------------------------------------------MSID  58
      K=K+1                                                             MSID  59
      KR=KR+1                                                           MSID  60
      IF(KR.LE.NNU)GOTO 19                                              MSID  61
      WRITE(IW6,901)                                                    MSID  62
      STOP                                                              MSID  63
C                                                                       MSID  64
```

```
   19 NREL(KR)=K                                                        MSID  65
      NRELVV(K)=KR                                                      MSID  66
      IF(K.LE.NNE) GOTO 20                                              MSID  67
      WRITE(IW6,902)NNE                                                 MSID  68
      STOP                                                              MSID  69
C                                                                       MSID  70
   20 NLN=NL+IDSD                                                       MSID  71
      NCONN(NLN,NE)=K                                                   MSID  72
      IPOS=IDSD+1                                                       MSID  73
      ITAB(IT,IPOS)=K                                                   MSID  74
      F1=FLOAT(NDSD+1-IDSD)/FLOAT(NDSD+1)                               MSID  75
      F2=1.-F1                                                          MSID  76
C                                                                       MSID  77
      DO 21 ID=1,NDIM                                                   MSID  78
   21 XYZ(ID,K)=XYZ(ID,N1)*F1+XYZ(ID,N2)*F2                             MSID  79
      WRITE(IW6,904)KR,(XYZ(ID,K),ID=1,NDIM)                            MSID  80
   22 CONTINUE                                                          MSID  81
      ITAB(IT,1)=IHASH                                                  MSID  82
C-----------------------------------------------------------------------MSID  83
C     FIRST ELEMENT ALONG EDGE HAS BEEN FOUND                           MSID  84
C-----------------------------------------------------------------------MSID  85
      ITAB(IT,LDIM1)=1                                                  MSID  86
C-----------------------------------------------------------------------MSID  87
C     COORDINATES OF NODES ALONG EDGE CALCULATED                        MSID  88
C     ASSUMING EDGE IS STRAIGHT                                         MSID  89
C-----------------------------------------------------------------------MSID  90
      ITAB(IT,LDIM)=1                                                   MSID  91
      GOTO 26                                                           MSID  92
C                                                                       MSID  93
   24 CONTINUE                                                          MSID  94
C                                                                       MSID  95
      DO 25 IDSD=1,NDSD                                                 MSID  96
      JDSD=NDSD+1-IDSD                                                  MSID  97
      NLJ=NL+JDSD                                                       MSID  98
   25 NCONN(NLJ,NE)=ITAB(IT,IDSD+1)                                     MSID  99
C-----------------------------------------------------------------------MSID 100
C     COUNT THE NUMBER OF ELEMENTS SHARING THIS EDGE                    MSID 101
C-----------------------------------------------------------------------MSID 102
      ITAB(IT,LDIM1)=ITAB(IT,LDIM1)+1                                   MSID 103
C                                                                       MSID 104
   26 CONTINUE                                                          MSID 105
C                                                                       MSID 106
      GO TO(90,90,90,90,90,27,27,90,90,90,90),LT                        MSID 107
      WRITE(IW6,920)MUS,LT                                              MSID 108
      STOP                                                              MSID 109
C-----------------------------------------------------------------------MSID 110
C     CALCULATE CO-ORDINATES OF NODES WITHIN ELEMENTS                   MSID 111
C-----------------------------------------------------------------------MSID 112
   27 NIND=LINFO(9,LT)                                                  MSID 113
      JLC=LINFO(5,LT)-NIND                                              MSID 114
C                                                                       MSID 115
      DO 30 INN=1,NIND                                                  MSID 116
      K=K+1                                                             MSID 117
      KR=KR+1                                                           MSID 118
      IF(K.GT.NNE)WRITE(IW6,902)NNE                                     MSID 119
      IF(KR.GT.NNU)WRITE(IW6,901)                                       MSID 120
      NREL(KR)=K                                                        MSID 121
      NRELVV(K)=KR                                                      MSID 122
      JLC=JLC+1                                                         MSID 123
      NCONN(JLC,NE)=K                                                   MSID 124
      INX1=INN                                                          MSID 125
      INX2=KNDX(INN)                                                    MSID 126
      NC=NCONN(INX1,NE)                                                 MSID 127
      NM=NCONN(INX2,NE)                                                 MSID 128
C                                                                       MSID 129
      DO 28 ID=1,NDIM                                                   MSID 130
```



```
   28 XYZ(ID,K)=0.5*(XYZ(ID,NC)+XYZ(ID,NM))                             MSID 131
      WRITE(IW6,904)KR,(XYZ(ID,K),ID=1,NDIM)                            MSID 132
C                                                                       MSID 133
   30 CONTINUE                                                          MSID 134
   90 CONTINUE                                                          MSID 135
  100 CONTINUE                                                          MSID 136
      IF(ID2.EQ.1)WRITE(IW6,910)ITAB                                    MSID 137
C-----------------------------------------------------------------------MSID 138
C     TOTAL NUMBER OF DISPLACEMENT NODES - ND                           MSID 139
C     MAXIMUM USER NO. OF DISPLACEMENT NODE - KRD                       MSID 140
C-----------------------------------------------------------------------MSID 141
      NN=K                                                              MSID 142
      ND=K                                                              MSID 143
      KRD=KR                                                            MSID 144
      RETURN                                                            MSID 145
  900 FORMAT(/10X,45HCOORDINATES OF DISPLACEMENT NODES ALONG EDGES//    MSID 146
     1 39H NODE          X           Y           Z/)                    MSID 147
  901 FORMAT(/1X,49HINCREASE NO. OF ADDITIONAL NODES (ROUTINE MIDSID))  MSID 148
  902 FORMAT(/1X,21H***ERROR*** MORE THAN,I5,                           MSID 149
     1 30HNODES IN MESH (ROUTINE MIDSID))                               MSID 150
  904 FORMAT(I5,3F12.3)                                                 MSID 151
  910 FORMAT(//1X,4HITAB/(1X,10I10))                                    MSID 152
  920 FORMAT(/1X,7HELEMENT,I5,2X,22HIS OF UNKNOWN TYPE ***,I5,2X,       MSID 153
     1 16H(ROUTINE MIDSID))                                             MSID 154
      END                                                               MSID 155
```

MSID 15 : KR – starting number of additional nodes.

MSID 16 : K – starting program node number of additional nodes.

MSID 19 : copy arrays NPL1, NPL2 to NP1, NP2 (NPL1, NPL2 are set using DATA statements. NP1, NP2 are allocated store dynamically in global array G. This procedure is adopted so that in case the size (NPL) of these arrays is changed, the changes that need to be carried out are minimal. Of course there is the duplication of data).

MSID 22–24 : zero array ITAB.

MSID 26 : loop on all elements.

MSID 28–32 : obtain element particulars.
NVN – no. of vertex nodes.
NEDG – no. of element sides.
NDSD – no. of additional (displacement) nodes along edge.
INDED – starting index to arrays NP1, NP2.

MSID 34 : loop on all edges of the element.

MSID 37 : index to location of node in NCONN.

MSID 38 : index to nodes at either end of element side, in NP1 and NP2.

MSID 39–40 : indexes of nodes at either end in NCONN.

MSID 41–42 : nodes at either end of element side.

MSID 44 : sort the nodes into ascending order.

MSID 45 : code for element side (consisting of node numbers at either end).

MSID 50 : start at the beginning if end of array has been reached, and make use of the gaps in array ITAB.

MSID 51 : look for the possibility that nodes along element edge have already been numbered; if so, branch off.

MSID 52 : if nodes along element edge have to be numbered then find a location with zero entry.
MSID 55 : such a location has been found. Loop on all additional (displacement) nodes along this edge.
MSID 59 : program number for the new node.
MSID 60 : user number for new node.
MSID 61–62 : check that no. of nodes does not exceed allocation for array NREL. If exceeded, print error message and stop. (The allocation for NREL is such that it ought to be more than what is required, and hence this should not happen.)
MSID 65 : enter program node number in array NREL.
MSID 66 : enter user node number in cross-reference array NRELVV.
MSID 67–68 : check that array allocation NRELVV is not exceeded.
MSID 71 : index of new node in array NCONN.
MSID 72 : enter new node number in NCONN.
MSID 73–74 : index of new node in array ITAB, and enter new node no.
MSID 75–76 : calculate interpolation ratios.
MSID 78–79 : calculate co-ordinates of new node, using linear interpolation on nodes at either end.
MSID 81 : end of loop on nodes along edge.
MSID 82 : enter code representing element side in ITAB.
MSID 86 : enter 1 indicate that nodes along element edge have been calculated (the value is also used to count the number of elements shared by this side).
MSID 91 : code to indicate co-ordinates along edge have been calculated assuming the edges are straight.
MSID 96 : for any element edge along which nodal co-ordinates have already been calculated. Loop on all nodes along edge excluding the ones at either end.
MSID 97–98 : indexes to positions of nodes in NCONN and ITAB.
MSID 99 : enter the node numbers in NCONN.
MSID 103 : increment counter of elements sharing edge by one.
MSID 105 : end of loop on all element edges.
MSID 113 : number of inner nodes (only for element types which have them, i.e. skip for the rest).
MSID 114 : index to node location in NCONN.
MSID 116 : loop on all inner nodes.
MSID 117 : program node number.
MSID 118 : user node number.
MSID 119–120 : check for array sizes NREL, NRELVV being exceeded.
MSID 121–122 : enter node numbers in NREL and NRELVV.
MSID 123–124 : enter number in NCONN.
MSID 125–126 : indexes to nodes of element used in interpolating co-ordinates of inner nodes.†
MSID 127–128 : node numbers (used for interpolation).†

MSID 130–131 : calculate co-ordinates of inner nodes.†
MSID 136 : end of element loop.
MSID 137 : print out array ITAB for debugging.
MSID 142–144 : maximum values of displacement node numbers. (KRD – user number; ND, NN – program number.)

In the above routine, IHASH = N1 * 10000 + N2, where N1 < N2. Routine **SORT2** sorts the nodes at either end of each element side.

Routine SORT2

```
      SUBROUTINE SORT2(N1,N2,I1,I2)                                     SORT   1
C*****************************************************************SORT   2
C     ROUTINE TO SORT TWO INTEGERS. I1 IS LESS THAN I2                SORT   3
C*****************************************************************SORT   4
      I1=N1                                                             SORT   5
      I2=N2                                                             SORT   6
      IF(I1.LT.I2)RETURN                                                SORT   7
      I1=N2                                                             SORT   8
      I2=N1                                                             SORT   9
      RETURN                                                            SORT  10
      END                                                               SORT  11
```

Sort 5–9: sort two nodes; assign I1 to the lower node number.

For a triangular element there are three sides. The nodes at either end of each side have the following *indexes*:

1 2 – side 1
2 3 – side 2
3 1 – side 3

Arrays NP1, NP2, NPL1, NPL2 are indexes to array NCONN, and give the indexes to the nodes at either end of the element sides. For element type 2 the values are

NP1 (1)	NP1 (2)	NP1 (3)	NP2 (1)	NP2 (2)	NP2 (3)
1	2	3	2	3	1

For element types 2, 3, 6 and 7, these indexes are the same. These are entered in NP1(1)–NP1(3), NP2(1)–NP2(3). Since all the relevant information is placed in a single array, each element type needs a starting index (INDED); therefore

INDED = 0 for element types 2, 3, 6 and 7.

INDED is obtained from array LINFO(50, 15)

→ element types
→ element parameters

† Note: these are specifically for element types 6 and 7 and are currently the only element types with inner nodes. Any new element type with inner nodes will require this part of the code to be modified.

$$INDED = LINFO(14, LT)$$

where LT = 2 the element type number. The contents of array LINFO are explained in section 6.7.

Routine **SETNP** sets up the arrays NP1 and NP2 for all element types.

Routine SETNP

```
      SUBROUTINE SETNP(NP1,NP2,NPL)                                     STNP   1
C*********************************************************************STNP   2
C     SET UP ARRAYS NP1 AND NP2 WHICH GIVE THE INDEX TO ARRAY           STNP   3
C     NCONN FOR NODES AT EITHER END OF EACH ELEMENT EDGE                STNP   4
C*********************************************************************STNP   5
      DIMENSION NPL1(21),NPL2(21),NP1(NPL),NP2(NPL)                     STNP   6
C---------------------------------------------------------------------STNP   7
C              INDEXES OF ARRAYS NPL1,NPL2,NP1,NP2                      STNP   8
C          INDEX            ELEMENT TYPE                              STNP   9
C          1 -  3            1, 2, 3, 6, 7                            STNP  10
C          4 -  7            4, 5                                     STNP  11
C          4 - 15            8, 9                                     STNP  12
C         16 - 21           10,11                                     STNP  13
C---------------------------------------------------------------------STNP  14
      DATA NPL1(1),NPL1(2),NPL1(3),NPL1(4),NPL1(5),NPL1(6),NPL1(7),     STNP  15
     1 NPL1(8),NPL1(9),NPL1(10),NPL1(11),NPL1(12),NPL1(13),NPL1(14),    STNP  16
     2 NPL1(15),NPL1(16),NPL1(17),NPL1(18),NPL1(19),NPL1(20),NPL1(21)/  STNP  17
     3 1,2,3,1,2,3,4,5,6,7,8,1,2,3,4,1,2,3,1,2,3/                       STNP  18
      DATA NPL2(1),NPL2(2),NPL2(3),NPL2(4),NPL2(5),NPL2(6),NPL2(7),     STNP  19
     1 NPL2(8),NPL2(9),NPL2(10),NPL2(11),NPL2(12),NPL2(13),NPL2(14),    STNP  20
     2 NPL2(15),NPL2(16),NPL2(17),NPL2(18),NPL2(19),NPL2(20),NPL2(21)/  STNP  21
     3 2,3,1,2,3,4,1,6,7,8,5,5,6,7,8,2,3,1,4,4,4/                       STNP  22
C                                                                       STNP  23
      DO 10 I=1,NPL                                                     STNP  24
      NP1(I)=NPL1(I)                                                    STNP  25
   10 NP2(I)=NPL2(I)                                                    STNP  26
C                                                                       STNP  27
      RETURN                                                            STNP  28
      END                                                               STNP  29
```

STNP 24–26 : set arrays NP1, NP2 equal to arrays NPL1, NPL2 respectively.

The normal procedure is to use a mesh with elements having straight sides. Sometimes, however, element sides have to be curved in order to properly describe the problem being analysed, e.g. circular tunnels or buried pipes. The simplest option is provided whereby the user specifies the list of element sides and the co-ordinates of the nodes which lie along the curved sides (in the case of linear strain elements, this is just one). Remembering that routine **MIDSID** has already numbered these nodes and calculated their co-ordinates, it is a simple matter to identify these nodes and replace their co-ordinates by the ones provided by the user. It is achieved in routine **CUREDG**.

Routine CUREDG

```
      SUBROUTINE CUREDG(IR5,IW6,MXND,NEL,NDIM,NNE,LTAB,LDIM,            CURE   1
     1 MUMAX,NNU,NPL,XYZ,NCONN,LTYP,MREL,NREL,ITAB,                     CURE   2
     2 NP1,NP2,NCRED,NDTY,NMX)                                          CURE   3
C*********************************************************************CURE   4
C     ROUTINE TO READ NODAL COORDINATES ALONG CURVED EDGES              CURE   5
C*********************************************************************CURE   6
```



```
      DIMENSION XYZ(NDIM,NNE),NCONN(MXND,NEL),LTYP(NEL),MREL(MUMAX),    CURE   7
     1 NREL(NNU),ITAB(LTAB,LDIM),NP1(NPL),NP2(NPL),CD(3,3),CDT(3,3)     CURE   8
      COMMON /ELINF / LINFO(50,15)                                      CURE   9
C                                                                       CURE  10
      IERS=0                                                            CURE  11
C                                                                       CURE  12
      WRITE(IW6,900)                                                    CURE  13
      DO 200 ISD=1,NCRED                                                CURE  14
      READ(IR5,*)MU,ND1,ND2,((CD(IU,JU),IU=1,NDIM),JU=1,NMX)            CURE  15
      WRITE(IW6,902)MU,ND1,ND2,((CD(IU,JU),IU=1,NDIM),JU=1,NMX)         CURE  16
C                                                                       CURE  17
      MPR=MREL(MU)                                                      CURE  18
      LT=LTYP(MPR)                                                      CURE  19
      NDN=LINFO(5,LT)                                                   CURE  20
      NVN=LINFO(2,LT)                                                   CURE  21
      NEDG=LINFO(3,LT)                                                  CURE  22
      NDSD=LINFO(7,LT)                                                  CURE  23
      IF(NDTY.EQ.2)NDSD=LINFO(8,LT)                                     CURE  24
      IF(NDSD.GT.0)GOTO 5                                               CURE  25
      WRITE(IW6,903)MU,NDTY                                             CURE  26
      GOTO 200                                                          CURE  27
C                                                                       CURE  28
    5 INDED=LINFO(14,LT)                                                CURE  29
C                                                                       CURE  30
      K1=NREL(ND1)                                                      CURE  31
      K2=NREL(ND2)                                                      CURE  32
C                                                                       CURE  33
      CALL SORT2(K1,K2,I1,I2)                                           CURE  34
      IHASH=10000*I1+I2                                                 CURE  35
      IT=5*I1                                                           CURE  36
      GOTO 8                                                            CURE  37
C                                                                       CURE  38
    6 IT=IT+1                                                           CURE  39
    8 IF(IT.GT.LTAB)IT=1                                                CURE  40
      IF(ITAB(IT,1).EQ.IHASH)GOTO 10                                    CURE  41
      IF(ITAB(IT,1).NE.0)GOTO 6                                         CURE  42
C                                                                       CURE  43
C *** EDGE NOT FOUND                                                    CURE  44
      IERS=IERS+1                                                       CURE  45
      WRITE(IW6,904)ND1,ND2                                             CURE  46
      GO TO 200                                                         CURE  47
C                                                                       CURE  48
C *** NOTE EDGE IS CURVED - FOR PLOTTING PURPOSES                       CURE  49
   10 IF(NDTY.EQ.2)GOTO 11                                              CURE  50
      ITAB(IT,LDIM)=2                                                   CURE  51
C                                                                       CURE  52
   11 DO 20 IEDG=1,NEDG                                                 CURE  53
      INDS=INDED+IEDG                                                   CURE  54
      IN1=NP1(INDS)                                                     CURE  55
      IN2=NP2(INDS)                                                     CURE  56
      N1=NCONN(IN1,MPR)                                                 CURE  57
      N2=NCONN(IN2,MPR)                                                 CURE  58
C                                                                       CURE  59
      IF(K1.EQ.N1.AND.K2.EQ.N2)GOTO 26                                  CURE  60
      IF(K2.EQ.N1.AND.K1.EQ.N2)GOTO 22                                  CURE  61
   20 CONTINUE                                                          CURE  62
C                                                                       CURE  63
      WRITE(IW6,908)MU,ND1,ND2                                          CURE  64
      GOTO 200                                                          CURE  65
C---------------------------------------------------------------------CURE  66
C     CHANGE AROUND COORDINATES IF THERE ARE MORE THAN                  CURE  67
C     ONE NODE AND THE NODES ARE IN THE REVERSE ORDER                   CURE  68
C---------------------------------------------------------------------CURE  69
   22 IF(NEDG.LE.1)GOTO 26                                              CURE  70
C                                                                       CURE  71
      DO 24 IDSD=1,NDSD                                                 CURE  72
```

```
      JBK=NDSD+1-IDSD                                                  CURE  73
      DO 24 ID=1,NDIM                                                  CURE  74
   24 CDT(ID,IDSD)=CD(ID,JBK)                                          CURE  75
C                                                                      CURE  76
      DO 25 IDSD=1,NDSD                                                CURE  77
      DO 25 ID=1,NDIM                                                  CURE  78
   25 CD(ID,IDSD)=CDT(ID,IDSD)                                         CURE  79
C                                                                      CURE  80
   26 CONTINUE                                                         CURE  81
      NS=NVN                                                           CURE  82
      IF(NDTY.EQ.2)NS=NDN                                              CURE  83
      NL=NS+(IEDG-1)*NDSD                                              CURE  84
C-----------------------------------------------------------------------CURE  85
C     CHANGE COORDINATES ALONG CURVED EDGE                             CURE  86
C-----------------------------------------------------------------------CURE  87
      DO 40 KSD=1,NDSD                                                 CURE  88
      NLN=NL+KSD                                                       CURE  89
      K=NCONN(NLN,MPR)                                                 CURE  90
C                                                                      CURE  91
      DO 38 ID=1,NDIM                                                  CURE  92
   38 XYZ(ID,K)=CD(ID,KSD)                                             CURE  93
   40 CONTINUE                                                         CURE  94
C                                                                      CURE  95
  200 CONTINUE                                                         CURE  96
C                                                                      CURE  97
      IF(IERS.EQ.0)RETURN                                              CURE  98
      WRITE(IW6,910)                                                   CURE  99
      STOP                                                             CURE 100
  900 FORMAT(/1X,32HLIST OF NODES ALONG CURVED EDGES/)                 CURE 101
  902 FORMAT(3I5,6F10.0)                                               CURE 102
  903 FORMAT(1X,7HELEMENT,I5,2X,18HDOES NOT HAVE TYPE,I4,2X,           CURE 103
     1 33HNODES ALONG SIDE (ROUTINE CUREDG))                           CURE 104
  904 FORMAT(/1X,32H***ERROR** EDGE CONTAINING NODES,2I5,2X,           CURE 105
     1 9HNOT FOUND)                                                    CURE 106
  908 FORMAT(/1X,7HELEMENT,I5,23H DOES NOT CONTAIN NODES,2I5)          CURE 107
  910 FORMAT(/1X,36HPROGRAM TERMINATED IN ROUTINE CUREDG)              CURE 108
      END                                                              CURE 109
```

CURE 11 : set error count to zero.
CURE 14 : loop on all elements sides which are curved.
CURE 15–16 : read and write co-ordinates of nodes along curved sides (excluding nodes at either end).
CURE 18–29 : data dependent on element type.
- MPR – program element number.
- LT – element type number.
- NDN – total number of displacement nodes in element.
- NVN – number of vertex nodes.
- NEDG – number of element sides.
- INDED – starting index to arrays NP1, NP2.

CURE 23 : NDSD – the number of **displacement** nodes along side (excluding nodes at either end).
CURE 24 : NDSD – the number of **pore pressure** nodes along side (excluding nodes at either end).
CURE 31–32 : program node numbers of nodes at either end.
CURE 34 : sort the numbers: I1 is the smaller of the two.
CURE 35 : IHASH – code to identify element side.

CURE 40 : start from the beginning, if end of array ITAB has been reached (the allocation for ITAB is more than is actually required).
CURE 41 : the entry for the element side has been found.
CURE 42 : look for zero entry.
CURE 45–47 : IHASH – entry for element edge has not been found (unlikely program error) – print out error message.
CURE 50–51 : make entry to indicate that the side is curved for plotting purposes. It is by-passed if pore pressure nodes are being numbered.
CURE 53 : loop on all edges of element to find the side which is curved.
CURE 54 : index for arrays NP1, NP2 for a given edge of a given elemen type.
CURE 55–56 : indexes to array NCONN, i.e. local node numbers.
CURE 57–58 : nodes at either end of edge.
CURE 60 : branch off if nodes match, i.e. they are in the correct anti-clockwise order.
CURE 61 : nodes match after being interchanged.
CURE 64–65 : the edge (identified by nodes at either end) specified by user cannot be found in element (probable user error).
CURE 70 : branch off if the edge contains only one side node.
CURE 72–75 : array CDT contains the rearranged node co-ordinates.
CURE 77–79 : array CD contains the nodal co-ordinates in the correct (anti-clockwise) sequence.
CURE 82 : index to local (displacement) node numbers.
CURE 83 : index to local (pore pressure) node numbers.
CURE 84 : index to local (displacement/pore pressure) node numbers.
CURE 88 : loop on all nodes along edge (excluding end nodes).
CURE 89–90 : NLN is index (local node no.) and K is the node number.
CURE 92–93 : replace the nodal co-ordinates.
CURE 96 : end of loop on all curved sides.
CURE 98–99 : if errors have been detected, print message and stop.

Routine **INTPLT** scans the co-ordinates of all the displacement nodes and establishes the size (extent) of the mesh. This is the first information (the minimum and maximum values of the co-ordinates) written to the **plot data** (PD) file, and it is used by a separate mesh-plotting program to calculate the appropriate scale for plotting the mesh.

Routine INTPLT

```
      SUBROUTINE INTPLT(IW6,IW8,NDIM,NNE,XYZ,ND)                       IPLT   1
C*********************************************************************IPLT   2
C     ROUTINE TO CALCULATE DIMENSIONS OF THE PLOT                      IPLT   3
C*********************************************************************IPLT   4
      DIMENSION XYZ(NDIM,NNE),CODMIN(3),CODMAX(3)                      IPLT   5
      COMMON /PARS  / PYI,ALAR,ASMVL,ZERO                              IPLT   6
C                                                                      IPLT   7
```

```
      DO 10 ID=1,NDIM                                                    IPLT   8
      CODMIN(ID)= ALAR                                                   IPLT   9
   10 CODMAX(ID)=-ALAR                                                   IPLT  10
C                                                                        IPLT  11
      DO 30 J=1,ND                                                       IPLT  12
      DO 20 ID=1,NDIM                                                    IPLT  13
      IF(XYZ(ID,J).GT.CODMAX(ID))CODMAX(ID)=XYZ(ID,J)                    IPLT  14
      IF(XYZ(ID,J).LT.CODMIN(ID))CODMIN(ID)=XYZ(ID,J)                    IPLT  15
   20 CONTINUE                                                           IPLT  16
   30 CONTINUE                                                           IPLT  17
C                                                                        IPLT  18
      WRITE(IW8)NDIM                                                     IPLT  19
      WRITE(IW8)(CODMAX(ID),ID=1,NDIM),(CODMIN(ID),ID=1,NDIM)            IPLT  20
      RETURN                                                             IPLT  21
      END                                                                IPLT  22
```

IPLT 8–10 : initialise the minimum and maximum values of co-ordinates to appropriate values.
IPLT 12 : loop on all displacement nodes.
IPLT 13–15 : store the minimum and maximum values of nodal co-ordinates.
IPLT 17 : end of displacement node loop.
IPLT 19–20 : write the minimum and maximum nodal co-ordinates to a file for plotting later (using the mesh-plotting program).

The data necessary to draw the mesh (i.e. by means of drawing all the element sides) and numbering the nodes and the elements are also written to the PD file in a standard format. This also applies to instructions such as change of pen colour used for plotting. The standard format consists of a set of co-ordinates and two integer codes. Two such entries are needed to draw an element side.

Routine SIDES

```
      SUBROUTINE SIDES(IW6,IW8,LTAB,LDIM,NDIM,NNE,MXND,NEL,              SIDE   1
     1 XYZ,NCONN,ITAB)                                                   SIDE   2
C*********************************************************************SIDE   3
C     PLOTS MESH                                                         SIDE   4
C*********************************************************************SIDE   5
      DIMENSION XYZ(NDIM,NNE),NCONN(MXND,NEL),ITAB(LTAB,LDIM),XYZD(3)    SIDE   6
C-------------------------------------------------------------------SIDE   7
C     LOOP ON ALL EDGES                                                  SIDE   8
C-------------------------------------------------------------------SIDE   9
      NSD=LDIM-3                                                         SIDE  10
C-------------------------------------------------------------------SIDE  11
C     PEN MOVEMENT : 3 - MOVETO : 1 - DRAWTO                             SIDE  12
C-------------------------------------------------------------------SIDE  13
      IONE=1                                                             SIDE  14
      ITHR=3                                                             SIDE  15
      IDUM=0                                                             SIDE  16
C-------------------------------------------------------------------SIDE  17
C     DUMMY COORDINATES                                                  SIDE  18
C-------------------------------------------------------------------SIDE  19
      DO 5 ID=1,NDIM                                                     SIDE  20
    5 XYZD(ID)=0.                                                        SIDE  21
C-------------------------------------------------------------------SIDE  22
C     PEN COLOUR IS BLACK FOR DRAWING MESH                               SIDE  23
C-------------------------------------------------------------------SIDE  24
      ICODE=-1                                                           SIDE  25
      WRITE(IW8)ICODE,(XYZD(ID),ID=1,NDIM),IDUM                          SIDE  26
```



```
C                                                                        SIDE  27
      DO 20 L=1,LTAB                                                     SIDE  28
      IF(ITAB(L,1).EQ.0)GOTO 20                                          SIDE  29
      N1=ITAB(L,1)/10000                                                 SIDE  30
      N2=ITAB(L,1)-N1*10000                                              SIDE  31
      WRITE(IW8)ITHR,(XYZ(ID,N1),ID=1,NDIM),IDUM                         SIDE  32
      IF(ITAB(L,LDIM).NE.2)GO TO 15                                      SIDE  33
C-------------------------------------------------------------------SIDE  34
C     DRAW CURVED SIDE - USING STRAIGHT LINES PASSING                    SIDE  35
C     THROUGH ALL DISPLACEMENT NODES                                     SIDE  36
C-------------------------------------------------------------------SIDE  37
      DO 10 ISD=1,NSD                                                    SIDE  38
      ND=ITAB(L,ISD+1)                                                   SIDE  39
   10 WRITE(IW8)IONE,(XYZ(ID,ND),ID=1,NDIM),IDUM                         SIDE  40
   15 WRITE(IW8)IONE,(XYZ(ID,N2),ID=1,NDIM),IDUM                         SIDE  41
   20 CONTINUE                                                           SIDE  42
      RETURN                                                             SIDE  43
      END                                                                SIDE  44
```

SIDE 10 : no. of nodes along an element edge (excluding end nodes).
SIDE 14–15 : codes which control pen movements.
SIDE 20–21 : dummy co-ordinates (used when pen colour is changed).
SIDE 25–26 : select pen colour (when negative, change pen colour). 1 – black; 2 – red; 3 – green. Write details to PD file.
SIDE 28 : loop on all entries of array ITAB (each entry represents an element side).
SIDE 29 : branch off if zero entry (non-zero entry indicates an element side).
SIDE 30–31 : nodes on either end of edge.
SIDE 32 : write co-ordinates to plot data (PD) file.
SIDE 33 : branch off if element edge is straight.
SIDE 38–41 : write intermediate (along element edge) node co-ordinates to PD file.
SIDE 42 : end of loop on ITAB entires.

6.5 NUMBERING THE ADDITIONAL PORE PRESSURE NODES

Now the procedure for numbering the additional displacement nodes is repeated for numbering the additional pore pressure nodes along element sides and element interiors. This is done in routine **MIDPOR**, which is very similar to the routine MIDSID. Only the higher-order (CuST) element uses this routine. All linear strain elements have pore pressure variables at the vertex nodes, which give a linear variation in pore pressure. These elements do not have additional pore pressure nodes.

This routine repeats the procedure (as in routine MIDSID) for the whole mesh, only this time the additional nodes are pore pressure nodes instead of displacement nodes.

Routine MIDPOR

```
      SUBROUTINE MIDPOR(IW6,MXND,NEL,LTAB,LDIM,NNU,NDIM,NNE,NPL,        MPOR   1
     1 XYZ,NCONN,LTYP,MRELVV,NRELVV,NREL,ITAB,NP1,NP2,NN,KRD,NNZ)       MPOR   2
C*************************************************************************MPOR   3
C     ROUTINE TO CALCULATE ADDITIONAL PORE-PRESSURE NODES FOR             MPOR   4
C     CONSOLIDATION ELEMENTS ( NODES WITH ONLY                            MPOR   5
C     EXCESS PORE PRESSURES AS VARIABLES)                                 MPOR   6
C*************************************************************************MPOR   7
      DIMENSION XYZ(NDIM,NNE),NCONN(MXND,NEL),LTYP(NEL),                  MPOR   8
     1 MRELVV(NEL),NRELVV(NNE),NREL(NNU),ITAB(LTAB,LDIM),                 MPOR   9
     1 NP1(NPL),NP2(NPL),SUM(3)                                           MPOR  10
      COMMON /ELINF / LINFO(50,15)                                        MPOR  11
      COMMON /PARS  / PYI,ALAR,ASMVL,ZERO                                 MPOR  12
C                                                                         MPOR  13
      KR=KRD                                                              MPOR  14
      K=NN                                                                MPOR  15
      LDIM1=LDIM-1                                                        MPOR  16
C                                                                         MPOR  17
      LT=LTYP(1)                                                          MPOR  18
      IF(LINFO(8,LT).NE.0)WRITE(IW6,900)                                  MPOR  19
C                                                                         MPOR  20
      DO 10 J=1,LDIM                                                      MPOR  21
      DO 10 I=1,LTAB                                                      MPOR  22
   10 ITAB(I,J)=0                                                         MPOR  23
C                                                                         MPOR  24
      DO 100 NE=1,NEL                                                     MPOR  25
      MUS=MRELVV(NE)                                                      MPOR  26
      LT=LTYP(NE)                                                         MPOR  27
      GOTO(100,100,100,100,100,100,12,100,100,100,100),LT                MPOR  28
      WRITE(IW6,910)MUS,LT                                                MPOR  29
      STOP                                                                MPOR  30
   12 NDN=LINFO(5,LT)                                                     MPOR  31
      NVN=LINFO(2,LT)                                                     MPOR  32
      NEDG=LINFO(3,LT)                                                    MPOR  33
      NDPT=LINFO(1,LT)                                                    MPOR  34
      INDED=LINFO(14,LT)                                                  MPOR  35
      NPSD=LINFO(8,LT)                                                    MPOR  36
C                                                                         MPOR  37
      DO 26 IS=1,NEDG                                                     MPOR  38
      NLP=NDN+(IS-1)*NPSD                                                 MPOR  39
      INDS=INDED+IS                                                       MPOR  40
      IN1=NP1(INDS)                                                       MPOR  41
      IN2=NP2(INDS)                                                       MPOR  42
      N1=NCONN(IN1,NE)                                                    MPOR  43
      N2=NCONN(IN2,NE)                                                    MPOR  44
C                                                                         MPOR  45
      CALL SORT2(N1,N2,I1,I2)                                             MPOR  46
      IHASH=10000*I1+I2                                                   MPOR  47
      IT=5*I1                                                             MPOR  48
      GOTO 18                                                             MPOR  49
C                                                                         MPOR  50
   16 IT=IT+1                                                             MPOR  51
   18 IF(IT.GT.LTAB) IT=1                                                 MPOR  52
      IF(ITAB(IT,1).EQ.IHASH) GOTO 24                                     MPOR  53
      IF(ITAB(IT,1).NE.0) GOTO 16                                         MPOR  54
C                                                                         MPOR  55
      DO 22 IPSD=1,NPSD                                                   MPOR  56
C-------------------------------------------------------------------------MPOR  57
C     CALCULATE CO-ORDINATES OF NODES ALONG THE EDGE                      MPOR  58
C-------------------------------------------------------------------------MPOR  59
      K=K+1                                                               MPOR  60
      KR=KR+1                                                             MPOR  61
      IF(KR.LE.NNU)GOTO 19                                                MPOR  62
      WRITE(IW6,901)                                                      MPOR  63
      STOP                                                                MPOR  64
```



```
C                                                                         MPOR  65
   19 NREL(KR)=K                                                          MPOR  66
      NRELVV(K)=KR                                                        MPOR  67
      IF(K.LE.NNE) GOTO 20                                                MPOR  68
      WRITE(IW6,902)NNE                                                   MPOR  69
      STOP                                                                MPOR  70
C                                                                         MPOR  71
   20 NLNP=NLP+IPSD                                                       MPOR  72
      NCONN(NLNP,NE)=K                                                    MPOR  73
      IPOS=IPSD+1                                                         MPOR  74
      ITAB(IT,IPOS)=K                                                     MPOR  75
      F1=FLOAT(NPSD+1-IPSD)/FLOAT(NPSD+1)                                 MPOR  76
      F2=1.-F1                                                            MPOR  77
C                                                                         MPOR  78
      DO 21 ID=1,NDIM                                                     MPOR
   21 XYZ(ID,K)=XYZ(ID,N1)*F1+XYZ(ID,N2)*F2                               MPOR
      WRITE(IW6,904)KR,(XYZ(ID,K),ID=1,NDIM)                              MPOR  81
   22 CONTINUE                                                            MPOR  82
C                                                                         MPOR  83
      ITAB(IT,1)=IHASH                                                    MPOR  84
      ITAB(IT,LDIM1)=1                                                    MPOR  85
      GOTO 26                                                             MPOR  86
C                                                                         MPOR  87
   24 DO 25 IPSD=1,NPSD                                                   MPOR  88
      JPSD=NPSD+1-IPSD                                                    MPOR  89
      NLPJ=NLP+JPSD                                                       MPOR  90
   25 NCONN(NLPJ,NE)=ITAB(IT,IPSD+1)                                      MPOR  91
C                                                                         MPOR  92
      ITAB(IT,LDIM1)=ITAB(IT,LDIM1)+1                                     MPOR  93
C                                                                         MPOR  94
   26 CONTINUE                                                            MPOR  95
C                                                                         MPOR  96
      GO TO(90,90,90,90,90,90,27,90,90,90,90),LT                          MPOR  97
C-------------------------------------------------------------------------MPOR  98
C     CALCULATE CO-ORDINATES OF NODES WITHIN ELEMENTS                     MPOR  99
C-------------------------------------------------------------------------MPOR 100
   27 NINP=LINFO(10,LT)                                                   MPOR 101
      JP=NDPT-NINP                                                        MPOR 102
C                                                                         MPOR 103
      DO 80 INP=1,NINP                                                    MPOR 104
      K=K+1                                                               MPOR 105
      KR=KR+1                                                             MPOR 106
      IF(KR.GT.NNU)WRITE(IW6,901)                                         MPOR 107
      IF(K.GT.NNE)WRITE(IW6,902)NNE                                       MPOR 108
      NREL(KR)=K                                                          MPOR 109
      NRELVV(K)=KR                                                        MPOR 1
      JP=JP+1                                                             MPOR
      NCONN(JP,NE)=K                                                      MPOR 1
C                                                                         MPOR 113
      DO 40 ID=1,NDIM                                                     MPOR 114
   40 SUM(ID)=ZERO                                                        MPOR 115
C                                                                         MPOR 116
      DO 50 IN=1,NVN                                                      MPOR 117
      NDE=NCONN(IN,NE)                                                    MPOR 118
      DO 50 ID=1,NDIM                                                     MPOR 119
   50 SUM(ID)=SUM(ID)+XYZ(ID,NDE)                                         MPOR 120
C                                                                         MPOR 121
      DO 60 ID=1,NDIM                                                     MPOR 122
   60 XYZ(ID,K)=SUM(ID)/FLOAT(NVN)                                        MPOR 123
      WRITE(IW6,904)KR,(XYZ(ID,K),ID=1,NDIM)                              MPOR 124
C                                                                         MPOR 125
   80 CONTINUE                                                            MPOR 126
C                                                                         MPOR 127
   90 CONTINUE                                                            MPOR 128
C                                                                         MPOR 129
  100 CONTINUE                                                            MPOR 130
```

```
C                                                                       MPOR 131
      NN=K                                                              MPOR 132
      NNZ=KR                                                            MPOR 133
      RETURN                                                            MPOR 134
  900 FORMAT(/10X,46HCOORDINATES OF PORE PRESSURE NODES ALONG EDGES//   MPOR 135
     1 39H NODE          X           Y           Z/)                    MPOR 136
  901 FORMAT(/1X,49HINCREASE NO. OF ADDITIONAL NODES (ROUTINE MIDPOR))  MPOR 137
  902 FORMAT(/1X,21H***ERROR*** MORE THAN,I5,                           MPOR 138
     1 30HNODES IN MESH (ROUTINE MIDPOR))                               MPOR 139
  904 FORMAT(I5,3F12.3)                                                 MPOR 140
  910 FORMAT(/1X,7HELEMENT,I5,2X,22HIS OF UNKNOWN TYPE ***,I5,2X,       MPOR 141
     1 16H(ROUTINE MIDPOR))                                             MPOR 142
      END                                                               MPOR 143
```

MPOR 14 : KR – starting **user** number of additional nodes.
MPOR 15 : K – starting **program** number of additional nodes.
MPOR 18–19 : write title – 'co-ordinates of pore pressure nodes'.
MPOR 21–23 : zero array ITAB.
MPOR 25 : loop on all elements.
MPOR 28 : skip if element type has no additional pore pressure nodes.
MPOR 31–36 : obtain element particulars.
- NVN – number of vertex nodes.
- NEDG – number of element sides.
- NDPT – total number of nodes.
- INDED – starting index to arrays NP1, NP2.
- NPSD – number of additional (pore pressure) nodes along edge.

MPOR 38 : loop on all edges of the element.
MPOR 39 : index to location of new node in NCONN.
MPOR 40 : index to nodes at either end of element side, in NP1, NP2.
MPOR 41–42 : indexes of nodes at either end in NCONN.
MPOR 43–44 : nodes at either end of element side.
MPOR 46 : sort the nodes into ascending order.
MPOR 47 : calculate unique code (consisting of node numbers at either end) representing the side.
MPOR 52 : start at the beginning, if end of array has been reached, and make use of the gaps in array ITAB.
MPOR 53 : look for the possibility that nodes along element edge have already been numbered; if so, branch off.
MPOR 54 : if nodes along element edge have to be numbered then find a location with zero entry.
MPOR 56 : such a location has been found. Loop on all additional (pore pressure) nodes along this edge.
MPOR 60 : program number for the new node.
MPOR 61 : user number for the new node.
MPOR 62–63 : check that number of nodes does not exceed allocation for array NREL. If exceeded, print error message and stop. (The allocation for NREL is such that this should not happen.)

MPOR 66 : enter program node number in array NREL.
MPOR 67 : enter user node number in cross-reference array NRELVV.
MPOR 68–69 : check that array allocation NRELVV is not exceeded.
MPOR 72 : index of new node in array NCONN.
MPOR 73 : enter new node number in NCONN.
MPOR 74–75 : index of new node in array ITAB, and enter new no.
MPOR 76–77 : calculate interpolation ratios.
MPOR 79–81 : calculate co-ordinates of new node, using linear interpolation on nodes at either end.
MPOR 82 : end of loop on nodes along edge.
MPOR 84 : enter code representing element side in ITAB.
MPOR 85 : enter 1 to indicate that nodes along element edge have been calculated (the value is also used to count the number of elements sharing this side).
MPOR 88 : for any element edge along which nodal co-ordinates have already been calculated. Loop on all nodes along edges excluding the ones at either end.
MPOR 89–91 : enter the node numbers in NCONN.
MPOR 93 : increment count on no. of elements sharing element side.
MPOR 95 : end of loop on all element edges.
MPOR 97–101† : no. of inner nodes (only for element types which have them; skip for the rest).
MPOR 102† : index to node location in NCONN.
MPOR 104† : loop on all inner nodes.
MPOR 105† : program node number.
MPOR 106† : user node number.
MPOR 107–108† : check for array sizes NREL, NRELVV being exceeded.
MPOR 109–110† : enter node number in NREL and NRELVV.
MPOR 111–112† : enter number in NCONN.
MPOR 114–123† : calculate co-ordinates of inner node.
MPOR 126† : end of loop on all inner nodes.
MPOR 130 : end of element loop.
MPOR 132–133 : maximum values of node numbers (all inclusive). (NNZ – user number; NN – program number.)

If the pore pressure nodes lie along a curved side (here again only relevant to CuST element) then the user again provides the co-ordinates of these nodes. It should be remembered that these nodes are different from the displacement nodes for a higher-order element like the CuST. Because of simplicity of programming, the displacement and pore pressure nodes are dealt with

† Note: these are specifically for element type 7, which is the only element type with inner nodes. Any new element type with inner nodes will require this part of the code to be modified.

separately. However, the same routine which was used for the displacement nodes is used again.

When exit is made from the routine MIDPOR, *all* the nodes (pore pressure and displacement) have been assigned numbers and their co-ordinates calculated. The total number of nodes NN is now known, and the largest user node number is NNZ (remembering that the additional nodes were numbered starting from 751). When the pore pressure nodes were numbered, the user node numbers were continued from the point left by the last additional displacement node. For example, if 832 was the last displacement node number then 833 is the node number of the first pore pressure node.

At this stage, all the information necessary to number the mesh is written to the PD file in routine **NUMSH**. Still adopting the same format to write the information as before, the node co-ordinates and numbers are written. For the purpose of numbering the elements, the centroid co-ordinate and element number are written to the file.

Routine NUMSH

```
      SUBROUTINE NUMSH(IW6,IW8,NDIM,NNE,MXND,NEL,MUMAX,NNU,XYZ,          NMSH   1
     1 NCONN,LTYP,MREL,NREL,NDZ,IPLOT)                                   NMSH   2
C*********************************************************************NMSH   3
C     ROUTINE TO NUMBER MESH                                            NMSH   4
C*********************************************************************NMSH   5
      DIMENSION XYZ(NDIM,NNE),NCONN(MXND,NEL),LTYP(NEL),                 NMSH   6
     1 MREL(MUMAX),NREL(NNU),XYZD(3),XYZC(3)                             NMSH   7
      COMMON /DEBUGS/ ID1,ID2,ID3,ID4,ID5,ID6,ID7,ID8,ID9,ID10           NMSH   8
      COMMON /ELINF / LINFO(50,15)                                       NMSH   9
C                                                                        NMSH  10
      IF(IPLOT.EQ.0)RETURN                                               NMSH  11
C----------------------------------------------------------------------NMSH  12
C     NDZ1 - STARTING VALUE OF USER NUMBER OF EDGE NODES                 NMSH  13
C----------------------------------------------------------------------NMSH  14
      NDZ1=NDZ+1                                                         NMSH  15
C----------------------------------------------------------------------NMSH  16
C     CODE TO INDICATE THAT A NUMBER IS TO BE PLOTTED                    NMSH  17
C----------------------------------------------------------------------NMSH  18
      ICODE=11                                                           NMSH  19
C----------------------------------------------------------------------NMSH  20
C     DUMMY COORDINATES                                                  NMSH  21
C----------------------------------------------------------------------NMSH  22
      DO 4 ID=1,NDIM                                                     NMSH  23
    4 XYZD(ID)=0.                                                        NMSH  24
      IDUM=0                                                             NMSH  25
      IZERO=0                                                            NMSH  26
C                                                                        NMSH  27
      NC=0                                                               NMSH  28
      IPL=IPLOT                                                          NMSH  29
      IF(IPL.EQ.1)GOTO 100                                               NMSH  30
    5 IF(IPL-3)10,20,30                                                  NMSH  31
C----------------------------------------------------------------------NMSH  32
C     PEN COLOUR IS BLACK FOR VERTEX NODES                               NMSH  33
C----------------------------------------------------------------------NMSH  34
   10 IPEN=-1                                                            NMSH  35
      WRITE(IW8)IPEN,(XYZD(ID),ID=1,NDIM),IDUM                           NMSH  36
      NN1=1                                                              NMSH  37
      NN2=NDZ                                                            NMSH  38
C                                                                        NMSH  39
```



```
   12 DO 15 JR=NN1,NN2                                                   NMSH  40
      IF(NREL(JR).EQ.0)GO TO 15                                          NMSH  41
      J=NREL(JR)                                                         NMSH  42
      JJ=JR                                                              NMSH  43
      WRITE(IW8)ICODE,(XYZ(ID,J),ID=1,NDIM),JJ                           NMSH  44
   15 CONTINUE                                                           NMSH  45
C                                                                        NMSH  46
      IF(NC.EQ.0)GOTO 100                                                NMSH  47
      NC=0                                                               NMSH  48
C----------------------------------------------------------------------NMSH  49
C     PEN COLOUR IS RED FOR EDGE NODES                                   NMSH  50
C----------------------------------------------------------------------NMSH  51
   20 IPEN=-2                                                            NMSH  52
      WRITE(IW8)IPEN,(XYZD(ID),ID=1,NDIM),IDUM                           NMSH  53
      NN1=NDZ1                                                           NMSH  5
      NN2=NNU                                                            NMSH  5.
      GOTO 12                                                            NMSH  56
C                                                                        NMSH  57
   30 IF(IPL.GT.4)GOTO 40                                                NMSH  58
      NC=1                                                               NMSH  59
      GOTO 10                                                            NMSH  60
C----------------------------------------------------------------------NMSH  61
C     PEN COLOUR IS GREEN FOR ELEMENTS                                   NMSH  62
C----------------------------------------------------------------------NMSH  63
   40 IPEN=-3                                                            NMSH  64
      WRITE(IW8)IPEN,(XYZD(ID),ID=1,NDIM),IDUM                           NMSH  65
C                                                                        NMSH  66
      DO 50 JR=1,MUMAX                                                   NMSH  67
      IF(MREL(JR).EQ.0)GOTO 50                                           NMSH  68
      J=MREL(JR)                                                         NMSH  69
C                                                                        NMSH  70
      DO 35 ID=1,NDIM                                                    NMSH  71
   35 XYZC(ID)=0.                                                        NMSH  72
C                                                                        NMSH  73
      LT=LTYP(J)                                                         NMSH  74
      NVN=LINFO(2,LT)                                                    NMSH  75
C                                                                        NMSH  76
      DO 46 I=1,NVN                                                      NMSH  77
      L=NCONN(I,J)                                                       NMSH  78
      DO 46 ID=1,NDIM                                                    NMSH  79
   46 XYZC(ID)=XYZC(ID)+XYZ(ID,L)/FLOAT(NVN)                             NMSH  80
C                                                                        NMSH  81
      JJ=JR                                                              NMSH  82
      WRITE(IW8)ICODE,(XYZC(ID),ID=1,NDIM),JJ                            NMSH  83
   50 CONTINUE                                                           NMSH  84
C                                                                        NMSH  8
      IPL=IPL-4                                                          NMSH  t
      IF(IPL.GT.1)GOTO 5                                                 NMSH  87
C----------------------------------------------------------------------NMSH  88
C     CLOSE FILE                                                         NMSH  89
C----------------------------------------------------------------------NMSH  90
  100 WRITE(IW8)IZERO,(XYZD(ID),ID=1,NDIM),IDUM                          NMSH  91
      RETURN                                                             NMSH  92
      END                                                                NMSH  93
```

NMSH 15 : starting value of midside nodes (user numbers).
NMSH 19 : code to indicate a number is to be plotted.
NMSH 23–24 : dummy co-ordinates (used when pen colour is changed).
NMSH 29 : plotting code (user specified, request of mesh detail, e.g. numbering).
NMSH 35–36 : select pen colour as black (negative value indicates change in pen colour) and write information to PD file.

NMSH 37–38 : the range of node numbers includes the vertex nodes.
NMSH 40–45 : write (displacement) node co-ordinates to PD file.
NMSH 47 : branch off if no more information on mesh is required.
NMSH 52–53 : select pen colour as red for nodal numbering and write information to PD file.
NMSH 54–55 : range of midside node numbers.
NMSH 59 : branch off to plot vertex node numbers.
NMSH 64–65 : select pen colour as green for plotting element numbers and write information to PD file.
NMSH 67 : loop on all elements.
NMSH 68 : by-pass if an element number is not used.
NMSH 71–72 : initialise element centroid co-ordinates.
NMSH 74–75 : element type number (LT), number of vertex nodes in element (NVN).
NMSH 77–80 : calculate element centroid co-ordinates.
NMSH 82–84 : write element centroid co-ordinates to PD file.
NMSH 91 : close file by writing a zero code.

The remaining tasks for the geometry part of the program are the calculation of the total number of **degrees of freedom (d.o.f.)** and the **frontwidth** and the core-store required in solving the equations using the **frontal method.** The first step is to find the number of d.o.f. at each node, considering all the elements connected to that node, and this is achieved by **MAKENZ**.

Array (NQ(NN) gives the number of d.o.f. of each node. A node may have a differing number of d.o.f. from the different elements of which it is a part. This can be illustrated by an example (Fig. 6.5): in it, nodes 1 and 5 have 3 d.o.f. from the linear strain triangle of type 3. They have d_x, d_y and $\bar{u}$ as variables, the displacements in x and y directions and the excess pore pressure. From the linear strain triangle of type 2 element, the three nodes have 2 d.o.f. (d_x and d_y only). Therefore nodes 1 and 5 have a *maximum* of *3* d.o.f. This is entered in array (NQ(NN).

The number of d.o.f. is entered against that node number in array NQ. Once this task is completed, the total number of d.o.f. in the mesh – NDF – is found by summing up the entries in array NQ.

The number of d.o.f. for each node for different element types is obtained from array LINFO(50, 15). The second index is for the element type number (LT). The first 20 entries are allocated to give out general information regarding the element type. Entries starting from 21 give the number of d.o.f. for each node of an element. (The sequence used for the nodes is the same as in Fig. 4.1.)

Routine MAKENZ

```
      SUBROUTINE MAKENZ(MXND,NEL,NN,NCONN,LTYP,NQ,INXL)                MKNZ   1
C*****************************************************************************MKNZ   2
C     SETS UP THE NQ ARRAY WHICH CONTAINS THE NUMBER                   MKNZ   3
C     OF DEGREES OF FREEDOM ASSOCIATED WITH EACH NODE                  MKNZ   4
C     FOR ELEMENTS IN THIS ASSEMBLY.                                   MKNZ   5
C*****************************************************************************MKNZ   6
```

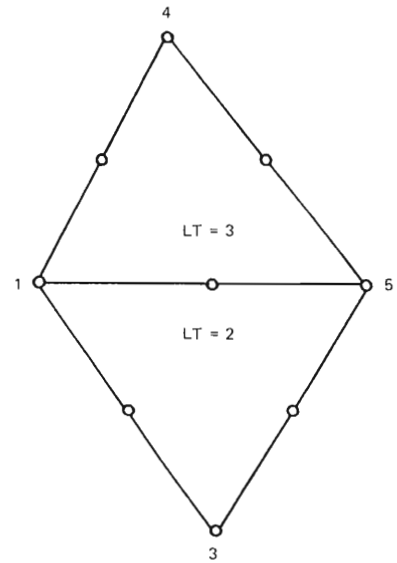



Fig. 6.5 – Same nodes with different d.o.f. when mixing different element types

```
      DIMENSION NCONN(MXND,NEL),LTYP(NEL),NQ(NN)                       MKNZ   7
      COMMON /ELINF/ LINFO(50,15)                                      MKNZ   8
C-----------------------------------------------------------------------MKNZ   9
C     INXL - INDEX TO NO. OF DEGREES OF FREEDOM OF FIRST NODE OF ELEMENTMKNZ  10
C     (SEE BLOCK DATA ROUTINES BDATA1, MAIN2)                          MKNZ  11
C-----------------------------------------------------------------------MKNZ  12
      DO 8 J=1,NN                                                      MKNZ  13
    8 NQ(J)=0                                                          MKNZ  14
C                                                                      MKNZ  15
      DO 20 J=1,NEL                                                    MKNZ  16
      IF(LTYP(J).LT.0) GOTO 20                                         MKNZ  17
      LT=LTYP(J)                                                       MKNZ  18
      NDPT=LINFO(1,LT)                                                 MKNZ  19
C                                                                      MKNZ  20
      DO 10 I=1,NDPT                                                   MKNZ  21
      NDFN=LINFO(I+INXL,LT)                                            MKNZ  22
      NOD=NCONN(I,J)                                                   MKNZ  23
      IF(NDFN.GT.NQ(NOD)) NQ(NOD)=NDFN                                 MKNZ  24
   10 CONTINUE                                                         MKNZ  25
   20 CONTINUE                                                         MKNZ  26
C                                                                      MKNZ  27
      RETURN                                                           MKNZ  28
      END                                                              MKNZ  29
```

MKNZ 13–14 : zero the number of d.o.f. of all nodes.
MKNZ 16 : loop on all elements.
MKNZ 17 : by-pass if element is not present in the current mesh.
MKNZ 18 : element type number.
MKNZ 19 : the total number of nodes in element.
MKNZ 21 : loop on all nodes of element.

MKNZ 22 : the number of d.o.f. of node.
MKNZ 23 : node number.
MKNZ 24 : enter if node associated with the current element has a greater number of d.o.f.
MKNZ 25 : end of loop on all nodes of element.
MKNZ 26 : end of element loop.

It is necessary for the purpose of internal housekeeping to assign unique numbers to each d.o.f. or variable. This number lies in the range 1 to NDF and will be referred to as the **global variable number (g.v.n.)** in the rest of the text. For simplicity, array NQ is used for this purpose. All the d.o.f. of a particular node are given consecutive numbers. Hence it is only necessary to know the g.v.n. of the first variable of each node. Array NW is set up to provide this information. The first d.o.f. of the tenth node, for example, is given the sum total of the d.o.f. of the first 9 nodes + 1. The NW entries will be

node →	1	2	3	4	5	6	7	8	(9)
d.o.f. →	3	3	3	3	2	2	2	2	
g.v.n. →	1	4	7	10	13	15	17	19	(21)

The last 'non-existent' node serves as a marker; for example, the difference in g.v.n. between consecutive node numbers is the d.o.f. of the first numbered node.

number of d.o.f. of node 5 = NW(6) − NW(5) = 15 − 13 = 2

number of d.o.f. of node 8 = NW(9) − NW(8) = 21 − 19 = 2

Hence the entry for the last 'non-existent' node will always be NDF + 1. The routine which carries out the above calculations is **CALDOF**.

Routine CALDOF

```
      SUBROUTINE CALDOF(IW6,NN,NN1,NDF,NW,NQ)                          CLDF   1
C*****************************************************************CLDF   2
C     ROUTINE TO CALCULATE GLOBAL NUMBER FOR D.O.F.                    CLDF   3
C*****************************************************************CLDF   4
      DIMENSION NW(NN1),NQ(NN)                                         CLDF   5
C                                                                      CLDF   6
      NC=1                                                             CLDF   7
      NW(1)=1                                                          CLDF   8
C                                                                      CLDF   9
      DO 10 I=1,NN                                                     CLDF  10
      NC=NC+NQ(I)                                                      CLDF  11
   10 NW(I+1)=NC                                                       CLDF  12
C                                                                      CLDF  13
      NDF=NW(NN1)-1                                                    CLDF  14
C                                                                      CLDF  15
      RETURN                                                           CLDF  16
      END                                                              CLDF  17
```

CLDF 7–8 : global variable nos. of first d.o.f. of first node.
These g.v.n. serve as indexes to arrays P, PT, DI, DA, etc.
CLDF 10 : loop on all nodes.

CLDF 11 : calculate global variable nos. of first d.o.f. of next node (= I + 1).
CLDF 12 : place value in array NW.
CLDF 14 : NDF is the total number of d.o.f. in mesh.

6.6 PRE-FRONTAL ROUTINES

Routines **MLAPZ** and **SFWZ** are the pre-frontal routines. The frontal method is described elsewhere in detail (Irons, 1970; Irons and Ahmad, 1980; Hinton and Owen, 1977). The function of these two routines is best illustrated by an example (Fig. 6.6).

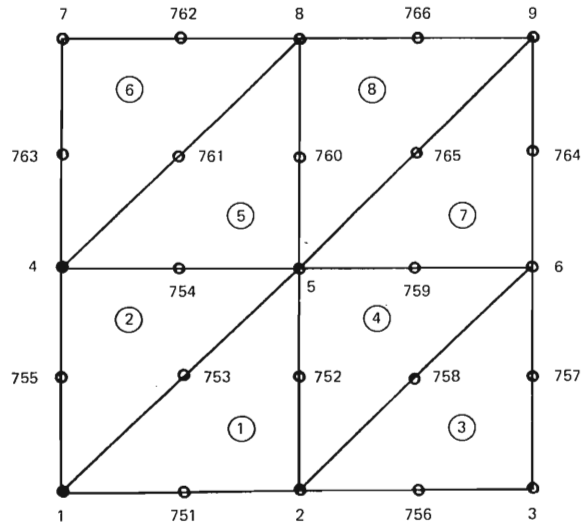


Fig. 6.6 – Example to illustrate frontal method

Element	Type	Mat	N1	N2	N3	N4	N5	N6
1	2	1	1	2	5	751	752	753
2	2	1	1	5	4	753	754	755
3	2	1	2	3	6	756	757	758
4	2	1	2	6	5	758	759	752
5	2	1	4	5	8	754	760	761
6	2	1	4	8	7	761	762	763
7	2	1	5	6	9	759	764	765
8	2	1	5	9	8	765	766	760

These are the input data, and assuming that no alternative efficient element numbering has been specified, the midside node numbers are given on the right-

hand side of the above table. It gives the contents of the array NCONN, translated into user node numbers. The actual midside node number entries in NCONN are different, even though the vertex node number entries are exactly the same. The array is modified so that the node numbers are made negative to indicate their last appearances.

No. of elements	N1	N2	N3	N4	N5	N6
1	1	2	5	−751	752	753
2	−1	5	4	−753	754	−755
3	2	−3	6	−756	−757	758
4	−2	6	5	−758	759	−752
5	4	5	8	−754	760	761
6	−4	8	−7	−761	−762	−763
7	5	−6	9	−759	−764	765
8	−5	−9	−8	−765	−766	−760

Considering the above table of element–nodal connectivity in the **reverse** order, the **first** time a node appears will be its last appearance. This method is used to find the last appearance of a node.

Routine MLAPZ

```
      SUBROUTINE MLAPZ(MXND,NEL,NN,NCONN,LTYP,NQ)                       MLPZ   1
C*****************************************************************MLPZ   2
C     MARKS LAST APPEARANCES OF NODES BY MAKING THEM NEGATIVE         MLPZ   3
C     IN NCONN ARRAY                                                  MLPZ   4
C*****************************************************************MLPZ   5
      DIMENSION NCONN(MXND,NEL),LTYP(NEL),NQ(NN)                      MLPZ   6
      COMMON /ELINF/ LINFO(50,15)                                     MLPZ   7
C                                                                     MLPZ   8
      NEL1=NEL+1                                                      MLPZ   9
C                                                                     MLPZ  10
      DO 30 M=1,NN                                                    MLPZ  11
      IF(NQ(M).EQ.0) GOTO 30                                          MLPZ  12
      DO 20 J=1,NEL                                                   MLPZ  13
      JB=NEL1-J                                                       MLPZ  14
      IF(LTYP(JB).LT.0) GOTO 20                                       MLPZ  15
      LT=LTYP(JB)                                                     MLPZ  16
      NDPT=LINFO(1,LT)                                                MLPZ  17
      DO 10 I=1,NDPT                                                  MLPZ  18
      IF(NCONN(I,JB).NE.M) GOTO 10                                    MLPZ  19
      NCONN(I,JB)=-NCONN(I,JB)                                        MLPZ  20
      GOTO 30                                                         MLPZ  21
   10 CONTINUE                                                        MLPZ  22
   20 CONTINUE                                                        MLPZ  23
   30 CONTINUE                                                        MLPZ  24
C                                                                     MLPZ  25
      RETURN                                                          MLPZ  26
      END                                                             MLPZ  27
```

MLPZ 11 : loop on all nodes.
MLPZ 12 : by-pass if node is not present in the mesh – number of d.o.f. is zero (probably due to removal of some elements).
MLPZ 13 : loop on all elements.

MLPZ 14 : element number in reverse order.
MLPZ 15 : by-pass if element is not present in mesh.
MLPZ 16–17 : element type number (LT) and the number of nodes associated with the element (NDPT).
MLPZ 18 : loop on all nodes associated with the element.
MLPZ 19 : if node is not found, then by-pass.
MLPZ 20 : make node number negative to indicate last appearance of node in mesh.
MLPZ 22 : end of loop on all nodes associated with the element.
MLPZ 23 : end of element loop.
MLPZ 24 : end of nodal loop.

Element assembled	List of active nodes after assembly	Nodes which remain active after elimination
1	1 2 5 **751** 752 753	1 2 5 0 752 753
2	**1** 2 5 4 752 **753** 754 **755**	0 2 5 4 752 0 754
3	**3** 2 5 4 752 6 754 **756** **757** 758	0 2 5 4 752 6 754 0 0 758
4	759 **2** 5 4 **752** 6 754 0 0 **758**	759 0 5 4 0 6 754
5	759 8 5 4 760 6 **754** 761	759 8 5 4 760 6 0 761
6	759 8 5 **4** 760 6 **7** **761** **762** **763**	759 8 5 0 760 6
7	**759** 8 5 9 760 **6** **764** 765	0 8 5 9 760 0 0 765
8	**766** **8** **5** **9** **760** 0 0 **765**	

In routine **SFWZ**, the program calculates the maximum frontwidth (and the core-store required to solve the equations using the frontal method) using the last appearances of nodes marked by the routine **MLAPZ**.

This is illustrated above. Making use of the previous table, after the first element is assembled the number of active nodes, which is six, reduces to five after the node which is underlined (only 751) is eliminated. The corresponding entry on the list on the right-hand side is zero. Scanning through the list of active nodes, the maximum number of nodes present at any stage is 10; hence the maximum frontwidth is 10 nodes. It is the maximum number of nodes that are active at any time. If each node has 2 d.o.f. then the maximum frontwidth is 20 d.o.f.

Routine SFWZ

```
      SUBROUTINE SFWZ(MNFZ,MXND,NEL,NN,MUMAX,NNZ,IFRZ,                  SFWZ   1
     1 NCONN,LTYP,MREL,NREL,NQ,NDEST,IFR,MULT,MCORE,NCORET)             SFWZ   2
C*****************************************************************SFWZ   3
```

```
C     WORKS OUT FRONT WIDTH FOR SYMMETRIC SOLUTION                    SFWZ   4
C     USING LAST APPEARANCES MARKED BY SUBROUTINE MLAPZ.              SFWZ   5
C*****************************************************************SFWZ   6
      DIMENSION NCONN(MXND,NEL),LTYP(NEL),MREL(MUMAX),NREL(NNZ)       SFWZ   7
      DIMENSION NQ(NN),NDEST(NN),IFR(IFRZ)                           SFWZ   8
      COMMON /DEVICE/ IR1,IR4,IR5,IW2,IW4,IW6,IW7,IW8,IW9            SFWZ   9
      COMMON /DEBUGS/ ID1,ID2,ID3,ID4,ID5,ID6,ID7,ID8,ID9,ID10       SFWZ  10
      COMMON /ELINF/ LINFO(50,15)                                    SFWZ  11
C                                                                    SFWZ  12
      INCORE=0                                                       SFWZ  13
C                                                                    SFWZ  14
      DO 6 J=1,NEL                                                   SFWZ  15
      IF(LTYP(J).LT.0) GOTO 6                                        SFWZ  16
      N=NCONN(1,J)                                                   SFWZ  17
      NA=IABS(N)                                                     SFWZ  18
      NDFN=NQ(NA)                                                    SFWZ  19
C                                                                    SFWZ  20
      DO 4 I=1,NDFN                                                  SFWZ  21
    4 IFR(I)=NA                                                      SFWZ  22
      NFZ=NDFN                                                       SFWZ  23
      MNFZ=NDFN                                                      SFWZ  24
      NDEST(NA)=1                                                    SFWZ  25
      GOTO 8                                                         SFWZ  26
    6 CONTINUE                                                       SFWZ  27
C                                                                    SFWZ  28
      WRITE(IW6,900)                                                 SFWZ  29
      STOP                                                           SFWZ  30
C                                                                    SFWZ  31
    8 CONTINUE                                                       SFWZ  32
C-----------------------------------------------------------------SFWZ  33
C     CONSIDER EACH ELEMENT IN TURN                                  SFWZ  34
C-----------------------------------------------------------------SFWZ  35
      DO 40 J=1,NEL                                                  SFWZ  36
C-----------------------------------------------------------------SFWZ  37
C     IGNORE OMITTED ELEMENTS                                        SFWZ  38
C-----------------------------------------------------------------SFWZ  39
      IF(LTYP(J).LT.0) GOTO 40                                       SFWZ  40
C-----------------------------------------------------------------SFWZ  41
C     CONSIDER EACH NODE OF THIS ELEMENT - DOES IT ALREADY HAVE      SFWZ  42
C     A ROW/COLUMN ALLOCATED TO IT IN THE FRONT ?                    SFWZ  43
C-----------------------------------------------------------------SFWZ  44
      LT=LTYP(J)                                                     SFWZ  45
      NDPT=LINFO(1,LT)                                               SFWZ  46
C                                                                    SFWZ  47
      DO 20 I=1,NDPT                                                 SFWZ  48
      N=NCONN(I,J)                                                   SFWZ  49
      NA=IABS(N)                                                     SFWZ  50
C                                                                    SFWZ  51
      DO 10 K=1,NFZ                                                  SFWZ  52
      IF(IFR(K).EQ.NA) GOTO 20                                       SFWZ  53
   10 CONTINUE                                                       SFWZ  54
C-----------------------------------------------------------------SFWZ  55
C     FIND A (LARGE ENOUGH) GAP OR PUT ON END                        SFWZ  56
C-----------------------------------------------------------------SFWZ  57
      K1=1                                                           SFWZ  58
   11 DO 12 K=K1,NFZ                                                 SFWZ  59
      IF(IFR(K).EQ.0) GOTO 15                                        SFWZ  60
   12 CONTINUE                                                       SFWZ  61
C-----------------------------------------------------------------SFWZ  62
C     PUT ON END                                                     SFWZ  63
C-----------------------------------------------------------------SFWZ  64
      K=NFZ+1                                                        SFWZ  65
      NFZ=NFZ+NQ(NA)                                                 SFWZ  66
      IF(NFZ.LE.IFRZ) GOTO 14                                        SFWZ  67
      WRITE(IW6,904)IFRZ                                             SFWZ  68
      STOP                                                           SFWZ  69
```



```
C                                                                    SFWZ  70
   14 K2=NFZ                                                         SFWZ  71
      IF(NFZ.GT.MNFZ)MNFZ=NFZ                                        SFWZ  72
      GOTO 18                                                        SFWZ  73
C                                                                    SFWZ  74
   15 DO 16 KK=K,NFZ                                                 SFWZ  75
      IF(IFR(KK).NE.0) GOTO 17                                       SFWZ  76
   16 CONTINUE                                                       SFWZ  77
C                                                                    SFWZ  78
      WRITE(IW6,905)                                                 SFWZ  79
      WRITE(IW6,997)J,I                                              SFWZ  80
      WRITE(IW6,998)NFZ                                              SFWZ  81
      WRITE(IW6,999)(IFR(LL),LL=1,NFZ)                               SFWZ  82
      STOP                                                           SFWZ  83
C                                                                    SFWZ  8
   17 K1=KK                                                          SFWZ  8
      IF(NQ(NA).GT.KK-K) GOTO 11                                     SFWZ  86
      K2=K+NQ(NA)-1                                                  SFWZ  87
   18 NDEST(NA)=K                                                    SFWZ  88
C                                                                    SFWZ  89
      DO 19 KK=K,K2                                                  SFWZ  90
   19 IFR(KK)=NA                                                     SFWZ  91
   20 CONTINUE                                                       SFWZ  92
CC    WRITE(IW6,999)(IFR(LL),LL=1,NFZ)                               SFWZ  93
C-----------------------------------------------------------------SFWZ  94
C     ELIMINATE NODES FROM FRONT THAT ARE MAKING THEIR LAST APPEARANCES.SFWZ  95
C-----------------------------------------------------------------SFWZ  96
      DO 30 I=1,NDPT                                                 SFWZ  97
      IF(NCONN(I,J).GT.0) GOTO 30                                    SFWZ  98
C                                                                    SFWZ  99
      DO 22 K=1,NFZ                                                  SFWZ 100
      N=NCONN(I,J)                                                   SFWZ 101
      NA=IABS(N)                                                     SFWZ 102
      IF(NA.EQ.IFR(K)) GOTO 23                                       SFWZ 103
   22 CONTINUE                                                       SFWZ 104
      WRITE(IW6,908)                                                 SFWZ 105
      STOP                                                           SFWZ 106
C                                                                    SFWZ 107
   23 K2=K+NQ(NA)-1                                                  SFWZ 108
      NCONN(I,J)=NCONN(I,J)*MULT                                     SFWZ 109
      DO 24 KK=K,K2                                                  SFWZ 110
      INCORE=INCORE+NFZ+4                                            SFWZ 111
   24 IFR(KK)=0                                                      SFWZ 112
      IF(K2.LT.NFZ) GOTO 30                                          SFWZ 113
C                                                                    SFWZ 114
   26 NFZ=NFZ-1                                                      SFWZ 11
      IF(NFZ.EQ.0) GOTO 30                                           SFWZ 1
      IF(IFR(NFZ).EQ.0) GOTO 26                                      SFWZ 11,
C                                                                    SFWZ 118
   30 CONTINUE                                                       SFWZ 119
C                                                                    SFWZ 120
      IF(ID3.NE.1)GOTO 40                                            SFWZ 121
      IF(NFZ.GT.0) WRITE(IW6,999)(IFR(LL),LL=1,NFZ)                  SFWZ 122
   40 CONTINUE                                                       SFWZ 123
C                                                                    SFWZ 124
      WRITE(IW6,910) MNFZ                                            SFWZ 125
C                                                                    SFWZ 126
      IF(ID4.EQ.1)WRITE(IW6,950)NDEST                                SFWZ 127
      MCORE=MNFZ*(MNFZ+1)/2+2*MNFZ+502                               SFWZ 128
      NCORET=MCORE+INCORE                                            SFWZ 129
      WRITE(IW6,915)MCORE                                            SFWZ 130
      WRITE(IW6,920)INCORE                                           SFWZ 131
      RETURN                                                         SFWZ 132
  900 FORMAT(41H NO ELEMENTS IN SOLUTION ! (ROUTINE SFWZ))           SFWZ 133
  904 FORMAT(48H ***ERROR** TOO MANY DEGREES OF FREEDOM IN FRONT,    SFWZ 134
     1 1X,7HEXCEEDS,I5,2X,14H(ROUTINE SFWZ))                         SFWZ 135
```

```
905 FORMAT(40H PROGRAM ERROR - NO NODE ON END OF FRONT/                SFWZ 136
   1 15H (ROUTINE SFWZ))                                              SFWZ 137
908 FORMAT(53H PROGRAM ERROR - LAST APPEARANCE NODE IS NOT IN FRONT,   SFWZ 138
   1 2X,14H(ROUTINE SFWZ))                                            SFWZ 139
910 FORMAT(/36H MAXIMUM FRONT WIDTH FOR SOLUTION = ,I4,                SFWZ 140
   1 19H DEGREES OF FREEDOM)                                          SFWZ 141
915 FORMAT(/44H MINIMUM CORE REQUIRED TO SOLVE EQUATIONS = ,I10)       SFWZ 142
920 FORMAT(/48H ADDITIONAL CORE REQUIRED FOR INCORE SOLUTION = ,I10)   SFWZ 143
950 FORMAT(//1X,5HNDEST/(1X,20I5))                                    SFWZ 144
997 FORMAT(5H J = ,I5,7H   I = ,I5)                                   SFWZ 145
998 FORMAT(7H NFZ = ,I12)                                             SFWZ 146
999 FORMAT(4H IFR/(1X,25I5))                                          SFWZ 147
    END                                                               SFWZ 148
```

SFWZ 13 : initialise buffer size for in-core solution of equations.
SFWZ 15 : loop on all elements (this loop is only to find the first node a place in the front).
SFWZ 16 : by-pass if element is not in current mesh.
SFWZ 17 : node number.
SFWZ 18 : absolute value of node number (nodes may be making their last appearance and can be negative).
SFWZ 19 : number of d.o.f. of node.
SFWZ 21–22 : place node number in the front for each d.o.f. of a node (all d.o.f. of a particular node are identified by the node number).
SFWZ 23 : number of d.o.f. in the front.
SFWZ 24 : maximum size of the front.
SFWZ 25 : make entry (in array NDEST) to indicate the position of first d.o.f. of node in the front.
SFWZ 26–27 : exit from element loop after one node has been placed in the front.
SFWZ 36 : loop on all elements.
SFWZ 40 : by-pass if element is not present in current mesh.
SFWZ 45–46 : element type number and total number of nodes in element.
SFWZ 48 : loop on all nodes in element.
SFWZ 49–50 : node number.
SFWZ 52–53 : search the front (i.e. array IFR) to see if node has already been allocated store, and if so, branch off.
SFWZ 58–61 : search for gaps (zero entry in IFR) to allocate place in front for a node making the first appearance. Branch off if a zero is found.
SFWZ 65–66 : no gaps found. Place new node (and its d.o.f.) at the end of the front.
SFWZ 67–68 : check that size of array IFR is not exceeded. If so, print out message and stop.
SFWZ 71 : update size of the front.
SFWZ 72 : update maximum size of the front (if the current size is greater than the previous maximum).
SFWZ 75–76 : scan the front (array IFR) to find the size of gap (i.e. with zero entries) and branch off when a non-zero entry is found.

SFWZ 79–82 : if end of front cannot be found, print error message and stop (this should never happen).
SFWZ 86–87 : check if gap in the front is of sufficient size to accommodate all d.o.f. of node appearing for the first time. (All d.o.f. of a given node are strung together and take up consecutive places in the front.) Since different nodes may have different number of d.o.f. it is necessary to check the size of the *gap*. The gaps in the front have been left by nodes which have been eliminated.
SFWZ 90–91 : place node number (for all d.o.f. of node) in the front.
SFWZ 92 : end of loop on all nodes of element.
SFWZ 97 : loop on all nodes of element.
SFWZ 98 : by-pass if node is not making its last appearance.
SFWZ 100 : loop on all nodes in the front. Scan the front for node number.
SFWZ 101–102 : node number.
SFWZ 103 : node has been found. Branch off.
SFWZ 105–106 : node appearing for the last time is not in the front. Print out error message and stop (this should never happen).
SFWZ 108 : find position of last d.o.f. of node in the front.
SFWZ 109 : multiply node number by MULT. (The node numbers are made positive only if the routine has been called by the geometry part of the program, i.e. MULT $= -1$; otherwise MULT $= 1$.)
SFWZ 110–112 : calculate core requirements for in-core solution.
SFWZ 113 : by-pass if node eliminated is not at the end of the front.
SFWZ 115–117 : if so, reduce the front and hence NFZ (current size of the front).
SFWZ 119 : end of loop on nodes of element.
SFWZ 121–122 : print out list of nodes in current front for debugging.
SFWZ 123 : end of element loop.
SFWZ 125 : print maximum front size (the frontwidth).
SFWZ 127 : contents of nodal destination vector NDEST (gives the destination of nodes in the front) are printed for debugging.
SFWZ 128 : minimum core required to solve the equations.
SFWZ 129 : total store required to solve all equations in-core.
SFWZ 130–131 : print out core requirements.

Of course, if one is carrying out a consolidation analysis, the number of d.o.f. varies from node to node. The above example, illustrated with simply the node numbers, takes a slightly different form. One has to consider the d.o.f. instead of the nodes. A list of currently active d.o.f. is maintained in array IFR. If a node has 3 d.o.f., the node number is entered in three consecutive places, representing the 3 d.o.f. Similarly for a node with 2 d.o.f.: when a node with 2 d.o.f. is eliminated, it leaves a gap of size 2. If a new node with 3 d.o.f. is assembled then

the current front is scanned from left to right first to find a suitable gap with at least three zeroes. Hence the gap of 2 is passed over and the new node and its variables are put at the end of the current front, thereby increasing the front-width momentarily by 3.

The final task to complete the geometry part of CRISP is to print out the complete element–nodal connectivity list, NCONN. Remember that NCONN contains the *program* node numbers. For each element, a temporary array of *user* node numbers is set up, and these are printed along with the element type number and material zone number. This is carried out by routine **GPOUT**. For debugging purposes, various arrays can be printed during the course of the geometry part of the program.

Routine GPOUT

```
      SUBROUTINE GPOUT(IW6,MXND,NEL,MUMAX,NN,NN1,NDF,NCONN,              GOUT   1
     1 MAT,LTYP,MRELVV,MREL,NRELVV,NW,NQ,NLST)                           GOUT   2
C*********************************************************************GOUT   3
C     ROUTINE TO PRINTOUT ARRAYS SET-UP IN GEOMETRY PART OF PROGRAM     GOUT   4
C*********************************************************************GOUT   5
      DIMENSION NCONN(MXND,NEL),MAT(NEL),LTYP(NEL),MRELVV(NEL),         GOUT   6
     1 MREL(MUMAX),NRELVV(NN),NW(NN1),NQ(NN),NLST(MXND)                 GOUT   7
      COMMON /DEBUGS/ ID1,ID2,ID3,ID4,ID5,ID6,ID7,ID8,ID9,ID10          GOUT   8
      COMMON /ELINF / LINFO(50,15)                                      GOUT   9
C                                                                       GOUT  10
      WRITE(IW6,902)                                                    GOUT  11
C                                                                       GOUT  12
      DO 20 JU=1,MUMAX                                                  GOUT  13
      IF(MREL(JU).EQ.0)GOTO 20                                          GOUT  14
      MPR=MREL(JU)                                                      GOUT  15
      LT=LTYP(MPR)                                                      GOUT  16
      NDPT=LINFO(1,LT)                                                  GOUT  17
C                                                                       GOUT  18
      DO 10 IN=1,NDPT                                                   GOUT  19
      NP=NCONN(IN,MPR)                                                  GOUT  20
   10 NLST(IN)=NRELVV(NP)                                               GOUT  21
C                                                                       GOUT  22
      WRITE(IW6,906)JU,LT,MAT(MPR),(NLST(IN),IN=1,NDPT)                 GOUT  23
   20 CONTINUE                                                          GOUT  24
C                                                                       GOUT  25
      IF(ID10.EQ.1)WRITE(IW6,908)(NQ(IN),IN=1,NN)                       GOUT  26
C                                                                       GOUT  27
      IF(ID10.EQ.1)WRITE(IW6,910)(NW(IN),IN=1,NN1)                      GOUT  28
C                                                                       GOUT  29
      WRITE(IW6,911)NN                                                  GOUT  30
      WRITE(IW6,912)NDF                                                 GOUT  31
C                                                                       GOUT  32
      RETURN                                                            GOUT  33
  902 FORMAT(//10X,30H  ELEMENT  MATERIAL  TYPE  AND,                   GOUT  34
     1 15H  NODE  NUMBERS//1X,7HELEMENT,1X,4HTYPE,2X,3HMAT,             GOUT  35
     2 19H   1    2    3    4,                                          GOUT  36
     3 55H    5    6    7    8    9   10   11   12   13   14   15,     GOUT  37
     4 35H   16   17   18   19   20   21   22/)                        GOUT  38
  906 FORMAT(I5,2I6,22I5)                                               GOUT  39
  908 FORMAT(/1X,2HNQ/(1X,20I5))                                        GOUT  40
  910 FORMAT(/1X,2HNW/(1X,20I5))                                        GOUT  41
  911 FORMAT(//25H TOTAL NUMBER OF NODES = ,I8)                         GOUT  42
  912 FORMAT(/40H TOTAL DEGREES OF FREEDOM IN SOLUTION = ,I8)           GOUT  43
      END                                                               GOUT  44
```


GOUT 13 : loop on all elements (in the user numbering sequence).
GOUT 14 : by-pass if element number is not used.
GOUT 15 : program element number.
GOUT 16 : element type number.
GOUT 17 : total number of nodes in element.
GOUT 19 : loop on all nodes of element.
GOUT 20 : (program) node number.
GOUT 21 : place user node number in output list (i.e. array NLST).
GOUT 23 : print out the element type, material zone number and the list of nodes associated with the element.
GOUT 24 : end of element loop.
GOUT 26 : print out array NQ, giving the number of d.o.f. of each node for debugging.
GOUT 28 : print out array NW, giving the g.v.n. of the first d.o.f. of each node for debugging.
GOUT 30–31 : write the total number of nodes and d.o.f. in the problem.

6.7 PROGRAMMING TECHNIQUES

It was shown earlier that the d.o.f. of each node for all element types are stored in a single array which resides in a COMMON block and which is initialised in a **BLOCK DATA** routine. The element type number LT is used an index to this array, LINFO. Note that array LINFO is referred to as LIN in the block data routine. For example, if one considers the six-noded triangle (LT = 2) then LINFO(21, 2)–LINFO(26, 2) contain the values

2 2 2 2 2 2

meaning that all six nodes have 2 d.o.f. (d_x and d_y). In contrast, in the 'consolidation' cubic strain triangle (LT = 7) the entries LINFO(21. 7)–LINFO(42, 7) contain

Location →	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
value →	3	3	3	2	2	2	2	2	2	2	2	2	2	2	2
	{1}			{2}									{3}		

Location →	16	17	18	19	20	21	22
value →	1	1	1	1	1	1	1
	{4}						{5}

where

{1} are the vertex nodes with 3 d.o.f. (d_x, d_y and $\bar{u}$).
{2} are the displacement nodes along side (edge) with 2 d.o.f. (d_x, d_y).
{3} are the displacement nodes within element with 2 d.o.f. (d_x, d_y).

$\{4\}$ are the pore pressure nodes along side with 1 d.o.f. ($\bar{u}$).
$\{5\}$ are the pore pressure nodes within element with 1 d.o.f. ($\bar{u}$).

Routine BDATA1

```
      BLOCK DATA                                                        BDAT   1
C*****************************************************************BDAT   2
C     DATA PRESENTED BY LIN (FIRST INDEX)                               BDAT   3
C     1 - TOTAL NUMBER OF NODES (DISPLACEMENT + POREPRESSURE).....NDPT  BDAT   4
C     2 - TOTAL NUMBER OF VERTEX NODES.....NVN                          BDAT   5
C     3 - TOTAL NUMBER OF ELEMENT EDGES.....NEDG                        BDAT   6
C     4 - TOTAL NUMER OF ELEMENT FACES (3D).....NFAC                    BDAT   7
C     5 - TOTAL NUMBER OF DISPLACEMENT NODES.....NDN                    BDAT   8
C     6 - TOTAL NUMBER OF POREPRESSURE NODES.....NPN                    BDAT   9
C     7 - NO. OF DISPLACEMENT NODES PER EDGE (EXCLUDING END NODES).NDSD BDAT  10
C     8 - NO. OF POREPRESSURE NODES PER EDGE (EXCLUDING END NODES).NPSD BDAT  11
C     9 - NUMBER OF INNER DISPLACEMENT NODES.....NIND                   BDAT  12
C    10 - NUMBER OF INNER POREPRESSURE NODES.....NINP                   BDAT  13
C    11 - NUMBER OF INTEGRATION POINTS.....NGP                          BDAT  14
C    12 - INDEX TO WEIGHTS AND INTEGRATION POINT COORDINATES.....INDX   BDAT  15
C    13 - INDEX TO VERTEX NODES OF ELEMENTS (ARRAY NFC).....INX         BDAT  16
C    14 - INDEX TO NODES ALONG EDGE (ARRAYS NP1, NP2).....INDED         BDAT  17
C    15 - NUMBER OF LOCAL OR AREA COORDINATES.....NL                    BDAT  18
C    16 - TOTAL NUMBER OF DEGREES OF FREEDOM (D.O.F.) IN ELEMENT...MDFE BDAT  19
C    17 - CENTROID INTEGRATION POINT NUMBER.....NCGP                    BDAT  20
C    21 - ONWARDS THE NUMBER OF D.O.F. OF EACH NODE OF ELEMENT.....NDFN BDAT  21
C                                                                       BDAT  22
C       ELEMENT TYPES (SECOND INDEX)                                    BDAT  23
C        1 - 3-NODED BAR .....(2-D)                  **                 BDAT  24
C        2 - 6-NODED LST TRIANGLE.....(2-D)                             BDAT  25
C        3 - 6-NODED LST TRIANGLE.....(2-D CONSOLIDATION)               BDAT  26
C        4 - 8-NODED QUADRILATERAL.....(2-D)         **                 BDAT  27
C        5 - 8-NODED QUADRILATERAL.....(2-D CONSOLIDATION)  **          BDAT  28
C        6 - 15-NODED CUST TRIANGLE.....(2-D)                           BDAT  29
C        7 - 22-NODED CUST TRIANGLE.....(2-D CONSOLIDATION)             BDAT  30
C        8 - 20-NODED BRICK.....(3-D)                **                 BDAT  31
C        9 - 20-NODED BRICK.....(3-D CONSOLIDATION)  **                 BDAT  32
C       10 - 10-NODED TETRA-HEDRA.....(3-D)          **                 BDAT  33
C       11 - 10-NODED TETRA-HEDRA.....(3-D CONSOLIDATION)  **          BDAT  34
C                                                                       BDAT  35
C            **  ELEMENT TYPES NOT IMPLEMENTED IN THIS VERSION          BDAT  36
C*****************************************************************BDAT  37
      REAL L                                                            BDAT  38
      COMMON /ELINF/ LIN(50,15)                                         BDAT  39
      COMMON /DATL / L(4,100)                                           BDAT  40
      COMMON /DATW / W(100)                                             BDAT  41
      COMMON /SAMP / POSSP(5),WEIGP(5)                                  BDAT  42
      DATA LIN(1,1),LIN(2,1),LIN(3,1),LIN(4,1),LIN(5,1),LIN(6,1),       BDAT  43
     1 LIN(7,1),LIN(8,1),LIN(9,1),LIN(10,1),LIN(11,1),LIN(12,1),        BDAT  44
     2 LIN(13,1),LIN(14,1),LIN(15,1),LIN(16,1),LIN(17,1),               BDAT  45
     3 LIN(21,1),LIN(22,1),LIN(23,1)/                                   BDAT  46
     3 3,2,1,1,3,0,1,0,0,0,5,0,0,0,1,6,3,2,2,2/                         BDAT  47
      DATA LIN(1,2),LIN(2,2),LIN(3,2),LIN(4,2),LIN(5,2),LIN(6,2),       BDAT  48
     1 LIN(7,2),LIN(8,2),LIN(9,2),LIN(10,2),LIN(11,2),LIN(12,2),        BDAT  49
     2 LIN(13,2),LIN(14,2),LIN(15,2),LIN(16,2),LIN(17,2),               BDAT  50
     3 LIN(21,2),LIN(22,2),LIN(23,2),LIN(24,2),LIN(25,2),LIN(26,2)/     BDAT  51
     4 6,3,3,1,6,0,1,0,0,0,7,5,0,0,3,12,7,2,2,2,2,2,2/                  BDAT  52
      DATA LIN(1,3),LIN(2,3),LIN(3,3),LIN(4,3),LIN(5,3),LIN(6,3),       BDAT  53
     1 LIN(7,3),LIN(8,3),LIN(9,3),LIN(10,3),LIN(11,3),LIN(12,3),        BDAT  54
     2 LIN(13,3),LIN(14,3),LIN(15,3),LIN(16,3),LIN(17,3),LIN(21,3),     BDAT  55
     3 LIN(22,3),LIN(23,3),LIN(24,3),LIN(25,3),LIN(26,3)/               BDAT  56
     4 6,3,3,1,6,3,1,0,0,0,7,5,0,0,3,15,7,3,3,3,2,2,2/                  BDAT  57
      DATA LIN(1,4),LIN(2,4),LIN(3,4),LIN(4,4),LIN(5,4),LIN(6,4),       BDAT  58
```



```
     1 LIN(7,4),LIN(8,4),LIN(9,4),LIN(10,4),LIN(11,4),LIN(12,4),        BDAT  59
     2 LIN(13,4),LIN(14,4),LIN(15,4),LIN(16,4),LIN(17,4),               BDAT  60
     3 LIN(21,4),LIN(22,4),LIN(23,4),LIN(24,4),LIN(25,4),               BDAT  61
     4 LIN(26,4),LIN(27,4),LIN(28,4)/                                   BDAT  62
     4 8,4,4,1,8,0,1,0,0,0,9,12,4,3,2,16,9,2,2,2,2,2,2,2/               BDAT  63
      DATA LIN(1,5),LIN(2,5),LIN(3,5),LIN(4,5),LIN(5,5),LIN(6,5),       BDAT  64
     1 LIN(7,5),LIN(8,5),LIN(9,5),LIN(10,5),LIN(11,5),LIN(12,5),        BDAT  65
     2 LIN(13,5),LIN(14,5),LIN(15,5),LIN(16,5),LIN(17,5),               BDAT  66
     3 LIN(21,5),LIN(22,5),LIN(23,5),LIN(24,5),LIN(25,5),               BDAT  67
     4 LIN(26,5),LIN(27,5),LIN(28,5)/                                   BDAT  68
     4 8,4,4,1,8,4,1,0,0,0,9,12,4,3,2,20,9,3,3,3,3,2,2,2/               BDAT  69
      DATA LIN(1,6),LIN(2,6),LIN(3,6),LIN(4,6),LIN(5,6),LIN(6,6),       BDAT  70
     1 LIN(7,6),LIN(8,6),LIN(9,6),LIN(10,6),LIN(11,6),LIN(12,6),        BDAT  71
     2 LIN(13,6),LIN(14,6),LIN(15,6),LIN(16,6),LIN(17,6),               BDAT  72
     3 LIN(21,6),LIN(22,6),LIN(23,6),                                   BDAT  73
     4 LIN(24,6),LIN(25,6),LIN(26,6),LIN(27,6),LIN(28,6),LIN(29,6),     BDAT  74
     5 LIN(30,6),LIN(31,6),LIN(32,6),LIN(33,6),LIN(34,6),LIN(35,6)/     BDAT  75
     5 15,3,3,1,15,0,3,0,3,0,16,21,0,0,3,30,16,                         BDAT  76
     6 2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2/                                 BDAT  77
      DATA LIN(1,7),LIN(2,7),LIN(3,7),LIN(4,7),LIN(5,7),                BDAT  78
     1 LIN(6,7),LIN(7,7),LIN(8,7),LIN(9,7),LIN(10,7),LIN(11,7),         BDAT  79
     2 LIN(12,7),LIN(13,7),LIN(14,7),LIN(15,7),LIN(16,7),LIN(17,7),     BDAT  80
     3 LIN(21,7),LIN(22,7),LIN(23,7),LIN(24,7),LIN(25,7),               BDAT  81
     4 LIN(26,7),LIN(27,7),LIN(28,7),LIN(29,7),LIN(30,7),               BDAT  82
     5 LIN(31,7),LIN(32,7),LIN(33,7),LIN(34,7),LIN(35,7),LIN(36,7),     BDAT  83
     6 LIN(37,7),LIN(38,7),LIN(39,7),LIN(40,7),LIN(41,7),LIN(42,7)/     BDAT  84
     7 22,3,3,1,15,10,3,2,3,1,16,21,0,0,3,40,16,3,3,3,2,2,2,2,2,2,2,2,BDAT  85
     8 2,2,2,1,1,1,1,1,1,1/                                             BDAT  86
      DATA LIN(1,8),LIN(2,8),LIN(3,8),LIN(4,8),LIN(5,8),LIN(6,8),       BDAT  87
     1 LIN(7,8),LIN(8,8),LIN(9,8),LIN(10,8),LIN(11,8),LIN(12,8),        BDAT  88
     2 LIN(13,8),LIN(14,8),LIN(15,8),LIN(16,8),LIN(17,8),               BDAT  89
     3 LIN(21,8),LIN(22,8),LIN(23,8),LIN(24,8),LIN(25,8),               BDAT  90
     4 LIN(26,8),LIN(27,8),LIN(28,8),                                   BDAT  91
     4 LIN(29,8),LIN(30,8),LIN(31,8),LIN(32,8),LIN(33,8),LIN(34,8),     BDAT  92
     5 LIN(35,8),LIN(36,8),LIN(37,8),LIN(38,8),LIN(39,8),LIN(40,8)/     BDAT  93
     6 20,8,12,6,20,0,1,0,0,0,27,37,4,3,3,60,27,3,3,3,3,3,3,3,3,3,3,3, BDAT  94
     7 3,3,3,3,3,3,3,3,3/                                               BDAT  95
      DATA LIN(1,9),LIN(2,9),LIN(3,9),LIN(4,9),LIN(5,9),                BDAT  96
     1 LIN(6,9),LIN(7,9),LIN(8,9),LIN(9,9),LIN(10,9),                   BDAT  97
     2 LIN(11,9),LIN(12,9),LIN(13,9),LIN(14,9),LIN(15,9),LIN(16,9),     BDAT  98
     3 LIN(17,9),LIN(21,9),LIN(22,9),LIN(23,9),LIN(24,9),LIN(25,9),     BDAT  99
     4 LIN(26,9),LIN(27,9),LIN(28,9),LIN(29,9),LIN(30,9),LIN(31,9),     BDAT 100
     5 LIN(32,9),LIN(33,9),LIN(34,9),LIN(35,9),LIN(36,9),LIN(37,9),     BDAT 101
     6 LIN(38,9),LIN(39,9),LIN(40,9)/                                   BDAT 102
     7 20,8,12,6,20,8,1,0,0,0,27,37,4,3,3,68,27,4,4,4,4,4,4,4,4,       BDAT 103
     8 3,3,3,3,3,3,3,3,3,3,3,3/                                         BDAT 104
      DATA LIN(1,10),LIN(2,10),LIN(3,10),LIN(4,10),LIN(5,10),           BDAT 105
     1 LIN(6,10),LIN(7,10),LIN(8,10),LIN(9,10),LIN(10,10),              BDAT 106
     2 LIN(11,10),LIN(12,10),LIN(13,10),LIN(14,10),LIN(15,10),          BDAT 107
     3 LIN(16,10),LIN(17,10),LIN(21,10),LIN(22,10),LIN(23,10),          BDAT 108
     4 LIN(24,10),LIN(25,10),LIN(26,10),LIN(27,10),LIN(28,10),          BDAT 109
     5 LIN(29,10),LIN(30,10)/                                           BDAT 110
     5 10,4,6,4,10,0,1,0,0,0,4,64,28,15,4,30,0,3,3,3,3,3,3,3,3,3/      BDAT 111
      DATA LIN(1,11),LIN(2,11),LIN(3,11),LIN(4,11),LIN(5,11),           BDAT 112
     1 LIN(6,11),LIN(7,11),LIN(8,11),LIN(9,11),LIN(10,11),              BDAT 113
     2 LIN(11,11),LIN(12,11),LIN(13,11),LIN(14,11),LIN(15,11),          BDAT 114
     3 LIN(16,11),LIN(17,11),LIN(21,11),LIN(22,11),LIN(23,11),          BDAT 115
     4 LIN(24,11),LIN(25,11),LIN(26,11),LIN(27,11),LIN(28,11),          BDAT 116
     5 LIN(29,11),LIN(30,11)/                                           BDAT 117
     5 10,4,6,4,10,4,1,0,0,0,4,64,28,15,4,34,0,4,4,4,4,3,3,3,3,3,3/    BDAT 118
C-----------------------------------------------------------------BDAT 119
C     AREA COORDINATES - LINEAR STRAIN TRIANGLE - ELEMENT TYPE 2,3      BDAT 120
C-----------------------------------------------------------------BDAT 121
      DATA L(1,6),L(2,6),L(3,6),L(1,7),L(2,7),L(3,7),L(1,8),L(2,8),     BDAT 122
     1 L(3,8),L(1,9),L(2,9),L(3,9),L(1,10),L(2,10),L(3,10),L(1,11),     BDAT 123
     1 L(2,11),L(3,11),L(1,12),L(2,12),L(3,12)/                         BDAT 124
```

```
     1 .797426985353087245,.101286507323456343,.101286507323456343       BDAT 125
     1,.101286507323456343,.797426985353087245,.101286507323456343       BDAT 126
     1,.101286507323456343,.101286507323456343,.797426985353087245       BDAT 127
     1,.597158717897698279E-01,.470142064105115082,.470142064105115082   BDAT 128
     1,.470142064105115082,.597158717897698279E-01,.470142064105115082   BDAT 129
     1,.470142064105115082,.470142064105115082,.597158717897698279E-01   BDAT 130
     1,.333333333333333329,.333333333333333329,.333333333333333329/      BDAT 131
C-----------------------------------------------------------------------BDAT 132
C     LOCAL COORDINATES - LINEAR STRAIN QUADRILATERAL - ELEM TYPE 4, 5   BDAT 133
C-----------------------------------------------------------------------BDAT 134
      DATA L(1,13),L(2,13),L(1,14),L(2,14),L(1,15),L(2,15),              BDAT 135
     1 L(1,16),L(2,16),L(1,17),L(2,17),L(1,18),L(2,18),                  BDAT 136
     1 L(1,19),L(2,19),L(1,20),L(2,20),L(1,21),L(2,21)/                  BDAT 137
     1 -0.774596669241483,-0.774596669241483,                            BDAT 138
     1  0.774596669241483,-0.774596669241483,                            BDAT 139
     1  0.774596669241483, 0.774596669241483,                            BDAT 140
     1 -0.774596669241483, 0.774596669241483,                            BDAT 141
     1  0.,-0.774596669241483,                                           BDAT 142
     1  0.774596669241483,0.,                                            BDAT 143
     1  0., 0.774596669241483,                                           BDAT 144
     1 -0.774596669241483,0.,                                            BDAT 145
     1  0.,0./                                                           BDAT 146
C-----------------------------------------------------------------------BDAT 147
C     AREA COORDINATES - CUBIC STRAIN TRIANGLE - ELEMENT TYPE 6,7        BDAT 148
C-----------------------------------------------------------------------BDAT 149
      DATA L(1,22),L(2,22),L(3,22),L(1,23),L(2,23),L(3,23),L(1,24),      BDAT 150
     1 L(2,24),L(3,24),L(1,25),L(2,25),L(3,25),L(1,26),L(2,26),L(3,26),  BDAT 151
     1 L(1,27),L(2,27),L(3,27),L(1,28),L(2,28),L(3,28),L(1,29),          BDAT 152
     1 L(2,29),L(3,29)/                                                  BDAT 153
     1 0.898905543365938,0.050547228317031,0.050547228317031,            BDAT 154
     1 0.050547228317031,0.898905543365938,0.050547228317031,            BDAT 155
     1 0.050547228317031,0.050547228317031,0.898905543365938,            BDAT 156
     1 0.658861384496478,0.170569307751761,0.170569307751761,            BDAT 157
     1 0.170569307751761,0.658861384496478,0.170569307751761,            BDAT 158
     1 0.170569307751761,0.170569307751761,0.658861384496478,            BDAT 159
     1 0.081414823414554,0.459292588292723,0.459292588292723,            BDAT 160
     1 0.459292588292723,0.081414823414554,0.459292588292723/            BDAT 161
      DATA L(1,30),L(2,30),L(3,30),L(1,31),L(2,31),L(3,31),              BDAT 162
     1 L(1,32),L(2,32),L(3,32),L(1,33),L(2,33),L(3,33),L(1,34),L(2,34),  BDAT 163
     1 L(3,34),L(1,35),L(2,35),L(3,35),L(1,36),L(2,36),L(3,36),          BDAT 164
     1 L(1,37),L(2,37),L(3,37)/                                          BDAT 165
     1 0.459292588292723,0.459292588292723,0.081414823414554,            BDAT 166
     1 0.008394777409958,0.728492392955404,0.263112829634638,            BDAT 167
     1 0.008394777409958,0.263112829634638,0.728492392955404,            BDAT 168
     1 0.263112829634638,0.008394777409958,0.728492392955404,            BDAT 169
     1 0.728492392955404,0.008394777409958,0.263112829634638,            BDAT 170
     1 0.728492392955404,0.263112829634638,0.008394777409958,            BDAT 171
     1 0.263112829634638,0.728492392955404,0.008394777409958,            BDAT 172
     1 0.333333333333333,0.333333333333333,0.333333333333333/            BDAT 173
C-----------------------------------------------------------------------BDAT 174
C     WEIGHTS - LINEAR STRAIN TRIANGLE - ELEMENT TYPE 2,3                BDAT 175
C-----------------------------------------------------------------------BDAT 176
      DATA W(6),W(7),W(8),W(9),W(10),W(11),W(12)/                        BDAT 177
     1 .062969590272413570,.062969590272413570,.062969590272413570,     BDAT 178
     1 .066197076394253089,.066197076394253089,.066197076394253089,     BDAT 179
     1 .112499999999999996/                                              BDAT 180
C-----------------------------------------------------------------------BDAT 181
C     WEIGHTS - LINEAR STRAIN QUADRILATERAL - ELEMENT TYPE 4,5           BDAT 182
C-----------------------------------------------------------------------BDAT 183
      DATA W(13),W(14),W(15),W(16),W(17),W(18),W(19),W(20),W(21)/        BDAT 184
     1 0.30864197530864,0.30864197530864,                                BDAT 185
     1 0.30864197530864,0.30864197530864,                                BDAT 186
     1 0.49382716049383,0.49382716049383,                                BDAT 187
     1 0.49382716049383,0.49382716049383,0.79012345679012/               BDAT 188
C-----------------------------------------------------------------------BDAT 189
C     WEIGHTS - CUBIC STRAIN TRIANGLE - ELEMENT TYPE 6,7                 BDAT 190
```



```
C-----------------------------------------------------------------------BDAT 191
      DATA W(22),W(23),W(24),W(25),W(26),W(27),W(28),W(29),              BDAT 192
     1 W(30),W(31),W(32),W(33),W(34),W(35),W(36),W(37)/                  BDAT 193
     1 .016229248811599,.016229248811599,.016229248811599,              BDAT 194
     1 .051608685267359,.051608685267359,.051608685267359,              BDAT 195
     1 .047545817133642,.047545817133642,.047545817133642,              BDAT 196
     1 .013615157087217,.013615157087217,.013615157087217,              BDAT 197
     1 .013615157087217,.013615157087217,.013615157087217,              BDAT 198
     1 .072157803838893/                                                BDAT 199
C-----------------------------------------------------------------------BDAT 200
C     ONE-DIMENSIONAL INTEGRATION                                        BDAT 201
C-----------------------------------------------------------------------BDAT 202
      DATA POSSP(1),POSSP(2),POSSP(3),POSSP(4),POSSP(5)/                 BDAT 203
     1 -0.906179845938664,-0.538469310105683,0.0,                        BDAT 20
     1 0.538469310105683,0.906179845938664/                              BDAT 20
      DATA WEIGP(1),WEIGP(2),WEIGP(3),WEIGP(4),WEIGP(5)/                 BDAT 206
     1 0.236926885056189,0.478628670499366,0.568888888888889,            BDAT 207
     1 0.478628670499366,0.236926885056189/                              BDAT 208
      END                                                                BDAT 209
```

Element type information

BDAT 43–47	:	1 – 3-noded bar†	(2-D)
BDAT 48–52	:	2 – 6-noded LST	(2-D)
BDAT 53–57	:	3 – 6-noded LST	(2-D consolidation)
BDAT 58–63	:	4 – 8 noded quadrilateral†	(2-D)
BDAT 64–69	:	5 – 8 noded quadrilateral†	(2-D consolidation)
BDAT 70–77	:	6 – 15-noded CuST	(2-D)
BDAT 78–86	:	7 – 22-noded CuST	(2-D consolidation)
BDAT 87–95	:	8 – 20-noded brick†	(3-D)
BDAT 96–104	:	9 – 20-noded brick†	(3-D consolidation)
BDAT 105–111	:	10 – 10-noded tetrahedra†	(3-D)
BDAT 112–118	:	11 – 10-noded tetrahedra†	(3-D consolidation)

BDAT 122–131 : area co-ordinates for LST (type = 2, 3).
BDAT 135–146 : local co-ordinates for element types 4 and 5.
BDAT 150–173 : area co-ordinates for CuST (type = 6, 7).
DBAT 177–180 : weights for LST (type = 2, 3).
BDAT 184–188 : weights for element types 4 and 5.
BDAT 192–199 : weights for CuST (type = 6, 7).
BDAT 203–208 : local co-ordinates and weights for one-dimensional integration along edges of 2-D elements.

Some of the arrays used in the geometry part of the program are required for the *main* part of the program, but not all of them. Also a substantial number of additional arrays are required for the main part of the program. As the arrays are dimensioned psuedo-dynamically it is necessary to move some of the arrays used in the geometry part of the program to fill the gaps left by the outgoing arrays and the gaps due to arbitrary size allocation. The arrays to be used in the

† These elements are not implemented in the program presented here (but see Appendix E).

main part of the program are allocated store by starting indexes being set up in array G. Then the arrays which have already been formed are moved to their new locations. This is executed by routine **SHFTIB**.

Routine SHFTIB

```
      SUBROUTINE SHFTIB(IW6,IA,IB,N)                                    SHFT   1
C*********************************************************************SHFT   2
C     ROUTINE TO SHIFT AN INTEGER ARRAY BACKWARDS                       SHFT   3
C*********************************************************************SHFT   4
      DIMENSION IA(N),IB(N)                                             SHFT   5
      COMMON /DEBUGS/ ID1,ID2,ID3,ID4,ID5,ID6,ID7,ID8,ID9,ID10          SHFT   6
C                                                                       SHFT   7
      DO 10 I=1,N                                                       SHFT   8
      J=N+1-I                                                           SHFT   9
   10 IA(J)=IB(J)                                                       SHFT  10
C                                                                       SHFT  11
      IF(ID9.EQ.0)RETURN                                                SHFT  12
      WRITE(IW6,900)N,IA                                                SHFT  13
  900 FORMAT(/1X,9HNUMBER = ,I6/(20I6/))                                SHFT  14
      RETURN                                                            SHFT  15
      END                                                               SHFT  16
```

SHFT 8 : to shift an INTEGER array with N elements to the right; loop on all array elements.
SHFT 9 : shift last term first, to avoid over-writing.
SHFT 10 : shift (transfer) array element.

The variables (or parameters) which have not been encountered until now are described. These govern the size of the arrays. They vary from element type to element type. If different element types are mixed then the size of the variables is defined as the largest for the mixed group of element types.

Values of NDMX, NPMX, LV, NIP, NL and MDFE are obtained as the maximum values for the different element types present in the mesh.

NDMX – maximum number of displacement nodes in any element.
NPMX – maximum number of pore pressure nodes in any element.
LV – maximum number of displacement nodes along a side (edge).
NL – maximum number of area co-ordinates.
MDFE – maximum number of d.o.f. in any element.
NB – maximum number of columns in the B matrix.
KES – maximum number of entries in the upper triangular element stiffness matrix.
ICT – The number of 'consolidation' elements in the mesh.
NVPN – The maximum number of variables at any node
(= NDIM for drained/undrained analyses)
(= NDIM + 1 for consolidation analyses; the additional variable being the excess pore pressure).

These parameters are determined in routine **MAXVAL**, which scans the different element types present in the mesh and selects the largest value for each parameter. Not all arrays are allocated store psuedo-dynamically. Some arrays have fixed dimensions and these reside in named COMMONs.

Routine MAXVAL

```
      SUBROUTINE MAXVAL(IW6,KLT,LTZ,NDIM,NVRN,NDMX,NPMX,NIP,            MXVL   1
     1 NS,NB,NL,NPT,NSP,NPR,NMT,MDFE,KES,NVPN,LV,MXEN,MXLD,MXFXT)       MXVL   2
C*********************************************************************MXVL   3
C     SETS MAXIMUM VALUES AND SIZES OF SOME ARRAYS                      MXVL   4
C*********************************************************************MXVL   5
      DIMENSION KLT(LTZ)                                                MXVL   6
      COMMON /ELINF / LINFO(50,15)                                      MXVL   7
      COMMON /PARS  / PYI,ALAR,ASMVL,ZERO                               MXVL   8
C-------------------------------------------------------------------MXVL   9
C     MXEN,MXLD - SIZE OF ARRAYS IN COMMON BLOCKS PRSLD,PRLDI           MXVL  10
C     MXLD - MAXIMUM NUMBER OF ELEMENT EDGES WITH PRESSURE LOADING      MXVL  11
C     MXEN - MAXIMUM NUMBER OF DISPLACEMENT NODES ALONG AN EDGE x 2     MXVL  12
C     MXFXT- MAXIMUM NUMBER OF FIXITIES (SIZE OF ARRAYS MF,TF,DXYT)     MXVL  13
C-------------------------------------------------------------------MXVL  14
      MXEN=10                                                           MXVL  15
      MXLD=100                                                          MXVL  16
      MXFXT=200                                                         MXVL  17
C----------SIZE OF MATERIAL PROPERTIES (PR) AND TYPE (NTY) ARRAYS       MXVL  18
      NPR=10                                                            MXVL  19
      NMT=10                                                            MXVL  20
C----------ONE-DIMENSIONAL INTEGRATION - NUMBER OF SAMPLING POINTS      MXVL  21
      NSP=5                                                             MXVL  22
C----------NS - SIZE OF D-MATRIX                                        MXVL  23
      NS=NVRN                                                           MXVL  24
C-------------------------------------------------------------------MXVL  25
C     NVPN - MAXIMUM NUMBER OF D.O.F. IN ANY NODE                       MXVL  26
C     ADD 1 (FOR PORE-PRESSURE VARIABLE)                                MXVL  27
C           IF CONSOLIDATION ELEMENTS ARE PRESENT                       MXVL  28
C-------------------------------------------------------------------MXVL  29
      ICT=0                                                             MXVL  30
      DO 15 LT=1,LTZ                                                    MXVL  31
      KC=KLT(LT)                                                        MXVL  32
      GOTO(15,15,12,15,12,15,12,15,12,15,12),LT                         MXVL  33
      GOTO 15                                                           MXVL  34
   12 ICT=ICT+KC                                                        MXVL  35
   15 CONTINUE                                                          MXVL  36
      NVPN=NDIM                                                         MXVL  37
      IF(ICT.NE.0)NVPN=NDIM+1                                           MXVL  38
C-------------------------------------------------------------------MXVL  39
C     MAXIMUM VALUES OF NDMX,NPMX,LV,NIP,NL,MDFE                        MXVL  40
C     FOR ANY ELEMENT IN MESH                                           MXVL  41
C-------------------------------------------------------------------MXVL  42
      NDMX=0                                                            MXVL  43
C-------------------------------------------------------------------MXVL  44
C     IN THE ABSENCE OF ANY CONSOLIDATION ELEMENTS IN THE MESH          MXVL  45
C     NPMX WILL REMAIN 0. IN ORDER TO PREVENT ARRAYS BEING SETUP        MXVL  46
C     WITH ZERO DIMENSIONS (IN ROUTINE MAIN2) NPMX IS SET TO 1          MXVL  47
C-------------------------------------------------------------------MXVL  48
      NPMX=1                                                            MXVL  49
      LV=0                                                              MXVL  50
      NIP=0                                                             MXVL  51
      NL=0                                                              MXVL  52
      MDFE=0                                                            MXVL  53
C                                                                       MXVL  54
      DO 30 LT=1,LTZ                                                    MXVL  55
      IF(KLT(LT).EQ.0)GOTO 30                                           MXVL  56
      IF(NDMX.LT.LINFO(5,LT))NDMX=LINFO(5,LT)                           MXVL  57
```

```
      IF(NPMX.LT.LINFO(6,LT))NPMX=LINFO(6,LT)                          MXVL  58
      IF(LV.LT.LINFO(7,LT))LV=LINFO(7,LT)                              MXVL  59
      IF(NIP.LT.LINFO(11,LT))NIP=LINFO(11,LT)                          MXVL  60
      IF(NL.LT.LINFO(15,LT))NL=LINFO(15,LT)                            MXVL  61
      IF(MDFE.LT.LINFO(16,LT))MDFE=LINFO(16,LT)                        MXVL  62
   30 CONTINUE                                                         MXVL  63
C-----------------------------------------------------------------------MXVL  64
C     NB  - NUMBER OF COLUMNS IN B - MATRIX                            MXVL  65
C     KES - SIZE OF UPPER TRIANGULAR ELEMENT STIFFNESS MATRIX ES       MXVL  66
C     LV  - MAXIMUM NUMBER OF DISPLACEMENT NODES ALONG ELEMENT EDGE    MXVL  67
C-----------------------------------------------------------------------MXVL  68
      NB=NDIM*NDMX                                                     MXVL  69
      KES=MDFE*(MDFE+1)/2                                              MXVL  70
      LV=LV+2                                                          MXVL  71
      NPT=LV                                                           MXVL  72
CC    WRITE(IW6,900)NDIM,NVPN,NPMX,LV,NIP,NL,MDFE                      MXVL  73
CC900 FORMAT(/1X,4HNDIM,I6,2X,4HNVPN,I6,2X,4HNPMX,I6,2X,2HLV,I6,       MXVL  74
CC   1 2X,3HNIP,I6,2X,2HNL,I6,2X,4HMDFE,I6)                           MXVL  75
CC    WRITE(IW6,910)NDMX,NB,KES,NPT                                    MXVL  76
CC910 FORMAT(/1X,4HNDMX,I6,2X,2HNB,I6,2X,3HKES,I6,2X,3HNPT,I6)         MXVL  77
      RETURN                                                           MXVL  78
      END                                                              MXVL  79
```

MXVL 15–17 : sizes of load/fixity arrays.
- MXEN – maximum no. of displacement nodes along edge X 2 (size of array in named COMMON PRSLD and PRLDI).
- MXLD – maximum no. of element sides with applied pressure loads.
- MXFXT – maximum no. of fixities (size of arrays MF, TF and DXYT in named COMMON FIX).

MXVL 19–20 : size of material properties' array.
- MPR – maximum number of properties per material.
- MPT – maximum number of different material zones.

MXVL 22–24 : NSP – number of sampling point in one-dimensional numerical integration along element side.
- NS – size of D matrix (= number of stress/strain components).

MXVL 31 : loop on all element types.

MXVL 35 : update count of consolidation elements.

MXVL 37 : maximum number of variables at any node.

MXVL 38 : add pore pressure variable if consolidation elements are present.

MXVL 43–53 : zero element type dependent parameters.

MXVL 55 : loop on all element types.

MXVL 56–62 : get the maximum value of the following parameters for element types in current mesh.
- NDMX – number of displacement nodes.
- NPMX – number of pore pressure nodes.
- LV – number of displacement nodes along side (at this stage excluding nodes at either end).
- NIP – number of integration points.

- NL – number of local/area co-ordinates.
- MDFE – number of d.o.f. in element.

MXVL 69–72 : calculate sizes.
- NB – number of columns in B matrix.
- KES – size (number of terms) of upper triangle of element stiffness matrix.

7

In Situ Stresses

7.1 INTRODUCTION

Chapter 5 dealt with how one could set up the *in situ* stresses from field data or from laboratory measurements of samples. This chapter is about setting up the *in situ* stresses for starting a finite element analysis. In addition to specifying the stresses, the user has to specify the *in situ* boundary conditions and stresses acting along any unrestrained boundary. A check is carried out to ensure that the element stresses and the external loads are in equilibrium at the *in situ* stage.

Fig. 7.1 shows the subroutine hierarchy with routine CPW acting as the main control routine. Section 7.2 gives a brief explanation of the subroutines listed in this chapter.

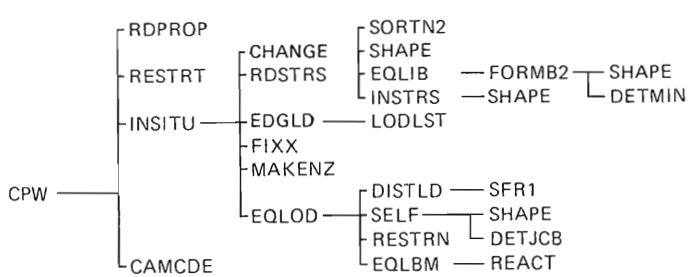


Fig. 7.1 – Subroutine hierarchy for *in situ* part of program

Section 7.3 gives the list of principal arrays which have been allocated store psuedo-dynamically in array G, as explained in section 4.3.2. Frequent reference is made to these arrays in Chapters 7 and 8.

Section 7.4 lists the routine CPW, which calls other control routines to carry out various tasks. Routine RDPROP reads the control parameters for the analysis and the material properties in section 7.5.

Routine RESTRT reads results written on a magnetic tape or disk file from a previous run if the analysis is being restarted. It is appropriate to discuss the stop–restart facility at the end of the analysis, and therefore it is dealt with in section 8.14.

In section 7.7, routine INSITU calls routine RDSTRS to read the *in situ* stresses. Routine EQLIB calculates the nodal loads equivalent to the *in situ* stresses. In section 7.8 the external pressure loads which should be in equilibrium with the *in situ* stresses are stored by EDGLD. The boundary conditions are read by routine FIXX. In section 7.9, routine EQLOD carries out an equilibrium check to ensure that the specified pressure loads are in equilibrium with the *in situ* stresses.

7.2 SUBROUTINE LIST

Fig. 7.1 shows the subroutine hierarchy, and here follows a brief explanation of each subroutine.

CPW – control routine delegates tasks of setting up the *in situ* stresses to routine INSITU.

RDPROP – reads control parameters for analysis (i.e. no. of increment blocks, type of analysis) and material properties.

RESTRT – deals with stopping and restarting an analysis. For a restarted analysis, it reads results from a past run (see section 8.14).

INSITU – control routine reads the *in situ* stresses and the boundary conditions and checks that the *in situ* stresses are in equilibrium.

CHANGE – elements removed have their element type number negated (i.e. array LTYP). (See section 8.4.)

RDSTRS – reads the *in situ* stresses specified at *in situ* nodes and interpolates values at integration points.

SORTN2 – to find *in situ* node with larger y co-ordinate.

SHAPE – calculates the shape functions and derivatives w.r.t. local co-ordinates.

EQLIB – calculates nodal loads equivalent to element stresses.

FORMB2 – calculates **B** matrix.

DETMIN – calculates determinant and inverse of Jacobian **J**.

INSTRS – prints out *in situ* stresses at each integration point for all elements.

EDGLD – stores pressure loads.

LODLST – stores pressure loads.

FIXX – reads fixities along element sides and stores them node by node.

EQLOD – control routine for equilibrium check. Current stresses must be in equilibrium with current loading.
DISTLD – calculates nodal loads equivalent to stresses along boundary.
SFR1 – used in numerical integration along element boundary.
SELF – calculates nodal loads equivalent to body forces for each element.
DETJCB – calculates Jacobian **J** and its determinant.
RESTRN – interprets nodal fixities in terms of g.v.n. to identify variables with prescribed values.
EQLBM – carries out an equilibrium check. Compares nodal loads equivalent to element stresses with nodal loads due to boundary loading and self-weight and prints them out.
REACT – calculates reactions to earth for prescribed variables and prints them.
ZEROSB – routines to zero REAL and INTEGER arrays.

7.3 DEFINITION OF PRINCIPAL ARRAYS

The principal arrays are now categorised according to the purpose they serve.

7.3.1 Loads

P – incremental loads assembled from various sources form the Right-Hand Side (RHS) when the equations are solved.
PT – sum of all incremental loads.
PIB – loads for the incremental block from various sources†.
XYFT – sum total of all directly specified nodal point loads.
XYFIB – directly specified nodal point loads for increment block.
PCONI – nodal loads equivalent to *in situ* stresses.
PCOR – out-of-balance or correcting loads (= the difference between external loads and loads equivalent to internal stresses).
PEQT – nodal loads equivalent to current stresses.
PEXI – excavation loads due to removal of elements.
PEXIB – excavation for increment block due to removal of elements.
R – reactions at nodes which are restrained or which have prescribed displacements.
FT – nodal loads equivalent to stresses in an element.

7.3.2 Displacements

DI – incremental displacements/excess pore pressures.
DA – cumulative displacements/excess pore pressures.

† Pressure loads on boundaries, body forces and forces due to removal or addition of elements.

7.3.3 Geometry and transformation

XYZ – x, y and z co-ordinates of all nodes (z only for 3-D).
SHFN – displacement shape functions.
DS – derivatives of shape functions w.r.t. local co-ordinates.
CARTD – Cartesian derivatives of shape functions.
ELCOD – local array of co-ordinates of displacement nodes in element.
ELCODP – local array of co-ordinates of pore pressure nodes in element.
AA – pore pressure shape functions.

7.3.4 Stresses and strains

VARINT – stress parameters at all integration points.
STR – strains at all integration points.

7.3.5 Stiffness and flow matrices

D – stress–strain relationship (constitutive model).
B – displacement–strain matrix.
DB – **D** post-multiplied by **B**.
SS – upper triangular part of $\mathbf{B}^T\mathbf{DB}$ (element stiffness matrix).
ES (SG) – square element stiffness matrix.

7.3.6 Flow and coupling matrices

E – multiplies pore pressure to give pore pressure gradients.
PE – $\mathbf{kE}$
RN – $\mathbf{B}^T\mathbf{M}$
ETE – flow matrix $\int_V \mathbf{E}^T\mathbf{kE}/\gamma\omega \, \mathrm{d}(\mathrm{vol})$.
RLT – coupling matrix $\int_V \mathbf{B}^T\mathbf{m}\bar{\mathbf{N}} \, \mathrm{d}(\mathrm{vol})$.

7.3.7 Integer arrays

NCONN – list of nodes associated with each element.
MAT – material zone numbers for each element.
LTYP – element type numbers for each element.
MRELVV – user element numbers.
MREL – program element numbers.
NRELVV – user node numbers.
NREL – program node numbers.
NW – global variable numbers of first variable of each node.
NQ – number of d.o.f. of each node.
JEL – list of element changes (added/removed).
IDFX – identifier of free nodal d.o.f. from restraints (0 – free; 1 – fixed).
IFR – list of nodes currently in front (during solution).
NDEST – destination of nodes to front.
NDL – index to front region of stiffness terms.
NWL – local array of element pore pressure d.o.f.

NMOD – identifier of yielded elements (not used in this version).
NP1,NP2 – indexes to array NCONN of nodes at either end of element sides.

7.4 CONTROLLING ROUTINE

Routine **CPW** is the main controlling routine which instructs other control routines to carry out various tasks (see Fig. 7.1). Routine **RDPROP** reads the control parameters for the analysis and the material properties. Routine **RESTRT** reads results written to a magnetic tape or disk file from a previous run if the analysis is being restarted. Routine **RDSTRS** reads the *in situ* stresses, boundary conditions and loads which are acting before the analysis is started.

Routine CPW

```
      SUBROUTINE CPW(NN,NEL,NDF,NNOD1,NTPE,NIP,NVRS,                    CPW    1
     1 NVRN,NDIM,MUMAX,NDZ,IFRZ,NNZ,NDMX,NPMX,                          CPW    2
     2 NS,NB,NL,NPR,NMT,NPT,NSP,NPL,MDFE,KES,NVPN,                      CPW    3
     3 INXL,MXEN,MXLD,MXFXT,LV,MCORE,LINK1,NVTX,ND,MDZ,NEDZ,            CPW    4
     4 XYZ,DI,DA,VARINT,P,PT,PIB,REAC,PCOR,PEQT,XYFT,XYFIB,             CPW    5
     5 STR,PEXIB,PEXI,PCONI,D,ELCOD,DS,SHFN,CARTD,B,DB,FT,SS,ES,ELCODP, CPW    6
     6 E,PE,RN,AA,ETE,RLT,                                              CPW    7
     7 NCONN,MAT,LTYP,MRELVV,MREL,NRELVV,NREL,NW,NQ,                    CPW    8
     8 JEL,IDFX,NDEST,NP1,NP2,IFR,NDL,NWL,NMOD,                         CPW    9
     3 CIP,LL,V,FXYZ,PR,PDISLD,PRES,NTY,A,MFZ,                          CPW   10
     4 NOIB,TTIME,TGRAV,IUPD,ICOR,IDCHK,INCT)                           CPW   11
C*********************************************************************CPW   12
C                                                                       CPW   13
C     MAIN CONTROLLING ROUTINE - INSITU STRESSES                        CPW   14
C                                                                       CPW   15
C*********************************************************************CPW   16
      REAL L,LL                                                         CPW   17
      INTEGER TF                                                        CPW   18
C--------USE THE FOLLOWING STATEMENT AFTER CONVERTING PROGRAM TO DOUBLE  CPW   19
C--------PRECISION.  ARRAY A ALWAYS USES ONE NUMERIC STORAGE LOCATION    CPW   20
CC    REAL A                                                            CPW   21
      DIMENSION XYZ(NDIM,NN),DI(NDF),DA(NDF),VARINT(NVRS,NIP,NEL),      CPW   22
     1 P(NDF),PT(NDF),PIB(NDF),REAC(NDF),PCOR(NDF),PEQT(NDF),XYFT(NDF), CPW   23
     2 XYFIB(NDF),STR(NVRN,NIP,NEL),PEXIB(NDF),PEXI(NDF),PCONI(NDF)     CPW   24
      DIMENSION D(NS,NS),ELCOD(NDIM,NDMX),DS(NDIM,NDMX),SHFN(NDMX),     CPW   25
     1 CARTD(NDIM,NDMX),B(NS,NB),DB(NS,NB),FT(NDIM,NDMX),               CPW   26
     2 SS(NB,NB),ES(KES)                                               CPW   27
      DIMENSION ELCODP(NDIM,NPMX),E(NDIM,NPMX),PE(NDIM,NPMX),           CPW   28
     1 RN(NB),AA(NPMX),ETE(NPMX,NPMX),RLT(NB,NPMX)                     CPW   29
      DIMENSION NCONN(NTPE,NEL),MAT(NEL),LTYP(NEL),MRELVV(NEL),         CPW   30
     1 MREL(MUMAX),NRELVV(NN),NREL(NNZ),NW(NNOD1),NQ(NN),JEL(NEL),      CPW   31
     2 IDFX(NDF),NDEST(NN),NP1(NPL),NP2(NPL)                           CPW   32
      DIMENSION IFR(IFRZ),NDL(MDFE),NWL(NPMX),NMOD(NIP,NEL)             CPW   33
      DIMENSION CIP(NDIM),LL(NL),V(LV),FXYZ(NDIM),PR(NPR,NMT),          CPW   34
     1 PDISLD(NDIM,LV),PRES(NDIM,LV),NTY(NMT),A(MFZ)                    CPW   35
C                                                                       CPW   36
      COMMON /FLOW  / NPLAX                                             CPW   37
      COMMON /DATL  / L(4,100)                                          CPW   38
      COMMON /DATW  / W(100)                                            CPW   39
      COMMON /ELINF / LINFO(50,15)                                      CPW   40
      COMMON /FIX   / DXYT(4,200),MF(200),TF(4,200),NF                  CPW   41
      COMMON /PRSLD / PRESLD(10,100),LEDG(100),NDE1(100),NDE2(100),NLED CPW   42
      COMMON /PRLDI / PRSLDI(10,100),LEDI(100),NDI1(100),NDI2(100),ILOD CPW   43
      COMMON /DEVICE/ IR1,IR4,IR5,IW2,IW4,IW6,IW7,IW8,IW9               CPW   44
      COMMON /PARS  / PYI,ALAR,ASMVL,ZERO                               CPW   45
C-----------------------------------------------------------------------CPW   46
      LINK2=1                                                           CPW   47
      TTIME=ZERO                                                        CPW   48
      READ(IR5,*)IDCHK                                                  CPW   49
      WRITE(IW6,922)IDCHK                                               CPW   50
      IF(IDCHK.EQ.0)WRITE(IW6,930)                                      CPW   51
      IF(IDCHK.EQ.1)WRITE(IW6,935)                                      CPW   52
      IF(IDCHK.EQ.2)WRITE(IW6,940)                                      CPW   53
C----------IF ONLY TO TEST GEOMETRY DATA STOP HERE                     CPW   54
      IF(IDCHK.EQ.1)STOP                                                CPW   55
      IF(LINK1.EQ.LINK2) GO TO 1                                        CPW   56
      WRITE(IW6,904)LINK1,LINK2                                         CPW   57
      STOP                                                              CPW   58
C                                                                       CPW   59
    1 CALL ZEROR3(STR,NVRN,NIP,NEL)                                     CPW   60
CC    WRITE(IW6,910)LINK2                                               CPW   61
      WRITE(IW6,801)NN,NEL,NDF,NNOD1,NTPE,NIP,NVRS                      CPW   62
      WRITE(IW6,802)NDIM,MUMAX,NDZ,IFRZ,NNZ,NDMX,NPMX                   CPW   63
      WRITE(IW6,803)NS,NB,NL,NPR,NMT,NPT,NSP                            CPW   64
      WRITE(IW6,804)NPL,MDFE,KES,NVPN,INXL,MXEN,MXLD                    CPW   65
      WRITE(IW6,805)MXFXT,LV,MCORE,NVTX,ND                              CPW   66
C                                                                       CPW   67
  801 FORMAT(/1X,8HNN    = ,I5,3X,8HNEL   = ,I5,3X,8HNDF   = ,I5,      CPW   68
     1 3X,8HNNOD1 = ,I5,3X,8HNTPE  = ,I5,3X,8HNIP   = ,I5,              CPW   69
     2 3X,8HNVRS  = ,I5)                                                CPW   70
C                                                                       CPW   71
  802 FORMAT(/1X,8HNDIM  = ,I5,3X,8HMUMAX = ,I5,3X,8HNDZ   = ,I5,      CPW   72
     1 3X,8HIFRZ  = ,I5,3X,8HNNZ   = ,I5,3X,8HNDMX  = ,I5,              CPW   73
     2 3X,8HNPMX  = ,I5)                                                CPW   74
C                                                                       CPW   75
  803 FORMAT(/1X,8HNS    = ,I5,3X,8HNB    = ,I5,3X,8HNL    = ,I5,      CPW   76
     1 3X,8HNPR   = ,I5,3X,8HNMT   = ,I5,3X,8HNPT   = ,I5,             CPW   77
     2 3X,8HNSP   = ,I5)                                                CPW   78
C                                                                       CPW   79
  804 FORMAT(/1X,8HNPL   = ,I5,3X,8HMDFE  = ,I5,3X,8HKES  = ,I5,      CPW   80
     1 3X,8HNVPN  = ,I5,3X,8HINXL  = ,I5,3X,8HMXEN  = ,I5,             CPW   81
     2 3X,8HMXLD  = ,I5)                                                CPW   82
C                                                                       CPW   83
  805 FORMAT(/1X,8HMXFXT = ,I5,3X,8HLV    = ,I5,3X,8HMCORE = ,I5,      CPW   84
     1 3X,8HNVTX  = ,I5,3X,8HND    = ,I5//1X,120(1H*))                 CPW   85
C-----------------------------------------------------------------------CPW   86
C     ROUTINE TO READ CONTROL OPTIONS AND MATERIAL PROPERTIES           CPW   87
C-----------------------------------------------------------------------CPW   88
      CALL RDPROP(NPR,NMT,NPLAX,NMAT,NOIB,INCS,INCF,INCT,               CPW   89
     1   IPRIM,IUPD,ICOR,PR,NTY,NDIM)                                   CPW   90
C-----------------------------------------------------------------------CPW   91
C     STOP/START FACILITY                                               CPW   92
C-----------------------------------------------------------------------CPW   93
      CALL RESTRT(INCS,INCF,NN,NVTX,ND,NEL,NDF,NTPE,NIP,                CPW   94
     1   NVRS,NVRN,MUMAX,NNZ,NNOD1,NDIM,MDZ,NEDZ,NL,INXL,               CPW   95
     2   NCONN,LTYP,MRELVV,MREL,NRELVV,NREL,NW,NMOD,                    CPW   96
     3   XYZ,DA,VARINT,PCOR,XYFT,STR,PCONI,TTIME,TGRAV)                 CPW   97
C-----------------------------------------------------------------------CPW   98
C     SETUP IN-SITU STRESSES AND CHECK FOR EQUILIBRIUM                  CPW   99
C-----------------------------------------------------------------------CPW  100
      IF(INCS.EQ.1)CALL INSITU(NN,NEL,NDF,NNOD1,NTPE,NIP,NDIM,NVRS,     CPW  101
     1   MUMAX,NNZ,NDZ,NPL,NDMX,NS,NB,NL,LV,NPR,NMT,NPT,NSP,            CPW  102
     2   XYZ,DA,VARINT,P,PT,PCOR,PEQT,XYFT,PEXIB,PCONI,ELCOD,DS,SHFN,    CPW  103
     3   CARTD,B,FT,NCONN,MAT,LTYP,MRELVV,MREL,NREL,NW,NQ,JEL,IDFX,      CPW  104
     4   NP1,NP2,NMOD,CIP,LL,V,PR,PDISLD,PRES,NTY,                      CPW  105
     5   A,MFZ,INXL,MXEN,MXLD,MXFXT,TGRAV,IPRIM)                        CPW  106
C-----------------------------------------------------------------------CPW  107
C     ROUTINE TO PRINT CAM-CLAY STRESS STATE CODES                      CPW  108
C-----------------------------------------------------------------------CPW  109
      CALL CAMCDE(IW6)                                                  CPW  110
C                                                                       CPW  111
      RETURN                                                            CPW  112
CC900 FORMAT(80A1)                                                      CPW  113
CC903 FORMAT(1X,80A1)                                                   CPW  114
```

```
  904 FORMAT(//10X,32HERROR ---- LINK  CODE   MISMATCH,2I5)          CPW  113
CC910 FORMAT(/10X,12HLINK CODE  =,I5)                               CPW  114
CC917 FORMAT(I5)                                                    CPW  115
CC918 FORMAT(//1X,120(1H*))                                         CPW  116
  922 FORMAT(/1X,20HDATA CHECK OPTION = ,I5/)                       CPW  117
  930 FORMAT(1X,32HCOMPLETE ANALYSIS IS CARRIED OUT/)               CPW  118
  935 FORMAT(1X,30HONLY GEOMETRY DATA ARE CHECKED/)                 CPW  119
  940 FORMAT(1X,42HGEOMETRY DATA AND IN-SITU STRESSES CHECKED/)     CPW  120
      END                                                           CPW  121
```

CPW 47–48 : read flag to stop analysis. (Allows only part of the input data to be checked, without carrying out the complete analysis.)

CPW 54–55 : check link number (allowing for the possibility that the program can be split into two parts; geometry part and main part. This ensures correct linkage between the parts).

CPW 87–88 : read control parameters and material properties.

CPW 92–95 : stop–restart facility. Write information to a file in magnetic tape or in disk. This enables the analysis to be stopped and restarted.

CPW 99–104 : *in situ* stresses are set up and equilibrium checked at this stage.

CPW 108 : print out Cam-clay codes.

7.5 CONTROL PARAMETERS AND MATERIAL PROPERTIES

Routine **RDPROP** reads the control parameters for the analysis and also reads the material properties for the different material zones specified in the mesh.

Routine RDPROP

```
      SUBROUTINE RDPROP(NPR,NMT,NPLAX,NMAT,NOIB,INCS,INCF,INCT,     RDPR   1
     1 IPRIM,IUPD,ICOR,PR,NTY,NDIM)                                 RDPR   2
C*****************************************************************RDPR   3
C     READ CONTROL OPTIONS AND MATERIAL PROPERTIES                  RDPR   4
C*****************************************************************RDPR   5
                                                                    RDPR   6
      DIMENSION PR(NPR,NMT),NTY(NMT)                                RDPR   7
      COMMON /DEVICE/ IR1,IR4,IR5,IW2,IW4,IW6,IW7,IW8,IW9           RDPR   8
      COMMON /OUT   / IBC,IRAC,NVOS,NVOF,NMOS,NMOF,NELOS,NELOF,ISR  RDPR   9
C-----------------------------------------------------------------RDPR  10
C     ICOR - OPTION TO APPLY OUT-OF-BALANCE LOADS AS CORRECTING     RDPR  11
C            LOADS IN THE NEXT INCREMENT                            RDPR  12
C     ICOR = 0 - CORRECTING LOADS ARE NOT APPLIED                   RDPR  13
C     ICOR = 1 - CORRECTING LOADS ARE APPLIED                       RDPR  14
C-----------------------------------------------------------------RDPR  15
      ICOR=0                                                        RDPR  16
C                                                                   RDPR  17
      READ(IR5,*)NPLAX,NMAT,NOIB,INCS,INCF,IPRIM,IUPD,ISR           RDPR  18
      WRITE(IW6,922)NPLAX,NMAT,NOIB,INCS,INCF,IPRIM,IUPD,ISR        RDPR  19
      NOINC=INCF-INCS+1                                             RDPR  20
      IF(NOINC.GT.0)GOTO 5                                          RDPR  21
      WRITE(IW6,925)NOINC,INCS,INCF                                 RDPR  22
      STOP                                                          RDPR  23
C                                                                   RDPR  24
    5 CONTINUE                                                      RDPR  25
C-----------------------------------------------------------------RDPR  26
C     INCT - COUNTER OF INCREMENT NUMBER                            RDPR  27
```



```
      INCT=INCS-1                                                   RDPR  28
      IF(NDIM.NE.3)GOTO 8                                           RDPR  29
      WRITE(IW6,928)                                                RDPR  30
      GOTO 10                                                       RDPR  31
    8 IF(NPLAX.EQ.0)WRITE(IW6,930)                                  RDPR  32
      IF(NPLAX.EQ.1)WRITE(IW6,931)                                  RDPR  33
   10 CONTINUE                                                      RDPR  34
C-----------------------------------------------------------------RDPR  35
C     READ OUTPUT REDUCING OPTIONS. THIS OPERATES ON RECORDS R AND T2 RDPR 36
C-----------------------------------------------------------------RDPR  37
      READ(IR5,*)IBC,IRAC,NVOS,NVOF,NMOS,NMOF,NELOS,NELOF           RDPR  38
      WRITE(IW6,945)IBC,IRAC,NVOS,NVOF,NMOS,NMOF,NELOS,NELOF        RDPR  39
C-----------------------------------------------------------------RDPR  40
C     READ MATERIAL PROPERTIES                                      RDPR
C-----------------------------------------------------------------RDPR  42
      CALL ZERO2(PR,NPR,NMT)                                        RDPR  43
C                                                                   RDPR  44
      WRITE(IW6,932)                                                RDPR  45
      DO 20 I=1,NMAT                                                RDPR  46
      READ(IR5,*)II,NTY(II),(PR(JJ,II),JJ=1,NPR)                    RDPR  47
      WRITE(IW6,936)II,NTY(II),(PR(JJ,II),JJ=1,NPR)                 RDPR  48
   20 CONTINUE                                                      RDPR  49
      RETURN                                                        RDPR  50
  922 FORMAT(/                                                      RDPR  51
     1 10X,46HPROBLEM TYPE.....................................=,I5/ RDPR  52
     2 10X,46HNUMBER OF MATERIALS..............................=,I5/ RDPR  53
     3 10X,46HNUMBER OF INCREMENT BLOCKS.......................=,I5/ RDPR  54
     4 10X,46HSTARTING INCR NUMBER OF ANALYSIS.................=,I5/ RDPR  55
     5 10X,46HFINISHING INCR NUMBER OF ANALYSIS................=,I5/ RDPR  56
     6 10X,46HNUMBER OF PRIMARY ELEMENT CHANGES................=,I5/ RDPR  57
     7 10X,46HOPTION TO UPDATE COORDINATES.....................=,I5/ RDPR  58
     8 10X,46HOPTION TO STOP/RESTART ANALYSIS..................=,I5/ RDPR  59
     9 /120(1H*)/)                                                  RDPR  60
  925 FORMAT(/1X,29HERROR IN NO. OF INCREMENTS = ,I5,               RDPR  61
     1 4X,7HINCS = ,I5,4X,7HINCF = ,I5,2X,16H(ROUTINE RDPROP))      RDPR  62
  928 FORMAT(//1X,22H3-DIMENSIONAL ANALYSIS)                        RDPR  63
  930 FORMAT(//1X,21HPLANE STRAIN ANALYSIS)                         RDPR  64
  931 FORMAT(//1X,22HAXI-SYMMETRIC ANALYSIS)                        RDPR  65
  932 FORMAT(//24H MATERIAL PROPERTY TABLE                          RDPR  66
     1 /1X,23(1H-)                                                  RDPR  67
     2 //2X,8HMAT TYPE,5X,1H1,11X,1H2,11X,1H3,11X,1H4,11X,1H5,      RDPR  68
     3 11X,1H6,11X,1H7,11X,1H8,11X,1H9,11X,2H10/)                   RDPR  69
  936 FORMAT(1X,2I5,(10E12.4/))                                     RDPR  70
  945 FORMAT(//120(1H*)/                                            RDPR  71
     1 10X,46HOPTION TO PRINT BOUNDARY CONDITIONS.....=,I5/         RDPR
     2 10X,46HOPTION TO PRINT REACTIONS...............=,I5/         RDPR
     3 10X,46HSTARTING VERTEX NODE NUMBER FOR OUTPUT..=,I5/         RDPR  74
     4 10X,46HFINISHING VERTEX NODE NUMBER FOR OUTPUT.=,I5/         RDPR  75
     5 10X,46HSTARTING MIDSIDE NODE NUMBER FOR OUTPUT.=,I5/         RDPR  76
     6 10X,46HFINISHING MIDSIDE NODE NUMBER FOR OUTPUT=,I5/         RDPR  77
     7 10X,46HSTARTING ELEMENT NUMBER FOR OUTPUT......=,I5/         RDPR  78
     8 10X,46HFINISHING ELEMENT NUMBER FOR OUTPUT.....=,I5/         RDPR  79
     9 /120(1H*)/)                                                  RDPR  80
      END                                                           RDPR  81
```

RDPR 15 : any out-of-balance loads are **not** carried forward to next increment (could be user specified, if need be).

RDPR 17–18 : read and write control parameters for analysis.

RDPR 19–20 : check INCF $\geqslant$ INCS; otherwise stop.

RDPR 28 : counter of increments.

RDPR 32–33 : print analysis **type.**
NPLAX = 0, plane strain
= 1, axisymmetry.
RDPR 43 : zero material property array.
RDPR 46–48 : read in material properties.

7.6 *IN SITU* STRESSES AT INTEGRATION POINTS

Arrays are set up to store the displacements at the nodes and the current values of stresses and strains at the integration points. The *in situ* stresses have to be defined at the **integration points** at the beginning of the analysis and not at the nodes (see Fig. 7.2). For most problems the variation of stresses is linear with depth and is constant in the horizontal direction. For most horizontally-laid layers the stresses and strength are the same at any given depth. Therefore it is sufficient to specify the variation of stresses with depth. Hence the stresses are defined at selected depths, probably to define non-linear variation by a series of piecewise linear curves. These selected points are defined as *in situ* **nodes**. These *in situ* nodes serve as reference points from which the stresses are interpolated. These *in situ* nodes should span the entire *primary* mesh. An error message will be printed if elements lie outside this *in situ* region. These *in situ* nodes are not to be confused with the nodes of the finite element mesh.

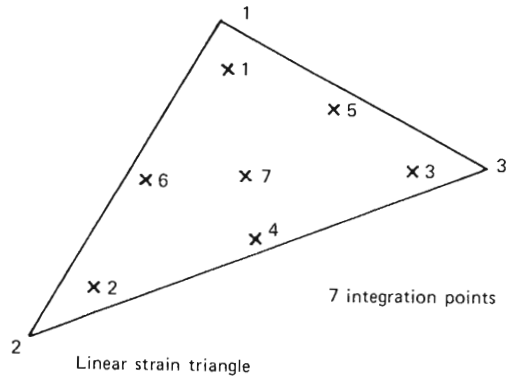
The stresses at all integration points are calculated by linear interpolation. A separate option is available to directly specify the *in situ* stresses at the integration points (for example where the ground has a slope and where the stresses are not the same in the horizontal direction) if the stress variation is such that the above simple option cannot deal with these specific situations.

Stress jumps usually in σ_h' can still be catered for by this option. *In situ* nodes A and B have the same co-ordinates. However, they have different horizontal stresses, as shown in Fig. 7.3. For clarity these are shown slightly apart. The vertical stress has to be continuous across CD and should have the same value for equilibrium to be satisfied.

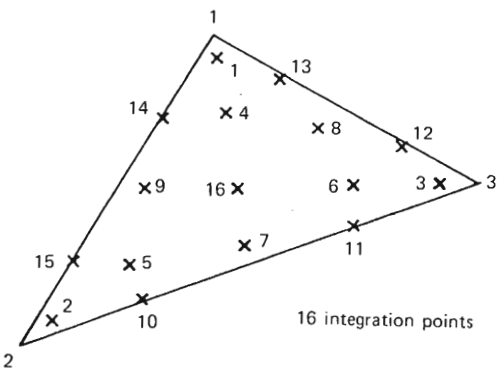
7.7 SETTING UP THE *IN SITU* STRESSES

Routine **RDSTRS** deals with the task of setting up the *in situ* stresses at the integration points. It is not sufficient just to set up the *in situ* stresses. The **boundary conditions** have to be specified either where element sides are restrained or where pressures act. These details are necessary to carry out an equilibrium check (see section 7.9) at the *in situ* stage.

The program carries out a check that the *in situ* stresses specified are in equilibrium with the loads (pressures) acting on the boundary. This loading is not to be confused with the loading applied during the analysis. (This is illustrated in some example problems in Chapter 9.) Routines other than RDSTRS are called, as shown in Fig. 7.1. The master control routine is **INSITU.**



Linear strain triangle



Cubic strain triangle

Fig. 7.2 – Integration scheme

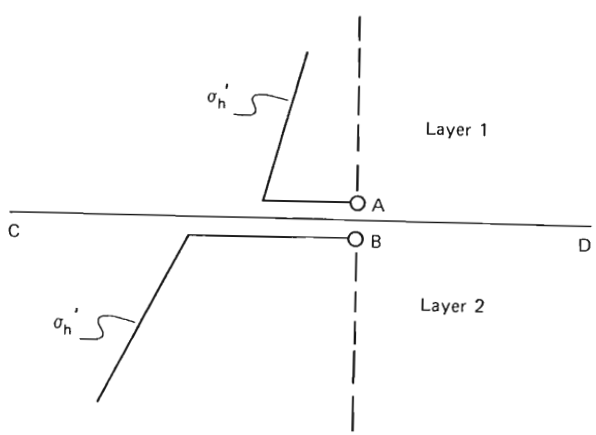


Fig. 7.3 – A jump in horizontal stresses is permissible

Routine INSITU

```
      SUBROUTINE INSITU(NN,NEL,NDF,NNOD1,NTPE,NIP,NDIM,NVRS,          INST   1
     1 MUMAX,NNZ,NDZ,NPL,NDMX,NS,NB,NL,LV,NPR,NMT,NPT,NSP,            INST   2
     2 XYZ,DA,VARINT,P,PT,PCOR,PEQT,XYFT,PEXIB,PCONI,                 INST   3
     3 ELCOD,DS,SHFN,CARTD,B,F,NCONN,MAT,LTYP,MRELVV,                 INST   4
     4 MREL,NREL,NW,NQ,JEL,IDFX,NP1,NP2,NMOD,CIP,LL,                  INST   5
     5 V,PR,PDISLD,PRES,NTY,A,MFZ,INXL,MXEN,MXLD,MXFXT,               INST   6
     6 TGRAVI,IPRIM)                                                  INST   7
C*********************************************************************INST   8
C     SETUP INSITU STRESSES AND CHECK FOR EQUILIBRIUM                 INST   9
C*********************************************************************INST  10
C                                                                     INST  11
      REAL LL                                                         INST  12
      INTEGER TF                                                      INST  13
C--------USE THE FOLLOWING STATEMENT AFTER CONVERTING PROGRAM TO DOUBLE INST  14
C--------PRECISION.  ARRAY A ALWAYS USES ONE NUMERIC STORAGE LOCATION   INST  15
CC    REAL A                                                          INST  16
      DIMENSION XYZ(NDIM,NN),DA(NDF),VARINT(NVRS,NIP,NEL),            INST  17
     1 P(NDF),PT(NDF),PCOR(NDF),PEQT(NDF),XYFT(NDF),                  INST  18
     2 PEXIB(NDF),PCONI(NDF)                                          INST  19
      DIMENSION ELCOD(NDIM,NDMX),DS(NDIM,NDMX),SHFN(NDMX),            INST  20
     1 CARTD(NDIM,NDMX),B(NS,NB),F(NDIM,NDMX)                         INST  21
      DIMENSION NCONN(NTPE,NEL),MAT(NEL),LTYP(NEL),MRELVV(NEL),       INST  22
     1 MREL(MUMAX),NREL(NNZ),NW(NNOD1),NQ(NN),JEL(NEL),               INST  23
     2 IDFX(NDF),NP1(NPL),NP2(NPL),NMOD(NIP,NEL)                      INST  24
      DIMENSION CIP(NDIM),LL(NL),V(LV),PR(NPR,NMT),                   INST  25
     1 PDISLD(NDIM,LV),PRES(NDIM,LV),NTY(NMT),A(MFZ)                  INST  26
      COMMON /DEVICE/ IR1,IR4,IR5,IW2,IW4,IW6,IW7,IW8,IW9             INST  27
      COMMON /FIX    / DXYT(4,200),MF(200),TF(4,200),NF               INST  28
      COMMON /PRSLD / PRESLD(10,100),LEDG(100),NDE1(100),NDE2(100),NLED INST  29
      COMMON /PARS   / PYI,ALAR,ASMVL,ZERO                            INST  30
      COMMON /OUT    / IBC,IRAC,NVOS,NVOF,NMOS,NMOF,NELOS,NELOF,ISR   INST  31
      COMMON /PRECSN/ NP                                              INST  32
C                                                                     INST  33
C----------CODE TO INDICATE STAGE OF THE ANALYSIS                     INST  34
      KSTGE=1                                                         INST  35
      CALL ZEROI1(JEL,NEL)                                            INST  36
C                                                                     INST  37
      IF(IPRIM.EQ.0) GO TO 28                                         INST  38
C-----------------------------------------------------------------    INST  39
C     READ  AND  REMOVE  ELEMENTS  TO  FORM  PRIMARY  MESH           INST  40
C-----------------------------------------------------------------    INST  41
      WRITE(IW6,907)                                                  INST  42
      READ(IR5,*)(JEL(J),J=1,IPRIM)                                   INST  43
      WRITE(IW6,920)(JEL(J),J=1,IPRIM)                                INST  44
C                                                                     INST  45
      CALL CHANGE(IW6,0,IPRIM,NN,NNOD1,NTPE,NIP,NEL,MUMAX,NNZ,NDF,NDIM, INST  46
     1   NVRS,NDMX,NL,NB,NS,NPR,NMT,NPT,NSP,NPL,XYZ,VARINT,P,PEXIB,   INST  47
     2   ELCOD,DS,SHFN,CARTD,B,F,NCONN,MAT,LTYP,MREL,NREL,             INST  48
     3   NW,JEL,NP1,NP2,MXEN,LL,PR,ZERO)                              INST  49
C-----------------------------------------------------------------    INST  50
C     INITIALISE PRESSURE LOADS                                       INST  51
C-----------------------------------------------------------------    INST  52
   28 NDIM1=NDIM+1                                                    INST  53
      CALL ZEROR1(PCONI,NDF)                                          INST  54
      CALL ZEROR2(PRESLD,MXEN,MXLD)                                   INST  55
      CALL ZEROI1(LEDG,MXLD)                                          INST  56
      CALL ZEROI1(NDE1,MXLD)                                          INST  57
      CALL ZEROI1(NDE2,MXLD)                                          INST  58
      CALL ZEROI1(MF,MXFXT)                                           INST  59
      CALL ZEROI2(TF,NDIM1,MXFXT)                                     INST  60
      CALL ZEROR2(DXYT,NDIM1,MXFXT)                                   INST  61
      CALL ZEROI2(NMOD,NIP,NEL)                                       INST  62
C-----------------------------------------------------------------    INST  63
C     SET  UP  IN-SITU  STRESS  SYSTEM
```



```
C-----------------------------------------------------------------    INST  64
      READ(IR5,*)KT,NI                                                INST  65
      WRITE(IW6,926)KT,NI                                             INST  66
C----------IF NI = 0 USE A DEFAULT VALUE OF 1 TO AVOID ARRAY SIZE OF 0. INST  67
      IF(NI.EQ.0)NI=1                                                 INST  68
C----------ALLOCATE STORE IN ARRAY A FOR SOME TEMPORARY ARRAYS        INST  69
      L1=NI*NP+1                                                      INST  70
      L2=L1+NVRS*NI*NP                                                INST  71
      L3=L2+NI                                                        INST  72
      L4=L3+NI                                                        INST  73
C                                                                     INST  74
      CALL RDSTRS(NN,NEL,NDF,NNOD1,MUMAX,NTPE,NIP,NVRS,NL,NB,NS,NPR,  INST  75
     1   NMT,NDIM,NDMX,KT,XYZ,VARINT,PEQT,ELCOD,DS,SHFN,              INST  76
     2   CARTD,B,F,NCONN,MAT,LTYP,MRELVV,MREL,NW,NMOD,                INST
     3   CIP,LL,PR,NTY,A(1),A(L1),A(L2),A(L3),NI)                     INST
C-----------------------------------------------------------------    INST  79
C     INITIALISE FIXED LOADS, TOTAL POINT LOADS AND TOTAL DISPLACEMENTS INST  80
C     NF - NUMBER OF FIXITIES                                         INST  81
C-----------------------------------------------------------------    INST  82
      NF=0                                                            INST  83
C                                                                     INST  84
      CALL ZEROR1(PCOR,NDF)                                           INST  85
      CALL ZEROR1(XYFT,NDF)                                           INST  86
      CALL ZEROR1(P,NDF)                                              INST  87
      CALL ZEROR1(DA,NDF)                                             INST  88
C-----------------------------------------------------------------    INST  89
C     READ LOADS IN EQUILIBRIUM WITH IN-SITU STRESSES                 INST  90
C-----------------------------------------------------------------    INST  91
      NLED=0                                                          INST  92
      TGRAVI=ZERO                                                     INST  93
      IF(KT.EQ.0)GO TO 62                                             INST  94
C                                                                     INST  95
      READ(IR5,*)NLODI,NFXI,TGRAVI                                    INST  96
      WRITE(IW6,952)NLODI,NFXI,TGRAVI                                 INST  97
C                                                                     INST  98
      IF(NLODI.EQ.0)GO TO 52                                          INST  99
      WRITE(IW6,960)                                                  INST 100
C                                                                     INST 101
      DO 50 KL=1,NLODI                                                INST 102
      READ(IR5,*)LNE,ND1,ND2,((PDISLD(ID,IV),ID=1,2),IV=1,NPT)        INST 103
      WRITE(IW6,964)LNE,ND1,ND2,((PDISLD(ID,IV),ID=1,2),IV=1,NPT)     INST 104
C                                                                     INST 105
      DO 100 IV=1,NPT                                                 INST 106
      DO 100 ID=1,NDIM                                                INST 107
      IDR=NDIM+1-ID                                                   INST
  100 PRES(ID,IV)=PDISLD(IDR,IV)                                      INST
C                                                                     INST
      DO 110 IV=1,NPT                                                 INST 111
      DO 110 ID=1,NDIM                                                INST 112
  110 PDISLD(ID,IV)=PRES(ID,IV)                                       INST 113
C                                                                     INST 114
      CALL EDGLD(IW6,NEL,NDIM,NTPE,NNZ,MUMAX,NPL,NCONN,LTYP,MREL,NREL, INST 115
     1   LNE,ND1,ND2,NP1,NP2,PDISLD,PRES,KL,NPT,1,MXLD)               INST 116
   50 CONTINUE                                                        INST 117
   52 IF(NFXI.EQ.0)GO TO 62                                           INST 118
C-----------------------------------------------------------------    INST 119
C     IN-SITU BOUNDARY CONDITIONS                                     INST 120
C-----------------------------------------------------------------    INST 121
      WRITE(IW6,930)                                                  INST 122
      CALL FIXX(IR5,IW6,NEL,NTPE,NDIM,NPL,LV,MUMAX,NNZ,NCONN,LTYP,    INST 123
     1   MREL,NREL,NP1,NP2,V,NFXI)                                    INST 124
C                                                                     INST 125
      CALL MAKENZ(NTPE,NEL,NN,NCONN,LTYP,NQ,INXL)                     INST 126
      CALL EQLOD(IW6,NN,NEL,NDF,NNOD1,NTPE,NDIM,MUMAX,NNZ,NDZ,NPR,NMT, INST 127
     1   NDMX,NL,NPL,NCONN,MAT,LTYP,MRELVV,MREL,NREL,NW,NQ,JEL,IDFX,  INST 128
     2   NP1,NP2,XYZ,P,PT,PCOR,PEQT,XYFT,PCONI,ELCOD,DS,SHFN,F,LL,PR,  INST 129
```

```
     3   NPT,NSP,MXEN,2,0,TGRAVI,IRAC,ZERO,KSTGE)                          INST 130
C                                                                          INST 131
   62 RETURN                                                               INST 132
  907 FORMAT(//1X,38HLIST  OF  REMOVED  ELEMENTS  TO  FORM ,               INST 133
     1 14H PRIMARY  MESH/1X,52(1H-)/)                                      INST 134
  920 FORMAT(20I6/)                                                        INST 135
  926 FORMAT(//10X,30HIN-SITU STRESS OPTION.....=,I10                      INST 136
     1       /10X,30HNUMBER OF IN-SITU NODES.....=,I10/)                   INST 137
  930 FORMAT(/1X,27HIN-SITU BOUNDARY CONDITIONS/1X,27(1H-)/)               INST 138
  952 FORMAT(/                                                             INST 139
     1 10X,46HNUMBER OF EDGES WITH PRESSURE LOAD.....=,I5/                 INST 140
     2 10X,46HNUMBER OF EDGES RESTRAINED.....=,I5/                         INST 141
     3 10X,46HIN-SITU GRAVITY ACCELERATION FIELD.....=,F8.1,2X,            INST 142
     4 1HG//)                                                              INST 143
  960 FORMAT(/1X,38HSPECIFIED NODAL VALUES OF SHEAR/NORMAL,                INST 144
     1 19H STRESSES (IN-SITU)/1X,57(1H-)/1X,4HELEM,                        INST 145
     2 1X,4HNDE1,2X,4HNDE2,2X,4HSHR1,8X,4HNOR1,8X,4HSHR2,8X,4HNOR2,        INST 146
     3 8X,4HSHR3,8X,4HNOR3,8X,4HSHR4,8X,4HNOR4,8X,4HSHR5,8X,4HNOR5/)       INST 147
  964 FORMAT(1X,3I4,10E12.4)                                               INST 148
      END                                                                  INST 149
```

INST 35 : zero list of element changes.
INST 37 : skip if no changes to the **initial** mesh.
INST 42–43 : read and write list of changes to initial mesh.
INST 45–48 : make all **removed** elements' **type** numbers negative in array LTYP.
INST 53–61 : zero all arrays for current analysis.
INST 65–66 : read and write *in situ* stress option and number of *in situ* nodes.
INST 70–73 : calculate pointers for some arrays in A for calculating *in situ* stresses (temporary usage).
INST 75–78 : calculate *in situ* stresses.
INST 83 : set counter of nodal fixities to zero.
INST 85–88 : zero loads/displacements' arrays.
INST 94 : skip if *in situ* stresses have been set to zero.
INST 96–97 : read and write the no. of loads/fixities to maintain equilibrium at *in situ* level.
INST 99 : skip if no pressure loads are applied.
INST 102 : loop to read pressure loads (which caused *in situ* stresses).
INST 103–104 : read and write pressure loads prescribed along element sides.
INST 106–113 : change sequence of pressures to suit storing.
INST 115–116 : enter pressure loads in PRESLD.
INST 117 : end of loop to read pressure loads.
INST 118 : skip if no prescribed fixities.
INST 123–124 : read sides which are restrained.
INST 126 : calculate d.o.f. of each node and total d.o.f. in mesh.
INST 127–130 : calculate loads equivalent to *in situ* stresses and carry out an equilibrium check at *in situ* stage.

7.7.1 Simulation of construction events

Simulation of a construction event (e.g. an embankment) is modelled by adding

a set of elements. To do this, these elements are 'removed' or 'inactivated' (they do not take part in the analysis until 'added' or 'reactivated') before the **first** increment. This is done by making the element type numbers (array LTYP) negative to identify the elements which have been removed. Routine **CHANGE** does this. These elements do not have any *in situ* stresses. *In situ* stresses are not assigned to elements not present at the beginning of the first increment. The *in situ* region need not enclose these elements. When it comes to setting up the *in situ* stresses, these elements are by-passed.

Routine **CHANGE** also calculates the implied loadings due to the removal of elements. To differentiate between the above two cases (when to – and when not to – calculate the implied loads) a flag IN is used in the argument list. Only when this is set to 1 are the implied loadings calculated. A detailed description is given in section 8.4.

7.7.2 Read *in situ* stresses

Routine **RDSTRS** deals with the task of setting up the *in situ* stresses at the integration points.

Routine RDSTRS

```
      SUBROUTINE RDSTRS(NN,NEL,NDF,NNOD1,MUMAX,NTPE,NIP,NVRS,NL,NB,NS,   RDST   1
     1 NPR,NMT,NDIM,NDMX,KT,XYZ,VARINT,PEQT,ELCOD,DS,SHFN,CARTD,         RDST   2
     2 B,FI,NCONN,MAT,LTYP,MRELVV,MREL,NW,NMOD,CIP,LL,PR,NTY,YI,         RDST   3
     3 VAR,NLI,NHI,NI)                                                   RDST   4
C*********************************************************************** RDST   5
C     SET  UP  IN-SITU  STRESSES                                         RDST   6
C*********************************************************************** RDST   7
      REAL L,LL                                                          RDST   8
      DIMENSION XYZ(NDIM,NN),VARINT(NVRS,NIP,NEL),PEQT(NDF)              RDST   9
      DIMENSION ELCOD(NDIM,NDMX),DS(NDIM,NDMX),SHFN(NDMX),               RDST  10
     1 CARTD(NDIM,NDMX),B(NS,NB),FI(NDIM,NDMX)                           RDST  11
      DIMENSION NCONN(NTPE,NEL),MAT(NEL),LTYP(NEL),MRELVV(NEL)           RDST  12
      DIMENSION MREL(MUMAX),NW(NNOD1),NMOD(NIP,NEL)                      RDST  13
      DIMENSION YI(NI),VAR(NVRS,NI),NLI(NI),NHI(NI)                      RDST  14
      DIMENSION CIP(NDIM),LL(NL),PR(NPR,NMT),NTY(NMT)                    RDST  15
      COMMON /PARS  / PYI,ALAR,ASMVL,ZERO                                RDST  16
      COMMON /DEVICE/ IR1,IR4,IR5,IW2,IW4,IW6,IW7,IW8,IW9                RDST  17
      COMMON /FLOW  / NPLAX                                              RDST  18
      COMMON /DATL  / L(4,100)                                           RDST  19
      COMMON /ELINF / LINFO(50,15)                                       RDST  20
C---------------------------------------------------------------------- RDST  21
C     ISTGE - CODE TO INDICATE STAGE OF THE ANALYSIS                     RDST  22
C---------------------------------------------------------------------- RDST  23
      ISTGE=1                                                            RDST  24
C---------------------------------------------------------------------- RDST  25
C     INITIALISE VARINT - INTEGRATION  POINT  VARIABLES                  RDST  26
C---------------------------------------------------------------------- RDST  27
      CALL ZERO3(VARINT,NVRS,NIP,NEL)                                    RDST  28
C---------------------------------------------------------------------- RDST  29
C     INITIALISE PEQT - CONTRIBUTION OF FORCES DUE TO ELEMENT IN-SITU    RDST  30
C     STRESSES                                                           RDST  31
C---------------------------------------------------------------------- RDST  32
      CALL ZERO1(PEQT,NDF)                                               RDST  33
      IF(KT.EQ.0) WRITE(IW6,904)                                         RDST  34
      IF(KT-1) 200,8,82                                                  RDST  35
C---------------------------------------------------------------------- RDST  36
C     READ NUMBER OF IN-SITU NODAL POINTS                                RDST  37
```

```
C-----------------------------------------------------------------------RDST  38
    8 WRITE(IW6,906)                                                    RDST  39
      DO 10 J=1,NI                                                      RDST  40
C-----------------------------------------------------------------------RDST  41
C     READ  NODE  COORDINATES  AND  VARIABLES                           RDST  42
C-----------------------------------------------------------------------RDST  43
      READ(IR5,*)IL,YI(IL),(VAR(JJ,IL),JJ=1,NVRS)                       RDST  44
   10 WRITE(IW6,910) IL,YI(IL),(VAR(JJ,IL),JJ=1,NVRS)                   RDST  45
C                                                                       RDST  46
      MI=NI-1                                                           RDST  47
      DO 20 IN=1,MI                                                     RDST  48
      N1=IN                                                             RDST  49
      N2=IN+1                                                           RDST  50
      Y1=YI(N1)                                                         RDST  51
      Y2=YI(N2)                                                         RDST  52
C                                                                       RDST  53
      CALL SORTN2(Y1,Y2,N1,N2,NMIN,NMAX)                                RDST  54
      NLI(IN)=NMIN                                                      RDST  55
      NHI(IN)=NMAX                                                      RDST  56
   20 CONTINUE                                                          RDST  57
C-----------------------------------------------------------------------RDST  58
C     LOOP  ON  ALL  GEOMETRY  MESH  ELEMENTS                           RDST  59
C-----------------------------------------------------------------------RDST  60
      DO 80 J=1,NEL                                                     RDST  61
      LT=LTYP(J)                                                        RDST  62
      IF(LT.LT.0)GOTO 80                                                RDST  63
CC    LT=IABS(LT)                                                       RDST  64
      JUS=MRELVV(J)                                                     RDST  65
      GO TO(80,22,22,22,22,22,22,22,22,22,22,22,80,80,80,80),LT         RDST  66
      WRITE(IW6,915)JUS,LT                                              RDST  67
      GOTO 80                                                           RDST  68
   22 KM=MAT(J)                                                         RDST  69
      NGP=LINFO(11,LT)                                                  RDST  70
      NDN=LINFO(5,LT)                                                   RDST  71
      INDX=LINFO(12,LT)                                                 RDST  72
      NAC=LINFO(15,LT)                                                  RDST  73
C                                                                       RDST  74
      DO 30 KN=1,NDN                                                    RDST  75
      NDE=NCONN(KN,J)                                                   RDST  76
      DO 30 ID=1,NDIM                                                   RDST  77
   30 ELCOD(ID,KN)=XYZ(ID,NDE)                                          RDST  78
C-----------------------------------------------------------------------RDST  79
C     LOOP  ON  ALL  INTEGRATION  POINTS                                RDST  80
C-----------------------------------------------------------------------RDST  81
      DO 60 IP=1,NGP                                                    RDST  82
C-----------------------------------------------------------------------RDST  83
C     CALCULATE  INTEGRATION  POINT  COORDINATES                        RDST  84
C-----------------------------------------------------------------------RDST  85
      IPA=IP+INDX                                                       RDST  86
      DO 35 IL=1,NAC                                                    RDST  87
   35 LL(IL)=L(IL,IPA)                                                  RDST  88
      CALL SHAPE(IW6,LL,NAC,DS,SHFN,NDIM,NDN,LT,1,JUS)                  RDST  89
C                                                                       RDST  90
      DO 40 ID=1,NDIM                                                   RDST  91
      SUM=ZERO                                                          RDST  92
C                                                                       RDST  93
      DO 38 I=1,NDN                                                     RDST  94
   38 SUM=SUM+SHFN(I)*ELCOD(ID,I)                                       RDST  95
   40 CIP(ID)=SUM                                                       RDST  96
      YY=CIP(2)                                                         RDST  97
C-----------------------------------------------------------------------RDST  98
C     SEARCH  FOR  RELEVANT  IN-SITU  LAYER                             RDST  99
C-----------------------------------------------------------------------RDST 100
      DO 45 JJJ=1,MI                                                    RDST 101
      NSM=NLI(JJJ)                                                      RDST 102
      NLA=NHI(JJJ)                                                      RDST 103
```



```
      YMIN=YI(NSM)                                                      RDST 104
      YMAX=YI(NLA)                                                      RDST 105
C                                                                       RDST 106
      IF(YY.LT.YMIN.OR.YY.GT.YMAX)GO TO 45                              RDST 107
      GO TO 48                                                          RDST 108
C                                                                       RDST 109
   45 CONTINUE                                                          RDST 110
      WRITE(IW6,950)JUS,IP                                              RDST 111
      GO TO 60                                                          RDST 112
C-----------------------------------------------------------------------RDST 113
C     DIRECT  INTERPOLATION  FROM  IN-SITU  MESH  NODES                 RDST 114
C-----------------------------------------------------------------------RDST 115
   48 DY=YI(JJJ)-YI(JJJ+1)                                              RDST 116
      YR=(YY-YMIN)/DY                                                   RDST 117
C                                                                       RDST 118
      DO 50 I=1,NVRS                                                    RDST 119
   50 VARINT(I,IP,J)=VAR(I,NSM)+(VAR(I,JJJ)-VAR(I,JJJ+1))*YR           RDST 120
CC    WRITE(IW6,951)J,IP,(VARINT(IU,IP,J),IU=1,NVRS)                    RDST 121
      KGO=NTY(KM)                                                       RDST 122
      GO TO(60,60,52,52,60,60),KGO                                      RDST 123
   52 P=(VARINT(1,IP,J)+VARINT(2,IP,J)+VARINT(3,IP,J))*0.333333         RDST 124
      PC=VARINT(NS+3,IP,J)                                              RDST 125
      IF(KGO.NE.3)GO TO 54                                              RDST 126
      PU=0.5*PC                                                         RDST 127
      GO TO 55                                                          RDST 128
   54 PU=PC/2.7182818                                                   RDST 129
   55 VARINT(NS+2,IP,J)=PR(3,KM)-PR(1,KM)*ALOG(P)-                      RDST 130
     1(PR(2,KM)-PR(1,KM))*ALOG(PU)                                      RDST 131
   60 CONTINUE                                                          RDST 132
   80 CONTINUE                                                          RDST 133
      GOTO 92                                                           RDST 134
C-----------------------------------------------------------------------RDST 135
C     DIRECT SPECIFICATION OF IN-SITU STRESSESS                         RDST 136
C-----------------------------------------------------------------------RDST 137
   82 IF(KT.NE.2)GO TO 92                                               RDST 138
      WRITE(IW6,955)                                                    RDST 139
C *** READ FOR ALL INTEGRATION POINTS                                   RDST 140
      DO 90 IM=1,NEL                                                    RDST 141
      READ(IR5,*)MUS                                                    RDST 142
      IL=MREL(MUS)                                                      RDST 143
      LT=LTYP(IL)                                                       RDST 144
      NGP=LINFO(11,LT)                                                  RDST 145
C                                                                       RDST 146
      DO 85 IP=1,NGP                                                    RDST 147
      READ(IR5,*)(VARINT(JJJ,IP,IL),JJJ=1,NVRS)                         RDST 148
   85 WRITE(IW6,960)(VARINT(JJJ,IP,IL),JJJ=1,NVRS)                      RDST 149
   90 CONTINUE                                                          RDST 150
C-----------------------------------------------------------------------RDST 151
C     CALCULATE EQUILIBRIUM LOADS FOR INSITU STRESSES                   RDST 152
C     ASSEMBLE ELEMENT CONTRIBUTION (FI) INTO PEQT                      RDST 153
C-----------------------------------------------------------------------RDST 154
   92 CR=1.                                                             RDST 155
      IF(NPLAX.EQ.1)CR=2.*PYI                                           RDST 156
C                                                                       RDST 157
      DO 100 J=1,NEL                                                    RDST 158
      LT=LTYP(J)                                                        RDST 159
      IF(LT.LE.0)GO TO 100                                              RDST 160
      MUS=MRELVV(J)                                                     RDST 161
      NDN=LINFO(5,LT)                                                   RDST 162
      NGP=LINFO(11,LT)                                                  RDST 163
      INDX=LINFO(12,LT)                                                 RDST 164
      NAC=LINFO(15,LT)                                                  RDST 165
C                                                                       RDST 166
      CALL EQLIB(J,MUS,LT,NGP,NIP,INDX,NTPE,NEL,NDIM,NN,NDMX,NDN,       RDST 167
     1   NS,NB,NAC,NVRS,XYZ,VARINT,ELCOD,DS,SHFN,CARTD,B,FI,            RDST 168
     2   NCONN,LL,ISTGE)                                                RDST 169
```

```
C                                                                       RDST 170
CC    WRITE(IW6,805)MUS,FI                                              RDST 171
CC805 FORMAT(/1X,2HFI,2X,7HELEMENT = ,I5/(1X,6E14.4))                   RDST 172
C-----------------------------------------------------------------------RDST 173
C     SLOT EQUILIBRIUM LOADS INTO PEQT                                  RDST 174
C-----------------------------------------------------------------------RDST 175
      DO 95 IK=1,NDN                                                    RDST 176
      NCOR=NCONN(IK,J)                                                  RDST 177
      N1=NW(NCOR)-1                                                     RDST 178
C                                                                       RDST 179
      DO 95 ID=1,NDIM                                                   RDST 180
   95 PEQT(N1+ID)=PEQT(N1+ID)+FI(ID,IK)                                 RDST 181
  100 CONTINUE                                                          RDST 182
C-----------------------------------------------------------------------RDST 183
C     OUTPUT EQUILIBRIUM LOADS                                          RDST 184
C-----------------------------------------------------------------------RDST 185
CC    WRITE(IW6,985)(PEQT(J2),J2=1,NDF)                                 RDST 186
C                                                                       RDST 187
      CALL INSTRS(IW6,NN,NEL,NTPE,NIP,NVRS,NDIM,NDMX,NMT,MUMAX,NS,NL,   RDST 188
     1   XYZ,VARINT,NCONN,MAT,LTYP,MREL,ELCOD,DS,SHFN,CIP,LL,NTY)       RDST 189
  200 CONTINUE                                                          RDST 190
      RETURN                                                            RDST 191
  904 FORMAT(//1X,36HIN-SITU STRESSES ALL SET TO ZERO/1X,36(1H-))       RDST 192
  906 FORMAT(//1X,19HIN-SITU MESH DATA/1X,19(1H-)/                      RDST 193
     1 /3X,4HNODE,8X,1HY,10X,2HSX,10X,2HSY,10X,2HSZ,                    RDST 194
     2 9X,3HTXY,10X,1HU,22X,2HPC/)                                      RDST 195
  910 FORMAT(1X,I5,10F12.3)                                             RDST 196
  915 FORMAT(1X,7HELEMENT,I5,2X,18HIS OF UNKNOWN TYPE,I5)               RDST 197
  950 FORMAT(1X,46HWARNING --- POINT OUTSIDE IN-SITU STRESS SPACE,      RDST 198
     1 2X,9HELEMENT =,I5,2X,4HIP =,I5,2X,16H(ROUTINE RDSTRS))           RDST 199
CC951 FORMAT(2I4,7E14.4)                                                RDST 200
  955 FORMAT(//1X,40HDIRECT SPECIFICATION OF IN-SITU STRESSES           RDST 201
     1 /1X,39(1H-))                                                     RDST 202
  960 FORMAT(1X,10E12.5)                                                RDST 203
CC985 FORMAT(/1X,37HEQUILIBRIUM LOADS FOR INSITU STRESSES/              RDST 204
CC    1 1X,37(1H-)//(10E12.4))                                          RDST 205
      END                                                               RDST 206
```

RDST 28 : zero array of stresses.
RDST 33 : zero array PEQT; loads equivalent to *in situ* stresses.
RDST 34 : if *in situ* stresses are zero.
RDST 35 : branch off, depending on *in situ* stress option.
RDST 40 : no. of *in situ* nodes.
RDST 44–45 : read and write stresses specified at *in situ* nodes.
RDST 47 : no. of *in situ* layers.
RDST 49–52 : nodes marking each layer.
RDST 54 : enter nodes in the order of increasing depth.
RDST 55–56 : sort nodes into top-down sequence.
RDST 61 : loop on all elements.
RDST 63 : skip if element is not present in **primary** mesh.
RDST 66 : skip if element **type** is not present.
RDST 69–73 : element **type** dependent parameters.
NGP – no. of integration points.
NDN – no. of displacement nodes in element.
INDX – index to arrays W and L for different element types.
RDST 75–78 : copy nodal co-ordinates into local array ELCOD.

RDST 82 : loop on all integration points.
RDST 87–88 : local/area co-ordinates of integration point.
RDST 89 : calculate shape functions SHFN.
RDST 91–97 : co-ordinates of integration point.
RDST 101–105 : co-ordinates of nodes at top and bottom of layer.
RDST 107 : search for integration point in each *in situ* layer.
RDST 108 : layer in which integration point lies is found.
RDST 111 : integration point co-ordinate is outside *in situ* space.
RDST 116 : calculate interpolation factor.
RDST 119–120 : interpolate stresses at integration point.
RDST 122 : material **type** number.
RDST 125–129 : calculate p'_c (PC) and critical state value of p' as PU for Cam-clay models.
RDST 130–131 : calculate voids ratio.
RDST 132 : end of loop on integration points.
RDST 133 : end of element loop.
RDST 138–139 : direct specification of stresses at integration point.
RDST 141 : loop on all elements.
RDST 147–149 : read and write stresses at each integration point.
RDST 150 : end of element loop.
RDST 158 : calculate loads equivalent to *in situ* stresses; loop on all elements.
RDST 160 : skip if element is not present in primary mesh.
RDST 162–165 : element **type** dependent parameters.
RDST 167–169 : calculate loads in equilibrium with stresses in element (into FI).
RDST 176–181 : slot FI into PEQT.
RDST 182 : end of element loop.
RDST 188–189 : print out *in situ* stresses at integration points.

Routine SORTN2

```
      SUBROUTINE SORTN2(Y1,Y2,N1,N2,NMIN,NMAX)                          SRTN   1
C*****************************************************************    SRTN   2
C     ROUTINE TO SORT TWO INTEGERS                                      SRTN   3
C*****************************************************************    SRTN   4
      NMIN=N1                                                           SRTN   5
      NMAX=N2                                                           SRTN   6
      IF(Y1.LT.Y2)RETURN                                                SRTN   7
      NMAX=N1                                                           SRTN   8
      NMIN=N2                                                           SRTN   9
      RETURN                                                            SRTN  10
      END                                                               SRTN  11
```

SRTN 5–9 : sort two nodes; assign NMAX to the node with larger y value.

7.7.3 Integration point co-ordinates

The shape functions are used to calculate the co-ordinates of the integration points from the nodal co-ordinates.

$$x(\xi,\eta) = \sum_{i=1}^{n} N_i(\xi,\eta)\, x_i,$$

$$y(\xi,\eta) = \sum_{i=1}^{n} N_i(\xi,\eta)\, y_i. \qquad (7.1)$$

Routine **SHAPE** calculates the values of the shape functions N_i–SHFN(NDN). It also calculates the derivatives of the shape functions w.r.t. the local co-ordinates: $\partial N_i/\partial\xi$, $\partial N_i/\partial\eta$, which are placed in array DS(NDIM,NDN). These quantities are required in the calculation of the **B** matrix (see routine FORMB2). During the course of the analysis, there are many occasions when only the shape functions are required and not their derivatives. This choice is made by assigning 1 to the parameter ICODE. If set to 2, derivatives are also calculated.

Routine SHAPE

```
      SUBROUTINE SHAPE(IW6,LL,NAC,DS,SHFN,NDIM,NDN,LT,ICODE,MUS)        SHPE   1
C*************************************************************************SHPE   2
C     SHAPE FUNCTIONS AND DERIVATIVES FOR DIFFERENT ELEMENT TYPES        SHPE   3
C*************************************************************************SHPE   4
      REAL LL                                                           SHPE   5
      DIMENSION LL(NAC),SHFN(NDN),DS(NDIM,NDN)                          SHPE   6
C                                                                       SHPE   7
      AC1=LL(1)                                                         SHPE   8
      AC2=LL(2)                                                         SHPE   9
      IF(NAC.LT.3)GOTO 10                                               SHPE  10
      AC3=LL(3)                                                         SHPE  11
      IF(NAC.LT.4)GOTO 10                                               SHPE  12
      AC4=LL(4)                                                         SHPE  13
C                                                                       SHPE  14
   10 GOTO(11,13,13,14,14,15,15,17,17,18,18),LT                         SHPE  15
      WRITE(IW6,900)MUS,LT                                              SHPE  16
  900 FORMAT(/1X,7HELEMENT,I5,2X,22HIS OF UNKNOWN TYPE ***,I5,2X,       SHPE  17
     1 15H(ROUTINE SHAPE))                                              SHPE  18
      STOP                                                              SHPE  19
C-----------------------------------------------------------------------SHPE  20
C     SHAPE FUNCTIONS AND DERIVATIVES FOR BAR ELEMENT                   SHPE  21
C-----------------------------------------------------------------------SHPE  22
   11 CONTINUE                                                          SHPE  23
      WRITE(IW6,910)MUS,LT                                              SHPE  24
  910 FORMAT(/1X,7HELEMENT,I5,2X,14HIS OF TYPE ***,I5,2X,               SHPE  25
     1 31HNOT IMPLEMENTED (ROUTINE SHAPE))                              SHPE  26
      GOTO 80                                                           SHPE  27
C-----------------------------------------------------------------------SHPE  28
C     SHAPE FUNCTIONS AND DERIVATIVES FOR LST                           SHPE  29
C-----------------------------------------------------------------------SHPE  30
   13 SHFN(1)=AC1*(2.*AC1-1.)                                           SHPE  31
      SHFN(2)=AC2*(2.*AC2-1.)                                           SHPE  32
      SHFN(3)=AC3*(2.*AC3-1.)                                           SHPE  33
      SHFN(4)=4.*AC1*AC2                                                SHPE  34
      SHFN(5)=4.*AC2*AC3                                                SHPE  35
      SHFN(6)=4.*AC1*AC3                                                SHPE  36
      IF(ICODE.EQ.1)GOTO 80                                             SHPE  37
C                                                                       SHPE  38
      DS(1,1)=4.*AC1-1.                                                 SHPE  39
      DS(1,2)=0.                                                        SHPE  40
      DS(1,3)=-(4.*AC3-1.)                                              SHPE  41
      DS(1,4)=4.*AC2                                                    SHPE  42
```



```
      DS(1,5)=-4.*AC2                                                   SHPE  43
      DS(1,6)=4.*(AC3-AC1)                                              SHPE  44
C                                                                       SHPE  45
      DS(2,1)=0.                                                        SHPE  46
      DS(2,2)=4.*AC2-1.                                                 SHPE  47
      DS(2,3)=-(4.*AC3-1.)                                              SHPE  48
      DS(2,4)=4.*AC1                                                    SHPE  49
      DS(2,5)=4.*(AC3-AC2)                                              SHPE  50
      DS(2,6)=-4.*AC1                                                   SHPE  51
      GO TO 80                                                          SHPE  52
C-----------------------------------------------------------------------SHPE  53
C     SHAPE FUNCTIONS AND DERIVATIVES FOR QUADRILATERALS                SHPE  54
C-----------------------------------------------------------------------SHPE  55
   14 CONTINUE                                                          SHPE  56
      WRITE(IW6,910)MUS,LT                                              SHPE  57
      GOTO 80                                                           SHPE  58
C-----------------------------------------------------------------------SHPE  59
C     SHAPE FUNCTIONS AND DERIVATIVES FOR CUBIC STRAIN TRIANGLE         SHPE  60
C-----------------------------------------------------------------------SHPE  61
   15 CONTINUE                                                          SHPE  62
      C1=32./3.                                                         SHPE  63
      C2=64.                                                            SHPE  64
      C3=128./3.                                                        SHPE  65
      C4=128.                                                           SHPE  66
      T11=AC1-0.25                                                      SHPE  67
      T12=AC1-0.50                                                      SHPE  68
      T13=AC1-0.75                                                      SHPE  69
      T21=AC2-0.25                                                      SHPE  70
      T22=AC2-0.50                                                      SHPE  71
      T23=AC2-0.75                                                      SHPE  72
      T31=AC3-0.25                                                      SHPE  73
      T32=AC3-0.50                                                      SHPE  74
      T33=AC3-0.75                                                      SHPE  75
C-----------------------------------------------------------------------SHPE  76
C     SHAPE FUNCTIONS                                                   SHPE  77
C-----------------------------------------------------------------------SHPE  78
      SHFN(1) =C1*AC1*T11*T12*T13                                       SHPE  79
      SHFN(2) =C1*AC2*T21*T22*T23                                       SHPE  80
      SHFN(3) =C1*AC3*T31*T32*T33                                       SHPE  81
      SHFN(4) =C3*AC1*AC2*T11*T12                                       SHPE  82
      SHFN(5) =C2*AC1*AC2*T11*T21                                       SHPE  83
      SHFN(6) =C3*AC1*AC2*T21*T22                                       SHPE  84
      SHFN(7) =C3*AC2*AC3*T21*T22                                       SHPE  85
      SHFN(8) =C2*AC2*AC3*T21*T31                                       SHPE  86
      SHFN(9) =C3*AC2*AC3*T31*T32                                       SHPE  87
      SHFN(10)=C3*AC1*AC3*T31*T32                                       SHPE  8
      SHFN(11)=C2*AC1*AC3*T11*T31                                       SHPE  89
      SHFN(12)=C3*AC1*AC3*T11*T12                                       SHPE  90
      SHFN(13)=C4*AC1*AC2*AC3*T11                                       SHPE  91
      SHFN(14)=C4*AC1*AC2*AC3*T21                                       SHPE  92
      SHFN(15)=C4*AC1*AC2*AC3*T31                                       SHPE  93
      IF(ICODE.EQ.1)GOTO 80                                             SHPE  94
C                                                                       SHPE  95
      DS(1,1)=C1*(T12*T13*(T11+AC1)+AC1*T11*(T13+T12))                  SHPE  96
      DS(1,2)= 0.                                                       SHPE  97
      DS(1,3)=-C1*(T32*T33*(AC3+T31)+AC3*T31*(T32+T33))                 SHPE  98
      DS(1,4)= C3*AC2*(T11*T12+AC1*(T11+T12))                           SHPE  99
      DS(1,5)= C2*AC2*T21*(AC1+T11)                                     SHPE 100
      DS(1,6)= C3*AC2*T21*T22                                           SHPE 101
      DS(1,7)=-C3*AC2*T21*T22                                           SHPE 102
      DS(1,8)=-C2*AC2*T21*(AC3+T31)                                     SHPE 103
      DS(1,9)=-C3*AC2*(T31*T32+AC3*(T31+T32))                           SHPE 104
      DS(1,10)=-C3*(AC1*AC3*(T31+T32)-T31*T32*(AC3-AC1))                SHPE 105
      DS(1,11)= C2*(AC1*AC3*(T31-T11)+T31*T11*(AC3-AC1))                SHPE 106
      DS(1,12)= C3*(AC1*AC3*(T11+T12)+T11*T12*(AC3-AC1))                SHPE 107
      DS(1,13)= C4*AC2*(AC1*AC3+T11*(AC3-AC1))                          SHPE 108
```

```
      DS(1,14)= C4*AC2*T21*(AC3-AC1)                                    SHPE 109
      DS(1,15)=-C4*AC2*(AC1*AC3+T31*(AC1-AC3))                          SHPE 110
C                                                                       SHPE 111
      DS(2,1) = 0.                                                      SHPE 112
      DS(2,2) = C1*(T22*T23*(AC2+T21)+AC2*T21*(T22+T23))                SHPE 113
      DS(2,3) =-C1*(T32*T33*(AC3+T31)+AC3*T31*(T32+T33))                SHPE 114
      DS(2,4) = C3*AC1*T11*T12                                          SHPE 115
      DS(2,5) = C2*AC1*T11*(AC2+T21)                                    SHPE 116
      DS(2,6) = C3*AC1*(T21*T22+AC2*(T21+T22))                          SHPE 117
      DS(2,7) = C3*(AC2*AC3*(T21+T22)+T21*T22*(AC3-AC2))                SHPE 118
      DS(2,8) = C2*(AC2*AC3*(T31-T21)+T21*T31*(AC3-AC2))                SHPE 119
      DS(2,9) =-C3*(AC2*AC3*(T31+T32)+T31*T32*(AC2-AC3))                SHPE 120
      DS(2,10)=-C3*AC1*(T31*T32+AC3*(T31+T32))                          SHPE 121
      DS(2,11)=-C2*AC1*T11*(AC3+T31)                                    SHPE 122
      DS(2,12)=-C3*AC1*T11*T12                                          SHPE 123
      DS(2,13)= C4*AC1*T11*(AC3-AC2)                                    SHPE 124
      DS(2,14)= C4*AC1*(AC2*AC3+T21*(AC3-AC2))                          SHPE 125
      DS(2,15)=-C4*AC1*(AC2*AC3+T31*(AC2-AC3))                          SHPE 126
      GO TO 80                                                          SHPE 127
C-----------------------------------------------------------------------SHPE 128
C     SHAPE FUNCTIONS AND DERIVATIVES FOR BRICK ELEMENT                 SHPE 129
C-----------------------------------------------------------------------SHPE 130
   17 CONTINUE                                                          SHPE 131
      WRITE(IW6,910)MUS,LT                                              SHPE 132
      GOTO 80                                                           SHPE 133
C-----------------------------------------------------------------------SHPE 134
C     SHAPE FUNCTIONS AND DERIVATIVES FOR TETRA-HEDRA                   SHPE 135
C-----------------------------------------------------------------------SHPE 136
   18 CONTINUE                                                          SHPE 137
      WRITE(IW6,910)MUS,LT                                              SHPE 138
      GOTO 80                                                           SHPE 139
   80 CONTINUE                                                          SHPE 140
      RETURN                                                            SHPE 141
      END                                                               SHPE 142
```

SHPE 8–13 : set up AC1, AC2, etc. equal to the integration point co-ordinates.
NL = 3 for two-dimensional triangular elements.
NL = 2 for two-dimensional quadrilateral elements.
NL = 3 for three-dimensional elements.

SHPE 15 : branch off for different element types.

SHPE 23 : shape functions and derivatives for bar element (LT = 1; not yet implemented).

SHPE 31–36 : shape functions for six-noded triangular element (LT = 2, 3).

SHPE 39–44 : calculate derivatives w.r.t. local co-ordinates – $\partial N_i/\partial\xi$, $\partial N_i/\partial\eta$ (LT = 2, 3).

SHPE 56 : shape functions and derivatives for quadrilateral element (LT = 4, 5) – not included in this version.

SHPE 63–75 : set up constants for LT = 6, 7.

SHPE 79–93 : shape functions for cubic strain triangle (LT = 6, 7).

SHPE 96–126 : calculate derivatives w.r.t. local co-ordinates – $\partial N_i/\partial\xi$, $\partial N_i/\partial\eta$ (LT = 6, 7).

SHPE 131 : calculate shape functions and derivatives for brick element (LT = 8, 9; not yet implemented).

SHPE 137 : calculate shape functions and derivatives for tetrahedra element (LT = 10, 11; not yet implemented).

7.7.4 Loads equivalent to *in situ* stresses

The nodal loads equivalent to the *in situ* stresses are calculated in routine **EQLIB** and placed in array F(NDF) for each element, and these are later used in the equilibrium calculations.

$$\mathrm{F(NDF)} = \begin{bmatrix} F_{xi} \\ F_{yi} \end{bmatrix}.$$

$$\begin{bmatrix} F_{xi} \\ F_{yi} \end{bmatrix} = \int \mathbf{B}_i^{\mathrm{T}} \cdot \sigma_0 \, \mathrm{d\,(vol)} = \int_{V_e} \begin{bmatrix} \dfrac{\partial N_i}{\partial x} \cdot \sigma_{x0} + \dfrac{\partial N_i}{\partial y} \cdot \tau_{xy0} \\ \dfrac{\partial N_i}{\partial y} \cdot \sigma_{y0} + \dfrac{\partial N_i}{\partial x} \cdot \tau_{xy0} \end{bmatrix} \mathrm{d\,(vol)}. \tag{7.2}$$

The $\mathbf{B}_i$ matrix is given by

$$\begin{bmatrix} \dfrac{\partial N_i}{\partial x} & 0 \\ 0 & \dfrac{\partial N_i}{\partial y} \\ \dfrac{N_i}{x}\ ^{\dagger} & 0 \\ \dfrac{\partial N_i}{\partial y} & \dfrac{\partial N_i}{\partial x} \end{bmatrix}.$$

The calculation

$$\mathbf{F}_i = \int \mathbf{B}^{\mathrm{T}} \cdot \sigma_0 \, \mathrm{d\,(vol)} \tag{7.3}$$

is expanded and written in long hand, leaving out all zero multiplications using

$$\begin{aligned} \mathrm{CARTD(1,I)} &= \frac{\partial N_i}{\partial x}, \\ \mathrm{CARTD(2,I)} &= \frac{\partial N_i}{\partial y}, \\ \mathrm{B(3,I)} &= \frac{N_i}{x}. \end{aligned} \tag{7.4}$$

I, *i* denote the *i*th node.

† This term is only present for axisymmetric problems, x being the radial distance of the integration point.

Routine EQLIB

```
      SUBROUTINE EQLIB(JJ,MUS,LT,NGP,NIP,INDX,NTPE,NEL,NDIM,NN,NDMX,NDN,EQLB   1
     1 NS,NB,NAC,NVRS,XYZ,VARINT,ELCOD,DS,SHFN,CARTD,B,F,NCONN,LL,ISTGE)EQLB   2
C*************************************************************************EQLB   3
C     ROUTINE TO CALCULATE FORCES EQUILIBRATING                           EQLB   4
C     ELEMENTAL STRESSES                                                  EQLB   5
C*************************************************************************EQLB   6
      REAL L,LL                                                           EQLB   7
      DIMENSION XYZ(NDIM,NN),VARINT(NVRS,NIP,NEL),ELCOD(NDIM,NDMX),       EQLB   8
     1 DS(NDIM,NDMX),SHFN(NDMX),CARTD(NDIM,NDMX),B(NS,NB),                EQLB   9
     2 F(NDIM,NDMX),LL(NAC),NCONN(NTPE,NEL)                               EQLB  10
      COMMON /PARS  / PYI,ALAR,ASMVL,ZERO                                 EQLB  11
      COMMON /DATW  / W(100)                                              EQLB  12
      COMMON /DATL  / L(4,100)                                            EQLB  13
      COMMON /FLOW  / NPLAX                                               EQLB  14
      COMMON /JACB  / XJACI(3,3),DJACB                                    EQLB  15
C                                                                         EQLB  16
      CR=1.                                                               EQLB  17
      IF(NPLAX.EQ.1)CR=2.*PYI                                             EQLB  18
C                                                                         EQLB  19
      CALL ZEROR2(F,NDIM,NDMX)                                            EQLB  20
C                                                                         EQLB  21
      DO 20 KN=1,NDN                                                      EQLB  22
      NDE=NCONN(KN,JJ)                                                    EQLB  23
      DO 20 ID=1,NDIM                                                     EQLB  24
   20 ELCOD(ID,KN)=XYZ(ID,NDE)                                            EQLB  25
C                                                                         EQLB  26
      DO 60 IP=1,NGP                                                      EQLB  27
      IPA=IP+INDX                                                         EQLB  28
C                                                                         EQLB  29
      DO 30 IL=1,NAC                                                      EQLB  30
   30 LL(IL)=L(IL,IPA)                                                    EQLB  31
      CALL FORMB2(JJ,MUS,R,RI,NDIM,NDMX,NDN,NS,                           EQLB  32
     1   NB,NAC,ELCOD,DS,SHFN,CARTD,B,LL,LT,IP,ISTGE)                     EQLB  33
      F9=CR*DJACB*W(IPA)                                                  EQLB  34
      IF(NPLAX.EQ.1)F9=F9*R                                               EQLB  35
C                                                                         EQLB  36
      U=VARINT(NS+1,IP,JJ)                                                EQLB  37
      SIGXT=VARINT(1,IP,JJ)+U                                             EQLB  38
      SIGYT=VARINT(2,IP,JJ)+U                                             EQLB  39
      SIGZT=VARINT(3,IP,JJ)+U                                             EQLB  40
      TXY=VARINT(4,IP,JJ)                                                 EQLB  41
      IF(NDIM.EQ.2)GOTO 35                                                EQLB  42
C                                                                         EQLB  43
      TYZ=VARINT(5,IP,JJ)                                                 EQLB  44
      TZX=VARINT(6,IP,JJ)                                                 EQLB  45
C                                                                         EQLB  46
      DO 50 IN=1,NDN                                                      EQLB  47
      F(1,IN)=F(1,IN)+(CARTD(1,IN)*SIGXT+CARTD(2,IN)*TXY                  EQLB  48
     1               +CARTD(3,IN)*TZX)*F9                                 EQLB  49
      F(2,IN)=F(2,IN)+(CARTD(2,IN)*SIGYT+CARTD(1,IN)*TXY                  EQLB  50
     1               +CARTD(3,IN)*TYZ)*F9                                 EQLB  51
      F(3,IN)=F(3,IN)+(CARTD(3,IN)*SIGZT+CARTD(2,IN)*TYZ                  EQLB  52
     1               +CARTD(1,IN)*TZX)*F9                                 EQLB  53
   50 CONTINUE                                                            EQLB  54
      GOTO 60                                                             EQLB  55
C                                                                         EQLB  56
   35 DO 40 IN=1,NDN                                                      EQLB  57
      F(1,IN)=F(1,IN)+(CARTD(1,IN)*SIGXT+SHFN(IN)*SIGZT*RI                EQLB  58
     1               +CARTD(2,IN)*TXY)*F9                                 EQLB  59
   40 F(2,IN)=F(2,IN)+(CARTD(2,IN)*SIGYT+CARTD(1,IN)*TXY)*F9              EQLB  60
   60 CONTINUE                                                            EQLB  61
      RETURN                                                              EQLB  62
      END                                                                 EQLB  63
```


EQLB 17–18 : multiplication factor for numerical integration.
EQLB 20 : zero array F.
EQLB 22–25 : copy nodal co-ordinates into local array ELCOD.
EQLB 27 : loop on all integration points.
EQLB 30–31 : integration point co-ordinates.
EQLB 32–33 : calculate components of **B** matrix.
EQLB 34–35 : multiplication factor for numerical integration.
EQLB 37–41 : total stresses σ for 2-D.
EQLB 44–45 : additional stress components for 3-D.
EQLB 47–53 : calculate $\int \mathbf{B}^{\mathrm{T}}\sigma \,.\, \mathrm{d}\,(\mathrm{vol})$ for 3-D, contribution from integration point.
EQLB 57–60 : calculate $\int \mathbf{B}^{\mathrm{T}}\sigma \,.\, \mathrm{d}\,(\mathrm{vol})$ for 2-D, contribution from integration point.
EQLB 61 : end of integration point loop.

7.7.5 *B* matrix

Routine **FORMB2** calculates the **B** matrix, which is made up of terms $\partial N_i/\partial x$, $\partial N_i/\partial y$. These Cartesian derivatives of shape functions are calculated using the chain differentiation rule:

$$\begin{aligned}\frac{\partial N_i}{\partial x} &= \frac{\partial N_i}{\partial \xi}\frac{\partial \xi}{\partial x} + \frac{\partial N_i}{\partial \eta}\frac{\partial \eta}{\partial x},\\ \frac{\partial N_i}{\partial y} &= \frac{\partial N_i}{\partial \xi}\frac{\partial \xi}{\partial y} + \frac{\partial N_i}{\partial \eta}\frac{\partial \eta}{\partial y}.\end{aligned} \tag{7.5}$$

$$\begin{bmatrix}\dfrac{\partial N_i}{\partial x}\\[2ex] \dfrac{\partial N_i}{\partial y}\end{bmatrix} = \begin{bmatrix}\dfrac{\partial \xi}{\partial x} & \dfrac{\partial \eta}{\partial x}\\[2ex] \dfrac{\partial \xi}{\partial y} & \dfrac{\partial \eta}{\partial y}\end{bmatrix}\begin{bmatrix}\dfrac{\partial N_i}{\partial \xi}\\[2ex] \dfrac{\partial N_i}{\partial \eta}\end{bmatrix}. \tag{7.6}$$

The $\partial N_i/\partial\xi$, $\partial N_i/\partial\eta$ terms are calculated in routine SHAPE. The Jacobian matrix $\mathbf{J}(\xi, \eta)$ is given by

$$J = \begin{bmatrix}\dfrac{\partial x}{\partial \xi} & \dfrac{\partial y}{\partial \xi}\\[2ex] \dfrac{\partial x}{\partial \eta} & \dfrac{\partial y}{\partial \eta}\end{bmatrix} = \sum_{i=1}^{n}\begin{bmatrix}\dfrac{\partial N_i}{\partial \xi}x_i & \dfrac{\partial N_i}{\partial \xi}y_i\\[2ex] \dfrac{\partial N_i}{\partial \eta}x_i & \dfrac{\partial N_i}{\partial \eta}y_i\end{bmatrix}. \tag{7.7}$$

The inverse of the Jacobian matrix is then given by

$$\mathbf{J}^{-1} = \begin{bmatrix} \dfrac{\partial \xi}{\partial x} & \dfrac{\partial \eta}{\partial x} \\ \dfrac{\partial \xi}{\partial y} & \dfrac{\partial \eta}{\partial y} \end{bmatrix} = \frac{1}{\det \mathbf{J}} \begin{bmatrix} \dfrac{\partial y}{\partial \eta} & \dfrac{-\partial y}{\partial \xi} \\ \dfrac{-\partial x}{\partial \eta} & \dfrac{\partial x}{\partial \xi} \end{bmatrix}. \tag{7.8}$$

Knowing this, $\partial N_i/\partial x$, $\partial N_i/\partial y$ can be calculated from the above equation. The determinant of **J** is calculated in routine **DETMIN**.

Routine FORMB2

```
      SUBROUTINE FORMB2(J,MUS,R,RI,NDIM,NDMX,NDN,NS,NB,NAC,          FRMB   1
     1 ELCOD,DS,SHFN,CARTD,B,LL,LT,IP,ISTGE)                         FRMB   2
C*****************************************************************FRMB   3
C     FORMS  B  MATRIX  FROM  AREA/LOCAL  COORDS  LL(NAC)           FRMB   4
C     IN  ELEMENT  J FOR INTEGRATION POINT IP                       FRMB   5
C*****************************************************************FRMB   6
                                                                    FRMB   7
      REAL LL                                                       FRMB   8
      DIMENSION ELCOD(NDIM,NDMX),DS(NDIM,NDMX),SHFN(NDMX),          FRMB   9
     1 CARTD(NDIM,NDMX),B(NS,NB),LL(NAC),XJACM(3,3)                 FRMB  10
      COMMON /FLOW  / NPLAX                                         FRMB  11
      COMMON /PARS  / PYI,ALAR,ASMVL,ZERO                           FRMB  12
      COMMON /DEVICE/ IR1,IR4,IR5,IW2,IW4,IW6,IW7,IW8,IW9           FRMB  13
      COMMON /JACB  / XJACI(3,3),DJACB                              FRMB  14
C-----------------------------------------------------------------FRMB  15
C     INITIALISE SHAPE FUNCTION AND DERIVATIVES (LOCAL COORDS)      FRMB  16
C-----------------------------------------------------------------FRMB  17
      CALL ZEROR2(DS,NDIM,NDMX)                                     FRMB  18
      CALL ZEROR1(SHFN,NDMX)                                        FRMB  19
      CALL ZEROR2(B,NS,NB)                                          FRMB  20
C                                                                   FRMB  21
      CALL SHAPE(IW6,LL,NAC,DS,SHFN,NDIM,NDN,LT,2,MUS)              FRMB  22
      CALL ZEROR2(XJACM,NDIM,NDIM)                                  FRMB  23
C                                                                   FRMB  24
      NDN2=2*NDN                                                    FRMB  25
C                                                                   FRMB  26
      DO 15 IDIM=1,NDIM                                             FRMB  27
      DO 15 JDIM=1,NDIM                                             FRMB  28
      SUM=ZERO                                                      FRMB  29
C                                                                   FRMB  30
      DO 12 IN=1,NDN                                                FRMB  31
   12 SUM=SUM+DS(IDIM,IN)*ELCOD(JDIM,IN)                            FRMB  32
   15 XJACM(IDIM,JDIM)=SUM                                          FRMB  33
C                                                                   FRMB  34
      CALL DETMIN(IW6,XJACM,XJACI,NDIM,DJACB,MUS,IP,ISTGE)          FRMB  35
CC    WRITE(IW6,902)DJACB                                           FRMB  36
CC902 FORMAT(9H JACOBIAN,2X,E16.5)                                  FRMB  37
C-----------------------------------------------------------------FRMB  38
C     CALCULATE  RADIUS  FOR  AXI-SYM  B  MATRIX                    FRMB  39
C-----------------------------------------------------------------FRMB  40
      R=ZERO                                                        FRMB  41
      RI=ZERO                                                       FRMB  42
      IF(NPLAX.EQ.0)GOTO 28                                         FRMB  43
C                                                                   FRMB  44
      DO 25 IN=1,NDN                                                FRMB  45
   25 R=R+ELCOD(1,IN)*SHFN(IN)                                      FRMB  46
      RI=-1.0/R                                                     FRMB  47
C                                                                   FRMB  48
   28 DO 35 IN=1,NDN                                                FRMB  49
      DO 35 ID=1,NDIM                                               FRMB  50
      SUM=ZERO                                                      FRMB  51
      DO 30 JD=1,NDIM                                               FRMB  52
   30 SUM=SUM-DS(JD,IN)*XJACI(ID,JD)                                FRMB  53
   35 CARTD(ID,IN)=SUM                                              FRMB  54
C                                                                   FRMB  55
      IF(NDIM.NE.2)GOTO 52                                          FRMB  56
C-----------------------------------------------------------------FRMB  57
C     2 - D ELEMENT                                                 FRMB  58
C-----------------------------------------------------------------FRMB  59
      DO 50 IN=1,NDN                                                FRMB  60
      B(1,IN)=CARTD(1,IN)                                           FRMB  61
      B(2,NDN+IN)=CARTD(2,IN)                                       FRMB  62
      IF(NPLAX.EQ.0)GOTO 45                                         FRMB  63
      B(3,IN)=SHFN(IN)*RI                                           FRMB  64
   45 B(4,NDN+IN)=B(1,IN)                                           FRMB  65
   50 B(4,IN)=B(2,NDN+IN)                                           FRMB  66
C                                                                   FRMB  67
   52 IF(NDIM.NE.3)GOTO 62                                          FRMB  68
C-----------------------------------------------------------------FRMB  69
C     3 - D ELEMENT                                                 FRMB  70
C-----------------------------------------------------------------FRMB  71
      DO 60 IN=1,NDN                                                FRMB  72
      B(1,IN)=CARTD(1,IN)                                           FRMB  73
      B(2,NDN+IN)=CARTD(2,IN)                                       FRMB  74
      B(3,NDN2+IN)=CARTD(3,IN)                                      FRMB  75
      B(4,IN)=CARTD(2,IN)                                           FRMB  76
      B(4,NDN+IN)=CARTD(1,IN)                                       FRMB  77
      B(5,NDN+IN)=CARTD(3,IN)                                       FRMB  78
      B(5,NDN2+IN)=CARTD(2,IN)                                      FRMB  79
      B(6,IN)=CARTD(3,IN)                                           FRMB  80
      B(6,NDN2+IN)=CARTD(1,IN)                                      FRMB  81
   60 CONTINUE                                                      FRMB  82
C                                                                   FRMB  83
   62 CONTINUE                                                      FRMB  84
      RETURN                                                        FRMB  85
      END                                                           FRMB  86
```


FRMB 17–19 : zero arrays for shape functions (SHFN), derivatives (DS) and strain matrix (B).

FRMB 21 : calculate shape functions (N_i) and derivatives w.r.t. ξ and η ($\partial N_i/\partial \xi$, $\partial N_i/\partial \eta$).

FRMB 22 : zero Jacobian matrix.

FRMB 26–32 : calculate components of Jacobian matrix **J**.

FRMB 34 : calculate determinant of **J** and inverse $\mathbf{J}^{-1}$.

FRMB 40 : zero radius R for axisymmetric analysis.

FRMB 44–45 : calculate radius, R, of integration point.

FRMB 46 : RI is the inverse of R.

FRMB 48–54 : calculate Cartesian derivatives of shape functions $\partial N_i/\partial x$, $\partial N_i/\partial y$. The negative sign is to allow for the sign convention that *compressive* strains are positive.

FRMB 60–66 : calculate **B** matrix for two-dimensional elements.

FRMB 64 : calculate row 3 of **B** matrix for axisymmetric elements only.

FRMB 72–82 : calculate **B** matrix for three-dimensional elements.

Routine DETMIN

```
      SUBROUTINE DETMIN(IW6,XJACM,XJACI,NDIM,DJACB,JL,IP,ISTGE)          DETM   1
C*****************************************************************     DETM   2
C     CALCULATES DETERMINANT AND INVERSE OF A SQUARE 3X3 MATRIX         DETM   3
C*****************************************************************     DETM   4
      DIMENSION XJACM(3,3),XJACI(3,3)                                   DETM   5
      COMMON /PARS  / PYI,ALAR,ASMVL,ZERO                               DETM   6
C                                                                       DETM   7
      IF(NDIM.NE.2)GOTO 20                                              DETM   8
      DJACB=XJACM(1,1)*XJACM(2,2)-XJACM(1,2)*XJACM(2,1)                 DETM   9
      IF(DJACB.GT.ZERO)GOTO 15                                          DETM  10
      GOTO 60                                                           DETM  11
C                                                                       DETM  12
   15 XJACI(1,1)= XJACM(2,2)/DJACB                                      DETM  13
      XJACI(2,2)= XJACM(1,1)/DJACB                                      DETM  14
      XJACI(1,2)=-XJACM(1,2)/DJACB                                      DETM  15
      XJACI(2,1)=-XJACM(2,1)/DJACB                                      DETM  16
      RETURN                                                            DETM  17
C                                                                       DETM  18
   20 XJACI(1,1)= (XJACM(2,2)*XJACM(3,3)-XJACM(2,3)*XJACM(3,2))         DETM  19
      XJACI(1,2)=-(XJACM(1,2)*XJACM(3,3)-XJACM(1,3)*XJACM(3,2))         DETM  20
      XJACI(1,3)= (XJACM(1,2)*XJACM(2,3)-XJACM(1,3)*XJACM(2,2))         DETM  21
C                                                                       DETM  22
      XJACI(2,1)=-(XJACM(2,1)*XJACM(3,3)-XJACM(2,3)*XJACM(3,1))         DETM  23
      XJACI(2,2)= (XJACM(1,1)*XJACM(3,3)-XJACM(1,3)*XJACM(3,1))         DETM  24
      XJACI(2,3)=-(XJACM(1,1)*XJACM(2,3)-XJACM(1,3)*XJACM(2,1))         DETM  25
C                                                                       DETM  26
      XJACI(3,1)= (XJACM(2,1)*XJACM(3,2)-XJACM(2,2)*XJACM(3,1))         DETM  27
      XJACI(3,2)=-(XJACM(1,1)*XJACM(3,2)-XJACM(1,2)*XJACM(3,1))         DETM  28
      XJACI(3,3)= (XJACM(1,1)*XJACM(2,2)-XJACM(2,1)*XJACM(1,2))         DETM  29
C                                                                       DETM  30
      DJACB=XJACM(1,1)*XJACI(1,1)+XJACM(1,2)*XJACI(2,1)+                DETM  31
     1      XJACM(1,3)*XJACI(3,1)                                       DETM  32
      IF(DJACB.GT.ZERO)GOTO 32                                          DETM  33
      GOTO 60                                                           DETM  34
C                                                                       DETM  35
   32 DJACBI=1.0/DJACB                                                  DETM  36
C                                                                       DETM  37
      DO 35 ID=1,NDIM                                                   DETM  38
      DO 35 JD=1,NDIM                                                   DETM  39
   35 XJACI(ID,JD)=XJACI(ID,JD)*DJACBI                                  DETM  40
      RETURN                                                            DETM  41
   60 WRITE(IW6,900)DJACB,JL,IP                                         DETM  42
  900 FORMAT(/1X,9HJACOBIAN  ,E16.5,3X,10HOF ELEMENT,I6,3X,             DETM  43
     1 17HINTEGRATION POINT,I5,3X,29HIS NEGATIVE  (ROUTINE DETMIN))     DETM  44
      WRITE(IW6,910)ISTGE                                               DETM  45
  910 FORMAT(/1X,36HCODE TO INDICATE STAGE OF ANALYSIS =,I5//           DETM  46
     1 4X,4HCODE,20X,21HSTAGE OF THE ANALYSIS//                         DETM  47
     2 6X,49H1 - CALLED BY RDSTRS/EQLIB/FORMB2 LOAD EQUIVALENT,         DETM  48
     3 19H TO INSITU STRESSES/6X,33H2 - CALLED BY CHANGE/EQLIB/FORMB2,  DETM  49
     4 32H CALCULATION OF IMPLIED LOADINGS/6X,                          DETM  50
     5 34H3 - CALLED BY FRONTZ/LSTIFF/FORMB2,                           DETM  51
     6 32H CALCULATION OF STIFFNESS MATRIX/                             DETM  52
     7 6X,38H4 - CALLED BY UPOUT/FORMB2 CALCULATION,                    DETM  53
     8 1X,24HOF STRAINS. OUTPUT STAGE)                                  DETM  54
      STOP                                                              DETM  55
      END                                                               DETM  56
```

DETM 8 : branch off if not two-dimensional problem.

DETM 9 : calculate determinant of Jacobian, **J**, for two-dimensional problems.

DETM 10 : check if determinant of **J** is positive.

DETM 13–16 : calculate inverse $\mathbf{J}^{-1}$.

DETM 19–29 : calculate the cofactors of **J** for the three-dimensional case.

DETM 31–32 : calculate determinant of **J**.

DETM 38–40 : calculate inverse, $\mathbf{J}^{-1}$.

DETM 42–44 : print out warning message if det $|\mathbf{J}| <$ zero.

DETM 45–54 : print out codes to identify stage of analysis for debugging purposes.

7.7.6 Print out *in situ* stresses

The *in situ* stresses have been calculated at all integration points. Also calculated are the equivalent nodal loads for these stresses.

The *in situ* stresses calculated at all integration points are printed out in routine **INSTRS** along with Cam-clay parameters p', q, p'_c and e, the voids ratio.

Routine INSTRS

```
      SUBROUTINE INSTRS(IW6,NN,NEL,NTPE,NIP,NVRS,NDIM,                  INSR   1
     1 NDMX,NMT,MUMAX,NS,NL,XYZ,VARINT,NCONN,MAT,LTYP,                  INSR   2
     2 MREL,ELCOD,DS,SHFN,CIP,LL,NTY)                                   INSR   3
C*****************************************************************     INSR   4
C     ROUTINE TO PRINT OUT IN-SITU STRESSES                            *INSR   5
C     BEFORE THE FIRST INCREMENT                                       *INSR   6
C*****************************************************************     INSR   7
      REAL L,LL                                                         INSR   8
      DIMENSION XYZ(NDIM,NN),VARINT(NVRS,NIP,NEL)                       INSR   9
      DIMENSION NCONN(NTPE,NEL),MAT(NEL),LTYP(NEL),MREL(MUMAX)          INSR  10
      DIMENSION ELCOD(NDIM,NDMX),DS(NDIM,NDMX),SHFN(NDMX),              INSR  11
     1 CIP(NDIM),LL(NL),NTY(NMT)                                        INSR  12
      COMMON /ELINF / LINFO(50,15)                                      INSR  13
      COMMON /DATL  / L(4,100)                                          INSR  14
      COMMON /PARS  / PYI,ALAR,ASMVL,ZERO                               INSR  15
C                                                                       INSR  16
      NS1=NS+1                                                          INSR  17
      WRITE(IW6,900)                                                    INSR  18
  900 FORMAT(/1X,34HINTEGRATION POINT IN-SITU STRESSES/                 INSR  19
     1 1X,34(1H-)/)                                                     INSR  20
      WRITE(IW6,901)                                                    INSR  21
C                                                                       INSR  22
      DO 60 MR=1,MUMAX                                                  INSR  23
      IF(MREL(MR).EQ.0)GO TO 60                                         INSR  24
      J=MREL(MR)                                                        INSR  25
      LT=LTYP(J)                                                        INSR  26
      IF(LTYP(J).LT.0)GO TO 60                                          INSR  27
      NDN=LINFO(5,LT)                                                   INSR  28
      NGP=LINFO(11,LT)                                                  INSR  29
      INDX=LINFO(12,LT)                                                 INSR  30
      NAC=LINFO(15,LT)                                                  INSR  31
      KM=MAT(J)                                                         INSR  32
      KGO=NTY(KM)                                                       INSR  33
      GO TO(11,11,12,12,60,60),KGO                                      INSR  34
      WRITE(IW6,910)MR,KGO                                              INSR  35
      GOTO 60                                                           INSR  36
   11 ICAM=0                                                            INSR  37
      GO TO 14                                                          INSR  38
   12 ICAM=1                                                            INSR  39
   14 CONTINUE                                                          INSR  40
      WRITE(6,902)MR                                                    INSR  41
```

```
C                                                                       INSR  42
      DO 18 KN=1,NDN                                                    INSR  43
      NDE=NCONN(KN,J)                                                   INSR  44
      DO 18 ID=1,NDIM                                                   INSR  45
   18 ELCOD(ID,KN)=XYZ(ID,NDE)                                          INSR  46
C                                                                       INSR  47
      DO 40 IP=1,NGP                                                    INSR  48
      IPA=IP+INDX                                                       INSR  49
C                                                                       INSR  50
      DO 25 IL=1,NAC                                                    INSR  51
   25 LL(IL)=L(IL,IPA)                                                  INSR  52
      CALL SHAPE (IW6,LL,NAC,DS,SHFN,NDIM,NDN,LT,1,MR)                  INSR  53
C                                                                       INSR  54
      DO 35 ID=1,NDIM                                                   INSR  55
      SUM=ZERO                                                          INSR  56
      DO 30 I=1,NDN                                                     INSR  57
   30 SUM=SUM+SHFN(I)*ELCOD(ID,I)                                       INSR  58
   35 CIP(ID)=SUM                                                       INSR  59
C                                                                       INSR  60
      IF(ICAM.NE.1)GO TO 38                                             INSR  61
      EI=VARINT(NS+2,IP,J)                                              INSR  62
      PCI=VARINT(NS+3,IP,J)                                             INSR  63
      PE=(VARINT(1,IP,J)+VARINT(2,IP,J)+VARINT(3,IP,J))*0.333333333     INSR  64
      QE=Q(VARINT(1,IP,J),NS,NDIM)                                      INSR  65
      WRITE(IW6,903)IP,(CIP(ID),ID=1,NDIM),                             INSR  66
     1 (VARINT(IK,IP,J),IK=1,NS1),PE,QE,PCI,EI                          INSR  67
      GO TO 40                                                          INSR  68
   38 WRITE(IW6,903)IP,(CIP(ID),ID=1,NDIM),(VARINT(IK,IP,J),IK=1,NS1)   INSR  69
   40 CONTINUE                                                          INSR  70
   60 CONTINUE                                                          INSR  71
      RETURN                                                            INSR  72
  901 FORMAT(1X,7H ELM-IP,5X,1HX,11X,1HY,11X,2HSX,10X,                  INSR  73
     1 2HSY,10X,2HSZ,10X,3HTXY,9X,1HU,10X,2HPE,                         INSR  74
     2 11X,1HQ,10X,2HPC,7X,4HVOID)                                      INSR  75
  902 FORMAT(I4)                                                        INSR  76
  903 FORMAT(1X,I5,10E12.4,F7.4)                                        INSR  77
  910 FORMAT(1X,7HELEMENT,I5,2X,27HIS OF UNKNOWN MATERIAL TYPE,I5,      INSR  78
     1 2X,16H(ROUTINE INSTRS))                                          INSR  79
      END                                                               INSR  80
```

INSR 23 : loop on all elements in **user** number sequence.
INSR 25 : **program** element no. (J).
INSR 27 : skip if element is not present in **primary** mesh.
INSR 28–31 : element type dependent parameters.
- NDN – no. of displacement nodes.
- NGP – no. of integration points.
- INDX – starting index to arrays W and L for different element types.
- KM – material **zone** number.
- KGO – material **type** number.

INSR 34–39 : separate elements into two categories.
- ICAM = 1, Cam-clay element.
- = 0, otherwise.

INSR 43–46 : copy nodal co-ordinates into local array ELCOD.
INSR 48 : loop on all integration points.
INSR 51–52 : local/area co-ordinates of the integration point.
INSR 53 : calculate shape functions SHFN.

INSR 55–59 : calculate integration point co-ordinates.
INSR 62–65 : calculate following parameters for Cam-clay models only.
- EI – voids ratio.
- PCI – pre-consolidation pressure (size of yield locus).
- PE – mean normal effective stress (p').
- QE – deviator stress (q).

INSR 66–67 : print out for Cam-clay elements.
INSR 69 : print out for non-Cam-clay elements.
INSR 70 : end of integration point loop.
INSR 71 : end of element loop.

7.8 PRESSURE LOADS AND BOUNDARY CONDITIONS

7.8.1 Pressure loads

The external loads which are in equilibrium with *in situ* stresses are now read in. These loads are specified as pressure loads acting along element sides which lie along the boundary. The pressures along the boundary which are restrained need not be specified. It is sufficient to specify the restraint boundary condition along these sides. Neither the pressures nor the restraint boundary conditions need to be specified along **free** surfaces. A free surface is defined as any boundary free of stress and restraint (e.g. ground surface).

The pressure loads along loading boundaries are read in routine INSITU. Routine **EDGLD** checks that the element side belongs to the element specified, and aligns the nodes to follow the anti-clockwise order. The pressure values are then stored in an array PRESLD in a named COMMON block PRSLD.

Routine EDGLD

```
      SUBROUTINE EDGLD(IW6,NEL,NDIM,NTPE,NNZ,MUMAX,NPL,NCONN,LTYP,MREL,  EDGL   1
     1 NREL,LNE,ND1,ND2,NP1,NP2,PDISLD,PRES,KLOD,NPT,KINS,MXLD)          EDGL   2
C*********************************************************************EDGL   3
C     ROUTINE TO ALIGN NODES ALONG LOADED EDGE IN THE ANTI-CLOCKWISE    *EDGL   4
C     ORDER AND TO STORE THE INFORMATION                                *EDGL   5
C     THE PRESSURES AT THE BEGINNING OF AN INCREMENT BLOCK ARE STORED   *EDGL   6
C     IN A TEMPORARY ARRAY COMMON BLOCK PRLDI                           *EDGL   7
C     THE RATIOS OF THESE LOADING ARE ADDED TO THE CUMULATIVE LIST      *EDGL   8
C     (COMMON BLOCK PRSLD)                                              *EDGL   9
C     OF PRESSURE LOADS AT THE BEGINNING OF EACH INCREMENT              *EDGL  10
C*********************************************************************EDGL  11
      DIMENSION NCONN(NTPE,NEL),LTYP(NEL),NP1(NPL),NP2(NPL)             EDGL  12
      DIMENSION NREL(NNZ),MREL(MUMAX)                                   EDGL  13
      DIMENSION PDISLD(NDIM,NPT),PRES(NDIM,NPT)                         EDGL  14
      COMMON /ELINF / LINFO(50,15)                                      EDGL  15
      COMMON /PRLDI / PRSLDI(10,100),LEDI(100),NDI1(100),NDI2(100),ILOD EDGL  16
C                                                                       EDGL  17
      CALL ZEROR2(PRES,NDIM,NPT)                                        EDGL  18
      NE=MREL(LNE)                                                      EDGL  19
      LI1=NREL(ND1)                                                     EDGL  20
      LI2=NREL(ND2)                                                     EDGL  21
      LT=LTYP(NE)                                                       EDGL  22
      IF(LT.GT.0)GOTO 15                                                EDGL  23
      WRITE(IW6,901)NE                                                  EDGL  24
  901 FORMAT(1X,7HELEMENT,I6,2X,27HNOT PRESENT IN CURRENT MESH,         EDGL  25
```

```
     1 1X,16H(ROUTINE  EDGLD))                                          EDGL  26
       RETURN                                                           EDGL  27
    15 NEDG=LINFO(3,LT)                                                 EDGL  28
       NDSD=LINFO(7,LT)                                                 EDGL  29
       NTSD=NDSD+2                                                      EDGL  30
       INDED=LINFO(14,LT)                                               EDGL  31
C                                                                       EDGL  32
       DO 20 K1=1,NEDG                                                  EDGL  33
       J1=NP1(K1+INDED)                                                 EDGL  34
       J2=NP2(K1+INDED)                                                 EDGL  35
       I1=NCONN(J1,NE)                                                  EDGL  36
       I2=NCONN(J2,NE)                                                  EDGL  37
       IF(LI1.EQ.I1.AND.LI2.EQ.I2)GO TO 25                              EDGL  38
       IF(LI1.EQ.I2.AND.LI2.EQ.I1)GO TO 21                              EDGL  39
    20 CONTINUE                                                         EDGL  40
       WRITE(IW6,903)KLOD,LNE,ND1,ND2                                   EDGL  41
   903 FORMAT(/13H **** ERROR :,I5,17H TH LOAD. ELEMENT,I5,             EDGL  42
      1 2X,25H DOES NOT CONTAIN NODES :,2I5,                            EDGL  43
      2 2X,15H(ROUTINE EDGLD))                                          EDGL  44
       STOP                                                             EDGL  45
C-------------------------------------------------------------------    EDGL  46
C      ALIGN NODES IN SEQUENCE                                          EDGL  47
C-------------------------------------------------------------------    EDGL  48
    21 LIT=LI1                                                          EDGL  49
       LI1=LI2                                                          EDGL  50
       LI2=LIT                                                          EDGL  51
       NT=ND1                                                           EDGL  52
       ND1=ND2                                                          EDGL  53
       ND2=NT                                                           EDGL  54
C-------------------------------------------------------------------    EDGL  55
C      PRES - CONTAINS THE PRESSURE COMPONENTS ALIGNED IN SEQUENCE      EDGL  56
C-------------------------------------------------------------------    EDGL  57
       DO 24 J=1,NTSD                                                   EDGL  58
       JBACK=NTSD+1-J                                                   EDGL  59
       DO 24 I=1,2                                                      EDGL  60
    24 PRES(I,J)=PDISLD(I,JBACK)                                        EDGL  61
       GO TO 35                                                         EDGL  62
C                                                                       EDGL  63
    25 DO 30 J=1,NTSD                                                   EDGL  64
       DO 30 I=1,2                                                      EDGL  65
    30 PRES(I,J)=PDISLD(I,J)                                            EDGL  66
C-------------------------------------------------------------------    EDGL  67
C      UPDATE OR READ IN A NEW LIST                                     EDGL  68
C-------------------------------------------------------------------    EDGL  69
    35 IF(KINS.EQ.0)GO TO 40                                            EDGL  70
C-------------------------------------------------------------------    EDGL  71
C      PRESSURE LOADS IN EQUILIBRIUM WITH IN-SITU STRESSES              EDGL  72
C      NEW LIST - READ DIRECTLY INTO COMMON PRSLD                       EDGL  73
C-------------------------------------------------------------------    EDGL  74
       CALL LODLST(IW6,LNE,ND1,ND2,PRES,NDIM,NPT,1,MXLD)                EDGL  75
       GO TO 55                                                         EDGL  76
C-------------------------------------------------------------------    EDGL  77
C      PRESSURE LOADS FOR NEW INCREMENT BLOCK READ INTO COMMON PRSLDI   EDGL  78
C-------------------------------------------------------------------    EDGL  79
    40 ILOD=KLOD                                                        EDGL  80
       LEDI(ILOD)=LNE                                                   EDGL  81
       NDI1(ILOD)=ND1                                                   EDGL  82
       NDI2(ILOD)=ND2                                                   EDGL  83
       IC=0                                                             EDGL  84
       DO 50 IV=1,NTSD                                                  EDGL  85
       DO 50 IJ=1,2                                                     EDGL  86
       IC=IC+1                                                          EDGL  87
    50 PRSLDI(IC,ILOD)=PRES(IJ,IV)                                      EDGL  88
    55 CONTINUE                                                         EDGL  89
       RETURN                                                           EDGL  90
       END                                                              EDGL  91
```


EDGL 18 : zero array PRES (which temporarily holds the applied pressure load).

EDGL 19 : program element number.

EDGL 20–21 : program node numbers of nodes at either end.

EDGL 22 : element type number.

EDGL 28–31 : element type dependent parameters.

- NEDG – no. of element sides (edges).
- NDSD – no. of displacement nodes along side (excluding nodes at either end).
- NTSD – total no. of displacement nodes along side.
- INDED – starting index to arrays NP1, NP2.

EDGL 33–40 : find element side with applied pressure load by comparing nodes at either end (normal and reverse sequence).

EDGL 41–44 : side not found in element; stop.

EDGL 49–54 : side found; reverse the nodes to conform with anti-clockwise sequence.

EDGL 58–61 : do the same with pressure components.

EDGL 64–66 : array PRES contains pressure load terms in correct sequence.

EDGL 70 : skip if load is for an increment block.

EDGL 75 : read directly into PRESLD in named COMMON PRSLD.

EDGL 80–88 : read into temporary array PRSLDI in named COMMON PRLDI.

Routine LODLST

```
      SUBROUTINE LODLST(IW6,LNE,ND1,ND2,PRES,NDIM,NPT,ILST,MXLD)        LDLS   1
C*********************************************************************LDLS   2
C     ROUTINE TO STORE CUMULATIVE LIST OF APPLIED                       LDLS   3
C     PRESSURE LOADING ALONG ELEMENT EDGES                              LDLS   4
C*********************************************************************LDLS   5
      DIMENSION PRES(NDIM,NPT)                                          LDLS   6
      COMMON /PRSLD / PRESLD(10,100),LEDG(100),NDE1(100),NDE2(100),NLED LDLS   7
C-------------------------------------------------------------------    LDLS   8
C     MXLD - SIZE OF ARRAYS LEDG,NDE1,NDE2,PRESLD (ROUTINE MAXVAL)      LDLS   9
C-------------------------------------------------------------------    LDLS  10
C----------SKIP IF NEW LIST                                             LDLS  11
      IF(NLED.EQ.0.OR.ILST.EQ.1)GO TO 22                                LDLS  12
C-------------------------------------------------------------------    LDLS  13
C     SEARCH FOR LNE IN EXISTING LIST                                   LDLS  14
C-------------------------------------------------------------------    LDLS  15
      DO 20 J=1,NLED                                                    LDLS  16
      IF(LNE.NE.LEDG(J))GO TO 20                                        LDLS  17
      N1=NDE1(J)                                                        LDLS  18
      N2=NDE2(J)                                                        LDLS  19
      IF(N1.EQ.ND1.AND.N2.EQ.ND2)GO TO 25                               LDLS  20
   20 CONTINUE                                                          LDLS  21
C-------------------------------------------------------------------    LDLS  22
C     ADD NEW EDGE TO THE LIST                                          LDLS  23
C-------------------------------------------------------------------    LDLS  24
   22 NLED=NLED+1                                                       LDLS  25
      IF(NLED.LE.MXLD)GO TO 23                                          LDLS  26
      WRITE(IW6,900)                                                    LDLS  27
  900 FORMAT(/27H INCREASE SIZE OF ARRAYS IN,                           LDLS  28
     1 51H COMMON BLOCK PRSLD ALSO SET MXLD IN ROUTINE MAXVAL/          LDLS  29
```

```
     2 25X,16H(ROUTINE LODLST))                                        LDLS  30
       STOP                                                            LDLS  31
    23 JE=NLED                                                         LDLS  32
       GO TO 30                                                        LDLS  33
C-------------------------------------------------------------------LDLS  34
C      UPDATE EXISTING LIST                                            LDLS  35
C-------------------------------------------------------------------LDLS  36
    25 JE=J                                                            LDLS  37
       GO TO 35                                                        LDLS  38
C                                                                      LDLS  39
    30 LEDG(JE)=LNE                                                    LDLS  40
       NDE1(JE)=ND1                                                    LDLS  41
       NDE2(JE)=ND2                                                    LDLS  42
C                                                                      LDLS  43
    35 IC=0                                                            LDLS  44
       DO 40 IPT=1,NPT                                                 LDLS  45
       DO 40 IK=1,NDIM                                                 LDLS  46
       IC=IC+1                                                         LDLS  47
    40 PRESLD(IC,JE)=PRESLD(IC,JE)+PRES(IK,IPT)                        LDLS  48
       RETURN                                                          LDLS  49
       END                                                             LDLS  50
```

LDLS 12 : skip if no existing list; therefore no need to scan.
LDLS 16 : loop on list of pressure loads.
LDSL 17 : not this element; look at next one.
LDLS 18–19 : nodes at either end of side.
LDLS 20 : element side has been found.
LDLS 21 : end of existing list.
LDLS 25 : it is a new element side with pressure load.
LDLS 27–30 : array size exceeded. Arrays LEDG, NDE1, NDE2 and PRESLD have to be increased in size. (Also make changes in all routines in which these appear. See Appendix C, which gives the list of routines.)
LDLS 32 : new position at end of list.
LDLS 37–38 : get the position in existing list; skip, as entries are not altered.
LDLS 40–42 : enter details for new side.
LDLS 45–48 : update pressure loads.

These two routines are also called when there are pressure loads applied along element sides in an increment block. Under these circumstances the applied pressure loads are stored in a separate set of arrays in named COMMON block PRLDI. At the beginning of each increment the ratio of load applied in that increment is added to the list of cumulative load array PRESLD. This procedure is adopted purely for equilibrium checks done at the end of each increment. At any given increment the stresses and the applied loads can be directly checked against each other.

7.8.2 Fixities

The details of restrained element sides are read in routine **FIXX**. The input data are read element side by element side. The element side is identified by nodes at either end, and the direction in which they are restrained is also specified. If an

element side is fixed in more than one direction then one entry (data record) is required per direction.

The routine checks the correctness of the node numbers with the nodes associated with the element. The nodal sequence is aligned to follow the anti-clockwise order about the element centre. Then the fixity information along the element side is converted into nodal fixities at all nodes which lie along this element side.

The same routine is called either to restrain element sides or to give the element side a prescribed displacement or excess pore pressure. The prescribed values are stored in the array DXYT(4,200). This allows for a maximum of 200 nodes rather arbitrarily. A maximum of 4 d.o.f. can be fixed at any given node; only the first three are used for two-dimensional analysis.

1	2	3	4
x-disp.	y-disp.	ex. p.p.	– for 2-D

where 'ex. p.p' denotes excess pore pressure. TF(4,200) stores the fixity code, which can take 1 or 0 for the displacements, and 0, 1 or 2 for the excess pore pressure.

1 – to specify the incremental value of displacement/excess p.p.
2 – to specify the **absolute** value of excess pore pressure.

There is a distinction between restraints and prescribed displacement/excess pore pressures. (The restraints are identified by zero values for the prescribed variables.) The displacement restraint is self-explanatory. For the excess pore pressures, if fixity code 1 is used along a boundary with zero prescribed values to represent, say, a draining boundary, then no changes in pore pressure take place. In that sense it is a pore pressure restraint.

It is appropriate to define the terminology used for the excess pore pressures because some terms are invented to have a precise meaning in relation to CRISP. The hydrostatic pore pressures at rest are referred to as *in situ* pore pressures. Since the program uses an incremental approach, the changes that take place in displacements are referred to as incremental displacements – hence the term 'incremental (excess) pore pressure'. Accumulated displacements over a number of increments are cumulative or **absolute** displacements. Similarly the summation of incremental changes to the excess pore pressures are referred to as **absolute** excess pore pressures. Therefore the pore pressures at any instance (i.e. the **total** pore pressure) are given by the *in situ* pore pressure plus the absolute excess pore pressure. Therefore the term '**absolute excess** pore pressure' is simply the accumulated changes in the excess pore pressure over a number of increments.

At the end of each increment block, all the prescribed values are set to zero. However, no changes are made to the fixity code of these nodes. Therefore there is no 'carry over' from one increment block to the next, i.e. 'no memory' in the case of prescribed displacement. However, there is a carry over in the sense that previously prescribed values are now fixed to zero. This procedure is adopted so

that restraint boundary conditions need not be specified in every increment block. They need to be specified only once, either with the *in situ* boundary condition or in the first increment block.

Routine FIXX

```
      SUBROUTINE FIXX(IR5,IW6,NEL,NTPE,NDIM,NPL,LV,MUMAX,NNZ,NCONN,LTYP,FIXX   1
     1 MREL,NREL,NP1,NP2,V,NFX)                                          FIXX   2
C*********************************************************************FIXX   3
C     ROUTINE TO MAINTAIN A LIST OF NODAL FIXITIES. INTERPRETS          FIXX   4
C     FIXITIES ALONG ELEMENT EDGES INTO NODAL FIXITIES                  FIXX   5
C*********************************************************************FIXX   6
      INTEGER TF                                                        FIXX   7
      DIMENSION NCONN(NTPE,NEL),LTYP(NEL),MREL(MUMAX),NREL(NNZ)         FIXX   8
      DIMENSION NP1(NPL),NP2(NPL),IND(5),FV(5),V(LV)                    FIXX   9
      COMMON /FIX   / DXYT(4,200),MF(200),TF(4,200),NF                  FIXX  10
      COMMON /ELINF / LINFO(50,15)                                      FIXX  11
                                                                        FIXX  12
C                                                                       FIXX  13
      NFZ=200                                                           FIXX  14
      NDIM1=NDIM+1                                                      FIXX  15
      IF(NFX.EQ.0)RETURN                                                FIXX  16
      WRITE(IW6,900)                                                    FIXX  17
C-----------------------------------------------------------------------FIXX  18
C     LOOP ON ALL FIXED EDGES I.E. EDGES WITH PRESCRIBED                FIXX  19
C     DISPLACEMENT/EXCESS PORE PRESSURES                                FIXX  20
C-----------------------------------------------------------------------FIXX  21
      DO 200 JX=1,NFX                                                   FIXX  22
      READ(IR5,*)ML,ND1,ND2,IVAR,IFX,V                                  FIXX  23
      WRITE(6,902)JX,ML,ND1,ND2,IVAR,IFX,V                              FIXX  24
      NE=MREL(ML)                                                       FIXX  25
      LI1=NREL(ND1)                                                     FIXX  26
      LI2=NREL(ND2)                                                     FIXX  27
      LT=LTYP(NE)                                                       FIXX  28
      LT=IABS(LT)                                                       FIXX  29
      NVN=LINFO(2,LT)                                                   FIXX  30
      NEDG=LINFO(3,LT)                                                  FIXX  31
      NDSD=LINFO(7,LT)                                                  FIXX  32
      IF(IVAR.EQ.NDIM1)NDSD=LINFO(8,LT)                                 FIXX  33
      NTSD=NDSD+2                                                       FIXX  34
      INDED=LINFO(14,LT)                                                FIXX  35
C                                                                       FIXX  36
      DO 20 K1=1,NEDG                                                   FIXX  37
      J1=NP1(K1+INDED)                                                  FIXX  38
      J2=NP2(K1+INDED)                                                  FIXX  39
      I1=NCONN(J1,NE)                                                   FIXX  40
      I2=NCONN(J2,NE)                                                   FIXX  41
      IF(LI1.EQ.I1.AND.LI2.EQ.I2)GO TO 25                               FIXX  42
      IF(LI1.EQ.I2.AND.LI2.EQ.I1)GO TO 21                               FIXX  43
   20 CONTINUE                                                          FIXX  44
      WRITE(IW6,903)JX,ML,ND1,ND2                                       FIXX  45
      GOTO 200                                                          FIXX  46
C-----------------------------------------------------------------------FIXX  47
C     ALIGN END NODES OF EDGE IN CORRECT SEQUENCE. (ANTICLOCKWSIE       FIXX  48
C     OREDER ABOUT ELEMENT CENTRE)                                      FIXX  49
C-----------------------------------------------------------------------FIXX  50
   21 LIT=LI1                                                           FIXX  51
      LI1=LI2                                                           FIXX  52
      LI2=LIT                                                           FIXX  53
      NT=ND1                                                            FIXX  54
      ND1=ND2                                                           FIXX  55
      ND2=NT                                                            FIXX  56
C                                                                       FIXX  57
      DO 24 J=1,NTSD                                                    FIXX  58
      JBACK=NTSD+1-J                                                    FIXX  59
      FV(J)=V(JBACK)                                                    FIXX  60
      GO TO 35                                                          FIXX  61
C                                                                       FIXX  62
   25 DO 30 J=1,NTSD                                                    FIXX  63
   30 FV(J)=V(J)                                                        FIXX  64
C-----------------------------------------------------------------------FIXX  65
C     IND - LIST OF NODES ALONG EDGE. START WITH END NODES              FIXX  66
C-----------------------------------------------------------------------FIXX  67
   35 IND(1)=LI1                                                        FIXX  68
      IND(NTSD)=LI2                                                     FIXX  69
      IF(NTSD.EQ.2)GO TO 42                                             FIXX  70
      LC1=NVN+(K1-1)*NDSD                                               FIXX  71
      IF(IVAR.EQ.NDIM1)LC1=LINFO(5,LT)+(K1-1)*NDSD                      FIXX  72
C-----------------------------------------------------------------------FIXX  73
C     INTERMEDIATE NODES (IF NTSD=2 NO INTERMEDIATE NODES)              FIXX  74
C-----------------------------------------------------------------------FIXX  75
      DO 40 JP=1,NDSD                                                   FIXX  76
      ILC=LC1+JP                                                        FIXX  77
   40 IND(JP+1)=NCONN(ILC,NE)                                           FIXX  78
C-----------------------------------------------------------------------FIXX  79
C     LOOP ON ALL NODES ALONG EDGE                                      FIXX  80
C-----------------------------------------------------------------------FIXX  81
   42 DO 100 KND=1,NTSD                                                 FIXX  82
      I=IND(KND)                                                        FIXX  83
      IF(NF.EQ.0)GO TO 58                                               FIXX  84
C                                                                       FIXX  85
      DO 50 J=1,NF                                                      FIXX  86
      IF(I.EQ.MF(J))GO TO 55                                            FIXX  87
   50 CONTINUE                                                          FIXX  88
C                                                                       FIXX  89
      GO TO 58                                                          FIXX  90
C-----------------------------------------------------------------------FIXX  91
C     UPDATE EXISTING VALUES                                            FIXX  92
C-----------------------------------------------------------------------FIXX  93
   55 JF=J                                                              FIXX  94
      GO TO 60                                                          FIXX  95
C                                                                       FIXX  96
   58 NF=NF+1                                                           FIXX  97
      IF(NF.LE.NFZ)GO TO 59                                             FIXX  98
      WRITE(IW6,904)                                                    FIXX  99
      STOP                                                              FIXX 100
   59 JF=NF                                                             FIXX 101
   60 MF(JF)=I                                                          FIXX 102
      TF(IVAR,JF)=IFX                                                   FIXX 103
      DXYT(IVAR,JF)=FV(KND)                                             FIXX 104
  100 CONTINUE                                                          FIXX 105
  200 CONTINUE                                                          FIXX 106
      RETURN                                                            FIXX 107
  900 FORMAT(/1X,4HSIDE,4X,7HELEMENT,3X,5HNODE1,3X,5HNODE2,             FIXX 108
     1 3X,3HDOF,3X,11HFIXITY CODE,6X,4HVAL1,6X,4HVAL2,6X,4HVAL3,        FIXX 109
     2 6X,4HVAL4,6X,4HVAL5/)                                            FIXX 110
  902 FORMAT(1X,I3,4X,I5,5X,I4,4X,I4,5X,I2,12X,I3,3X,5F10.3)            FIXX 111
  903 FORMAT(/13H **** ERROR :,I5,19H TH FIXITY. ELEMENT,               FIXX 112
     1 I5,25H DOES NOT CONTAIN NODES :,2I5,2X,14H(ROUTINE FIXX))        FIXX 113
  904 FORMAT(/40H INCREASE SIZE OF ARRAYS MF, TF AND DXYT/              FIXX 114
     1 1X,34HIN COMMON BLOCK FIX (ROUTINE FIXX))                        FIXX 115
      END
```


FIXX 13 : maximum size of arrays in named COMMON FIX.

FIXX 14 : maximum number of variables at any node (last one being the pore pressure variable).

FIXX 21 : loop on all sides which have prescribed variables.

FIXX 22–23 : read and write details of side with prescribed variables.

FIXX 24 : (program no.) element with side which is fixed.
FIXX 25–26 : (program nos.) nodes at either end of side.
FIXX 27–28 : element **type** no.
FIXX 29–34 : element **type** dependent parameters.
NVN – no. of vertex nodes.
NEDG – no. of sides (edges).
NDSD – no. of displacement nodes along side (excluding end nodes).
NTSD – total no. of nodes along side.
INDED – starting index to arrays NP1, NP2.
FIXX 32 : NDSD – no. of pore pressure nodes along side (excluding end nodes).
FIXX 36 : loop on all edges of element (to find side which is fixed).
FIXX 37–42 : find element side with prescribed variable by comparing nodes at either end (normal and reverse sequence).
FIXX 44 : side not found in element; consider next side with prescribed variable, after printing message.
FIXX 50–55 : side found; reverse nodes to conform with anti-clockwise sequence.
FIXX 57–59 : do the same with prescribed values.
FIXX 62–63 : array FV contains prescribed values in correct sequence.
FIXX 67–68 : enter nodes at either end in IND.
FIXX 69 : skip if no nodes along side.
FIXX 70–71 : index to array NCONN for nodes along side.
FIXX 75–77 : enter node(s) along side in IND.
FIXX 81 : loop on all nodes along side.
FIXX 83 : skip if first node (i.e. no existing list).
FIXX 85–86 : scan through existing list.
FIXX 93 : position of node in existing list.
FIXX 96–97 : new node; add to the end of the list. Increment count on no. of fixities.
FIXX 98–99 : if allocation of array size is exceeded, print message and stop.
FIXX 101–103 : enter details of nodal fixity (fixity code and prescribed values) – pore pressure variable is placed in location NDIM + 1, even if it is the only variable at that node.
FIXX 104 : end of loop on all nodes along side.
FIXX 105 : end of loop on all sides with prescribed variables.

7.9 EQUILIBRIUM CHECK

Routine **EQLOD** is the master control routine, which checks the equilibrium of internal stresses with external loading. (For convenience, the self-weight loading is considered as part of the external loading.)

The first term of (7.9) on the R.H.S. is calculated by routine DISTLD and SFR1. The second term is calculated by SELF (making use of SHAPE and DETJCB). The third term has already been calculated in routine RDSTRS using EQLIB and placed in array PEQT. Routine RESTRN recognises the nodes which are restrained. The following calculation is carried out to calculate P_{cor}.

$$P_{cor} = \int_S \mathbf{N}^T \boldsymbol{\tau}\, d(\text{area}) + \int_V \mathbf{N}^T \mathbf{w}\, d(\text{vol}) - \int_V \mathbf{B}^T \boldsymbol{\sigma}\, d(\text{vol}). \quad (7.9)$$

$\int_S \mathbf{N}^T \boldsymbol{\tau}\, d(\text{area})$ – pressure loads along element boundary.

$\int_V \mathbf{N}^T \mathbf{w}\, d(\text{vol})$ – self-weight or distributed loads.

$\int_V \mathbf{B}^T \boldsymbol{\sigma}\, d(\text{vol})$ – nodal loads equivalent to element stresses summed for all elements present in current mesh.

P_{cor} – the error in equilibrium calculated for each nodal point except the ones which are either restrained or have prescribed values.

Routine EQLOD

```
      SUBROUTINE EQLOD(IW6,NN,NEL,NDF,NNOD1,NTPE,NDIM,MUMAX,NNZ,NDZ,NPR,EQLD    1
     1 NMT,NDMX,NL,NPL,NCONN,MAT,LTYP,MRELVV,MREL,NREL,NW,NQ,JEL,IDFX,  EQLD    2
     2 NP1,NP2,XYZ,P,PT,PCOR,PEQT,XYFT,PCONI,ELCOD,DS,SHFN,F,LL,        EQLD    3
     3 PR,NPT,NSP,MXEN,IEQOP,ICOR,TGRAV,IRAC,FRACT,KSTGE)               EQLD    4
C*********************************************************************EQLD    5
C     ROUTINE TO CALCULATE EQUIVALENT NODAL LOADS FOR                  EQLD    6
C     APPLIED LOADING TO CARRY OUT AN EQUILIBRIUM CHECK                EQLD    7
C*********************************************************************EQLD    8
      REAL LL                                                          EQLD    9
      DIMENSION NCONN(NTPE,NEL),MAT(NEL),LTYP(NEL),MRELVV(NEL),        EQLD   10
     1 MREL(MUMAX),NREL(NNZ),NW(NNOD1),NQ(NN),JEL(NEL),                EQLD   11
     2 IDFX(NDF),NP1(NPL),NP2(NPL)                                     EQLD   12
      DIMENSION XYZ(NDIM,NN),P(NDF),PT(NDF),PCOR(NDF),PEQT(NDF),       EQLD   13
     1 XYFT(NDF),PCONI(NDF),ELCOD(NDIM,NDMX),DS(NDIM,NDMX),SHFN(NDMX), EQLD   14
     2 F(NDIM,NDMX),LL(NL),PR(NPR,NMT),PRES(10)                        EQLD   15
      COMMON /PRSLD / PRESLD(10,100),LEDG(100),NDE1(100),NDE2(100),NLED EQLD   16
      COMMON /ELINF / LINFO(50,15)                                     EQLD   17
      COMMON /PARS  / PYI,ALAR,ASMVL,ZERO                              EQLD   18
C                                                                      EQLD   19
      CALL ZEROR1(PT,NDF)                                              EQLD   20
C----------------------------------------------------------------------EQLD   21
C     (1) PRESSURE LOADING ALONG ELEMENT EDGE                          EQLD   22
C----------------------------------------------------------------------EQLD   23
      IF(NLED.EQ.0.AND.TGRAV.LT.ASMVL)GO TO 62                         EQLD   24
      IF(NLED.EQ.0)GO TO 32                                            EQLD   25
C                                                                      EQLD   26
      DO 30 KE=1,NLED                                                  EQLD   27
      LNE=LEDG(KE)                                                     EQLD   28
      NE=MREL(LNE)                                                     EQLD   29
      LT=LTYP(NE)                                                      EQLD   30
```

```
      IF(LT.GT.0)GOTO 10                                                EQLD  31
      IF(KSTGE.EQ.4)GOTO 30                                             EQLD  32
      WRITE(IW6,900)LNE                                                 EQLD  33
  900 FORMAT(/1X,45H *** ERROR : IN SITU PRESSURE LOAD APPLIED TO,1X,    EQLD  34
     1  7HELEMENT,I5,2X,28HWHICH IS NOT PRESENT IN MESH,1X,             EQLD  35
     2  15H(ROUTINE EQLOD)/)                                           EQLD  36
      GOTO 30                                                           EQLD  37
   10 ND1=NDE1(KE)                                                      EQLD  38
      ND2=NDE2(KE)                                                      EQLD  39
      DO 20 KV=1,MXEN                                                   EQLD  40
   20 PRES(KV)=PRESLD(KV,KE)                                            EQLD  41
C                                                                       EQLD  42
      CALL DISTLD(IW6,NN,NEL,NDF,NNOD1,NTPE,NDIM,MUMAX,NNZ,NPL,XYZ,PT,  EQLD  43
     1   NCONN,LTYP,MREL,NREL,NW,NP1,NP2,PRES,LNE,ND1,ND2,               EQLD  44
     2   NPT,NSP,0,1,1.)                                                EQLD  45
   30 CONTINUE                                                          EQLD  46
C-----------------------------------------------------------------------EQLD  47
C     (2) SELF WEIGHT LOADING                                           EQLD  48
C-----------------------------------------------------------------------EQLD  49
   32 IF(TGRAV.LT.ASMVL) GO TO 62                                       EQLD  50
      DO 60 KL=1,NEL                                                    EQLD  51
      LT=LTYP(KL)                                                       EQLD  52
      IF(LT.LT.0)GO TO 60                                               EQLD  53
      JK=MRELVV(KL)                                                     EQLD  54
      NDN=LINFO(5,LT)                                                   EQLD  55
      INDX=LINFO(12,LT)                                                 EQLD  56
      NAC=LINFO(15,LT)                                                  EQLD  57
      KM=MAT(KL)                                                        EQLD  58
C-----------------------------------------------------------------------EQLD  59
C     FIND IF ELEMENT HAS BEEN ADDED IN THIS INCREMENT BLOCK            EQLD  60
C     THEN USE LOAD RATIO FRACT ON GRAVITY LOADING                      EQLD  61
C-----------------------------------------------------------------------EQLD  62
      DO 40 IM=1,NEL                                                    EQLD  63
      MUS=JEL(IM)                                                       EQLD  64
      IF(MUS.EQ.0)GO TO 42                                              EQLD  65
      MPR=MREL(MUS)                                                     EQLD  66
      IF(KL.EQ.MPR)GO TO 44                                             EQLD  67
   40 CONTINUE                                                          EQLD  68
   42 FA=1.                                                             EQLD  69
      GO TO 45                                                          EQLD  70
   44 FA=FRACT                                                          EQLD  71
   45 DENS=PR(8,KM)*TGRAV*FA                                            EQLD  72
C                                                                       EQLD  73
      CALL SELF(IW6,KL,NN,NEL,NTPE,NDN,NDIM,NAC,NPR,NMT,XYZ,            EQLD  74
     1   ELCOD,DS,SHFN,F,NCONN,MAT,LL,PR,LT,INDX,DENS,JK,KSTGE)         EQLD  75
C                                                                       EQLD  76
      DO 55 KK=1,NDN                                                    EQLD  77
      NCOR=NCONN(KK,KL)                                                 EQLD  78
      KKK=NW(NCOR)-1                                                    EQLD  79
C                                                                       EQLD  80
      DO 55 ID=1,NDIM                                                   EQLD  81
   55 PT(KKK+ID)=PT(KKK+ID)+F(ID,KK)                                    EQLD  82
   60 CONTINUE                                                          EQLD  83
   62 CONTINUE                                                          EQLD  84
C-----------------------------------------------------------------------EQLD  85
C     ADD CONTRIBUTIONS FROM POINT LOADS                                EQLD  86
C-----------------------------------------------------------------------EQLD  87
      DO 70 J=1,NDF                                                     EQLD  88
   70 PT(J)=PT(J)+XYFT(J)+PCONI(J)                                      EQLD  89
C-----------------------------------------------------------------------EQLD  90
C     FIND DOF WHICH ARE RESTRAINED                                     EQLD  91
C-----------------------------------------------------------------------EQLD  92
      CALL RESTRN(NDF,NNOD1,NDIM,NW,IDFX)                               EQLD  93
C-----------------------------------------------------------------------EQLD  94
C     EQUILIBRIUM CHECK                                                 EQLD  95
```



```
C-----------------------------------------------------------------------EQLD  96
      CALL EQLBM(IW6,NN,NNOD1,NDF,NDIM,NNZ,NDZ,NREL,NW,NQ,IDFX,         EQLD  97
     1   P,PT,PCOR,PEQT,IEQOP,ICOR,IRAC)                                EQLD  98
      RETURN                                                            EQLD  99
      END                                                               EQLD 100
```

EQLD 20 : zero total load array (PT).
EQLD 24 : skip if (a) no applied pressure loads and (b) no gravity loading.
EQLD 25 : skip if no applied pressure loads.
EQLD 27 : loop on all sides with applied pressure loads.
EQLD 31 : skip if element with pressure load is present in mesh.
EQLD 32 : skip check and calculation of equivalent pressure loads i element has been removed.
EQLD 33–36 : print message if element is not present at *in situ* stage (probable user error).
EQLD 38–39 : nodes at either end.
EQLD 40–41 : values of applied pressure loads.
EQLD 43–45 : calculate nodal loads from pressure loading and put into PT.
EQLD 50 : skip if no gravity loads.
EQLD 51 : loop on all elements.
EQLD 53 : skip if element is not present in current mesh.
EQLD 55–57 : NDN – no. of displacement nodes.
INDX – starting index for arrays W and L, for different element types.
EQLD 58 : material zone number.
EQLD 63 : scan array JEL to see if element was added in this block.
EQLD 65 : zero indicates end of list.
EQLD 66–67 : element has been added in this block.
EQLD 69 : use factor of 1 for elements which were already present before the start of current block.
EQLD 71 : use FRACT for added element.
EQLD 72 : calculate $n\gamma$ term.
n – centrifugal acceleration field.
EQLD 74–75 : calculate $\int_V \mathbf{N}^T \mathbf{w}\, d\,(\text{vol})$.
EQLD 77 : loop on all nodes of element.
EQLD 81–82 : slot loads in PT.
EQLD 83 : end of element loop.
EQLD 88–89 : add directly specified point loads.
EQLD 93 : identify (by entering 1 in IDFX against d.o.f.) restrained d.o.f.
EQLD 97–98 : carry out an equilibrium check.

Routine **EQLOD** is called at the *in situ* stage as well as at the end of each increment.

7.9.1 Pressure loads

Routine **DISTLD** calculates nodal loads equivalent to the current pressure loading, and τ values are obtained from array PRES.

Routine DISTLD

```
      SUBROUTINE DISTLD(IW6,NN,NEL,NDF,NNOD1,NTPE,NDIM,MUMAX,NNZ,       DIST   1
     1 NPL,XYZ,RHS,NCONN,LTYP,MREL,NREL,NW,NP1,NP2,PRES,LNE,            DIST   2
     2 ND1,ND2,NPT,NSP,IPRINT,IST,FC)                                   DIST   3
C*********************************************************************DIST   4
C     ROUTINE TO CALCULATE EQUIVALENT NODAL LOADS FOR SPECIFIED        *DIST   5
C     PRESSURE LOADING ALONG ELEMENT EDGES USING 5 POINT (NSP)         *DIST   6
C     INTEGRATION RULE. INTEGRATES POLYNOMIAL OF ORDER NINE OR LESS    *DIST   7
C     EXACTLY. ARRAYS ILOC,PRES,PEQLD,ELCD,SHF,DERIV ARE               *DIST   8
C     TO CATER FOR A MAXIMUM OF FIVE NODES (NPT) ALONG AN ELEMENT EDGE *DIST   9
C     (ALL 2-D ELEMENTS UP TO ORDER FIVE).                             *DIST  10
C*********************************************************************DIST  11
      DIMENSION NCONN(NTPE,NEL),LTYP(NEL),MREL(MUMAX),                  DIST  12
     1 NREL(NNZ),NW(NNOD1),NP1(NPL),NP2(NPL)                            DIST  13
      DIMENSION RHS(NDF),XYZ(NDIM,NN),PRES(NDIM,NPT)                    DIST  14
      DIMENSION ILOC(5),PSP(2),DSP(2),PEQLD(2,5),ELCD(2,5)              DIST  15
      DIMENSION SHF(5),DERIV(5),PCOM(3)                                 DIST  16
      COMMON /FLOW  / NPLAX                                             DIST  17
      COMMON /ELINF / LINFO(50,15)                                      DIST  18
      COMMON /SAMP  / POSSP(5),WEIGP(5)                                 DIST  19
      COMMON /PARS  / PYI,ALAR,ASMVL,ZERO                               DIST  20
      COMMON /LOADS / FB(2,15)                                          DIST  21
      NP=5                                                              DIST  22
      TPI=2.*PYI                                                        DIST  23
      NE=MREL(LNE)                                                      DIST  24
      LI1=NREL(ND1)                                                     DIST  25
      LT=LTYP(NE)                                                       DIST  26
      IF(IST.EQ.1)GOTO 5                                                DIST  27
      LT=IABS(LT)                                                       DIST  28
    5 IF(LT.GT.0)GOTO 10                                                DIST  29
      WRITE(IW6,900)LNE                                                 DIST  30
  900 FORMAT(/1X,44H**** ERROR : YOU HAVE PUT A PRESSURE LOAD ON,       DIST  31
     1           8H ELEMENT,I5,2X,28HWHICH IS NOT PRESENT IN MESH,      DIST  32
     2           17H (ROUTINE DISTLD)/)                                 DIST  33
      RETURN                                                            DIST  34
   10 NVN=LINFO(2,LT)                                                   DIST  35
      NEDG=LINFO(3,LT)                                                  DIST  36
      NDSD=LINFO(7,LT)                                                  DIST  37
      NTSD=NDSD+2                                                       DIST  38
      INDED=LINFO(14,LT)                                                DIST  39
C                                                                       DIST  40
      DO 20 K1=1,NEDG                                                   DIST  41
      J1=NP1(K1+INDED)                                                  DIST  42
      J2=NP2(K1+INDED)                                                  DIST  43
      I1=NCONN(J1,NE)                                                   DIST  44
      IF(LI1.EQ.I1)GOTO 25                                              DIST  45
   20 CONTINUE                                                          DIST  46
      WRITE(IW6,903)LNE,ND1,ND2                                         DIST  47
  903  FORMAT(/21H **** ERROR : ELEMENT,I5,                             DIST  48
     1 2X,22H DOES NOT HAVE NODES :,2I5,                                DIST  49
     2 3X,16H(ROUTINE DISTLD))                                          DIST  50
      RETURN                                                            DIST  51
C-----------------------------------------------------------------------DIST  52
C     STORE LOCATIONS OF NODE (IN NCONN) IN ARRAY ILOC                  DIST  53
C-----------------------------------------------------------------------DIST  54
   25 LC1=NVN+(J1-1)*NDSD                                               DIST  55
      ILOC(1)=J1                                                        DIST  56
      ILOC(NTSD)=J2                                                     DIST  57
      IF(NDSD.EQ.0)GOTO 31                                              DIST  58
```



```
C                                                                       DIST  59
      DO 30 JP=1,NDSD                                                   DIST  60
   30 ILOC(JP+1)=LC1+JP                                                 DIST  61
C-----------------------------------------------------------------------DIST  62
C     SET UP LOCAL ARRAY FOR CO-ORDINATES IN ELCD                       DIST  63
C-----------------------------------------------------------------------DIST  64
   31 DO 32 KC=1,NTSD                                                   DIST  65
      ILC=ILOC(KC)                                                      DIST  66
      NDE=NCONN(ILC,NE)                                                 DIST  67
C                                                                       DIST  68
      DO 32 ID=1,NDIM                                                   DIST  69
   32 ELCD(ID,KC)=XYZ(ID,NDE)                                           DIST  70
C     INITIALISE PEQLD                                                  DIST  71
      CALL ZEROR2(PEQLD,NDIM,NP)                                        DIST  72
C-----------------------------------------------------------------------DIST  73
C     LOOP FOR NUMERICAL INTEGRATION                                    DIST  74
C-----------------------------------------------------------------------DIST  75
      DO 60 ISP=1,NSP                                                   DIST  76
      XI=POSSP(ISP)                                                     DIST  77
C-----------------------------------------------------------------------DIST  78
C     EVALUATE SHAPE FUNCTION FOR SAMPLING POINT                        DIST  79
C-----------------------------------------------------------------------DIST  80
      CALL SFR1(IW6,XI,SHF,DERIV,NTSD,LNE,LT)                           DIST  81
C----------CALCULATE COMPONENTS OF THE EQUIVALENT NODAL LOADS - PEQLD   DIST  82
      DO 40 IDOF=1,NDIM                                                 DIST  83
      PSP(IDOF)=ZERO                                                    DIST  84
      DSP(IDOF)=ZERO                                                    DIST  85
C                                                                       DIST  86
      DO 40 IEDG=1,NTSD                                                 DIST  87
      PSP(IDOF)=PSP(IDOF)+PRES(IDOF,IEDG)*SHF(IEDG)                     DIST  88
   40 DSP(IDOF)=DSP(IDOF)+ELCD(IDOF,IEDG)*DERIV(IEDG)                   DIST  89
C                                                                       DIST  90
      DV=WEIGP(ISP)                                                     DIST  91
      IF(NPLAX.EQ.0)GOTO 48                                             DIST  92
      RAD=0.0                                                           DIST  93
C                                                                       DIST  94
      DO 45 IEDG=1,NTSD                                                 DIST  95
   45 RAD=RAD+ELCD(1,IEDG)*SHF(IEDG)                                    DIST  96
      DV=DV*TPI*RAD                                                     DIST  97
   48 PCOM(1)=DSP(1)*PSP(2)-DSP(2)*PSP(1)                               DIST  98
      PCOM(2)=DSP(1)*PSP(1)+DSP(2)*PSP(2)                               DIST  99
C                                                                       DIST 100
      DO 50 IEDG=1,NTSD                                                 DIST 101
      DO 50 ID=1,NDIM                                                   DIST 102
   50 PEQLD(ID,IEDG)=PEQLD(ID,IEDG)+PCOM(ID)*SHF(IEDG)*DV               DIST 103
C                                                                       DIST 104
   60 CONTINUE                                                          DIST 105
      IF(IPRINT.EQ.1)WRITE(IW6,905)LNE,ND1,ND2,                         DIST 106
     1 ((PEQLD(ID,IP),ID=1,2),IP=1,NTSD)                                DIST 107
  905 FORMAT(1X,3I4,10E12.4/)                                           DIST 108
C-----------------------------------------------------------------------DIST 109
C     SLOT LOADS INTO ARRAY RHS                                         DIST 110
C-----------------------------------------------------------------------DIST 111
      DO 80 IJ=1,NTSD                                                   DIST 112
      JL=ILOC(IJ)                                                       DIST 113
      NDE=NCONN(JL,NE)                                                  DIST 114
      N1=NW(NDE)-1                                                      DIST 115
C                                                                       DIST 116
      DO 80 ID=1,NDIM                                                   DIST 117
      FB(ID,JL)=FB(ID,JL)+PEQLD(ID,IJ)                                  DIST 118
   80 RHS(N1+ID)=RHS(N1+ID)+PEQLD(ID,IJ)*FC                             DIST 119
      RETURN                                                            DIST 120
      END                                                               DIST 121
```

DIST 24 : program element number.
DIST 25 : program node number of node at one end of side with pressure load.
DIST 26 : element type number.
DIST 27–28 : if IST = 0 then calculate loads equivalent to pressure loads acting on elements currently being removed.
DIST 29–33 : check that the element on which pressure load is put is present in mesh. If not, print error message.
DIST 35–39 : element type dependent parameters.
- NVN – number of vertex nodes in element.
- NEDG – number of element edges (sides).
- NDSD – number of displacement nodes along element side (excluding nodes at either end).
- NTSD – number of (displacement) nodes along side (edge).
- INDED – starting index to arrays NP1, NP2.

DIST 41 : loop on all element sides (loop to find side with pressure load).
DIST 42–43 : indexes to array NCONN.
DIST 44 : node at one end.
DIST 45 : skip if node numbers do not match (this is not the side which is loaded).
DIST 47–50 : side with pressure load not found in this element; print message (probable user error).
DIST 56–57 : store indexes to array NCONN for nodes at either end of side.
DIST 60–61 : do the same with side nodes (if any).
DIST 65–70 : set up local array with co-ordinates of nodes along side.
DIST 76 : loop on all integration points.
DIST 77 : local co-ordinate of integration point.
DIST 81 : calculate shape functions.
DIST 88 : calculate stress components at integration point.
DIST 89 : calculate derivatives $\partial x/\partial\xi$, $\partial y/\partial\xi$ at integration point.
DIST 91 : weighting factor.
DIST 95–96 : calculate radial distance of integration point (for axisymmetric problems).
DIST 98–99 : calculate x and y components of load at integration point.
DIST 101–103 : calculate nodal loads equivalent to applied pressure.
DIST 105 : end of loop on all integration points.
DIST 106–107 : print out calculated nodal loads.
DIST 112–119 : slot nodal loads in array RHS.

$$P_{xi} = \int_{S_e} N_i \left[\tau \cdot \frac{\partial x}{\partial \xi} - \sigma \cdot \frac{\partial y}{\partial \xi} \right] d\xi,$$

$$P_{yi} = \int_{S_e} N_i \left[\sigma \cdot \frac{\partial x}{\partial \xi} + \tau \cdot \frac{\partial y}{\partial \xi} \right] d\xi. \qquad (7.10)$$

Integration is taken along the loaded element edge S_e; ξ is the local co-ordinate along the element edge, and takes values between -1 and $+1$.

Routine **SFR1** calculates the shape functions N_i at sampling points. Numerical integration is used to carry out the above calculations. σ, τ are the normal and shear values of the applied stress distribution.

Routine SFR1

```
      SUBROUTINE SFR1(IW6,S,SHF,DERIV,NSD,LNE,LT)                       SFR1   1
C*****************************************************************     SFR1   2
C     SHAPE FUNCTIONS AND DERIVATIVES FOR ONE-DIMENSIONAL               *SFR1   3
C     GAUSSIAN INTEGRATION ALONG ELEMENT EDGE                           *SFR1   4
C*****************************************************************     SFR1   5
      DIMENSION SHF(NSD),DERIV(NSD)                                     SFR1   6
C-----------------------------------------------------------------      SFR1   7
C     INITIALISE                                                        SFR1   8
C-----------------------------------------------------------------      SFR1   9
      CALL ZEROR1(SHF,NSD)                                              SFR1  10
      CALL ZEROR1(DERIV,NSD)                                            SFR1  11
C                                                                       SFR1  12
      GO TO(80,21,31,41,51),NSD                                         SFR1  13
      WRITE(IW6,900)LNE,LT                                              SFR1  14
  900 FORMAT(1X,7HELEMENT,I5,2X,7HOF TYPE,I5,2X,                        SFR1  15
     1 22HUNKNOWN (ROUTINE SFR1))                                       SFR1  16
      STOP                                                              SFR1  17
C-----------------------------------------------------------------      SFR1  18
C     2 NODES ALONG EDGE                                                SFR1  19
C-----------------------------------------------------------------      SFR1  20
   21 CONTINUE                                                          SFR1  21
      WRITE(IW6,910)LT                                                  SFR1  22
  910 FORMAT(/1X,12HELEMENT TYPE,I5,2X,                                 SFR1  23
     1 30HNOT IMPLEMENTED (ROUTINE SFR1))                               SFR1  24
      GO TO 80                                                          SFR1  25
C-----------------------------------------------------------------      SFR1  26
C     3 NODES ALONG EDGE                                                SFR1  27
C-----------------------------------------------------------------      SFR1  28
   31 CONTINUE                                                          SFR1  29
      SHF(1)=0.5*S*(S-1.)                                               SFR1  30
      SHF(2)=(1.-S)*(1.+S)                                              SFR1  31
      SHF(3)=0.5*S*(S+1.)                                               SFR1  32
      DERIV(1)=S-0.5                                                    SFR1  33
      DERIV(2)=-2.*S                                                    SFR1  34
      DERIV(3)=S+0.5                                                    SFR1  3
      GO TO 80                                                          SFR1  3
C-----------------------------------------------------------------      SFR1  37
C     4 NODES ALONG EDGE                                                SFR1  38
C-----------------------------------------------------------------      SFR1  39
   41 CONTINUE                                                          SFR1  40
      WRITE(IW6,910)LT                                                  SFR1  41
      GO TO 80                                                          SFR1  42
C-----------------------------------------------------------------      SFR1  43
C     5 NODES ALONG EDGE                                                SFR1  44
C-----------------------------------------------------------------      SFR1  45
   51 S0=S                                                              SFR1  46
      S1=S+0.5                                                          SFR1  47
      S2=S-0.5                                                          SFR1  48
      S3=S+1.0                                                          SFR1  49
      S4=S-1.0                                                          SFR1  50
      C1=2./3.                                                          SFR1  51
      C2=8./3.                                                          SFR1  52
      C3=4.                                                             SFR1  53
      SHF(1)= C1*S0*S1*S2*S4                                            SFR1  54
      SHF(2)=-C2*S0*S2*S3*S4                                            SFR1  55
```

```
      SHF(3)= C3*S1*S2*S3*S4                                          SFR1  56
      SHF(4)=-C2*S0*S1*S3*S4                                          SFR1  57
      SHF(5)= C1*S0*S1*S2*S3                                          SFR1  58
      DERIV(1)= C1*(S2*S4*(S1+S0)+S0*S1*(S2+S4))                      SFR1  59
      DERIV(2)=-C2*(S2*S4*(S3+S0)+S0*S3*(S2+S4))                      SFR1  60
      DERIV(3)= C3*(S3*S4*(S1+S2)+S1*S2*(S3+S4))                      SFR1  61
      DERIV(4)=-C2*(S3*S4*(S1+S0)+S1*S0*(S3+S4))                      SFR1  62
      DERIV(5)= C1*(S2*S3*(S1+S0)+S1*S0*(S2+S3))                      SFR1  63
   80 CONTINUE                                                        SFR1  64
      RETURN                                                          SFR1  65
      END                                                             SFR1  66
```

SFR1 13 : branch off depending on no. of displacement nodes.
SFR1 21–25 : for element types with two nodes along side (no such element types in this version).
SFR1 30–32 : shape functions along element side for LST.
SFR1 33–35 : derivatives of shape functions.
SFR1 40 : shape functions and derivatives for element types with four nodes along element side (no such element types in this version).
SFR1 54–58 : shape functions (CuST).
SFR1 59–63: derivatives of shape functions (CuST).

7.9.2 Self-weight loads (body forces)

The self-weight loads given by

$$\int_V \mathbf{N}^{\mathrm{T}} \mathbf{w}\, \mathrm{d}\,(\mathrm{vol})$$

are calculated in routine **SELF**.

$$\begin{bmatrix} P_{xi} \\ P_{yi} \end{bmatrix} = \int_{V_e} N_i \gamma \begin{bmatrix} 0 \\ -1 \end{bmatrix} \mathrm{d}\,(\mathrm{vol}). \tag{7.11}$$

Gravity is assumed to act in the direction of the $-y$ axis.

Routine SELF

```
      SUBROUTINE SELF(IW6,I,NN,NEL,NTPE,NDN,NDIM,NAC,NPR,NMT,XYZ,     SELF   1
     1 ELCOD,DS,SHFN,F,NCONN,MAT,LL,PR,LT,INDX,DENS,MUS,KSTGE)        SELF   2
C*********************************************************************SELF   3
C     CALCULATES SELF WEIGHT LOADS                                     SELF   4
C*********************************************************************SELF   5
      REAL L,LL                                                       SELF   6
      DIMENSION NCONN(NTPE,NEL),MAT(NEL)                              SELF   7
      DIMENSION XYZ(NDIM,NN),ELCOD(NDIM,NDN),DS(NDIM,NDN),SHFN(NDN),  SELF   8
     1 F(NDIM,NDN),LL(NAC),PR(NPR,NMT),GCOM(3)                        SELF   9
      COMMON /ELINF / LINFO(50,15)                                    SELF  10
      COMMON /DATL  / L(4,100)                                        SELF  11
      COMMON /DATW  / W(100)                                          SELF  12
      COMMON /FLOW  / NPLAX                                           SELF  13
      COMMON /PARS  / PYI,ALAR,ASMVL,ZERO                             SELF  14
C                                                                     SELF  15
      TPI=2.*PYI                                                      SELF  16
      NGP=LINFO(11,LT)                                                SELF  17
      K=MAT(I)                                                        SELF  18
```



```
C-----------------------------------------------------------------------SELF  19
C     INITIALISE ARRAY F                                                SELF  20
C-----------------------------------------------------------------------SELF  21
      CALL ZEROR2(F,NDIM,NDN)                                           SELF  22
C                                                                       SELF  23
      IF(DENS.LE.ASMVL)GO TO 100                                        SELF  24
      GCOM(1)= ZERO                                                     SELF  25
      GCOM(2)=-DENS                                                     SELF  26
      GCOM(3)= ZERO                                                     SELF  27
C-----------------------------------------------------------------------SELF  28
C     SET UP LOCAL ARRAY FOR CO-ORDINATES                               SELF  29
C-----------------------------------------------------------------------SELF  30
      DO 10 KC=1,NDN                                                    SELF  31
      NDE=NCONN(KC,I)                                                   SELF  32
C                                                                       SELF  33
      DO 10 ID=1,NDIM                                                   SELF  34
   10 ELCOD(ID,KC)=XYZ(ID,NDE)                                          SELF  35
C-----------------------------------------------------------------------SELF  36
C     LOOP FOR NUMERICAL INTEGRATION                                    SELF  37
C-----------------------------------------------------------------------SELF  38
      DO 60 IP=1,NGP                                                    SELF  39
      IPA=IP+INDX                                                       SELF  40
C                                                                       SELF  41
      DO 35 IL=1,NAC                                                    SELF  42
   35 LL(IL)=L(IL,IPA)                                                  SELF  43
C-----------------------------------------------------------------------SELF  44
C     EVALUATE SHAPE FUNCTION FOR INTEGRATION POINT                     SELF  45
C-----------------------------------------------------------------------SELF  46
      CALL SHAPE(IW6,LL,NAC,DS,SHFN,NDIM,NDN,LT,2,MUS)                  SELF  47
      CALL DETJCB(IW6,DJACB,NDN,NDIM,ELCOD,DS,IP,MUS,KSTGE)             SELF  48
      DV=DJACB*W(IPA)                                                   SELF  49
      IF(NPLAX.EQ.0)GO TO 45                                            SELF  50
C                                                                       SELF  51
      RAD=ZERO                                                          SELF  52
C                                                                       SELF  53
      DO 40 IN=1,NDN                                                    SELF  54
   40 RAD=RAD+ELCOD(1,IN)*SHFN(IN)                                      SELF  55
      DV=DV*TPI*RAD                                                     SELF  56
C                                                                       SELF  57
   45 DO 50 IN=1,NDN                                                    SELF  58
      DO 50 ID=1,NDIM                                                   SELF  59
   50 F(ID,IN)=F(ID,IN)+GCOM(ID)*SHFN(IN)*DV                            SELF  60
   60 CONTINUE                                                          SELF  61
  100 CONTINUE                                                          SELF  62
      RETURN                                                            SELF  63
      END                                                               SELF  64
```

SELF 17 : number of integration points.
SELF 18 : material zone number.
SELF 22 : zero array F, self-weight loads of element.
SELF 24 : skip, if no self-weight loading.
SELF 25–27 : earth's gravity acts in the negative y direction.
SELF 31–35 : copy nodal co-ordinates into local array.
SELF 39 : loop on all integration points.
SELF 40 : index to arrays W and L (IPA is the starting index – 1).
SELF 42–43 : local/area co-ordinates of integration point.
SELF 47 : calculate shape functions and their derivatives w.r.t. local co-ordinates.
SELF 48 : calculate Jacobian of transformation.
SELF 49 : weighting factor.

SELF 52–55 : calculate radial distance of integration point (axisymmetric problems only).

SELF 58–60 : calculate nodal loads equivalent to self-weight.

$$F = \int_V \mathbf{N}^T \mathbf{w}\, d\,(\text{vol}).$$

SELF 61 : end of integration point loop.

Routine DETJCB

```
      SUBROUTINE DETJCB(IW6,DJACB,NDN,NDIM,ELCOD,DS,IP,MUS,KSTGE)       DETJ   1
C*****************************************************************************DETJ   2
C     CALCULATES DETERMINANT OF JACOBIAN MATRIX                          *DETJ   3
C*****************************************************************************DETJ   4
      DIMENSION ELCOD(NDIM,NDN),DS(NDIM,NDN),XJAC(3,3)                    DETJ   5
      COMMON /PARS  / PYI,ALAR,ASMVL,ZERO                                 DETJ   6
C-----------------------------------------------------------------------------DETJ   7
C     NXJ - SIZE OF ARRAY XJAC                                            DETJ   8
C-----------------------------------------------------------------------------DETJ   9
      NXJ=3                                                               DETJ  10
      CALL ZEROR2(XJAC,NXJ,NXJ)                                           DETJ  11
C                                                                         DETJ  12
      DO 10 ID=1,NDIM                                                     DETJ  13
      DO 10 JD=1,NDIM                                                     DETJ  14
      DO 10 IN=1,NDN                                                      DETJ  15
   10 XJAC(ID,JD)=XJAC(ID,JD)+DS(ID,IN)*ELCOD(JD,IN)                      DETJ  16
C                                                                         DETJ  17
      IF(NDIM.NE.2)GOTO 20                                                DETJ  18
      DJACB=XJAC(1,1)*XJAC(2,2)-XJAC(1,2)*XJAC(2,1)                       DETJ  19
      GOTO 50                                                             DETJ  20
C                                                                         DETJ  21
   20 DJACB=XJAC(1,1)*(XJAC(2,2)*XJAC(3,3)-XJAC(2,3)*XJAC(3,2))           DETJ  22
      DJACB=DJACB-XJAC(1,2)*(XJAC(2,1)*XJAC(3,3)-XJAC(2,3)*XJAC(3,1))     DETJ  23
      DJACB=DJACB+XJAC(1,3)*(XJAC(2,1)*XJAC(3,2)-XJAC(2,2)*XJAC(3,1))     DETJ  24
C                                                                         DETJ  25
   50 IF(DJACB.GT.ZERO)GO TO 60                                           DETJ  26
      WRITE(IW6,900)DJACB,MUS,IP                                          DETJ  27
  900 FORMAT(1X,10H JACOBIAN ,E16.5,3X,11HIS NEGATIVE,2X,                 DETJ  28
     1 7HELEMENT,I5,2X,10HINT. POINT,I5,2X,16H(ROUTINE DETJCB))           DETJ  29
C                                                                         DETJ  30
      WRITE(IW6,910)KSTGE                                                 DETJ  31
  910 FORMAT(/1X,36HCODE TO INDICATE STAGE OF ANALYSIS =,I5//             DETJ  32
     1 4X,4HCODE,20X,21HSTAGE OF THE ANALYSIS//                           DETJ  33
     1 6X,46H1 - CALLED BY INSITU/EQLOD/SELF CALCULATION OF,              DETJ  34
     2 1X,24HINSITU SELF WEIGHT LOADS/6X,13H2 - CALLED BY,                DETJ  35
     3 1X,44HANS/CHANGE/SELF LOADS DUE TO ELEMENT CHANGES/                DETJ  36
     4 6X,44H3 - CALLED BY ANS/SEL1/SELF INCREMENTAL SELF,                DETJ  37
     5 1X,12HWEIGHT LOADS/6X,25H4 - CALLED BY UPOUT/EQLOD,                DETJ  38
     6 45H/SELF SELF WEIGHT LOADS FOR EQUILIBRIUM CHECK)                  DETJ  39
      STOP                                                                DETJ  40
   60 RETURN                                                              DETJ  41
      END                                                                 DETJ  42
```

DETJ 11 : zero Jacobian matrix, **J**.

DETJ 13–16 : calculate components of Jacobian matrix.

DETJ 19 : calculate det |**J**| for 2-D.

DETJ 22–24 : calculate det |**J**| for 3-D.

DETJ 26 : check if det |**J**| is positive.

DETJ 27–29 : if not, print error message and stop.

Routines SELF, SHAPE and DETJCB are used in the simulation of construction by the addition of elements. These routines also perform the same calculations to determine loads equivalent to the self-weight of removed elements.

The loads equivalent to element stresses are given by

$$\int_V \mathbf{B}^T \boldsymbol{\sigma}\, d\,(\text{vol})$$

and were calculated in routine RDSTRS using routine EQLIB. The loads equivalent to the *in situ* stresses have been summed into PEQT(NDF).

7.9.3 Restrained nodes

PCOR, as mentioned, is calculated at all 'free' nodes. Routine **RESTRN** goes through the list of nodal fixities and inserts 1 against all d.o.f. which are either restrained or have a prescribed value in array IDFX(NDF). This enables routine EQLBM to identify those variables which are free from those with restraints or prescribed values.

Routine RESTRN

```
      SUBROUTINE RESTRN(NDF,NNOD1,NDIM,NW,IDFX)                         RSTR   1
C*****************************************************************************RSTR   2
C     ROUTINE TO IDENTIFY ALL DISPLACEMENT BOUNDARY CONDITIONS          RSTR   3
C     WHICH ARE SPECIFIED. ( SET IDFX = 1 FOR ALL DOF                   RSTR   4
C     WHICH ARE RESTRAINED.)                                            RSTR   5
C*****************************************************************************RSTR   6
      INTEGER TF                                                        RSTR   7
      DIMENSION NW(NNOD1),IDFX(NDF)                                     RSTR   8
      COMMON /FIX    / DXYT(4,200),MF(200),TF(4,200),NF                 RSTR   9
C-----------------------------------------------------------------------------RSTR  10
C     LOOP ON ALL NODES WITH ONE OR MORE FIXITIES                       RSTR  11
C-----------------------------------------------------------------------------RSTR  12
      DO 10 J=1,NDF                                                     RSTR  13
   10 IDFX(J)=0                                                         RSTR  14
C                                                                       RSTR  15
      IF(NF.EQ.0)RETURN                                                 RSTR  16
      DO 40 JN=1,NF                                                     RSTR  17
      NDE=MF(JN)                                                        RSTR  18
      NFS=NW(NDE)-1                                                     RSTR  19
C-----------------------------------------------------------------------------RSTR  20
C     BY-PASS IF NODE HAS ONLY PORE-PRESSURE DOF                        RSTR  21
C-----------------------------------------------------------------------------RSTR  22
      JP=NW(NDE+1)-NW(NDE)                                              RSTR  23
      IF(JP.EQ.1)GO TO 40                                               RSTR  24
C                                                                       RSTR  25
      DO 20 JF=1,NDIM                                                   RSTR  26
      NCDE=TF(JF,JN)                                                    RSTR  27
      IF(NCDE.EQ.0)GO TO 20                                             RSTR  28
      IDFX(NFS+JF)=1                                                    RSTR  29
   20 CONTINUE                                                          RSTR  30
   40 CONTINUE                                                          RSTR  31
      RETURN                                                            RSTR  32
      END                                                               RSTR  33
```

RSTR 13–14 : zero array which indicates variables which are restrained or have prescribed values.

RSTR 16 : skip if no fixities (unlikely).

RSTR 17 : loop on all fixities.
RSTR 18 : node with fixity.
RSTR 19 : starting index for g.v.n.
RSTR 23 : number of d.o.f. of node.
RSTR 24 : if only 1 d.o.f., skip (assumed to be the pore pressure variable).
RSTR 26 : loop on all displacement variables of node.
RSTR 27 : fixity code.
RSTR 28 : if d.o.f. is free, skip.
RSTR 29 : enter as fixed/prescribed.
RSTR 30 : end of loop on all displacement d.o.f. of node.
RSTR 31 : end of loop on all fixities.

7.9.4 Equilibrium check

Calculation of PCOR at each free node is done in routine **EQLBM** by consulting array IDFX(NDF) to check whether the entry is 0, indicating the d.o.f. is free.

Routine EQLBM

```
      SUBROUTINE EQLBM(IW6,NN,NNOD1,NDF,NDIM,NNZ,NDZ,NREL,              EQBM   1
     1 NW,NQ,IDFX,P,PT,PCOR,PEQT,IEQOP,ICOR,IRAC)                       EQBM   2
C*********************************************************************EQBM   3
C     CARRIES OUT AN EQUILIBRIUM CHECK                                  EQBM   4
C     CALCULATE AND PRINTOUT UNBALANCED NODAL LOADS                     EQBM   5
C*********************************************************************EQBM   6
      DIMENSION NREL(NNZ),NW(NNOD1),NQ(NN),IDFX(NDF)                    EQBM   7
      DIMENSION P(NDF),PT(NDF),PCOR(NDF),PEQT(NDF)                      EQBM   8
      DIMENSION PAR(6),RMAX(6),TER(3)                                   EQBM   9
      COMMON /PARS  / PYI,ALAR,ASMVL,ZERO                               EQBM  10
C-------------------------------------------------------------------EQBM  11
C     MP - ARRAY SIZE OF PAR, RMAX                                      EQBM  12
C-------------------------------------------------------------------EQBM  13
      MP=6                                                              EQBM  14
      NDIM1=NDIM+1                                                      EQBM  15
      NDIM2=2*NDIM                                                      EQBM  16
      IF(IRAC.EQ.1)CALL REACT(IW6,NN,NNOD1,NDF,NDIM,NNZ,                EQBM  17
     1   NREL,NW,NQ,IDFX,PEQT,PT)                                       EQBM  18
C-------------------------------------------------------------------EQBM  19
C     INCLUDE ALL PORE-PRESSURE TERMS IN THE LIST OF FIXED D.O.F.       EQBM  20
C     ALL EXCESS PORE PRESSURE D.O.F. ARE CONSIDERED TO BE FIXED        EQBM  21
C-------------------------------------------------------------------EQBM  22
      DO 2 NI=1,NN                                                      EQBM  23
      NQL=NQ(NI)                                                        EQBM  24
      IF(NQL.NE.1.AND.NQL.NE.NDIM1)GO TO 2                              EQBM  25
      ILC=NW(NI)+NQL-1                                                  EQBM  26
      IDFX(ILC)=1                                                       EQBM  27
    2 CONTINUE                                                          EQBM  28
C-------------------------------------------------------------------EQBM  29
C     CALCULATE OUT-OF-BALANCE LOADS FOR ALL FREE D.O.F.                EQBM  30
C-------------------------------------------------------------------EQBM  31
      DO 5 IK=1,NDF                                                     EQBM  32
      IF(IDFX(IK).EQ.1) GO TO 3                                         EQBM  33
      PCOR(IK)=PT(IK)-PEQT(IK)                                          EQBM  34
      GO TO 5                                                           EQBM  35
    3 PCOR(IK)=ZERO                                                     EQBM  36
    5 CONTINUE                                                          EQBM  37
C-------------------------------------------------------------------EQBM  38
C     OUTPUT EQUILIBRIUM, OUT-OF-BALANCE AND APPLIED NODAL LOADS        EQBM  39
```



```
C-------------------------------------------------------------------EQBM  40
      IF(IEQOP.EQ.0)GOTO 25                                             EQBM  41
      WRITE(IW6,900)                                                    EQBM  42
      WRITE(IW6,904)                                                    EQBM  43
C                                                                       EQBM  44
      DO 20 JR=1,NNZ                                                    EQBM  45
      IF(NREL(JR).EQ.0)GOTO 20                                          EQBM  46
      J=NREL(JR)                                                        EQBM  47
      NQL=NQ(J)                                                         EQBM  48
      IF(NQL.LE.1)GOTO 20                                               EQBM  49
      IF(IEQOP.EQ.1.AND.JR.GT.NDZ)GOTO 20                               EQBM  50
      N1=NW(J)                                                          EQBM  51
      N2=N1+NDIM-1                                                      EQBM  52
      WRITE(IW6,901)JR,(P(JJ),JJ=N1,N2),                                EQBM  53
     1 (PT(JJ),JJ=N1,N2),(PEQT(JJ),JJ=N1,N2),(PCOR(JJ),JJ=N1,N2)        EQBM  54
   20 CONTINUE                                                          EQBM  55
   25 CALL ZEROR1(RMAX,MP)                                              EQBM  56
C-------------------------------------------------------------------EQBM  57
C     CALCULATE MAXIMUM OF APPLIED AND OUT-OF-BALANCE                   EQBM  58
C     LOADS IN ALL DIRECTIONS                                           EQBM  59
C-------------------------------------------------------------------EQBM  60
      DO 50 IK=1,NN                                                     EQBM  61
      NQL=NQ(IK)                                                        EQBM  62
      IF(NQL.LE.1)GOTO 50                                               EQBM  63
      N1=NW(IK)                                                         EQBM  64
      N2=N1+NDIM-1                                                      EQBM  65
      IC=0                                                              EQBM  66
C                                                                       EQBM  67
      DO 35 KN=N1,N2                                                    EQBM  68
      IC=IC+1                                                           EQBM  69
      PAR(IC)=PT(KN)                                                    EQBM  70
   35 PAR(IC+NDIM)=PCOR(KN)                                             EQBM  71
C                                                                       EQBM  72
      DO 40 IC=1,NDIM2                                                  EQBM  73
      RV=PAR(IC)                                                        EQBM  74
      IF(ABS(RV).LT.ASMVL)GOTO 40                                       EQBM  75
      IF(ABS(RV).GT.RMAX(IC))RMAX(IC)=ABS(RV)                           EQBM  76
   40 CONTINUE                                                          EQBM  77
   50 CONTINUE                                                          EQBM  78
C-------------------------------------------------------------------EQBM  79
C     OUTPUT MAXIMUM OF (1) APPLIED LOADS (2) OUT-OF-BALANCE LOADS      EQBM  80
C     IN ALL DIRECTIONS                                                 EQBM  81
C-------------------------------------------------------------------EQBM  82
      WRITE(IW6,902)                                                    EQBM  83
C                                                                       EQBM  84
      IWARN=0                                                           EQBM  85
      PMAXT=RMAX(1)                                                     EQBM  86
      DO 55 ID=2,NDIM                                                   EQBM  87
   55 IF(RMAX(ID).GT.PMAXT)PMAXT=RMAX(ID)                               EQBM  88
      IF(PMAXT.LT.ASMVL) GOTO 132                                       EQBM  89
      DO 130 ID=1,NDIM                                                  EQBM  90
  130 TER(ID)=100.*RMAX(ID+NDIM)/PMAXT                                  EQBM  91
      GOTO 125                                                          EQBM  92
  132 IWARN=1                                                           EQBM  93
      DO 135 ID=1,NDIM                                                  EQBM  94
  135 TER(ID)=ZERO                                                      EQBM  95
C                                                                       EQBM  96
  125 WRITE(IW6,903)                                                    EQBM  97
      WRITE(IW6,905)                                                    EQBM  98
      WRITE(IW6,907)(RMAX(JQ),JQ=1,NDIM2),(TER(ID),ID=1,NDIM)           EQBM  99
      IF(IWARN.EQ.1)WRITE(IW6,910)                                      EQBM 100
C-------------------------------------------------------------------EQBM 101
C     ZERO PCOR IF NO CORRECTING LOADS ARE TO BE APPLIED IN NEXT INCR   EQBM 102
C-------------------------------------------------------------------EQBM 103
      IF(ICOR.NE.0)RETURN                                               EQBM 104
C                                                                       EQBM 105
```

```
      DO 140 IK=1,NDF                                                    EQBM 106
  140 PCOR(IK)=ZERO                                                      EQBM 107
      RETURN                                                             EQBM 108
  900 FORMAT(//67X,19HLOADS EQUIVALENT TO/9X,                            EQBM 109
     1 24HINCREMENTAL APPLIED LOAD,7X,18HTOTAL APPLIED LOAD,             EQBM 110
     1 10X,16HELEMENT STRESSES,11X,19HOUT-OF-BALANCE LOAD/               EQBM 111
     2 9X,24(1H-),7X,18(1H-),10X,16(1H-),11X,19(1H-))                    EQBM 112
  901 FORMAT(1X,I5,2X,8E14.4)                                            EQBM 113
  902 FORMAT(//1X,17HEQUILIBRIUM CHECK/1X,17(1H-))                       EQBM 114
  903 FORMAT(/8X,20HMAXIMUM APPLIED LOAD,12X,                            EQBM 115
     1 24HMAXM OUT-OF-BALANCE LOAD,10X,                                  EQBM 116
     2 31HPERCENTAGE ERROR IN EQUILIBRIUM/                               EQBM 117
     3 8X,20(1H-),12X,24(1H-),10X,31(1H-)/)                              EQBM 118
  904 FORMAT(/1X,5H NODE,8X,1HX,13X,1HY,13X,1HX,13X,1HY,13X,1HX,         EQBM 119
     1 13X,1HY,13X,1HX,13X,1HY//)                                        EQBM 120
  905 FORMAT(11X,1HX,15X,1HY,16X,1HX,15X,1HY,17X,1HX,15X,1HY/)           EQBM 121
  907 FORMAT(1X,4E16.5,2F16.5)                                           EQBM 122
  910 FORMAT(/40H WARNING **** NO APPLIED LOADING - CHECK,               EQBM 123
     1 1X,49HWHETHER ALL BOUNDARY CONDITIONS ARE DISPLACEMENTS,          EQBM 124
     2 2X,15H(ROUTINE EQLBM))                                            EQBM 125
      END                                                                EQBM 126
```

EQBM 15–16 : indexes to arrays PAR and RMAX.

EQBM 17–18 : calculate reactions-to-earth at nodes which are restrained (or have prescribed displacements). Identified by 1 in array IDFX against g.v.n.

EQBM 23 : loop on all nodes.

EQBM 24 : number of d.o.f. of node.

EQBM 25 : if node has only displacement d.o.f., by-pass.

EQBM 27 : enter 1 against pore pressure d.o.f. which are not included in the equilibrium check.

EQBM 32 : loop on all d.o.f.

EQBM 33 : by-pass either if restrained or if pore pressure d.o.f.

EQBM 34 : calculate out-of-balance load at d.o.f.

EQBM 36 : enter zero for d.o.f. if restrained or if pore pressure d.o.f.

EQBM 41 : skip if details of equilibrium check are not to be printed.

EQBM 45 : loop on all nodes in user sequence number.

EQBM 47 : program node no.

EQBM 48 : no. of d.o.f.

EQBM 49 : by-pass if node has only pore pressure d.o.f. (it is implicitly assumed that if a node has only 1 d.o.f. then that is pore pressure d.o.f.).

EQBM 50 : only print details at *vertex* nodes if IEQOP = 1.

EQBM 51–52 : g.v.n. of first and last displacement d.o.f. of node.

EQBM 53–54 : print out incremental, out-of-balance, equilibrium and total loads.

EQBM 61 : loop on all nodes.

EQBM 63 : skip if node has only pore pressure variable.

EQBM 64–65 : g.v.n. of first and last displacement d.o.f. of node.

EQBM 68 : loop on all d.o.f. of node.

EQBM 70–71 : copy total (PT) and out-of-balance (PCOR) loads at node.

EQBM 73–77 : update maximum values of PT and PCOR.

EQBM 87–88 : get maximum value of total load.

EQBM 89 : if it is negligible then no applied loading. (Could be an analysis where displacements are prescribed at boundary, i.e. displacement or strain controlled analysis.)

EQBM 90–91 : calculate percentage error in equilibrium (out-of-balance loads as a percentage of total load).

EQBM 94–95 : no applied load. Set it to zero (no way of calculating percentage error in load, as no loads have been applied).

EQBM 99–100 : print percentage error.

EQBM 106–107 : if errors in loads are not to be carried forward to nex increment, then zero them.

7.9.5 Reactions

At all nodes which are restrained or have a prescribed value,

$$P_{\text{cor}} = \int_S \mathbf{N}^{\mathrm{T}} \boldsymbol{\tau} \, \mathrm{d}\,(\text{area}) + \int_V \mathbf{N}^{\mathrm{T}} \mathbf{w} \, \mathrm{d}\,(\text{vol}) - \int_V \mathbf{B}^{\mathrm{T}} \boldsymbol{\sigma} \, \mathrm{d}\,(\text{vol}), \tag{7.12}$$

and is the reaction-to-earth. These are printed. Again IDFX(NDF) is made use of to indicate the d.o.f. which are fixed or have prescribed values.

Routine REACT

```
      SUBROUTINE REACT(IW6,NN,NNOD1,NDF,NDIM,NNZ,NREL,NW,NQ,IDFX,PEQT,   RECT   1
     1 PT)                                                               RECT   2
C*********************************************************************RECT   3
C     CALCULATES REACTION TO EARTH AT RESTRAINED NODES                  RECT   4
C*********************************************************************RECT   5
      DIMENSION PEQT(NDF),PT(NDF),NW(NNOD1),NREL(NNZ),NQ(NN),IDFX(NDF)  RECT   6
      DIMENSION R(500),NDENO(500),NDIR(500)                             RECT   7
C-------------------------------------------------------------------RECT   8
C     NCT - SIZE OF ARRAYS R, NDENO AND NDIR                            RECT
C-------------------------------------------------------------------RECT
      NCT=500                                                           RECT   11
C-------------------------------------------------------------------RECT   12
C     ICT - COUNTER OF TOTAL NO. OF REACTIONS                           RECT   13
C-------------------------------------------------------------------RECT   14
      ICT=0                                                             RECT   15
C                                                                       RECT   16
      DO 25 JR=1,NNZ                                                    RECT   17
      IF(NREL(JR).EQ.0)GOTO 25                                          RECT   18
      J=NREL(JR)                                                        RECT   19
      NQL=NQ(J)                                                         RECT   20
C-------------------------------------------------------------------RECT   21
C     SKIP IF NODE HAS PORE PRESSURE D.O.F. ONLY                        RECT   22
C-------------------------------------------------------------------RECT   23
      IF(NQL.LE.1)GOTO 25                                               RECT   24
      N1=NW(J)                                                          RECT   25
      N2=N1+NDIM-1                                                      RECT   26
      IDF=0                                                             RECT   27
C                                                                       RECT   28
      DO 20 KN=N1,N2                                                    RECT   29
      IDF=IDF+1                                                         RECT   30
```

```
      IF(IDFX(KN).NE.1)GOTO 20                                          RECT  31
      ICT=ICT+1                                                         RECT  32
      IF(ICT.GT.NCT)GOTO 30                                             RECT  33
      R(ICT)=-(PEQT(KN)-PT(KN))                                         RECT  34
      NDENO(ICT)=JR                                                     RECT  35
      NDIR(ICT)=IDF                                                     RECT  36
   20 CONTINUE                                                          RECT  37
   25 CONTINUE                                                          RECT  38
C                                                                       RECT  39
      WRITE(IW6,901)                                                    RECT  40
      WRITE(IW6,903)(NDENO(JCT),NDIR(JCT),R(JCT),JCT=1,ICT)             RECT  41
      RETURN                                                            RECT  42
   30 WRITE(IW6,906)                                                    RECT  43
      STOP                                                              RECT  44
  901 FORMAT(//1X,18H LIST OF REACTIONS/2X,17(1H-)/                     RECT  45
     1 2X,3(4HNODE,4X,9HDIRECTION,7X,8HREACTION,11X)/)                  RECT  46
  903 FORMAT(3(1X,I5,5X,I4,5X,E14.4,10X))                               RECT  47
  906 FORMAT(/1X,35HINCREASE ARRAY SIZE OF R,NDENO,NDIR,                RECT  48
     1 1X,16HIN ROUTINE REACT)                                          RECT  49
      END                                                               RECT  50
```

RECT 15 : counter of total no. of reactions (each variable is dealt with separately).
RECT 17 : loop on all nodes in **user** sequence number.
RECT 18 : skip if user has not used this node no.
RECT 19 : program node number.
RECT 20 : number of d.o.f. of node.
RECT 24 : by-pass if node has only 1 d.o.f. (assumed to be pore pressure variable).
RECT 25–26 : g.v.n. of first and last displacement d.o.f. of node.
RECT 29 : loop on all displacement d.o.f. (variables).
RECT 30 : displacement variable no. of node (i.e. 1 or 2).
RECT 31 : skip if not restrained or prescribed.
RECT 32 : increment count of reactions by 1.
RECT 33 : skip if array size is exceeded.
RECT 34 : calculate reactions-to-earth.
RECT 35 : enter user node number.
RECT 36 : enter direction $(1-x; 2-y)$.
RECT 37 : end of loop on all displacement d.o.f. of node.
RECT 38 : end of loop on all nodes.
RECT 40–41 : print out list of reactions.
RECT 43–44 : print message to increase array size, and stop.

7.9.6 Initialising arrays

A set of routines to zero real and integer arrays of one, two and three dimensions is used throughout the program. Whenever an array needs to be zeroed, a subroutine call is made to the appropriate routine.

Routine ZEROSB

```
      SUBROUTINE ZEROI1(N,LN)                                           ZERO   1
C*****************************************************************************ZERO   2
C     ROUTINE TO INITIALISE A 1-DIMENSIONAL INTEGER ARRAY                 ZERO   3
C*****************************************************************************ZERO   4
      DIMENSION N(LN)                                                   ZERO   5
C                                                                       ZERO   6
      DO 10 I=1,LN                                                      ZERO   7
   10 N(I)=0                                                            ZERO   8
      RETURN                                                            ZERO   9
      END                                                               ZERO  10
      SUBROUTINE ZEROI2(N,L1,L2)                                        ZERO  11
C*****************************************************************************ZERO  12
C     ROUTINE TO INITIALISE A 2-DIMENSIONAL INTEGER ARRAY               ZERO  13
C*****************************************************************************ZERO  14
      DIMENSION N(L1,L2)                                                ZERO  15
C                                                                       ZERO  16
      DO 10 J=1,L2                                                      ZERO  17
      DO 10 I=1,L1                                                      ZERO  18
   10 N(I,J)=0                                                          ZERO  19
      RETURN                                                            ZERO  20
      END                                                               ZERO  21
      SUBROUTINE ZEROR1(V,LV)                                           ZERO  22
C*****************************************************************************ZERO  23
C     ROUTINE TO INITIALISE A 1-DIMENSIONAL REAL ARRAY                  ZERO  24
C*****************************************************************************ZERO  25
      DIMENSION V(LV)                                                   ZERO  26
C                                                                       ZERO  27
      DO 10 I=1,LV                                                      ZERO  28
   10 V(I)=0.                                                           ZERO  29
      RETURN                                                            ZERO  30
      END                                                               ZERO  31
      SUBROUTINE ZEROR2(V,L1,L2)                                        ZERO  32
C*****************************************************************************ZERO  33
C     ROUTINE TO INITIALISE A 2-DIMENSIONAL REAL ARRAY                  ZERO  34
C*****************************************************************************ZERO  35
      DIMENSION V(L1,L2)                                                ZERO  36
C                                                                       ZERO  37
      DO 10 J=1,L2                                                      ZERO  38
      DO 10 I=1,L1                                                      ZERO  39
   10 V(I,J)=0.                                                         ZERO  40
      RETURN                                                            ZERO  41
      END                                                               ZERO  42
      SUBROUTINE ZEROR3(V,L1,L2,L3)                                     ZERO  43
C*****************************************************************************ZERO  44
C     ROUTINE TO INITIALISE A 3-DIMENSIONAL REAL ARRAY                  ZERO  45
C*****************************************************************************ZERO  46
      DIMENSION V(L1,L2,L3)                                             ZERO  47
C                                                                       ZERO  48
      DO 10 K=1,L3                                                      ZERO  49
      DO 10 J=1,L2                                                      ZERO  50
      DO 10 I=1,L1                                                      ZERO  51
   10 V(I,J,K)=0.                                                       ZERO  52
      RETURN                                                            ZERO  53
      END                                                               ZERO  54
```

ZERO 1–54 : zero array in separate routines as follows:

Dimensions	Array type	Routine name
1	INTEGER	ZEROI1
2	INTEGER	ZEROI2
1	REAL	ZEROR1
2	REAL	ZEROR2
3	REAL	ZEROR3

8

Analysis

8.1 INTRODUCTION

Having set the *in situ* stresses, the analysis proper can begin. An equilibrium check has also been carried out to make sure that external loads specified by the user are equivalent to the element *in situ* stresses. It should be remembered that these (*in situ*) loads are different from the loading applied during the course of the analysis.

In some analyses the simple option of no initial stresses may have been selected. However, in most geotechnical problems the *in situ* stresses play an important role. CRISP stores the current stress state, and this governs behavio under the subsequent loading. This chapter deals with the response of the soil t a given loading.

The loads are divided into steps, called increment blocks. These increment blocks in turn are divided into increments. The use of increment blocks is for convenience. The analysis can be divided into the following steps:

(i) calculation of incremental loads;
(ii) application of the boundary conditions;
(iii) assembly of the stiffness matrix;
(iv) solution of the equations;
(v) calculation of strains and stresses;
(vi) output of results.

Section 8.2 explains the use of increment blocks. Section 8.3 presents a brief

explanation of the subroutines listed in this chapter. Section 8.4 deals with the calculation of incremental loads. Section 8.5 presents the details of the load increment loop. Sections 8.6 to 8.8 deal with the calculation of the element stiffness matrix and the global stiffness matrix. The frontal solution is dealt with in separate sections, 8.9 to 8.12. Section 8.13 considers the calculation of incremental strains and stresses, and the printing out of the various parameters. Section 8.14 lists the subroutine which deals with stopping and restarting an analysis.

The previous chapter dealt with the setting up of the *in situ* stresses and satisfying the equilibrium conditions at that stage. Those readers interested in an analysis with zero *in situ* stresses may have skipped the previous chapter. However, a number of routines are common to the *in situ* part and the analysis part of the program. Where applicable we refer the reader back to the explanations in the previous chapter.

8.2 INCREMENT BLOCKS

The entire loading is divided into a number of **increments**. The increments can be grouped into a number of **increment blocks**. As mentioned in Chapter 4, this facility is provided for two reasons.

(i) If the loads for each analysis increment had to be specified separately there would be a very large amount of data input needed for most problems. Much of this information would be repeated many times (e.g. which element sides were being loaded).

(ii) When performing an excavation (or construction) analysis the program calculates the implied loads due to the removal (or addition) of the elements specified by the user. These implied loads will often be too large to be applied in a single increment when the material behaviour is non-linear. The use of an increment block spreads these implied loads over several increments. (Note that this procedure introduces an extra approximation in the modelling of excavations: the stiffness of an element is removed entirely in the first increment of a block whereas the loads are spread over all increments in the block.)

8.3 CONTROL ROUTINE

The master control routine is **ANS**. This loops around all increment blocks in the analysis. This is the outer loop. The inner loop is on all increments within the increment block. Each increment block contains at least one increment.

Routine ANS consists of a series of subroutine calls to various routines, delegating tasks to them (Fig. 8.1). A brief explanation of each subroutine discussed in this chapter is given below.

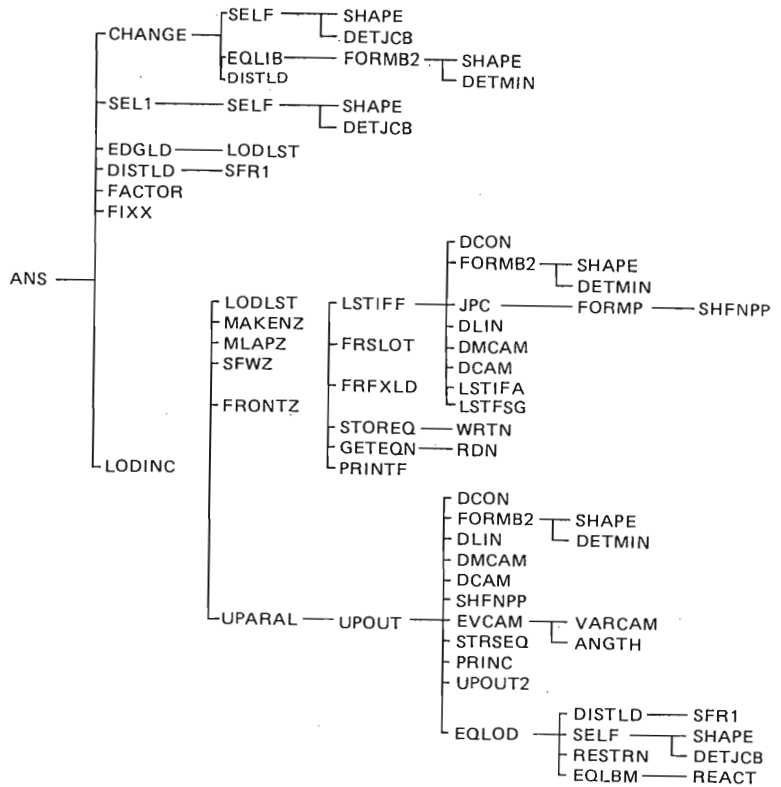


Fig. 8.1 – Subroutine hierarchy for analysis part of program

ANS – main control routine for analysis. Reads control parameters for increment block. Delegates tasks to routines CHANGE, SEL1 and LODINC.

CHANGE – calculates implied loads due to removal and addition of elements.

SEL1 – calculates nodal loads for self-weight loads.

FACTOR – reads load ratios, output options and time steps for each increment within an increment block.

LODINC – control routine delegates calculation of stiffness matrices and solution of equations to FRONTZ and printing out the results to UPOUT (via UPARAL).

LSTIFF – calculates element stiffness matrix.

JPC – for consolidation analysis, calculates components of stiffness matrix.

FORMP – calculates **E** matrix (pore pressure gradients), i.e. Cartesian derivatives of pore pressure shape functions.

SHFNPP – calculates pore pressure shape functions.

LSTIFA – calculates $\int_V \mathbf{B}^T\mathbf{D}\mathbf{B}\,\mathrm{d}\,(\mathrm{vol})$.

LSTFSG – rearranges rows/columns of element stiffness matrix and forms a one-dimensional (upper triangular matrix stored columnwise) matrix acceptable to FRONTZ.

FRONTZ – calls LSTIFF to calculate the element stiffness matrix, and solves the assembled equations using the frontal method.

FRSLOT – slots upper triangular element stiffness matrix in appropriate places in the front.

FRFXLD – deals with prescribed displacements and applied loads for nodes being elminated. Also prints them out.

PRINTF – debugging routine to print out element stiffness matrix and stiffness terms and load terms in the front.

STOREQ – stores contents of the buffer of eliminated coefficients in backing store when the buffer fills up during frontal solution.

WRTN – routine used by STOREQ to write to backing store.

GETEQN – performs the reverse task to STOREQ. Gets back a bufferful of eliminated coefficients from backing store when the buffer becomes empty during back-substitution in the frontal solution.

RDN – routine used by GETEQN to read from backing store.

UPARAL – sets up temporary arrays for storing output tables to be printed in UPOUT2.

UPOUT – output routine. Increments displacements and calculates stress increments and prints out the results. Writes results to magnetic tape or disk, but in the latter case only if it is the last increment of the analysis (used in stopping and restarting an analysis).

EVCAM – calculates extra stress parameters for Cam-clays.

VARCAM – assigns codes to indicate stress state for Cam-clays.

ANGTH – calculates angle θ in π plane.

PRINC – calculates principal stresses in xy plane.

CAMCDE – prints out explanations of codes assigned to identify stress states for Cam-clay models only.

UPOUT2 – prints out additional stress parameters calculated for Cam-clays.

STRSEQ – calculates forces equilibriating element stresses.

RESTRT – stop–restart facility (see section 9.2).

The main task of the routine ANS is to form a single load vector PIB(NDF) from the user-specified pressure loads and self-weight loads and to translate restraints/prescribed displacements (including pore pressures) into a list of nodal fixities. Within the increment loop, both of these are accessed. Implied loadings due to removal of elements (excavation) and addition of elements (construction) are also calculated and assembled into PIB(NDF).

Routine ANS

```
      SUBROUTINE ANS(NN,NEL,NDF,NNOD1,NTPE,NIP,NVRS,NVRN,NDIM,MUMAX,     ANS    1
     1 NDZ,IFRZ,NNZ,NDMX,NPMX,NS,NB,NL,NPR,NMT,NPT,NSP,NPL,             ANS    2
     2 MDFE,KES,NVPN,INXL,MXEN,MXLD,MXFXT,LV,NVTX,ND,                   ANS    3
     3 XYZ,DI,DA,VARINT,P,PT,PIB,REAC,PCOR,PEQT,XYFT,XYFIB,             ANS    4
     4 STR,PEXIB,PEXI,PCONI,D,ELCOD,DS,SHFN,CARTD,B,DB,FT,SS,ES,        ANS    5
     1 ELCODP,E,PE,RN,AA,ETE,RLT,                                       ANS    6
     2 NCONN,MAT,LTYP,MRELVV,MREL,NRELVV,NREL,NW,NQ,                    ANS    7
     3 JEL,IDFX,NDEST,NP1,NP2,IFR,NDL,NWL,NMOD,                         ANS    8
     4 CIP,LL,V,FXYZ,PR,PDISLD,PRES,NTY,A,MFZ,NOIB,                     ANS    9
     5 TTIME,TGRAV,IUPD,ICOR,IBC,IDCHK,INCT)                            ANS   10
C*************************************************************************ANS   11
C     MAIN CONTROLLING ROUTINE                                           ANS    1
C*************************************************************************ANS   13
      REAL L,LL                                                         ANS   14
      INTEGER TF                                                        ANS   15
C--------USE THE FOLLOWING STATEMENT AFTER CONVERTING PROGRAM TO DOUBLE  ANS   16
C--------PRECISION.  ARRAY A ALWAYS USES ONE NUMERIC STORAGE LOCATION    ANS   17
CC    REAL A                                                            ANS   18
      DIMENSION XYZ(NDIM,NN),DI(NDF),DA(NDF),VARINT(NVRS,NIP,NEL),      ANS   19
     1 P(NDF),PT(NDF),PIB(NDF),REAC(NDF),PCOR(NDF),PEQT(NDF),XYFT(NDF), ANS   20
     2 XYFIB(NDF),STR(NVRN,NIP,NEL),PEXIB(NDF),PEXI(NDF),PCONI(NDF)     ANS   21
      DIMENSION D(NS,NS),ELCOD(NDIM,NDMX),DS(NDIM,NDMX),SHFN(NDMX),     ANS   22
     1 CARTD(NDIM,NDMX),B(NS,NB),DB(NS,NB),FT(NDIM,NDMX),               ANS   23
     2 SS(NB,NB),ES(KES)                                               ANS   24
      DIMENSION ELCODP(NDIM,NPMX),E(NDIM,NPMX),PE(NDIM,NPMX),           ANS   25
     1 RN(NB),AA(NPMX),ETE(NPMX,NPMX),RLT(NB,NPMX)                      ANS   26
      DIMENSION NCONN(NTPE,NEL),MAT(NEL),LTYP(NEL),MRELVV(NEL),         ANS   27
     1 MREL(MUMAX),NRELVV(NN),NREL(NNZ),NW(NNOD1),NQ(NN),JEL(NEL),      ANS   28
     2 IDFX(NDF),NDEST(NN),NP1(NPL),NP2(NPL)                           ANS   29
      DIMENSION IFR(IFRZ),NDL(MDFE),NWL(NPMX),NMOD(NIP,NEL)             ANS   30
      DIMENSION CIP(NDIM),LL(NL),V(LV),FXYZ(NDIM),PR(NPR,NMT),          ANS   31
     1 PDISLD(NDIM,NPT),PRES(NDIM,NPT),NTY(NMT),A(MFZ)                  ANS   32
      DIMENSION RINCC(50),DTM(50),IOPT(50)                              ANS   33
      COMMON /FLOW  / NPLAX                                             ANS   34
      COMMON /DATL  / L(4,100)                                          ANS   35
      COMMON /DATW  / W(100)                                            ANS   36
      COMMON /ELINF / LINFO(50,15)                                      ANS   37
      COMMON /FIX   / DXYT(4,200),MF(200),TF(4,200),NF                  ANS   38
      COMMON /PRSLD / PRESLD(10,100),LEDG(100),NDE1(100),NDE2(100),NLED ANS   39
      COMMON /PRLDI / PRSLDI(10,100),LEDI(100),NDI1(100),NDI2(100),ILOD ANS   40
      COMMON /DEVICE/ IR1,IR4,IR5,IW2,IW4,IW6,IW7,IW8,IW9               ANS   41
      COMMON /PARS  / PYI,ALAR,ASMVL,ZERO                               ANS   4
C-------------------------------------------------------------------------ANS
C     MAXIMUM NUMBER OF INCREMENTS IN A INCREMENT BLOCK                  ANS   4
C-------------------------------------------------------------------------ANS   45
      INCZ=50                                                           ANS   46
      NDIM1=NDIM+1                                                      ANS   47
      IF(IDCHK.EQ.0)GOTO 10                                             ANS   48
      WRITE(IW6,907)                                                    ANS   49
      STOP                                                              ANS   50
C-------------------------------------------------------------------------ANS   51
C     START  OF  INCREMENT  LOOP                                         ANS   52
C-------------------------------------------------------------------------ANS   53
   10 DO 250 J=1,NOIB                                                   ANS   54
C                                                                       ANS   55
      WRITE(IW6,908) J                                                  ANS   56
C-------------------------------------------------------------------------ANS   57
C     INITIALISE  LOAD  VECTOR                                           ANS   58
C-------------------------------------------------------------------------ANS   59
      CALL ZEROR1(XYFIB,NDF)                                            ANS   60
      CALL ZEROR1(PIB,NDF)                                              ANS   61
      CALL ZEROR1(PEXIB,NDF)                                            ANS   62
C                                                                       ANS   63
```

```
      DO 20 JJ=1,MXEN                                                   ANS   64
      DO 20 II=1,MXLD                                                   ANS   65
   20 PRSLDI(II,JJ)=ZERO                                                ANS   66
C                                                                       ANS   67
      ILOD=0                                                            ANS   68
      CALL ZEROI1(JEL,NEL)                                              ANS   69
      CALL ZEROI1(IOPT,INCZ)                                            ANS   70
      CALL ZEROR1(DTM,INCZ)                                             ANS   71
      CALL ZEROR1(RINCC,INCZ)                                           ANS   72
      FRACT=ZERO                                                        ANS   73
C-----------------------------------------------------------------------ANS   74
C     READ  INCREMENT  CONTROL  OPTIONS                                 ANS   75
C-----------------------------------------------------------------------ANS   76
      READ(IR5,*)IBNO,INC1,INC2,ICHEL,NLOD,ILDF,NFX,IOUTS,              ANS   77
     1 IOCD,DTIME,ITMF,DGRAV                                            ANS   78
      WRITE(IW6,912)IBNO,INC1,INC2,ICHEL,NLOD,ILDF,NFX,IOUTS,           ANS   79
     2 IOCD,DTIME,ITMF,DGRAV                                            ANS   80
      NOINC=INC2+1-INC1                                                 ANS   81
      IF(NOINC.LE.INCZ)GOTO 70                                          ANS   82
      WRITE(IW6,950)NOINC                                               ANS   83
      STOP                                                              ANS   84
   70 IF(IBNO.EQ.J) GO TO 72                                            ANS   85
      WRITE(IW6,913) IBNO,J                                             ANS   86
      STOP                                                              ANS   87
   72 IF(ICHEL.EQ.0) GO TO 76                                           ANS   88
C-----------------------------------------------------------------------ANS   89
C     ALTER  GEOMETRY  AS  SPECIFIED                                    ANS   90
C-----------------------------------------------------------------------ANS   91
      WRITE(IW6,914)                                                    ANS   92
      READ(IR5,*)(JEL(JJ),JJ=1,ICHEL)                                   ANS   93
      WRITE(IW6,920) (JEL(JJ),JJ=1,ICHEL)                               ANS   94
C                                                                       ANS   95
      CALL CHANGE(IW6,1,ICHEL,NN,NNOD1,NTPE,NIP,NEL,MUMAX,NNZ,NDF,NDIM, ANS   96
     1   NVRS,NDMX,NL,NB,NS,NPR,NMT,NPT,NSP,NPL,XYZ,VARINT,PIB,PEXIB,    ANS   97
     2   ELCOD,DS,SHFN,CARTD,B,FT,NCONN,MAT,LTYP,MREL,NREL,               ANS   98
     3   NW,JEL,NP1,NP2,MXEN,LL,PR,TGRAV)                                ANS   99
C-----------------------------------------------------------------------ANS  100
C     CALCULATE  BODY  FORCE  LOAD  VECTOR                              ANS  101
C     FOR SELF-WEIGHT LOADING AND GRAVITY LOADING                       ANS  102
C-----------------------------------------------------------------------ANS  103
   76 CALL SEL1(IW6,ICHEL,NN,NNOD1,NTPE,NIP,NEL,NDF,MUMAX,NL,NDIM,      ANS  104
     1   NDMX,NPR,NMT,XYZ,PIB,ELCOD,DS,SHFN,FT,NCONN,MAT,                ANS  105
     2   LTYP,MRELVV,MREL,NW,JEL,LL,PR,NTY,DGRAV)                        ANS  106
C-----------------------------------------------------------------------ANS  107
C     READ LOAD FACTORS, TIME FACTORS AND OUTPUT OPTIONS                ANS  108
C-----------------------------------------------------------------------ANS  109
      CALL FACTOR(IR5,IW6,NOINC,ILDF,IOCD,ITMF,IOUTS,                   ANS  110
     1   RINCC,DTM,IOPT,DTIME)                                          ANS  111
      IF(NLOD.EQ.0)GO TO 95                                             ANS  112
      IF(NLOD.GT.0)GO TO 82                                             ANS  113
C-----------------------------------------------------------------------ANS  114
C     PRESSURE LOADING ALONG ELEMENT EDGE                               ANS  115
C-----------------------------------------------------------------------ANS  116
      WRITE(IW6,1000)                                                   ANS  117
      NLDS=IABS(NLOD)                                                   ANS  118
      IF(NDIM.EQ.2)GOTO 78                                              ANS  119
      WRITE(IW6,955)                                                    ANS  120
  955 FORMAT(/1X,34HNO OPTION TO CALCULATE NODAL LOADS,1X,              ANS  121
     1 50HFROM PRESSURE LOADING IN 3-D PROBLEM (ROUTINE ANS))           ANS  122
      STOP                                                              ANS  123
C                                                                       ANS  124
   78 DO 80 KLOD=1,NLDS                                                 ANS  125
      READ(IR5,*)LNE,ND1,ND2,((PDISLD(ID,IV),ID=1,NDIM),IV=1,NPT)       ANS  126
      WRITE(IW6,1002)LNE,ND1,ND2,((PDISLD(ID,IV),ID=1,NDIM),IV=1,NPT)   ANS  127
C                                                                       ANS  128
      DO 100 IV=1,NPT                                                   ANS  129
```



```
      DO 100 ID=1,NDIM                                                  ANS  130
      IDR=NDIM+1-ID                                                     ANS  131
  100 PRES(ID,IV)=PDISLD(IDR,IV)                                        ANS  132
C                                                                       ANS  133
      DO 110 IV=1,NPT                                                   ANS  134
      DO 110 ID=1,NDIM                                                  ANS  135
  110 PDISLD(ID,IV)=PRES(ID,IV)                                         ANS  136
C                                                                       ANS  137
      CALL EDGLD(IW6,NEL,NDIM,NTPE,NNZ,MUMAX,NPL,NCONN,LTYP,MREL,NREL,  ANS  138
     1   LNE,ND1,ND2,NP1,NP2,PDISLD,PRES,KLOD,NPT,0,MXLD)                ANS  139
C                                                                       ANS  140
      CALL DISTLD(IW6,NN,NEL,NDF,NNOD1,NTPE,NDIM,MUMAX,NNZ,             ANS  141
     1   NPL,XYZ,PIB,NCONN,LTYP,MREL,NREL,NW,                           ANS  142
     2   NP1,NP2,PRES,LNE,ND1,ND2,NPT,NSP,1,1,1.)                       ANS  143
   80 CONTINUE                                                          ANS  144
      GO TO 95                                                          ANS  145
C-----------------------------------------------------------------------ANS  146
C     READ INCREMENTAL POINT LOADS                                      ANS  147
C-----------------------------------------------------------------------ANS  148
   82 WRITE(IW6,916)                                                    ANS  149
C                                                                       ANS  150
      DO 90 JJ=1,NLOD                                                   ANS  151
      READ(IR5,*)KK,(FXYZ(ID),ID=1,NDIM)                                ANS  152
      WRITE(IW6,940)KK,(FXYZ(ID),ID=1,NDIM)                             ANS  153
C----------NO PROVISION FOR PORE PRESSURE TERMS IN 'APPLIED' NODAL LOADSANS  154
      FTT=ZERO                                                          ANS  155
      KJ=NREL(KK)                                                       ANS  156
      N1=NW(KJ)-1                                                       ANS  157
      IDF=NW(KJ+1)-NW(KJ)                                               ANS  158
      IF(IDF.EQ.1)GO TO 84                                              ANS  159
C                                                                       ANS  160
      DO 83 ID=1,NDIM                                                   ANS  161
   83 XYFIB(N1+ID)=FXYZ(ID)                                             ANS  162
      IF(IDF.EQ.NDIM1)XYFIB(N1+NDIM1)=FTT                               ANS  163
      GO TO 90                                                          ANS  164
   84 XYFIB(N1+1)=FTT                                                   ANS  165
   90 CONTINUE                                                          ANS  166
C                                                                       ANS  167
   95 IF(NFX.EQ.0) GO TO 137                                            ANS  168
C-----------------------------------------------------------------------ANS  169
C     READ  CHANGE  TO  NODAL  FIXITIES                                 ANS  170
C-----------------------------------------------------------------------ANS  171
      WRITE(IW6,931)                                                    ANS  172
C                                                                       ANS  173
      CALL FIXX(IR5,IW6,NEL,NTPE,NDIM,NPL,LV,MUMAX,NNZ,NCONN,LTYP,      ANS  174
     1   MREL,NREL,NP1,NP2,V,NFX)                                       ANS  175
  137 CONTINUE                                                          ANS  176
C-----------------------------------------------------------------------ANS  177
C     START OF INCREMENT LOOP                                           ANS  178
C-----------------------------------------------------------------------ANS  179
      DO 200 JS=INC1,INC2                                               ANS  180
      INCT=INCT+1                                                       ANS  181
      IF(JS.EQ.INCT)GO TO 138                                           ANS  182
      WRITE(IW6,933)JS,INCT                                             ANS  183
      STOP                                                              ANS  184
  138 JC=JS+1-INC1                                                      ANS  185
      FRACLD=RINCC(JC)                                                  ANS  186
      FRACT=FRACT+FRACLD                                                ANS  187
      DTIMEI=DTM(JC)                                                    ANS  188
      TTIME=TTIME+DTIMEI                                                ANS  189
      DGRAVI=FRACLD*DGRAV                                               ANS  190
      TGRAV=TGRAV+DGRAVI                                                ANS  191
      IOUT=IOPT(JC)                                                     ANS  192
C-----------------------------------------------------------------------ANS  193
C     FLAG TO INDICATE THE VERY LAST INCREMENT IN CURRENT RUN.          ANS  194
C     TO ALLOW RESULTS FROM THIS INCREMENT TO BE WRITTEN TO DISK FILE   ANS  195
```

```
C     IF STOP/RESTART OPTION ISR = 1 IS BEING USED.                    ANS  196
C-----------------------------------------------------------------------ANS  197
                                                                       ANS  198
      IWL=0                                                            ANS  199
      IF(J.EQ.NOIB.AND.JS.EQ.INC2)IWL=1                                ANS  200
C                                                                      ANS  201
      CALL LODINC(NN,NEL,NDF,NNOD1,NTPE,NIP,NVRS,                      ANS  202
     1   NVRN,NDIM,MUMAX,NDZ,IFRZ,NNZ,NDMX,NPMX,                       ANS  203
     2   NS,NB,NL,NPR,NMT,NPT,NSP,NPL,MDFE,KES,NVPN,                   ANS  204
     3   INXL,MXEN,MXLD,LV,NVTX,ND,                                    ANS  205
     4   XYZ,DI,DA,VARINT,P,PT,PIB,REAC,PCOR,PEQT,XYFT,XYFIB,          ANS  206
     5   STR,PEXIB,PEXI,PCONI,D,ELCOD,DS,SHFN,CARTD,B,DB,FT,SS,        ANS  207
     6   ES,ELCODP,E,PE,RN,AA,ETE,RLT,                                 ANS  208
     7   NCONN,MAT,LTYP,MRELVV,MREL,NRELVV,NREL,NW,NQ,                 ANS  209
     8   JEL,IDFX,NDEST,NP1,NP2,IFR,NDL,NWL,NMOD,                      ANS  210
     9   CIP,LL,V,FXYZ,PR,PDISLD,PRES,NTY,A,MFZ,                       ANS  211
     1   DTIMEI,TTIME,DGRAVI,TGRAV,IOUT,JS,J,FRACLD,                   ANS  212
     2   FRACT,ICOR,IUPD,IBC,NLOD,NLDS,IWL)                            ANS  213
C                                                                      ANS  214
  200 CONTINUE                                                         ANS  215
C-----------------------------------------------------------------------ANS  216
C     ZERO ALL NON-ZERO PRESCRIBED VALUES                              ANS  217
C-----------------------------------------------------------------------ANS  218
      IF(NF.EQ.0)GOTO 240                                              ANS  219
C                                                                      ANS  220
      DO 220 JJ=1,MXFXT                                                ANS  221
      DO 220 II=1,4                                                    ANS  222
  220 DXYT(II,JJ)=ZERO                                                 ANS  223
C                                                                      ANS  224
  240 CONTINUE                                                         ANS  225
C                                                                      ANS  226
  250 CONTINUE                                                         ANS  227
  907 FORMAT(/1X,24HANALYSIS NOT CARRIED OUT/)                         ANS  228
  908 FORMAT(//120(1H=)//                                              ANS  229
     1 1X,43HSTART OF LOAD INCREMENT BLOCK NUMBER ,I5/1X,48(1H-))      ANS  230
  912 FORMAT(/                                                         ANS  231
     11X,23HINCR BLOCK NUMBER.....=,I5,4X,23HSTARTING INCR NUMBER..=,I8/ANS  232
     21X,23HFINISHING INCR NUMBER.=,I5,4X,23HNO. OF ELEMENT CHANGES=,I8/ANS  233
     31X,23HNUMBER OF LOADS.......=,I5,4X,23HLOAD RATIO OPTION.....=,I8/ANS  234
     41X,23HNUMBER OF FIXITIES....=,I5,4X,23HSTD OUTPUT CODE.......=,I8/ANS  235
     51X,23HOUTPUT OPTION.........=,I5,                               ANS  236
     64X,23HTIME INCREMENT........=,F10.1/                            ANS  237
     71X,23HTIME RATIO OPTION.....=,I5,                               ANS  238
     84X,23HINCR IN GRAVITY FIELD.=,F10.1/)                           ANS  239
  913 FORMAT(//1X,26HERROR IN INCR BLOCK NUMBER,2I6)                   ANS  240
  914 FORMAT(//28H LIST OF ELEMENT ALTERATIONS/1X,27(1H-)/)            ANS  241
  916 FORMAT(//32H LIST OF INCREMENTAL NODAL LOADS/1X,31(1H-)/)        ANS  242
  920 FORMAT(1X,10I8)                                                  ANS  243
  931 FORMAT(/1X,29HPRESCRIBED BOUNDARY CONDITONS/1X,29(1H-)/)         ANS  244
  933 FORMAT(//1X,25HERROR IN INCREMENT NUMBER,2I6,2X,13H(ROUTINE ANS)) ANS  245
  940 FORMAT(1X,I5,3F8.1)                                              ANS  246
  950 FORMAT(/1X,46HINCREASE SIZE OF ARRAYS RINCC, DTM AND IOPT TO,    ANS  247
     1 I5,2X,28HALSO SET INCZ IN ROUTINE ANS)                          ANS  248
 1000 FORMAT(39H SPECIFIED NODAL VALUES OF SHEAR/NORMAL,               ANS  249
     1 36H STRESSES AND EQUIVALENT NODAL LOADS/1X,74(1H-)/5HOELEM,     ANS  250
     2 1X,4HNDE1,2X,4HNDE2,2X,4HSHR1,8X,4HNOR1,8X,4HSHR2,8X,4HNOR2,    ANS  251
     3 8X,4HSHR3,8X,4HNOR3,8X,4HSHR4,8X,4HNOR4,8X,4HSHR5,8X,4HNOR5/    ANS  252
     1 1X,16H(LOAD DIRECTION),2X,3H(X),9X,3H(Y),9X,3H(X),9X,3H(Y),     ANS  253
     2 9X,3H(X),9X,3H(Y),9X,3H(X),9X,3H(Y),9X,3H(X),9X,3H(Y)/)         ANS  254
 1002 FORMAT(1X,3I4,10E12.4)                                           ANS  255
      RETURN                                                           ANS  256
      END                                                              ANS  257
C-----------------------------------------------------------------------ANS  258
      FUNCTION Q(A,N,NDIM)                                             ANS  259
      DIMENSION A(N)                                                   ANS  260
      Q2=0.5*((A(1)-A(2))*(A(1)-A(2))+(A(2)-A(3))*(A(2)-A(3))          ANS  261
     1  +(A(3)-A(1))*(A(3)-A(1)))+3.*A(4)*A(4)
```



```
      IF(NDIM.EQ.2)GOTO 10                                             ANS  262
      Q2=Q2+3.*A(5)*A(5)+3.*A(6)*A(6)                                  ANS  263
   10 Q=SQRT(Q2)                                                       ANS  264
      RETURN                                                           ANS  265
      END                                                              ANS  266
C-----------------------------------------------------------------------ANS  267
      FUNCTION EDS(A,N,NDIM)                                           ANS  268
      DIMENSION A(N)                                                   ANS  269
      EDS2=0.5*((A(1)-A(2))*(A(1)-A(2))+(A(2)-A(3))*(A(2)-A(3))        ANS  270
     1  +(A(3)-A(1))*(A(3)-A(1)))+.75*A(4)*A(4)                        ANS  271
      IF(NDIM.EQ.2)GOTO 10                                             ANS  272
      EDS2=EDS2+0.75*A(5)*A(5)+0.75*A(6)*A(6)                          ANS  273
   10 EDS=2.*SQRT(EDS2)/3.                                             ANS  27
      RETURN                                                           ANS  2
      END                                                              ANS  2
```

ANS 46 : maximum number of increments in a block.

ANS 54 : loop on all **blocks**.

ANS 60–72 : zero arrays dependent on block.

- XYFIB – point loads.
- PIB – global load array.
- PEXIB – loads due to removal of elements.
- PRSLDI – applied pressure loads.
- JEL – element changes.
- IOPT – output options.
- DTM – time steps.
- RINCC – load ratios.

ANS 77–80 : read and write control parameters for block.

ANS 81 : calculate no. of increments in block ($\leqslant 50$).

ANS 85–86 : check increment block number.

ANS 88 : skip if no changes to mesh.

ANS 93–94 : read and write list of element changes.

ANS 96–99 : make changes to mesh and calculate implied loads.

ANS 104–106 : calculate gravity loading for current block (only if there is a change in the gravity acceleration field).

ANS 110–111 : read separate lists of load ratios, output options and time steps for each increment in block.

ANS 112 : skip if no loads have been applied.

ANS 113 : point loads are applied.

ANS 120–122 : applied pressure loads for 3-D are not allowed; print message and stop.

ANS 125 : loop on all sides with applied pressure loads.

ANS 126–127 : read and write applied pressure loads along element side.

ANS 129–136 : change order of pressure loads to suit program.

ANS 138–139 : enter this in common block after checking.

ANS 141–143 : calculate nodal loads from the pressure loads and place them in PIB.

ANS 151 : loop to read directly specified point loads.

ANS 152–153 : read and write point loads.

ANS 156 : program node number.
ANS 157 : g.v.n. of first d.o.f. of node −1.
ANS 158 : no. of d.o.f. of node.
ANS 159 : skip if it has only 1 d.o.f. (assumed to be pore pressure d.o.f.).
ANS 161–162: enter load in XYFIB.
ANS 165 : enter FTT (this is to permit future changes to allow specification of flow rate at the boundary).
ANS 168 : skip if no fixities have been specified.
ANS 174–175: read specified fixities.
ANS 180 : loop on all increments in block.
ANS 182–183: check increment number, print message and stop if out of sequence.
ANS 185–192: update all relevant values for current increment.
ANS 201–212: calculations for current increment (assembly/elimination/output).
ANS 214 : end of block.
ANS 220–222: set all prescribed values of displacements/pore pressures to zero.
ANS 260–264: calculate q.
ANS 270–274: calculate ϵ.

8.4 LOADS

8.4.1 Loads of excavation/construction

Routine **CHANGE**, which is called by ANS, scans the list of element changes. The sign of an entry in array LTYP for each of these elements would indicate whether the element is being added (simulating construction) or removed (simulating excavation). If the sign of LTYP is negative then the element is being added. If it is positive then the element is being removed.

Elements that are added have zero stresses and no memory of any stress history. Therefore these elements cannot have Cam-clay material properties. They can only be elastic models of type 1 or 2 (both are linear elastic models). An attempt to use critical state models for added elements would lead to an error when the D matrix is calculated in the program (e.g. a zero size of yield locus passing through current stress state).

There is also a restriction on elements that are removed to simulate excavation. These should not be added again (simulating, for example, refilling of an excavated trench). The nodal loads due to the addition of elements are given by

$$\text{F(NDIM,NDMX)} = \int_V \mathbf{N}^T \mathbf{w}\, \text{d (vol)}. \tag{8.1}$$

The nodal loads for the removed elements are given by

$$\text{F(NDIM,NDMX)} = \int_V \mathbf{B}^T \boldsymbol{\sigma}\, \text{d (vol)} - \int \mathbf{N}^T \mathbf{w}\, \text{d (vol)} - \int_S \mathbf{N}^T \boldsymbol{\tau}\, \text{d (area)}, \tag{8.2}$$

where $\boldsymbol{\sigma}$ is the **current total** stress in the element.

$\int_V \mathbf{N}^T \mathbf{w}\, \text{d (vol)}$ is calculated in routine SELF.

$\int_V \mathbf{B}^T \boldsymbol{\sigma}\, \text{d (vol)}$ is calculated in routine EQLIB (see section 7.7.4).

$\int_S \mathbf{N}^T \boldsymbol{\tau}\, \text{d (area)}$ is calculated in routine DISTLD.

The element contributions are accumulated in PI(NDF).

Because of the approximate way in which the excavation process is simulated and in order to satisfy the equilibrium at the end of each increment, an array PEXIB is required. It consists of nodal loads equivalent to the stresses in elements which are being removed. Remembering that these elements vanish in the first increment of the increment block, this causes an imbalance. If the removal of elements is carried out in an increment block with just one increment then there is no problem. Nodal loads equal and opposite to element stresses are applied to cancel out the stresses in the removed elements.

However, if these loads are spread over a number of increments, this obviously results in an imbalance, as the stresses in the removed elements only decrease gradually to zero. Equilibrium will be satisfied at the end of the increment block. Array PEXIB provides the balancing loads to maintain the correctness of the equilibrium check.

Routine CHANGE

```
      SUBROUTINE CHANGE(IW6,IN,NCH,NN,NNOD1,NTPE,NIP,NEL,MUMAX,NNZ,NDF, CHNG   1
     1 NDIM,NVRS,NDMX,NL,NB,NS,NPR,NMT,NPT,NSP,NPL,XYZ,VARINT,          CHNG   2
     2 PI,PEXIB,ELCOD,DS,SHFN,CARTD,B,F,NCONN,MAT,LTYP,MREL,NREL,       CHNG   3
     3 NW,JEL,NP1,NP2,MXEN,LL,PR,TGRAV)                                CHNG   4
C*********************************************************************CHNG   5
C     REMOVES/ADDS ELEMENTS FROM/TO GEOMETRY MESH AND CALCULATES       CHNG   6
C     IMPLIED LOADS                                                    CHNG   7
C*********************************************************************CHNG   8
      REAL LL                                                          CHNG   9
      DIMENSION PRES(10)                                               CHNG  10
      DIMENSION XYZ(NDIM,NN),VARINT(NVRS,NIP,NEL),PI(NDF),PEXIB(NDF),  CHNG  11
     1 ELCOD(NDIM,NDMX),DS(NDIM,NDMX),SHFN(NDMX),CARTD(NDIM,NDMX),     CHNG  12
     2 B(NS,NB),F(NDIM,NDMX),LL(NL),PR(NPR,NMT)                        CHNG  13
      DIMENSION NCONN(NTPE,NEL),MAT(NEL),LTYP(NEL),MREL(MUMAX),        CHNG  14
     1 NREL(NNZ),NW(NNOD1),JEL(NEL),NP1(NPL),NP2(NPL)                  CHNG  15
      COMMON /ELINF / LINFO(50,15)                                     CHNG  16
      COMMON /PRSLD / PRESLD(10,100),LEDG(100),NDE1(100),NDE2(100),NLED CHNG  17
      COMMON /LOADS / FB(2,15)                                         CHNG  18
C-------------------------------------------------------------------CHNG  19
C     ISTGE - CODE TO INDICATE STAGE OF THE ANALYSIS                   CHNG  20
C-------------------------------------------------------------------CHNG  21
```

```
      ISTGE=2                                                          CHNG  22
      KSTGE=2                                                          CHNG  23
C-----------------------------------------------------------------------CHNG  24
C     LOOP ON ALL ELEMENTS WHICH APPEAR IN                             CHNG  25
C     THE LIST OF CHANGES                                              CHNG  26
C-----------------------------------------------------------------------CHNG  27
      DO 150 J=1,NCH                                                   CHNG  28
      JK=JEL(J)                                                        CHNG  29
      JJ=MREL(JK)                                                      CHNG  30
C-----------------------------------------------------------------------CHNG  31
C     EXCLUDED (REMOVED)  ELEMENTS  HAVE  ELEMENT  TYPE  NEGATED       CHNG  32
C-----------------------------------------------------------------------CHNG  33
      LTYP(JJ)=-LTYP(JJ)                                               CHNG  34
      LT=LTYP(JJ)                                                      CHNG  35
      IJ=ISIGN(1,LT)                                                   CHNG  36
C-----------------------------------------------------------------------CHNG  37
C     BY-PASS  FOR  INITIAL  MESH  CHANGE  ONLY                        CHNG  38
C-----------------------------------------------------------------------CHNG  39
      IF(IN.EQ.0) GOTO 150                                             CHNG  40
      LT=IABS(LT)                                                      CHNG  41
      INDX=LINFO(12,LT)                                                CHNG  42
      NDN=LINFO(5,LT)                                                  CHNG  43
      NGP=LINFO(11,LT)                                                 CHNG  44
      NAC=LINFO(15,LT)                                                 CHNG  45
      KM=MAT(JJ)                                                       CHNG  46
      DENS=PR(8,KM)*TGRAV                                              CHNG  47
      IF(IJ.GT.0) GOTO 125                                             CHNG  48
C-----------------------------------------------------------------------CHNG  49
C     CALCULATE  BOUNDARY  FORCES (IN ROUTINE EQLIB) AND SELF-WEIGHT   CHNG  50
C     FORCES (IN ROUTINE SELF) FOR  REMOVED  ELEMENTS                  CHNG  51
C-----------------------------------------------------------------------CHNG  52
      CALL EQLIB(JJ,JK,LT,NGP,NIP,INDX,NTPE,NEL,NDIM,NN,NDMX,NDN,      CHNG  53
     1   NS,NB,NAC,NVRS,XYZ,VARINT,ELCOD,DS,SHFN,                      CHNG  54
     1   CARTD,B,F,NCONN,LL,ISTGE)                                     CHNG  55
C                                                                      CHNG  56
      DO 10 I=1,NDN                                                    CHNG  57
      NCOR=NCONN(I,JJ)                                                 CHNG  58
      II=IABS(NCOR)                                                    CHNG  59
      N1=NW(II)-1                                                      CHNG  60
C                                                                      CHNG  61
      DO 10 ID=1,NDIM                                                  CHNG  62
      PEXIB(N1+ID)=PEXIB(N1+ID)+F(ID,I)                                CHNG  63
   10 PI(N1+ID)=PI(N1+ID)+F(ID,I)                                      CHNG  64
      DENS=PR(8,KM)*TGRAV                                              CHNG  65
C                                                                      CHNG  66
      CALL SELF(IW6,JJ,NN,NEL,NTPE,NDN,NDIM,NAC,NPR,NMT,XYZ,           CHNG  67
     1   ELCOD,DS,SHFN,F,NCONN,MAT,LL,PR,LT,INDX,DENS,JK,KSTGE)        CHNG  68
C                                                                      CHNG  69
      DO 20 KK=1,NDN                                                   CHNG  70
      NCOR=NCONN(KK,JJ)                                                CHNG  71
      KKK=NW(NCOR)-1                                                   CHNG  72
C                                                                      CHNG  73
      DO 20 ID=1,NDIM                                                  CHNG  74
      PEXIB(KKK+ID)=PEXIB(KKK+ID)-F(ID,KK)                             CHNG  75
   20 PI(KKK+ID)=PI(KKK+ID)-F(ID,KK)                                   CHNG  76
C-----------------------------------------------------------------------CHNG  77
C     CALCULATE FORCES EQUAL TO BOUNDARY STRESSES FOR REMOVED ELEMENTS CHNG  78
C-----------------------------------------------------------------------CHNG  79
      DO 80 KE=1,NLED                                                  CHNG  80
      LNE=LEDG(KE)                                                     CHNG  81
      IF(LNE.NE.JK)GOTO 80                                             CHNG  82
      ND1=NDE1(KE)                                                     CHNG  83
      ND2=NDE2(KE)                                                     CHNG  84
C                                                                      CHNG  85
      DO 60 KV=1,MXEN                                                  CHNG  86
   60 PRES(KV)=PRESLD(KV,KE)                                           CHNG  87
```



```
      CALL ZEROR2(FB,2,15)                                             CHNG  88
C                                                                      CHNG  89
      CALL DISTLD(IW6,NN,NEL,NDF,NNOD1,NTPE,NDIM,MUMAX,NNZ,NPL,        CHNG  90
     1   XYZ,PI,NCONN,LTYP,MREL,NREL,NW,NP1,NP2,PRES,LNE,ND1,ND2,NPT,  CHNG  91
     2   NSP,0,0,-1.)                                                  CHNG  92
C                                                                      CHNG  93
      DO 70 KK=1,NDN                                                   CHNG  94
      NCOR=NCONN(KK,JJ)                                                CHNG  95
      KKK=NW(NCOR)-1                                                   CHNG  96
C                                                                      CHNG  97
      DO 70 ID=1,NDIM                                                  CHNG  98
   70 PEXIB(KKK+ID)=PEXIB(KKK+ID)-FB(ID,KK)                            CHNG  99
   80 CONTINUE                                                         CHNG 100
      GOTO 150                                                         CHNG 101
C-----------------------------------------------------------------------CHNG 102
C     CALCULATE  SELF-WEIGHT  FORCES  FOR  ADDED  ELEMENTS             CHNG 103
C-----------------------------------------------------------------------CHNG 104
  125 CALL SELF(IW6,JJ,NN,NEL,NTPE,NDN,NDIM,NAC,NPR,NMT,XYZ,           CHNG 105
     1   ELCOD,DS,SHFN,F,NCONN,MAT,LL,PR,LT,INDX,DENS,JK,KSTGE)        CHNG 106
C                                                                      CHNG 107
      DO 140 KK=1,NDN                                                  CHNG 108
      NCOR=NCONN(KK,JJ)                                                CHNG 109
      KKK=NW(NCOR)-1                                                   CHNG 110
C                                                                      CHNG 111
      DO 140 ID=1,NDIM                                                 CHNG 112
  140 PI(KKK+ID)=PI(KKK+ID)+F(ID,KK)                                   CHNG 113
  150 CONTINUE                                                         CHNG 114
      RETURN                                                           CHNG 115
      END                                                              CHNG 116
```

CHNG 28 : loop on all elements which appear in the list of changes.

CHNG 29 : get user element number.

CHNG 30 : get program element number.

CHNG 34 : change sign of type number.

CHNG 36 : IJ = −1 for removed element.
IJ = 1 for added element.

CHNG 40 : skip calculation of implied loads if changes are to the initial mesh to form the primary mesh. Note that the primary mesh is the mesh in the first increment and the initial mesh is the complete mesh defined in the geometry part of program.

CHNG 42–45 : element type dependent parameters.
INDX – a starting index to arrays W and L.
NDN – no. of displacement nodes in element.
NGP – no. of integration points in element.

CHNG 46 : material zone number of element.

CHNG 47 : calculate $n\gamma$ term, where n = centrifugal acceleration field and γ = unit weight of soil.

CHNG 48 : skip if element is being added.

CHNG 53–55 : calculate $\int \mathbf{B}^{\mathrm{T}}\boldsymbol{\sigma}\,\mathrm{d}(\mathrm{vol})$ for element being removed (entered in array F).

CHNG 57–64 : slot array F into PI.

CHNG 67–68 : calculate $\int N^T w \, d(vol)$ for removed element (entered in array F).
CHNG 70–76 : slot array F into PI.
CHNG 80 : loop to find if element being removed has applied pressure load.
CHNG 82 : skip if element not found in list of pressure loads.
CHNG 83–84 : nodes at either end of element side with pressure load.
CHNG 90–92 : calculate equivalent nodal loads for pressure loads.
CHNG 94–100 : slot load terms in array PEXIB.
CHNG 101 : completion of calculations for current (removed) element.
CHNG 105–106 : calculate $\int N^T w \, d(vol)$ for added element (entered in F).
CHNG 108–113 : slot array F into PI.
CHNG 114 : end of loop on element changes.

8.4.2 Loads from body forces

If there is a change in the body forces as in the case of a centrifuge test when the speed is changed, then there is a change in the self-weight loads throughout the soil mass. This is indicated by the parameter DGRAV, which is defined as a ratio of earth's gravity (i.e. *g*). **SEL1** is the routine which controls the calculation of the equivalent nodal loads from the change in body force for each element. Routine SELF is again used in the calculation of these loads.

Routine SEL1

```
      SUBROUTINE SEL1(IW6,ICHEL,NN,NNOD1,NTPE,NIP,NEL,NDF,          SEL1   1
     1 MUMAX,NL,NDIM,NDMX,NPR,NMT,XYZ,P,ELCOD,DS,SHFN,              SEL1   2
     2 F,NCONN,MAT,LTYP,MRELVV,MREL,NW,JEL,LL,PR,NTY,DGRAV)         SEL1   3
C*************************************************************************SEL1   4
C     CALCULATES  SELF-WEIGHT  LOAD  VECTOR                         SEL1   5
C*************************************************************************SEL1   6
      REAL LL                                                       SEL1   7
      DIMENSION XYZ(NDIM,NN),P(NDF),ELCOD(NDIM,NDMX),               SEL1   8
     1 DS(NDIM,NDMX),SHFN(NDMX),F(NDIM,NDMX)                        SEL1   9
      DIMENSION NCONN(NTPE,NEL),MAT(NEL),LTYP(NEL),MRELVV(NEL),     SEL1  10
     1 MREL(MUMAX),NW(NNOD1),JEL(NEL)                               SEL1  11
      DIMENSION LL(NL),PR(NPR,NMT),NTY(NMT)                         SEL1  12
      COMMON /ELINF / LINFO(50,15)                                  SEL1  13
      COMMON /PARS  / PYI,ALAR,ASMVL,ZERO                           SEL1  14
C---------CODE TO INDICATE STAGE OF THE ANALYSIS                    SEL1  15
      KSTGE=3                                                       SEL1  16
C-------------------------------------------------------------------SEL1  17
C     ITERATE  FOR  ALL  ELEMENTS                                   SEL1  18
C-------------------------------------------------------------------SEL1  19
      DO 50 J=1,NEL                                                 SEL1  20
      JK=MRELVV(J)                                                  SEL1  21
C-------------------------------------------------------------------SEL1  22
C     BY-PASS  ADDITION  IF  ELEMENT  NOT  IN  CURRENT  MESH        SEL1  23
C-------------------------------------------------------------------SEL1  24
      LT=LTYP(J)                                                    SEL1  25
      IF(LT.LT.0)GO TO 50                                           SEL1  26
      GOTO(50,22,22,22,22,22,22,22,22,22,22),LT                     SEL1  27
      WRITE(IW6,900)JK,LT                                           SEL1  28
  900 FORMAT(1X,7HELEMENT,I5,2X,18HIS OF UNKNOWN TYPE,I5,           SEL1  29
```



```
     1 14H(ROUTINE SEL1))
   22 INDX=LINFO(12,LT)                                             SEL1  30
      NDN=LINFO(5,LT)                                               SEL1  31
      NAC=LINFO(15,LT)                                              SEL1  32
      K=MAT(J)                                                      SEL1  33
      DENS=DGRAV*PR(8,K)                                            SEL1  34
      IF(DENS.LE.ASMVL)GO TO 50                                     SEL1  35
      CALL SELF(IW6,J,NN,NEL,NTPE,NDN,NDIM,NAC,NPR,NMT,XYZ,         SEL1  36
     1   ELCOD,DS,SHFN,F,NCONN,MAT,LL,PR,LT,INDX,DENS,JK,KSTGE)     SEL1  37
C                                                                   SEL1  38
      DO 30 JJ=1,NDN                                                SEL1  39
      JN=NCONN(JJ,J)                                                SEL1  40
      JL=NW(JN)-1                                                   SEL1  41
C                                                                   SEL1  42
      DO 30 ID=1,NDIM                                               SEL1  43
   30 P(JL+ID)=P(JL+ID)+F(ID,JJ)                                    SEL1  44
   50 CONTINUE                                                      SEL1  45
      RETURN                                                        SEL1  46
      END                                                           SEL1  47
                                                                    SEL1  48
```

SEL1 20 : loop on all elements.
SEL1 25 : element type number.
SEL1 26 : skip; element is not present in current mesh.
SEL1 27 : skip if bar element (LT = 1). No self-weight.
SEL1 31–33 : element type dependent parameters.
INDX – starting index to arrays W and L.
NDN – no. of displacement nodes in element.
SEL1 34 : material zone number.
SEL1 35 : $n\gamma$.
n – centrifugal acceleration field.
γ – unit weight of soil.
SEL1 36 : skip if no self-weight loads.
SEL1 37–38 : calculate nodal loads equivalent to self-weight.

$$F = \int_V N^T w \, d(vol).$$

SEL1 40–45 : slot F in array PI (in this routine called P).
SEL1 46 : end of loop on all elements.

8.4.3 Load ratios

The external loads (pressure loads along mesh boundary) are yet to be specified. However, at this stage the distribution of the loading between the individual increments is read in. The output options (printing out of displacements, stresses, etc.) and the time steps (in a consolidation analysis) are also read in.

Routine FACTOR

```
      SUBROUTINE FACTOR(IR5,IW6,NOINC,ILDF,IOCD,ITMF,IOUTS,         FACT   1
     1 RINCC,DTM,IOPT,DTIME)                                        FACT   2
C*************************************************************************FACT   3
C     LOAD RATIOS, TIME RATIOS (CONSOLIDATION ANALYSIS) AND OUTPUT  FACT   4
C     OPTIONS FOR ALL INCREMENTS IN THE BLOCK                       FACT   5
```

```
C*****************************************************************FACT   6
      DIMENSION RINCC(NOINC),DTM(NOINC),IOPT(NOINC)                 FACT   7
      COMMON /PARS  / PYI,ALAR,ASMVL,ZERO                           FACT   8
C-----------------------------------------------------------------FACT   9
C     READ LOAD RATIOS FOR INCREMENTS                             FACT  10
C-----------------------------------------------------------------FACT  11
      FSTD=1.0/FLOAT(NOINC)                                         FACT  12
      IF(ILDF.EQ.0)GO TO 98                                         FACT  13
      WRITE(IW6,948)                                                FACT  14
      READ(IR5,*)(RINCC(IN),IN=1,NOINC)                             FACT  15
      WRITE(IW6,954)(RINCC(IN),IN=1,NOINC)                          FACT  16
      GO TO 122                                                     FACT  17
   98 DO 100 IK=1,NOINC                                             FACT  18
  100 RINCC(IK)=FSTD                                                FACT  19
C-----------------------------------------------------------------FACT  20
C     READ OUTPUT OPTIONS                                         FACT  21
C-----------------------------------------------------------------FACT  22
  122 IF(IOCD.EQ.0)GO TO 127                                        FACT  23
      WRITE(IW6,960)                                                FACT  24
      READ(IR5,*)(IOPT(IN),IN=1,NOINC)                              FACT  25
      WRITE(IW6,964)(IOPT(IN),IN=1,NOINC)                           FACT  26
      GO TO 131                                                     FACT  27
C                                                                   FACT  28
  127 DO 130 IK=1,NOINC                                             FACT  29
  130 IOPT(IK)=IOUTS                                                FACT  30
C-----------------------------------------------------------------FACT  31
C     READ TIME RATIOS FOR INCREMENTS                             FACT  32
C-----------------------------------------------------------------FACT  33
  131 IF(DTIME.LT.ASMVL.OR.ITMF.EQ.0)GO TO 132                      FACT  34
      WRITE(IW6,965)                                                FACT  35
      READ(IR5,*)(DTM(IN),IN=1,NOINC)                               FACT  36
      WRITE(IW6,968)(DTM(IN),IN=1,NOINC)                            FACT  37
      GO TO 136                                                     FACT  38
C                                                                   FACT  39
  132 DO 135 IK=1,NOINC                                             FACT  40
  135 DTM(IK)=FSTD*DTIME                                            FACT  41
  136 CONTINUE                                                      FACT  42
      RETURN                                                        FACT  43
  948 FORMAT(/1X,34HLIST OF LOAD RATIOS FOR INCREMENTS/1X,34(1H-)/) FACT  44
  954 FORMAT(1X,10F8.1)                                             FACT  45
  960 FORMAT(/1X,35HLIST OF OUTPUT CODES FOR INCREMENTS/1X,35(1H-)/)FACT  46
  964 FORMAT(1X,10I6)                                               FACT  47
  965 FORMAT(/1X,33HLIST OF TIME STEPS FOR INCREMENTS/1X,33(1H-)/)  FACT  48
  968 FORMAT(1X,8F10.0)                                             FACT  49
      END                                                           FACT  50
```

FACT 12 : equal load/time ratios.
FACT 13 : skip if no separate list of load ratios is to be read.
FACT 15–16 : read separate list of load ratios for each increment.
FACT 18–19 : equal load ratio for all increments.
FACT 23 : skip if no separate list of output options is to be read.
FACT 25–26 : read separate list of output options for each increment.
FACT 29–30 : identical standard output option for all increments.
FACT 34 : skip if no separate list of time steps is to be read or if DTIME for the increment block is equal to zero.
FACT 35–37 : read separate list of time steps for each increment.
FACT 40–41 : equal time steps for all increments.

The loading (NLOD), self-weight loads (DGRAV) and prescribed displacements (and pore pressures) (NFIX) are specified for the entire increment block, and are applicable to that particular increment block only. The loading and any non-zero prescribed displacement for the individual increments are taken as ratios (< 1) of that for the entire increment block.

There is no restriction on how these loading and non-zero prescribed displacements are divided among the increments in an increment block. They are *equally* divided between all the increments if ILDF = 0 in record R. However, if the user wants to distribute the loading (and non-zero prescribed displacements) unevenly between the increments, then by setting ILDF = 1 a separate list of load ratios is read in record T1. (This is generally useful in an analysis where large load increments can be applied when the problem is in the elastic state and smaller load increments as plastic yielding takes place.)

It should be noted that the same ratios R(I) etc. (record T1) apply to the pressure loading (NLOD – record U), the gravity loading (DGRAV – record R) and the prescribed displacements (and pore pressures) (NFIX – record V).

The sum of ratios R(I) must be equal to 1.

8.4.4 Loads from pressure along mesh boundary

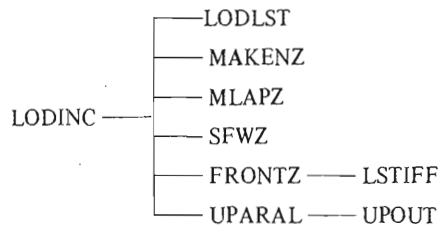
The external loading is now read by the program. There are two options: the user can convert applied pressure loading into equivalent nodal loads and specify these directly as nodal loads along with the node numbers, or the user can specify the pressure loads (both normal and shear components) along element sides. When using the latter option the user in fact is specifying the nodal values of the pressure distribution. The second option is the more convenient because calculation of equivalent nodal loads is not straightforward for a higher-order element like the cubic strain triangle.

The directly specified nodal loads are stored in an array (XYFIB(NDF)). For the specified pressure distributions these are stored in a set of arrays in the form they are read in. Later, using routines DISTLD and SFR1, these are converted into equivalent nodal loads and are added to array PIB.

8.5 LOAD INCREMENT LOOP

The control routine **LODINC** delegates the calculation of the stiffness matrix (carried out by routine LSTIFF) and solution to routine FRONTZ, and printing out the results to UPOUT (via UPARAL).

Sections 8.6 to 8.8 deal with the calculation of the element stiffness matrix. This is considered separately from the frontal solution routine which is described in sections 8.9 to 8.12. Even though the calculation of element stiffnesses and elimination using the frontal method take place alternately, this is done for the sake of clarity of presentation.



Routine LODINC

```
      SUBROUTINE LODINC(NN,NEL,NDF,NNOD1,NTPE,NIP,NVRS,                 LDNC   1
     1 NVRN,NDIM,MUMAX,NDZ,IFRZ,NNZ,NDMX,NPMX,                          LDNC   2
     2 NS,NB,NL,NPR,NMT,NPT,NSP,NPL,MDFE,KES,NVPN,                      LDNC   3
     3 INXL,MXEN,MXLD,LV,NVTX,ND,                                       LDNC   4
     4 XYZ,DI,DA,VARINT,P,PT,PIB,REAC,PCOR,PEQT,XYFT,XYFIB,             LDNC   5
     5 STR,PEXIB,PEXI,PCONI,D,ELCOD,DS,SHFN,CARTD,B,DB,                 LDNC   6
     6 FT,SS,ES,ELCODP,E,PE,RN,AA,ETE,RLT,                              LDNC   7
     7 NCONN,MAT,LTYP,MRELVV,MREL,NRELVV,NREL,NW,NQ,                    LDNC   8
     8 JEL,IDFX,NDEST,NP1,NP2,IFR,NDL,NWL,NMOD,                         LDNC   9
     9 CIP,LL,V,FXYZ,PR,PDISLD,PRES,NTY,A,MFZ,                          LDNC  10
     1 DTIMEI,TTIME,DGRAVI,TGRAV,IOUT,JS,J,FRACLD,                      LDNC  11
     2 FRACT,ICOR,IUPD,IBC,NLOD,NLDS,IWL)                               LDNC  12
C*********************************************************************LDNC  13
C     LOAD INCREMENT ROUTINE                                           LDNC  14
C*********************************************************************LDNC  15
      REAL LL                                                           LDNC  16
C--------USE THE FOLLOWING STATEMENT AFTER CONVERTING PROGRAM TO DOUBLE LDNC  17
C--------PRECISION.  ARRAY A ALWAYS USES ONE NUMERIC STORAGE LOCATION   LDNC  18
CC    REAL A                                                            LDNC  19
      DIMENSION XYZ(NDIM,NN),DI(NDF),DA(NDF),VARINT(NVRS,NIP,NEL),      LDNC  20
     1 P(NDF),PT(NDF),PIB(NDF),REAC(NDF),PCOR(NDF),PEQT(NDF),XYFT(NDF), LDNC  21
     2 XYFIB(NDF),STR(NVRN,NIP,NEL),PEXIB(NDF),PEXI(NDF),PCONI(NDF)     LDNC  22
      DIMENSION D(NS,NS),ELCOD(NDIM,NDMX),DS(NDIM,NDMX),SHFN(NDMX),     LDNC  23
     1 CARTD(NDIM,NDMX),B(NS,NB),DB(NS,NB),FT(NDIM,NDMX),               LDNC  24
     2 SS(NB,NB),ES(KES)                                                LDNC  25
      DIMENSION ELCODP(NDIM,NPMX),E(NDIM,NPMX),PE(NDIM,NPMX),           LDNC  26
     1 RN(NB),AA(NPMX),ETE(NPMX,NPMX),RLT(NB,NPMX)                      LDNC  27
      DIMENSION NCONN(NTPE,NEL),MAT(NEL),LTYP(NEL),MRELVV(NEL),         LDNC  28
     1 MREL(MUMAX),NRELVV(NN),NREL(NNZ),NW(NNOD1),NQ(NN),JEL(NEL),      LDNC  29
     2 IDFX(NDF),NDEST(NN),NP1(NPL),NP2(NPL)                            LDNC  30
      DIMENSION IFR(IFRZ),NDL(MDFE),NWL(NPMX),NMOD(NIP,NEL)             LDNC  31
      DIMENSION CIP(NDIM),LL(NL),V(LV),FXYZ(NDIM),PR(NPR,NMT),          LDNC  32
     1 PDISLD(NDIM,NPT),PRES(NDIM,NPT),NTY(NMT),A(MFZ)                  LDNC  33
      COMMON /PRSLD / PRESLD(10,100),LEDG(100),NDE1(100),NDE2(100),NLED LDNC  34
      COMMON /PRLDI / PRSLDI(10,100),LEDI(100),NDI1(100),NDI2(100),ILOD LDNC  35
      COMMON /DEVICE/ IR1,IR4,IR5,IW2,IW4,IW6,IW7,IW8,IW9               LDNC  36
      COMMON /PRECSN/ NP                                                LDNC  37
C                                                                       LDNC  38
      WRITE(IW6,915)JS,J,FRACLD                                         LDNC  39
      WRITE(IW6,917)DGRAVI,TGRAV                                        LDNC  40
      WRITE(IW6,919)DTIMEI,TTIME                                        LDNC  41
C-----------------------------------------------------------------------LDNC  42
C     BOUNDARY CONDITIONS (LOADS AND DISPLACEMENTS) ARE PRINTED         LDNC  43
C     EVERY IBC INCREMENTS                                              LDNC  44
C     IBC =  0 NOT PRINTED IN ANY INCREMENT                             LDNC  45
C     IBC =  1 PRINTED IN EACH INCREMENT                                LDNC  46
C     IBC = 10 PRINTED IN EVERY 10TH INCREMENT                          LDNC  47
C-----------------------------------------------------------------------LDNC  48
      IOPBC=0                                                           LDNC  49
      IF(IBC.EQ.0)GOTO 130                                              LDNC  50
      NJS=IBC*(JS/IBC)                                                  LDNC  51
      IF(NJS.EQ.JS)IOPBC=1                                              LDNC  52
```



```
C                                                                       LDNC  53
  130 DO 140 IM=1,NDF                                                   LDNC  54
      XYFT(IM)=XYFT(IM)+XYFIB(IM)*FRACLD                                LDNC  55
  140 P(IM)=FRACLD*PIB(IM)+FRACLD*XYFIB(IM)                             LDNC  56
C                                                                       LDNC  57
      DO 145 IM=1,NDF                                                   LDNC  58
  145 PEXI(IM)=(1.0-FRACT)*PEXIB(IM)                                    LDNC  59
C-----------------------------------------------------------------------LDNC  60
C     UPDATE LIST OF PRESSURE LOADING ALONG ELEMENT EDGES               LDNC  61
C-----------------------------------------------------------------------LDNC  62
      IF(NLOD.GE.0)GO TO 162                                            LDNC  63
C                                                                       LDNC  64
      DO 160 ISD=1,NLDS                                                 LDNC  65
      LNE=LEDI(ISD)                                                     LDNC  66
      ND1=NDI1(ISD)                                                     LDNC  67
      ND2=NDI2(ISD)                                                     LDNC  68
      ICT=0                                                             LDNC  69
C *** N2D = 2 FOR TWO DIMENSIONAL PROBLEMS                              LDNC  70
      N2D=2                                                             LDNC  71
      DO 150 IK=1,NPT                                                   LDNC  72
      DO 150 IJ=1,N2D                                                   LDNC  73
      ICT=ICT+1                                                         LDNC  74
  150 PRES(IJ,IK)=FRACLD*PRSLDI(ICT,ISD)                                LDNC  75
      CALL LODLST(IW6,LNE,ND1,ND2,PRES,NDIM,NPT,0,MXLD)                 LDNC  76
  160 CONTINUE                                                          LDNC  77
  162 CONTINUE                                                          LDNC  78
C-----------------------------------------------------------------------LDNC  79
C     INITIALISE INCREMENTAL DISPLACEMENTS                              LDNC  80
C-----------------------------------------------------------------------LDNC  81
      CALL ZEROR1(DI,NDF)                                               LDNC  82
C-----------------------------------------------------------------------LDNC  83
C       PREFRONT                                                        LDNC  84
C-----------------------------------------------------------------------LDNC  85
      CALL MAKENZ(NTPE,NEL,NN,NCONN,LTYP,NQ,INXL)                       LDNC  86
      CALL MLAPZ(NTPE,NEL,NN,NCONN,LTYP,NQ)                             LDNC  87
      CALL SFWZ(MNFZ,NTPE,NEL,NN,MUMAX,NNZ,IFRZ,NCONN,LTYP,             LDNC  88
     1   MREL,NREL,NQ,NDEST,IFR,1,MCORE,NCORET)                         LDNC  89
C-----------------------------------------------------------------------LDNC  90
C     SOLVE  EQUATIONS USING  FRONTAL  SOLUTION                         LDNC  91
C-----------------------------------------------------------------------LDNC  92
      MFZN=MFZ/NP                                                       LDNC  93
      CALL FRONTZ(MNFZ,DTIMEI,NN,NNOD1,NEL,NDF,NTPE,NIP,NPR,NMT,        LDNC  94
     1   KES,NS,NB,NDIM,NDMX,NVRS,NPMX,INXL,MDFE,IFRZ,MUMAX,NNZ,NL,     LDNC  95
     2   XYZ,DI,DA,VARINT,P,PCOR,D,ELCOD,DS,SHFN,CARTD,B,DB,SS,ES,      LDNC  96
     3   ELCODP,E,PE,RN,AA,ETE,RLT,NCONN,MAT,LTYP,MRELVV,MREL,          LDNC  97
     4   NRELVV,NREL,NW,NQ,IDFX,NDEST,IFR,NDL,NWL,                      LDNC  98
     5   NMOD,LL,PR,NTY,A,MFZN,FRACLD,IOPBC)                            LDNC  99
C-----------------------------------------------------------------------LDNC 100
C     UPDATE  AND  OUTPUT  CALCULATIONS                                 LDNC 101
C-----------------------------------------------------------------------LDNC 102
      CALL UPARAL(TTIME,TGRAV,IOUT,NN,ND,NNOD1,NEL,NDF,NTPE,NIP,NPT,    LDNC 103
     1   NSP,NPL,NDZ,NVRS,NVRN,NDIM,MUMAX,NNZ,NDMX,NPMX,NS,NB,NL,INXL,  LDNC 104
     2   NPR,NMT,MXEN,XYZ,DI,DA,VARINT,P,PT,PCOR,PEQT,XYFT,STR,PEXI,     LDNC 105
     3   PCONI,D,ELCOD,DS,SHFN,CARTD,B,FT,AA,NCONN,MAT,LTYP,MREL,MRELVV, LDNC 106
     4   NREL,NW,NQ,JEL,IDFX,NP1,NP2,NWL,NMOD,CIP,LL,PR,                 LDNC 107
     5   NTY,A,MFZ,ICOR,IUPD,FRACT,JS,IWL)                              LDNC 108
C-----------------------------------------------------------------------LDNC 109
      RETURN                                                            LDNC 110
  915 FORMAT(//120(1H=)//                                               LDNC 111
     1 1X,32HSTART OF LOAD INCREMENT NUMBER  ,I5,                       LDNC 112
     2 4X,22HINCREMENT BLOCK NUMBER,I5,4X,13HLOAD RATIO = ,F5.2/        LDNC 113
     3 1X,90(1H-))                                                      LDNC 114
  917 FORMAT(/22H INCR GRAVITY LEVEL = ,E12.4,                          LDNC 115
     1 24H  TOTAL GRAVITY LEVEL = ,E12.4)                               LDNC 116
  919 FORMAT(/18H TIME INCREMENT = ,E12.4,4X,15H  TOTAL TIME = ,E12.4)   LDNC 117
      END                                                               LDNC 118
```

LDNC 39–41 : write load ratios, centrifugal acceleration field and time steps for current increment.
LDNC 49–52 : option to print out boundary conditions in selected increments.
LDNC 54–56 : increment accumulated point loads by the loads applied in current increment. Also calculate the loads for current increment.
LDNC 63 : skip if no pressure loads are present (no need to update list of pressure loads).
LDNC 65–74 : loop on all sides with pressure load.
LDNC 75 : enter pressures applied for current increment in PRES.
LDNC 76 : update the cumulative list of pressure loads.
LDNC 77 : end of loop on all sides with pressure loads.
LDNC 82 : zero incremental displacements array.
LDNC 86 : calculate d.o.f. of each node.
LDNC 87 : mark last appearance of nodes in array NCONN.
LDNC 88–89 : calculate maximum frontwidth for current mesh and core requirement to solve the equations.
LDNC 94–99 : assemble stiffness matrices, eliminate and solve for unknown displacements using the frontal method.
LDNC 103–108 : calculate incremental strains and stresses; update cumulative displacements, strains and stresses and print the results.

The incremental loads for the current increment are calculated as a fraction of the incremental loads for the increment block.

$$\text{PI (NDF)} = \text{FRACLD} * \text{PIB (NDF)}. \tag{8.3}$$

An equilibrium check is carried out at the end of each increment. Routine LODLST updates the current level of (external) pressure loads. There are two lists of pressure loads: one is the current level of accumulated pressure load and the other is the pressure loads applied in the current increment block. For each increment, the appropriate ratio of pressure loads from that for the increment block is added to the current list so that the external pressure loads should equal the element stresses at the end of each increment.

8.6 ELEMENT STIFFNESS MATRIX

Routine **LSTIFF** is called to calculate each element stiffness matrix. Routine LSTIFF carries out the following calculation.

$$K = \sum_{i=1}^{\text{NGP}} \mathbf{B}_i^{\mathrm{T}} \mathbf{D}_i \mathbf{B}_i \, |\mathbf{J}| \, W_i, \tag{8.4}$$

where i is the integration point. The calculation of the **B** matrix has been dealt

with in Chapter 7 in detail. The calculation of the **D** matrix has been considered in Chapter 5. Depending on the constitutive relationship being used, a different routine is called. Routine DCON is a general linear elastic model which deals with either isotropic or cross-anisotropic material behaviour. The **D** matrix is independent of the stress level and therefore is a constant for a given element. Hence it is calculated once for each element outside the integration point loop.

An **'extended'** element stiffness matrix is calculated in a consolidation analysis. The technique used in solving the transient problem is known as the **time-marching** method. This has been discussed in section 3.6.2, and the final form of the equations are reproduced here:

$$\begin{bmatrix} \mathbf{K} & \mathbf{L} \\ \mathbf{L}^{\mathrm{T}} & -\Phi\Delta t \end{bmatrix} \cdot \begin{bmatrix} \Delta a \\ \Delta b \end{bmatrix} = \begin{bmatrix} \mathbf{P} \\ \Phi \mathbf{b}_0 \, \Delta t \end{bmatrix} \tag{8.5}$$

where

$$\mathbf{K} = \int_V \mathbf{B}^{\mathrm{T}} \mathbf{D} \mathbf{B} \, \mathrm{d}\,(\mathrm{vol}),$$

$$\mathbf{L} = \int_V \mathbf{E}^{\mathrm{T}} \mathbf{m} \bar{\mathbf{N}} \, \mathrm{d}\,(\mathrm{vol}),$$

$$\Phi = \int_V \mathbf{E}^{\mathrm{T}} \frac{\mathbf{k}}{\gamma_{\mathrm{w}}} \mathbf{E} \, \mathrm{d}\,(\mathrm{vol})$$

Here $\Phi \mathbf{b}_0 \Delta t$ is the term which is calculated and added to array PI in routine LSTIFF. These terms are only for pore pressure degrees of freedom. $\mathbf{b}_0$ represents the excess pore pressure at the beginning of the *current* increment (these values are in array DA).

Routine LSTIFF

```
      SUBROUTINE LSTIFF(K,MUS,INXL,SG,KSG,DTIME,NN,NNOD1,NEL,NDF,NTPE,   STIF    1
     1 NIP,NPR,NMT,NS,NB,NL,NDIM,NDMX,NVRS,NPMX,LT,XYZ,DA,VARINT,P,      STIF    2
     2 D,ELCOD,DS,SHFN,CARTD,B,DB,SS,ELCODP,E,PE,RN,AA,ETE,RLT,          STIF
     3 NCONN,MAT,NW,NWL,NMOD,LL,PR,NTY)                                  STIF
C*********************************************************************STIF    5
C           CALCULATION AND ASSEMBLY OF STIFFNESS MATRIX              *STIF    6
C*********************************************************************STIF    7
      REAL L,LL                                                         STIF    8
      DIMENSION PERM(3)                                                 STIF    9
      DIMENSION SG(KSG),XYZ(NDIM,NN),DA(NDF),VARINT(NVRS,NIP,NEL),      STIF   10
     1 P(NDF),D(NS,NS),ELCOD(NDIM,NDMX),DS(NDIM,NDMX),                  STIF   11
     2 SHFN(NDMX),CARTD(NDIM,NDMX),B(NS,NB),DB(NS,NB),                  STIF   12
     3 SS(NB,NB),ELCODP(NDIM,NPMX),E(NDIM,NPMX),PE(NDIM,NPMX),          STIF   13
     4 RN(NB),AA(NPMX),ETE(NPMX,NPMX),RLT(NB,NPMX)                      STIF   14
      DIMENSION NCONN(NTPE,NEL),MAT(NEL),NW(NNOD1),                     STIF   15
     1 NWL(NPMX),NMOD(NIP,NEL),LL(NL),PR(NPR,NMT),NTY(NMT)              STIF   16
      COMMON /FLOW  / NPLAX                                             STIF   17
      COMMON /DATW  / W(100)                                            STIF   18
      COMMON /DATL  / L(4,100)                                          STIF   19
      COMMON /PARS  / PYI,ALAR,ASMVL,ZERO                               STIF   20
      COMMON /DEVICE/ IR1,IR4,IR5,IW2,IW4,IW6,IW7,IW8,IW9               STIF   21
      COMMON /ELINF / LINFO(50,15)                                      STIF   22
      COMMON /JACB  / XJACI(3,3),DJACB                                  STIF   23
```

```
C-----------------------------------------------------------------------STIF  24
      CR=1.0                                                           STIF  25
      IF(NPLAX.EQ.1)CR=2.0*PYI                                         STIF  26
C----------INITIALISE SS                                              STIF  27
      CALL ZEROR2(SS,NB,NB)                                            STIF  28
C                                                                      STIF  29
      NDN=LINFO(5,LT)                                                  STIF  30
      NPN=LINFO(6,LT)                                                  STIF  31
      NGP=LINFO(11,LT)                                                 STIF  32
      INDX=LINFO(12,LT)                                                STIF  33
      NAC=LINFO(15,LT)                                                 STIF  34
      NDV=NDIM*NDN                                                     STIF  35
      NDPT=LINFO(1,LT)                                                 STIF  36
      GOTO(1,1,2,1,2,1,2,1,2,1,2),LT                                   STIF  37
      WRITE(IW6,910)MUS,LT                                             STIF  38
  910 FORMAT(1X,7HELEMENT,I5,2X,18HIS OF UNKNOWN TYPE,I5,              STIF  39
     1 2X,16H(ROUTINE LSTIFF))                                         STIF  40
      STOP                                                             STIF  41
C                                                                      STIF  42
    1 ICPL=0                                                           STIF  43
      IBLK=1                                                           STIF  44
      NPN=0                                                            STIF  45
      GOTO 14                                                          STIF  46
    2 ICPL=1                                                           STIF  47
      IBLK=0                                                           STIF  48
C----------INITIALISE RLT AND ETE                                     STIF  49
      CALL ZEROR2(RLT,NB,NPMX)                                         STIF  50
      CALL ZEROR2(ETE,NPMX,NPMX)                                       STIF  51
C-----------------------------------------------------------------------STIF  52
C     SETUP LOCAL ARRAY OF NW AS NWL GIVING THE INDEX TO               STIF  53
C     PORE-PRESSURE VARIABLES                                          STIF  54
C-----------------------------------------------------------------------STIF  55
      IPP=0                                                            STIF  56
C----------INXL - INDEX TO NODAL D.O.F. (SEE ROUTINES BDATA1, MAXVAL)  STIF  57
      DO 12 IV=1,NDPT                                                  STIF  58
      IQ=LINFO(IV+INXL,LT)                                             STIF  59
      IF(IQ.EQ.NDIM)GO TO 12                                           STIF  60
      IPP=IPP+1                                                        STIF  61
      NDE=NCONN(IV,K)                                                  STIF  62
      NDE=IABS(NDE)                                                    STIF  63
C----------COORDINATES OF POREPRESSURE NODES OF ELEMENT                STIF  64
      DO 10 ID=1,NDIM                                                  STIF  65
   10 ELCODP(ID,IPP)=XYZ(ID,NDE)                                       STIF  66
      NWL(IPP)=NW(NDE)+IQ-1                                            STIF  67
   12 CONTINUE                                                         STIF  68
C                                                                      STIF  69
   14 KM=MAT(K)                                                        STIF  70
C-----------------------------------------------------------------------STIF  71
C     LOCAL ARRAY OF COORDINATES OF DISPLACEMENT NODES OF ELEMENT      STIF  72
C-----------------------------------------------------------------------STIF  73
      DO 20 KN=1,NDN                                                   STIF  74
      NDE=NCONN(KN,K)                                                  STIF  75
      NDE=IABS(NDE)                                                    STIF  76
C                                                                      STIF  77
      DO 20 ID=1,NDIM                                                  STIF  78
   20 ELCOD(ID,KN)=XYZ(ID,NDE)                                         STIF  79
CC    WRITE(IW6,801)ELCOD                                              STIF  80
CC801 FORMAT(/1X,5HELCOD/(1X,10F6.1))                                  STIF  81
C                                                                      STIF  82
      IF(NTY(KM)-2)26,28,28                                            STIF  83
C----------CONSTANT ELASTICITY D MATRIX                               STIF  84
   26 CALL DCON(K,IBLK,NEL,NDIM,NS,NPR,NMT,MAT,PR,D,BK)                STIF  85
C-----------------------------------------------------------------------STIF  86
C     ITERATE FOR ALL INTEGRATION POINTS                               STIF  87
C-----------------------------------------------------------------------STIF  88
   28 DO 80 IP=1,NGP                                                   STIF  89
```



```
      IPA=IP+INDX                                                      STIF  90
C                                                                      STIF  91
      DO 30 IL=1,NAC                                                   STIF  92
   30 LL(IL)=L(IL,IPA)                                                 STIF  93
C-----------------------------------------------------------------------STIF  94
C     FORM B MATRIX FOR CURRENT INTEGRATION POINT                      STIF  95
C-----------------------------------------------------------------------STIF  96
      ISTGE=3                                                          STIF  97
      CALL FORMB2(K,MUS,R,RI,NDIM,NDMX,NDN,NS,                         STIF  98
     1   NB,NAC,ELCOD,DS,SHFN,CARTD,B,LL,LT,IP,ISTGE)                  STIF  99
      F9=CR*DJACB*W(IPA)                                               STIF 100
C                                                                      STIF 101
      IF(ICPL.EQ.1)CALL JPC(MUS,NDIM,NPN,NS,NB,NAC,                    STIF 102
     1   DS,CARTD,B,ELCODP,E,RN,AA,LL,LT,IP,ISTGE)                     STIF 103
      IF(NPLAX.EQ.1)F9=F9*R                                            STIF 104
      KGO=NTY(KM)                                                      STIF 105
      GO TO(39,32,33,34),KGO                                           STIF 106
      WRITE(IW6,900)MUS,KGO                                            STIF 107
  900 FORMAT(1X,7HELEMENT,I5,2X,27HIS OF UNKNOWN MATERIAL TYPE,I5,     STIF 108
     1 16H(ROUTINE LSTIFF))                                            STIF 109
      STOP                                                             STIF 110
C-----------------------------------------------------------------------STIF 111
C     D MATRIX                                                         STIF 112
C-----------------------------------------------------------------------STIF 113
   32 CALL DLIN(IP,K,IBLK,NEL,NDIM,NDN,NS,NPR,NMT,                     STIF 114
     1   ELCOD,SHFN,MAT,D,PR,INDX,BK)                                  STIF 115
      GO TO 39                                                         STIF 116
   33 CALL DMCAM(IP,K,IBLK,NEL,NIP,NVRS,NDIM,NS,NPR,NMT,               STIF 117
     1   VARINT,MAT,D,PR,BK)                                           STIF 118
      GO TO 39                                                         STIF 119
   34 CALL DCAM(IP,K,IBLK,NEL,NIP,NVRS,NDIM,NS,NPR,NMT,                STIF 120
     1   VARINT,MAT,D,PR,ITP,BK)                                       STIF 121
      GO TO 39                                                         STIF 122
C-----------------------------------------------------------------------STIF 123
C     FORM D*B AND B*D*B                                               STIF 124
C-----------------------------------------------------------------------STIF 125
   39 CALL LSTIFA(SS,B,D,DB,F9,NS,NB)                                  STIF 126
C-----------------------------------------------------------------------STIF 127
C     BYPASS IF NOT COUPLED CONSOLIDATION                              STIF 128
C-----------------------------------------------------------------------STIF 129
      IF(ICPL.EQ.0)GO TO 80                                            STIF 130
C-----------------------------------------------------------------------STIF 131
C     FORM PERM*E                                                      STIF 132
C-----------------------------------------------------------------------STIF 133
      PERM(1)=PR(9,KM)                                                 STIF 134
      PERM(2)=PR(10,KM)                                                STIF 135
      PERM(3)=PERM(1)                                                  STIF 136
      GAMMAW=PR(7,KM)                                                  STIF 137
C                                                                      STIF 138
      DO 40 JJ=1,NPN                                                   STIF 139
      DO 40 IM=1,NDIM                                                  STIF 140
      PE(IM,JJ)=PERM(IM)*E(IM,JJ)                                      STIF 141
   40 CONTINUE                                                         STIF 142
C-----------------------------------------------------------------------STIF 143
C     FORM ET*PERM*E                                                   STIF 144
C-----------------------------------------------------------------------STIF 145
      DO 50 II=1,NPN                                                   STIF 146
      DO 50 JJ=1,NPN                                                   STIF 147
      DO 50 KK=1,NDIM                                                  STIF 148
   50 ETE(II,JJ)=ETE(II,JJ)+E(KK,II)*PE(KK,JJ)*DTIME*F9/GAMMAW         STIF 149
C-----------------------------------------------------------------------STIF 150
C     FORM LT                                                          STIF 151
C-----------------------------------------------------------------------STIF 152
      DO 60 II=1,NDV                                                   STIF 153
      DO 60 JJ=1,NPN                                                   STIF 154
   60 RLT(II,JJ)=RLT(II,JJ)+RN(II)*AA(JJ)*F9                           STIF 155
```

```
C-----------------------------------------------------------------------STIF 156
C     END OF INTEGRATION POINT LOOP                                     STIF 157
C-----------------------------------------------------------------------STIF 158
   80 CONTINUE                                                          STIF 159
C-----------------------------------------------------------------------STIF 160
C     CALCULATE LOWER HALF OF STIFFNESS MATRIX USING SYMMETRY           STIF 161
C-----------------------------------------------------------------------STIF 162
      DO 82 JJ=2,NB                                                     STIF 163
      JJM1=JJ-1                                                         STIF 164
C                                                                       STIF 165
      DO 82 II=1,JJM1                                                   STIF 166
   82 SS(JJ,II)=SS(II,JJ)                                               STIF 167
C-----------------------------------------------------------------------STIF 168
C     FORM STIFFNESS MATRIX SG FROM SS, RLT AND ETE                     STIF 169
C-----------------------------------------------------------------------STIF 170
      CALL LSTFSG(SG,KSG,NDF,NB,NDIM,NDMX,NPMX,DA,P,SS,ETE,             STIF 171
     1   RLT,NWL,NPN,NDN,LT,ICPL)                                       STIF 172
C                                                                       STIF 173
      NR=LINFO(16,LT)                                                   STIF 174
      NT=NR*(NR+1)/2                                                    STIF 175
CC    CALL PRINTF(IW6,SG,NT,NR,P,NDF,1)                                 STIF 176
      RETURN                                                            STIF 177
      END                                                               STIF 178
```

STIF 28 : zero array SS.
STIF 30–36 : set up data (parameters) dependent on element type.
STIF 37 : branch off, depending on whether element is *consolidation* type or not.

STIF 43–45 : *drained/undrained* element. Set IBLK = 1 to indicate bulk modulus of water is to be added to **D** matrix.

47–68 only for consolidation elements.

STIF 47–48 : *consolidation* element. Set ICPL = 1 and IBLK = 0 to indicate no changes to the **D** matrix.
STIF 50–51 : zero arrays RLT (link matrix) and ETE (flow matrix).
STIF 58 : loop on all nodes of element.
STIF 61 : counter of number of pore pressure nodes (and variables). It is also the index to array NWL.
STIF 65–66 : array ELCODP contains the co-ordinates of nodes of element with pore pressure variable.
STIF 67 : array NWL contains the index to NCONN of pore pressure variable.
STIF 74 : loop on all displacement nodes.
STIF 75–79 : array ELCOD contains the co-ordinates of displacement nodes of element.
STIF 85 : calculate **D** matrix for elastic model – only for the one which is independent of current stress state or geometry; therefore it is calculated only once outside the integration point loop.
STIF 89 : loop on all integration points.
STIF 92–93 : obtain integration point co-ordinates from array L and set up LL.

STIF 98–99 : calculate **B** matrix.
STIF 102–103 : calculate **E**, $\bar{\mathbf{N}}$ and $\mathbf{m}^T\mathbf{B}$ stored in E, AA and RN.
STIF 106 : branch off for different material types.
(1 – linear elastic, 2 – non-homogeneous elastic,)
(3 – modified Cam-clay, 4 – Cam-clay).
STIF 114–115 : calculate **D** matrix for elastic model (non-homogeneous).
STIF 117–118 : calculate **D** matrix for modified Cam-clay.
STIF 120–121 : calculate **D** matrix for Cam-clay.
STIF 126 : calculate $\mathbf{B}^T\mathbf{D}\mathbf{B}$. d (vol) and accumulate in SS.
STIF 130 : branch off, if not consolidation element.

134–155 for consolidation elements only.

STIF 134–136 : obtain permeabilities in x, y and z directions.
STIF 137 : unit weight of water.
STIF 139–142 : calculate matrix **kE** as PE.
STIF 146–149 : calculate $\int \mathbf{E}^T \frac{\mathbf{k}}{\gamma_w} \mathbf{E}$ d (vol) Δt as ETE.
STIF 153–155 : calculate $\mathbf{L}^T$ and place it in RLT.
STIF 163–167 : calculate lower triangle of stiffness matrix using symmetry.
STIF 171–172 : form stiffness matrix SG from SS, RLT and ETE.
STIF 174–176 : print out SG in triangular form for debugging.

8.7 CONSOLIDATION COMPONENT OF STIFFNESS MATRIX

8.7.1 Flow matrix

Routine **JPC** calculates the additional components that make up the extended element stiffness matrix.

Routine JPC

```
      SUBROUTINE JPC(J,NDIM,NPN,NS,NB,NAC,                              JPC    1
     1 DS,CARTD,B,ELCODP,E,RN,AA,LL,LT,IP,ISTGE)                        JPC    2
C*********************************************************************JPC    3
C     CALCULATES SHAPE FUNCTIONS AND DERIVATIVES                        JPC    4
C     FOR EXCESS PORE PRESSURE VARIATION                                JPC    5
C*********************************************************************JPC    6
      REAL LL                                                           JPC    7
      DIMENSION DS(NDIM,NPN),CARTD(NDIM,NPN),B(NS,NB),                  JPC    8
     1 ELCODP(NDIM,NPN),E(NDIM,NPN),RN(NB),AA(NPN),LL(NAC)              JPC    9
      COMMON /FLOW  / NPLAX                                             JPC   10
      COMMON /PARS  / PYI,ALAR,ASMVL,ZERO                               JPC   11
C                                                                       JPC   12
      CALL FORMP(J,NDIM,NPN,NAC,DS,AA,CARTD,                            JPC   13
     1   ELCODP,LL,LT,IP,ISTGE)                                         JPC   14
C-----------------------------------------------------------------------JPC   15
C     FORM RN                                                           JPC   16
C-----------------------------------------------------------------------JPC   17
      NCOM=NDIM                                                         JPC   18
      IF(NPLAX.EQ.1.AND.NCOM.EQ.2)NCOM=NDIM+1                           JPC   19
C                                                                       JPC   20
      DO 30 IB=1,NB                                                     JPC   21
      SUM=ZERO                                                          JPC   22
```

```
C                                                                       JPC   23
      DO 20 ID=1,NCOM                                                   JPC   24
   20 SUM=SUM+B(ID,IB)                                                  JPC   25
   30 RN(IB)=SUM                                                        JPC   26
C-----------------------------------------------------------------------JPC   27
C     FORM E                                                            JPC   28
C-----------------------------------------------------------------------JPC   29
      DO 50 IN=1,NPN                                                    JPC   30
      DO 50 ID=1,NDIM                                                   JPC   31
   50 E(ID,IN)=CARTD(ID,IN)                                             JPC   32
      RETURN                                                            JPC   33
      END                                                               JPC   34
```

JPC 13–14 : calculate shape functions ($\bar{N}_i$) and derivatives ($\partial\bar{N}_i/\partial\xi, \partial\bar{N}_i/\partial\eta$) for pore pressure variations (arrays AA, DS). Derivatives w.r.t. Cartesian co-ordinates ($\partial\bar{N}_i/\partial x$, $\partial\bar{N}_i/\partial y$) (array CARTD).

JPC 21–26 : calculate array RN ($= \mathbf{B}^{\mathrm{T}}\mathbf{m}$, where $\mathbf{B}$ is the strain–displacement matrix and $\mathbf{m}^{\mathrm{T}} = [1, 1, 1, 0]$).

JPC 30–32 : calculate $\mathbf{E}$ matrix. ($E_i = \partial\bar{N}_i/\partial x, \partial\bar{N}_i/\partial y$.)

Routine **FORMP** calculates the Cartesian derivatives of pore pressure shape functions. These are then used in the calculation of the E matrix by routine JPC.

Routine FORMP

```
      SUBROUTINE FORMP(J,NDIM,NPN,NAC,DS,SFP,CARTD,                     FRMP   1
     1 ELCODP,LL,LT,IP,ISTGE)                                           FRMP   2
C***********************************************************************FRMP   3
C     FORMS  CARTD  MATRIX  FOR  AREA  COORDS  LL(NAC)                  FRMP   4
C     IN  TRIANGLE  J FOR INTEGRATION POINT IP                          FRMP   5
C***********************************************************************FRMP   6
                                                                        FRMP   7
      REAL LL                                                           FRMP   8
      DIMENSION LL(NAC)                                                 FRMP   9
      DIMENSION DS(NDIM,NPN),SFP(NPN),CARTD(NDIM,NPN),                  FRMP  10
     1 ELCODP(NDIM,NPN),XJACM(3,3)                                      FRMP  11
      COMMON /DEVICE/ IR1,IR4,IR5,IW2,IW4,IW6,IW7,IW8,IW9               FRMP  12
      COMMON /PARS  / PYI,ALAR,ASMVL,ZERO                               FRMP  13
      COMMON /JACB  / XJACI(3,3),DJACB                                  FRMP  14
C-----------------------------------------------------------------------FRMP  15
C     CALCULATE SHAPE FUNCTION AND DERIVATIVES (LOCAL COORDS)           FRMP  16
C-----------------------------------------------------------------------FRMP  17
      CALL SHFNPP(IW6,LL,NAC,DS,SFP,NDIM,NPN,LT,1,J)                    FRMP  18
C                                                                       FRMP  19
CC    WRITE(IW6,902)DJACB                                               FRMP  20
C                                                                       FRMP  21
      DO 35 IN=1,NPN                                                    FRMP  22
      DO 35 ID=1,NDIM                                                   FRMP  23
      SUM=ZERO                                                          FRMP  24
C                                                                       FRMP  25
      DO 30 JD=1,NDIM                                                   FRMP  26
   30 SUM=SUM-DS(JD,IN)*XJACI(ID,JD)                                    FRMP  27
   35 CARTD(ID,IN)=SUM                                                  FRMP  28
      RETURN                                                            FRMP  29
CC902 FORMAT(9H JACOBIAN,2X,E16.5)                                      FRMP  30
      END
```

FRMP 17 : calculate shape functions and derivatives w.r.t. local co-ordinate for pore pressure variation.

FRMP 21–27 : calculate Cartesian derivatives of shape functions $\partial\bar{N}_i/\partial x$, $\partial\bar{N}_i/\partial y$.

Routine **SHFNPP** calculates the pore pressure shape functions and the derivatives with respect to the local co-ordinates.

Routine SHFNPP

```
      SUBROUTINE SHFNPP(IW6,LL,NAC,DS,SFP,NDIM,NPN,LT,IFL,MUS)          SHPP   1
C***********************************************************************SHPP   2
C     SHAPE FUNCTIONS AND DERIVATIVES FOR PORE PRESSURE VARIATION       SHPP   3
C***********************************************************************SHPP   4
      REAL LL,L1,L2,L3,L4                                               SHPP   5
      DIMENSION SFP(NPN),DS(NDIM,NPN),LL(NAC)                           SHPP   6
C                                                                       SHPP   7
      L1=LL(1)                                                          SHPP   8
      L2=LL(2)                                                          SHPP   9
      IF(NAC.LT.3)GOTO 10                                               SHPP  10
      L3=LL(3)                                                          SHPP  11
      IF(NAC.LT.4)GOTO 10                                               SHPP  12
      L4=LL(4)                                                          SHPP  13
C                                                                       SHPP  14
   10 GOTO(80,80,13,80,25,80,37,80,49,80,71),LT                         SHPP  15
      WRITE(IW6,900)MUS,LT                                              SHPP  16
  900 FORMAT(/1X,7HELEMENT,I5,2X,22HIS OF UNKNOWN TYPE ***,I5,2X,       SHPP  17
     1 16H(ROUTINE SHFNPP))                                             SHPP  18
      STOP                                                              SHPP  19
C-----------------------------------------------------------------------SHPP  20
C     LINEAR STRAIN TRIANGLE                                            SHPP  21
C-----------------------------------------------------------------------SHPP  22
   13 IF(IFL.EQ.0)GO TO 23                                              SHPP  23
      DS(1,1)=1.                                                        SHPP  24
      DS(1,2)=0.                                                        SHPP  25
      DS(1,3)=-1.                                                       SHPP  26
      DS(2,1)=0.                                                        SHPP  27
      DS(2,2)=1.                                                        SHPP  28
      DS(2,3)=-1.                                                       SHPP  29
C                                                                       SHPP  30
   23 SFP(1)=L1                                                         SHPP  31
      SFP(2)=L2                                                         SHPP  32
      SFP(3)=L3                                                         SHPP  33
      RETURN                                                            SHPP  34
C-----------------------------------------------------------------------SHPP  35
C     QUADRILATERAL ELEMENT                                             SHPP  36
C-----------------------------------------------------------------------SHPP  37
   25 CONTINUE                                                          SHPP  38
      WRITE(IW6,910)MUS,LT                                              SHPP  39
  910 FORMAT(/1X,7HELEMENT,I5,2X,14HIS OF TYPE ***,I5,2X,               SHPP  40
     1 32HNOT IMPLEMENTED (ROUTINE SHFNPP))                             SHPP  41
      RETURN                                                            SHPP  42
C-----------------------------------------------------------------------SHPP  43
C     CUBIC VARIATION IN PORE-PRESSURE                                  SHPP  44
C-----------------------------------------------------------------------SHPP  45
   37 C1=9./2.                                                          SHPP  46
      C2=27./2.                                                         SHPP  47
      C3=27.                                                            SHPP  48
      T11=L1-1./3.                                                      SHPP  49
      T12=L1-2./3.                                                      SHPP  50
      T21=L2-1./3.                                                      SHPP  51
      T22=L2-2./3.                                                      SHPP  52
      T31=L3-1./3.                                                      SHPP  53
      T32=L3-2./3.                                                      SHPP  54
      IF(IFL.EQ.0)GO TO 40                                              SHPP  55
```

```
C
      DS(1,1)=C1*(T11*T12+L1*(T11+T12))                                 SHPP  56
      DS(1,2)=0.                                                        SHPP  57
      DS(1,3)=-C1*(T31*T32+L3*(T31+T32))                                SHPP  58
      DS(1,4)=C2*L2*(L1+T11)                                            SHPP  59
      DS(1,5)=C2*L2*T21                                                 SHPP  60
      DS(1,6)=-C2*L2*T21                                                SHPP  61
      DS(1,7)=-C2*L2*(L3+T31)                                           SHPP  62
      DS(1,8)=C2*L3*T31-C2*L1*(L3+T31)                                  SHPP  63
      DS(1,9)=C2*L3*(L1+T11)-C2*L1*T11                                  SHPP  64
      DS(1,10)=C3*L2*L3-C3*L2*L1                                        SHPP  65
C                                                                       SHPP  66
      DS(2,1)=0.                                                        SHPP  67
      DS(2,2)=C1*(T21*T22+L2*(T21+T22))                                 SHPP  68
      DS(2,3)=-C1*(T31*T32+L3*(T31+T32))                                SHPP  69
      DS(2,4)=C2*L1*T11                                                 SHPP  70
      DS(2,5)=C2*L1*(L2+T21)                                            SHPP  71
      DS(2,6)=C2*L3*(L2+T21)-C2*L2*T21                                  SHPP  72
      DS(2,7)=C2*L3*T31-C2*L2*(L3+T31)                                  SHPP  73
      DS(2,8)=-C2*L1*(L3+T31)                                           SHPP  74
      DS(2,9)=-C2*L1*T11                                                SHPP  75
      DS(2,10)=C3*L1*L3-C3*L1*L2                                        SHPP  76
C                                                                       SHPP  77
   40 SFP(1) =C1*L1*T11*T12                                             SHPP  78
      SFP(2) =C1*L2*T21*T22                                             SHPP  79
      SFP(3) =C1*L3*T31*T32                                             SHPP  80
      SFP(4) =C2*L1*L2*T11                                              SHPP  81
      SFP(5) =C2*L1*L2*T21                                              SHPP  82
      SFP(6) =C2*L2*L3*T21                                              SHPP  83
      SFP(7) =C2*L2*L3*T31                                              SHPP  84
      SFP(8) =C2*L1*L3*T31                                              SHPP  85
      SFP(9) =C2*L1*L3*T11                                              SHPP  86
      SFP(10)=C3*L1*L2*L3                                               SHPP  87
      RETURN                                                            SHPP  88
C-----------------------------------------------------------------------SHPP  89
C     BRICK ELEMENT                                                     SHPP  90
C-----------------------------------------------------------------------SHPP  91
   49 CONTINUE                                                          SHPP  92
      WRITE(IW6,910)MUS,LT                                              SHPP  93
      RETURN                                                            SHPP  94
C-----------------------------------------------------------------------SHPP  95
C     TETRA-HEDRA ELEMENT                                               SHPP  96
C-----------------------------------------------------------------------SHPP  97
   71 CONTINUE                                                          SHPP  98
      WRITE(IW6,910)MUS,LT                                              SHPP  99
   80 RETURN                                                            SHPP 100
      END                                                               SHPP 101
                                                                        SHPP 102
```

SHPP 8–13 : set up L1, L2, etc. equal to integration point co-ordinates.
SHPP 15 : branch off for different element types.
SHPP 24–29 : calculate derivatives w.r.t. local co-ordinates for linear strain triangle (LT = 3) – $\partial\bar{N}_i/\partial\xi$, $\partial\bar{N}_i/\partial\eta$.
SHPP 31–33 : calculate shape functions – $\bar{N}_i$ – for linear strain triangle.
SHPP 38 : calculate shape functions and derivatives for quadrilateral element (not implemented here).
SHPP 46–54 : calculate some constants.
SHPP 55 : if IFL = 0, only calculate the shape functions for LST element.
SHPP 57–77 : calculate derivatives w.r.t. local co-ordinates for cubic strain triangle (LT = 7) – $\partial\bar{N}_i/\partial\xi$, $\partial\bar{N}_i/\partial\eta$.
SHPP 79–88 : calculate shape functions for cubic strain triangle (LT = 7) – $\bar{N}_i$.

SHPP 93 : shape functions and derivatives for brick element (not implemented here).
SHPP 99 : shape functions and derivatives for tetrahedra element (not implemented here).

Routine **LSTIFA** is called to calculate $\int_V \mathbf{B}^T\mathbf{DB}\ \mathrm{d(vol)}$ from **B** and **D**.

Routine LSTIFA

```
      SUBROUTINE LSTIFA(SS,B,D,DB,F9,NS,NB)                             LSTA   1
C*********************************************************************LSTA   2
C     ROUTINE TO CALCULATE D*B AND BT*D*B                               LSTA   3
C     FOR EACH INTEGRATION POINT                                        LSTA   4
C*********************************************************************LSTA   5
      DIMENSION SS(NB,NB),D(NS,NS),DB(NS,NB),B(NS,NB)                   LSTA   6
C-----------------------------------------------------------------------LSTA   7
C     FORM D*B                                                          LSTA   8
C-----------------------------------------------------------------------LSTA   9
      CALL ZEROR2(DB,NS,NB)                                             LSTA  10
C                                                                       LSTA  11
      DO 20 JJ=1,NB                                                     LSTA  12
      DO 20 II=1,NS                                                     LSTA  13
      DO 20 KK=1,NS                                                     LSTA  14
   20 DB(II,JJ)=DB(II,JJ)+D(II,KK)*B(KK,JJ)                             LSTA  15
C-----------------------------------------------------------------------LSTA  16
C     FORM BT*D*B                                                       LSTA  17
C-----------------------------------------------------------------------LSTA  18
      DO 30 JJ=1,NB                                                     LSTA  19
      DO 30 II=1,JJ                                                     LSTA  20
      DO 30 KK=1,NS                                                     LSTA  21
   30 SS(II,JJ)=SS(II,JJ)+DB(KK,JJ)*B(KK,II)*F9                         LSTA  22
      RETURN                                                            LSTA  23
      END                                                               LSTA  24
```

LSTA 10 : zero array DB.
LSTA 12–15 : calculate **(DB)** matrix.
LSTA 19–22 : calculate $\mathbf{B}^T\mathbf{(DB)}$ for current integration point and add it to $\mathrm{SS} = \Sigma\ \mathbf{B}^T\mathbf{(DB)}$.

For consolidation analysis the extended element stiffness matrix is calculated from three different matrices (see section 8.8). The latter part of routine LSTIFF does this operation, and the technique used is described in the next section.

8.8 USE OF INDEXES IN STIFFNESS CALCULATIONS

Chapter 6 described the use of indexes for different element types with a single array partitioned into a number of regions (each catering for a different element type). Then only the **starting** index was necessary and the generality of the program was retained.

The other area where this index system is heavily relied on is the formation of the element stiffness matrix SG in routine LSTIFF (the array is known as ES in

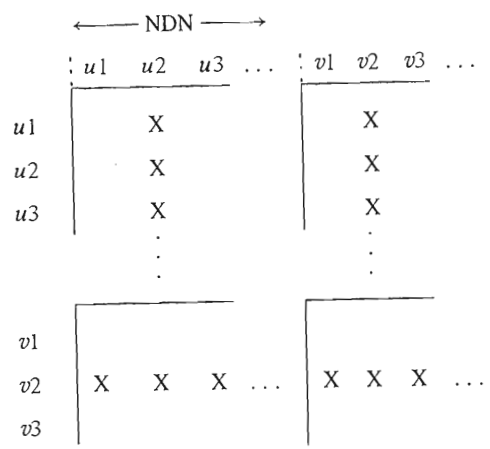
routine FRONTZ). As in the case of the global stiffness matrix, all variables (d.o.f.) of a node are placed together in matrix SG.

For drained/undrained analysis the stiffness matrix for each node is made up of sub-matrices which are or order 2 X 2. The two components are for the two directions (x and y for plane strain, r and z for axisymmetric problems). Therefore for a six-noded element (LT = 2), the element stiffness matrix is of order 12 X 12. The array SS then completely defines the element stiffness matrix.

$$\mathrm{SS} = \int_V \mathbf{B}^{\mathrm{T}} \mathbf{D}\, \mathbf{B}\, \mathrm{d}\,(\mathrm{vol}). \tag{8.6}$$

The **B** matrix is defined such that the x components of **a** for all nodes are placed together (followed by the y components of **a**, where d_x and d_y are the displacements (denoted by u and v in Fig. 8.2) in the x and y directions respectively. Hence when the displacement stiffness matrix SS is calculated, it has the same structure.

The matrix SS has NDN X NDIM number of rows/columns, where NDIM = 2 for two-dimensional problems. This is the number of displacement variables in the elements and is equal to the no. of rows/columns in SS.



(8.7)

Fig. 8.2 – Matrix SS

For a LST element (LT = 2, NDN = 6), $u2$ occupies the 2nd row/column in matrix SS and $v2$ occupies the 2 + NDN = 8th row/column. In SG they occupy consecutive rows/columns. Therefore when SG is formed from SS the rows/columns have to be interchanged. Remembering that SS is a two-dimensional array and SG is a one-dimensional array storing the upper triangular stiffness matrix columnwise, forming SG is not straightforward.

This part of the program also has to be capable of dealing with different element types. The simplest programming technique is to set up a pointer array which gives the information of which row/column of array SS goes into which row/column of array SG. Again these pointers are different for different element types. For example, the first 15 indexes are for two-dimensional 'non-consolidation' elements – element types 2, 6. For element type 6 (the 15-noded cubic strain triangle), all 15 indexes are relevant. The number in brackets next to each element type (in routine LSTIFF) is the number of indexes that are relevant. For element type 7 the indexes are given by KD(50)–KD(64).

To complete the details, one needs the reference point (the starting index – 1, i.e. 49 for element type 7), and this is provided by array NXD. Therefore NXD(7) = 49. The relevant equations for a consolidation analysis were given in section 8.7, using the following notations:

$$\begin{aligned} \mathbf{K} &= \mathrm{SS}, \\ \mathbf{L} &= \mathrm{RLT}, \\ \Phi\Delta t &= \mathrm{ETE}. \end{aligned}$$

In a consolidation analysis, if a node as 3 d.o.f. (d_x, d_y, $\bar{u}$) then its nodal stiffness is a 3 X 3 sub-matrix. It consists of components from arrays SS, RLT and ETE as follows:

$$\begin{bmatrix} \mathrm{SS}^{\dagger} & \mathrm{SS}^{\dagger} & \mathrm{RLT}^{\dagger} \\ \mathrm{SS} & \mathrm{SS}^{\dagger} & \mathrm{RLT}^{\dagger} \\ \mathrm{RLT} & \mathrm{RLT} & \mathrm{ETE}^{\dagger} \end{bmatrix},$$

The rows/columns of array SS represent the displacement stiffness terms. The array RLT is a coupling matrix, linking displacement loads to the pore pressure variables. Array ETE contains pore pressure stiffness terms. Now it is necessary to form the matrix SG from the three matrices SS, RLT and ETE such that all variables of a node are placed together.

Even though forming the SG array from arrays SS, RLT and ETE may appear complicated at first sight, it is straightforward when one uses the index system. The number of rows in RLT corresponds to the number of displacement d.o.f., and therefore array KD can again be used to point out which row of RLT should go into which row of SG. Similarly the number of columns of array RLT represents the pore pressure d.o.f. The number of rows/columns of ETE is equal to the number of pore pressure d.o.f. The array KP is set in similar lines to array KD. It gives the information regarding which row/column of ETE should go into which row/column of SG. The same indexes apply to the columns of RLT.

Arrays KD and KP give different indexes for different element types. They are set up exactly in the same manner as arrays W(100) and L(4,100) in routine BDATA1. The only difference is that the starting indexes are provided by two

† Only upper triangular terms need to be considered because of symmetry.

local arrays – NXD for displacement variables and NXP for pore pressure variables. The reason these are local arrays instead of global arrays is that this is the only routine where this information is needed. On the other hand, arrays W and L are used in many different parts of the program and serve a global requirement.

KP(1) . . . . . KP(3) are for LT = 3 LST for 3 p.p. d.o.f.

KP(8) . . . . . KP(17) are for LT = 7 CuST for 10 p.p. d.o.f.

For example, for element type 3 KP(1) = 3. Then row/column 1 of array ETE will take up row/column 3 in SG. Similarly KP(2) = 6. Then row/column 2 of array ETE will take up row/column 6 of array SG.

ETE(1, 1) must be placed in SG(3,3) [SG(6)]

ETE(2, 2) must be placed in SG(6,6) [SG(21)]

ETE(1, 2) must be placed in SG(3,6) [SG(18)]

ETE(2, 1) must be placed in SG(6,3) †

In the above, SG gives the row and column number respectively. However, since array SG is an upper triangular matrix which is stored columnwise, the value within [SG( )] gives the actual position in array SG (this is calculated by the program as NIA + NCN in DO loops terminating on labels 140 and 160.

The arrays E, RN, AA, ETE and RLT have been set up for the maximum requirement of element types in the mesh. The sizes are based on the following parameters: NDMX × NDIM (where NDMX is the maximum number of displacement nodes) and NPMX (the maximum number of pore pressure d.o.f.) of all the elements in the mesh.

element type 3:	ND2 = 2 × 6 = 12;	NPP = 3
E(3,3) RN(12)	AA(3) ETE(3,3)	RLT(12,3)
element type 7:	ND2 = 2 × 15 = 30;	NPP = 10
E(3,10) RN(30)	AA(10) ETE(10,10)	RLT(30,10)

The number of rows in array E is set equal to 3 for the three directions respectively (general three-dimensional formulation) under all circumstances.

Remembering that SG is only an upper triangular matrix then only the upper triangular parts of matrices SS and ETE (including the diagonals) are used. However, the whole of RLT is needed. The terms from arrays SS and ETE are entered in SG, column by column, up to the diagonal term.

Routine LSTFSG

```
      SUBROUTINE LSTFSG(SG,KSG,NDF,NB,NDIM,NDMX,NPMX,DA,P,SS,ETE,      FMSG   1
     1 RLT,NWL,NPN,NDN,LT,ICPL)                                        FMSG   2
C*****************************************************************FMSG   3
```

† Since this is in lower triangular matrix it is not considered (because of symmetry, i.e. ETE(2,1) = ETE(1,2).


```
C     FORM ELEMENT STIFFNESS MATRIX SG FROM SS, RLT AND ETE             FMSG   4
C*****************************************************************FMSG   5
      DIMENSION KP(29),KD(94),NXP(15),NXD(15)                           FMSG   6
      DIMENSION SG(KSG),DA(NDF),P(NDF),SS(NB,NB),ETE(NPMX,NPMX),        FMSG   7
     1 RLT(NB,NPMX),NWL(NPMX)                                           FMSG   8
      COMMON /PARS/ PYI,ALAR,ASMVL,ZERO                                 FMSG   9
C-----------------------------------------------------------------FMSG  10
C     INDEX TO ROWS/COLUMNS OF SG FOR ROWS/COLUMNS OF ETE               FMSG  11
C     INDEX TO COLUMNS OF SG FOE COLUMNS OF RLT (FOR CONSOLIDATION)     FMSG  12
C-----------------------------------------------------------------FMSG  13
C----------ELEMENT TYPE 3 - LST-------------------------------------FMSG  14
      DATA KP(1),KP(2),KP(3)/                                           FMSG  15
     1     3,6,9/                                                       FMSG  16
C----------ELEMENT TYPE 5 - QUADRILATERAL---------------------------FMSG  17
      DATA KP(4),KP(5),KP(6),KP(7)/                                     FMSG  18
     1     3,6,9,12/                                                    FMSG  19
C----------ELEMENT TYPE 7 - CUST------------------------------------FMSG  20
      DATA KP(8),KP(9),KP(10),KP(11),KP(12),KP(13),KP(14),KP(15),       FMSG  21
     1     KP(16),KP(17)/                                               FMSG  22
     2     3,6,9,34,35,36,37,38,39,40/                                  FMSG  23
C----------ELEMENT TYPE 9 - BRICK-----------------------------------FMSG  24
      DATA KP(18),KP(19),KP(20),KP(21),KP(22),KP(23),KP(24),KP(25)/     FMSG  25
     2     4,8,12,16,20,24,28,32/                                       FMSG  26
C----------ELEMENT TYPE 11 - TETRA-HEDRA----------------------------FMSG  27
      DATA KP(26),KP(27),KP(28),KP(29)/                                 FMSG  28
     1     4,8,12,16/                                                   FMSG  29
C-----------------------------------------------------------------FMSG  30
C     INDEX TO FIRST DISPLACEMENT VARIABLE OF EACH NODE IN SG           FMSG  31
C     INDEX TO ROWS/COLUMNS OF SG FROM ROWS/COLUMNS OF SS               FMSG  32
C     INDEX TO ROWS OF SG FOR ROWS OF RLT (FOR CONSOLIDATION ELEMENT)   FMSG  33
C-----------------------------------------------------------------FMSG  34
C----------ELEMENT TYPE 1(2), 2(6), 4(8), 6(15)---------------------FMSG  35
      DATA KD(1),KD(2),KD(3),KD(4),KD(5),KD(6),KD(7),KD(8),KD(9),KD(10),FMSG  36
     2     KD(11),KD(12),KD(13),KD(14),KD(15)/                          FMSG  37
     3     1,3,5,7,9,11,13,15,17,19,21,23,25,27,29/                     FMSG  38
C----------ELEMENT TYPE 8(20), 10(10)-------------------------------FMSG  39
      DATA KD(16),KD(17),KD(18),KD(19),KD(20),KD(21),KD(22),KD(23),     FMSG  40
     1     KD(24),KD(25),KD(26),KD(27),KD(28),KD(29),KD(30),KD(31),     FMSG  41
     2     KD(32),KD(33),KD(34),KD(35)/                                 FMSG  42
     3     1,4,7,10,13,16,19,22,25,28,31,34,37,40,43,46,49,52,55,58/    FMSG  43
C----------ELEMENT TYPE 3(6)----------------------------------------FMSG  44
      DATA KD(36),KD(37),KD(38),KD(39),KD(40),KD(41)/                   FMSG  45
     1     1,4,7,10,12,14/                                              FMSG  46
C----------ELEMENT TYPE 5(8)----------------------------------------FMSG  47
      DATA KD(42),KD(43),KD(44),KD(45),KD(46),KD(47),KD(48),KD(49)/     FMSG  48
     1     1,4,7,10,13,15,17,19/                                        FMSG   4
C----------ELEMENT TYPE 7(15)---------------------------------------FMSG  50
      DATA KD(50),KD(51),KD(52),KD(53),KD(54),KD(55),KD(56),KD(57),     FMSG  51
     1     KD(58),KD(59),KD(60),KD(61),KD(62),KD(63),KD(64)/            FMSG  52
     2     1,4,7,10,12,14,16,18,20,22,24,26,28,30,32/                   FMSG  53
C----------ELEMENT TYPE 9(20)---------------------------------------FMSG  54
      DATA KD(65),KD(66),KD(67),KD(68),KD(69),KD(70),KD(71),KD(72),     FMSG  55
     1     KD(73),KD(74),KD(75),KD(76),KD(77),KD(78),KD(79),KD(80),     FMSG  56
     2     KD(81),KD(82),KD(83),KD(84)/                                 FMSG  57
     3     1,5,9,13,17,21,25,29,33,36,39,42,45,48,51,54,57,60,63,66/    FMSG  58
C----------ELEMENT TYPE 11(10)--------------------------------------FMSG  59
      DATA KD(85),KD(86),KD(87),KD(88),KD(89),                          FMSG  60
     1     KD(90),KD(91),KD(92),KD(93),KD(94)/                          FMSG  61
     2     1,5,9,13,17,20,23,26,29,32/                                  FMSG  62
C-----------------------------------------------------------------FMSG  63
C     NXP AND NXD GIVE STARTING INDEX TO ARRAYS KP AND KD               FMSG  64
C     RESPECTIVELY FOR DIFFERENT ELEMENT TYPES                          FMSG  65
C-----------------------------------------------------------------FMSG  66
      DATA NXP(1),NXP(2),NXP(3),NXP(4),NXP(5),NXP(6),NXP(7),            FMSG  67
     1     NXP(8),NXP(9),NXP(10),NXP(11)/                               FMSG  68
     2     0,0,0,0,3,0,7,0,17,0,25/                                     FMSG  69
```

```
      DATA NXD(1),NXD(2),NXD(3),NXD(4),NXD(5),NXD(6),NXD(7),          FMSG  70
     1     NXD(8),NXD(9),NXD(10),NXD(11)/                            FMSG  71
     2     0,0,35,0,41,0,49,15,64,15,84/                             FMSG  72
C-----------------------------------------------------------------------FMSG  73
C---------- SIZE OF ARRAYS KP AND KD-----------------------------------FMSG  74
      NKP=29                                                          FMSG  75
      NKD=94                                                          FMSG  76
C-----------------------------------------------------------------------FMSG  77
      INXD=NXD(LT)                                                    FMSG  78
C---------BYPASS IF NOT COUPLED CONSOLIDATION                         FMSG  79
      IF(ICPL.EQ.0)GOTO 96                                            FMSG  80
C-----------------------------------------------------------------------FMSG  81
C     COUPLED CONSOLIDATION                                           FMSG  82
C-----------------------------------------------------------------------FMSG  83
      INXP=NXP(LT)                                                    FMSG  84
C---------CALCULATE RIGHT HAND SIDE FOR PORE PRESSURES                FMSG  85
      DO 94 II=1,NPN                                                  FMSG  86
      N1=NWL(II)                                                      FMSG  87
      SUM=ZERO                                                        FMSG  88
C                                                                     FMSG  89
      DO 92 JJ=1,NPN                                                  FMSG  90
      N2=NWL(JJ)                                                      FMSG  91
   92 SUM=SUM+ETE(II,JJ)*DA(N2)                                       FMSG  92
   94 P(N1)=P(N1)+SUM                                                 FMSG  93
C-----------------------------------------------------------------------FMSG  94
C     FORM SG FROM SS                                                 FMSG  95
C-----------------------------------------------------------------------FMSG  96
   96 DO 150 J=1,NDN                                                  FMSG  97
      NJ=KD(J+INXD)-1                                                 FMSG  98
C                                                                     FMSG  99
      DO 150 JD=1,NDIM                                                FMSG 100
      NJA=NJ+JD                                                       FMSG 101
      JA=J+(JD-1)*NDN                                                 FMSG 102
      NCN=NJA*(NJA-1)/2                                               FMSG 103
C                                                                     FMSG 104
      DO 150 I=1,NDN                                                  FMSG 105
      NI=KD(I+INXD)-1                                                 FMSG 106
C                                                                     FMSG 107
      DO 140 ID=1,NDIM                                                FMSG 108
      NIA=NI+ID                                                       FMSG 109
      IA=I+(ID-1)*NDN                                                 FMSG 110
      IF(NIA.GT.NJA)GOTO 140                                          FMSG 111
      LOC=NCN+NIA                                                     FMSG 112
      SG(LOC)=SS(IA,JA)                                               FMSG 113
  140 CONTINUE                                                        FMSG 114
  150 CONTINUE                                                        FMSG 115
C                                                                     FMSG 116
      IF(ICPL.EQ.0)GOTO 200                                           FMSG 117
C-----------------------------------------------------------------------FMSG 118
C     SLOT RLT                                                        FMSG 119
C-----------------------------------------------------------------------FMSG 120
      DO 160 JA=1,NPN                                                 FMSG 121
      NJA=KP(JA+INXP)                                                 FMSG 122
      NCN=NJA*(NJA-1)/2                                               FMSG 123
C                                                                     FMSG 124
      DO 160 I=1,NDN                                                  FMSG 125
      NI=KD(I+INXD)-1                                                 FMSG 126
C                                                                     FMSG 127
      DO 160 ID=1,NDIM                                                FMSG 128
      NIA=NI+ID                                                       FMSG 129
      IA=I+(ID-1)*NDN                                                 FMSG 130
      LOC=NIA+NCN                                                     FMSG 131
      IF(NIA.GT.NJA)LOC=NIA*(NIA-1)/2+NJA                             FMSG 132
  160 SG(LOC)=RLT(IA,JA)                                              FMSG 133
C-----------------------------------------------------------------------FMSG 134
C     SLOT ETE                                                        FMSG 135
```



```
C-----------------------------------------------------------------------FMSG 136
      DO 180 JE=1,NPN                                                 FMSG 137
      NJ=KP(JE+INXP)                                                  FMSG 138
      NCN=NJ*(NJ-1)/2                                                 FMSG 139
C                                                                     FMSG 140
      DO 180 IE=1,JE                                                  FMSG 141
      NI=KP(IE+INXP)                                                  FMSG 142
  180 SG(NI+NCN)=-ETE(IE,JE)                                          FMSG 143
  200 CONTINUE                                                        FMSG 144
      RETURN                                                          FMSG 145
      END                                                             FMSG 146
```

FMSG 80 : branch off, if not a consolidation element.
FMSG 86–93 : calculate RHS pore pressure **load** terms.
FMSG 97–115 : place stiffness matrix SS in appropriate place in upper triangular matrix SG (which is a one-dimensional array stored columnwise).
FMSG 117 : branch off, if not a consolidation element.
FMSG 121–133 : place coupling matrix RLT in appropriate locations in SG.
FMSG 137–143 : place flow matrix ETE in appropriate locations in SG.

8.9 PRE-FRONTAL ROUTINES

The pre-frontal stage consists of calls to routines MAKENZ, MLAPZ and SFWZ respectively. These routines have been dealt with in some detail in Chapter 6, and for the sake of completeness are summarised here.

Routine MAKENZ calculates the d.o.f. of all nodes currently present in the mesh. All nodes which are not connected to any of the elements currently present in the mesh are assigned zero d.o.f., which is entered in array NQ. This permits the program to skip these nodes when solving the equations and also at the output stage.

The routine MLAPZ marks the last appearance of each node, indicating when a node is ready for elimination. This is done by making the node number negative in array NCONN in the element in which the node makes its last appearance.

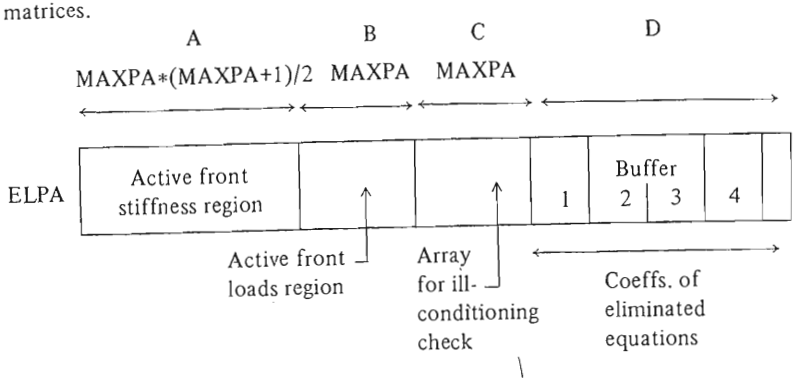
Finally routine SFWZ is called to run through the element list without actually calculating the element stiffness matrices but counting the number of active variables and assigning places in the front for new variables and simulating elimination of variables making their last appearance. This ensures that subsequent solving of equations using the **frontal method** progresses without any hitch. The amount of store required for solution and the maximum frontwidth are some of the other information calculated in routine SFWZ. This completes the pre-frontal stage.

8.10 FRONTAL SOLUTION

The routine **FRONTZ** assembles the element stiffness matrices and the load vector and solves for the unknown displacements. It uses the well known frontal

method (Irons, 1970) to solve the assembled equations. In this method the global stiffness matrix is never fully assembled.

The frontal working area ELPA(MFZN) is divided into four regions – A, B, C and D. The total allocation of ELPA, MFZN is arbitrary. The allocation G(LG) in routine MAIN is (usually fixed) based on the size of the problem and the limits imposed by the computer system. After store has been allocated for the main arrays, the rest of G(LG) is allocated as the working region for the frontal matrices.



Regions A, B, C and D are all based on the maximum frontwidth MAXPA. Region A caters for a symmetrical matrix to be stored in triangular form, columnwise, one-dimensionally. The maximum size of the array is MAXPA (rows/columns).

```
1   2   3   4   .....   MAXPA
x   x   x   x     .   .   x    1
    x   x   x     .   .   x    2
        x   x     .   .   x    3
            x     .   .   x    4
                  x   .   x    ...
                      x   x    ...
                          x    MAXPA
```

The frontwidth varies from 0 to MAXPA during different stages of the assembly/elimination phase. Therefore regions A, B and C are only full when the frontwidth is equal to the *maximum* frontwidth. Because of the one-dimensional storage of a triangular stiffness matrix (region A), each time a row/column is to be reduced (operated on) indexes are used for the first and last term (diagonal) in a given column.

If the JGth column is to be reduced then

MG0 = JG * (JG−1)/2 is the total no. of terms up to the (JG−1)th column of the region

MG1 (= MG0+1) is the index of the first term in column JG

MGZ (= MG0+JG) is the index to the last term in column JG

Routine FRONTZ

```
      SUBROUTINE FRONTZ(MAXPA,DTIME,NN,NNOD1,NEL,NDF,NTPE,NIP,NPR,NMT,   FRNT   1
     1   KES,NS,NB,NDIM,NDMX,NVRS,NPMX,INXL,MDFE,IFRZ,MUMAX,NNZ,NL,      FRNT   2
     2   XYZ,DI,DA,VARINT,P,PCOR,D,ELCOD,DS,SHFN,CARTD,B,DB,SS,ES,       FRNT   3
     3   ELCODP,E,PE,RN,AA,ETE,RLT,NCONN,MAT,LTYP,MRELVV,MREL,           FRNT   4
     4   NRELVV,NREL,NW,NQ,IDFX,NDEST,IFR,NDL,NWL,                       FRNT   5
     5   NMOD,LL,PR,NTY,ELPA,MFZN,FRACLD,IOPBC)                          FRNT   6
C*****************************************************************     FRNT   7
C     FRONTAL SOLUTION FOR SYMMETRIC MATRICES WITH                      FRNT   8
C     NDFN DEGREES OF FREEDOM PER NODE                                  FRNT   9
C*****************************************************************     FRNT  10
      REAL LL                                                           FRNT  11
      INTEGER TF                                                        FRNT  12
      CHARACTER*4 IWR,MBUF                                              FRNT  13
      DIMENSION XYZ(NDIM,NN),DI(NDF),DA(NDF),VARINT(NVRS,NIP,NEL),      FRNT  14
     1 P(NDF),PCOR(NDF),D(NS,NS),ELCOD(NDIM,NDMX),DS(NDIM,NDMX),       FRNT  15
     2 SHFN(NDMX),CARTD(NDIM,NDMX),B(NS,NB),DB(NS,NB),                 FRNT  16
     2 SS(NB,NB),ES(KES)                                               FRNT  17
      DIMENSION ELCODP(NDIM,NPMX),E(NDIM,NPMX),PE(NDIM,NPMX),           FRNT  18
     1 RN(NB),AA(NPMX),ETE(NPMX,NPMX),RLT(NB,NPMX)                     FRNT  19
      DIMENSION NCONN(NTPE,NEL),MAT(NEL),LTYP(NEL),MRELVV(NEL),         FRNT  20
     1 MREL(MUMAX),NRELVV(NN),NREL(NNZ),NW(NNOD1),NQ(NN),              FRNT  21
     2 IDFX(NDF),NDEST(NN),IFR(IFRZ),NDL(MDFE),NWL(NPMX),NMOD(NIP,NEL) FRNT  22
      DIMENSION LL(NL),PR(NPR,NMT),NTY(NMT),ELPA(MFZN)                  FRNT  23
      DIMENSION IBUF(6),MBUF(6),RBUF(3),IWR(4)                          FRNT  24
      COMMON /FIX   / DXYT(4,200),MF(200),TF(4,200),NF                  FRNT  25
      COMMON /ELINF / LINFO(50,15)                                      FRNT  26
      COMMON /DEVICE/ IR1,IR4,IR5,IW2,IW4,IW6,IW7,IW8,IW9               FRNT  27
      COMMON /PARS  / PYI,ALAR,ASMVL,ZERO                               FRNT  28
      DATA IWR(1),IWR(2),IWR(3),IWR(4)/' FIX','ED =',' LO','AD ='/      FRNT  29
C                                                                       FRNT  30
      KURPA=0                                                           FRNT  31
      NPAR=MAXPA*(MAXPA+1)/2                                            FRNT  32
      NBAXO=NPAR+2*MAXPA+1                                              FRNT  33
      IBA=NBAXO                                                         FRNT  34
      NVABZ=0                                                           FRNT  35
      NBAXZ=MFZN                                                        FRNT  36
      INITL=1                                                           FRNT  37
      NDIM1=NDIM+1                                                      FRNT  3
      IC=0                                                              FRNT  3
C                                                                       FRNT  40
C                                                                       FRNT  41
      IF(IOPBC.EQ.1)WRITE(IW6,910)                                      FRNT  42
  910 FORMAT(//31H PRESCRIBED BOUNDARY CONDITIONS/1X,30(1H-)/)          FRNT  43
C-----------------------------------------------------------------      FRNT  44
C     ZERO LIST OF FIXED D.O.F.                                         FRNT  45
C-----------------------------------------------------------------      FRNT  46
      CALL ZEROI1(IDFX,NDF)                                             FRNT  47
C                                                                       FRNT  48
      DO 10 IJ=1,MFZN                                                   FRNT  49
   10 ELPA(IJ)=ZERO                                                     FRNT  50
C-----------------------------------------------------------------      FRNT  51
C     LOOP ON ELEMENTS                                                  FRNT  52
C-----------------------------------------------------------------      FRNT  53
      DO 62 NE=1,NEL                                                    FRNT  54
      LT=LTYP(NE)                                                       FRNT  55
      IF(LT.LT.0.AND.NE.EQ.NEL)GO TO 61                                 FRNT  56
      IF(LT.LT.0) GOTO 62                                               FRNT  57
      MUS=MRELVV(NE)                                                    FRNT  58
```

```
C                                                                       FRNT  59
      CALL LSTIFF(NE,MUS,INXL,ES,KES,DTIME,NN,NNOD1,NEL,NDF,NTPE,NIP,   FRNT  60
     1    NPR,NMT,NS,NB,NL,NDIM,NDMX,NVRS,NPMX,LT,XYZ,DA,VARINT,P,      FRNT  61
     2    D,ELCOD,DS,SHFN,CARTD,B,DB,SS,ELCODP,E,PE,RN,AA,ETE,          FRNT  62
     3    RLT,NCONN,MAT,NW,NWL,NMOD,LL,PR,NTY)                          FRNT  63
C-----------------------------------------------------------------------FRNT  64
C     ASSEMBLE ELEMENT STIFFNESS MATRIX INTO FRONTAL REGION             FRNT  65
C-----------------------------------------------------------------------FRNT  66
      CALL FRSLOT(NN,NEL,NTPE,KES,MDFE,IFRZ,NCONN,NQ,NDEST,IFR,NDL,     FRNT  67
     1    ES,ELPA,MFZN,LT,NE,KURPA,INXL)                                FRNT  68
C                                                                       FRNT  69
CC    CALL PRINTF(IW6,ELPA(1),MFZN,KURPA,ELPA(NPAR+1),KURPA,2)          FRNT  70
C-----------------------------------------------------------------------FRNT  71
C     ASSEMBLE RIGHT HAND SIDE / FIX DEGREES OF FREEDOM                 FRNT  72
C-----------------------------------------------------------------------FRNT  73
      NDPT=LINFO(1,LT)                                                  FRNT  74
      CALL FRFXLD(IW6,NN,NNOD1,NEL,NDF,NTPE,NDIM,DA,P,PCOR,             FRNT  75
     1    NCONN,NRELVV,NW,NQ,IDFX,NDEST,ELPA,MFZN,FRACLD,NE,NDPT,NPAR,  FRNT  76
     2    IC,IOPBC,IBUF,MBUF,RBUF,IWR)                                  FRNT  77
CC    CALL PRINTF(IW6,ELPA(1),MFZN,KURPA,ELPA(NPAR+1),KURPA,2)          FRNT  78
C-----------------------------------------------------------------------FRNT  79
C     ELIMINATE                                                         FRNT  80
C-----------------------------------------------------------------------FRNT  81
      DO 60 J=1,NDPT                                                    FRNT  82
      IF(NCONN(J,NE).GT.0) GOTO 60                                      FRNT  83
      NA=-NCONN(J,NE)                                                   FRNT  84
      NDFN=NQ(NA)                                                       FRNT  85
      ND=NDEST(NA)+NDFN-1                                               FRNT  86
C-----------------------------------------------------------------------FRNT  87
C     LOOP ON ALL D.O.F. OF NODE BEING ELIMINATED                       FRNT  88
C-----------------------------------------------------------------------FRNT  89
      DO 58 JJ=1,NDFN                                                   FRNT  90
      NVABZ=NVABZ+1                                                     FRNT  91
      NDEQN=IBA+KURPA+4                                                 FRNT  92
      IF(NDEQN.GT.NBAXZ)CALL STOREQ(ELPA,MFZN,NBAXO,IBA,NDEQN,KURPA,IW7)FRNT  93
      NPA=ND+1-JJ                                                       FRNT  94
      IBDIAG=IBA+NPA                                                    FRNT  95
      NDIAG=IBDIAG                                                      FRNT  96
      IF(INITL.NE.0) NDIAG=NPA*(NPA+1)/2                                FRNT  97
      PIVOT=ELPA(NDIAG)                                                 FRNT  98
      ELPA(NDIAG)=ZERO                                                  FRNT  99
      IF(ABS(PIVOT).GT.ASMVL) GOTO 34                                   FRNT 100
      WRITE(IW6,911)                                                    FRNT 101
  911 FORMAT(36H ERROR - ZERO PIVOT (ROUTINE FRONTZ))                   FRNT 102
      STOP                                                              FRNT 103
C                                                                       FRNT 104
   34 MGZ=0                                                             FRNT 105
      JGZ=KURPA                                                         FRNT 106
      IBO=IBA                                                           FRNT 107
      IF(INITL.EQ.0) IBA=IBA+KURPA                                      FRNT 108
      L12=2-INITL                                                       FRNT 109
C                                                                       FRNT 110
      DO 50 LHSRHS=L12,2                                                FRNT 111
      IF(LHSRHS.EQ.2) JGZ=1                                             FRNT 112
C                                                                       FRNT 113
      DO 48 JG=1,JGZ                                                    FRNT 114
      IBA=IBA+1                                                         FRNT 115
      GOTO(36,40),LHSRHS                                                FRNT 116
C                                                                       FRNT 117
   36 MGO=MGZ                                                           FRNT 118
      MGZ=MGO+JG                                                        FRNT 119
      IF(NPA.GT.JG)GOTO 38                                              FRNT 120
      MG=MGO+NPA                                                        FRNT 121
      GOTO 42                                                           FRNT 122
C                                                                       FRNT 123
   38 MG=JG+NPA*(NPA-1)/2                                               FRNT 124
      GOTO 42                                                           FRNT 125
```



```
C                                                                       FRNT 126
   40 MGO=NPAR                                                          FRNT 127
      MG=MGO+NPA                                                        FRNT 128
      MGZ=MGO+KURPA                                                     FRNT 129
C                                                                       FRNT 130
   42 NDELT=IBO-MGO                                                     FRNT 131
      CONST=ELPA(MG)                                                    FRNT 132
      ELPA(IBA)=CONST                                                   FRNT 133
      IF(ABS(CONST).LT.ASMVL) GOTO 48                                   FRNT 134
      CONST=CONST/PIVOT                                                 FRNT 135
      ELPA(MG)=ZERO                                                     FRNT 136
      IF(INITL.NE.LHSRHS) GOTO 44                                       FRNT 137
      MG=NPAR+MAXPA+JG                                                  FRNT 138
      ELPA(MG)=ELPA(MG)+ELPA(MGZ)*ELPA(MGZ)                             FRNT 139
C                                                                       FRNT 140
   44 MG1=MGO+1                                                         FRNT 141
      DO 46 I=MG1,MGZ                                                   FRNT 142
      K=I+NDELT                                                         FRNT 143
      ELPA(I)=ELPA(I)-CONST*ELPA(K)                                     FRNT 144
   46 CONTINUE                                                          FRNT 145
   48 CONTINUE                                                          FRNT 146
   50 CONTINUE                                                          FRNT 147
C                                                                       FRNT 148
      ELPA(IBDIAG)=PIVOT                                                FRNT 149
      IBA=NDEQN                                                         FRNT 150
      ELPA(IBA)=FLOAT(KURPA)                                            FRNT 151
      ELPA(IBA-1)=FLOAT(NPA)                                            FRNT 152
      ELPA(IBA-2)=FLOAT(NW(NA)+NDFN-JJ)                                 FRNT 153
      IF(INITL.EQ.0) GOTO 56                                            FRNT 154
C-----------------------------------------------------------------------FRNT 155
C      SKIP MORE ON RESOLN                                              FRNT 156
C-----------------------------------------------------------------------FRNT 157
      MG=NPAR+MAXPA+NPA                                                 FRNT 158
      CRIT=SQRT(ELPA(MG))/ABS(PIVOT)                                    FRNT 159
      ELPA(MG)=ZERO                                                     FRNT 160
      IF(CRIT.LT.1.0E8) GOTO 52                                         FRNT 161
      WRITE(IW6,912)                                                    FRNT 162
  912 FORMAT(51H PROBABLE SERIOUS ILL-CONDITIONING (ROUTINE FRONTZ))    FRNT 163
      STOP                                                              FRNT 164
C                                                                       FRNT 165
   52 IF(CRIT.LT.1.E4.AND.PIVOT.GT.0.) GOTO 54                          FRNT 166
CC    WRITE(IW6,912)                                                    FRNT 167
CC912 FORMAT(26H POSSIBLE ILL CONDITIONING)                             FRNT 168
C                                                                       FRNT 169
   54 CONTINUE                                                          FRNT 170
C                                                                       FRNT 171
   56 IF(NPA.EQ.KURPA) KURPA=KURPA-1                                    FRNT 172
      IFR(NPA)=0                                                        FRNT 173
C                                                                       FRNT 174
   58 CONTINUE                                                          FRNT 175
C                                                                       FRNT 176
      NCONN(J,NE)=-NCONN(J,NE)                                          FRNT 177
      IF(KURPA.GT.ND) GOTO 60                                           FRNT 178
   59 IF(KURPA.EQ.0)GOTO 60                                             FRNT 179
      IF(IFR(KURPA).GT.0) GOTO 60                                       FRNT 180
      KURPA=KURPA-1                                                     FRNT 181
      GOTO 59                                                           FRNT 182
C                                                                       FRNT 183
   60 CONTINUE                                                          FRNT 184
C                                                                       FRNT 185
CC    CALL PRINTF(IW6,ELPA(1),MFZN,KURPA,ELPA(NPAR+1),KURPA,2)          FRNT 186
C-----------------------------------------------------------------------FRNT 187
C     OUTPUT BUFFER                                                     FRNT 188
C-----------------------------------------------------------------------FRNT 189
   61 IF(NE.NE.NEL) GOTO 62                                             FRNT 190
      IF(IC.EQ.0) GOTO 62                                               FRNT 191
```

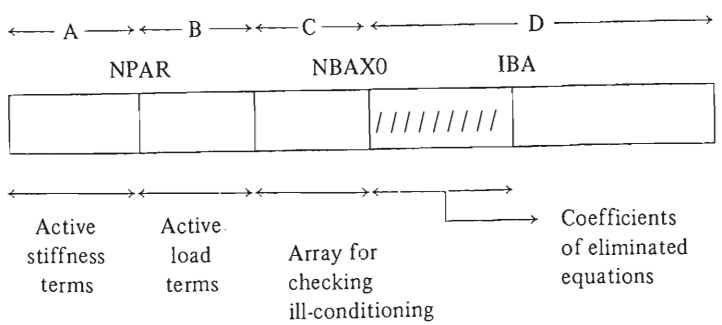
```
      IF(IC.EQ.1.AND.IOPBC.EQ.1)WRITE(IW6,921)IBUF(1),IBUF(2),          FRNT 192
     1 MBUF(1),MBUF(2),RBUF(1)                                          FRNT 193
  921 FORMAT(5H NODE,I5,7H D.O.F.,I3,2A4,E13.4)                         FRNT 194
      IF(IC.EQ.2.AND.IOPBC.EQ.1)WRITE(IW6,922)(IBUF(2*IM-1),IBUF(2*IM), FRNT 195
     1 MBUF(2*IM-1),MBUF(2*IM),RBUF(IM),IM=1,2)                         FRNT 196
  922 FORMAT(2(5H NODE,I5,7H D.O.F.,I3,2A4,E13.4,4X))                   FRNT 197
   62 CONTINUE                                                          FRNT 198
C-------------------------------------------------------------------    FRNT 199
C                                                                       FRNT 200
C     BACK SUBSTITUTE                                                   FRNT 201
C-------------------------------------------------------------------    FRNT 202
   70 IF(NVABZ.EQ.0) GOTO 80                                            FRNT 203
      IF(IBA.EQ.NBAX0) CALL GETEQN(ELPA,MFZN,NBAX0,IBA,IW7)             FRNT 204
      NVABZ=NVABZ-1                                                     FRNT 205
      KURPA=IFIX(ELPA(IBA))                                             FRNT 206
      NPA=IFIX(ELPA(IBA-1))                                             FRNT 207
      NIC=IFIX(ELPA(IBA-2))                                             FRNT 208
      IBAR=IBA-4                                                        FRNT 209
      IBA=IBAR-KURPA                                                    FRNT 210
      IBDIAG=IBA+NPA                                                    FRNT 211
      PIVOT=ELPA(IBDIAG)                                                FRNT 212
C                                                                       FRNT 213
      ELPA(IBDIAG)=ZERO                                                 FRNT 214
      CONST=ELPA(IBAR+1)                                                FRNT 215
C                                                                       FRNT 216
      DO 72 I=1,KURPA                                                   FRNT 217
      K=I+IBA                                                           FRNT 218
      CONST=CONST-ELPA(I)*ELPA(K)                                       FRNT 219
   72 CONTINUE                                                          FRNT 220
C                                                                       FRNT 221
      ELPA(NPA)=CONST/PIVOT                                             FRNT 222
      DI(NIC)=ELPA(NPA)                                                 FRNT 223
C                                                                       FRNT 224
      ELPA(IBDIAG)=PIVOT                                                FRNT 225
      GOTO 70                                                           FRNT 226
C                                                                       FRNT 227
   80 CONTINUE                                                          FRNT 228
      RETURN                                                            FRNT 229
      END
```

FRNT 31 : KURPA, the current *frontwidth*, is zeroed.

FRNT 32 : size of front – based on maximum frontwidth (= MAXPA).

FRNT 33 : size of front and active load terms + region for ill-conditioning check (= NBAX0).

FRNT 34 : is the index to last coefficient of equation eliminated last (during the elimination phase). Set initial value of IBA which is the index to **first** empty location (given by NBAX0 + 1) in buffer region.



FRNT 35 : zero counter of variables eliminated.

FRNT 36 : set NBAXZ equal to the size of working region (size of array ELPA).

FRNT 37 : set flag to indicate that solution is initial solution (not a re-solution).

FRNT 38 : NDIM1 = NDIM + 1.

FRNT 39 : zero counter of no. of terms (fixities/loads) in output buffer of boundary conditions (b.c.)/loads.

FRNT 47 : zero global array of indicators of d.o.f. with prescribed values.

FRNT 54 : loop on all elements.

FRNT 56 : branch off to print output buffer of b.c./loads (in the event that the last element is not present in the current mesh and the contents of unfilled output buffer have not been printed).

FRNT 57 : by-pass if element is not present in current mesh.

FRNT 60–63 : calculate element stiffness matrix (also flow and coupling matrices for consolidation element), placed in array SG.

FRNT 67–68 : assemble element stiffness matrix into front.

FRNT 70 : print out contents of *active* front region (only if debugging).

FRNT 75–77 : assemble load term or fix d.o.f. of node making last appearance.

FRNT 78 : print out contents of front (stiffness and loads) for debugging.

Elimination phase

FRNT 82 : loop on all nodes of element.

FRNT 83 : by-pass if node is not making its last appearance.

FRNT 90 : loop on all d.o.f. of node making last appearance.

FRNT 91 : increment counter of no. of variables (d.o.f.) eliminated.

FRNT 92 : set up new pointer position in buffer to indicate last positio of entries for next equation to be eliminated. When each equation is eliminated, all coefficients (equals the front-width) plus four terms are entered in the buffer. Here NDEQN points to the last term.

FRNT 93 : buffer cannot accommodate all coefficients of current equation being eliminated (NDEQN > NBAXZ). Therefore write contents of buffer to backing store.

FRNT 94 : row/column no. of equation to be eliminated (variables of a node are eliminated in reverse order, i.e. last variable (d.o.f.) is eliminated first).

FRNT 95 : location of pivotal term in buffer.

FRNT 96 : pointer of pivotal term (if re-solution) in buffer.

FRNT 97 : for initial solution, index to pivotal term in front region.

```
FRNT  98      : pivot.
FRNT  99      : replace pivotal term by zero in front.
FRNT 100      : check pivot is not equal to zero.
FRNT 101–102  : if so, print error message and stop.
FRNT 105      : the index to first term in front is MGZ + 1.
FRNT 106      : no. of columns in front (= frontwidth).
FRNT 107      : remember index of last entry to buffer from equation
                eliminated last (IB0 + 1 points to the next empty location in
                buffer).
FRNT 108      : if resolution.
FRNT 109      : L12 = 1 for new solution; both stiffness and load terms have
                to be reduced.
                : L12 = 2 for re-solution; only load terms have to be reduced.
FRNT 111      : loop on stiffness and load terms, depending on L12.
FRNT 112      : reset JGZ to one column of load terms, if resolution.
FRNT 114      : loop on JGZ columns.
FRNT 115      : position in buffer for the next coefficient.
FRNT 116      : branch off for stiffness and load terms.
```

118–124 for reduction of stiffness terms only.

```
FRNT 118      : index to first term in JGth column is MG0 + 1.
FRNT 119      : index to diagonal term in JGth column.
FRNT 120      : branch off if equation eliminated is to the right of equation
                JG.
FRNT 121      : index to coefficient being eliminated (it is along the row part
                of equation NPA).
FRNT 124      : index to coefficient being eliminated (it is on column part of
                equation NPA).
```

127–129 for reduction of load terms only.

```
FRNT 127      : first load term is given by MG0 + 1.
FRNT 128      : index to load term of equation being eliminated.
FRNT 129      : index to last load term.

FRNT 131      : a shift (additive) index.
FRNT 132      : coefficient being eliminated – CONST.
FRNT 133      : place CONST in buffer.
FRNT 134      : check if CONST = 0; if so, this is the outer loop by-pass,
                i.e. terms in column JG are not affected.
FRNT 135      : multiplication factor.
FRNT 136      : set pivotal term to zero in front.
FRNT 137      : by-pass if no reduction of stiffness terms.
FRNT 138      : index to location in ill-conditioning check region for
                equation JG.
```



```
FRNT 139      : add square of diagonal term of equation JG. (ill-conditioning
                check).
FRNT 141      : index to first term in column JG.
FRNT 142      : loop on all terms in column JG.
FRNT 143      : index to terms of equation being eliminated, in buffer.
FRNT 144      : reduce term in column JG.
FRNT 145      : end of inner loop – on all rows in column JG.
FRNT 146      : end of outer loop – on all columns in front.
FRNT 147      : end of LHS/RHS loop.
FRNT 149      : place pivot in the diagonal position in buffer.
FRNT 150      : reset pointer to last entry of current equation in buffer.
FRNT 151      : enter current size of front.
FRNT 152      : index to pivotal term.
FRNT 153      : global variable no. (index to array DI).
FRNT 154      : by-pass if re-solution.
FRNT 158–159  : entry corresponding to eliminated equation in ill-
                conditioning check region.
FRNT 160      : reset entry to zero.
FRNT 161–170  : print out warning messages if equations are ill-conditioned.
FRNT 172–173  : if eliminated equation was occupying the last row/column
                then reduce front size.
FRNT 175      : end of loop on d.o.f. of node making its last appearance.
FRNT 177      : make node no. positive in NCONN.
FRNT 178–181  : reduce front size if eliminated variables were at the end of
                front.
FRNT 184      : end of loop on all nodes in element.
FRNT 186      : print out contents of active front (stiffness and load terms).
                Only if program is being debugged.
FRNT 190–197  : print out contents of b.c./load output buffer, if not empty.
```

Back-substitution phase

```
FRNT 202      : branch off if all unknowns have been solved for.
FRNT 203      : if buffer is empty, get coefficients from backing store.
FRNT 204      : decrement no. of unknowns yet to be solved by one.
FRNT 205      : current size of front (= KURPA).
FRNT 206      : position of equation of unknown variable in front (also the
                index to the pivotal term).
FRNT 207      : global variable number.
FRNT 208      : index to load term of equation is IBAR + 1.
FRNT 209      : index of first coefficient (IBA + 1) of equation.
FRNT 210      : index of diagonal coefficient of equation.
FRNT 211      : get pivot.
FRNT 213      : replace by zero.
FRNT 214      : RHS load term of equation.
```

FRNT 216 : loop on all terms (a column of solved unknowns) in front.
FRNT 217 : index to coefficient in buffer.
FRNT 218 : reduce RHS term.
FRNT 221 : calculate incremental displacement/excess pore pressure (i.e. solve for unknowns).
FRNT 222 : place displacement/excess pore pressure in array DI.
FRNT 224 : place pivotal term in buffer (in case of resolution).
FRNT 225 : solve next unknown.
FRNT 227 : end of back-substitution loop; all unknowns solved.

8.11 FRONTAL SOLVER

The frontal solver in the program uses a one-dimensional array and is for the solving of symmetric stiffness matrices only. Therefore only problems of material behaviour which obey the associated flow rule can be analysed. The solver can handle variable d.o.f. of nodes and is independent of the type of element being used.

The solver also allows for a re-solution facility. However, some modifications to the program would be necessary if this was needed. The re-solution facility would be useful if the Modified Newton–Raphson approach were required. Within the iterative cycle the stiffness remains constant. Therefore the Left-Hand Side (LHS) stiffness terms are reduced once and can be re-used to solve the Right-Hand Side (RHS), which varies from iteration to iteration. CRISP uses an incremental approach (not iterative) and the re-solution facility is not used in the present version.

Given below are some of the general features often found in frontal solvers. A minimum amount of core (which is calculated in SFWZ) is necessary to solve the equations. This is calculated as the core required to keep all the stiffness terms, load terms and terms for ill-conditioning check when the frontwidth is at its maximum. The program is not capable of solving the equations if this minimum core is not provided.

Equations of prescribed displacements are not dealt with any differently from the other equations. A large number ALAR is added to the diagonal term, and the prescribed value multiplied by ALAR is added to the corresponding RHS term.

There is an ill-conditioning check (Irons, 1968). This check does not involve the RHS terms and is based on the reduction in the diagonal term since becoming active until it becomes a pivot.

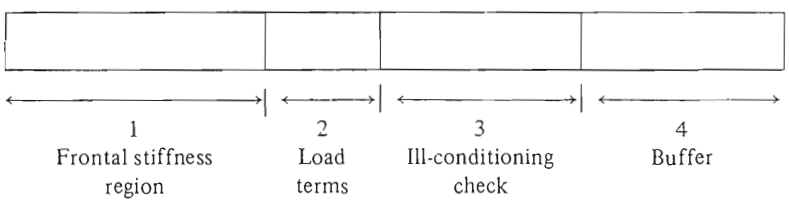
The use of higher-order elements makes the frontal method more attractive compared to band solvers. A point often made is that the nodal numbering is irrelevant whereas the element numbering should be efficient for the frontal method. In contrast, for band solvers the element numbering is irrelevant whereas the nodal numbering must be such that it produces the smallest band-width. The program should allow for the flexibility of re-numbering either the

nodes or the elements internally or by a separate stand-alone program so that the user specified numbers are only used for communicating with the user.

CRISP does not make any attempt to re-number the elements. The user element sequence at input is considered to be the frontal assembly order, if alternative element numbering is not provided. An element ordering efficient for the frontal method can be read in separately (record G2). This option is provided for flexibility and efficiency. Not all finite element meshes have a regular grid (or pattern) of elements. An efficient frontal ordering of a finite element mesh may not readily be apparent. When preparing the mesh, the user may number the elements which are of main interest (for example in a tunnel analysis, the elements surrounding the tunnel) first. With this form of input option an efficient frontal numbering of elements is uncoupled from the concern of the user, who chooses to number the element in a manner convenient to him/her.

8.12 SOLUTION OF THE EQUATIONS

The frontal method begins as soon as the first element stiffness matrix has been assembled into the frontal region. The frontal region is made up of a one-dimensional array partitioned into four different regions. The organisation of this mainly depends on the particular implementation. In this particular version it is as shown below:



The boundaries are fixed and are based on the maximum frontwidth MAXPA which was determined in routine SFWZ.

The element stiffness matrix is assembled into the appropriate locations in region 1. The elimination phase begins for all equations which are fully assembled. For these equations the corresponding variable is checked to see whether it is prescribed. The corresponding load term is also assembled into its assigned location in region 2. The coefficients of the complete equation are transferred to the buffer, one by one; at each stage (i.e. for each transfer) the relevant column of terms is modified (operated on). The next element is now assembled and the whole procedure is repeated.

The frontal solution step is divided into three parts:

(i) adding element stiffness matrix into front – routine FRSLOT;
(ii) dealing with prescribed displacements and applied loads – routine FRFXLD;
(iii) forward elimination and backward substitution – routine FRONTZ.

Routine FRSLOT

For each element in turn (by making a call to routine LSTIFF) the element stiffness matrix ES (called SG in routine LSTIFF) is assembled into the active front region. Both the element stiffness matrix (ES) and the front stiffness region (ELPA(1)–ELPA(NPAR) are upper triangular matrices.

```
      SUBROUTINE FRSLOT(NN,NEL,NTPE,KES,MDFE,IFRZ,NCONN,NQ,NDEST,      FRST   1
     1 IFR,NDL,ES,ELPA,MFZ,LT,NE,KURPA,INXL)                           FRST   2
C*****************************************************************   FRST   3
C     ROUTINE TO SLOT ELEMENT STIFFNESS MATRIX IN APPROPRIATE         FRST   4
C     LOCATIONS (AS INDICATED BY ARRAY NDL) IN FRONTAL REGION         FRST   5
C*****************************************************************   FRST   6
      DIMENSION NCONN(NTPE,NEL),NQ(NN),NDEST(NN),IFR(IFRZ),NDL(MDFE),  FRST   7
     1 ES(KES),ELPA(MFZ)                                               FRST   8
      COMMON /ELINF / LINFO(50,15)                                     FRST   9
C-----------------------------------------------------------------    FRST  10
C     FIND CURRENT SIZE OF GRANDPA (KURPA)                            FRST  11
C-----------------------------------------------------------------    FRST  12
                                                                       FRST  13
      NDPT=LINFO(1,LT)                                                 FRST  14
C                                                                      FRST  15
      DO 10 J=1,NDPT                                                   FRST  16
      N=NCONN(J,NE)                                                    FRST  17
      NA=IABS(N)                                                       FRST  18
      NDFN=NQ(NA)                                                      FRST  19
      ND=NDEST(NA)-1                                                   FRST  20
C                                                                      FRST  21
      DO 6 I=1,NDFN                                                    FRST  22
      ND=ND+1                                                          FRST  23
    6 IFR(ND)=NA                                                       FRST  24
      IF(ND.LT.KURPA) GOTO 10                                          FRST  25
      KURPA=ND                                                         FRST  26
   10 CONTINUE                                                         FRST  27
C-----------------------------------------------------------------    FRST  28
C     ASSEMBLE ELEMENT STIFFNESS INTO GRANDPA                         FRST  29
C-----------------------------------------------------------------    FRST  30
      IT=0                                                             FRST  31
      DO 16 J=1,NDPT                                                   FRST  32
      N=NCONN(J,NE)                                                    FRST  33
      NA=IABS(N)                                                       FRST  34
      ND=NDEST(NA)-1                                                   FRST  35
C-----------------------------------------------------------------    FRST  36
C     INXL - INDEX TO NODAL D.O.F. (SEE ROUTINES BDATA1 AND MAIN2)    FRST  37
C-----------------------------------------------------------------    FRST  38
      NDFN=LINFO(J+INXL,LT)                                            FRST  39
      DO 16 JJ=1,NDFN                                                  FRST  40
      IT=IT+1                                                          FRST  41
   16 NDL(IT)=ND+JJ                                                    FRST  42
      IS=0                                                             FRST  43
      DO 20 J=1,IT                                                     FRST  44
      NDJ=NDL(J)                                                       FRST  45
      KS=NDJ*(NDJ-1)/2                                                 FRST  46
      DO 20 I=1,J                                                      FRST  47
      IS=IS+1                                                          FRST  48
      NDI=NDL(I)                                                       FRST  49
      IF(NDI.GT.NDJ)GO TO 18                                           FRST  50
      KX1=KS+NDI                                                       FRST  51
      GO TO 20                                                         FRST  52
   18 KX1=NDI*(NDI-1)/2+NDJ                                            FRST  53
   20 ELPA(KX1)=ELPA(KX1)+ES(IS)                                       FRST  54
      RETURN                                                           FRST  55
      END
```


FRST 13 : total number of nodes in element.
FRST 15 : loop on all nodes in element.
FRST 21 : loop on all variables (d.o.f.) of node.
FRST 23 : enter node numbers in the list of active nodes (i.e. nodes which are currently in front).
FRST 25 : update KURPA if the front has expanded.
FRST 26 : end of loop on all nodes of element.
FRST 31 : loop on all nodes of element.
FRST 32–41 : set up array NDL, which gives the index to front region for the rows/columns of array SG (element stiffness matrix). Indicates which row/column of SG should go into which row/column of the front.
FRST 43–53 : making use of array NDL, add in all stiffness terms SG into *active* front region in appropriate rows/columns.

Routine FRFXLD

This routine deals with fixities (prescribed displacements and excess pore pressures) and loads. The DO 25 loop is to find which of the variables (d.o.f. d_x, d_y and $\bar{u}$) have prescribed values and to fix them.

```
      SUBROUTINE FRFXLD(IW6,NN,NNOD1,NEL,NDF,NTPE,NDIM,DA,P,PCOR,      FXLD   1
     1 NCONN,NRELVV,NW,NQ,IDFX,NDEST,ELPA,MFZ,FRACLD,NE,NDPT,NPAR,IC,  FXLD   2
     2 IOPBC,IBUF,MBUF,RBUF,IWR)                                       FXLD   3
C*****************************************************************   FXLD   4
C      ROUTINE TO ASSEMBLE LOAD TERM AND FIX VARIABLES WITH PRESCRIBED FXLD   5
C      VALUES FOR NODES MAKING LAST APPEARANCE                         FXLD   6
C*****************************************************************   FXLD   7
      INTEGER TF                                                       FXLD   8
      CHARACTER*4 IWR,MBUF                                             FXLD   9
      DIMENSION DA(NDF),P(NDF),PCOR(NDF),NCONN(NTPE,NEL),NRELVV(NN),   FXLD  10
     1 NW(NNOD1),NQ(NN),IDFX(NDF),NDEST(NN),ELPA(MFZ)                  FXLD  11
      DIMENSION IBUF(6),MBUF(6),RBUF(3),IWR(4),NTT(4)                  FXLD  12
      COMMON /FIX   / DXYT(4,200),MF(200),TF(4,200),NF                 FXLD  13
      COMMON /PARS  / PYI,ALAR,ASMVL,ZERO                              FXLD  14
C-----------------------------------------------------------------    FXLD  15
      NDIM1=NDIM+1                                                     FXLD  16
      DO 30 I=1,NDPT                                                   FXLD  17
      IF(NCONN(I,NE).GT.0) GOTO 30                                     FXLD  18
      MA=-NCONN(I,NE)                                                  FXLD  19
      NUNDE=NRELVV(MA)                                                 FXLD  20
C-----------------------------------------------------------------    FXLD  21
C     FIND IF FIXED                                                   FXLD  22
C-----------------------------------------------------------------    FXLD  23
      DO 21 J=1,NF                                                     FXLD  24
      IF(MA.EQ.MF(J)) GOTO 22                                          FXLD  25
   21 CONTINUE                                                         FXLD  26
      GOTO 26                                                          FXLD  27
C                                                                      FXLD  28
   22 DO 19 ID=1,NDIM                                                  FXLD  29
   19 NTT(ID)=TF(ID,J)                                                 FXLD  30
      KDF=NQ(MA)                                                       FXLD  31
C-----------------------------------------------------------------    FXLD  32
C     PORE PRESSURE VARIABLE IS THE FIRST (WHEN NODE HAS PORE-        FXLD  33
C     PRESSURE VARIABLE ONLY) OR LAST (NDIM+1) D.O.F. OF ANY NODE.    FXLD  34
C     BUT ALWAYS USE LOCATION NDIM+1 OF ARRAYS NTT AND TF FOR         FXLD  35
C     PORE PRESSURE FIXITY                                            FXLD  36
```

```
C-----------------------------------------------------------------------FXLD  37
      IF(KDF.EQ.1.OR.KDF.EQ.NDIM1)NTT(NDIM1)=TF(NDIM1,J)                FXLD  38
C                                                                       FXLD  39
      FAC=FRACLD                                                        FXLD  40
      LCI=NW(MA)                                                        FXLD  41
      MDI=NDEST(MA)                                                     FXLD  42
      DO 25 IDOF=1,KDF                                                  FXLD  43
      ISF=0                                                             FXLD  44
      NTTI=NTT(IDOF)                                                    FXLD  45
      INDX=IDOF                                                         FXLD  46
C----------NODE HAS PORE PRESSURE VARIABLE ONLY                         FXLD  47
      IF(KDF.NE.1)GO TO 23                                              FXLD  48
      NTTI=NTT(NDIM1)                                                   FXLD  49
      INDX=NDIM1                                                        FXLD  50
      ISF=NDIM                                                          FXLD  51
   23 IF(NTTI.EQ.0)GO TO 25                                             FXLD  52
      MDF=MDI+IDOF-1                                                    FXLD  53
      NPA=MDF*(MDF+1)/2                                                 FXLD  54
      ELPA(NPA)=ELPA(NPA)+ALAR                                          FXLD  55
      NPRF=NPAR+MDF                                                     FXLD  56
C-----------------------------------------------------------------------FXLD  57
C     RESET NODAL PORE PRESSURE FIXITY CODE OF 2 BY 1 AND THE           FXLD  58
C     MAGNITUDE TO ZERO                                                 FXLD  59
C-----------------------------------------------------------------------FXLD  60
      IF(IDOF.NE.KDF)GOTO 24                                            FXLD  61
      IF(NTTI.NE.2)GOTO 24                                              FXLD  62
      FAC=1.                                                            FXLD  63
      TF(NDIM1,J)=1                                                     FXLD  64
      DXYT(INDX,J)=DXYT(INDX,J)-DA(LCI+IDOF-1)                          FXLD  65
24    ELPA(NPRF)=ELPA(NPRF)+DXYT(INDX,J)*ALAR*FAC                       FXLD  66
CC    WRITE(IW6,806)I,IDOF                                              FXLD  67
CC806 FORMAT(/1X,4HI = ,I5,3X,7HIDOF = ,I5)                             FXLD  68
      IC=IC+1                                                           FXLD  69
      LC=LCI+IDOF-1                                                     FXLD  70
      IDFX(LC)=1                                                        FXLD  71
      IBUF(2*IC-1)=NUNDE                                                FXLD  72
      IBUF(2*IC)=IDOF+ISF                                               FXLD  73
      MBUF(2*IC-1)=IWR(1)                                               FXLD  74
      MBUF(2*IC)=IWR(2)                                                 FXLD  75
      RBUF(IC)=DXYT(INDX,J)*FAC                                         FXLD  76
      IF(IC.EQ.3.AND.IOPBC.EQ.1)WRITE(IW6,910)(IBUF(2*IM-1),IBUF(2*IM), FXLD  77
     1 MBUF(2*IM-1),MBUF(2*IM),RBUF(IM),IM=1,3)                         FXLD  78
  910   FORMAT(2(5H NODE,I5,7H D.O.F.,I3,2A4,E13.4,4X),                 FXLD  79
     1 5H NODE,I5,7H D.O.F.,I3,2A4,E13.4)                               FXLD  80
      IF(IC.EQ.3)IC=0                                                   FXLD  81
      IF(IDOF.NE.1.AND.IDOF.NE.NDIM1)GOTO 25                            FXLD  82
      IF(NTTI.NE.2)GOTO 25                                              FXLD  83
      DXYT(INDX,J)=0.                                                   FXLD  84
   25 CONTINUE                                                          FXLD  85
C                                                                       FXLD  86
   26 MDEST=NPAR+NDEST(MA)-1                                            FXLD  87
      MSO=NW(MA)-1                                                      FXLD  88
      NDFN=NQ(MA)                                                       FXLD  89
C                                                                       FXLD  90
      DO 27 JJ=1,NDFN                                                   FXLD  91
      MDEST=MDEST+1                                                     FXLD  92
      MSO=MSO+1                                                         FXLD  93
      ELPA(MDEST)=ELPA(MDEST)+P(MSO)+PCOR(MSO)                          FXLD  94
      IF(ABS(P(MSO)).LT.ASMVL.AND.ABS(PCOR(MSO)).LT.ASMVL) GOTO 27      FXLD  95
      ISF=0                                                             FXLD  96
      IF(NDFN.EQ.1)ISF=NDIM                                             FXLD  97
      IC=IC+1                                                           FXLD  98
      IBUF(2*IC-1)=NUNDE                                                FXLD  99
      IBUF(2*IC)=JJ+ISF                                                 FXLD 100
      MBUF(2*IC-1)=IWR(3)                                               FXLD 101
      MBUF(2*IC)=IWR(4)                                                 FXLD 102
```



```
      RBUF(IC)=P(MSO)+PCOR(MSO)                                         FXLD 103
      IF(IC.EQ.3.AND.IOPBC.EQ.1)WRITE(IW6,910)(IBUF(2*IM-1),IBUF(2*IM), FXLD 104
     1 MBUF(2*IM-1),MBUF(2*IM),RBUF(IM),IM=1,3)                         FXLD 105
      IF(IC.EQ.3)IC=0                                                   FXLD 106
   27 CONTINUE                                                          FXLD 107
C                                                                       FXLD 108
   30 CONTINUE                                                          FXLD 109
      RETURN                                                            FXLD 110
      END                                                               FXLD 111
```

FXLD 17 : loop on all nodes of element to assemble RHS load terms. Fix d.o.f. with prescribed values of nodes making last appearance.

FXLD 18 : by-pass if node is not making last appearance.

FXLD 24 : scan the list of nodes with (displacement/pore pressure) fixities.

FXLD 25 : branch off if node is found in the list of fixities.

FXLD 27 : skip if node is free.

FXLD 29–30 : copy fixity code of node into array NTT for all displacement variables.

FXLD 38 : copy a further code (for excess pore pressure) only if node has pore pressure variable. Note that irrespective of whether the node has displacement variables or not, the last location is always reserved for the pore pressure variable.

FXLD 43 : loop on all d.o.f. of node (making last appearance).

FXLD 45 : obtain fixity code of d.o.f.

FXLD 48 : if node has only one d.o.f. it is assumed that this is the pore pressure variable.

FXLD 49–51 : set up indexes.

FXLD 52 : by-pass if d.o.f. is *free*.

FXLD 53–56 : if *fixed*, add a large number ALAR to the diagonal term.

FXLD 61 : if d.o.f. is not the last d.o.f. of node (it is to trap the pore pressure variable).

FXLD 62 : by-pass if fixity code is not 2 (option to specify *absolute* value of excess pore pressure).

FXLD 63–64 : fixity code 2 is replaced by 1 (load ratio is set to 1).

FXLD 65 : calculate the incremental change in excess pore pressure.

FXLD 66 : add the prescribed value multiplied by a large number ALAR to the RHS load term.

FXLD 69–76 : enter details of prescribed d.o.f. in output buffer.

FXLD 77–81 : print out contents of buffer, if it is full, and then reset pointer.

FXLD 82 : by-pass if not the first d.o.f. or the last (presumably pore pressure) d.o.f. of node.

FXLD 83–84 : by-pass if d.o.f. has not a fixity code 2; otherwise reset incremental value (of excess pore pressure) to zero. This ensures that no further change in pore pressure takes place, in the rest of the analysis.

FXLD 85 : end of loop on all d.o.f. of node.
FXLD 87–89 : calculate index of load term in active front region.
FXLD 91 : loop on all d.o.f. of node.
FXLD 93–94 : enter load term in active load region in front.
FXLD 95 : branch off if load has negligible value.
FXLD 98–102 : enter load term in output buffer (only if it is of significant value, i.e. > 1.E-20). This is to ensure only loads of significant magnitude are printed.
FXLD 103–106 : print contents of buffer if it is full and reset pointer.
FXLD 107 : end of loop on all d.o.f. of node.

Subroutine **PRINTF** can be called to print out the contents of the frontal region at different stages of the solution, i.e. after assembling an element stiffness matrix and after elimination of the variables making their last appearance. The output produced can be substantial, and for normal runs it is not recommended to do this. Hence all calls to PRINTF have been commented out. But is is useful for debugging purposes.

Routine PRINTF

```
      SUBROUTINE PRINTF(IW6,ELPA,MFZ,NR,RHS,NRHS,IOPT)                  PRNT   1
C*****************************************************************************PRNT   2
C     PRINTS OUT UPPER TRIANGULAR MATRIX                                 PRNT   3
C*****************************************************************************PRNT   4
      CHARACTER*4 IFORM,IG                                               PRNT   5
      DIMENSION RHS(NRHS),BUFF(10),IFORM(4),IG(10),ELPA(MFZ)             PRNT   6
      DATA IFORM(2),IFORM(3),IFORM(4)/'X,10','E12.','4)  '/              PRNT   7
      DATA IG(1),IG(2),IG(3),IG(4),IG(5),IG(6),IG(7),IG(8),IG(9),IG(10)/PRNT   8
     1 '  (1',' (13',' (25',' (37',' (49',' (61',' (73',' (85',         PRNT   9
     2 ' (97','(109'/                                                   PRNT  10
C                                                                       PRNT  11
      IF(IOPT.EQ.1)WRITE(IW6,900)                                       PRNT  12
      IF(IOPT.EQ.2)WRITE(IW6,901)                                       PRNT  13
      IF(NRHS.EQ.0) RETURN                                              PRNT  14
C                                                                       PRNT  15
      NSUB=(NR+9)/10                                                    PRNT  16
C                                                                       PRNT  17
      DO 20 JJ=1,NSUB                                                   PRNT  18
      J2=10*JJ                                                          PRNT  19
      J1=J2-9                                                           PRNT  20
      IF(J2.GT.NR)J2=NR                                                 PRNT  21
      WRITE(IW6,904) J1,J2                                              PRNT  22
C                                                                       PRNT  23
      DO 18 II=1,JJ                                                     PRNT  24
      IFORM(1)=IG(1)                                                    PRNT  25
      I2=10*II                                                          PRNT  26
      I1=I2-9                                                           PRNT  27
      IF(I2.GT.NR)I2=NR                                                 PRNT  28
      WRITE(IW6,905)I1,I2                                               PRNT  29
C                                                                       PRNT  30
      DO 16 I=I1,I2                                                     PRNT  31
      JI=0                                                              PRNT  32
      IF(JJ.GT.II) GOTO 12                                              PRNT  33
      J1=I                                                              PRNT  34
      IF=I-I1+1                                                         PRNT  35
      IFORM(1)=IG(IF)                                                   PRNT  36
   12 DO 14 J=J1,J2                                                     PRNT  37
```



```
      JI=JI+1                                                           PRNT  38
      IJ=J*(J-1)/2+I                                                    PRNT  39
   14 BUFF(JI)=ELPA(IJ)                                                 PRNT  40
      WRITE(IW6,IFORM)(BUFF(K),K=1,JI)                                  PRNT  41
   16 CONTINUE                                                          PRNT  42
C                                                                       PRNT  43
   18 CONTINUE                                                          PRNT  44
C                                                                       PRNT  45
   20 CONTINUE                                                          PRNT  46
C                                                                       PRNT  47
      WRITE(IW6,910)(RHS(K),K=1,NR)                                     PRNT  48
      RETURN                                                            PRNT  49
  900 FORMAT(1X,17HELEMENT STIFFNESS)                                   PRNT  50
  901 FORMAT(1X,7HGRANDPA)                                              PRNT  51
  904 FORMAT(8H COLUMNS,I4,3H TO,I4)                                    PRNT  52
  905 FORMAT(5H ROWS,I4,3H TO,I4)                                       PRNT  53
  910 FORMAT(4H RHS/(1X,10E12.4))                                       PRNT  54
      END                                                               PRNT  55
```

PRNT 16 : split the matrix into **segments** of sub-matrices, each 10 X 10 in size, convenient for printing.
PRNT 18 : loop on all column segments.
PRNT 20–21 : first and last column numbers in segment JJ; reset J2 to last d.o.f. NRHS, if it exceeds NRHS.
PRNT 22 : write column numbers.
PRNT 24 : loop on all row segments (up to diagonal segment).
PRNT 25 : select format type (to retain triangular shape of stiffness matrix in output).
PRNT 26–27 : first and last rows in segment II, JJ.
PRNT 28 : reset I2 to last d.o.f. if it exceeds NRHS.
PRNT 29 : write row numbers.
PRNT 31 : loop on all rows in segment.
PRNT 32 : counter of terms in a row in segment (the output is row by row within segment).
PRNT 33 : if above diagonal, retain format type to print full square sub-matrix (not triangular).
PRNT 34 : diagonal segment; start row with diagonal term.
PRNT 35–36 : new format type to retain triangular shape of matrix in output.
PRNT 37 : loop on all terms in row I.
PRNT 40 : enter stiffness term in output buffer.
PRNT 41 : output a row of terms in segment.
PRNT 42 : end of loop on all columns in segment.
PRNT 44 : end of loop on all row segments in a column.
PRNT 46 : end of loop on all column segments.
PRNT 48 : print out RHS vector.

During the course of the solution, if the buffer size is not sufficient to hold all the eliminated coefficients then whenever the buffer becomes full (saturated), the contents of the buffer are written to a backing store. This is carried out by routines **STOREQ** and **WRTN**.

Routine STOREQ

```
      SUBROUTINE STOREQ(ELPA,MFZ,NBAXO,IBA,NDEQN,KURPA,IW7)            STEQ   1
C*************************************************************************STEQ   2
C     WHEN SATURATED WRITES BUFFER TO DISK                               STEQ   3
C*************************************************************************STEQ   4
      DIMENSION ELPA(MFZ)                                               STEQ   5
      LREC=IBA-NBAXO                                                    STEQ   6
      CALL WRTN(IW7,ELPA(NBAXO+1),LREC)                                 STEQ   7
      WRITE(IW7) LREC                                                   STEQ   8
      IBA=NBAXO                                                         STEQ   9
      NDEQN=IBA+KURPA+4                                                 STEQ  10
      RETURN                                                            STEQ  11
      END                                                               STEQ  12
```

STEQ 6 : calculate length of record (number of terms to be written to backing store).
STEQ 7 : write to the backing store the contents of buffer.
STEQ 8 : write the length of record to backing store.
STEQ 9 : reset pointer of last location used in buffer to the first position in buffer (as buffer is now empty).
STEQ 10 : pointer of last entry from next equation.

Routine WRTN

```
      SUBROUTINE WRTN(N,A,M)                                            WRTN   1
C*************************************************************************WRTN   2
C     WRITES ONE DIMESNIONAL ARRAY                                      WRTN   3
C*************************************************************************WRTN   4
      DIMENSION A(M)                                                    WRTN   5
      WRITE(N) A                                                        WRTN   6
      RETURN                                                            WRTN   7
      END                                                               WRTN   8
```

WRTN 6 : write a one-dimensional REAL array of given size M to unit N.

Later, during back-substitution, the reverse process takes place. If eliminated coefficients have been written to the backing store then, while back-substituting, if the buffer becomes empty, the next set of entries is retrieved from the backing store. This task is carried out by the routines **GETEQN** and **RDN**.

Routine GETEQN

```
      SUBROUTINE GETEQN(ELPA,MFZ,NBAXO,IBA,IW7)                         GTQN   1
C*************************************************************************GTQN   2
C     READS BUFFERFUL WHEN BACK-SUBSTITUTING                            GTQN   3
C*************************************************************************GTQN   4
      DIMENSION ELPA(MFZ)                                               GTQN   5
C                                                                       GTQN   6
      BACKSPACE IW7                                                     GTQN   7
      READ (IW7) LREC                                                   GTQN   8
      BACKSPACE IW7                                                     GTQN   9
      BACKSPACE IW7                                                     GTQN  10
      CALL RDN(IW7,ELPA(NBAXO+1),LREC)                                  GTQN  11
      BACKSPACE IW7                                                     GTQN  12
      IBA=NBAXO+LREC                                                    GTQN  13
      RETURN                                                            GTQN  14
      END                                                               GTQN  15
```


GTQN 8 : read no. of terms in backing store (length of record).
GTQN 11 : read back from backing store a bufferful of coefficients.
GTQN 13 : reset pointer to end of buffer (i.e. the buffer is now full).

Routine RDN

```
      SUBROUTINE RDN(N,A,M)                                             RDNM   1
C*************************************************************************RDNM   2
C     READ ONE DIMENSIONAL ARRAY                                        RDNM   3
C*************************************************************************RDNM   4
      DIMENSION A(M)                                                    RDNM   5
      READ(N) A                                                         RDNM   6
      RETURN                                                            RDNM   7
      END                                                               RDNM   8
```

RDNM 6 : read from a file N, a one-dimensional REAL array of length M.

8.13 CALCULATION OF OUTPUT PARAMETERS

The following set of calculations is now carried out. Before that, a number of arrays containing tables to be printed are allocated store dynamically in the region previously used for solving the equations. This is carried out by routine **UPARAL**.

Routine UPARAL

```
      SUBROUTINE UPARAL(TTIME,TGRAV,IOUT,NN,ND,NNOD1,NEL,NDF,NTPE,NIP,  UPAR   1
     1 NPT,NSP,NPL,NDZ,NVRS,NVRN,NDIM,MUMAX,NNZ,NDMX,NPMX,NS,NB,NL,INXL,UPAR   2
     2   NPR,NMT,MXEN,XYZ,DI,DA,VARINT,P,PT,PCOR,PEQT,XYFT,STR,PEXI,    UPAR   3
     3   PCONI,D,ELCOD,DS,SHFN,CARTD,B,FT,AA,NCONN,MAT,LTYP,MREL,MRELVV,UPAR   4
     4   NREL,NW,NQ,JEL,IDFX,NP1,NP2,NWL,NMOD,CIP,LL,PR,                UPAR   5
     5   NTY,A,MFZ,ICOR,IUPD,FRACT,JS,IWL)                             UPAR   6
C*************************************************************************UPAR   7
C     ROUTINE TO ALLOCATE ARRAY STORE FOR USE IN UPOUT                  UPAR   8
C*************************************************************************UPAR   9
      REAL LL                                                           UPAR  10
C--------USE THE FOLLOWING STATEMENT AFTER CONVERTING PROGRAM TO DOUBLE UPAR  11
C--------PRECISION.  ARRAY A ALWAYS USES ONE NUMERIC STORAGE LOCATION   UPAR  12
CC     REAL A                                                           UPAR  13
      DIMENSION XYZ(NDIM,NN),DI(NDF),DA(NDF),VARINT(NVRS,NIP,NEL),      UPAR  14
     1 P(NDF),PT(NDF),PCOR(NDF),PEQT(NDF),XYFT(NDF),                   UPAR  15
     2 STR(NVRN,NIP,NEL),PEXI(NDF),PCONI(NDF)                          UPAR  16
      DIMENSION D(NS,NS),ELCOD(NDIM,NDMX),DS(NDIM,NDMX),SHFN(NDMX),     UPAR  17
     1 CARTD(NDIM,NDMX),B(NS,NB),FT(NDIM,NDMX),AA(NPMX)                UPAR  18
      DIMENSION NCONN(NTPE,NEL),MAT(NEL),LTYP(NEL),MRELVV(NEL),         UPAR  19
     1 MREL(MUMAX),NREL(NNZ),NW(NNOD1),NQ(NN),JEL(NEL),                UPAR  20
     2 IDFX(NDF),NP1(NPL),NP2(NPL),NWL(NPMX),NMOD(NIP,NEL)             UPAR  21
      DIMENSION CIP(NDIM),LL(NL),PR(NPR,NMT),NTY(NMT),A(MFZ)            UPAR  22
      COMMON /DEVICE/ IR1,IR4,IR5,IW2,IW4,IW6,IW7,IW8,IW9               UPAR  23
      COMMON /PRECSN/ NP                                                UPAR  24
C                                                                       UPAR  25
C----------MAXIMUM NUMBER OF CAM-CLAY STRESS OUTPUT PARAMETERS          UPAR  26
      NCV=10                                                            UPAR  27
```

```
C=====================================================================UPAR  28
C     INDEXES ARE FOR ARRAYS USED IN PRINTING OUT CAM-CLAY            UPAR  29
C     STRESS PARAMETERS IN SUBROUTINE UPOUT2                          UPAR  30
C=====================================================================UPAR  31
C     A(1) - A(M1-1) = CAM-CLAY STRESS PARAMETERS......VARC(NCV,NIP,NEL)UPAR  32
C    A(M1) - A(M2-1) = CODE TO INDICATE STRESS STATE.....NCODE(NIP,NEL)UPAR  33
C    A(M2) - A(M3-1) = INDICATORS OF APPROACHING CS.....LCS(NIP,NEL)UPAR  34
C    A(M3) - A(M4-1) = INDICATORS OF NEGATIVE P'.....LNGP(NIP,NEL)UPAR  35
C    A(M4) - A(M5-1) = ELEMENT PROGRAM NUMBERS.....NELPR(NEL)UPAR  36
C    A(M5) - A(M6-A) = ELEMENT    USER NUMBERS.....NELUS(NEL)UPAR  37
C    A(M6) - A(M7-1) = INDICATOR OF ELEMENTS WITH CAM-CLAY....NELCM(NEL)UPAR  38
C    A(M7) - A(M8-1) = INDICATOR OF ELEMENENTS APPROACH CS.....MCS(NEL)UPAR  39
C    A(M8) - A(M9-1) = INDICATOR OF ELEMENTS WITH NEGATIVE P'..MNGP(NEL)UPAR  40
C---------------------------------------------------------------------UPAR  41
      NIEL=NIP*NEL                                                    UPAR  42
      M1=NCV*NIEL*NP+1                                                UPAR  43
      M2=M1+NIEL                                                      UPAR  44
      M3=M2+NIEL                                                      UPAR  45
      M4=M3+NIEL                                                      UPAR  46
      M5=M4+NEL                                                       UPAR  47
      M6=M5+NEL                                                       UPAR  48
      M7=M6+NEL                                                       UPAR  49
      M8=M7+NEL                                                       UPAR  50
      M9=M8+NEL                                                       UPAR  51
      LZ=M9                                                           UPAR  52
      IF(LZ.LE.MFZ)GO TO 10                                           UPAR  53
      WRITE(IW6,901)LZ,MFZ                                            UPAR  54
  901 FORMAT(36H ALLOCATED STORE EXCEEDED; REQUIRED ,I7,              UPAR  55
     1 2X,9HALLOCATED,2X,I7,2X,15H(ROUTINE UPOUT))                   UPAR  56
      STOP                                                            UPAR  57
C                                                                     UPAR  58
   10 WRITE(IW6,902)LZ,MFZ                                            UPAR  59
  902 FORMAT(/34H ARRAY STORE USED IN ROUTINE UPOUT,I8,               UPAR  60
     1 2X,17HOUT OF ALLOCATED ,I7/)                                   UPAR  61
      CALL UPOUT(TTIME,TGRAV,IOUT,NN,ND,NNOD1,NEL,NDF,NTPE,NIP,       UPAR  62
     1 NPT,NSP,NPL,NDZ,NVRS,NVRN,NDIM,MUMAX,NNZ,NDMX,NPMX,NS,NB,NL,INXL,UPAR  63
     2   NPR,NMT,MXEN,XYZ,DI,DA,VARINT,P,PT,PCOR,PEQT,XYFT,STR,PEXI,  UPAR  64
     3   PCONI,D,ELCOD,DS,SHFN,CARTD,B,FT,AA,NCONN,MAT,LTYP,MREL,MRELVV,UPAR  65
     4   NREL,NW,NQ,JEL,IDFX,NP1,NP2,NWL,NMOD,CIP,LL,PR,             UPAR  66
     5   NTY,A,MFZ,ICOR,IUPD,FRACT,JS,IWL,NCV,                       UPAR  67
     6   A(1),A(M1),A(M2),A(M3),A(M4),A(M5),A(M6),                   UPAR  68
     7   A(M7),A(M8))                                                UPAR  69
      RETURN                                                          UPAR  70
      END                                                             UPAR  71
```

UPAR 42–52 : calculate indexes to various arrays to assign them store in array A.

UPAR 53–56 : if size of A is insufficient then print message.

UPAR 62–69 : calculate incremental strains and stresses and print out results.

The unknown **incremental** displacements (this being an incremental method) and excess pore pressures are solved for and placed in DI(NDF). In routine **UPOUT** the total displacements are calculated by updating DA(NDF) by DI(NDF). Then the incremental **strains** are calculated.

$$\Delta\epsilon = \mathbf{B}\,\Delta\mathbf{a}. \tag{8.8}$$

The cumulative strains in STR(NVRN, NIP, NEL) are incremented by the incremental strains. The incremental stresses are then calculated from the incremental strains.

$$\Delta\sigma_i' = \mathbf{D}_{ep}\,\Delta\epsilon, \tag{8.9}$$

$$\sigma_i' = \sigma_{i-1}' + \Delta\sigma_i'. \tag{8.10}$$

The **B** and **D** matrices calculated at this stage are essentially the same as when they were calculated for the element stiffness matrix. This fact is again made use of in some programs by writing the **B** and **D** matrices to a file, element by element, and reading them back. The main reason for not doing this in CRISP is that the order the elements are called in the solution routines can be different from the order the results are printed at the output stage.

The current element stresses in VARINT(NVRS, NIP, NEL) are then updated. The nodal loads equivalent to current stresses are calculated according to

$$\text{PEQT(NDF)} = \int_V \mathbf{B}^{\mathrm{T}}\sigma_i \text{ d (vol)}. \tag{8.11}$$

Also calculated from the boundary stresses and self-weight loading (both external) is a set of nodal loads,

$$\text{PT(NDF)} = \int_V \mathbf{N}^{\mathrm{T}}\mathbf{w} \text{ d (vol)} + \int_S \mathbf{N}^{\mathrm{T}}\tau \text{ d (area)}. \tag{8.12}$$

In order to satisfy the equilibrium condition, PT = PEQT. This is known as the equilibrium check. The procedure is to calculate the difference as PCOR = PT − PEQT. The percentage error is defined as

$$\%\text{ error} = \frac{|\text{PCOR}|_{\max}}{|\text{PT}|_{\max}}. \tag{8.13}$$

This completes the increment.

Routine UPOUT

```
      SUBROUTINE UPOUT(TTIME,TGRAV,IOUT,NN,ND,NNOD1,NEL,NDF,NTPE,NIP,    UOUT   1
     1 NPT,NSP,NPL,NDZ,NVRS,NVRN,NDIM,MUMAX,NNZ,NDMX,NPMX,NS,NB,NL,INXL,UOUT   2
     2   NPR,NMT,MXEN,XYZ,DI,DA,VARINT,P,PT,PCOR,PEQT,XYFT,STR,PEXI,     UOUT   3
     3   PCONI,D,ELCOD,DS,SHFN,CARTD,B,FT,AA,NCONN,MAT,LTYP,MREL,MRELVV,UOUT   4
     4   NREL,NW,NQ,JEL,IDFX,NP1,NP2,NWL,NMOD,CIP,LL,PR,                 UOUT   5
     5   NTY,A,MFZ,ICOR,IUPD,FRACT,JS,IWL,NCV,                           UOUT   6
     6   VARC,NCODE,LCS,LNGP,NELPR,NELUS,NELCM,MCS,MNGP)                 UOUT   7
C*********************************************************************UOUT   8
C     UPDATE  AND  OUTPUT  ROUTINE                                      UOUT   9
C*********************************************************************UOUT  10
      REAL L,LL                                                        UOUT  11
      INTEGER TF                                                       UOUT  12
      DIMENSION XYZ(NDIM,NN),DI(NDF),DA(NDF),VARINT(NVRS,NIP,NEL),     UOUT  13
     1 P(NDF),PT(NDF),PCOR(NDF),PEQT(NDF),XYFT(NDF),                  UOUT  14
     2 STR(NVRN,NIP,NEL),PEXI(NDF),PCONI(NDF)                         UOUT  15
      DIMENSION D(NS,NS),ELCOD(NDIM,NDMX),DS(NDIM,NDMX),SHFN(NDMX),    UOUT  16
     1 CARTD(NDIM,NDMX),B(NS,NB),FT(NDIM,NDMX),AA(NPMX)               UOUT  17
      DIMENSION NCONN(NTPE,NEL),MAT(NEL),LTYP(NEL),MRELVV(NEL),        UOUT  18
     1 MREL(MUMAX),NREL(NNZ),NW(NNOD1),NQ(NN),JEL(NEL),               UOUT  19
     2 IDFX(NDF),NP1(NPL),NP2(NPL),NWL(NPMX),NMOD(NIP,NEL)            UOUT  20
      DIMENSION CIP(NDIM),LL(NL),PR(NPR,NMT),NTY(NMT),A(MFZ)           UOUT  21
      DIMENSION VARC(NCV,NIP,NEL),MCS(NEL),MNGP(NEL)                   UOUT  22
      DIMENSION LCS(NIP,NEL),LNGP(NIP,NEL),NCODE(NIP,NEL)              UOUT  23
```

```
      DIMENSION NELPR(NEL),NELUS(NEL),NELCM(NEL)                        UOUT  24
      DIMENSION ST(6),VARO(6),SS(6),SPA(3),SST(6),ED(2)                 UOUT  25
      COMMON /DATL  / L(4,100)                                          UOUT  26
      COMMON /DATW  / W(100)                                            UOUT  27
      COMMON /FLOW  / NPLAX                                             UOUT  28
      COMMON /ELINF / LINFO(50,15)                                      UOUT  29
      COMMON /FIX   / DXYT(4,200),MF(200),TF(4,200),NF                  UOUT  30
      COMMON /PRSLD / PRESLD(10,100),LEDG(100),NDE1(100),NDE2(100),NLED UOUT  31
      COMMON /DEVICE/ IR1,IR4,IR5,IW2,IW4,IW6,IW7,IW8,IW9               UOUT  32
      COMMON /PARS  / PYI,ALAR,ASMVL,ZERO                               UOUT  33
      COMMON /COUNT / NCS,NNGP                                          UOUT  34
      COMMON /OUT   / IBC,IRAC,NVOS,NVOF,NMOS,NMOF,NELOS,NELOF,ISR      UOUT  35
      COMMON /JACB  / XJACI(3,3),DJACB                                  UOUT  36
C                                                                       UOUT  37
      ISTGE=4                                                           UOUT  38
      LED=2                                                             UOUT  39
      NS1=NS+1                                                          UOUT  40
      NDIM1=NDIM+1                                                      UOUT  41
C-----------------------------------------------------------------------UOUT  42
C     BREAK OUTPUT CODE                                                 UOUT  43
C-----------------------------------------------------------------------UOUT  44
      IOUT4=IOUT/1000                                                   UOUT  45
      IOUT3=(IOUT-1000*IOUT4)/100                                       UOUT  46
      IOUT2=(IOUT-1000*IOUT4-100*IOUT3)/10                              UOUT  47
      IOUT1=(IOUT-1000*IOUT4-100*IOUT3-10*IOUT2)                        UOUT  48
      IF(IOUT1.LT.1)GOTO 4                                              UOUT  49
      LT1=LTYP(1)                                                       UOUT  50
      LT1=IABS(LT1)                                                     UOUT  51
      GOTO(1,1,2,1,2,1,2,1,2,1,2),LT1                                   UOUT  52
    1 WRITE(IW6,902)                                                    UOUT  53
      GOTO 4                                                            UOUT  54
    2 WRITE(IW6,901)                                                    UOUT  55
C-----------------------------------------------------------------------UOUT  56
C     UPDATE  ABSOLUTE  DISPLACEMENTS                                   UOUT  57
C-----------------------------------------------------------------------UOUT  58
    4 CR=1.                                                             UOUT  59
      IF(NPLAX.EQ.1)CR=2.*PYI                                           UOUT  60
C                                                                       UOUT  61
      DO 5 KD=1,NDF                                                     UOUT  62
    5 DA(KD)=DA(KD)+DI(KD)                                              UOUT  63
C                                                                       UOUT  64
      DO 10 JR=1,NNZ                                                    UOUT  65
      IF(NREL(JR).EQ.0)GOTO 10                                          UOUT  66
      J=NREL(JR)                                                        UOUT  67
      NQL=NQ(J)                                                         UOUT  68
      IF(NQL.EQ.0) GOTO 10                                              UOUT  69
      N1=NW(J)                                                          UOUT  70
      IF(IOUT1.EQ.0)GOTO 10                                             UOUT  71
      IF(IOUT1.EQ.1.AND.JR.GT.NDZ)GOTO 10                               UOUT  72
      IF(JR.LT.NDZ)GOTO 6                                               UOUT  73
      IF(JR.LT.NMOS.OR.JR.GT.NMOF)GOTO 10                               UOUT  74
      GOTO 8                                                            UOUT  75
    6 CONTINUE                                                          UOUT  76
      IF(JR.LT.NVOS.OR.JR.GT.NVOF)GOTO 10                               UOUT  77
    8 CONTINUE                                                          UOUT  78
C-----------------------------------------------------------------------UOUT  79
C     OUTPUT  DISPLACEMENTS                                             UOUT  80
C-----------------------------------------------------------------------UOUT  81
      N2=N1+NQL-1                                                       UOUT  82
      IF(NQL.EQ.3)WRITE(IW6,900)JR,(DI(JJ),JJ=N1,N2),(DA(JJ),JJ=N1,N2)  UOUT  83
      IF(NQL.EQ.2)WRITE(IW6,910)JR,(DI(JJ),JJ=N1,N2),(DA(JJ),JJ=N1,N2)  UOUT  84
      IF(NQL.EQ.1)WRITE(IW6,911)JR,DI(N1),DA(N1)                        UOUT  85
   10 CONTINUE                                                          UOUT  86
      IF(IOUT2.EQ.2)WRITE(IW6,904)                                      UOUT  87
      IF(IOUT2.EQ.1)WRITE(IW6,906)                                      UOUT  88
C                                                                       UOUT  89
```



```
      CALL ZEROR1(PT,NDF)                                               UOUT  90
      CALL ZEROR1(PEQT,NDF)                                             UOUT  91
C-----------------------------------------------------------------------UOUT  92
C     INITIALISE                                                        UOUT  93
C-----------------------------------------------------------------------UOUT  94
      DO 18 IM=1,NEL                                                    UOUT  95
      MCS(IM)=0                                                         UOUT  96
      MNGP(IM)=0                                                        UOUT  97
      NELPR(IM)=0                                                       UOUT  98
      NELCM(IM)=0                                                       UOUT  99
      NELUS(IM)=0                                                       UOUT 100
      DO 18 IP=1,NIP                                                    UOUT 101
      LCS(IP,IM)=0                                                      UOUT 102
   18 LNGP(IP,IM)=0                                                     UOUT 103
C-----------------------------------------------------------------------UOUT 104
C     CALCULATE  STRESSES  AND  STRAINS  IN  ELEMENTS                   UOUT 105
C     IEL COUNTER - NO. OF ELEMENTS PROCESSED                           UOUT 106
C-----------------------------------------------------------------------UOUT 107
      IEL=0                                                             UOUT 108
      NCAM=0                                                            UOUT 109
C                                                                       UOUT 110
      DO 200 MR=1,MUMAX                                                 UOUT 111
      CALL ZEROR1(SST,NS)                                               UOUT 112
CC    IWRN=0                                                            UOUT 113
      ICAM=0                                                            UOUT 114
      IELST=0                                                           UOUT 115
      J=MREL(MR)                                                        UOUT 116
      IF(J.EQ.0)GOTO 200                                                UOUT 117
      LT=LTYP(J)                                                        UOUT 118
      IF(LT.LT.0)GOTO 200                                               UOUT 119
      NDN=LINFO(5,LT)                                                   UOUT 120
      NGP=LINFO(11,LT)                                                  UOUT 121
      INDX=LINFO(12,LT)                                                 UOUT 122
      NPN=LINFO(6,LT)                                                   UOUT 123
      NDPT=LINFO(1,LT)                                                  UOUT 124
      NAC=LINFO(15,LT)                                                  UOUT 125
C-----------------------------------------------------------------------UOUT 126
C     SETUP LOCAL NODAL COORDINATES OF ELEMENT                          UOUT 127
C-----------------------------------------------------------------------UOUT 128
      DO 20 KN=1,NDN                                                    UOUT 129
      NDE=NCONN(KN,J)                                                   UOUT 130
      DO 20 ID=1,NDIM                                                   UOUT 131
   20 ELCOD(ID,KN)=XYZ(ID,NDE)                                          UOUT 132
C                                                                       UOUT 133
      GOTO(25,25,23,25,23,25,23,25,23,25,23),LT                         UOUT 134
C-----------------------------------------------------------------------UOUT 135
C     SETUP LOCAL ARRAY OF NW AS NWL GIVING THE INDEX TO                UOUT 136
C     PORE-PRESSURE VARIABLES                                           UOUT 137
C-----------------------------------------------------------------------UOUT 138
   23 IPP=0                                                             UOUT 139
      DO 24 IV=1,NDPT                                                   UOUT 140
      IQ=LINFO(IV+INXL,LT)                                              UOUT 141
      IF(IQ.NE.NDIM1.AND.IQ.NE.1)GOTO 24                                UOUT 142
      IPP=IPP+1                                                         UOUT 143
      NDE=NCONN(IV,J)                                                   UOUT 144
      NWL(IPP)=NW(NDE)+IQ-1                                             UOUT 145
   24 CONTINUE                                                          UOUT 146
   25 IF(IOUT2.NE.2)GOTO 26                                             UOUT 147
      IF(MR.GE.NELOS.AND.MR.LE.NELOF)WRITE(IW6,908)MR                   UOUT 148
      IF(MR.GE.NELOS.AND.MR.LE.NELOF)WRITE(IW6,914)                     UOUT 149
   26 KM=MAT(J)                                                         UOUT 150
      KGO=NTY(KM)                                                       UOUT 151
      IF(NTY(KM)-2)27,28,28                                             UOUT 152
   27 CALL  DCON(J,0,NEL,NDIM,NS,NPR,NMT,MAT,PR,D,BK)                   UOUT 153
      IELST=1                                                           UOUT 154
   28 IEL=IEL+1                                                         UOUT 155
```

```
      NELUS(IEL)=MR                                                     UOUT 156
      NELPR(IEL)=J                                                      UOUT 157
C-----------------------------------------------------------------------UOUT 158
C     INITIALISE  FT                                                    UOUT 159
C-----------------------------------------------------------------------UOUT 160
      CALL ZEROR2(FT,NDIM,NDN)                                          UOUT 161
C-----------------------------------------------------------------------UOUT 162
C     LOOP ON INTEGRATION POINTS                                        UOUT 163
C-----------------------------------------------------------------------UOUT 164
      DO 125 IP=1,NGP                                                   UOUT 165
      IPA=IP+INDX                                                       UOUT 166
C                                                                       UOUT 167
      DO 35 IL=1,NAC                                                    UOUT 168
   35 LL(IL)=L(IL,IPA)                                                  UOUT 169
C-----------------------------------------------------------------------UOUT 170
C     FORM  B  MATRIX                                                   UOUT 171
C-----------------------------------------------------------------------UOUT 172
      CALL FORMB2(J,MR,R,RI,NDIM,NDMX,NDN,NS,NB,NAC,ELCOD,DS,           UOUT 173
     1   SHFN,CARTD,B,LL,LT,IP,ISTGE)                                   UOUT 174
C                                                                       UOUT 175
      CALL ZEROR1(ST,NS)                                                UOUT 176
C                                                                       UOUT 177
      DO 44 II=1,NDN                                                    UOUT 178
      IN=NCONN(II,J)                                                    UOUT 179
      N1=NW(IN)                                                         UOUT 180
      N2=N1+1                                                           UOUT 181
      ST(1)=ST(1)+CARTD(1,II)*DI(N1)                                    UOUT 182
      ST(2)=ST(2)+CARTD(2,II)*DI(N2)                                    UOUT 183
      ST(3)=ST(3)+SHFN(II)*DI(N1)*RI                                    UOUT 184
      ST(4)=ST(4)+CARTD(1,II)*DI(N2)+CARTD(2,II)*DI(N1)                 UOUT 185
      IF(NDIM.EQ.2)GOTO 44                                              UOUT 186
      N3=N1+2                                                           UOUT 187
      ST(3)=ST(3)+CARTD(3,II)*DI(N3)                                    UOUT 188
      ST(5)=ST(5)+CARTD(3,II)*DI(N2)+CARTD(2,II)*DI(N3)                 UOUT 189
      ST(6)=ST(6)+CARTD(3,II)*DI(N1)+CARTD(1,II)*DI(N3)                 UOUT 190
   44 CONTINUE                                                          UOUT 191
C                                                                       UOUT 192
      ED(1)=EDS(STR(1,IP,J),NS,NDIM)                                    UOUT 193
C                                                                       UOUT 194
      DO 45 IS=1,NS                                                     UOUT 195
   45 STR(IS,IP,J)=STR(IS,IP,J)+ST(IS)                                  UOUT 196
      ED(2)=EDS(STR(1,IP,J),NS,NDIM)                                    UOUT 197
C-----------------------------------------------------------------------UOUT 198
C     CALCULATE  STRESSES                                               UOUT 199
C-----------------------------------------------------------------------UOUT 200
      GOTO(59,52,53,54),KGO                                             UOUT 201
   52 CALL  DLIN(IP,J,0,NEL,NDIM,NDN,NS,NPR,NMT,                        UOUT 202
     1   ELCOD,SHFN,MAT,D,PR,INDX,BK)                                   UOUT 203
      IELST=1                                                           UOUT 204
      GOTO 59                                                           UOUT 205
   53 CALL DMCAM(IP,J,0,NEL,NIP,NVRS,NDIM,NS,NPR,NMT,VARINT,MAT,D,      UOUT 206
     1     PR,BK)                                                       UOUT 207
      GOTO 58                                                           UOUT 208
   54 CALL DCAM(IP,J,0,NEL,NIP,NVRS,NDIM,NS,NPR,NMT,VARINT,MAT,D,PR,    UOUT 209
     1     ITP,BK)                                                      UOUT 210
   58 ICAM=1                                                            UOUT 211
C                                                                       UOUT 212
   59 DO 60 II=1,NS                                                     UOUT 213
      SS(II)=0.                                                         UOUT 214
C                                                                       UOUT 215
      DO 60 JJ=1,NS                                                     UOUT 216
   60 SS(II)=SS(II)+D(II,JJ)*ST(JJ)                                     UOUT 217
C-----------------------------------------------------------------------UOUT 218
C     UPDATE  ABSOLUTE  STRESSES                                        UOUT 219
C-----------------------------------------------------------------------UOUT 220
      DO 65 JJ=1,NS                                                     UOUT 221
```



```
      SST(JJ)=SST(JJ)+SS(JJ)                                            UOUT 222
      VARO(JJ)=VARINT(JJ,IP,J)                                          UOUT 223
   65 VARINT(JJ,IP,J)=VARINT(JJ,IP,J)+SS(JJ)                           UOUT 224
C-----------------------------------------------------------------------UOUT 225
C     CALCULATE  PORE  PRESSURES                                        UOUT 226
C-----------------------------------------------------------------------UOUT 227
      GOTO(70,70,66,70,66,70,66,70,66,70,66),LT                         UOUT 228
   66 CALL SHFNPP(IW6,LL,NAC,DS,AA,NDIM,NPN,LT,0,MR)                    UOUT 229
      SUM=0.                                                            UOUT 230
C                                                                       UOUT 231
      DO 68 IC=1,NPN                                                    UOUT 232
      IVR=NWL(IC)                                                       UOUT 233
   68 SUM=SUM+AA(IC)*DI(IVR)                                            UOUT 234
      V=ST(1)+ST(2)+ST(3)                                               UOUT 235
      UI=SUM                                                            UOUT 236
      GOTO 72                                                           UOUT 237
   70 V=ST(1)+ST(2)+ST(3)                                               UOUT 238
      UI=PR(7,KM)*V*BK                                                  UOUT 239
   72 VARINT(NS+1,IP,J)=VARINT(NS+1,IP,J)+UI                            UOUT 240
C                                                                       UOUT 241
      IF(KGO.NE.3.AND.KGO.NE.4)GOTO 85                                  UOUT 242
C-----------------------------------------------------------------------UOUT 243
C     CALCULATE  EXTRA  VARIABLES  FOR  CAM-CLAY  ONLY                  UOUT 244
C-----------------------------------------------------------------------UOUT 245
      CALL EVCAM(VARINT,NEL,NVRS,NDIM,NIP,IP,J,MR,KM,                   UOUT 246
     1   IEL,NS,NPR,NMT,PR,NTY,NCAM,V,NCODE,LCS,LNGP,                   UOUT 247
     2   MCS,MNGP,NELCM,VARC,NGP,ED,LED)                                UOUT 248
C                                                                       UOUT 249
   85 CALL STRSEQ(J,IP,IPA,NVRS,NIP,NEL,NDN,NDIM,NS,                    UOUT 250
     1   VARINT,SHFN,CARTD,FT,DJACB,R,RI,CR)                            UOUT 251
C-----------------------------------------------------------------------UOUT 252
C     OUTPUT ABSOLUTE STRESSES                                          UOUT 253
C-----------------------------------------------------------------------UOUT 254
      CALL PRINC(VARINT(1,IP,J),VARINT(2,IP,J),VARINT(4,IP,J),SPA)      UOUT 255
      IF(IOUT2.EQ.0)GOTO 125                                            UOUT 256
      IF(IOUT2.EQ.1)GOTO 120                                            UOUT 257
      IKM=IP                                                            UOUT 258
      GOTO 122                                                          UOUT 259
  120 IF(IOUT2.NE.1.OR.IP.NE.NGP)GOTO 125                               UOUT 260
      IKM=MR                                                            UOUT 261
C                                                                       UOUT 262
  122 DO 124 ID=1,NDIM                                                  UOUT 263
      SUM=ZERO                                                          UOUT 264
C                                                                       UOUT 265
      DO 123 IN=1,NDN                                                   UOUT 266
  123 SUM=SUM+SHFN(IN)*ELCOD(ID,IN)                                     UOUT 267
  124 CIP(ID)=SUM                                                       UOUT 268
      IF(MR.LT.NELOS.OR.MR.GT.NELOF)GOTO 125                            UOUT 269
      WRITE(IW6,916)IKM,(CIP(ID),ID=1,NDIM),                            UOUT 270
     1 (VARINT(IK,IP,J),IK=1,NS1),(SPA(JL),JL=1,3)                      UOUT 271
  125 CONTINUE                                                          UOUT 272
C-----------------------------------------------------------------------UOUT 273
C     ASSEMBLE EQUILIBRATING NODAL FORCES INTO GLOBAL ARRAY - PEQT      UOUT 274
C-----------------------------------------------------------------------UOUT 275
      DO 150 IK=1,NDN                                                   UOUT 276
      II=NCONN(IK,J)                                                    UOUT 277
      N1=NW(II)-1                                                       UOUT 278
C                                                                       UOUT 279
      DO 150 ID=1,NDIM                                                  UOUT 280
  150 PEQT(N1+ID)=PEQT(N1+ID)+FT(ID,IK)                                 UOUT 281
  200 CONTINUE                                                          UOUT 282
C-----------------------------------------------------------------------UOUT 283
C     OUTPUT ADDITIONAL PARAMETERS AND WARNING MESSAGES                 UOUT 284
C     FOR CAM-CLAYS                                                     UOUT 285
C-----------------------------------------------------------------------UOUT 286
      CALL UPOUT2(IW6,NEL,NIP,LTYP,MAT,NCAM,IOUT3,IEL,                  UOUT 287
```

```
     1   NCODE,LCS,LNGP,NELPR,NELUS,NELCM,MCS,MNGP,VARC)                    UOUT 288
C-----------------------------------------------------------------------UOUT 289
C     UPDATE  NODAL  CO-ORDINATES                                         UOUT 290
C-----------------------------------------------------------------------UOUT 291
      IF(IUPD.EQ.0)GOTO 225                                               UOUT 292
      WRITE(IW6,926)                                                      UOUT 293
C                                                                         UOUT 294
      DO 220 J=1,ND                                                       UOUT 295
      N1=NW(J)-1                                                          UOUT 296
C                                                                         UOUT 297
      DO 220 ID=1,NDIM                                                    UOUT 298
  220 XYZ(ID,J)=XYZ(ID,J)+DI(N1+ID)                                       UOUT 299
  225 CONTINUE                                                            UOUT 300
C-----------------------------------------------------------------------UOUT 301
C     OUTPUT EQUILIBRIUM AND OUT-OF-BALANCE NODAL LOADS                   UOUT 302
C-----------------------------------------------------------------------UOUT 303
      DO 230 IM=1,NDF                                                     UOUT 304
  230 PEQT(IM)=PEQT(IM)+PEXI(IM)                                          UOUT 305
C                                                                         UOUT 306
C----------CODE TO INDICATE STAGE OF THE ANALYSIS                         UOUT 307
      KSTGE=4                                                             UOUT 308
      CALL EQLOD(IW6,NN,NEL,NDF,NNOD1,NTPE,NDIM,MUMAX,NNZ,NDZ,            UOUT 309
     1   NPR,NMT,NDMX,NL,NPL,NCONN,MAT,LTYP,MRELVV,MREL,NREL,             UOUT 310
     2   NW,NQ,JEL,IDFX,NP1,NP2,XYZ,P,PT,PCOR,PEQT,XYFT,PCONI,ELCOD,       UOUT 311
     3   DS,SHFN,FT,LL,PR,NPT,NSP,MXEN,IOUT4,                             UOUT 312
     4   ICOR,TGRAV,IRAC,FRACT,KSTGE)                                     UOUT 313
C-----------------------------------------------------------------------UOUT 314
C     WRITE RESULTS ON SAVE FILE                                          UOUT 315
C-----------------------------------------------------------------------UOUT 316
      IF(ISR.EQ.0)GOTO 250                                                UOUT 317
      IF(ISR.EQ.2)GOTO 240                                                UOUT 318
      IF(ISR.EQ.1.AND.IWL.EQ.1)GOTO 240                                   UOUT 319
      GOTO 250                                                            UOUT 320
  240 WRITE(IW2) TTIME,TGRAV,XYZ,VARINT,STR,DA,XYFT,PCOR,PCONI,LTYP,NMODUOUT 321
      WRITE(IW2) NF,MF,TF,DXYT                                            UOUT 322
      WRITE(IW2) NLED,LEDG,NDE1,NDE2,PRESLD                               UOUT 323
C                                                                         UOUT 324
  250 CONTINUE                                                            UOUT 325
      RETURN                                                              UOUT 326
  900 FORMAT(1X,I5,6E15.5)                                                UOUT 327
  901 FORMAT(//46H NODAL DISPLACEMENTS AND EXCESS PORE PRESSURES/         UOUT 328
     1 1X,45(1H-)//26X,11HINCREMENTAL,36X,8HABSOLUTE//                    UOUT 329
     12X,4HNODE,7X,2HDX,13X,2HDY,13X,2HDU,13X,2HDX,13X,2HDY,13X,2HDU/)    UOUT 330
  902 FORMAT(//20H NODAL DISPLACEMENTS/1X,19(1H-)//                       UOUT 331
     1 18X,11HINCREMENTAL,33X,8HABSOLUTE//                                UOUT 332
     1 2X,4HNODE,7X,2HDX,13X,2HDY,28X,2HDX,13X,2HDY/)                     UOUT 333
  904 FORMAT(//40H ABSOLUTE STRESSES AT INTEGRATION POINTS/1X,39(1H-)//)UOUT 334
  906 FORMAT(//30H STRESSES AT ELEMENT CENTROIDS/1X,29(1H-)//8H ELEMENT,UOUT 335
     1 3X,1HX,13X,1HY,11X,2HSX,11X,2HSY,11X,2HSZ,10X,3HTXY,12X,1HU,       UOUT 336
     1 10X,5HSIG-1,8X,5HSIG-2,7X,5HTH-XY)                                 UOUT 337
  908 FORMAT(/15H ELEMENT NUMBER,I5/1X,19(1H-))                           UOUT 338
  910 FORMAT(1X,I5,2E15.5,15X,2E15.5)                                     UOUT 339
  911 FORMAT(1X,I5,30X,E15.5,30X,E15.5)                                   UOUT 340
  914 FORMAT(2X,2HIP,7X,1HX,13X,1HY,11X,2HSX,11X,2HSY,11X,2HSZ,           UOUT 341
     2 10X,3HTXY,12X,1HU,10X,5HSIG-1,8X,5HSIG-2,7X,5HTH-XY)               UOUT 342
  916 FORMAT(1X,I3,9E13.5,F10.1)                                          UOUT 343
  926 FORMAT(/48H WARNING **** THE NODAL CO-ORDINATES ARE UPDATED/)       UOUT 344
      END                                                                 UOUT 345
```

UOUT 38 : ISTGE – code to indicate stage of the analysis.

UOUT 41 : NDIM1 – the maximum number of variables in any node for consolidation element.

UOUT 45–48 : break output code for different tables to be printed.

UOUT 50–55 : select appropriate title for displacement table based on first element type.

UOUT 62–63 : update total displacements/excess pore pressures.

UOUT 65 : loop on all nodes (user specified sequence).

UOUT 66 : skip if node was not used.

UOUT 67–68 : program node number, J; number of d.o.f. of node, NQL.

UOUT 69 : skip if node has no d.o.f. (probably the node does not exist in the current mesh, and disappeared when elements associated with it had been removed).

UOUT 70 : g.v.n. of first d.o.f. of node.

UOUT 71–72 : skip, depending on output option for displacement table.

UOUT 74–77 : skip printing of displacements if out of user requested nodal range.

UOUT 82 : g.v.n. of last d.o.f. of node.

UOUT 83–85 : print out node number, incremental and cumulative displacements, and excess pore pressures.

UOUT 95–103 : zero output arrays.

UOUT 108 : zero counter of number of elements processed (IEL).

UOUT 109 : zero counter of number of Cam-clay elements processed (NCAM).

UOUT 111 : loop on all elements (in the user specified sequence).

UOUT 114 : zero flag to indicate whether current element has Cam-clay properties.

UOUT 115 : as above, but for element with elastic properties.

UOUT 116 : program element number.

UOUT 117 : by-pass if element number (MR) was not used.

UOUT 118–119 : skip if element not present in current mesh.

UOUT 120–125 : element type dependent parameters.

- NDN – number of displacement nodes.
- NGP – number of integration points.
- INDX – starting index to arrays W and L.
- NPN – number of pore pressure nodes.
- NDPT – total number of nodes.
- NAC – number of area/local co-ordinates.

UOUT 129–132 : set up local array (ELCOD) of co-ordinates of displacement nodes in element.

UOUT 134 : by-pass if not a consolidation element.

UOUT 140–145 : set up local array NWL, which gives the g.v.n. of pore pressure variables.

UOUT 150 : material zone number.

UOUT 151 : material type number.

UOUT 153 : calculate **D** matrix for linear elastic material.

UOUT 154 : set flag to indicate element with elastic properties.

UOUT 155 : increment counter of number of elements processed.

UOUT 156–157 : enter user element number and program element number.

UOUT 165 : loop on all integration points.
UOUT 166 : index to arrays W and L.
UOUT 168–169 : local/area co-ordinates of integration point.
UOUT 173–174 : calculate **B** matrix.
UOUT 180–181 : g.v.n. of first and last d.o.f. of node.
UOUT 182–185 : calculate incremental strains for 2-D.

$$\Delta\boldsymbol{\epsilon} = \mathbf{B}\,\Delta\mathbf{a}.$$

UOUT 188–190 : additional components for 3-D.
UOUT 193 : calculate deviator strain at the beginning of current increment.
UOUT 195–196 : calculate total strains.
UOUT 197 : calculate deviator strain at the end of current increment.
UOUT 202–203 : calculate **D** matrix for non-homogeneous elastic model.
UOUT 206–207 : calculate **D** matrix for modified Cam-clay.
UOUT 209–210 : calculate **D** matrix for Cam-clay.
UOUT 213–217 : calculate incremental effective stresses.

$$\Delta\boldsymbol{\sigma}' = \mathbf{D}_{ep}\,\Delta\boldsymbol{\epsilon}.$$

UOUT 221–224 : store stresses at the beginning of increment in VARO and calculate current stresses.
UOUT 228 : skip if not a consolidation element (pore pressures are calculated differently).
UOUT 229 : calculate shape functions for excess pore pressures.
UOUT 232–234 : interpolate for excess pore pressure at integration point.
UOUT 235 : calculate volumetric strain.
UOUT 236 : excess pore pressure.
UOUT 238–239 : calculate volumetric strain and excess pore pressure for drained/undrained analysis.
UOUT 240 : update pore pressure.
UOUT 246–248 : calculate output parameters for Cam-clays.
UOUT 250–251 : calculate nodal loads (FT) equal to element stresses.
UOUT 255 : calculate principal stresses in x–y plane – for 2-D only.
UOUT 258 : integration point number is to be printed.
UOUT 261 : element number is to be printed.
UOUT 263–268 : calculate co-ordinates of integration point.
UOUT 270–271 : output stresses at integration point.
UOUT 272 : end of integration point loop.
UOUT 276–281 : slot nodal loads equal to element stresses (FT) in PEQT.
UOUT 282 : end of loop on elements.
UOUT 287–288 : print out additional parameters and warning messages for Cam-clays.
UOUT 295–299 : update nodal co-ordinates by incremental displacements if flag IUPD has been set to 1.

UOUT 304–305 : add (excavation) loads due to removal of elements to PEQT for equilibrium check.
UOUT 309–313 : to carry out an equilibrium check.
UOUT 321–323 : write results to file on magnetic tape or disk (to use the stop–restart facility to continue an analysis).

The rest of the output for the current increment can be divided into four parts:

(i) the nodal displacements (and excess pore pressures);
(ii) the general stresses σ_x', σ_y', σ_z', τ_{xy}, u;
(iii) the parameters for Cam-clay models;
(iv) the out-of-balance loads at the end of this increment.

Routine **EVCAM** calculates the parameters for Cam-clay models.

Routine EVCAM

```
      SUBROUTINE EVCAM(VARINT,NEL,NVRS,NDIM,NIP,IP,J,MR,KM,IEL,          EVCM   1
     1 NS,NPR,NMT,PR,NTY,NCAM,V,NCODE,LCS,LNGP,MCS,MNGP,NELCM,           EVCM   2
     2 VARC,NGP,ED,LED)                                                  EVCM   3
C*************************************************************************EVCM   4
C     CALCULATE EXTRA STRESS PARAMETERS FOR CAM-CLAYS                     EVCM   5
C*************************************************************************EVCM   6
      DIMENSION VARINT(NVRS,NIP,NEL),NELCM(NEL),PR(NPR,NMT),NTY(NMT)      EVCM   7
      DIMENSION VARC(10,NIP,NEL),MCS(NEL),MNGP(NEL)                       EVCM   8
      DIMENSION LCS(NIP,NEL),LNGP(NIP,NEL),NCODE(NIP,NEL),ED(LED)         EVCM   9
      COMMON /PARS  /PYI,ALAR,ASMVL,ZERO                                  EVCM  10
      COMMON /COUNT /NCS,NNGP                                             EVCM  11
C                                                                         EVCM  12
      U=VARINT(NS+1,IP,J)                                                 EVCM  13
      IF(IP.NE.1)GO TO 11                                                 EVCM  14
      NCS=0                                                               EVCM  15
      NNGP=0                                                              EVCM  16
   11 ICS=0                                                               EVCM  17
      INGP=0                                                              EVCM  18
      QT=Q(VARINT(1,IP,J),NS,NDIM)                                        EVCM  19
      PE=(VARINT(1,IP,J)+VARINT(2,IP,J)+VARINT(3,IP,J))/3.                EVCM  20
      EV=-V*(1.+VARINT(NS+2,IP,J))+VARINT(NS+2,IP,J)                      EVCM  21
      EE=VARINT(NS+2,IP,J)                                                EVCM  22
      PYE=VARINT(NS+3,IP,J)                                               EVCM  23
      PCO=ABS(PYE)                                                        EVCM  24
      CALL VARCAM(IP,MR,KM,ICS,INGP,IEL,NIP,NEL,NCODE,VARC,PR,NTY,        EVCM  25
     1   PE,QT,PCO,PYE,U,EV,EE,PC,ED,LED,NPR,NMT)                         EVCM  26
      VARINT(NS+3,IP,J)=PC                                                EVCM  27
      VARINT(NS+2,IP,J)=EE                                                EVCM  28
C                                                                         EVCM  29
      NCS=NCS+ICS                                                         EVCM  30
      IF(ICS.EQ.1)LCS(IP,IEL)=IP                                          EVCM  31
      NNGP=NNGP+INGP                                                      EVCM  32
      IF(INGP.EQ.1)LNGP(IP,IEL)=IP                                        EVCM  33
      CALL ANGTH(VARINT,NEL,NIP,NVRS,IP,J,THETA)                          EVCM  34
      VARC(10,IP,IEL)=THETA                                               EVCM  35
C                                                                         EVCM  36
      IF(NDIM.EQ.3)VARC(10,IP,IEL)=ZERO                                   EVCM  37
      IF(IP.NE.NGP)RETURN                                                 EVCM  38
      NCAM=NCAM+1                                                         EVCM  39
      NELCM(IEL)=1                                                        EVCM  40
      IF(NCS.NE.0)MCS(IEL)=1                                              EVCM  41
      IF(NNGP.NE.0)MNGP(IEL)=1                                            EVCM  42
      RETURN                                                              EVCM  43
      END                                                                 EVCM  44
```

EVCM 13 : pore pressure.
EVCM 14 : by-pass if not **first** integration point.
EVCM 15–16 : set counters (of integration points) to zero.
NCS – number of points approaching critical state.
NNGP – number of points with **negative** p'.
EVCM 17–18 : set identifiers to zero.
EVCM 19–24 : calculate stress parameters.
QT – q, deviator stress.
PE – p', mean normal effective stress.
EV – voids ratio, calculated from cumulative strains.
EE – voids ratio, calculated from stress state.
PYE – size of current yield locus.
PCO – absolute value of PYE.
EVCM 25–26 : calculate Cam-clay parameters to be output.
EVCM 27–28 : update size of yield locus and voids ratio.
EVCM 30 : add critical state flag to counter.
= 1, if approaching critical state.
= 0, otherwise.
EVCM 31 : if integration point is approaching critical state, enter number.
EVCM 32 : add negative p' flag to counter.
= 1, if integration point has negative p'.
= 0, otherwise.
EVCM 33 : if integration point has negative p' then enter it.
EVCM 34 : calculate angle THETA.
EVCM 35 : enter THETA.
EVCM 38 : if not **last** integration point then return.
EVCM 39 : increment counter of elements with Cam-clay properties.
EVCM 40 : enter 1 to indicate element has Cam-clay properties.
EVCM 41 : enter 1 to indicate element has integration point(s) approaching critical state.
EVCM 42 : enter 1 to indicate element has integration point(s) with negative p'.

Routine VARCAM

```
      SUBROUTINE VARCAM(IP,MR,KM,ICS,INGP,IEL,NIP,NEL,NCODE,VARC,PR,     VRCM   1
     1 NTY,PE,QT,PCO,PYE,U,EV,EE,PC,ED,LED,NPR,NMT)                      VRCM   2
C*************************************************************************VRCM   3
C *** FOR CAM-CLAYS ONLY  ***                                            *VRCM   4
C *** THIS ROUTINE DETERMINES THE CURRENT STRESS STATE AT THE            *VRCM   5
C *** END OF THE CURRENT INCREMENT AND USES IST TO INDICATE THE          *VRCM   6
C *** STRESS STATE OF THE INTEGRATION POINT WITH REFERENCE               *VRCM   7
C *** TO THE CURRENT YIELD LOCUS                                         *VRCM   8
C                                                                        *VRCM   9
C *** TYPE  CODE FOR STRESS STATES                    IST                *VRCM  10
C ***  0    SOIL IS ELASTIC   WITH P>PCS AND Q<M*P    - 0               *VRCM  11
C ***  1    SOIL IS ELASTIC   WITH P<PCS AND Q<M*P    - 1               *VRCM  12
C ***  2    SOIL IS ELASTIC   WITH P<PCS AND Q>M*P    - 2               *VRCM  13
C ***  3    SOIL IS HARDENING WITH P>PCS AND Q<M*P    - 3               *VRCM  14
C ***  4    SOIL IS SOFTENING WITH P<PCS AND Q>M*P    - 4               *VRCM  15
```



```
C ***  7    SOIL IS HARDENING WITH P>PCS AND Q>M*P    - 7               *VRCM  16
C ***  8    SOIL IS HARDENING WITH P<PCS AND Q>M*P    - 8               *VRCM  17
C                                                                        *VRCM  18
C *** WHERE    P - EFFECTIVE MEAN NORMAL STRESS                          *VRCM  19
C ***        PCS - CRITICAL STATE VALUE OF P FOR CURRENT YIELD LOCUS     *VRCM  20
C *** TYPES 7 AND 8 ARE IMPERMISSIBLE AND ARISE FROM NUMERICAL           *VRCM  21
C *** PROBLEMS.                                                          *VRCM  22
C*************************************************************************VRCM  23
      DIMENSION PR(NPR,NMT),NCODE(NIP,NEL),VARC(10,NIP,NEL)              VRCM  24
      DIMENSION ED(LED),NTY(NMT)                                         VRCM  25
      COMMON /DEVICE/ IR1,IR4,IR5,IW2,IW4,IW6,IW7,IW8,IW9                VRCM  26
C----------------------------------------------------------------------VRCM  27
C     FIND NEW YIELD LOCUS                                               VRCM  28
C----------------------------------------------------------------------VRCM  29
      PC=ABS(PYE)                                                        VRCM  30
      IF(NTY(KM).NE.3) GO TO 10                                          VRCM  31
      PCS=PC/2.                                                          VRCM  32
      PY=PE+QT*QT/(PE*PR(4,KM)*PR(4,KM))                                 VRCM  33
      GO TO 12                                                           VRCM  34
   10 PCS=PC/2.7182818                                                   VRCM  35
      PY=PE*EXP(QT/(PR(4,KM)*PE))                                        VRCM  36
   12 CONTINUE                                                           VRCM  37
      IF(PY.GT.PC) GO TO 13                                              VRCM  38
C----------MATERIAL IS EITHER ELASTIC OR HAS SOFTENED.                   VRCM  39
      IF(PE.GT.PCS) GO TO 14                                             VRCM  40
C----------MATERIAL IS IN REGION 1 OR 2 OR 4                             VRCM  41
      IF(QT.GT.0.999*PR(4,KM)*PE) GO TO 15                               VRCM  42
C----------MATERIAL IS IN REGION 2 AND ELASTIC                           VRCM  43
      IST=1                                                              VRCM  44
      GO TO 17                                                           VRCM  45
   15 CONTINUE                                                           VRCM  46
C----------MATERIAL IS IN REGION 2 OR 4                                  VRCM  47
      IF(PYE.LT.0.) GO TO 16                                             VRCM  48
C----------MATERIAL IS ELASTIC AND IN REGION 2                           VRCM  49
  155 IST=2                                                              VRCM  50
      GO TO 17                                                           VRCM  51
   16 CONTINUE                                                           VRCM  52
      IF(ED(2).LT.ED(1)) GOTO 155                                        VRCM  53
C----------MATERIAL HAS SOFTENED IN REGION 4                             VRCM  54
      IST=4                                                              VRCM  55
      PCS=PCS*PY/PC                                                      VRCM  56
      PC=PY                                                              VRCM  57
      GO TO 17                                                           VRCM  58
   14 CONTINUE                                                           VRCM  59
C----------MATERIAL IS IN REGION 0 AND ELASTIC                           VRCM  60
      IST=0                                                              VRCM  61
      GO TO 17                                                           VRCM  62
   13 CONTINUE                                                           VRCM  63
C----------MATERIAL HAS HARDENED                                         VRCM  64
      IF(PE.GT.PCS) GO TO 18                                             VRCM  65
C----------MATERIAL IS IN REGION 8 AND IS INVALID                        VRCM  66
      PCS=PCS*PY/PC                                                      VRCM  67
      PC=PY                                                              VRCM  68
      IST=8                                                              VRCM  69
      GO TO 17                                                           VRCM  70
   18 CONTINUE                                                           VRCM  71
C----------MATERIAL IS IN REGION 3 OR 7                                  VRCM  72
      IF(QT.GT.1.001*PR(4,KM)*PE) GO TO 19                               VRCM  73
C----------MATERIAL IS IN REGION 3                                       VRCM  74
      PCS=PCS*PY/PC                                                      VRCM  75
      PC=PY                                                              VRCM  76
      IST=3                                                              VRCM  77
      GO TO 17                                                           VRCM  78
C----------MATERIAL IS IN REGION 7                                       VRCM  79
   19 IST=7                                                              VRCM  80
      PC=PY                                                              VRCM  81
```

```
   17 WARN=QT/PE                                                        VRCM  82
      IF(WARN.LT.0.95*PR(4,KM)) WARN=0.                                 VRCM  83
      WARN=(WARN-PR(4,KM))/PR(4,KM)                                     VRCM  84
      IF(ABS(WARN).LT.0.05.AND.PYE.LT.0.) ICS=1                         VRCM  85
C-----------------------------------------------------------------------VRCM  86
C          CALCULATE NEW VOIDS RATIO                                    VRCM  87
C-----------------------------------------------------------------------VRCM  88
      IF(PE.GT.0.) GO TO 20                                             VRCM  89
      INGP=1                                                            VRCM  90
      GO TO 21                                                          VRCM  91
   20 EE=PR(3,KM)-PR(1,KM)*ALOG(PE)-(PR(2,KM)-PR(1,KM))*ALOG(PCS)       VRCM  92
   21 CONTINUE                                                          VRCM  93
C                                                                       VRCM  94
      VARC(1,IP,IEL)=PE                                                 VRCM  95
      VARC(2,IP,IEL)=QT                                                 VRCM  96
      VARC(3,IP,IEL)=PE+U                                               VRCM  97
      VARC(4,IP,IEL)=PC                                                 VRCM  98
      VARC(5,IP,IEL)=QT/PE                                              VRCM  99
      VARC(6,IP,IEL)=QT/(PE*PR(4,KM))                                   VRCM 100
      VARC(7,IP,IEL)=PY/PCO                                             VRCM 101
      VARC(8,IP,IEL)=EE                                                 VRCM 102
      VARC(9,IP,IEL)=EV                                                 VRCM 103
      NCODE(IP,IEL)=IST                                                 VRCM 104
      RETURN                                                            VRCM 105
      END                                                               VRCM 106
```

VRCM 30 : absolute value of current yield locus.

VRCM 31–33 : calculate critical state value of p' (PCS) and the yield locus (PY) passing through the stress state (p', q) for modified Cam-clay.

VRCM 35–36 : do the same for Cam-clay.

VRCM 38 : skip if yield locus has expanded.

VRCM 40 : skip if on the **wet** side.

VRCM 42 : new stress state is on the **dry** side; skip if above critical state line (CSL).

VRCM 44 : stress state is below CSL and on the dry side and is therefore elastic; assign value of 1.

VRCM 48 : on the **dry** side and above CSL (either elastic or softening). Skip if previously yielded (PYE $<$ 0).

VRCM 50 : stress state is elastic, on the dry side and above CSL; assign value of 2.

VRCM 53 : skip if deviator strain has not increased; then probably unloading from yielded state to elastic.

VRCM 55–57 : softening; assign value of 4. Also set PC and PCS to new values.

VRCM 61 : yield locus has not expanded and the stress state is on the **wet** side, i.e. still elastic; assign value of 0.

VRCM 65 : yield locus has expanded; skip if on the **wet** side.

VRCM 67–69 : yield locus has expanded, i.e. hardening on the **dry** side. Impermissible stress state; assign value of 8.

VRCM 73 : on **wet** side; yield locus has expanded. Skip if above CSL.

VRCM 75–77 : yield locus has expanded and the stress state is on the wet side, i.e. hardening; assign value 3.

VRCM 80–81 : above CSL on **wet** side; impermissible stress state; assign value of 7.

VRCM 82–85 : calculate η to see if stress state is close to the CSL, if so, assign 1 to ICS.

VRCM 89–90 : if $p' < 0$, assign value of 1 to INGP and skip calculation of voids ratio (EE) from stress state.

VRCM 92 : calculate voids ratio (EE) from current stress state.

VRCM 95–103 : enter current values of parameters to be printed out.

PE	– mean normal effective stress (p').
QT	– deviator stress (q).
PE+U	– total stress (p).
PC	– yield locus size.
QT/PE	– η.
QT/M*PE	– η/M.
PY/PCO	– yield ratio (YR).
EE	– voids ratio from stress state.
EV	– voids ratio from strains.

VRCM 104 : enter stress state code IST in array NCODE.

Wherever applicable, a set of warning messages is printed from Cam-clays. The warning messages are as follows:

(a) integration points are approaching the critical state;
(b) integration points have negative values for p'.

For example, for case (a):

****** WARNING ****** ELEMENT 8 HAS INTEGRATION
POINTS 1 2 0 0 0 6 0 APPROACHING CRITICAL STATE

This indicates that out of the seven integration points, the 1st, 2nd and 6th are approaching the critical state.

Before the final part, (iv), of the output is printed, the following message may be printed.

WARNING **** THE NODAL CO-ORDINATES ARE UPDATED

This warning message is self-explanatory and is printed when IUPD is set equal to 1 (in the input data) in order to update the co-ordinates at the end of each increment.

Routine ANGTH

```
      SUBROUTINE ANGTH(VARINT,NEL,NIP,NVRS,IP,J,THETA)                  ANGT   1
C*********************************************************************ANGT   2
C     ROUTINE TO CALCULATE ANGLE IN PI PLANE                            ANGT   3
C*********************************************************************ANGT   4
      DIMENSION VARINT(NVRS,NIP,NEL)                                    ANGT   5
      COMMON /PARS  / PYI,ALAR,ASMVL,ZERO                               ANGT   6
```

```
C                                                                       ANGT   7
      SX=VARINT(1,IP,J)                                                 ANGT   8
      SY=VARINT(2,IP,J)                                                 ANGT   9
      SZ=VARINT(3,IP,J)                                                 ANGT  10
      TXY=VARINT(4,IP,J)                                                ANGT  11
C                                                                       ANGT  12
      PIBY4=0.25*PYI                                                    ANGT  13
      SD=0.5*(SX-SY)                                                    ANGT  14
      SM=0.5*(SX+SY)                                                    ANGT  15
      RAD=SQRT(SD*SD+TXY*TXY)                                           ANGT  16
      SIG1=SM+RAD                                                       ANGT  17
      SIG3=SM-RAD                                                       ANGT  18
      DY=SY-SM                                                          ANGT  19
      IF(ABS(TXY).LT.ASMVL.AND.ABS(DY).LT.ASMVL)GOTO 8                  ANGT  20
      THXY2=ATAN2(TXY,DY)                                               ANGT  21
      GOTO 9                                                            ANGT  22
    8 THXY2=0.5*PYI                                                     ANGT  23
    9 THXY=0.5*THXY2                                                    ANGT  24
      THXYD=THXY*180./PYI                                               ANGT  25
      IF(ABS(THXY).LT.PIBY4)GOTO 10                                     ANGT  26
      PSIGX=SIG1                                                        ANGT  27
      PSIGY=SIG3                                                        ANGT  28
      GOTO 15                                                           ANGT  29
   10 PSIGX=SIG3                                                        ANGT  30
      PSIGY=SIG1                                                        ANGT  31
   15 PSIGZ=SZ                                                          ANGT  32
C                                                                       ANGT  33
      SIGX=(PSIGZ-PSIGY)/SQRT(2.)                                       ANGT  34
      SIGY=(2.*PSIGX-PSIGY-PSIGZ)/SQRT(6.)                              ANGT  35
CC    RADO=SQRT(SIGX*SIGX+SIGY*SIGY)                                    ANGT  36
      IF(ABS(SIGX).LT.ASMVL.AND.ABS(SIGY).LT.ASMVL)GOTO 20              ANGT  37
C                                                                       ANGT  38
      THETA=ATAN2(SIGY,SIGX)                                            ANGT  39
      IF(THETA.LT.ZERO)THETA=2.*PYI+THETA                               ANGT  40
      THETA=THETA*180./PYI                                              ANGT  41
      GOTO 25                                                           ANGT  42
C                                                                       ANGT  43
   20 THETA=ALAR                                                        ANGT  44
   25 CONTINUE                                                          ANGT  45
C                                                                       ANGT  46
      RETURN                                                            ANGT  47
      END                                                               ANGT  48
```

ANGT 8–11 : effective stresses.
ANGT 14–19 : calculate principal stresses in x–y plane.
ANGT 20 : check for Mohr's circle being a point; if so, skip.
ANGT 21 : calculate angle between major principal stress direction and x axis (THXY2 = $2\theta_{xy}$).
ANGT 23–24 : calculate θ_{xy}.
ANGT 27–28 : major principal stress direction is closer to x axis.
ANGT30–31 : major principal stress direction is closer to y axis.
ANGT 32 : calculate intermediate stresses.
ANGT 34–35 : calculate components of stress in x' and y' directions.
ANGT 37 : check for stress state being co-incident with origin (i.e. stress state represents hydrostatic stress conditions).
ANGT 39 : calculate angle between x' axis and the stress state.
ANGT 40 : if negative, add 2π to bring it into the range 0 to 2 π.
ANGT 41 : θ in degrees.
ANGT 44 : if hydrostatic stress state then set θ to a large value.

Routine PRINC

```
      SUBROUTINE PRINC(C,D,E,B)                                         PRNC   1
C*************************************************************************PRNC   2
C     CALCULATES  PRINCIPAL  STRESSES  AND  THEIR  DIRECTIONS           PRNC   3
C*************************************************************************PRNC   4
      DIMENSION B(3)                                                    PRNC   5
      COMMON /PARS  / PYI,ALAR,ASMVL,ZERO                               PRNC   6
C                                                                       PRNC   7
      AP=C+D                                                            PRNC   8
      AD=C-D                                                            PRNC   9
      S=SQRT(.25*AD*AD+E*E)                                             PRNC  10
      B(1)=.5*AP+S                                                      PRNC  11
      B(2)=.5*AP-S                                                      PRNC   1
      B(3)=90.                                                          PRNC   1
      IF(ABS(AD).LT.ASMVL) GO TO 2                                      PRNC  14
      B(3)=28.6479*ATAN(2.*E/AD)                                        PRNC  15
    2 RETURN                                                            PRNC  16
      END                                                               PRNC  17
```

PRNC 10 : radius of Mohr's circle in x–y plane (r–z for axisymmetry).
PRNC 11–12 : calculate major and minor principal stresses.
PRNC 13 : set angle between the x axis and major principal stress direction to 90° in anticipation of σ_x being equal to σ_y.
PRNC 14 : skip if $\sigma_x = \sigma_y$ (the angle is 90°).
PRNC 15 : the angle between x axis and major principal stress direction (in degrees).

Routine CAMCDE

```
      SUBROUTINE CAMCDE(IW6)                                            CAMC   1
C*************************************************************************CAMC   2
C     OUTPUT CODE TO IDENTIFY STRESS STATE FOR CAM CLAYS                CAMC   3
C*************************************************************************CAMC   4
C                                                                       CAMC   5
      WRITE(IW6,901)                                                    CAMC   6
      WRITE(IW6,902)                                                    CAMC   7
      WRITE(IW6,903)                                                    CAMC
      WRITE(IW6,904)                                                    CAMC
  901 FORMAT(1X//120(1H=)/                                              CAMC  10
     1 /30X,40HCODE FOR STRESS STATE FOR CAM-CLAYS ONLY/30X,            CAMC  11
     1 40(1H-)//30X,34HCODE GIVES THE STRESS STATE OF THE/              CAMC  12
     1 30X,39HINTEGRATION POINT WITH REFERENCE TO THE/                  CAMC  13
     1 30X,20HCURRENT YIELD LOCUS.                                      CAMC  14
     1 //30X,47H        STRESS   STATE                    CODE)         CAMC  15
  902 FORMAT(30X,46HSOIL IS ELASTIC   WITH P>PCS AND Q<M*P    -  0/     CAMC  16
     1 30X,46HSOIL IS ELASTIC   WITH P<PCS AND Q<M*P    -  1/          CAMC  17
     1 30X,46HSOIL IS ELASTIC   WITH P<PCS AND Q>M*P    -  2/          CAMC  18
     1 30X,46HSOIL IS HARDENING WITH P>PCS AND Q<M*P    -  3)          CAMC  19
  903 FORMAT(30X,46HSOIL IS SOFTENING WITH P<PCS AND Q>M*P    -  4/     CAMC  20
     1 30X,46HSOIL IS HARDENING WITH P>PCS AND Q>M*P    -  7)          CAMC  21
  904 FORMAT(30X,46HSOIL IS HARDENING WITH P<PCS AND Q>M*P    -  8/     CAMC  22
     1 30X,46HSOIL HAS NEGATIVE P                       -  9/          CAMC  23
     1 /30X,40HWHERE   P - EFFECTIVE MEAN NORMAL STRESS                 CAMC  24
     1 /30X,37H      PCS - CRITICAL STATE VALUE OF P                    CAMC  25
     1 /30X,46HTYPES 7 AND 8 ARE IMPERMISSIBLE AND ARISE FROM/          CAMC  26
     1 30X,18HNUMERICAL PROBLEMS/)                                      CAMC  27
      RETURN                                                            CAMC  28
      END                                                               CAMC  29
```

CAMC 6–9 : write explanation of stress state code for Cam-clay models only.

Routine UPOUT2

```
      SUBROUTINE UPOUT2(IW6,NEL,NIP,LTYP,MAT,NCAM,IOUT3,IEL,          UPOT   1
     1 NCODE,LCS,LNGP,NELPR,NELUS,NELCM,MCS,MNGP,VARC)                UPOT   2
C*********************************************************************UPOT   3
C *** OUTPUT ADDITIONAL PARAMETERS CAM-CLAYS                          UPOT   4
C*********************************************************************UPOT   5
      DIMENSION MAT(NEL),LTYP(NEL),VARC(10,NIP,NEL)                    UPOT   6
      DIMENSION NCODE(NIP,NEL),LNGP(NIP,NEL),LCS(NIP,NEL)              UPOT   7
      DIMENSION NELPR(NEL),NELUS(NEL),NELCM(NEL),MCS(NEL),MNGP(NEL)    UPOT   8
      COMMON /ELINF / LINFO(50,15)                                     UPOT   9
      COMMON /OUT   / IBC,IRAC,NVOS,NVOF,NMOS,NMOF,NELOS,NELOF,ISR     UPOT  10
C                                                                      UPOT  11
      IF(NCAM.EQ.0)GOTO 100                                            UPOT  12
      IF(IOUT3.EQ.0)GOTO 25                                            UPOT  13
      IF(IOUT3.EQ.1)WRITE(IW6,911)                                     UPOT  14
      IF(IOUT3.EQ.1)WRITE(IW6,902)                                     UPOT  15
      IF(IOUT3.EQ.2)WRITE(IW6,912)                                     UPOT  16
      IF(IOUT3.EQ.2)WRITE(IW6,901)                                     UPOT  17
C                                                                      UPOT  18
      DO 20 ILM=1,IEL                                                  UPOT  19
      J=NELPR(ILM)                                                     UPOT  20
      IC=NELCM(ILM)                                                    UPOT  21
      IF(IC.NE.1)GOTO 20                                               UPOT  22
      MR=NELUS(ILM)                                                    UPOT  23
      KM=MAT(J)                                                        UPOT  24
      LT=LTYP(J)                                                       UPOT  25
      NGP=LINFO(11,LT)                                                 UPOT  26
      IF(MR.LT.NELOS.OR.MR.GT.NELOF)GOTO 16                            UPOT  27
      IF(IOUT3.EQ.1)GOTO 12                                            UPOT  28
      IF(IOUT3.EQ.2)WRITE(IW6,904)MR                                   UPOT  29
C                                                                      UPOT  30
      DO 10 IGP=1,NGP                                                  UPOT  31
      WRITE(IW6,905)IGP,(VARC(IK,IGP,ILM),IK=1,10),NCODE(IGP,ILM)      UPOT  32
   10 CONTINUE                                                         UPOT  33
      GOTO 16                                                          UPOT  34
   12 WRITE(IW6,905)MR,(VARC(IK,NGP,ILM),IK=1,10),                     UPOT  35
     1 (NCODE(IP,ILM),IP=1,NGP)                                        UPOT  36
C-----------------------------------------------------------------------UPOT  37
C     WARNING MESSAGES - CAM-CLAYS                                     UPOT  38
C-----------------------------------------------------------------------UPOT  39
   16 IF(MCS(ILM).EQ.0)GOTO 17                                         UPOT  40
      WRITE(IW6,916)MR,(LCS(IP,ILM),IP=1,NGP)                          UPOT  41
   17 IF(MNGP(ILM).EQ.0)GOTO 20                                        UPOT  42
      WRITE(IW6,917)MR,(LNGP(IP,ILM),IP=1,NGP)                         UPOT  43
   20 CONTINUE                                                         UPOT  44
      GOTO 100                                                         UPOT  45
C-----------------------------------------------------------------------UPOT  46
C     WARNING MESSAGES (ONLY) FOR CAM-CLAYS                            UPOT  47
C-----------------------------------------------------------------------UPOT  48
   25 WRITE(IW6,935)                                                   UPOT  49
      DO 40 ILM=1,IEL                                                  UPOT  50
      J=NELPR(ILM)                                                     UPOT  51
      IC=NELCM(ILM)                                                    UPOT  52
      IF(IC.NE.1)GOTO 40                                               UPOT  53
      MR=NELUS(ILM)                                                    UPOT  54
      LT=LTYP(J)                                                       UPOT  55
      NGP=LINFO(11,LT)                                                 UPOT  56
      IF(MCS(ILM).EQ.0)GOTO 37                                         UPOT  57
      WRITE(IW6,916)MR,(LCS(IP,ILM),IP=1,NGP)                          UPOT  58
   37 IF(MNGP(ILM).EQ.0)GOTO 40                                        UPOT  59
      WRITE(IW6,917)MR,(LNGP(IP,ILM),IP=1,NGP)                         UPOT  60
   40 CONTINUE                                                         UPOT  61
```



```
  100 CONTINUE                                                         UPOT  62
      RETURN                                                           UPOT  63
  901 FORMAT(2X,6HELM-IP,6X,2HPE,11X,1HQ,11X,2HPT,11X,                UPOT  64
     1 2HPC,9X,3HETA,5X,5HETA/M,6X,2HYR,4X,6HE-STRS,3X,                UPOT  65
     2 6HE-STRN,3X,4HTH-3,2X,3HCDE)                                    UPOT  66
  902 FORMAT(2X,6HELM-IP,6X,2HPE,11X,1HQ,11X,2HPT,11X,                UPOT  67
     1 2HPC,9X,3HETA,5X,5HETA/M,6X,2HYR,4X,6HE-STRS,3X,                UPOT  68
     2 6HE-STRN,3X,4HTH-3,2X,14H 1 2 3 4 5 6 7)                        UPOT  69
  904 FORMAT(I4)                                                       UPOT  70
  905 FORMAT(2X,I4,4E13.5,2F9.3,3X,F6.3,2F9.4,F7.1,2X,8I2/5X,9I2)      UPOT  71
CC906 FORMAT(/18H CENTROID STRESSES/1X,17(1H-)/)                       UPOT  72
  911 FORMAT(/33H CAM CLAY PARAMETERS AT CENTROIDS/                    UPOT  73
     1 1X,32(1H-)/)                                                    UPOT  74
  912 FORMAT(/42H CAM CLAY PARAMETERS AT INTEGRATION POINTS/           UPOT  75
     1 1X,41(1H-)/)                                                    UPOT  76
  916 FORMAT(29H ******WARNING****** ELEMENT ,I3,                      UPOT  77
     1 24H HAS INTEGRATION POINTS ,7I2,27H APPROACHING CRITICAL STATE, UPOT  78
     2 2X,9I3)                                                         UPOT  79
  917 FORMAT(29H ******WARNING****** ELEMENT ,I3,                      UPOT  80
     1 24H HAS INTEGRATION POINTS ,7I2,18H PE LESS THAN ZERO,2X,9I3)   UPOT  81
CC925 FORMAT(2X,I4,2E14.5,F10.3,2X,16I3)                               UPOT  82
  935 FORMAT(//)                                                       UPOT  83
      END                                                              UPOT  84
```

UPOT 12 : skip if no elements with Cam-clay properties.
UPOT 13 : skip if only warning messages are to be printed.
UPOT 14–17 : write title for output tables.
UPOT 19 : loop on elements that were processed in routine UPOUT.
UPOT 20 : J is program element number.
UPOT 21 : IC is flag to indicate (if set to 1) element with Cam-clay properties.
UPOT 22 : skip if element does not have Cam-clay properties.
UPOT 23 : MR is the user element number.
UPOT 24–26 : element type dependent parameters.
- KM – material zone number.
- LT – element type number.
- NGP – number of integration points.

UPOT 31–33 : write output parameters for all integration points.
UPOT 35–36 : write output parameters for centroid (last integration point).
UPOT 41 : element with integration point(s) approaching critical state; print message.
UPOT 43 : element with integration point(s) with negative p'; print message.
UPOT 44 : end of loop on elements.
UPOT 50 : loop on all elements processed in routine UPOUT.
UPOT 53 : by-pass if element does not have Cam-clay properties.
UPOT 54 : MR is the user element number.
UPOT 58 : element with integration point(s) approaching critical state; print message.
UPOT 60 : element with integration point(s) with negative p'; print message.
UPOT 61 : end of loop on elements.

A list of reactions-to-earth at the d.o.f. where the displacements are prescribed is now printed.

(iv) The final part of the output consists of the incremental applied load, the out-of-balance load, the loads equivalent to element stresses and the total applied load at the nodes.

$$\text{out-of-balance loads} = \text{total applied loads} - \text{loads equivalent to element stresses.} \tag{8.14}$$

The nodal loads equivalent to current element stresses are calculated by routine **STRSEQ**.

Routine STRSEQ

```
      SUBROUTINE STRSEQ(JJ,IP,IPA,NVRS,NIP,NEL,NDN,NDIM,NS,              STRS   1
     1 VARINT,SHFN,CARTD,F,DJACB,R,RI,CR)                                STRS   2
C*********************************************************************STRS   3
C     ROUTINE TO CALCULATE FORCES EQUILIBRATING                         STRS   4
C     ELEMENTAL STRESSES (INTEGRATION POINT CONTRIBUTION)               STRS   5
C*********************************************************************STRS   6
      DIMENSION VARINT(NVRS,NIP,NEL),SHFN(NDN),CARTD(NDIM,NDN)          STRS   7
      DIMENSION F(NDIM,NDN)                                             STRS   8
      COMMON /DATW  / W(100)                                            STRS   9
      COMMON /FLOW  / NPLAX                                             STRS  10
C                                                                       STRS  11
      F9=CR*DJACB*W(IPA)                                                STRS  12
      IF(NPLAX.EQ.1)F9=F9*R                                             STRS  13
C                                                                       STRS  14
      U=VARINT(NS+1,IP,JJ)                                              STRS  15
      SIGXT=VARINT(1,IP,JJ)+U                                           STRS  16
      SIGYT=VARINT(2,IP,JJ)+U                                           STRS  17
      SIGZT=VARINT(3,IP,JJ)+U                                           STRS  18
      TXY=VARINT(4,IP,JJ)                                               STRS  19
      IF(NDIM.EQ.2)GOTO 35                                              STRS  20
C                                                                       STRS  21
      TYZ=VARINT(5,IP,JJ)                                               STRS  22
      TZX=VARINT(6,IP,JJ)                                               STRS  23
C                                                                       STRS  24
      DO 30 IN=1,NDN                                                    STRS  25
      F(1,IN)=F(1,IN)+(CARTD(1,IN)*SIGXT+CARTD(2,IN)*TXY                STRS  26
     1               +CARTD(3,IN)*TZX)*F9                               STRS  27
      F(2,IN)=F(2,IN)+(CARTD(2,IN)*SIGYT+CARTD(1,IN)*TXY                STRS  28
     1               +CARTD(3,IN)*TYZ)*F9                               STRS  29
      F(3,IN)=F(3,IN)+(CARTD(3,IN)*SIGZT+CARTD(2,IN)*TYZ                STRS  30
     1               +CARTD(1,IN)*TZX)*F9                               STRS  31
   30 CONTINUE                                                          STRS  32
      GOTO 60                                                           STRS  33
C                                                                       STRS  34
   35 DO 40 IN=1,NDN                                                    STRS  35
      F(1,IN)=F(1,IN)+(CARTD(1,IN)*SIGXT+SHFN(IN)*SIGZT*RI              STRS  36
     1               +CARTD(2,IN)*TXY)*F9                               STRS  37
   40 F(2,IN)=F(2,IN)+(CARTD(2,IN)*SIGYT+CARTD(1,IN)*TXY)*F9            STRS  38
   60 RETURN                                                            STRS  39
      END                                                               STRS  40
```

STRS 12–13 : calculate weighting factors.
STRS 15 : pore pressure.
STRS 16–19 : total stresses for 2-D problems.

STRS 22–23 : additional shear stresses for 3-D analysis.
STRS 25–32 : calculate nodal loads equivalent to stresses in element in 3-D analysis.

$$F = \int_V B^T \sigma \, d(\text{vol}).$$

STRS 35–38 : do the same for 2-D analysis.

8.14 STOP–RESTART FACILITY

There are two options in stopping and starting an analysis, as explained in section 4.2.5. Option 2 is for use with magnetic tape as results from every increment are saved. These data can then be used by post-processor programs to produce plots. The geometry data are written first to provide details of the mesh for post-processing.

The other option is provided in the absence of a magnetic tape facility and is solely for stopping and restarting an analysis. This permits a large analysis to be broken down into a number of manageable runs. For a restarted analysis, subroutine **RESTRT** reads the results at the end of the previous run. The results at the end of the current run are written to a separate file in routine UPOUT.

Routine RESTRT

```
      SUBROUTINE RESTRT(INCS,INCF,NN,NVTX,ND,NEL,NDF,NTPE,NIP,          REST   1
     1 NVRS,NVRN,MUMAX,NNZ,NNOD1,NDIM,MDZ,NEDZ,NL,INXL,                 REST   2
     2 NCONN,LTYP,MRELVV,MREL,NRELVV,NREL,NW,NMOD,                      REST   3
     3 XYZ,DA,VARINT,PCOR,XYFT,STR,PCONI,TTIME,TGRAV)                   REST   4
C*********************************************************************REST   5
C     STOP/START FACILITY                                               REST   6
C*********************************************************************REST   7
      INTEGER TF                                                        REST   8
      DIMENSION NCONN(NTPE,NEL),LTYP(NEL),MRELVV(NEL),MREL(MUMAX),      REST   9
     1 NRELVV(NN),NREL(NNZ),NW(NNOD1),NMOD(NIP,NEL)                     REST  10
      DIMENSION XYZ(NDIM,NN),DA(NDF),VARINT(NVRS,NIP,NEL),PCOR(NDF),    REST  11
     1 XYFT(NDF),STR(NVRN,NIP,NEL),PCONI(NDF)                           REST
      COMMON /DEVICE/ IR1,IR4,IR5,IW2,IW4,IW6,IW7,IW8,IW9               REST
      COMMON /FIX   / DXYT(4,200),MF(200),TF(4,200),NF                  REST  14
      COMMON /PRSLD / PRESLD(10,100),LEDG(100),NDE1(100),NDE2(100),NLED REST  15
      COMMON /OUT   / IBC,IRAC,NVOS,NVOF,NMOS,NMOF,NELOS,NELOF,ISR      REST  16
C                                                                       REST  17
      IF(ISR.EQ.0)RETURN                                                REST  18
C                                                                       REST  19
      IF(ISR.EQ.2)GOTO 20                                               REST  20
C                                                                       REST  21
      IF(ISR.EQ.1)GOTO 10                                               REST  22
      WRITE(IW6,910)ISR                                                 REST  23
  910 FORMAT(/24H ***ERROR : INADMISSIBLE,1X,                           REST  24
     1  21HSTOP/RESTART OPTION =,I5)                                    REST  25
      STOP                                                              REST  26
C                                                                       REST  27
   10 CONTINUE                                                          REST  28
      IF(INCS.EQ.1)RETURN                                               REST  29
C----------DISK FILE OPTION (ONLY ONE INCREMENT IS READ/WRITTEN)         REST  30
      READ(IR1)TTIME,TGRAV,XYZ,VARINT,STR,DA,XYFT,PCOR,PCONI,LTYP,NMOD  REST  31
      READ(IR1)NF,MF,TF,DXYT                                            REST  32
```

```
      READ(IR1)NLED,LEDG,NDE1,NDE2,PRESLD                              REST  33
      RETURN                                                           REST  34
C                                                                      REST  35
C-----------------------------------------------------------------------REST  36
C     STOP/RESTART OPTION SUITABLE WITH TWO MAGNETIC TAPES.            REST  37
C     THE RESULTS FROM ALL PREVIOUS INCREMENTS ARE READ.               REST  38
C-----------------------------------------------------------------------REST  39
   20 CONTINUE                                                         REST  40
      REWIND IW2                                                       REST  41
      IF(INCS.NE.1)GO TO 22                                            REST  42
C-----------------------------------------------------------------------REST  43
C     WRITE GEOMETRY DATA ON UNIT IW2 FOR A NEW ANALYSIS               REST  44
C-----------------------------------------------------------------------REST  45
      WRITE(IW2)NN,NVTX,ND,NEL,NDF,NTPE,NIP,NVRS,NVRN,MUMAX,NNZ,MDZ,   REST  46
     1 NEDZ,NL,INXL                                                    REST  47
      WRITE(IW2)NCONN,NREL,MREL,NRELVV,MRELVV,NW                       REST  48
      WRITE(IW2)XYZ                                                    REST  49
      RETURN                                                           REST  50
C-----------------------------------------------------------------------REST  51
C     READ GEOMETRY DATA FROM UNIT IR1 AND WRITE TO UNIT IW2           REST  52
C     FOR A RE-STARTED ANALYSIS                                        REST  53
C-----------------------------------------------------------------------REST  54
   22 INCS1=INCS-1                                                     REST  55
      REWIND IR1                                                       REST  56
      READ(IR1)NNT,NVTXT,NDT,NELT,NDFT,NTPET,NIPT,NVRST,NVRNT,         REST  57
     1 MUMAXT,NNZT,MDZT,NEDZT,NLT,INXLT                                REST  58
      READ(IR1)NCONN,NREL,MREL,NRELVV,MRELVV,NW                        REST  59
      READ(IR1)XYZ                                                     REST  60
      WRITE(IW2)NNT,NVTXT,NDT,NELT,NDFT,NTPET,NIPT,NVRST,NVRNT,        REST  61
     1 MUMAXT,NNZT,MDZT,NEDZT,NLT,INXLT                                REST  62
      WRITE(IW2)NCONN,NREL,MREL,NRELVV,MRELVV,NW                       REST  63
      WRITE(IW2)XYZ                                                    REST  64
C-----------------------------------------------------------------------REST  65
C     READ STORED RESULTS OF PREVIOUS INCREMENTS FROM UNIT IR1         REST  66
C-----------------------------------------------------------------------REST  67
      DO 24 I=1,INCS1                                                  REST  68
      READ(IR1)TTIME,TGRAV,XYZ,VARINT,STR,DA,XYFT,PCOR,PCONI,LTYP,NMOD REST  69
      READ(IR1)NF,MF,TF,DXYT                                           REST  70
      READ(IR1)NLED,LEDG,NDE1,NDE2,PRESLD                              REST  71
C-----------------------------------------------------------------------REST  72
C     AND STORE RESULTS ON UNIT IW2 FOR SUBSEQUENT RUN                 REST  73
C-----------------------------------------------------------------------REST  74
      WRITE(IW2)TTIME,TGRAV,XYZ,VARINT,STR,DA,XYFT,PCOR,PCONI,LTYP,NMOD REST  75
      WRITE(IW2)NF,MF,TF,DXYT                                          REST  76
      WRITE(IW2)NLED,LEDG,NDE1,NDE2,PRESLD                             REST  77
   24 CONTINUE                                                         REST  78
C                                                                      REST  79
      RETURN                                                           REST  80
      END                                                              REST  81
```

Stop–restart option = 1

REST 31–33 : if a restarted analysis, read results of last increment of previous analysis from disk file.

Stop-restart option = 2

REST 42 : if a restarted run, then skip.
REST 46–49 : write geometric data to unit 2 for a fresh analysis.
REST 57–64 : for a restarted run, copy geometric data from unit 1 to unit 2.
REST 68 : loop on all previous increments for a restarted run.
REST 69–77 : copy results from all increments from unit 1 to unit 2.
REST 78 : end of loop on all previous increments.

9

Examples

9.1 INTRODUCTION

This chapter deals with the use of CRISP. Section 9.2 contains various hints on using the program, which should be read in conjunction with the input specifications (Appendix A). Sections 9.3 to 9.9 describe some example problems, including full details of the input data. The examples are chosen mainly to illustrate the different features of the program. It is not practical, for space reasons, to present completely realistic analysis here, and therefore simple situations have been considered and most of the analyses contain one increment. Therefore care is needed in interpreting the way the examples are presented.

9.2 USER'S GUIDE TO INPUT

9.2.1 Introduction

The input data with which the user must supply the program can be divided into the following categories:

(i) information describing the finite element mesh, i.e. the co-ordinates of nodal points associated with each finite element;
(ii) material properties (and perhaps *in situ* stresses) associated with each finite element;
(iii) boundary conditions for the analysis (i.e. imposed displacements and loads).

Experience shows that mistakes in the specification of the finite element mesh are often made by program users. These mistakes sometimes result in a mesh which is valid so far as the program is concerned but is simply not the mesh which the user intended. For this reason we have made it possible to produce a plot of the finite element mesh together with element and node numbers. This allows the program user to detect any errors in the geometric input data before embarking on a full analysis. (There is an option to run only the geometric part of the program.)

9.2.2 General hints

We believe that the following suggestions should assist.

(i) First solve a problem to which you know the exact answer. It will use many of the program options that will be needed for the real analysis, but the problem will be simpler. For example, users of our program may find it useful to solve a problem of one-dimensional consolidation, or to analyse a triaxial test using one of the critical state models or to do a two-dimensional elastic stress analysis and compare the results with a standard theory of elasticity solution. The main point of doing an exercise like this is to check that you understand how to operate the program correctly. It will also help in giving some idea about the magnitude of suitable time steps, load increments and the accuracy obtained from different meshes.

(ii) When analysing the real problem, ensure that there is an independent check of the results. Of course it is impossible to do this precisely (you would not be using a program if that were the case), but simple order of magnitude checks using conventional methods can identify gross mistakes. Repeating the analysis using another program is a good check, but often this will not be possible because of the cost or non-availability of another program.

(iii) Study the results of the computer analysis. If doing an analysis with a critical state model then plot some effective stress paths. Do the results seem to exhibit any strange behaviour, e.g. are there large discrepancies between the stresses in neighbouring elements or between successive increments?

9.2.3 Size of increments

Great care is needed in selecting increment sizes. The **whole** loading is divided into a number of small load steps, i.e. increments. How many load increments should one use? What is the right size of load increment? The answers to these questions depend on the problem being solved. However, there are a few guidelines. Use as many increments as possible. More increments will be needed for a drained analysis compared to an undrained analysis.

When the overall behaviour is elastic, larger load increments may be used. On the wet side of critical state, the soil undergoes hardening and the yield surface

expands. If one could limit the load increment size such that the expansion of the yield locus is within 5% at any integration point, the results can be expected to be reasonable. Within 2% would improve the result. Similarly on the dry side the yield locus shrinks in size as plastic yielding takes place. A much tighter control is recommended for softening. A parameter called the 'yield ratio' is printed for each integration point. This parameter is defined as the ratio of the yield locus size at the end of the current increment to the size at the beginning of the current increment. The size of the yield locus is defined by the p'_c value, the value of p' on the $q = 0$ axis.

9.2.4 User's guide to input

Record A

The title is usually set by the user to be descriptive of the subject of the finite element analysis. The title appears on the program's plot of the finite element mesh as well as near the start of the printed program output. If different meshes are used to tackle the same problem then the titles should be different, e.g.

FOOTING ANALYSIS – MESH 1 – 60 LST ELEMENTS

and

FOOTING ANALYSIS – MESH 2 – 100 LST ELEMENTS

Record B

Element types (MXTYP)

Although it is possible to include more than one type of finite element in a mesh, normally all elements will be of the same type. The element type is defined by MXTYP, which at present can take one of the four values associated with the elements shown in Fig. 4.1.

The variations of displacements (and consequently strains) and, where appropriate, pore pressures are summarised in the following table.

MXTYP	Element name	Displacement	Strain	Excess pore pressure
2	Linear strain triangle (LST)	Quadratic	Linear	N/A
3	LST with linearly varying excess pore pressure	Quadratic	Linear	Linear
6	Cubic strain triangle (CuST)	Quartic	Cubic	N/A
7	CuST with cubic variation of excess pore pressure	Quartic	Cubic	Cubic

All the elements are basically standard displacement finite elements, which are described in most texts on the finite element method (e.g. Zienkiewicz, 1977).

Note that NVTX refers to the number of vertex (i.e. corner) nodes in the finite element mesh. The program automatically generates node numbers and co-ordinates for any nodes lying on element sides or within elements.

Although CRISP allows the user complete freedom in the choice of element type, the following recommendations should lead to the selection of an appropriate element type:

(i) plane strain analysis: for drained or undrained analysis, use element type 2 (linear strain triangle) and for consolidation analysis use element type 3.

(ii) axisymmetric analysis: for drained analysis or consolidation analysis where collapse is not expected then element types 2 and 3 will probably be adequate (i.e. the same as (i) above). For undrained analysis or a situation where collapse is expected then element types 6 and 7 are recommended. Recent research has shown that in axisymmetric analyses the constraint of no volume change (which occurs in undrained situations) leads to finite element meshes 'locking up' if elements such as the LST are used (Sloan and Randolph, 1982).

How many elements?

It is difficult to lay down rules for the number of finite elements needed in a mesh to analyse a particular problem. The following hints may assist inexperienced analysts:

(i) avoid the pitfall of using too few elements – remember that in the case of the linear strain triangle, for example, stresses will vary linearly across the element;
(ii) avoid the pitfall of using too many elements – in most situations between 50 and 100 LSTs will be adequate, as will between 20 and 30 CuSTs.
(iii) the mesh should be finer (i.e. elements should be smaller) in regions where rapidly varying strains/stresses are to be expected (e.g. near loaded boundaries).

Mixing different element types

As mentioned above, the possibility exists of mixing different element types in a CRISP analysis. The only element types for which mixing is recommended in the current program version are element type 2 with element type 3, and element type 6 with element type 7. This could be done in a consolidation analysis where part of the continuum is expected to behave in a completely drained or completely undrained mode in comparison to the rest.

Record C

The parameters NUMAX and MUMAX need to be specified only if there are gaps in the vertex node numbering and element numbering respectively. This information is necessary to allocate sizes to arrays which store the node numbers. Rather than arbitrarily allocating sizes to these arrays, which imposes a limit on the number of vertex nodes and elements, this procedure is preferred.

Record D

The normal option is to set all these values to zero. These flags need to be set only when the program is being tested (left in for the benefit of users/programmers who may want to change the program, e.g. to incorporate a new element type). This feature helps to ensure that the changes made to the program are correct.

This debugging option may also be used to track down any errors in the specification of the finite element mesh (but this is best dealt with by a data-checking program).

Record E

The program calculates the co-ordinates of nodes along sides and element interiors by linear interpolation, assuming that the elements are straight-edged. However, in some analyses (e.g. circular tunnel, buried pipe) it is more appropriate for the element sides to be curved to accurately model the physical problem. The program does not have the facility to calculate co-ordinates of nodes assuming that the element sides are curved. This feature is included so that (see records I and J) the user can directly specify the co-ordinates along the (few) curved sides.

If all the element sides in the mesh are curved, the user may envisage writing a small program to generate the nodal co-ordinates automatically and input as described below (records I and J).

Records I and J are then used to specify the co-ordinates of nodes, for each element side. It should be noted that displacement and pore pressure nodes are dealt with separately. For element types 2, 6 (non-consolidation elements) and also element type 3 (does not have pore pressure nodes along side), NSPZ and NPCUR must be set to zero. Records J are then omitted from input.

If element type 7 is used with curved sides then the user must ensure that the co-ordinates of both displacement and pore pressure nodes which are specified separately lie along the curved side.

Records F and H

Element and nodal numbering

The program user must assign each element and each vertex node in the finite element mesh unique (integer) numbers in the following ranges:

$1 \leqslant$ node number $\leqslant 750$.

$1 \leqslant$ element number $\leqslant$ MUMAX (user specified; if equal to zero, then NEL).

It is not necessary for either the node numbers or the element numbers to form a complete set of consecutive integers, i.e. there may be 'gaps' in the numbering scheme adopted. This facility means that users may modify existing finite element meshes by removing elements without the need for renumbering the whole mesh. The geometry part of the program assigns numbers in the range 751 upwards to nodes on element sides and in element interiors.

Co-ordinate system

It is recommended that the user adopts a co-ordinate system with the y axis pointing upwards (Fig. 9.1).

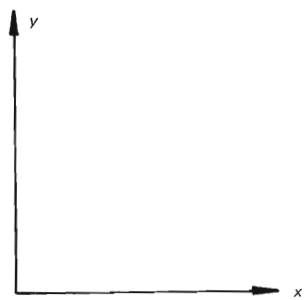


Fig. 9.1 – Co-ordinate system

Note the x axis points to the right – if the x axis points to the left then the program will calculate element areas and stiffnesses as negative quantities. (This recommendation is linked to the program's expectation that element node numbers are listed in record H in an anti-clockwise sense. In principle it is possible to use a co-ordinate system with the x axis pointing to the left, but then it would be necessary to list element node numbers in a clockwise sense, and a different sign convention for shear stresses would be needed in records P1, P3 and U.

The user may rotate the co-ordinate system if desired (i.e. so that the y axis no longer points vertically upwards), but should be noted that the following input options for the program will not work in the normal fashion:

(i) specification of material self-weight loads (excavation, construction and gravity increase – records M and R);
(ii) elastic properties varying linearly with depth (record M);
(iii) axisymmetric analysis.

When the axisymmetric analysis option is selected (record L1) it is assumed that the y axis is the axis of symmetry (i.e. the x axis is in the radial direction).

Units

The user can choose any appropriate length for describing the co-ordinates of nodal points. It is important, however, that the units chosen to describe material properties, stresses and loads in the program are consistent. In a drained or undrained analysis the user can only select the units for two quantities independently – the units for describing all other items are then automatically determined. Since the unit of length is always determined by the co-ordinate data, the user has one choice remaining and this can most simply be regarded as relating the units of force that are to be used. For example, if length and force units are chosen to be metres (m) and kilonewtons (kN) respectively then stresses and elastic moduli must be in kN/m^2 and unit weights must be in kN/m^3 (see Table 9.1).

Table 9.1 Consistent set of units

	1	2	3	4
Length	m	mm	mm	ft
Force	kN	N	mN†	lbf
Time	sec	sec	sec	hr
Pressure, stress	kN/m^2	N/mm^2	mN/mm^2 ‡	lbf/ft^2
Density	kN/m^3	N/mm^3	mN/mm^3	lbf/ft^3
Permeability	m/s	mm/s	mm/s	ft/hr

† millinewtons.
‡ mN/mm^2 and kPa (i.e. kN/m^2) are equal in magnitude.

When a consolidation analysis is performed, suitable units of time must also be chosen, and the units chosen for permeability imply certain units for increment time steps (e.g. if permeability has units of metres/year then time steps will be in units of years).

Material zone numbers (IMAT)

The user must assign a **zone** number (in the range 1 to 10) to each finite element. The zone number associates each element with a particular set of material properties (record M of program input). Thus if there are three zones of

soil with different material properties, zones 1 and 2 may be modelled by Cam-clay with distinct material parameters and zone 3 may be modelled by linear elastic properties. (Note: 'gaps' in the numbers of zones are not allowed.)

Records G1 and G2

The frontal method of solving equations requires an efficient element numbering. The numbering adopted by the user may not necessarily be the most efficient. Inefficient element numbering in any analysis of medium to large sized problems may prove to be prohibitively expensive. No attempt is made in the program to renumber the elements for efficient use of the frontal method. With element renumbering programs (for the frontal method) becoming available, the option to specify an alternative frontal sequence of the elements is allowed for in records G1 and G2. If this alternative element number is specified (IRNFR = 1) by the user then the elements are assembled in the sequence as specified in record G2. If no alternative element numbering is provided (IRNFR = 0) then the elements are assembled in the same sequence as presented in records H. However, the results output at the end of analysis (stresses at integration points for each element) will be printed in the ascending order of element numbering adopted by the user.

Records I and J

The element number is followed by nodes N1 and N2 (which are at either end of the side) to identify the element side. Then the nodal co-ordinates of nodes along the element side are given *in sequence from node N1 to N2* (note that the co-ordinates of nodes N1 and N2 are not specified).

The program only uses the pore pressure node co-ordinates for plotting purposes. The user should calculate co-ordinates which are consistent with those of the displacement nodes on the curved element side. When performing calculations involving the pore pressure nodes (e.g. interpolating pore pressures inside elements for nodal values) the program makes the implicit assumption that the pore pressure nodes are positioned in a definite relation to the displacement nodes (which define the element geometry). The fact that the program does not actually need to use these co-ordinates in any calculations might appear surprising at first!

Record K

The facility to stop the analysis at different stages of the program is useful in checking the finite element mesh before launching on a complete analysis. From past experience most of the data errors occur in specifying the finite element mesh. This intermediate step enables the user to split a complete analysis into three distinct parts: (a) geometry, (b) *in situ* stresses and (c) analysis.

In view of the costliness of finite element analysis it is sensible to make sure that as far as possible no errors in parts (a) and (b) are present before doing the

analysis. Some data errors in (c) cannot be readily checked. For example, a value of E (Young's modulus) may be incorrectly specified as 300 instead of 3000. Some programs may check that E has a positive value. In CRISP, no checks are carried out on the material properties. A zero value for permeability in a consolidation analysis causes the analysis to fail with a ZERO PIVOT error in the solution routine FRONTZ. If the unit weight of water is specified as zero in a consolidation analysis the program will terminate with an error message saying that there is an attempt to divide by zero, in routine LSTIFF.

IDCHK = 1: the program runs the geometry part of the program and creates a plot data (PD) file which is then used by a separate program (mesh-plotting program; see Appendix B) to draw the mesh.

IDCHK = 2: this will run the geometry part of the program and then setup the *in situ* stresses at all integration points. Also it will carry out an equilibrium check to ensure the boundary conditions (restraints and loads) are equal to the *in situ* element stresses.

Record L1

NMAT must be equal to the number of different material zones specified in the geometry part of the program.

$$\text{INCF} \geqslant \text{INCS}$$

If INCS > 1 then this analysis is a continuation of a previous analysis (see section 9.2.5) and records O to Q3 are omitted.

IPRIM

CRISP allows soil constructions or excavations to be modelled in an analysis via the addition or removal of elements as the analysis proceeds. All the elements that appear at any stage in the analysis must have been included in the input data for the geometry part of the program. IPRIM is the number of finite elements that must be removed to form the **primary** finite element mesh before the analysis is started.

IUPD

IUPD = 0: this corresponds to the normal assumption that is made in linear elastic finite element programs and also in most finite element programs with non-linear material behaviour. External loads and internal stresses are assumed to be in equilibrium in relation to the original (i.e. undeformed) geometry of the finite element mesh. This is usually known as the 'small displacement' assumption.

IUPD = 1: when this option is used the nodal co-ordinates are updated after each increment of the analysis by adding to the co-ordinates the displacements under-

gone by the nodes during the increment. The stiffness matrix of the continuum is then calculated with respect to these new co-ordinates during the next analysis increment. The intention of this process is that at the end of the analysis, equilibrium will be satisfied in the final (deformed) configuration. Although this approach would seem to be intuitively more appropriate when there are significant deformations, it should be noted that it does not constitute a rigorous treatment of the large strain–displacement behaviour for which new definitions of strains and stresses are required (e.g. Carter *et al.*, 1977). Various research workers have examined the influence of a large strain formulation on the load–deformation response calculated by the finite element method using elastic–perfectly-plastic models of soil behaviour. The general conclusion seems to be that the influence of large strain effects is not very significant for the range of material parameters associated with most soils. In most situations the inclusion of large strain effects leads to a stiffer load–deformation response near failure and some enhancement of the load carrying capacity of the soil. If a program user is mainly interested in the estimation of a collapse load using an elastic–perfectly-plastic soil model than it is probably best to use the small displacement approach (i.e. IUPD = 0). Collapse loads can then be compared (and should correspond with those obtained from a classical theory of plasticity approach).

NOIB

The analysis is sub-divided into one or more *increment blocks*. Each increment block consists of one or more increments. The use of the increment block is adopted for two reasons: (a) removal of elements (excavation) and addition of elements (construction) can be carried out over a number of increments and (b) with repeated application of loading (or non-zero prescribed displacements) increments can be grouped together as an increment block (provided that no boundary conditions have changed) thereby reducing the amount of data input.

Record L2

This permits the user to reduce the printed output by suppressing the printing of nodal loads and boundary conditions and the reactions in each increment by setting IBC = 0 and IRAC = 0 respectively.

The next four parameters control the displacement output of each increment. Because there are two separate ranges of numbers, two for vertex nodes and the other two are used for midside nodes.

$$0 \leqslant \text{NVOS} \leqslant \text{NVOF} \leqslant \text{MUMAX}$$

$$0 \leqslant \text{NMOS} \leqslant \text{NMOF} \leqslant \text{NNZ}$$

This permits the user to request only the nodal output for nodes within the specified ranges. For example, if the user is interested in the vertex nodes 5 to 10 and other nodes 780 to 790, then NVOS = 5, NVOF = 10, NMOS = 780 and NMOF = 790. Since these parameters operate in conjunction with the parameter

IOUT in record R, IOUT must be set to 2. If IOUT is set to 1 for the above case, no displacements for the midside node are printed (see also explanations for parameter IOUT under record R).

The same option applies to output from elements. The output from only the user specified range of elements is printed. Here again this option is affected by the parameter IOUT in record R.

Record M

The parameters hown in the material properties' table in the input specification have the meanings shown below. With a few possible exceptions (mentioned later) all the parameters should be regarded as being effective stress properties, i.e. they either relate changes in strain to changes in effective stresses or describe the soil's strength in terms of the effective stresses that are acting in the soil skeleton.

Anisotropic elastic properties

The anisotropic elastic properties relate strains to changes in stress via the following equations:

$$\epsilon_x = \frac{1}{E_h}\sigma_x - \frac{\nu_{vh}}{E_v}\sigma_y - \frac{\nu_{hh}}{E_h}\sigma_z,$$

$$\epsilon_y = -\frac{\nu_{hv}}{E_h}\sigma_x + \frac{1}{E_v}\sigma_y - \frac{\nu_{hv}}{E_h}\sigma_z,$$

$$\epsilon_z = -\frac{\nu_{hh}}{E_h}\sigma_x - \frac{\nu_{vh}}{E_v}\sigma_y + \frac{1}{E_h}\sigma_z,$$

$$\gamma_{xy} = \frac{1}{G_{hv}}\tau_{xy}.$$

Note that suffixes 'h' (for horizontal) and 'v' (for vertical) have been adopted here to clarify the type of anisotropic properties which the program expects to be specified for soil. This is because soil deposits are often formed by a process of sedimentation in horizontal layers and the associated soil fabric and stress history lead to one set of properties for the x–z (or h) plane (E_h and ν_{hh}) and another set relating to the vertical direction (v or y) and the coupling between horizontal and vertical directions (E_v, ν_{hh}, ν_{hv}, G_{hv}). The significance of these properties can be deduced from the above equations, but the following may make the meanings clearer:

an increase in vertical stress leads to an increase in vertical strain $\Delta\sigma_y/E_v$ and a tensile strain $(\nu_{vh}/E_v)\Delta\sigma_y$ (in the absence of any changes in horizontal stresses). Hence ν_{vh} is the Poisson's ratio which gives the ratio of horizontal strain to vertical strain caused by a stress increment in the vertical direction and a similar statement can be made as to the meaning of ν_{hv}.

Note, however, that the program requires only the specification of ν_{vh} and not ν_{hv}. This is because energy/reversibility considerations for an elastic material lead to the relationship

$$\frac{\nu_{hv}}{E_h} = \frac{\nu_{vh}}{E_v}.$$

Elastic, linear variation with depth

The elastic Young's modulus at a depth y is given by the equation

$$E = E_0 + m(y_0 - y).$$

However, Poisson's ratio is assumed to be a constant.

Critical state parameters

The selection of critical state parameters is discussed in Chapter 5.

α

K_w is the bulk modulus of water, which is defined as $\alpha K'$. When an undrained analysis is performed, K_w is normally set to a value between 50 and 500 times K' (i.e. α in the range 50 to 500). The reason for this will be made clear following a description of how the program uses this value. The effective stress law can be written in matrix notation:

$$\sigma = \sigma' + \mathbf{m}u.$$

Here u is the pore water pressure and $\mathbf{m}$ is a vector indicating which stress terms participate in the effective stress relation. For example, if a fully three-dimensional stress condition is considered:

$$\sigma = [\sigma_x \quad \sigma_y \quad \sigma_z \quad \tau_{xy} \quad \tau_{yz} \quad \tau_{zx}]^T,$$
$$\sigma' = [\sigma'_x \quad \sigma'_y \quad \sigma'_z \quad \tau_{xy} \quad \tau_{yz} \quad \tau_{zx}]^T,$$

and

$$\mathbf{m} = [1 \quad 1 \quad 1 \quad 0 \quad 0 \quad 0]^T.$$

Suppose an element of soil undergoes an incremental total stress change $\Delta\sigma$ which results in a change of pore pressure Δu and incremental strains $\Delta\epsilon$. Suppose also that incremental effective stresses are related to incremental strains by the relationship

$$\Delta\sigma' = \mathbf{D}'\Delta\epsilon$$

($\mathbf{D}'$ may describe either an elastic or an elasto-plastic law). The assumption is now made that the volumetric strain experienced by the soil is due entirely to a change in the volume of pore water. The volumetric strain experienced by the soil element can be written as $\mathbf{m}^T\Delta\epsilon$, and the volumetric strain experienced by

the pore water is equal to $[(1+e)/e]\,\mathbf{m}^T\Delta\epsilon$, where e is the current voids ratio. Then the change in pore water pressure is given by

$$\Delta u = K_w[(1+e)/e]\,\mathbf{m}^T\Delta\epsilon.$$

Combining this with the effective stress law and the incremental effective stress–strain relation, the following equation is obtained:

$$\Delta\sigma = \mathbf{D}'\Delta\epsilon + \mathbf{m}\,K_w[(1+e)/e]\,\mathbf{m}^T\Delta\epsilon.$$

CRISP uses this equation in the following way.

(i) The program expects in an undrained analysis that the material properties supplied relate to changes in effective stress.
(ii) When calculating the element stiffness matrices the program adds in the terms corresponding to the volumetric stiffness of the pore water.
(iii) Following the solution of the finite element equations the program calculates the changes in effective stresses and pore water pressure separately.

In a drained analysis the user sets $\alpha = 0$ (i.e. $K_w = 0$) and no changes in pore pressure are calculated. For elastic material behaviour the above procedure for an undrained analysis is equivalent to using a value of Poisson's ratio close to 0.5. However, the above procedure has the advantage that the pore pressure changes are calculated explicitly, and exactly the same technique is valid for an elasto-plastic material law. It is well known that in conventional linear elastic finite element analysis the use of a value very close to 0.5 can lead to numerical ill-conditioning of the finite element equations. The use of a value of α in the range suggested above is equivalent to the use of a value of Poisson's ratio in the range 0.49 to 0.499 and should give reasonably accurate results.

γ

γ is the bulk unit weight of the soil. This value is used by the program when

(a) calculating implicit loads caused by excavation (removal of elements) or construction (addition of elements) sequences
(b) the gravity acceleration field is increased (or decreased) during an analysis (e.g. during geotechnical centrifuge test) (see record R).

Records O, P1, P2 and P3

In an elasto-plastic analysis the stiffness matrix of a finite element will be dependent on the stress state within the element. In general the stress state will vary across an element and the stiffness terms are calculated by integrating expressions dependent on these varying stresses over the volume of each element. CRISP integrates these expressions numerically by 'sampling' the

stresses at particular points within the element and then using standard numerical integration rules for *triangular* areas.

The purpose of record types O, P1, P2 and P3 is to enable the program to calculate the stresses (and for Cam-clay the size of the yield locus specified by p_c') before the analysis starts. For the purpose of specifying the *in situ* stresses the mesh is divided into a number of horizontal layers (option 1). For most problems the *in situ* stresses do not vary in the horizontal direction, and it is assumed that the stresses vary only with depth. Therefore the user specifies a set of *in situ* nodes along a vertical section and the stresses at these points. The *in situ* stresses at the integration points (see Fig. 7.2) are interpolated from the stresses specified at the *in situ* nodes.

However, for problems where the stresses do vary in the horizontal direction, a separate option (option 2: KT = 2) is provided in specifying the stresses. In this option (see records P2 and P3) the user has to specify directly the *in situ* stresses at each integration point for all the elements.

For Cam-clays it is important to try to establish the *in situ* stress state as accurately as possible. This is discussed in Chapter 5 to which the reader is referred.

Records Q1, Q2 and Q3

The user has to specify the external loading (pressure loading along the boundary) and self-weight loading (due to body forces) that is in equilibrium with the *in situ* stresses. The zero displacement boundary condition has to be specified along the boundary that is supported (or restrained). In specifying these conditions the user must consider the entire boundary of the mesh and ensure that along any part of the boundary which is loaded (i.e. not free of stress) either the pressure loading or the restraint has to be specified.

The specified loading is expected to be in equilibrium with the *in situ* stresses. An equilibrium check is carried out, and any imbalance in nodal loads (between the external load and *in situ* stresses) is printed out.

Record R

When a non-linear or consolidation analysis performed using CRISP it is necessary to divide either the loading or the time span of the analysis (or both if there is consolidation with non-linear material properties) into a number of increments. Thus if a total stress of 20 kPa is applied to part of the boundary of the finite element mesh it might be divided into ten equal increments of 2 kPa, each of which is applied in turn. The total number of increments that are necessary will vary from problem to problem, but in general about 50 increments would be required in a drained or undrained analysis using one of the Cam-clays which goes as far as **collapse**. CRISP calculates the incremental displacements for each increment using a tangent stiffness approach, i.e. the current stiffness properties are based on the stress at the start of each increment. While it is desirable to use as many increments as possible to obtain accurate

results, the escalating computer costs that this entails will inevitably mean that some compromise is made between accuracy and cost. The recommended way of reviewing the results to determine whether enough increments have been used in an analysis is to examine the values of yield ratio (YR) at each integration point. When plastic hardening is taking place the value of YR gives the ratio the size of yield locus following the increment to the size before the increment. Thus a value of 1.10 means that the yield locus has grown in size by 10%. Values of about 1.02 (0.98, if softening) are generally regarded as leading to sufficiently accurate calculations. If values greater than 1.05 (less than 0.95, if softening) are seen, then the size of the load increments should be reduced. When one of the Cam-clay models is softening (i.e. yielding dry of critical), smaller increments (than the size suggested by the above discussion) may be necessary.

The time intervals for consolidation analysis (DTIME) should be chosen after giving consideration to the following factors (see also the discussion in Chapter 3 relating to the TINY program):

(i) the amount of pore pressure dissipation expected within the time step;
(ii) in a non-linear analysis the increments of effective stress must not be too large (i.e. the same criteria apply as for a drained or undrained analysis);
(iii) it is a good idea to use the same number of increments in each log cycle of time (thus for linear elastic analysis the same number of time increments would be used in carrying the analysis forward from one day to ten days as from ten days to hundred days). Not less than three time steps should be used per log cycle of time (for a log base of ten). Thus a suitable scheme might be:

Increment no.	DTIME	Total time
1	1	1
2	1	2
3	3	5
4	5	10
5	10	20
6	30	50
7	50	100
8	100	200
9	300	500
10	500	1000

This scheme would be modified slightly near the start and end of an analysis (see below);
(iv) if a very small time increment is used near the start of the analysis then the finite element equations will be ill-conditioned;
(v) when a change in pore pressure boundary condition is applied the associated time step should be large enough to allow the effect of

consolidation to be experienced by those nodes in the mesh with excess pore pressure variables that are close to the boundary. If this is not done then the solution will predict excess pore pressures that show oscillations (both in time and in space).

The application of (v) will often mean that the true undrained response will not be captured in the solution. The following procedure, however, usually leads to satisfactory results:

(a) apply loads in the first increment (or first few increments for a non-linear analysis), but do not introduce any pore pressure boundary conditions;
(b) introduce the excess pore pressure boundary conditions in the increment following the application of the loads.

Boundary conditions (NLOD, NFIX, ILDF, ITMF)

CRISP allows the user to describe a sequence of increments as an **'increment block'**. This facility is provided for two reasons.

(i) If the loads for each analysis increment had to be specified separately there would be a very large amount of data input needed for most problems. Much of this information would be repeated many times (e.g. which element sides were being loaded).
(ii) When performing an excavation (or construction) analysis the program calculates the implied loadings due to the removal (or addition) of the elements specified by the user. These implied loadings will often be too large to be applied in a single increment when the material behaviour is non-linear. The use of an increment block spreads these implied loads over several increments. (Note that this procedure introduces an extra approximation in the modelling of excavations: the stiffness of an element is removed entirely in the first increment of a block, whereas the loads are spread over all increments in the block.)

The program user should note the significance of specifying *incremental* loads in the input data. The total loads acting at any particular time are given by adding together all the previous incremental loads. Thus if part of the mesh is loaded and then subsequently these loads are removed, it will be necessary to specify *negative* incremental loads. Total loads and total fixities remain in force from incremental block to incremental block if there is no action to remove them.

The following example is intended to clarify these points for a consolidation analysis:

(a) part of the boundary of a soil mass is loaded with a load of ten units (this is applied in ten equal increments);

(b) consolidation takes place for some period of time (over ten increments);
(c) the load is removed from the boundary of the soil mass in five equal increments;
(d) consolidation takes place with no total load acting.

	Loads	
Increment no.	Incremental load applied	Total load acting
1	1	1
2	1	2
3	1	3
4	1	4
5	1	5
6	1	6
7	1	7
8	1	8
9	1	9
10	1	10
11	0	10
12	0	10
.	.	.
.	.	.
.	.	.
21	−2	8
22	−2	6
23	−2	4
24	−2	2
25	−2	0
26	0	0
27	0	0
28	0	0
29	0	0
30	0	0

etc.

One possible way of translating this sequence of loading into input data would be to make increments 1 to 10 the first increment block with an incremental load of 10 units and 10 load factors equal to 0.1. The second increment block (increments 11 to 20) would have no incremental loads and the third (increments 21 to 25) would have an incremental load of −10 with 5 load factors equal to 0.2.

DGRAV

DGRAV is used in problems in which the material's self-weight is increased during an analysis (e.g. in the 'wind-up' stage of a centrifuge test, increasing centrifugal acceleration can be regarded as having this effect).

Records R, T1, U and V

The loading (NLOD), self-weight loads (DGRAV) and prescribed displacements (NFIX) are specified for the entire increment block, and are applicable to that particular increment block. The loading and any non-zero prescribed displacement for the individual increments are taken as ratios (<1) of that for the increment block.

There is no restriction on how these loading and non-zero prescribed displacements are divided among the increments in an increment block. They are *equally* divided between all the increments if ILDF = 0 in record R. However, if the user wants to distribute the loading (and non-zero prescribed displacements) unevenly between the increments, then by setting ILDF = 1 a separate list of load ratios is read in record T1. (This is generally useful in an analysis where large load increments can be applied when the problem is in the elastic state and smaller load increments as plastic yielding takes place.)

It should be noted that the same ratios R(I) etc. (record T1) apply to the pressure loading (NLOD – record U), the gravity loading (DGRAV – record R) and the prescribed displacements (NFIX – record V).

The sum of ratios R(I) must be equal to 1. However, some of these ratios can take zero values, as illustrated in the example given under record T3.

Records R and T3

In a consolidation analysis the time increment DTIME (>0) is specified for the entire increment block. If ITMF = 0 in record R then DTIME is *equally* divided among all the increments in the increment block. However, if ITMF = 1 then the user directly specifies (in record T3) the time increments for each increment. Unlike the load ratios R(I) etc. (in record T1) these are actual time steps for the increments and not *ratios*. None of these can be zero, and for reasons of consistency, DTIME in record R must be set equal to the sum of all the time steps in the increment block.

The use of records T1 and T3 is illustrated by an example. In a consolidation analysis of 100 secs total duration spread over 9 increments, the load is gradually applied in 3 secs and the subsequent transient response is required.

(a) First the example is used to illustrate the use of a single increment block. This option is not applicable if there is a change in pore pressure boundary condition at the end of the loading phase. Then option (b) must be used.

in record R

IBNO	INCA	INCB	ICHEL	NLOD	ILDF	NFIX	IOUT
1	1	9	0	–	1	–	–

IOCD	DTIME	ITMF	DGRAV
–	100	1	0

in record T1

R(1)	R(2)	R(3)	R(4)	R(5)	R(6)	R(7)	R(8)	R(9)
0.33	0.33	0.34		0	0	0	0	0

in record T3

DTM(1)	DTM(2)	DTM(3)	DTM(4)	DTM(5)	DTM(6)
1	1	1	2	5	10

DTM(7)	DTM(8)	DTM(9)
10	20	50

(b) As an alternative, the analysis could be split into two increment blocks. In the first increment block the loading is applied, whereas in the second, consolidation takes place with no change in the load.

record R

IBNO	INCA	INCB	ICHEL	NLOD	ILDF	NFIX	IOUT
1	1	3	0	–	0	–	–
2	4	9	0	–	0	–	–

IOCD	DTIME	ITMF	DGRAV
–	3.	0	0
–	97.	1	0

record T1

not present for both increment blocks (ILDF = 0 in record R)

record T3

	DTM(1)	DTM(2)	DTM(3)	DTM(4)	DTM(5)	DTM(6)
incr block 1	not present (ITMF = 0 in record R)					
incr block 2	2	5	10	10	20	50

record V

Displacement fixity:
Any displacement fixities (i.e. zero prescribed displacements) only need to be specified once, either at the *in situ* stage (in the presence of *in situ* stresses) or in the first increment block. Once specified, these zero displacement (or pore

pressure) fixities remain in effect during the rest of the analysis. Therefore these need not be re-specified for each and every increment block.

Pore pressure fixity 2:
When a fixity code of 1 is used, the incremental changes of excess pore pressures along element sides are treated in exactly the same fashion as incremental displacements. When the incremental change in the excess pore pressure that needs to be prescribed is known (for example along a drainage boundary or along a boundary where a known pressure head is applied) then a fixity code of 1 is used. However, if the pore pressures have changed from the *in situ* values probably owing to loading or unloading then it is not possible to know the incremental changes in the pore pressures beforehand. Then the user has to prescribe the absolute value of the pore pressure using the fixity code of 2. But in CRISP it is not possible to fix the **absolute value** of the pore pressure directly. This has to be done indirectly by fixing the **absolute value of *excess*** pore pressure. Remembering that

$$\text{abs p.p.} = \textit{in situ}\ \text{p.p.} + \text{abs excess p.p.}$$

$$\text{abs excess p.p.} = \text{abs p.p.} - \textit{in situ}\ \text{p.p.}$$

This is illustrated with an example (Fig. 9.2): consider a excavation of a trench in a saturated clay. The trench is excavated in layers of 2 m; assuming the unit weight of water is 10 kN/m^3 the *in situ* values of pore pressure at nodes 1, 2 and 3 are, respectively, 0, 20 and 40 kPa. For node 2, after excavating the first layer, the absolute pore pressure = 0. Therefore absolute excess pore pressure = 0 − 20 = −20 kPa. Similarly after two layers have been excavated the absolute excess pore pressure along the base (at a depth of 4 m) and at nodes 6 and 3 is

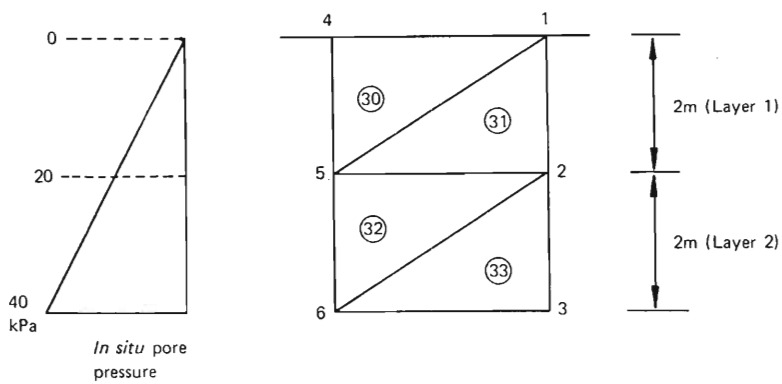


Fig. 9.2 – Example to illustrate pore pressure fixities

given by 0 − 40 = −40 kPa. This particular feature often causes confusion to the user. The most common mistake is to incorrectly fix the **absolute excess pore pressure** to 0 using a fixity code of **2**.

9.2.5 Stop–restart facility

CRISP can be stopped and restarted, allowing a lengthy analysis to be split into a number of shorter analyses. This facility is particularly useful for reviewing and perhaps altering the size of load increments without having to repeat the entire analysis.

The input data for a *starting* run is exactly the same as for a normal run except that ISR (record L1) is set to 1 or 2 rather than zero. When a run is *restarted* ISR is set to 1 or 2 and records O to Q3 are omitted from the input data (in this case the details of the current stresses are read from the restart file).

A value of INCS > 1 on record L1 indicates that this is a restarted run. INCS must follow on in sequence from the previous analysis. When ISR = 1 it is only possible to restart the analysis from the last increment of a previous run. When ISR = 2 it is possible to restart from any previous increment. Mixing ISR = 1 and ISR = 2 in a series of runs is not permitted. The results from a previous run are always read from unit IR1 and the results from the current run are stored on unit IW2. As mentioned in section 4.2.5 restart files for ISR = 2 will be large and probably require use of magnetic tapes.

9.3 LINEAR ELASTIC: ONE-DIMENSIONAL CONSOLIDATION

The first example is identical to the one-dimensional consolidation analyses performed by the TINY program in section 3.6.4. The mesh (Fig. 9.3) consists of 12 LST elements, and the depths of the LST elements are the same as in section 3.6.4. The input data for CRISP are given in Fig. 9.4. The boundary conditions are illustrated in Fig. 9.5. Since the problem is one dimensional, the problem type could be chosen as either plane strain or axisymmetry. In fact a plane strain analysis is performed. The mesh is prevented from moving in the lateral direction. This reduces the problem to its one-dimensional form. The base of the mesh is restrained, which is also an impermeable boundary. The top is a free-draining boundary.

A uniform load of 10 kPa is applied to the surface in the first increment block, which consists of a single increment. The pore pressure boundary condition corresponding to the top drainage surface is applied in the next increment. Exactly the same time steps are used as in section 3.6.4. The second increment block contains 11 increments. It should be emphasized again that loading is specified in one increment and the relevant pore pressure boundary conditions are specified in the next increment. This applies only to pore pressure boundary conditions. In general, either a loading is applied to a node (possibly

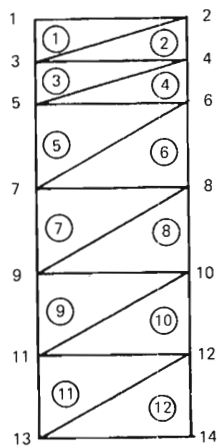


Fig. 9.3 – Mesh for Terzaghi 1-D consolidation (12 LST elements of type 3)

```
record
 A   ¦  ONE DIMENSIONAL TERZAGHI CONSOLIDATION
 B   ¦    14  12  3  3  2  0
 C   ¦    0  0
 D   ¦    0  0  0  0  0  0  0  0  0  0  0
 E   ¦    0  0  0  0
 F   ¦    1  0.  10.
 F   ¦    2  1.  10.
 F   ¦    3  0.   9.
 F   ¦    4  1.   9.
 F   ¦    5  0.   8.
 F   ¦    6  1.   8.
 F   ¦    7  0.   6.
 F   ¦    8  1.   6.
 F   ¦    9  0.   4.
 F   ¦   10  1.   4.
 F   ¦   11  0.   2.
 F   ¦   12  1.   2.
 F   ¦   13  0.   0.
 F   ¦   14  1.   0.
 G1  ¦    0
 H   ¦     1  3  1   2   1   3
 H   ¦     2  3  1   3   4   2
 H   ¦     3  3  1   4   3   5
 H   ¦     4  3  1   5   6   4
 H   ¦     5  3  1   6   5   7
 H   ¦     6  3  1   7   8   6
 H   ¦     7  3  1   8   7   9
 H   ¦     8  3  1   9  10   8
 H   ¦     9  3  1  10   9  11
 H   ¦    10  3  1  11  12  10
 H   ¦    11  3  1  12  11  13
 H   ¦    12  3  1  13  14  12
 K   ¦    0
 L1  ¦    0  1  2  1  12  0  0  0
```



```
 L2  ¦    0  0  1  14  0  0  1  12
 M   ¦    1  1  1.E3  1.E3  0.25  0.25  0.4E3  0  10.  0.  1.E-9  1.E-9
 O   ¦    0  0
 R   ¦    1  1  1  0  -1  0  13  11  0  1.  0  0.
 U   ¦    1  1  2  0.  10.  0.  10.  0.  10.
 V   ¦     1   1  3  1  1  0.  0.  0.
 V   ¦     2   2  4  1  1  0.  0.  0.
 V   ¦     3   3  5  1  1  0.  0.  0.
 V   ¦     4   4  6  1  1  0.  0.  0.
 V   ¦     5   5  7  1  1  0.  0.  0.
 V   ¦     6   6  8  1  1  0.  0.  0.
 V   ¦     7   7  9  1  1  0.  0.  0.
 V   ¦     8   8  10  1  1  0.  0.  0.
 V   ¦     9   9  11  1  1  0.  0.  0.
 V   ¦    10  10  12  1  1  0.  0.  0.
 V   ¦    11  11  13  1  1  0.  0.  0.
 V   ¦    12  12  14  1  1  0.  0.  0.
 V   ¦    12  13  14  2  1  0.  0.  0.
 R   ¦    2  2  12  0  0  0  1  11  0  2.E9  1  0.
 T3  ¦    1.E6  1.E6  2.E6  6.E6  1.E7  2.E7  6.E7
 T3  ¦    1.E8  2.E8  6.E8  1.E9
 V   ¦     1   1  2  3  2  0.  0.  0.
```

Fig. 9.4 – Input data for Terzaghi 1-D consolidation

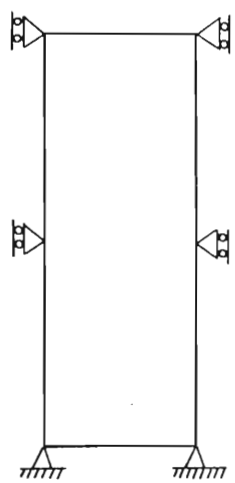


Fig. 9.5 – Boundary conditions for Terzaghi 1-D consolidation

zero) or it is restrained. Loads are applied, and only have effect on free nodes. There is no point in applying a load to a restrained node or a node with prescribed displacements (the effect would be the same as applying no load at all – think of a giant hand restraining a node or moving it by a prescribed amount).

The time step for the increment in which load is applied is chosen such that no dissipation would be expected. The choice of time steps was discussed in section 3.6.4. Exactly the same results were obtained as presented in Chapter 3, confirming that both programs are similar (Fig. 9.6).

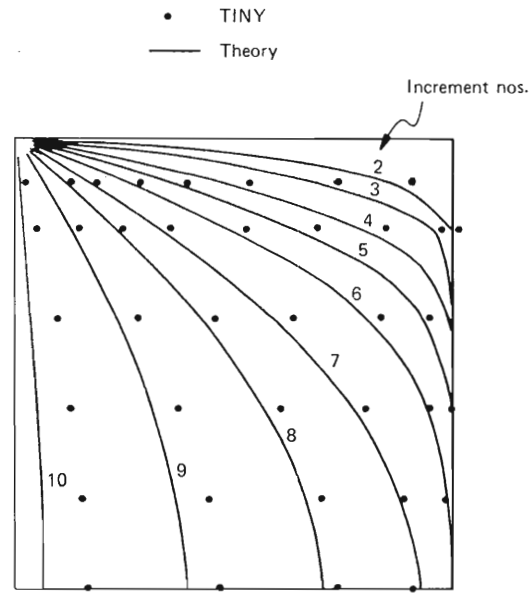


Fig. 9.6 – Plot of degree of consolidation against T_v for Terzaghi 1-D consolidation

9.4 ELASTIC ANALYSES

The next series of examples is chosen to illustrate different aspects of CRISP. A single mesh as shown in Fig. 9.7 is used in all the examples. It consists of 80 LST elements and 54 vertex nodes.

The problem considered is a linear elastic layer of finite depth subjected to a uniform circular surface pressure. The following material properties have been chosen for the elastic layer:

$$E = 3000 \text{ kPa}, \quad \nu = 0.25 \qquad \text{(hence } G = 1200 \text{ kPa)}.$$

The applied pressure is 30 kPa. The boundary condition for the mesh is as follows: the outer vertical boundary is restrained in the x direction and is assumed to be smooth, i.e. free to move in the y direction. The base of the layer is assumed to be rough and hence is restrained in both x and y directions. The y axis, being the axis of symmetry, is restrained in the x direction but is free to move in the y direction (Fig. 9.8). In fact in axisymmetric problems it is not

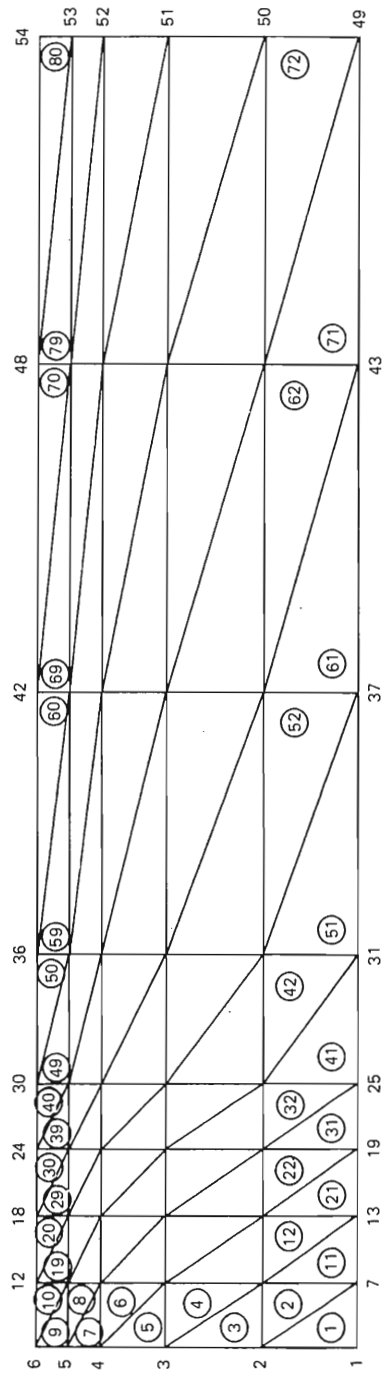


Fig. 9.7 – Mesh for example problem

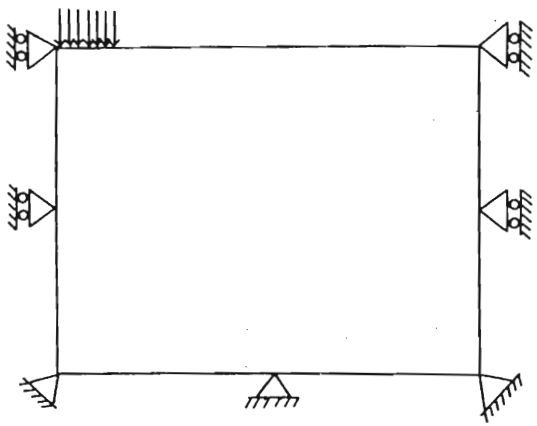


Fig. 9.8 – Boundary conditions used in analysis

necessary to restrain the axis of symmetry but this was done here. The mesh represents a radial section of the axisymmetric problem. All calculations are carried out over a full rotation (2π) of this radial section.

9.4.1 Linear elastic – drained analysis

The first anlysis is a **drained** one. Since $K_w = \alpha K'$, α, which is the 7th material property in the list of material properties, is set to zero. All other material properties are set to zero excepi for the elastic properties. Since this is a linear elastic analysis, the load is applied in a single increment. The input data are given in Fig. 9.9.

```
record
 A   ¦ CIRCULAR LOAD ON AN ELASTIC FOUNDATION
 B   ¦    54   80    3    2    2    8
 C   ¦     0    0
 D   ¦     0    0    0    0    0    0    0    0    0    0
 E   ¦     0    0    0    0
 F   ¦     1      0.000      0.000
 F   ¦     2      0.000      3.000
 F   ¦     3      0.000      6.000
 F   ¦     4      0.000      8.000
 F   ¦     5      0.000      9.000
 F   ¦     6      0.000     10.000
 F   ¦     7      2.000      0.000
 F   ¦     8      2.000      3.000
 F   ¦     9      2.000      6.000
 F   ¦    10      2.000      8.000
 F   ¦    11      2.000      9.000
 F   ¦    12      2.000     10.000
 F   ¦    13      4.000      0.000
 F   ¦    14      4.000      3.000
 F   ¦    15      4.000      6.000
 F   ¦    16      4.000      8.000
 F   ¦    17      4.000      9.000
```



```
 F   ¦    18      4.000     10.000
 F   ¦    19      6.000      0.000
 F   ¦    20      6.000      3.000
 F   ¦    21      6.000      6.000
 F   ¦    22      6.000      8.000
 F   ¦    23      6.000      9.000
 F   ¦    24      6.000     10.000
 F   ¦    25      8.000      0.000
 F   ¦    26      8.000      3.000
 F   ¦    27      8.000      6.000
 F   ¦    28      8.000      8.000
 F   ¦    29      8.000      9.000
 F   ¦    30      8.000     10.000
 F   ¦    31     12.000      0.000
 F   ¦    32     12.000      3.000
 F   ¦    33     12.000      6.000
 F   ¦    34     12.000      8.000
 F   ¦    35     12.000      9.000
 F   ¦    36     12.000     10.000
 F   ¦    37     20.000      0.000
 F   ¦    38     20.000      3.000
 F   ¦    39     20.000      6.000
 F   ¦    40     20.000      8.000
 F   ¦    41     20.000      9.000
 F   ¦    42     20.000     10.000
 F   ¦    43     30.000      0.000
 F   ¦    44     30.000      3.000
 F   ¦    45     30.000      6.000
 F   ¦    46     30.000      8.000
 F   ¦    47     30.000      9.000
 F   ¦    48     30.000     10.000
 F   ¦    49     40.000      0.000
 F   ¦    50     40.000      3.000
 F   ¦    51     40.000      6.000
 F   ¦    52     40.000      8.000
 F   ¦    53     40.000      9.000
 F   ¦    54     40.000     10.000
 G1  ¦     0
 H   ¦     1    2    1    1    7    2
 H   ¦     2    2    1    2    7    8
 H   ¦     3    2    1    2    8    3
 H   ¦     4    2    1    3    8    9
 H   ¦     5    2    1    3    9    4
 H   ¦     6    2    1    4    9   10
 H   ¦     7    2    1    4   10    5
 H   ¦     8    2    1    5   10   11
 H   ¦     9    2    1    5   11    6
 H   ¦    10    2    1    6   11   12
 H   ¦    11    2    1    7   13    8
 H   ¦    12    2    1    8   13   14
 H   ¦    13    2    1    8   14    9
 H   ¦    14    2    1    9   14   15
 H   ¦    15    2    1    9   15   10
 H   ¦    16    2    1   10   15   16
 H   ¦    17    2    1   10   16   11
 H   ¦    18    2    1   11   16   17
 H   ¦    19    2    1   11   17   12
 H   ¦    20    2    1   12   17   18
 H   ¦    21    2    1   13   19   14
 H   ¦    22    2    1   14   19   20
```

```
H  |   23   2   1   14   20   15
H  |   24   2   1   15   20   21
H  |   25   2   1   15   21   16
H  |   26   2   1   16   21   22
H  |   27   2   1   16   22   17
H  |   28   2   1   17   22   23
H  |   29   2   1   17   23   18
H  |   30   2   1   18   23   24
H  |   31   2   1   19   25   20
H  |   32   2   1   20   25   26
H  |   33   2   1   20   26   21
H  |   34   2   1   21   26   27
H  |   35   2   1   21   27   22
H  |   36   2   1   22   27   28
H  |   37   2   1   22   28   23
H  |   38   2   1   23   28   29
H  |   39   2   1   23   29   24
H  |   40   2   1   24   29   30
H  |   41   2   1   25   31   26
H  |   42   2   1   26   31   32
H  |   43   2   1   26   32   27
H  |   44   2   1   27   32   33
H  |   45   2   1   27   33   28
H  |   46   2   1   28   33   34
H  |   47   2   1   28   34   29
H  |   48   2   1   29   34   35
H  |   49   2   1   29   35   30
H  |   50   2   1   30   35   36
H  |   51   2   1   31   37   32
H  |   52   2   1   32   37   38
H  |   53   2   1   32   38   33
H  |   54   2   1   33   38   39
H  |   55   2   1   33   39   34
H  |   56   2   1   34   39   40
H  |   57   2   1   34   40   35
H  |   58   2   1   35   40   41
H  |   59   2   1   35   41   36
H  |   60   2   1   36   41   42
H  |   61   2   1   37   43   38
H  |   62   2   1   38   43   44
H  |   63   2   1   38   44   39
H  |   64   2   1   39   44   45
H  |   65   2   1   39   45   40
H  |   66   2   1   40   45   46
H  |   67   2   1   40   46   41
H  |   68   2   1   41   46   47
H  |   69   2   1   41   47   42
H  |   70   2   1   42   47   48
H  |   71   2   1   43   49   44
H  |   72   2   1   44   49   50
H  |   73   2   1   44   50   45
H  |   74   2   1   45   50   51
H  |   75   2   1   45   51   46
H  |   76   2   1   46   51   52
H  |   77   2   1   46   52   47
H  |   78   2   1   47   52   53
H  |   79   2   1   47   53   48
H  |   80   2   1   48   53   54
K  |    0
L1 |    1   1   1    1    1    0    0    1
```



```
L2 |     0    0    6    18   771   787   0   0
M  |  1  1  3.E3  3.E3  0.25  0.25  1.2E3  0.  0.  0.  0.  0.
O  |   0  0
R  |  1  1  1  0  -2  0  26  2  0   0.0  0   0.0
U  |  10   6  12  0.0  30.0  0.0  30.0  0.0  30.0
U  |  20  12  18  0.0  30.0  0.0  30.0  0.0  30.0
V  |     1    1    2    1    1   0.  0.  0.
V  |     3    2    3    1    1   0.  0.  0.
V  |     5    3    4    1    1   0.  0.  0.
V  |     7    4    5    1    1   0.  0.  0.
V  |     9    5    6    1    1   0.  0.  0.
V  |     1    1    7    1    1   0.  0.  0.
V  |     1    1    7    2    1   0.  0.  0.
V  |    11    7   13    1    1   0.  0.  0.
V  |    11    7   13    2    1   0.  0.  0.
V  |    21   13   19    1    1   0.  0.  0.
V  |    21   13   19    2    1   0.  0.  0.
V  |    31   19   25    1    1   0.  0.  0.
V  |    31   19   25    2    1   0.  0.  0.
V  |    41   25   31    1    1   0.  0.  0.
V  |    41   25   31    2    1   0.  0.  0.
V  |    51   31   37    1    1   0.  0.  0.
V  |    51   31   37    2    1   0.  0.  0.
V  |    61   37   43    1    1   0.  0.  0.
V  |    61   37   43    2    1   0.  0.  0.
V  |    71   43   49    1    1   0.  0.  0.
V  |    71   43   49    2    1   0.  0.  0.
V  |    72   49   50    1    1   0.  0.  0.
V  |    74   50   51    1    1   0.  0.  0.
V  |    76   51   52    1    1   0.  0.  0.
V  |    78   52   53    1    1   0.  0.  0.
V  |    80   53   54    1    1   0.  0.  0.
```

Fig. 9.9 – Input data for linear elastic (drained) analysis

The calculated central and edge settlements by CRISP are 55 and 30 mm respectively. Poulos (1967) has presented theoretical solutions which give a central displacement of 55 mm for a layer of the same thickness. In a separate solution for a rough-based layer but with $\nu = 0.3$ the theory predicts 52 mm for the central displacement and 27 mm for the edge settlement (Milovic, 1970). Harr (1966) (see Table 5.1 of Poulos and Davis, 1974) has also presented an approximate solution, which gives an edge settlement of 31 mm for the above problem. The comparison is good and the reader can see the authors following their own advice in section 9.2 in calibrating the program against solutions.

9.4.2 Non-homogeneous elastic model – drained analysis

In this analysis the variation of Young's modulus is as shown in Fig. 9.10. A value of 2000 kPa is assumed at the surface and there is a linear increase to 4000 kPa at the base of the layer. This gives an average E value of 3000 kPa, which is the same as that for the linear elastic analysis. The only difference to the input is the material properties (record M), which are as follows:

Record

```
M | 1  2  2000.  10.  200.  0.25  0.  0.  0.  0.  0.  0.
```

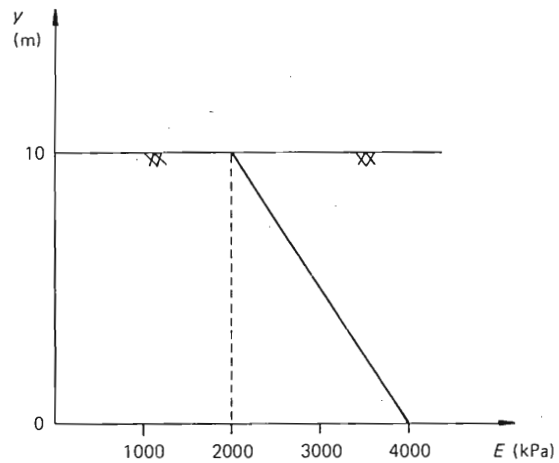


Fig. 9.10 – Variation of Young's modulus with depth for non-homogeneous elastic model

Poisson's ratio is assumed constant throughout and equal to 0.25. The central displacement was calculated to be 64 mm.

9.4.3 Linear elastic – undrained analysis

The only difference between an undrained analysis and a drained analysis is that the undrained analysis requires the specification of the parameter α. This is used in the calculation of an equivalent bulk modulus for water within the program. A value of 100 is chosen and the only difference to the input data is record M.

Record

M | 1 1 3000. 3000. 0.25 0.25 1200. 0. 100. 0. 0. 0.

The calculated central settlement is 33 mm.

9.4.4 Linear elastic – consolidation analysis

The elastic parameters are the same as for the drained analysis. The additional parameters that need to be specified are the unit weight of water, which is taken as 10 kN/m^3, and the permeabilities in the x and y directions, which are taken as equal to 10^{-8} m/s.

$$7\text{th property} = \gamma_w = 10\ \text{kN/m}^3,$$

$$9\text{th property} = k_x = 10^{-8}\ \text{m/s},$$

$$10\text{th property} = k_y = 10^{-8}\ \text{m/s}.$$

The initial stresses are assumed to be zero as in the case of the previous analyses. Because this is a consolidation analysis, the element type is 3. Again the load is

equal to 30 kPa and is applied in the first increment over a time step of 1 s. The top surface is assumed to be a drainage boundary and all other boundaries are assumed to be impermeable. The pore pressure boundary condition is applied by using a fixity code of 2 along element sides on the surface. In the second increment block, which consists of 9 increments, the following time steps were used.

10000. 10000. 20000. 60000.
100000. 200000. 600000.
1000000. 2000000.

The input data are shown in Fig. 9.11.

```
record
 A   |   CIRCULAR LOAD ON AN ELASTIC FOUNDATION - CONSOLIDATION
 B   |     54   80    3    3    2    8
 C   |      0    0
 D   |      0    0    0    0    0    0    0    0    0    0
 E   |      0    0    0    0
 F   |      1     0.000     0.000
 F   |      2     0.000     3.000
 F   |      3     0.000     6.000
                 ..........
                 ..........
 F   |     54    40.000    10.000
 G1  |      0
 H   |      1    3    1    1    7    2
 H   |      2    3    1    2    7    8
 H   |      3    3    1    2    8    3
 H   |      4    3    1    3    8    9
 H   |      5    3    1    3    9    4
                 ..........
                 ..........
 H   |     80    3    1   48   53   54
 K   |      0
 L1  |      1    1    2    1   10    0    0    1
 L2  |      0    0    1   50  771  787    1   40
 M   |  1  1  3.E3  3.E3  0.25  0.25  1.2E3  0.  10.  0.  1.E-8  1.E-8
 O   |  0  0
 R   |  1  1  1  0  -2  0  26  12  0  1.0  0  0.0
 U   |  10   6  12  0.0  30.0  0.0  30.0  0.0  30.0
 U   |  20  12  18  0.0  30.0  0.0  30.0  0.0  30.0
 V   |     1    1    2    1    1   0.   0.   0.
 V   |     3    2    3    1    1   0.   0.   0.
 V   |     5    3    4    1    1   0.   0.   0.
                 ............
                 ............
 V   |    76   51   52    1    1   0.   0.   0.
 V   |    78   52   53    1    1   0.   0.   0.
 V   |    80   53   54    1    1   0.   0.   0.
 R   |   2  2  10  0  0  0  8  12  0  4.E6  1  0.0
 T3  |   1.E4  1.E4  2.E4  6.E4  1.E5  2.E5  6.E5  1.E6  2.E6
 V   |    10   6  12   3   2  0.   0.   0.
 V   |    20  12  18   3   2  0.   0.   0.
 V   |    30  18  24   3   2  0.   0.   0.
 V   |    40  24  30   3   2  0.   0.   0.
```

```
V   ¦   50  30  36   3   2  0.  0.  0.
V   ¦   60  36  42   3   2  0.  0.  0.
V   ¦   70  42  48   3   2  0.  0.  0.
V   ¦   80  48  54   3   2  0.  0.  0.
```

Fig. 9.11 – Input data for consolidation analysis

At the end of the 10th increment it was found that not all excess pore pressures have dissipated. In this run the results of the 10th increment (i.e. the last increment) had been written to a disk file because the stop–restart facility was being used (see record L1). It was decided to continue the analysis and apply a further 4 increments with the following time steps.

6000000 10000000 20000000 60000000

The nodal co-ordinates and element–nodal connectivity list are always included in the input data. Therefore the input data for the restarted run are the same as far as the first A to K records are concerned. The rest of the input data are shown in Fig. 9.12. It should be noted that ISR = 1 in record L1 to indicate that the option to stop–restart the analysis is being used, and INCS is set to 11 to indicate that results at the end of increment 10 are stored in a disk file. For a restarted analysis, records O to Q3 are omitted. The boundary conditions need not be specified again, as there are no changes to them.

```
record
 K   ¦  0
 L1  ¦  1    1    1    11   14   0    0    1
 L2  ¦  0    0    1    50   771  787  1    40
 M   ¦  1  1  3.E3  3.E3  0.25  0.25  1.2E3  0.  10.  0.  1.E-8  1.E-8
 R   ¦  1  11  14  0  0  0  0  12  0  9.6E7  1  0.0
 T3  ¦  6.E6  1.E7  2.E7  6.E7
```

Fig. 9.12 – Input data for restarted consolidation analysis

The results from the drained, undrained and consolidation analyses are compared in Fig. 9.13, where the average settlement is plotted against $\log_{10}$(time). As one would expect, the immediate settlement in the consolidation analysis is equal to the one obtained from the undrained analysis. After all the pore pressures are dissipated, the final settlement is equal to the one from the drained analysis. Also shown in this figure is the settlement from the drained analysis using the non-homogeneous elastic model.

Booker and Randolph (1984) present theoretical solutions for the consolidation of a semi-infinite elastic medium under a uniform surface loading over a circular area. They define the degree of consolidation as U:

$$U = \frac{w(t) - w(0+)}{w(\infty) - w(0+)}, \tag{9.1}$$

where $w(t)$ is the average settlement of the loaded area at time t. This solution is compared in Fig. 9.14 with the CRISP results in a plot of $c_v t/a^2$ against U, where a is the radius of the loaded area. The comparison is fairly good for the

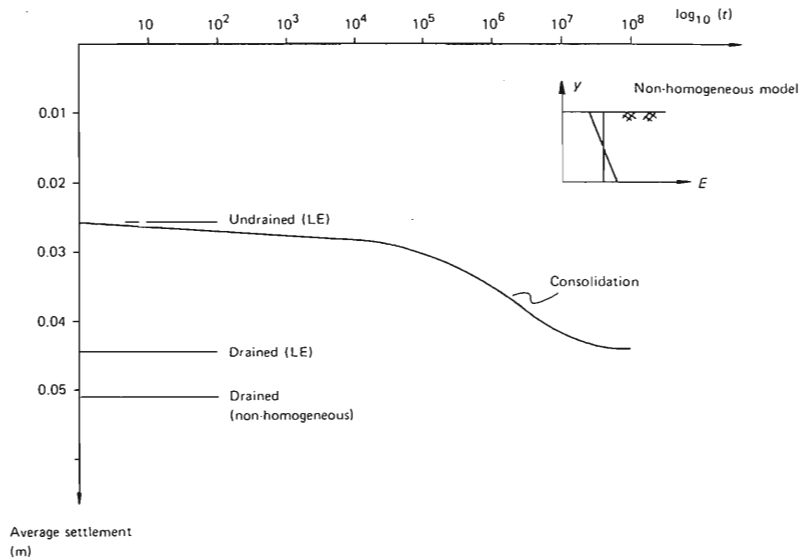


Fig. 9.13 – Comparison of average settlement in different types of analysis

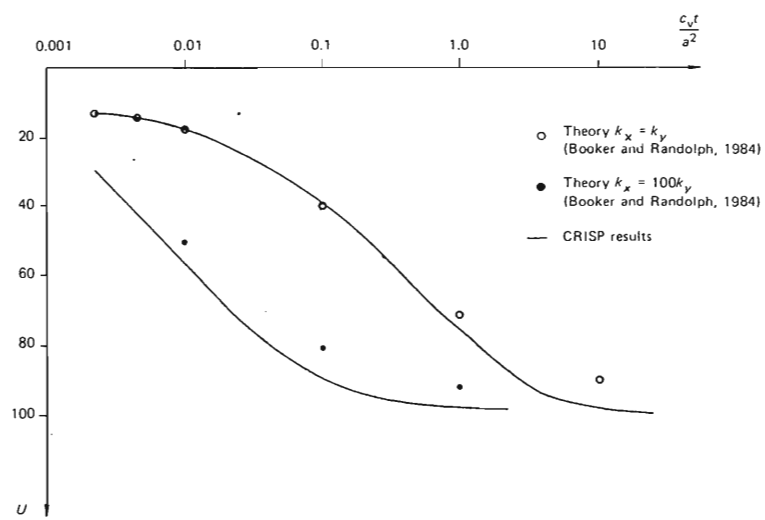


Fig. 9.14 – Comparison of finite element results with theory

case $k_x = k_y$ bearing in mind that the CRISP results are for a layer of finite thickness. In a separate analysis, $k_x = 100k_y$, and the comparison is reasonable. Booker and Randolph give the final central settlement as 75 mm. This is the same result as given by Poulos and Davis (1974) for an elastic half-space. By deducting the settlement at a depth equal to the thickness of the layer used in the CRISP analysis, a value of 66 mm was obtained. This can be compared with the CRISP result of 55 mm.

9.5 UNDRAINED ANALYSIS – CAM-CLAY

9.5.1 Undrained analysis – normally consolidated clay

To illustrate the use of the critical state model, first the 10 m layer is assumed to be one-dimensionally normally consolidated. An initial stress state of a typical point is denoted by A in Fig. 9.15. The point A lies on the K_{nc} line as well as on the yield locus.

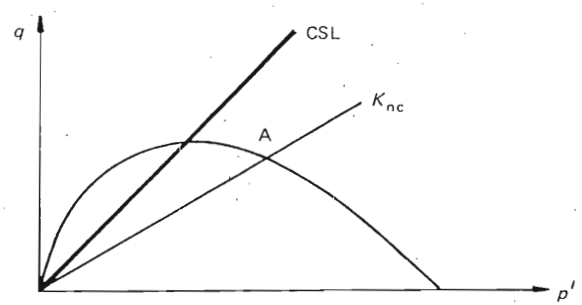


Fig. 9.15 – *In situ* stress state for one-dimensionally normally consolidated soil

The principal difference from the earlier elastic analysis is the specification of initial stresses. In an elastic analysis the initial stresses do not affect the results in any way and hence usually are taken as zero. If one considers a point at a depth of 10 m and taking the unit weight of saturated soil and water as 20 and 10 kN/m^3 respectively,

$$\sigma_v = 20 \times 10 = 200 \text{ kPa}, \quad u_0 = 10 \times 10 \text{ kPa}, \quad \sigma'_v = 100 \text{ kPa}.$$

The value of K_{nc} which is needed for the calculation of σ'_h is calculated from the following expression due to Jaky (1944):

$$K_{nc} = 1 - \sin(\phi'), \tag{5.11 \textit{bis}}$$

where ϕ' is calculated from the equation (c.f. eg. (5.8))

$$\sin(\phi') = \frac{3M}{6 + M}. \tag{9.2}$$

The Cam-clay parameter M was taken as 0.888, which gave $K_{nc} = 0.613$. This in turn gave $\sigma'_h = 61.3$ kPa. This gives

$$q = 100 - 61.3 = 38.7 \text{ kPa},$$

$$p' = (100 + 2 \times 61.3)/3 = 74.2 \text{ kPa}.$$

The size of the yield locus (i.e. p'_c) is calculated from the expression of the Cam-clay yield locus, since the stress state lies on the surface.

$$q = Mp' \ln(p'_c/p'). \tag{9.3}$$

Substituting the above values gives $p'_c = 133.5$. All the stresses are equal to zer along the surface, and the variation is linear with depth.

The other Cam-clay parameters were chosen to be

$$\kappa = 0.062, \quad \lambda = 0.161, \quad \Gamma = 2.759.$$

The part of the input data different from that for the linear elastic analysis is given in Fig. 9.16. Note that the displacement boundary conditions are specified along with the initial stresses.

```
record
 A   ¦  CIRCULAR LOAD ON N.C. CAM-CLAY  *** UNDRAINED
                        ............
                        ............
 L1  ¦       1    1    5    1   30    0    0    1
 L2  ¦       0    0    6   18  771  787    5   30
 M   ¦  1  4  0.062  0.161  1.759  0.888  0.25  0.  100.  20.  0.  0.
 O   ¦     1    2
 P1  ¦     1  0.  61.3  100.0  61.3  0.  100.  0.  133.5
 P1  ¦     2 10.   0.     0.    0.   0.    0.  0.    0.
 Q1  ¦     0  26  1.
 Q3  ¦       1    1    2    1    1   0.  0.  0.
 Q3  ¦       3    2    3    1    1   0.  0.  0.
 Q3  ¦       5    3    4    1    1   0.  0.  0.
                        ............
                        ............
 Q3  ¦      76   51   52    1    1   0.  0.  0.
 Q3  ¦      78   52   53    1    1   0.  0.  0.
 Q3  ¦      80   53   54    1    1   0.  0.  0.
 R   ¦  1  1  5  0  -2  0  0  0  1  0.0  0  0.0
 T2  ¦       112  12   12   12  222
 U   ¦  10   6  12  0.0  1.0  0.0  1.0  0.0  1.0
 U   ¦  20  12  18  0.0  1.0  0.0  1.0  0.0  1.0
 R   ¦  2  6  10  0  -2  0  0  0  1  0.0  0  0.0
 T2  ¦  112  12  12  12  222
 U   ¦  10   6  12  0.0  5.0  0.0  5.0  0.0  5.0
 U   ¦  20  12  18  0.0  5.0  0.0  5.0  0.0  5.0
 R   ¦  3  11  15  0  -2  0  0  0  1  0.0  0  0.0
 T2  ¦  112  12  12  12  222
 U   ¦  10   6  12  0.0  5.0  0.0  5.0  0.0  5.0
 U   ¦  20  12  18  0.0  5.0  0.0  5.0  0.0  5.0
 R   ¦  4  16  20  0  -2  0  0  0  1  0.0  0  0.0
 T2  ¦  112  12  12  12  222
 U   ¦  10   6  12  0.0  5.0  0.0  5.0  0.0  5.0
 U   ¦  20  12  18  0.0  5.0  0.0  5.0  0.0  5.0
```

```
R   ¦  5  21  30  0  -2  0  0  0  1  0.0  0  0.0
T2  ¦  112  12  12  12  222
U   ¦  10  6  12  0.0  5.0  0.0  5.0  0.0  5.0
U   ¦  20  12  18  0.0  5.0  0.0  5.0  0.0  5.0
```

Fig. 9.16 – Input data for undrained analysis (normally consolidated – Cam-clay)

Small load steps of 0.2 kPa are chosen for the first 5 increments. As soon as any load is applied, plastic yielding takes place. Then the loads are increased at the rate of 1 kPa per increment up to 16 kPa. A further 10 increments of 0.5 kPa are finally applied. The central settlement is plotted against the vertical pressure in Fig. 9.17. Large settlements take place which increase fairly steadily as more load is applied. Fig. 9.18 shows the region approaching the critical state.

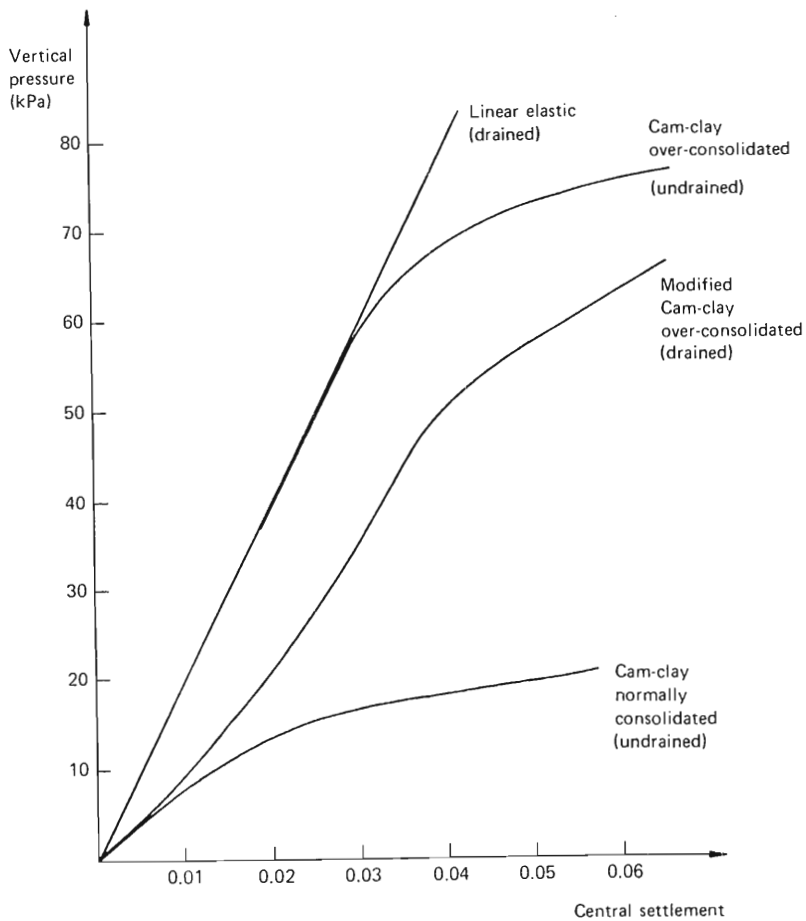


Fig. 9.17 – Comparison of pressure curves for different soil models

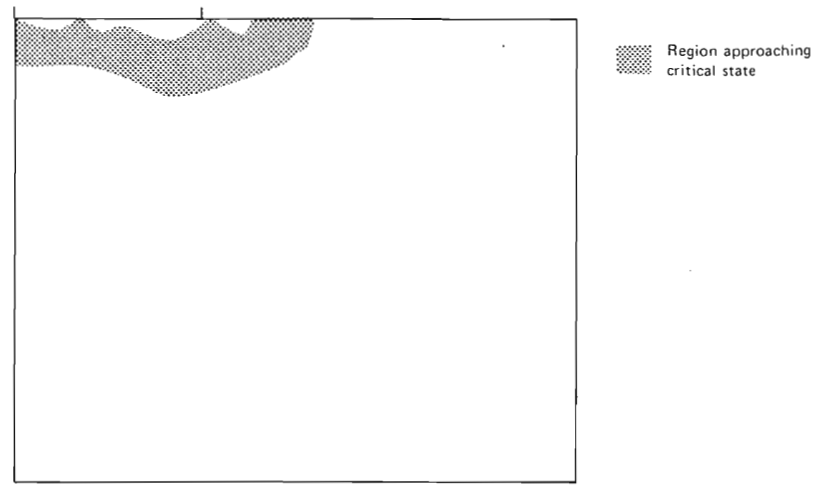


Fig. 9.18 – Region approaching critical state for normally consolidation soil

9.5.2 Undrained analysis – over-consolidated clay

The analysis conducted above is perhaps a little unrealistic. The strength is zero at the surface and increases linearly with depth. According to the theoretical analysis of Davis and Booker (1973) a rigid perfectly-plastic solid with this strength distribution will support only very small loads. From a practical point of view one would not expect to be able to put much load on such an extremely soft deposit. In real situations, attempts will be made to either lower the water table or pre-consolidate by applying dead loads. It is perhaps more realistic to consider an over-consolidated clay. The initial stress state, where the clay layer has been subjected to a vertical pressure of 50 kPa which was subsequently removed, is considered. The OCR at a depth of h metres is then given by

$$\text{OCR} = \frac{50 + \gamma' h}{\gamma' h} . \tag{9.4}$$

The OCR at a depth of 10 m is then $= (50 + 10 \times 10)/100 = 1.5$. A number of empirical relationships have been proposed for the relationship between K_0 and OCR (Wroth, 1975; Parry, 1982). The procedure by Wroth (1975) was discussed in section 5.5.3 and his equation for lightly over-consolidated clays is adopted here. The top 1 m is heavily over-consolidated and a linear variation is assumed for σ_h'. The distributions of σ_h' and σ_v' are shown in Fig. 9.19. The passive failure line is also indicated in that figure.

The calculation of the size of yield locus is as in the previous analysis but using the maximum stresses experienced. For example, considering the stress state at the base of the layer, the maximum vertical effective stress is 150 kPa. The horizontal effective stress is 92 kPa.

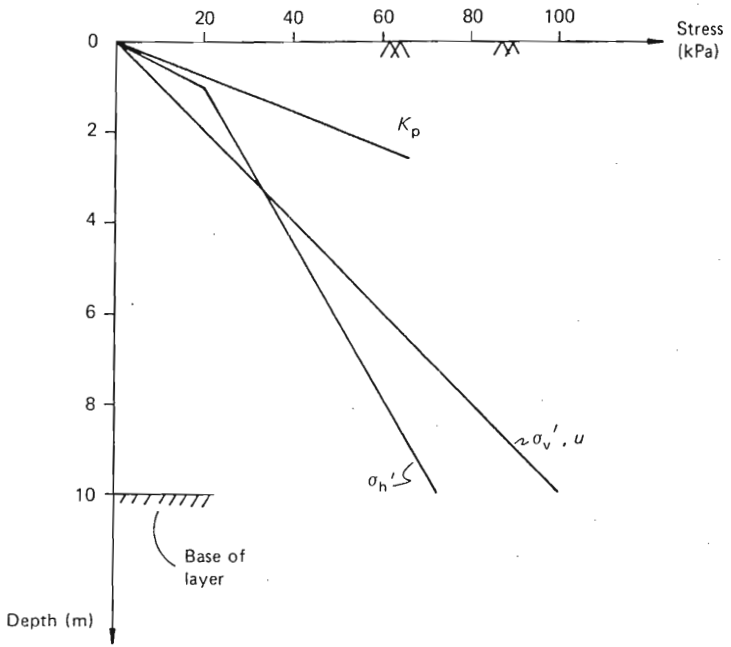


Fig. 9.19 – *In situ* stress distribution in analysis

$$q = 150 - 92 = 58,$$

$$p' = (150 + 2 \times 92)/3 = 111.3.$$

Using the Cam-clay yield locus, $p_c' = 200.25$. For the surface, $\sigma_{vm}' = 50$ kPa and $\sigma_{hm}' = 30.7$. This gives a value of 66.75 for p_c'. Because the horizontal effective stress distribution is approximated to be a bilinear curve, only three *in situ* nodes are needed to define it. The input data are shown in Fig. 9.20.

The initial response is elastic. Since the layer is over-consolidated and the test paths are undrained (i.e. vertical within the yield locus until yielding takes place), quite large load steps can be applied.

The initial yielding takes place around 40 kPa. In fact a single load increment of 40 kPa could have been applied. At the onset of yielding, small load steps are required. Increments of 1 kPa are sufficiently small enough under these circumstances. The softening again leads to a large settlement. The load–settlement curve is shown in Fig. 9.17. Even though the first yielding takes place around 40 kPa it is not until about 55 kPa that any deviation from the elastic response is noticeable. Beyond 65 kPa, significant yielding/softening takes place, which is accompanied by large settlements. Fig. 9.21 shows the yielding zones at different stages of the loading.


```
record
 A   ¦  CIRCULAR LOAD ON O.C. CAM-CLAY *** UNDRAINED
                    ............
                    ............
 L1  ¦  1  1  5  1  41  0  0  1
 L2  ¦  0  0  6  18  771  787  5  30
 M   ¦  1  4  0.062  0.161  1.759  0.888  0.25  0.  100.  20.  0.  0.
 O   ¦  1  3
 P1  ¦  1  0.  75.0  100.0  75.0  0.  100.  0.  201.0
 P1  ¦  2  9.  19.9  10.  19.9  0.  10.  0.  80.4
 P1  ¦  3 10.  0.  0.  0.  0.  0.  0.  66.75
                    ............
                    ............
 R   ¦  1  1  1  0  -2  0  0  222  0  0.0  0  0.0
 V   ¦  10  6  12  0.0  40.0  0.0  40.0  0.0  40.0
 V   ¦  20  12  18  0.0  40.0  0.0  40.0  0.0  40.0
 R   ¦  2  2  11  0  -2  0  0  0  1  0.0  0  0.0
 T2  ¦  112  12  12  12  102  12  102  12  102  222
 U   ¦  10  6  12  0.0  10.0  0.0  10.0  0.0  10.0
 U   ¦  20  12  18  0.0  10.0  0.0  10.0  0.0  10.0
 R   ¦  3  12  21  0  -2  0  0  0  1  0.0  0  0.0
 T2  ¦  112  12  12  12  102  12  102  12  102  222
 U   ¦  10  6  12  0.0  10.0  0.0  10.0  0.0  10.0
 U   ¦  20  12  18  0.0  10.0  0.0  10.0  0.0  10.0
 R   ¦  4  22  31  0  -2  0  0  0  1  0.0  0  0.0
 T2  ¦  112  12  12  12  102  12  102  12  102  222
 U   ¦  10  6  12  0.0  10.0  0.0  10.0  0.0  10.0
 U   ¦  20  12  18  0.0  10.0  0.0  10.0  0.0  10.0
 R   ¦  5  32  41  0  -2  0  0  0  1  0.0  0  0.0
 T2  ¦  112  12  12  12  102  12  102  12  102  222
 U   ¦  10  6  12  0.0  10.0  0.0  10.0  0.0  10.0
 U   ¦  20  12  18  0.0  10.0  0.0  10.0  0.0  10.0
```

Fig. 9.20 – Input data for undrained analysis (over-consolidated – Cam-clay)

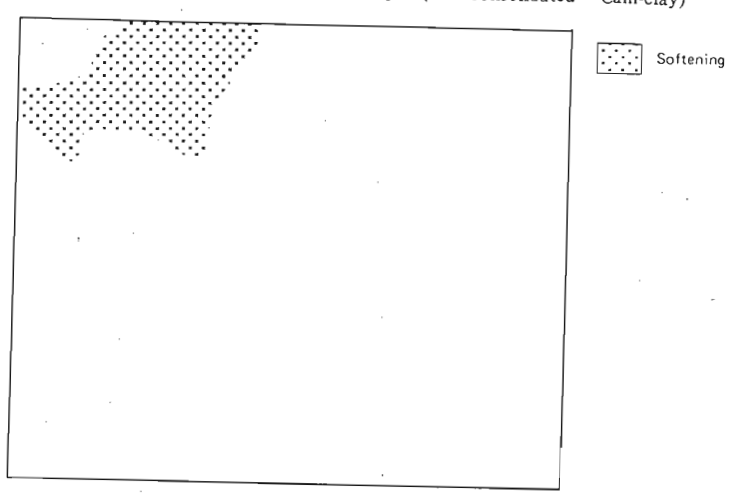


Fig. 9.21(a) – Zone of yielding after 61 increments (62 kPa) (Cam-clay, over-consolidated, undrained)

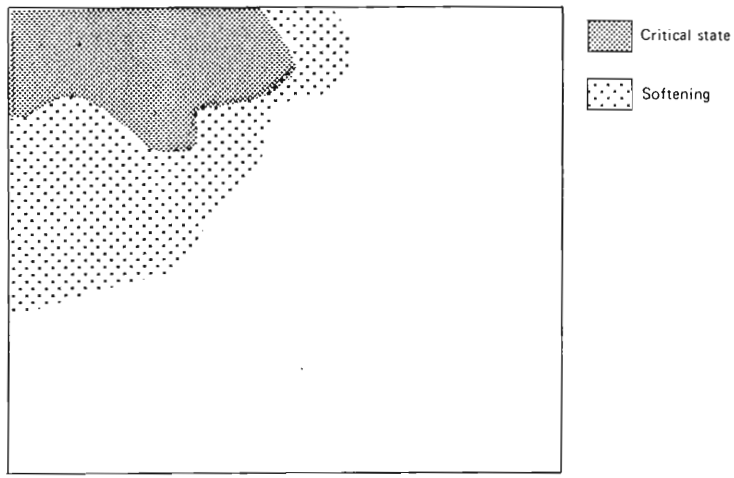


Fig. 9.21(b) – Zone approaching critical state after 80 increments (81 kPa vertical pressure) (Cam-clay, over-consolidated, undrained)

The stress paths for element centroids 16 and 19 are shown in Fig. 9.22. The effective stress paths are vertically upwards as predicted by theory. There is no change in p' until yielding.

9.6 DRAINED ANALYSIS – MODIFIED CAM-CLAY

In order to demonstrate a drained analysis, the modified Cam-clay (MCC) model is used. The stress history of over-consolidation is assumed to be the same as in the previous example. However, because of the difference in the yield locus the values of p_c' will be different.

The Cam-clay parameters are assumed to be the same as in the previous analysis except for the parameter Γ. For the Cam-clay model, Γ was taken as 2.759 and this gives a value of 2.858 for N. By assuming that both yield locii meet at the isotropic consolidation line the value of Γ is calculated as follows:

$$\Gamma = N - (\lambda - \kappa) \times \ln(2). \tag{9.5}$$

This yields a value of 2.789 for the MCC model.

A total load of 40 kPa was applied over the first 10 increments (Fig. 9.23). The initial response is elastic. The load–displacement curve is shown in Fig. 9.17. Even though the first yielding does not take place until about 40 kPa the response is curved upward. This is because of the increase in p' and since the effective bulk modulus is assumed to be

$$K' = \frac{(1+e)p'}{\kappa} \tag{9.6}$$

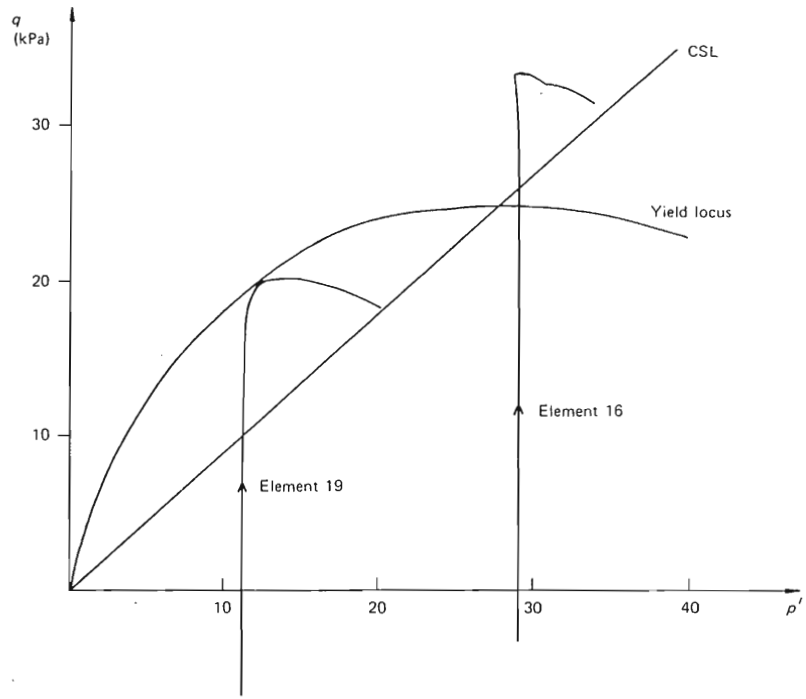


Fig. 9.22 – Effective stress paths (Cam-clay, undrained)

```
record
 A   ¦  CIRCULAR LOAD ON O.C. MOD CAM-CLAY *** DRAINED
                  ..........
                  ..........
 L1  ¦  1  1  1  1  10  0  0  1
 L2  ¦  0  0  6  18  771  787  5  30
 M   ¦  1  3  0.062  0.161  1.789  0.888  0.25  0.  0.  20.  0.  0.
 O   ¦  1  3
 P1  ¦  1  0.  75.0  100.0  75.0  0.  100.  0.  150.1
 P1  ¦  2  9.  19.9  10.  19.9  0.  10.  0.  60.0
 P1  ¦  3 10.  0.  0.  0.  0.  0.  0.  50.0
                  ..........
                  ..........
 R   ¦  1  1  10  0  -2  0  0  0  1  0.0  0  0.0
 T2  ¦  112  12  102  12  102  12  102  12  102  222
 U   ¦  10  6  12  0.0  40.0  0.0  40.0  0.0  40.0
 U   ¦  20  12  18  0.0  40.0  0.0  40.0  0.0  40.0
```

Fig. 9.23 – Input data for drained analysis (over-consolidated – MCC)

and there is a stiffening effect as the load builds up. Then a change of slope takes place around 50 kPa with increased settlements.

The zone of yielding is shown in Fig. 9.24.

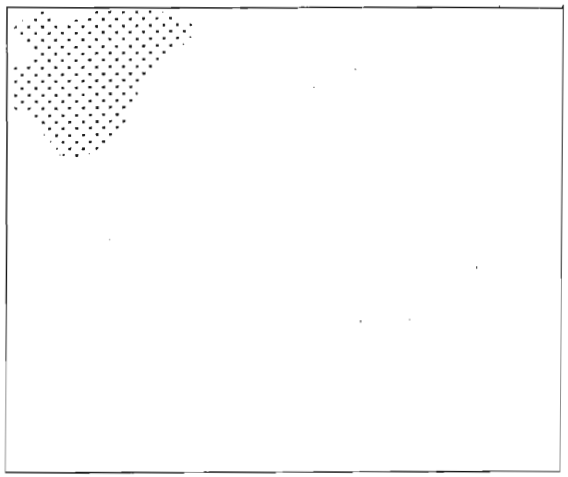


Fig. 9.24 – Zone of yielding/hardening (modified Cam-clay, drained analysis)

9.7 EMBANKMENT CONSTRUCTION

This problem is to illustrate the feature in the program to add elements. The details of the mesh, boundary conditions and properties of the elastic foundation are the same as in the drained analysis in section 9.4.1. The embankment is modelled by 16 LST elements of type 2. This is also a drained analysis. However, this is a plane strain analysis (NPLAX = 0 in record L1). The y axis is an axis of symmetry and the embankment is restrained in the x direction along this axis. The embankment is assigned a material zone number of 2. However, its material properties are the same as the elastic layer. The bulk unit weight of the embankment is taken as 20 kN/m^3. The *in situ* stresses are again set to zero. To indicate that body forces under earth's gravity are acting, DGRAV = 1 in record R. The input data are shown in Fig. 9.25.

```
record
 A   |  AN EMBANKMENT ON AN ELASTIC FOUNDATION
 B   |     63    96     3     2     2     8
 C   |      0     0
 D   |      0     0     0     0     0     0     0     0     0     0
 E   |      0     0     0     0
 F   |      1       0.000       0.000
 F   |      2       0.000       3.000
 F   |      3       0.000       6.000
                   ..............
                   ..............
 F   |     52      40.000       8.000
 F   |     53      40.000       9.000
 F   |     54      40.000      10.000
 F   |     55       0.000      11.000
```



```
 F   |     56       0.000      12.000
 F   |     57       2.000      11.000
 F   |     58       2.000      12.000
 F   |     59       4.000      11.000
 F   |     60       4.000      12.000
 F   |     61       6.000      11.000
 F   |     62       6.000      11.500
 F   |     63       8.000      11.000
 G1  |      0
 H   |      1     2     1     1     7     2
 H   |      2     2     1     2     7     8
 H   |      3     2     1     2     8     3
 H   |                ..............
                      ..............
 H   |     78     2     1    47    52    53
 H   |     79     2     1    47    53    48
 H   |     80     2     1    48    53    54
 H   |     81     2     2     6    12    57
 H   |     82     2     2     6    57    55
 H   |     83     2     2    12    18    59
 H   |     84     2     2    12    59    57
 H   |     85     2     2    18    24    61
 H   |     86     2     2    18    61    59
 H   |     87     2     2    24    30    63
 H   |     88     2     2    24    63    61
 H   |     89     2     2    30    36    63
 H   |     90     2     2    55    57    58
 H   |     91     2     2    55    58    56
 H   |     92     2     2    57    59    60
 H   |     93     2     2    57    60    58
 H   |     94     2     2    59    61    62
 H   |     95     2     2    59    62    60
 H   |     96     2     2    61    63    62
 K   |      0
 L1  |      0     2     1     1     1    16     0     1
 L2  |      0     0     6    18   771   787     5    30
 M   |  1  1  3.E3  3.E3  0.25  0.25  1.2E3  0.   0.  0.  0.  0.
 M   |  2  1  3.E3  3.E3  0.25  0.25  1.2E3  0.   0. 20.  0.  0.
 N   |    81   82   83   84   85   86   87   88   89   90
 N   |    91   92   93   94   95   96
 O   |    0  0
 R   |    1  1  1  16  0  0  28  2  0  0.0  0  1.0
 S   |    81   82   83   84   85   86   87   88   89   90
 S   |    91   92   93   94   95   96
 V   |     1    1    2    1    1   0.   0.   0.
 V   |     3    2    3    1    1   0.   0.   0.
 V   |     5    3    4    1    1   0.   0.   0.
                 ..............
                 ..............
 V   |    76   51   52    1    1   0.   0.   0.
 V   |    78   52   53    1    1   0.   0.   0.
 V   |    80   53   54    1    1   0.   0.   0.
 V   |    82    6   55    1    1   0.   0.   0.
 V   |    91   55   56    1    1   0.   0.   0.
```

Fig. 9.25 – Input data for embankment construction analysis

Elements 81 to 96 which represent the embankment (Fig. 9.26) are removed at the beginning (record N), and IPRIM = 16 in record L1, the number of elements being removed. These elements are added in the first increment block. Fig. 9.27 shows the surface settlement of the elastic layer.

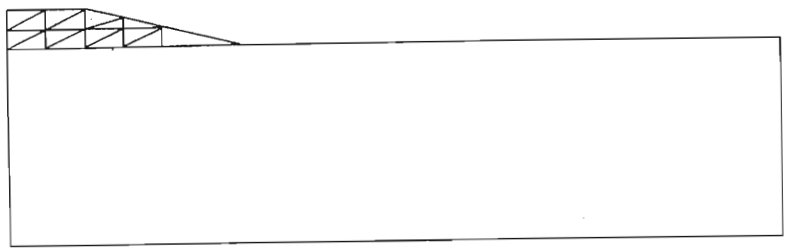


Fig. 9.26 – Elements used in modelling embankment

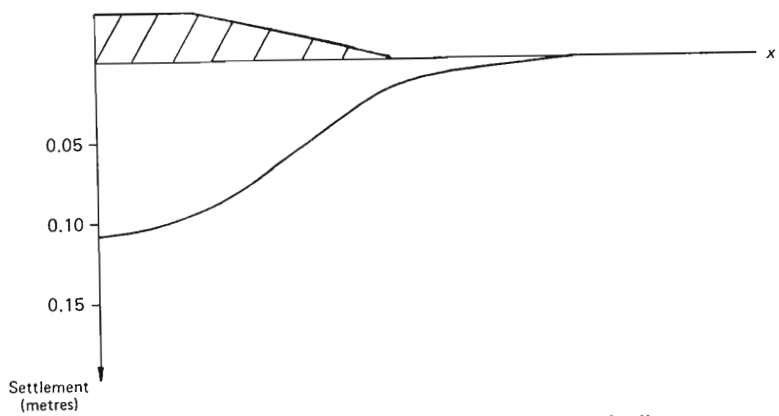


Fig. 9.27 – Surface settlement profile under embankment loading

If this was an analysis with a non-linear model the loading due to the weight of embankment in general would be applied in a number of increments. For example, if it is taken as 10 increments, only the following records need to be changed. record L1 to indicate there are 10 increments.

Record
L1 | 0 2 1 1 10 16 0 0

Record
R | 1 1 10 16 0 0 28 2 0 0. 0 1.

Then the loading will be applied in 10 equal increments.

9.8 EXCAVATION

As in the case of the embankment construction in section 9.7 this example is to illustrate the feature to remove elements. The details of the mesh and elastic properties are the same as in that example. This analysis is an axisymmetric drained analysis.

The simulation of an excavation process is carried out by removing the following elements (see Fig. 9.28).

5 6 7 8 9 10 15 16 17 18 19 20

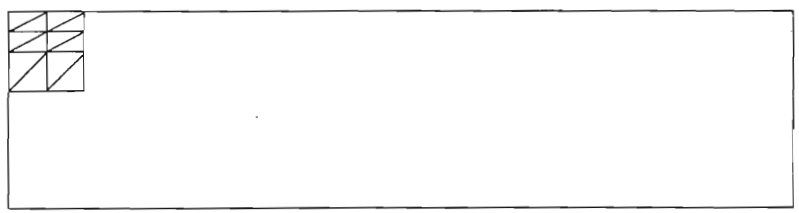


Fig. 9.28 – Elements removed in the simulation of an excavation event

The *in situ* stresses have to be specified for this example. K_0 is taken as 0.61 to calculate the initial stresses. To indicate that the *in situ* stresses were generated under earth's gravity, TGRAVI = 1 (record Q1). The input data are shown in Fig. 9.29.

```
record
 A   ¦  EXCAVATION IN AN ELASTIC FOUNDATION
 B   ¦    54   80    3    2    2    0
                 ..............
                 ..............
 K   ¦      0
 L1  ¦      1    1    1    1    1    0    0    1
 L2  ¦      0    0    6   18  771  787    5   30
 M   ¦      1  1  3.E3  3.E3  0.25  0.25  1.2E3  0.  0.  20.  0.  0.
 O   ¦      1  2
 P1  ¦    1  0.  61.  100.  61.  0.  100.  0.  0.
 P1  ¦    2  10.  0.    0.  0.  0.    0.  0.  0.
 Q1  ¦    0   26  1.
 Q3  ¦     1    1   2   1   1  0.  0.  0.
 Q3  ¦     3    2   3   1   1  0.  0.  0.
 Q3  ¦     5    3   4   1   1  0.  0.  0.
                 ............
                 ............
 Q3  ¦    76   51  52   1   1  0.  0.  0.
 Q3  ¦    78   52  53   1   1  0.  0.  0.
 Q3  ¦    80   53  54   1   1  0.  0.  0.
 V   ¦    80   53   54    1    1   0.  0.  0.
 R   ¦      1  1  1  12  0  0  0  2  0  0.0  0  0.0
 S   ¦    5  6  7  8  9  10  15  16  17  18  19  20
```

Fig. 9.29 – Input data for excavation analysis

The elements are removed in the first increment block in a single increment (ICHEL = 12 in record R). The list of elements removed is specified in record S. Earth's gravity is already acting at the *in situ* stage and hence DGRAV = 0. As explained in section 9.7 the loads due to the excavation could have been applied over a number of increments. The displacements around the excavation are shown in Fig. 9.30.

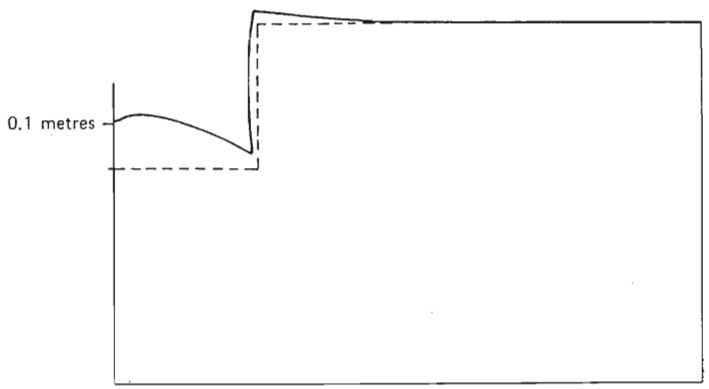


Fig. 9.30 – Displacements around an excavation

9.9 UNDRAINED TRIAXIAL TEST

Fig. 9.31 shows the mesh, which consists of 2 CuSt elements (type 6). Because stress conditions are uniform, arbitary dimensions are assumed. The use of two LST elements (element type 2) would probably be adequate for this analysis, but CuST elements are used (element type 6) to demonstrate this higher-order element (which is often to be preferred for axisymmetric analysis (see section 9.2)). There are four vertex nodes in the mesh.

The soil sample is isotropically consolidated to 200 kPa and then isotropically unloaded to a mean normal stress of 150 kPa. A standard undrained compression test is then carried out. The Cam-clay parameters selected for the soil are as follows:

$$\lambda = 0.30, \quad \kappa = 0.05, \quad M = 1.0, \quad \Gamma - 1 = 2.953, \quad \nu' = 0.3.$$

As this is an undrained analysis, the bulk modulus of water is required to complete the data on material properties.

$$\text{Effective bulk modulus of soil} = \frac{(1+e)p'}{\kappa}.$$

$$K' = \frac{(1+1.5517)150}{0.05}$$

$$= 7655 \text{ kPa}.$$

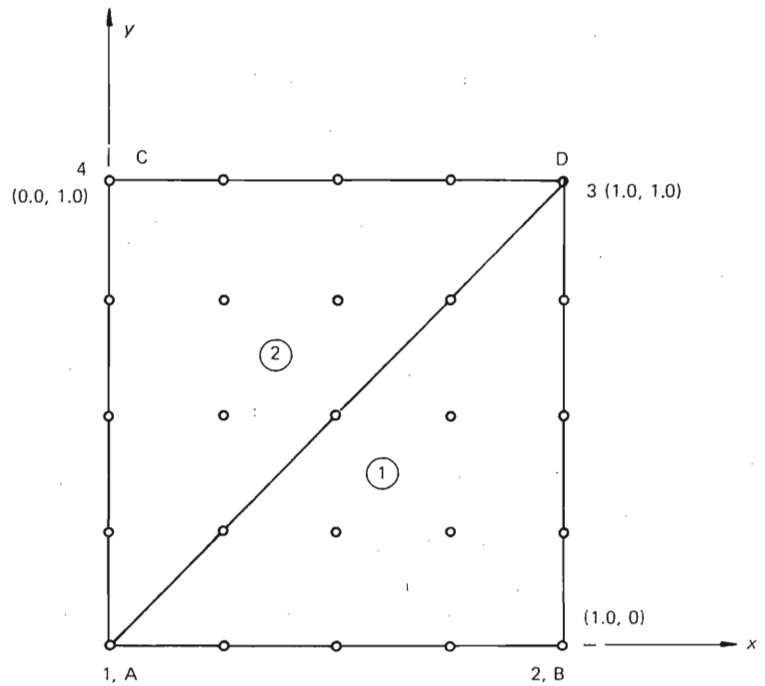


Fig. 9.31 – Undrained triaxial test on over-consolidated clay

In general, the bulk modulus of water is taken as 100 K' (7.655×10^5 kPa). In the analysis presented here, α is taken as 65, which gives a value of 5.0×10^5 kPa for the bulk modulus of water. More than 50 load increments are recommended for a finite element analysis which simulates a triaxial test. The purpose of the example presented here is to demonstrate the capabilities of the program. Therefore only six load increments are used for illustrative purposes. A strain-controlled test is considered here with 0.5% axial strain in each increment, leading to a total axial strain of 3%.

AB is restricted to move horizontally and AC is restricted to move vertically. CD is displaced vertically downwards to simulate a strain-controlled test. Different output options are specified in each increment.

The results of the analysis are plotted in $q{:}\epsilon_a$, $u{:}\epsilon_a$ and $q{:}p'$ space (Fig. 9.32). These are compared with the theoretical solution and also with an analysis using a large number of increments. These differences between theory and predictions are due to the large increment size which was used solely for illustrative purposes. During the second increment the yield ratio (YR) is on average about 1.12, i.e. the yield locus has grown in size by 12%. Stricter control on YR is recommended so that the change in size of the yield locus is not more than 1% (i.e. YR 1.01 or 0.99). These results emphasise the importance in selecting the

size of load increment: as shown here, large load increments may lead to erroneous results. The input data to the program are given in Fig. 9.33.

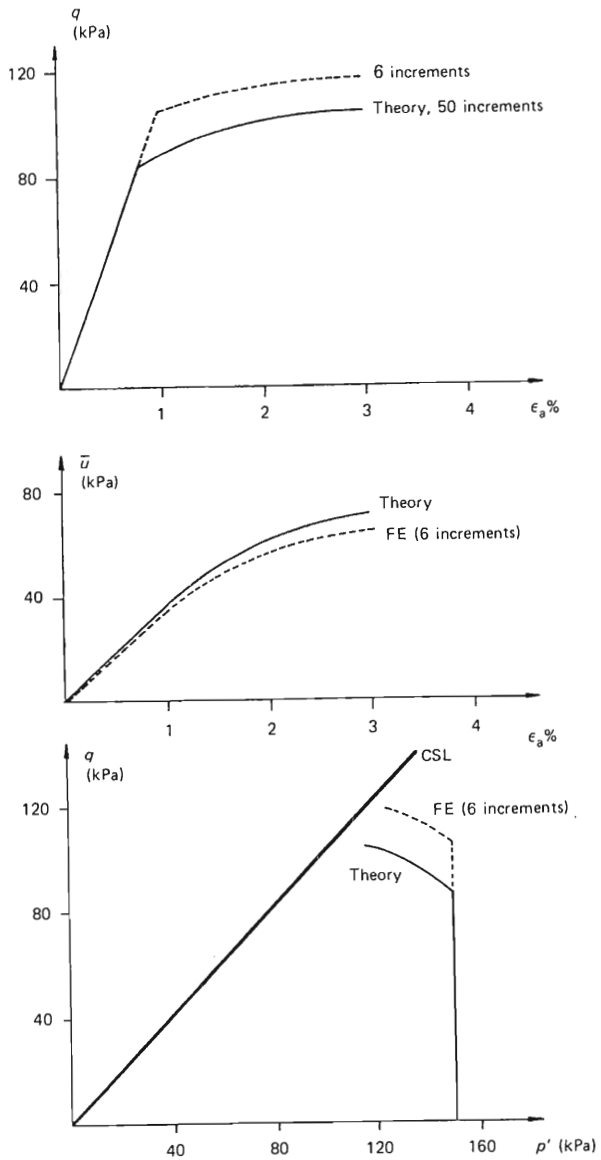


Fig. 9.32 – Comparison of theoretical and finite element results

9.10 INTERPRETATION OF ANALYSES USING CAM-CLAY

To make the interpretation of the results of analyses easier, the stress state of each integration point is assigned a number by CRISP. This indicates whether yielding is taking place and if so, whether the soil is hardening or softening. The different numerical codes are illustrated in Fig. 9.33(a).

Also of interest to the analyst is the amount by which the yield locus is expanding or contracting when yielding is taking place. This information is given by the yield ratio (YR) parameter which appears in the printed output:

$$\mathrm{YR} = \frac{p'_y}{p'_{co}}$$

where p'_{co} is the pre-consolidation pressure at the start of the load increment and p'_y is the pre-consolidation pressure at the end of the load increment (assuming that the sample has yielded).

If the soil is yielding and hardening then values of YR slightly greater than one will be seen. Values of YR less than one mean that either the soil is behaving elastically or the soil is yielding and softening. Fig. 9.33(b) illustrates some of the different possibilities where the initial yield locus always corresponds to p'_{co}. Soil in initial states B1 and C1 yields (and hardens) to points B2 and C2. Although C1 is initially elastic and B1 is already yielding, both stress changes lead to the same value of YR ($= p'_{yB}/p'_{co}$). Soil which remains elastic (e.g. A1 to A2 in Fig. 9.33(b)) has a YR which is calculated by constructing a fictitious yield locus through the current stress point to give the p'_y value ($\mathrm{YR} = p'_{yA}/p'_{co}$).

Examples of the use of the critical state models can be found in a number of publications. For example Mair *et al.* (1981) and Seneviratne & Gunn (1985) compare finite element predictions with tests on model tunnels. Bassett *et al.* (1981) and Almeida *et al.* (1986)) compare data from centrifuged model embankments with finite element analyses. These latter analyses clearly demonstrate how the strengthening of a clay foundation can be explained by a finite element model which combines a critical state soil model with consolidation following the stages of construction. In analyses such as these the basic mode of behaviour of different points in the soil mass can first be traced by examining values of η/M, YR and the numerical codes as the analysis progresses. Stress paths (i.e. p', q plots) of interesting points are then drawn. At present this is done by hand but (we hope) it will soon be an option in post-processing programs.

```
record
 A   |  * UNDRAINED TRIAXIAL TEST * MODIFIED CAM-CLAY *
 B   |      4    2    3    6    2    8
 C   |      0    0
 D   |      0    0    0    0    0    0    0    0    0    0
 E   |      0    0    0    0
 F   |      1         0.         0.
 F   |      2         1.         0.
 F   |      3         1.         1.
 F   |      4         0.         1.
 G1  |      0
 H   |      1    6    1    1    2    3
 H   |      2    6    1    1    3    4
 K   |      0
 L1  |      1    1    1    1    6    0    0    0
 L2  |      1    1    1    4   751  768  1  2
 M   |    1  3   0.05  0.3  2.9535  1.0  0.3   0   65.   0   0   0
 O   |      1    2
 P1  |     1   0.   150.   150.   150.   0.   0.   0.   200.
 P1  |     2   1.   150.   150.   150.   0.   0.   0.   200.
 Q1  |     2   2   0.
 Q2  |    1  2  3  0.  150.  0.  150.  0.  150.  0.  150.  0.  150.
 Q2  |    2  4  3  0.  150.  0.  150.  0.  150.  0.  150.  0.  150.
 Q3  |    1  1  2  2  1  0.  0.  0.  0.  0.
 Q3  |    2  1  4  1  1  0.  0.  0.  0.  0.
 R   |    1  1  6  0  0  0  1  0  1  0.  0  0.
 T2  |   1221 1211  121 1112  121 1222
 U   |    2  4  3  2  1  -0.03  -0.03  -0.03  -0.03  -0.03
```

Fig. 9.33 – Input data for undrained triaxial test – MCC

Appendix A: Input specification

A.1 DATA FORMAT

The data for the program are free format and the particular data items must appear in the correct order on a data record. The term 'record' describes the data which would be typed in via a computer terminal and occupies one line of a computer disk file (the normal method of preparation) or punched card. Unless specified otherwise it should be assumed that one record of data is represented by a single line of input or a punched card. The data items are separated by one or more spaces. For the sake of clarity, users should use at least two spaces. In fact, the data specified in a 'record', for example record D, could be spread over any number of lines, the only restriction being that they should be in the correct order.

For example, let the input for data record D be

0 0 1 0 1 0 1 1 0 0

This could have been input by

```
        .......
line 1  0  0  1  0
line 2  1  0  1
line 3  1  1  0  0
```

However, this usage is recommended only when it is absolutely necessary, because checking the input data for errors is made easier if each record of data is confined to a single line of input.

Data items are indicated below by mnemonic names, i.e. names which suggest the data item required by the program. The FORTRAN naming convention is used: names beginning with the letters I, J, K, L, M and N show that the program is expecting an INTEGER data item, whereas names beginning with any other letter show that the program is expecting a REAL data item. INTEGER data items must not contain a decimal point, but REAL data items may optionally do so. REAL data items may be entered in the FORTRAN exponent format if desired, i.e. 0.0011 may be entered as 1.1E-3. Individual data items must not contain spaces.

A.2 INPUT DATA

In this section, † indicates extra explanation in section 9.2.

† **Record A** (one only)

TITLE (up to 80 characters)

TITLE – title for the analysis

† **Record B** (one only)

NVTX	NEL	MXNDV	MXTYP	NDIM	IPLOT

NVTX – number of vertex nodes in the mesh
NEL – number of elements in the mesh
MXNDV – maximum number of vertex nodes in any element
MXTYP – element type with most number of total nodes (per element) in mesh
- 2 – linear strain triangle (LST) with displacements unknown
- 3 – linear strain triangle (LST) with displacements and excess pore pressures unknown (linear variation in pore pressure)
- 6 – cubic strain triangle (CuST) with displacements unknown
- 7 – cubic strain triangle (CuST) with displacements and excess pore pressures unknown (cubic variation in pore pressure)

NDIM – number of dimensions to problem
- 2 – two-dimensional problem

IPLOT – plotting option parameter with the following possible values:
- 0 – no plotting
- 1 – unnumbered mesh
- 2 – mesh and vertex node numbers
- 3 – mesh and midside node numbers

- 4 – mesh and all node numbers
- 5 – mesh and element numbers
- 6 – mesh, vertex numbers and element numbers
- 7 – mesh, midside numbers and element numbers
- 8 – mesh and all numbers

† **Record C** (one only)

NUMAX	MUMAX

NUMAX‡ – maximum value of user vertex node number
MUMAX‡ – maximum value of user element number

† **Record D** (one only)

ID1	ID2	ID3	ID10

ID1 ... ID10 – debugging option. To print out various arrays in geometry part of program, when testing the program.
- 0 – no printout
- 1 – list of arrays printed are given below: if set to 1, the following are printed:

ID1 – print NCONN after exit from routine CONECT which reads input data of list of nodes connected to each element (routine MARKZ)
ID2 – print ITAB after co-ordinates of all displacement nodes along element sides have been calculated (routine MIDSID)
ID3 – print IFR after all variables have been allocated places in FRONT (routine SFWZ)
ID4 – print NDEST after all variables have been allocated places in FRONT (routine SFWZ)
ID5 – print NCONN, MREL, NRELVV after all vertex node co-ordinates and element–nodal connectivity have been read (routine CONECT)
ID6 – print MFRN (optimum frontal order of elements specified by the user); only relevant if IRNFR = 1 (routine CONECT)
ID7 – print NCONN, MREL, MRELVV, NREL, NRELVV, LTYP, MAT, NQ after all nodes have been numbered and co-ordinates calculated (routine MARKZ)

‡ Use of 0 is only applicable if user node and element numbers begin with 1 and there are no gaps in the numbering.

ID8 – print contents of array G (both INTEGER and REAL parts are printed separately – routine MAST)
ID9 – print out INTEGER arrays which have been shifted (routine SHFTIB)
ID10 – print NQ and NW (routine GPOUT)

in order to avoid lots of output, all the values must be set to zero for normal runs (this option is used only when debugging the program)

† **Record E** (one only)

NSDZ	NSPZ	NDCUR	NPCUR

These four parameters are only relevant if the element sides are curved and the user intends to specify the co-ordinates of nodes along these sides; otherwise (default option) all four variables must be set to 0.

NSDZ – Number of nodes along element Sides excluding end nodes (Displacement nodes)
NSPZ – Number of nodes along element Sides excluding end nodes (excess Pore pressure nodes)
NDCUR – Number of CURved sides (Displacement nodes)
NPCUR – Number of CURved sides (Pore pressure nodes)

† **Record F** (NVTX records)

NODE	X	Y

NODE – vertex node number
X – x co-ordinate of node
Y – y co-ordinate of node

† **Record G1** (one only)

IRNFR

IRNFR – option to specify separate list of optimum frontal numbering of elements
1 – read separate list (see record G2)
2 – use the sequence in which elements are read (see record H)

† **Record G2** – only included if IRNFR = 1 in record G1

MFRU(1)	MFRU(2)	.....	MFRU(NEL)

MFRU(1) ... MFRU(NEL) – optimum frontal numbering of elements

† **Record H** (NEL records)

KEL	ITYP	IMAT	N1	N2	N3

KEL – element number
ITYP – element type number
2 – 6-noded LST (2-D)
3 – 6-noded LST (2-D consolidation)
6 – 15-noded CuST (2-D)
7 – 22-noded CuST (2-D consolidation)
IMAT – material zone number in the range 1 to 10
N1, N2, N3 – vertex node numbers listed in anti-clockwise order

† **Record I** (NDCUR records – only included if NDCUR > 0)

MU	ND1	ND2	X1	Y1	X2	Y2	.....	XN	YN

MU – element number
ND1,ND2 – nodes at either end of element side
X1,Y1
X2,Y2
.....
.....
XN,YN
– nodal co-ordinates of curved element side for displacement nodes – NSDZ (excluding end nodes)

† **Record J** (NPCUR records – only included if NPCUR > 0) (for consolidation elements only)‡

MU	ND1	ND2	X1	Y1	X2	Y2	.....	XN	YN

MU – element number
ND1,ND2 – nodes at either end of element side
X1,Y1
X2,Y2
.....
.....
XN,YN
– nodal co-ordinates of curved element side for pore pressure nodes – NSPZ (excluding end nodes)

‡ Not required for element type 3.

† **Record K** (one only)

IDCHK

IDCHK – option to stop analysis at different stages
- 0 – run complete analysis
- 1 – run geometry part of the program (enables the mesh to be plotted and checked)
- 2 – run geometry part of the program, read *in situ* stresses and boundary conditions and carry out an equilibrium check

† **Record L1** (one only)

NPLAX	NMAT	NOIB	INCS	INCF	IPRIM	IUPD	ISR

NPLAX – plane strain/axisymmetric analysis option
- 0 – plane strain
- 1 – axisymmetric

NMAT – number of material zones

NOIB – total number of increment blocks

INCS – increment number at start of analysis

INCF – increment number at finish of analysis

IPRIM – number of elements to be removed to form **primary** mesh

IUPD – geometry updating option
- 0 – co-ordinates are not updated after each increment
- 1 – co-ordinates are updated after each increment

ISR – stop–restart option
- 0 – stop–restart facility is not used
- 1 – limited stop–restart option (analysis can only be restarted and continued from where the previous run was stopped). Conveniently used with a disk file
- 2 – full stop–restart option making use of two magnetic tapes. Analysis can be restarted from any increment in the past

† **Record L2** (one only)

IBC	IRAC	NVOS	NVOF	NMOS	NMOF	NELOS	NELOF

IBC – boundary conditons output option
- 0 – boundary conditions are not printed
- 1 – boundary conditions are printed

IRAC – reactions output option
- 0 – reactions are not printed
- 1 – reactions are printed

NVOS – starting vertex node number for output‡

NVOF – finishing vertex node number of output‡

NMOS – starting midside node number for output‡

NMOF – finishing midside node number for output‡

NELOS – starting element number for output‡

NELOF – finishing element number for output‡

† **Record M** (NMAT records)

MAT	NTY	P(1)	P(2)	.....	P(10)

MAT – material **zone** number – all elements given the same number in record H will have the following properties (maximum of 10 different zones)

NTY – material property **type** as in the table below:
- 1 – elastic, anisotropic
- 2 – elastic, linear variation with depth
- 3 – modified Cam-clay
- 4 – Cam-clay

NTY / Property	1	2	3	4	5
P(1)	E_h	E_0	κ	κ	
P(2)	E_v	y_0	λ	λ	
P(3)	ν_{hh}	m	$\Gamma - 1$	$\Gamma - 1$	
P(4)	ν_{vh}	ν	M	M	
P(5)	G_{hv}	0	G or ν'	G or ν'	
P(6)	0	0	0	0	
P(7)	← 0 for drained, α for undrained, γ_w for consolidation →				
P(8)	← γ →				
P(9)	← k_x for consolidation, 0 for drained or undrained →				
P(10)	← k_y for consolidation, 0 for drained or undrained →				

‡ This allows one to reduce the output and print out the results for nodes and elements which are within a specified range. This option is applied on the output codes specified in record R.

† **Record N** – only included if IPRIM > 0

JEL(1)	JEL(2)	.....	JEL(IPRIM)

JEL(1) etc. – list of element numbers to be removed to form mesh at start of analysis

† **Record O** (one only) – records O to Q3 are omitted for a restarted analysis using the stop–restart facility

KT	NI

KT – *in situ* stress option
- 0 – set *in situ* stresses to zero
- 1 – interpolate *in situ* stresses from a given set of nodes representing layers
- 2 – direct specification of *in situ* stresses at all integration points

NI – the number of *in situ* nodes‡ (giving NI-1 *in situ* layers)

† **Record P1** (NI records) – only included if KT = 1

IL	YI	V(1)	V(2)	...	V(7)

IL – *in situ* node number§
YI – y co-ordinate of *in situ* node
V(1) – σ'_x
V(2) – σ'_y
V(3) – σ'_z
V(4) – τ_{xy}
V(5) – u
V(6) – 0
V(7) – p'_c (zero, if not Cam-clay)

Record P2 (only included if KT = 2)

There are NEL sets of records P2 and P3 – one set for each element

MUS

MUS – element number

‡ These *in situ* nodes are not to be confused with the nodes in the finite element mesh. These *in situ* nodes serve as reference points for interpolating *in situ* stresses.

§ Records P1 must be input in ascending order of *in situ* node numbers. No gaps are allowed in the *in situ* node numbers.

Record P3 (NGP records – only included if KT = 2)‡

VAR(1)	VAR(2)	.....	VAR(7)

VAR(1) ... VAR(7) – stress parameters at each integration point (σ'_x, σ'_y, σ'_z, τ_{xy}, u, e, p'_c) where
- e – voids ratio
- p'_c – pre-consolidation pressure

for all models other than the Cam-clays, e and p'_c must be set to zero

Record Q1 § (one only – omit if *in situ* stresses are set to zero, i.e. KT = 0, in record O)

NLODI	NFXI	TGRAVI

NLODI – number of element sides with pressure loading (which is in equilibrium with the *in situ* stresses)
NFXI – number of element sides with fixities in the mesh
TGRAVI – *in situ* gravity acceleration field – 0, 1 or n

Record Q2 § (NLODI records – only included if NLODI > 0)

L	N1	N2	T1	S1	T3	S3	T2	S2

– linear strain triangle

see record U(b) for details

L	N1	N2	T1	S1	T3	S3	T4	S4	T5	S5	T2	S2

– cubic strain triangle

see record U (b) for details

‡ NGP – the number of integration points and one line of data for each integration point.
= 7 (for LST element types 2, 3)
= 16 (for CuST element types 6, 7)

§ Records Q1, Q2 and Q3 are omitted if *in situ* stresses are all set to zero (i.e. KT = 0 in record O).

Record Q3 ‡ (NFXI records – only included if NFXI > 0)

ML	ND1	ND2	IVAR	IFX	0	0	0

– linear strain triangle

ML	ND1	ND2	IVAR	IFX	0	0	0	0	0

– cubic strain triangle

ML – element number

ND1, ND2 – node numbers at the end of element side which is fixed

IVAR – the variable that is fixed
- 1 – x displacement
- 2 – y displacement

IFX – fixity code = 1

Record R (one only, but the group of records R to V is repeated for each increment block, i.e. NOIB times)

IBNO	INCA	INCB	ICHEL	NLOD	ILDF	NFIX	IOUT	IOCD
DTIME	ITMF	DGRAV						

IBNO – increment block number

INCA – increment number at the start of the current increment block (INCA ⩾ INCS)

INCB – increment number at the end of the current increment block (INCB ⩽ INCF)

ICHEL – number of elements to be added/removed for the current increment block

NLOD – number of incremental nodal loads or (if NLOD is negative) the number of element sides with pressure loading

ILDF – load ratios
- 0 – the loading is equally distributed over the INCB–INCA+1 increments
- 1 – read separate list of load ratios for each increment (record T1)

NFIX – number of element sides with prescribed value of the variable

IOUT – standard output for this increment block – a four-digit number **abcd** where (see also record L2)

‡ Records Q1, Q2 and Q3 are omitted if *in situ* stresses are all set to zero (i.e. KT = 0 in record O).

a – out-of-balance loads
- 0 – no out-of-balance loads
- 1 – out-of-balance loads at vertex nodes
- 2 – out-of-balance loads at all nodes

b – extra parameters for Cam-clay models only
- 0 – no output
- 1 – parameters at element centroids‡
- 2 – parameters at all integration points

c – option for general stresses
- 0 – no stresses printed
- 1 – stresses at element centroids‡
- 2 – stresses at all integration points

d – option for nodal displacements
- 0 – no displacements printed
- 1 – displacements at vertex nodes
- 2 – displacements at all nodes

IOCD – output option
- 0 – standard output given by IOUT for each increment in the increment block
- 1 – read separate list of output options for each increment (record T2)

DTIME – time increment for consolidation analysis

ITMF – time increments
- 0 – time increment DTIME is equally divided between all the increments in the increment block
- 1 – read separate list of time steps for each increment (record T3)

DGRAV – increment in gravity acceleration field = (Δn – change in number of gravities)

note: the number of increments in the increment block NOINC (= INCB – INCA + 1) must not exceed 50.

Record S – only included if ICHEL > 0

JEL(1)	JEL(2)	...	JEL(ICHEL)

JEL(1) etc. – list of element numbers which are added/removed in this increment block

‡ Centroid is the last integration point in the element (it is the 7th in LST and the 16th in CuST).

Record T1 – only included if ILDF = 1

R(1)	R(2)	...	R(NOINC)

R(1) etc. – the ratio of incremental loads to be applied in each increment

Record T2 – only included if IOCD = 1

IOPT(1)	IOPT(2)	...	IOPT(NOINC)

IOPT(1) etc. – the output options for each increment

Record T3 – only included if ITMF = 1

DTM(1)	DTM(2)	...	DTM(NOINC)

DTM(1) etc. – the time steps for each increment (these are not ratios)

where NOINC = INCB – INCA + 1

†**Record U** (NLOD records)

(a) NLOD > 0

N	DFX	DFY

N – node number
DFX – increment of x force
DFY – increment of y force

(b) NLOD < 0

L	N1	N2	T1	S1	T3	S3	T2	S2

– linear strain triangle

L	N1	N2	T1	S1	T3	S3	T4	S4	T5	S5	T2	S2

– cubic strain triangle

L – element number
N1, N2 – node numbers at end of the loaded element side
T1 – increment of shear stress at N1

S1 – increment of normal stress at N1
T3,T4,T5 – increment of shear stress at edge nodes 3, 4 and 5 (see Fig. A.1)
S3,S4,S5 – increment of normal stress at edge nodes 3, 4 and 5
T2 – increment of shear stress at N2
S2 – increment of normal stress at N2

sign convention for stresses:

shear stresses which act in an anti-clockwise direction about element centroid are positive. Normal stresses – compressive stresses are positive

Record V (NFIX records)

ML	ND1	ND2	IVAR	IFX	V1	V3	V2			

(a)

ML	ND1	ND2	IVAR	IFX	V1	V3	V4	V5	V2

(b)

ML	ND1	ND2	IVAR	IFX	V1	V2	0

(c)

ML	ND1	ND2	IVAR	IFX	V1	V3	V4	V2	0

(d)

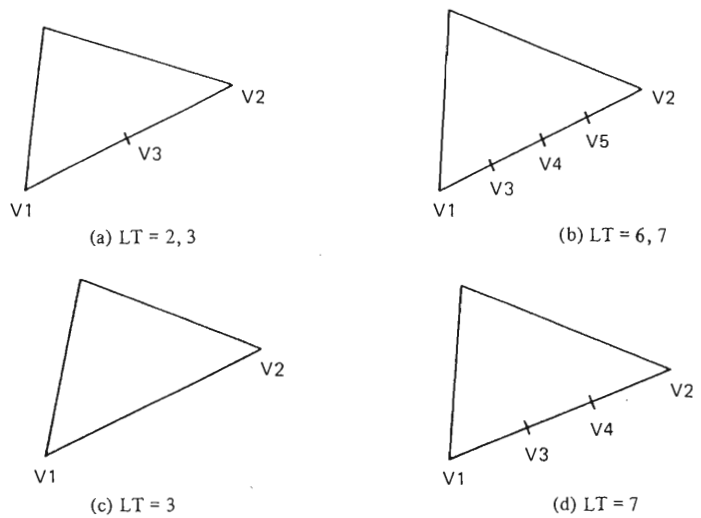
ML – element number
ND1,ND2 – node numbers at the end of fixed element side
IVAR – the variable that is prescribed
  1 – x displacement
  2 – y displacement
  3 – excess pore pressure
IFX – fixity code
  1 – incremental value of variable
  2 – absolute value of **excess pore pressure**
V1, V2 – prescribed value at end nodes
V3, V4, V5 – prescribed values at nodes along element side (excluding end nodes)

(a) displacement fixity
linear strain triangle – element types 2 and 3
IVAR = 1 or 2; IFX = 1

(b) displacement fixity
cubic strain triangle – element types 6 and 7
IVAR = 1 or 2; IFX = 1

(c) excess pore pressure fixity
linear strain triangle – element type 3
IVAR = 3; IFX = 1 or 2

(d) Excess pore pressure fixity
cubic strain triangle – element type 7
IVAR = 3; IFX = 1 or 2



(a) LT = 2, 3 (b) LT = 6, 7

(c) LT = 3 (d) LT = 7

Fig. A.1 – Displacement and pore pressure fixities

A.3 DATA SUMMARY

Record type	No. of records	Read in subroutine	Data
A	1	MAST	TITLE
B	1	MAST	NVTX NEL MXNDV NDIM IPLOT
C	1	MAST	NUMAX MUMAX
D	1	MARKZ	ID1 ID2 ID3 ..... ID10
E	1	MARKZ	NSDZ NSPZ NDCUR NPCUR
F	NVTX	RDCOD	NODE X Y
G1	1	CONECT	IRNFR
G2	–	CONECT	MFRU(1) MFRU(2) ..... MFRU(NEL)
H	NEL	CONECT	KEL ITYP IMAT N1 N2 N3
I	NDCUR	CUREDG	MU ND1 ND2 X1 Y1 ... XN YN
J	NPCUR	CUREDG	MU ND1 ND2 X1 Y1 ... XN YN

Record type	No. of records	Read in subroutine	Data
K	1	CPW	IDCHK
L1	1	RDPROP	NPLAX NMAT NOIB INCS INCF IPRIM IUPD ISR
L2	1	RDPROP	IBC IRAC NVOS NVOF NMOF NELOS NELOF
M	NMAT	RDPROP	MAT NTY P(1) P(2) ... P(10)
N	1	INSITU	JEL(1) JEL(2) ... JEL(IPRIM)
O	1	INSITU	KT NI
P1	NI	RDSTRS	IL YI V(1) V(2) ... V(7)
P2	NEL	RDSTRS	MUS
P3	NGP*NEL	RDSTRS	VAR(1) VAR(2) ..... VAR(7)
Q1	1	INSITU	NLODI NFXI TGRAVI
Q2	NLODI	INSITU	L N1 N2 T1 S1 T3 S3 T2 S2 or N1 N2 T1 S1 T3 S3 T4 S4 T5 S5 T2 S2
Q3	NFXI	FIXX	ML ND1 ND2 IVAR IFX 0 0 0 or ML ND1 ND2 IVAR IFX 0 0 0
R	1	ANS	IBNO INCA INCB ICHEL NLOD ILDF NFIX IOUT IOCD DTIME ITMF DGRAV
S	1	ANS	JEL(1) JEL(2) ... JEL(ICHEL)
T1	1	FACTOR	R(1) R(2) ... R(NOINC)†
T2	1	FACTOR	IOPT(1) IOPT(2) .. IOPT(NOINC)†
T3	1	FACTOR	DTM(1) DTM(2) ... DTIM(NOINC)†
U	NLOD	ANS	N DFX DFY or L N1 N2 T1 S1 T3 S3 T2 S2 or L N1 N2 T1 S1 T3 S3 T4 S4 T5 S5 T2 S2
V	NFIX	FIXX	ML ND1 ND2 IVAR IFX V1 V3 V2 or ML ND1 ND2 IVAR IFX V1 V3 V4 V5 V2 or ML ND1 ND2 IVAR IFX V1 V2 0 or ML ND1 ND2 IVAR IFX V1 V3 V4 V2 0

† NOINC = INCB – INCA + 1.

N.B. The group of records R to V is repeated NOIB times.

Appendix B: Mesh-plotting using GINO-F

B.1 INFORMATION WRITTEN TO PLOT DATA FILE

The data written to the Plot Data (PD) file on unit 8 by CRISP provide all the information necessary to draw the mesh and number the nodes and elements. The contents of the PD file are different for different values of IPLOT (record B of input data). Therefore if different types of plot are required – for example an unnumbered mesh or mesh with only element numbers – then it would require two separate runs of CRISP with appropriate values for the parameter IPLOT. The mesh-plotting program only interprets the information written to the PD file and has no control over the different types of plot as available in CRISP.

The information written to the PD file has the following format:

NDIM
CODMAX(ID), ID = 1,NDIM. CODMIN(ID), ID = 1,NDIM.

followed by a number of records of the form

INT1 XYZ(ID), ID = 1, NDIM. INT2

For two-dimensional problems, NDIM = 2.

CODMAX and CODMIN contain the maximum and minimum values of the nodal co-ordinates. These define the extent of the mesh and are used in calculating a default scale for the plot.

XYZ(NDIM) is either the nodal co-ordinates or the element centroid co-ordinates.

INT1 is a control code.
If negative, indicates change of pen colour.
−1 – Black. Used in drawing element sides and to plot vertex node numbers.
−2 – Red. Used to plot midside node numbers.
−3 – Green. Used to plot element numbers.
If INT1 = 11 then a number is to be plotted.
If INT1 = 3 indicates MOVE the pen to the position XYZ(NDIM).
If INT1 = 1 indicates DRAW to the position XYZ(NDIM).
If INT1 = 0 indicates end of PD file.

INT2 is a node or an element number or a dummy integer.

When INT1 is negative it indicates a change of pen colour. The rest of the record is then ignored.

When INT1 is either 1 or 3 the value INT2 is ignored, i.e. when drawing element sides.

When INT1 = 11 then the number given by INT2 is plotted at the position given by XYZ(NDIM).

Therefore two records are needed to draw an element side.

The first record will then contain INT1 = 3 with the co-ordinates of one end of the element side. The second record will contain INT1 = 1 with the co-ordinates of the other end of the element side.

The option to choose a scale and whether or not to rotate the plot in the plotter space is controlled by the input to the mesh-plotting program.

B.2 MESH-PLOTTING PROGRAM

A listing of the program which makes use of the GINO-F routines and plots the mesh is given below. This program was used in the Cambridge University computer, an IBM 3081. Slight changes may be necessary to run it in other installations.

It reads the PD file from unit IRP (set to 1), which was created by CRISP. The following file containing control data is read from unit IR5 (set to 5). The control data consist of

record 1 ID1 – the debug flag. If set to 1, prints out the data in the PD file. This option is only used when something goes wrong and no plot is produced. The normal option is to set to 0.

record 2 IROTPL – option to rotate plot. If set to 1, the plot is rotated through 90°. This option can be used to make better use of the plotting region, i.e. if the mesh has y dimension greater than x dimension. Otherwise set to 0.

The plotting region requested is 300 X 250 mm. This can easily be altered. The mesh itself is plotted within a region of 250 X 200 mm. This leaves a gap of 25 mm all around the mesh to allow for node numbers to be plotted.

Routine **INTPLT** initialises the plotter and deals with the positioning and rotation of the plot.

Routine **WIDTH** calculates the number of digits in the element or node number.

Note that the plotting options (vertex node numbers, midside node numbers and element numbers) have been specified using IPLOT in record B of input data to CRISP. If a different type of plot is required (e.g. an unnumbered mesh) then CRISP has to be re-run with the appropriate value for IPLOT in record B.

```
      CHARACTER*80 ITITLE
      DIMENSION CMIN(2),CMAX(2),CD(2)
C*********************************************************************
C     PROGRAM TO PLOT FINITE ELEMENT MESH FROM CRISP PROGRAM
C     USING GINO-F GRAPHICS
C     IRP   - FILE CONTAINING PLOT DATA
C     IR5   - FILE CONTAINING CONTROL DATA FOR PLOT
C     IW6   - OUTPUT FILE (TO PRINTER)
C*********************************************************************
      PIBY2=2.*ATAN(1.)
      PIBY6=PIBY2/3.
      THETA=(5./6.)*PIBY6
      IRP=1
      IR5=5
      IW6=6
      IW8=8
C
      READ(IRP)ITITLE
C
      READ(IRP)NDIM
      WRITE(IW6,909)NDIM
  909 FORMAT(//1X,6HNDIM =,I4/)
      READ(IRP)(CMAX(ID),ID=1,NDIM),(CMIN(ID),ID=1,NDIM)
      WRITE(IW6,912)(CMAX(ID),ID=1,NDIM),(CMIN(ID),ID=1,NDIM)
  912 FORMAT(/1X,15HPLOT DIMENSIONS/(6F8.1))
C
      XW=CMAX(1)-CMIN(1)
      YW=CMAX(2)-CMIN(2)
C
      WRITE(IW6,950)XW,YW
  950 FORMAT(/1X,4HXW =,F8.2,2X,4HYW =,F8.2)
C
      XLN=CMIN(1)
      YLN=CMIN(2)
C
C *** READ PLOT CONTROL PARAMETERS
```



```
C   RECORD 1 - ID1 (DEBUG OPTION : PRINT INFO READ FROM PLOT DATA FILE)
      READ(IR5,*)ID1
      WRITE(IW6,922)ID1
  922 FORMAT(/10X,15HDEBUG OPTION = ,I8/)
C     RECORD 2 - IROTPL
C *** IROTPL (IF EQ 1) - ROTATE PLOT BY 90 DEGREES (AC DIRECTION)
      READ(IR5,*)IROTPL
      WRITE(IW6,930)IROTPL
  930 FORMAT(//
     1 10X,46HROTATION OF PLOT..............................  =,I8/)
C *** SIZE OF REGION REQUESTED FOR PLOT IS 300 X 250 MM
C *** DEFAULT SIZE OF PLOT IS 250 X 200 MM
C *** (THIS GIVES A 25 MM SPACE ALL AROUND THE MESH)
      PLOTL=250.
      PLOTW=200.
C
      IF(IROTPL.NE.1)GOTO 35
      AW=XW
      XW=YW
      YW=AW
C
   35 SCALM=PLOTW/XW
      SCALY=PLOTL/YW
      IF(SCALY.LT.SCALM)SCALM=SCALY
C
      WRITE(IW6,940)SCALM
  940 FORMAT(
     1 10X,46HSCALE FOR PLOT.............................  =,F9.3/)
C
      CALL INTPLT(IW8,IROTPL,SCALM,XLN,YLN,PLOTL,PLOTW)
C
C *** READ PLOT INFO FROM FILE
   10 READ(IRP)ICODE,(CD(ID),ID=1,NDIM),N
      IF(ID1.EQ.0)GOTO 12
      WRITE(IW6,910)ICODE,(CD(ID),ID=1,NDIM),N
  910 FORMAT(1X,I5,2F8.1,I8)
   12 CONTINUE
      IF(ICODE.EQ.0)GOTO 99
      IF(ICODE.GT.0)GOTO 15
C *** IF ICODE IS NEGATIVE CHANGE PEN COLOUR
C *** SELECT PEN COLOUR
C *** IN GINO-F PEN COLOUR FOR GREEN IS 5 AND NOT 3 AS IN IBM 3081
      ICODE=IABS(ICODE)
      IF(ICODE.EQ.3)ICODE=5
      CALL PENSEL(ICODE,IDUM,IDUM)
      GOTO 10
C
   15 IF(ICODE.GT.10)GOTO 25
C *** DRAW ELEMENT SIDE
      IPEN=ICODE-1
      IF(IPEN.EQ.2)CALL MOVTO2(CD(1),CD(2))
      IF(IPEN.EQ.0)CALL LINTO2(CD(1),CD(2))
      GOTO 10
C *** PLOT NUMBER WITH OFFSET
C *** IF ICODE IS GT 10 - PLOT NUMBER
   25 CALL MOVTO2(CD(1),CD(2))
CC    CALL MOVBY2(XCS,YCS)
C *** PLOT NUMBER
      CALL WIDTH(N,NW)
```

```
      CALL CHAINT(N,NW)
      GOTO 10
C *** CLOSE STREAM, PACKAGE
   99 CALL DEVEND
      STOP
      END
      SUBROUTINE INTPLT(IW8,IROTPL,SCALM,XLN,YLN,PLOTL,PLOTW)
C**********************************************************************
C     INITIALISE PLOTTER
C**********************************************************************
C
C *** THE FOLLOWING STATEMENT IS FOR OTHER INSTALLATIONS,
C *** NOT REQUIRED IN IBM 3081
CC    CALL HP7220
C *** THE FOLLOWING STATEMENT IS FOR IBM 3081
      CALL GINPLT
      PLOTL=PLOTL+50.
      PLOTW=PLOTW+50.
      CALL DEVPAP(PLOTL,PLOTW,1)
C
      CALL MOVTO2(0.,0.)
      CALL UNITS(1.0)
C
      CHAHIG=2.5
      CHAWID=1.5
C
      IF(IROTPL.EQ.1)GOTO 20
      CALL MOVTO2(0.,10.)
      CALL CHASIZ(CHAWID,CHAHIG)
      CALL SCALE(SCALM)
      ZX=25./SCALM-XLN
      ZY=25./SCALM-YLN
      CALL SHIFT2(ZX,ZY)
      RETURN
   20 CONTINUE
      CALL CHAANG(90.)
      CALL MOVTO2(240.,10.)
      CALL CHAARR(ITITLE,20,4)
      CALL CHASIZ(CHAWID,CHAHIG)
      CALL SCALE(SCALM)
      ZX=210./SCALM+YLN
      ZY=-(XLN-25./SCALM)
      CALL SHIFT2(ZX,ZY)
      CALL ROTAT2(90.)
      CALL MOVTO2(XLN,YLN)
      RETURN
      END
      SUBROUTINE WIDTH(N,NW)
C
      IW=0
      NC=N
C
   10 NC=NC/10
      IW=IW+1
      IF(NC.GT.0)GOTO 10
      NW=IW
      RETURN
      END
```

Appendix C: Explanations of error and warning messages

ANS

(a) ERROR IN INCR BLOCK NUMBER NA NB

The increment block numbers must be in sequence. The program has an internal counter and it expects the increment block number to be NB, but in the data input it has the number NA (probable user error).

(b) ERROR IN INCREMENT NUMBER INC1 INC2 (ROUTINE ANS)

When reading the control parameters for the current increment block the firs. and last increments are read as INC1 and INC2. INC1 must be in sequence (if this is not the first increment block in the analysis, INC1 must have the value next to the last increment in the previous block) and if it is not equal to the counter within the program, the above message is printed. The above message will also appear when INC2 < INC1.

(c) When the number of increments in a block exceeds the allocated 50, the following message is printed:

INCREASE SIZE OF ARRAYS RINCC, DTM AND IOPT TO NH
ALSO SET INCZ IN ROUTINE ANS

INCZ must be set equal to the actual number of increments (NH in this example) in addition to the array sizes being increased in routine ANS.

CUREDG

(a) ***ERROR** EDGE CONTAINING NODES N1 N2 NOT FOUND

Each element side is given a unique code IHASH (= 10000 * I1 + I2, I1 and I2 being the (program) node numbers at either end with I1 < I2) when the program calculates the co-ordinates of the nodes along the side. This code is entered in the first column of array ITAB(LTAB,LDIM). When the user specifies the nodal co-ordinates along the **curved** element sides, as a first step the code for the element side is calculated and the above array is scanned to find it. If it is not found then the above message is printed (probable user error in specifying the nodes N1 and N2).

(b) ELEMENT M1 DOES NOT CONTAIN NODES N1 N2

When the user specifies the co-ordinates of nodes along curved element sides he/she identifies the element side by the element number and the nodes at either end of the side. When either or both these nodes (N1 and N2) are not found in element M1, this statement appears (probable user error).

(c) If errors of category (a) or (b) have been encountered, the program is stopped after the input data of nodal co-ordinates along curved sides have been read, with the following message.

PROGRAM TERMINATED IN ROUTINE CUREDG

DETJCB

JACOBIAN R1 IS NEGATIVE. ELEMENT M1 INT. POINT N (ROUTINE DETJCB)

When the determinant of the Jacobian matrix is negative this message is printed. The value of the determinant is R1. This is followed by a code to indicate at what stage of the analysis this error occurred.

DETMIN

JACOBIAN R1 OF ELEMENT M INTEGRATION POINT N IS NEGATIVE (ROUTINE DETMIN)

The determinant of the Jacobian matrix of element M at integration point N is negative. This is followed by a code to indicate at what stage of the analysis this error occurred. This is mainly for the benefit of the programmer when testing the program. This message with a code of 1 probably means an error in specifying the mesh geometry. Check whether the nodes associated with an element are specified in the anti-clockwise order. Also check whether the co-ordinates of the nodes have been specified correctly.

DISTLD

**** ERROR : ELEMENT M DOES NOT HAVE NODES : N1 N2 (ROUTINE DISTLD)

The nodes N1, N2 are used to identify the side of element M which has a pressure loading. This message should not be printed, since before entering this routine a check is carried out to find whether nodes N1 and N2 belong to element M. Therefore this message can only mean a program error.

EDGLD

(a) ELEMENT M1 NOT PRESENT IN CURRENT MESH (ROUTINE EDGLD)

The element with the pressure loaded side does not exist in the current mesh (probable user error).

(b) The error message is the same as issued by routine DISTLD and is printed when data errors are detected.

EQLBM

WARNING **** NO APPLIED LOADING – CHECK WHETHER ALL BOUNDARY CONDITIONS ARE DISPLACEMENTS (ROUTINE EQLBM)

This is a precautionary message to draw attention to the fact that no load of any significant magnitude has been applied in the current increment. Probably the analysis is displacement controlled.

FIXX

(a) ***** ERROR : LTH FIXITY. ELEMENT M DOES NOT CONTAIN NODES : N1 N2 (ROUTINE FIXX)

This is output when either or both of the nodes N1, N2 are not present in element M. This error is encountered when reading the list of fixities and it is the Lth fixity.

(b) When more than 200 fixities have been specified the following message is printed. The size of arrays MF, TF and DXYT has to be increased.

INCREASE SIZE OF ARRAYS MF, TF AND DXYT IN COMMON BLOCK FIX (ROUTINE FIXX)

This message is printed as soon as the number of fixities exceeds 200, but does not say by how much the array sizes have to be increased. The COMMON statement labelled FIX appears in the following routines and a change in size of the arrays means altering all these routines:

ANS CPW FIXX FRFXLD FRONTZ INSITU RESTRN
RESTRT UPOUT

The parameter MXFXT must be set to the new size of the arrays in routine MAXVAL.

FRONTZ

(a) PROBABLE SERIOUS ILL-CONDITIONING (ROUTINE FRONTZ)

This is an indication of possible numerical problems, for example when a very stiff structure is lightly sprung to earth.

(b) ERROR – ZERO PIVOT (ROUTINE RONTZ)

This happens when the diagonal stiffness term of the unknown equation which is about to be eliminated is found to be equal to zero. The above message can be expected when the permeabilities have been incorrectly set to zero in the material property table or when the time increment has been specified as zero in a **consolidation** analysis. Otherwise check that elastic parameters have not been inadvertently specified as zero.

INSTRS

ELEMENT M IS OF UNKNOWN MATERIAL TYPE L (ROUTINE INSTRS)

Check that the material type number of element M is within the permissible range.

LODLST

The size of arrays LEDG, NDE1, NDE2 and PRESLD is set at 100. If more than 100 element sides are subjected to pressure load in the input data, the following message is printed.

INCREASE SIZE OF ARRAYS IN COMMON BLOCK PRSLD
ALSO SET MXLD IN ROUTINE MAXVAL (ROUTINE LODLST)

The value of MXLD in routine MAXVAL must be set to the new (increased) size of these arrays. The above arrays occur in the following routines and they have to be changed:

ANS CPW EQLOD INSITU LODINC LODLST RESTRT UPOUT

LSTIFF

If the program stops in routine LSTIFF with a message that an attempt has been made *to divide by zero* then in a consolidation analysis check that PR(7,KM), the unit weight of water, is not specified as zero in the input data. In a supposedly non-consolidation analysis, check for the presence of consolidation elements (types 3 and 7 in 2-D) in the input data.

MAST

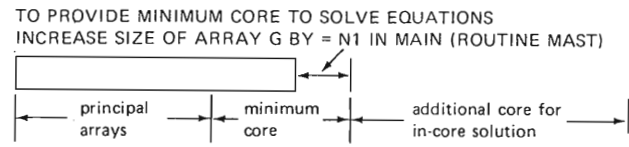
INADMISSIBLE VALUE FOR MXTYP LT (ROUTINE MAST)

MXTYP (in record B) is out of the admissible range 1 to 7.

INCREASE SIZE OF ARRAY G BY N1 FOR GEOMETRY
PART OF PROGRAM (ROUTINE MAST)

The array allocation G(55000) is insufficient for the geometry part of the program. The size of G must be increased by the specified amount in routine MAIN.

TO PROVIDE MINIMUM CORE TO SOLVE EQUATIONS
INCREASE SIZE OF ARRAY G BY = N1 (ROUTINE MAST)



The situation is illustrated in the above figure. N1 is the amount by which the size of array G has to be increased in routine MAIN. Note that this only provides the minimum core, and the equations are solved out-of-core.

MIDPOR and MIDSID

The following two statements appear when the estimated allocation for the additional nodes has been exceeded. The first statement is for the user node numbers and the second is for the program node numbers. These messages are unlikely to be printed, because the estimated allocation for the additional nodes is always much higher than the actual number.

(a) INCREASE NO. OF ADDITIONAL NODES (ROUTINE MIDPOR or MIDSID)

(b) *** ERROR *** MORE THAN NNE NODES IN MESH

RDPROP

ERROR IN NO. OF INCREMENTS = N1 INCS = N2 INCF = N3
(ROUTINE RDPROP)

This message is printed when the increment at the finish of the analysis (N3) is less than the increment number at start of the analysis (N2).

RDSTRS

WARNING – POINT OUTSIDE IN-SITU STRESS SPACE
(ROUTINE RDSTRS)

The above message is followed by the (program) element number and the integration point number. This happens when the *in situ* region (defined by a set of layers) does not cover the entire region of the **primary** mesh.

REACT

INCREASE ARRAY SIZE OF R, NDENO, NDIR IN ROUTINE REACT

When the number of reactions exceeds 150, this message is printed. The array size of R, NDENO and NDIR must be increased, and NCT must be set equal to this new size.

SFWZ

(a) NO ELEMENTS IN SOLUTION : (ROUTINE SFWZ)

When no elements are left in the mesh (possibly due to user error; elements removed incorrectly) this message appears.

(b) *** ERROR ** TOO MANY DEGREES OF FREEDOM IN FRONT
EXCEEDS IFRZ (ROUTINE SFWZ)

The allocation for maximum frontwidth is IFRZ and is set equal to 300 in routine MAST. If the maximum frontwidth exceeds 300 this message is printed. The allocation for the maximum frontwidth (IFRZ) must be increased in routine MAST.

(c) PROGRAM ERROR – NO NODE ON END OF FRONT
(ROUTINE SFWZ)

The message is unlikely to appear. When the FRONT shrinks owing to variables being eliminated from the end of the FRONT, the FRONT size is re-calculated. This message would indicate a program error.

(d) PROGRAM ERROR – LAST APPEARANCE NODE IS NOT IN FRONT

This is also an unlikely error.

SHAPE

(a) ELEMENT M IS OF UNKNOWN TYPE *** LT
(ROUTINE SHAPE)

When a request is made to calculate shape functions for inadmissible element types, this message is printed.

(b) ELEMENT M IS OF TYPE LT NOT IMPLEMENTED
(ROUTINE SHAPE)

This message is printed when a request is made to calculate shape functions for element types which have not been implemented.

SHFNPP

(the messages are the same as for routine SHAPE)

UPOUT

WARNING **** THE NODAL CO-ORDINATES ARE UPDATED

This is just to tell the user that the above option is being used.

Appendix D: Incorporation of a new soil model

Relatively few changes are necessary for incorporating a new soil model into CRISP. The soil model is assigned a material type number in the range 1 to 10, which has not been previously allocated. The user provides the routine (for example DSOILN) which calculates the **D** matrix for a given stress state. For the purpose of illustration the new model is assigned material type number 5. The material constants can be specified in one data record cosisting of 10 values. In general this is sufficient to specify all the material constants. The data record is laid out as follows.

MAT NTY P(1) P(2) ..... P(10)

where

MAT	is the material zone number; all elements given the same number will have the following properties.
NTY	is the material type number, which is specific for each soil model.
P(1)–P(6)	are user-defined soil parameters; these could be E, ν' or G.
P(7)	0 for drained, α for undrained, γ_w for consolidation.
P(8)	γ soil density (weight/unit volume).
P(9)	k_x for consolidation, 0 for drained or undrained.
P(10)	k_y for consolidation, 0 for drained or undrained.

If more material constants are required to be specified, additional parameters can be read. The material constants that are read are placed in PR(NPR,NMT)

and the material type numbers in NTY(NMT). The **D** matrix is needed only twice during the analysis: once to calculate the stiffness matrix (called by routine LSTIFF) and once to calculate the incremental stresses from strains (called by routine UPOUT). When the **D** matrix has to be calculated, a subroutine call is made; the stresses and other parameters are passed as arguments. In return, the routine calculates the components of the **D** matrix and puts them in array D(NS,NS).

The user can select the arguments when a subroutine call is made to the routine DSOILN.

```
      CALL DSOILN (IP,K,IBLK,NEL,NIP,NVRS,NDIM,NS,
     1 NPR,NMT,VARINT,MAT,D,PR,IPLSTK,BK)
```

The first statement of routine DSOILN is then

```
      SUBROUTINE DSOILN(I7,I,IET,NEL,NIP,NVRS,NDIM,NS,
     1 NPR,NMT,VARINT,MAT,D,PR,IPLSTK,BK)
```

IP,I7	integration point
K,I	element number
IBLK,IET	code to indicate whether to add bulk modulus of water to the stiffness terms or not
NEL	total number of elements
NIP	the maximum number of integration points in any element
NVRS	number of stress components and parameters
NDIM	number of dimensions to problem
NS	number of stress/strain components
NMT	allowable number of different material zones
VARINT	current stress parameters
MAT	material zone numbers of elements
IPLSTK	return code set by routine. 0–elastic, 1–plastic
BK	bulk modulus of soil

This is followed by the relevant comment statements about the model. Then comes the DIMENSION and COMMON statements.

```
      DIMENSION MAT(NEL),VARINT(NVRS,NIP,NEL)
      DIMENSION D(NS,NS),PR(NPR,NMT)
```

The following subroutines need changing to incorporate the new soil model:

MAST MAXVAL LSTIFF RDSTRS UPOUT UPOUT2

MAST

No changes are necessary to this routine if the new soil model does not require more than 10 material constants. If the new model requires (say) 12 material

constants, the size of array PR is changed from

PR(10,10) to PR(12,10)

Note that 12 material constants must then be provided for all material models in the data (zeroes being added for the present models).

MAXVAL

If the size of array PR has been changed in routine MAST then statement

NPR = 10

is replaced by

NPR = 12

LSTIFF

A statement call is made to the routine which calculates the **D** matrix when the stiffness terms are to be calculated. The statement

GOTO(39,32,33,34), KGO

is replaced by

GOTO(39,32,33,34,35),KGO

and the following statements are included:

```
      GOTO 39
35 CALL DSOILN(IP,K,IBLK,NEL,NIP,NVRS,NDIM,NS,NPR,NMT,
  1  VARINT,MAT,D,PR,IPLSTK,BK)
```

before the statement

39 CALL LSTIFA(SS, B, D, DB, F9, NS, NB)

RDSTRS

From values specified at *in situ* nodes, the stresses at integration points are interpolated. For Cam-clay models the voids ratio is calculated from p' and p'_c. For elastic models, no extra parameters are calculated. If the new model is a linear elastic model (of type number 5) then no changes need to be made to routine RDSTRS. If the new soil model is a version of the critical state model then any relevant calculations can be carried out, as done for Cam-clays, between statements.

GO TO(60,60,52,52,60,60), KGO

and

60 CONTINUE

This can be done by using the following statement:

GO TO(60,60,52,52,55,60), KGO

and then inserting

```
      GOTO 60
55    CONTINUE
        .............
      < calculations for new model >
        ............
```

before the statement

60 CONTINUE

UPOUT

This routine calls routines that calculate the **D** matrix to evaluate the stress increments from the strain increments. The changes are similar to the ones made to routine LSTIFF. The statement

GOTO(59,52,53,54), KGO

is replaced by

GOTO(59,52,53,54,55), KGO

and the following statements are included

```
      GOTO 59
55 CALL DSOILN(IP,J,0,NEL,NIP,NVRS,NDIM,NS,NPR,NMT,
  1  VARINT,MAT,D,PR,IPLSTK,BK)
     IELST = 1  or  ICAM = 1
```

– the last if the new model is a critical state model – before statement

59 DO 60 II = 1, NS

The stresses calculated in these routines are output in two tables:

(a) contains the general stresses σ'_x, σ'_y, σ'_z, τ_{xy}, u, σ_{I}, σ_{III} and θ_{xy} (τ_{yz} and τ_{zx} for 3-D);

(b) contains stress parameters relevant to the particular model, e.g. for Cam-clays p', q, p'_c, e_{strs}, e_{strn}, codes. e_{strs} and e_{strn} are the voids ratios calculated from the stress state and from the volumetric strains respectively. The codes are the numbers indicating the type of Cam-clay behaviour (see section 9.10).

The parameters in category (a) are printed for all types of soil model. However, the parameters in category (b) are calculated by routines specific to particular models, i.e. for Cam-clay models – routines EVCAM and VARCAM.

These additional parameters are output in routine UPOUT2. Therefore if the new soil model requires additional parameters to be output then the user has to provide the relevant routines for his/her new model; further changes are then necessary to routines UPOUT and UPOUT2.

These additional parameters for each integration point may be stored in array VARC(NCV,NIP,NEL) until they can be output by routine UPOUT2. If more parameters are required per integration point, NCV (at present equal to 10) can be increased in routine UPARAL. If codes are required to issue warning messages then the following arrays can be used.

LCS(NIP,NEL), LNGP(NIP,NEL)
MCS(NEL),MNGP(NEL)
NCODE(NIP,NEL),NELCM(NEL)

Array NELCM(NEL) is used to identify the different material types that require additional output parameters.

1 – Cam-clays

therefore the new soil model can be assigned the number 2.

In summarising it can be said that if the new soil model is an elastic model which does not require additional material constants to be read or additional parameters to be output then only routines LSTIFF and UPOUT need to be changed. If additional input data (material constants) are required then changes are also necessary to routines MAST and MAXVAL. Further changes are necessary if additional output parameters are to be printed. The user then has to provide new routines to calculate these output parameters.

Appendix E: Incorporation of a new element type

E.1 INTRODUCTION

Almost any new element type could be incorporated into CRISP. Three element types are discussed below for the purposes of illustrating the basic techniques. They are listed in the order of increasing difficulty to incorporate. The eight-noded quadrilateral can be incorporated more readily than the 20-noded brick element. The three-noded beam element would require major reorganisation because of its additional d.o.f. being rotation, whereas it is tacitly assumed for two-dimensional analysis that the third d.o.f., if present, is excess pore pressure. The use of a beam element with consolidation elements would require modification the way the d.o.f. of a node are identified.

(i) 8-noded quadrilaterals.
(ii) 20-noded brick element.
(iii) 3-noded beam element (with bending stiffness).

It should be pointed out that the way CRISP has been written makes the incorporation of elements (i) and (ii) fairly straightforward. Tentatively the following element type numbers have been allocated for elements (i) and (ii).

Element	Type number
8-noded quadrilateral	4
8-noded quadrilateral (consolidation)	5
20-noded brick	8
20-noded brick . . . . . (consolidation)	9

Element type information has been included in block data routine BDATA1 for the above elements. The local node numbering assumed in setting up the data for the new elements is shown in Fig. E.1. Before incorporating a new element type the user should understand how array LINFO is organised in routine BDATA1 and how this information is used in the rest of the program.

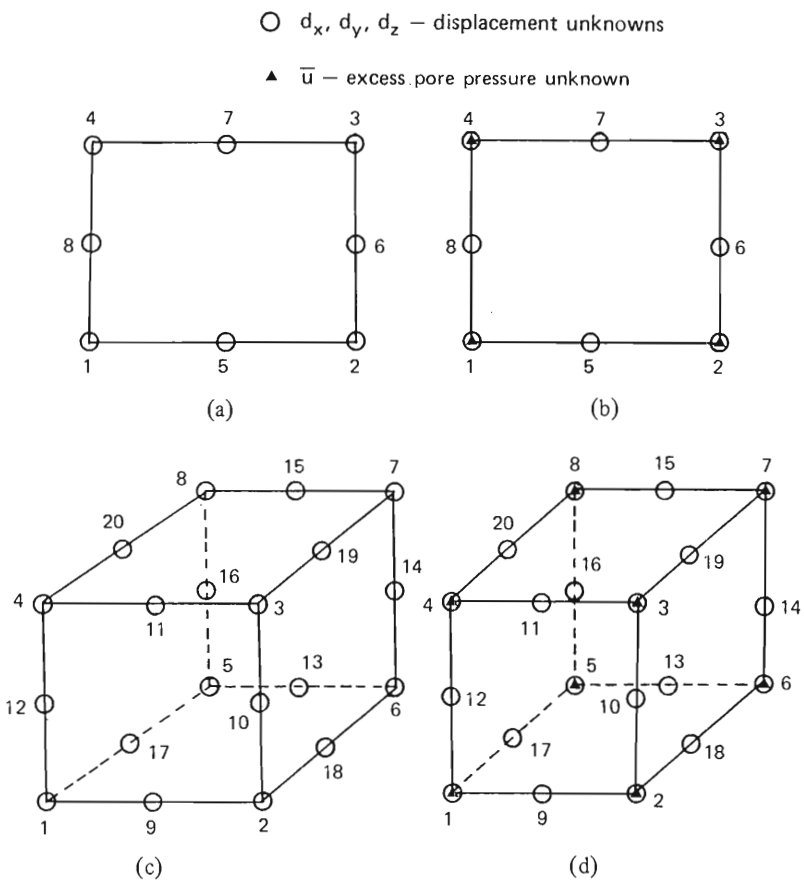


Fig. E.1 – Different element types

(a) LSQ (element type 4): 8 nodes, 16 d.o.f.
(b) LSQ (element type 5) – consolidation: 8 nodes, 20 d.o.f.
(c) LSB (element type 8): 20 nodes, 60 d.o.f.
(d) LSB (element type 9) – consolidation: 20 nodes, 68 d.o.f.

All vertex or corner nodes are numbered first (these are local node numbers which are different from the node numbers assigned in the input data). The node numbers for nodes along element edges or sides and element interiors then

follow. This sequence of numbering adopted and set in BDATA1 is used in the rest of the program. For example, in routine MIDSID for element type 4, information set in SETNP gives the vertex nodes at either end of node 5 as 1 and 2. This is then used in calculating the co-ordinates of node 5.

E.2 ELEMENT TYPE DEPENDENT DATA

If we consider element type 9, which is the 20-noded brick element used for consolidation analysis, the vertex nodes 1 to 8 have 4 d.o.f. (displacements in x, y, z directions and the excess pore pressure). Nodes 9 to 20 have only 3 d.o.f. (the displacements).

The element type information is stored in the LINFO array in routine BDATA1. Row 9 (the row assigned to each element is its type number) is allocated to this element; that is each element is allocated a row of LINFO.

LINFO column entry	Explanation
1	The total number of nodes (including pore pressure nodes) is 20. NDPT = 20.
2	Total number of vertex or corner nodes is 8. NVN = 8.
3	Total number of element edges is 12. NEDG = 12.
4	The element has 6 faces. NFAC = 6. Note that for all two-dimensional elements, NFAC = 1.
5	The number of displacement nodes is 20. NDN = 20.
6	There are 8 nodes with pore pressure variables. NPN = 8.
7	There is only one displacement node along each edge, at the mid-point. NDSD = 1.
8	Even though there is a node at the midpoint of each edge it does not have a pore pressure variable. NPSD = 0.
9	There are no inner displacement nodes. NIND = 0.
10	There are no inner pore pressure nodes. NINP = 0.
11	The number of integration points is 27 if using the 3 X 3 X 3 scheme. NGP = 27.
12	Index to the weights (array W) and local co-ordinates (array L). Different regions of these arrays are allocated to different element types as follows: W(1) – W(5) for 3-noded bar – any scheme up to 5 point W(6) – W(12) for LST elements – 7-point scheme W(13) – W(21) for quadrilaterals – 3 X 3 scheme W(22) – W(37) for CuST elements – 16-point scheme W(38) – W(64) for 20-noded brick – 3 X 3 X 3 scheme All that is needed is a pointer to indicate the last location allocated to the previous element type. For the case of the 20-noded brick elements it is 37. NDX = 37.

LINFO column entry	Explanation
13	Index to vertex nodes (not used in the present version). INX = 0.
14	Index to nodes along element edges. This gives the starting index to arrays NP1 and NP2 which are set up in routine SETNP. In order to calculate the co-ordinates of nodes along element edges it is necessary to know which nodes are at either end. Arrays NP1 and NP2 give the local node numbers (which are the index to array NCONN) for each element edge. Different regions of NP1 and NP2 are allocated to different element types (similar to arrays W and L) and this provides the starting index. For example, node 18 is the midside node along edge 10. NP1(INDED + 10) = 2; NP2(INDED + 10) = 6. This means that nodes 2 and 6 are at either end of node 18. INDED is the starting index for different element types. For the brick element it is 3. INDED = 3.
15	The number of local co-ordinates is 3 (ξ, η and ζ). NL = 3.
16	Total number of d.o.f. in element. 8 vertex nodes have 4 d.o.f., each giving 8 X 4 = 32 d.o.f. 12 midside nodes have 3 d.o.f., each giving 12 X 3 = 36 d.o.f. The total number of d.o.f. is 68. MDFE = 68. This is used in calculating the size of the element stiffness matrix (see routine MAXVAL).
17	Centroid integration point. The last integration point, which is 27, is situated at the centroid of the element. This is used in outputting representative values of stresses and strains of an element. However, not all integration schemes have an integration point at the centroid. Under these circumstances the last integration point is used here. NCGP = 27. If the integration scheme used has an integration point at the centroid it is assigned the last number for convenience.
21	onwards. Each entry gives the number of d.o.f. of each node as labelled in Fig. E.1.

Node number	1 to 8	9 to 20
Location in LINFO	21 to 28	29 to 40
d.o.f.	4	3

Local co-ordinates

The integration points have been numbered as shown in Fig. E.2. The order in which they have been numbered is not important. However, once numbered, the same sequence is implicitly assumed in different parts of the program and it should be consistent with the initial numbering. The local co-ordinates are stored in a region allocated in array L.

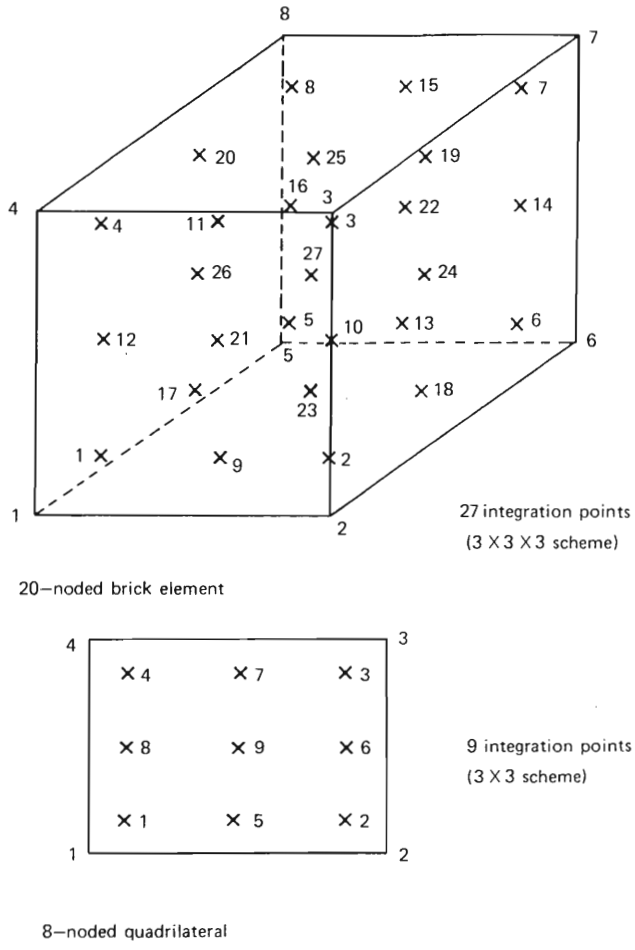


Fig. E.2 – Integration points

For example, L(1,38), L(2,38) and L(3,38) are the ξ, η and ζ co-ordinates of integration point 1. L(1,39), L(2,39) and L(3,39) are the ξ, η and ζ co-ordinates of integration point 2, and so on. Note that the quadrilateral elements have only two local co-ordinates ξ and η. Then only L(1,IP), L(2,IP) need to be set.

Weighting factors

The array W is set up in a similar manner to array L. Array W gives the weighting factors for each integration point shown in Fig. E.2.

W(38) is the weighting factor for IP = 1
W(39) is the weighting factor for IP = 2 and so on.

E.3 INCORPORATION OF 8-NODED QUADRILATERALS

The only routines which need changing are as follows:

SHAPE – include displacement shape functions and derivatives w.r.t. local co-ordinates for element types 4 and 5. This replaces the WRITE(IW6, 910) statement.

The displacement shape functions are as follows:

$$
\begin{aligned}
N1 &= -(\zeta\zeta - 1)(\xi\xi - 1)(1 + \xi\xi + \zeta\zeta)/4 \\
N2 &= \ (\zeta\zeta - 1)(\xi\xi + 1)(1 - \xi\xi + \zeta\zeta)/4 \\
N3 &= \ (\zeta\zeta + 1)(\xi\xi + 1)(\xi\xi + \zeta\zeta - 1)/4 \\
N4 &= -(\zeta\zeta + 1)(\xi\xi - 1)(\zeta\zeta - \xi\xi - 1)/4 \\
N5 &= \ (\zeta\zeta + 1)(\xi\xi + 1)(\xi\xi - 1)/2 \\
N6 &= -(\xi\xi + 1)(\zeta\zeta + 1)(\zeta\zeta - 1)/2 \\
N7 &= -(\zeta\zeta + 1)(\xi\xi + 1)(\xi\xi - 1)/2 \\
N8 &= \ (\xi\xi - 1)(\zeta\zeta + 1)(\zeta\zeta - 1)/2
\end{aligned}
$$

SHFNPP – include pore pressure shape functions and derivatives w.r.t. local co-ordinates for element type 5. This replaces the WRITE(IW6,910) statement.

The pore pressure shape functions are as follows:

$$
\begin{aligned}
M1 &= \ (\xi\xi - 1)(\zeta\zeta - 1)/4 \\
M2 &= -(\xi\xi + 1)(\zeta\zeta - 1)/4 \\
M3 &= \ (\xi\xi + 1)(\zeta\zeta + 1)/4 \\
M4 &= -(\xi\xi - 1)(\zeta\zeta + 1)/4
\end{aligned}
$$

The internal node numbering used is fairly important (see Fig. E.1). All other node co-ordinates are calculated from the vertex nodes. For example, the co-ordinates of node 5 are calculated from the co-ordinates of nodes 1 and 2. The data NP1 and NP2 set in routine SETNP give the nodes at either end of the first edge as 1 and 2. The node along the side 1–2 is numbered first after the vertex nodes because it is assigned a number 5.

E.4 INCORPORATION OF 20-NODED BRICK ELEMENTS

The list of subroutines which need changing is listed below:

BDATA1

Include appropriate data in arrays W and L in designated locations using the DATA statements.

W(38) ... W(64) – weighting factors
L(1,38) ... L(3,64) – local co-ordinates

(see Table 8.1, p. 198, of Zienkiewicz, 1977)

MAST

If both element types 8 and 9 are to be used then set LTZ = 9.

SETNP

Arrays NP1 and NP2 have already been set up to identify each element edge by the nodes at either end (see Fig. E.1).

SHAPE

Include appropriate shape function statements and derivatives w.r.t. local co-ordinates for element types 8 and 9. (These should be consistent with the node numbering in Fig. E.1.)

SHFNPP

Include pore pressure shape functions for element type 9 only. (These should be consistent with the node numbering in Fig. E.1.)

FIXX

The routine FIXX can only handle two-dimensional elements. Therefore FIXX is renamed as FIXX2, and all call statements CALL FIXX(. . . . .) are replaced by IF(NDIM.EQ.2) CALL FIXX2(. . . . .). The routines INSITU and ANS are the only ones which call routine FIXX.

This statement is followed by the statement

IF(NDIM.EQ.3) CALL FIXX3(. . . . .)

Routine FIXX3, which is for the 20-noded brick element, is listed in section E.7.

General changes that need to be carried out to routines EQLBM, INSTRS and UPOUT for implementing the three-dimensional analysis option are as shown below.

The header and write statements in these routines at present only cater for two-dimensional elements. All write statements listed below (identified by the format statements) should be preceded by IF(NDIM.EQ.2) and a further write statement added for the three-dimensional case preceded by IF(NDIM.EQ.3). This is illustrated for the UPOUT routine.

The changes to routine UPOUT are as follows:

```
   1 WRITE(IW6,902)
     GOTO 6
   2 WRITE(IW6,901)
     GOTO 6
   3 WRITE(IW6,933)
     GOTO 6
   4 WRITE(IW6,934)
C-------------------------------------------------------------------
C     UPDATE  ABSOLUTE  DISPLACEMENTS
C-------------------------------------------------------------------
```

```
                .................
                .................
                .................
C-------------------------------------------------------------------
      N2=N1+NQL-1
      IF(NDIM.EQ.3)GOTO 9
      IF(NQL.EQ.3)WRITE(IW6,900)JR,(DI(JJ),JJ=N1,N2),(DA(JJ),JJ=N1,N2)
      IF(NQL.EQ.2)WRITE(IW6,910)JR,(DI(JJ),JJ=N1,N2),(DA(JJ),JJ=N1,N2)
      IF(NQL.EQ.1)WRITE(IW6,911)JR,DI(N1),DA(N1)
      GOTO 10
    9 CONTINUE
      IF(NQL.EQ.4)WRITE(IW6,940)JR,(DI(JJ),JJ=N1,N2),(DA(JJ),JJ=N1,N2)
      IF(NQL.EQ.3)WRITE(IW6,941)JR,(DI(JJ),JJ=N1,N2),(DA(JJ),JJ=N1,N2)
   10 CONTINUE
      IF(NDIM.EQ.3)GOTO 12
      IF(IOUT2.EQ.2)WRITE(IW6,904)
      IF(IOUT2.EQ.1)WRITE(IW6,906)
      GOTO 14
C
   12 CONTINUE
      IF(IOUT2.EQ.2)WRITE(IW6,904)
      IF(IOUT2.EQ.1)WRITE(IW6,936)
C
   14 CONTINUE

                .................
                .................
                .................

      IF(MR.LT.NELOS.OR.MR.GT.NELOF)GOTO 26
      WRITE(IW6,908)MR
      IF(NDIM.EQ.2)WRITE(IW6,914)
      IF(NDIM.EQ.3)WRITE(IW6,944)

                .................
                .................
                .................

  933 FORMAT(//20H NODAL DISPLACEMENTS/1X,19(1H-)//
     1 18X,11HINCREMENTAL,51X,8HABSOLUTE//
     1 2X,4HNODE,7X,2HDX,13X,2HDY,13X,2HDZ,28X,2HDX,13X,2HDY,13X,2HDZ/)
  934 FORMAT(//46H NODAL DISPLACEMENTS AND EXCESS PORE PRESSURES/
     1 1X,45(1H-)//26X,11HINCREMENTAL,51X,8HABSOLUTE//
     1 2X,4HNODE,7X,2HDX,13X,2HDY,13X,2HDZ,13X,2HDU,
     1 13X,2HDX,13X,2HDY,13X,2HDZ,13X,2HDU/)
  936 FORMAT(//30H STRESSES AT ELEMENT CENTROIDS/1X,29(1H-)//8H ELEMENT,
     1 3X,1HX,13X,1HY,12X,1HZ,11X,2HSX,11X,2HSY,11X,3HSZ,11X,3HTXY,
     1 11X,3HTYZ,10X,3HTZX,11X,1HU)
  940 FORMAT(1X,I5,8E15.5)
  941 FORMAT(1X,I5,3E15.5,15X,3E15.5)
  944 FORMAT(2X,2HIP,7X,1HX,12X,1HY,12X,1HZ,11X,2HSX,
     1 11X,2HSY,11X,2HSZ,10X,3HTXY,10X,3HTYZ,10X,3HTZX,9X,1HU)
```


The complete list of changes is as follows.

Routine	No. of new write statements	Format statements	Explanations
EQLBM	6	900	header for out-of-balance loads.
		904	header for out-of-balance loads.
		901	output statement.
		903	header for overall equilibrium check.
		905	header for overall equilibrium check.
		907	output equilibrium check.
INSTRS	2	901	header for stresses.
		903	output stresses.
UPOUT	7	901	header for displacements.
		902	header for displacements.
		910	output displacements.
		911	output displacements.
		906	header for stresses.
		914	output element number.
		916	output stresses.

E.5 INCORPORATION OF ANY OTHER ELEMENT TYPE

At present, array LINFO has 15 rows allowing for new element types to be incorporated. Tentatively the first 11 rows have already been allocated to different element types. A new element type that is different from any element in the current library could be assigned type number 12, and the next one the number 13, and so on. If a new element with type number greater than 15 is introduced then the number of rows in array LINFO should be increased to 20. This change should be made to all routines which access this array which resides in named common ELINF. (See Appendix F, which gives the list of subroutines which contain the common block ELINF.)

Here follows a list of subroutines which need to be changed.

BDATA1

Add element type dependent data to following arrays:

LINFO (12,) – define element type
L(,) – local co-ordinates of integration points.
W( ) – weighting factors

MAST

The maximum admissible type number has been set to 7 in routine MAST, i.e. LTZ = 7. If a new element type 12 is introduced then set LTZ = 12 in routine MAST.

Increase the size of arrays NAD and KLT to 12. Set NAD(12) to the number of additional nodes in the element (= total number of nodes minus vertex nodes) at the end of the existing DATA statement.

SETNP

Increase the size of arrays NPL1 and NPL2 to required amount and set NPL to the increased value in routine MAST. Add relevant entries to NPL1 and NPL2 in the DATA statement. A separate entry is used for each element edge, and NPL1 gives the local node number of the node at one end and NPL2 the one at the other end. The total number of entries for an element equals the number of edges or sides the element has.

MIDPOR and MIDSID

Changes are only required if new elements have inner displacement or pore pressure nodes. For example, see the CuST element type.

LSTFSG

For consolidation elements with type number greater than 11, arrays KP and KD have to be extended to provide data regarding the new element type. These should be consistent with the node numbering in Fig. E.1. These arrays are used in reorganising the rows/columns so that all d.o.f. of a particular element occupy consecutive rows/columns in array SG.

The following is a list of routines which have GOTO statements where the range is the admissible element type numbers. Therefore in general there are 11 possible destinations, depending on the element type. A new entry should be added (if LT > 11) to this statement corresponding to the new element type in each routine where the GOTO statement appears.

Subroutine	LT range	Remarks on destination
MAXVAL	11	12 – for consolidation elements. 11 – for non-consolidation elements.
MIDSID	11	27 – nodes with inner node (i.e. CuST). 90 – other elements.
MIDPOR	11	12 – elements with inner pore pressure nodes, i.e. CuST of type 9. 100 – all other elements.
	11	27 – elements with inner pore pressure nodes. 90 – all other element types.
RDSTRS	15	replace 80 to 22 for new element type.
SHAPE	11	add shape functions at appropriate place and delete WRITE statement.
SEL1	11	22 for all 2-D and 3-D elements.

Subroutine	LT range	Remarks on destination
LSTIFF	11	1 – non-consolidation elements. 2 – consolidation elements.
SHFNPP	11	80 – non-consolidation elements. Add pore pressure shape function statements.
UPOUT	11	1 – non-consolidation elements. 2 – consolidation elements.
	11	25 – non-consolidation elements. 23 – consolidation elements.
	11	66 – consolidation elements. 70 – non-consolidation elements.

Element type dependent information has been set up in the following routines.

Routine	Arrays	Remarks
BDATA1	LINFO	Define element type.
	L	Local co-ordinates.
	W	Weighting factors.
MAST	NAD, KLT	Number of additional nodes in each element.
SETNP	NP1, NP2	Local node numbers at either end of each edge or side.
LSTFSG	KP, KD	Used in reorganising element stiffness matrix with consecutive rows/columns assigned to all d.o.f. of a node.

E.6 CHANGING THE INTEGRATION SCHEME

Under certain circumstances reduced integration using a 2 X 2 scheme is preferable to the full 3 X 3 integration scheme for the 8-noded quadrilaterals. The present version is set up for the full integration scheme. In order to use the 2 X 2 scheme, the following changes need to be carried out in routine BDATA1.

Replace 9 in LINFO(11,4) and LINFO(11,5) by 4

There is no integration point at the centroid for the 2 X 2 scheme; therefore the results at the last integration point are output.Then

W(13)–(W(16)	are used for the weighting factors.
L(,13)–L(,16)	are used for the local co-ordinates for the integration points.

However, if the user would like both integration schemes available then the element with the 2 X 2 integration scheme can be introduced as a new element type (for example 12). This would need changes to other routines as well (see section E.5).

E.7 NEW SUBROUTINE FOR 3-D ELEMENT: 20-NODED BRICK

```
      SUBROUTINE FIXX3(IR5,IW6,NEL,NTPE,NDIM,NPL,LV,NCONN,LTYP,MUMAX,
     1 NNZ,NP1,NP2,MREL,NREL,V,NFX)
C*********************************************************************
C     ROUTINE TO MAINTAIN A LIST OF NODAL FIXITIES. INTERPRETS
C     FIXITIES ALONG (3-D) ELEMENT FACE INTO NODAL FIXITIES.
C     AT PRESENT TO CATER FOR THE 3-D BRICK ELEMENTS ONLY.
C*********************************************************************
      INTEGER TF
      DIMENSION NCONN(NTPE,NEL),LTYP(NEL),MREL(MUMAX),NREL(NNZ)
      DIMENSION IND(8),FV(8),V(LV),NP1(NPL),NP2(NPL)
      DIMENSION KX(48),NDU(8),NDP(8),NXC(4),NXM(4),KNL(8)
      COMMON /FIX   / DXYT(4,200),MF(200),TF(4,200),NF
      COMMON /ELINF / LINFO(50,15)
C---------------------------------------------------------------------
C     ARRAY KX(48) GIVES THE INDEX TO ARRAY NCONN FOR THE FOUR
C     CORNER NODES OF EACH FACE OF THE ELEMENT FOLLOWED BY THE
C     MIDSIDE NODES.
C---------------------------------------------------------------------
      DATA KX(1),KX(2),KX(3),KX(4),KX(5),KX(6),KX(7),KX(8),KX(9),
     1 KX(10),KX(11),KX(12),KX(13),KX(14),KX(15),KX(16),KX(17),
     1 KX(18),KX(19),KX(20),KX(21),KX(22),KX(23),KX(24),KX(25),
     1 KX(26),KX(27),KX(28),KX(29),KX(30),KX(31),KX(32),KX(33),
     1 KX(34),KX(35),KX(36),KX(37),KX(38),KX(39),KX(40),KX(41),
     1 KX(42),KX(43),KX(44),KX(45),KX(46),KX(47),KX(48)/
     1 1,2,3,4,9,10,11,12,6,5,8,7,13,16,15,14,1,5,6,2,17,13,18,9,
     1 2,6,7,3,18,14,19,10,4,3,7,8,11,19,15,20,5,1,4,8,17,12,20,16/
C
      DO 5 IU=1,8
      KNL(IU)=0
      NDU(IU)=0
      NDP(IU)=0
    5 CONTINUE
C
      NFZ=200
      NDIM1=NDIM+1
      IF(NFX.EQ.0)RETURN
      WRITE(IW6,900)
C---------------------------------------------------------------------
C     IF NEW 3-D ELEMENT TYPES ARE ADDED THEN NC, NFCD
C     AND LVL (= NFCD) SHOULD BE OBTAINED FROM ARRAY LINFO
C     IN ORDER TO MAKE THE ROUTINE GENERAL.
C---------- NC   - NUMBER OF VERTEX NODES ON ELEMENT FACE
C---------- NFCD - TOTAL NUMBER OF DISPLACEMENT NODES ON FACE
      NC=4
      NFCD=8
C---------------------------------------------------------------------
C     LOOP ON ALL FACES WITH FIXITIES I.E. FACES WITH PRESCRIBED
C     DISPLACEMENT/EXCESS PORE PRESSURES.
C---------------------------------------------------------------------
      LVL=NFCD
      DO 200 JX=1,NFX
      READ(IR5,*)ML,(NDU(J),J=1,NC),IVAR,IFX,(FV(K),K=1,LVL)
      WRITE(IW6,910)JX,ML,(NDU(J),J=1,NC),IVAR,IFX,(FV(K),K=1,LVL)
      NE=MREL(ML)
C
      DO 30 IN=1,NC
      ND=NDU(IN)
```



```
   30 NDP(IN)=NREL(ND)
C
      LT=LTYP(NE)
      LT=IABS(LT)
      NFAC=LINFO(4,LT)
C---------- LOOP ON ALL FACES OF ELEMENT.
C---------- TO IDENTIFY THE FACE OF THE ELEMENT WITH
C---------- PRESCRIBED VALUES.
      DO 90 IFAC=1,NFAC
      ISX=NFCD*(IFAC-1)
C---------- GET INDEXES OF NODES TO NCONN
      DO 40 IN=1,NC
      NXC(IN)=KX(ISX+IN)
C---------- IF NOT PORE-PRESSURE D.O.F., ADDITIONAL NODES ALONG EDGE
C---------- ARE PRESENT
      IF(IVAR.NE.NDIM1)NXM(IN)=KX(ISX+NC+IN)
   40 CONTINUE
C--------- GET VERTEX NODES OF FACE FROM NCONN
      DO 50 IN=1,NC
      IP=NXC(IN)
   50 KNL(IN)=NCONN(IP,NE)
C---------- LOOP ON ALL STARTING NODES.
C---------- TRY TO MATCH THE NODES SPECIFIED BY THE USER
C---------- WITH THE NODES ON EACH FACE. EACH NODE IN
C---------- TURN IS CONSIDERED AS A STARTING NODE.
      DO 80 IS=1,NC
      ISV=IS
C---------- TRY MATCHING THE NODES
      DO 60 IN=1,NC
      IF(NDP(IN).NE.KNL(IN))GOTO 65
   60 CONTINUE
      GOTO 95
C---------START WITH THE NEXT NODE.  THE SEQUENCE OF
C---------THE NODES ARE STILL THE SAME
   65 CALL ALTER(IW6,KNL,NC)
   80 CONTINUE
   90 CONTINUE
C---------- FACE NOT FOUND
      WRITE(IW6,930)JX,ML,(NDU(J),J=1,NC)
C
      GOTO 200
C
   95 IF(ISV.EQ.1)GOTO 105
      IS1=ISV-1
C---------- SORT THE INDEXES TO MATCH WITH NODE SEQUENCE KNL
      DO 100 IM=1,IS1
      CALL ALTER(IW6,NXC,NC)
      IF(IVAR.NE.NDIM1)CALL ALTER(IW6,NXM,NC)
  100 CONTINUE
C---------- IF PORE PRESSURE FIXITY
  105 CONTINUE
      IF(IVAR.NE.NDIM1)GOTO 125
C
      DO 120 IL=1,NC
      IP=NXC(IL)
  120 IND(IL)=NCONN(IP,NE)
      NSDN=NC
      GOTO 132
C---------- IF DISPLACEMENT FIXITY
```

```
  125 DO 130 IL=1,NC
      IM=NXC(IL)
      IN=NXM(IL)
      IND(2*IL-1)=NCONN(IM,NE)
  130 IND(2*IL)=NCONN(IN,NE)
      NSDN=NFCD
  132 CONTINUE
C-----------------------------------------------------------------
C     LOOP ON ALL NODES ALONG FACE
C-----------------------------------------------------------------
  142 DO 180 KND=1,NSDN
      I=IND(KND)
      IF(NF.EQ.0)GO TO 158
C
      DO 150 J=1,NF
      IF(I.EQ.MF(J))GO TO 155
  150 CONTINUE
C
      GO TO 158
C-----------------------------------------------------------------
C     UPDATE EXISTING VALUES
C-----------------------------------------------------------------
  155 JF=J
      GO TO 160
C
  158 NF=NF+1
      IF(NF.LE.NFZ)GO TO 159
      WRITE(IW6,940)
      STOP
  159 JF=NF
  160 MF(JF)=I
      TF(IVAR,JF)=IFX
      DXYT(IVAR,JF)=FV(KND)
  180 CONTINUE
  200 CONTINUE
      RETURN
  900 FORMAT(/19X,16H......NODES.....,8X,6HFIXITY/
     1 1X,4HFACE,4X,7HELEMENT,3X,16H1    2    3    4,
     1 3X,3HDOF,3X,4HCODE,5X,4HVAL1,5X,4HVAL2,5X,4HVAL3,
     2 5X,4HVAL4,5X,4HVAL5,5X,4HVAL6,5X,4HVAL7,5X,4HVAL8/)
  910 FORMAT(1X,I3,4X,I5,3X,I4,1X,I4,1X,I4,1X,I4,4X,I2,3X,I3,3X,8F9.3)
  930 FORMAT(/1X,20H***** ERROR : FIXITY,I4,2X,8HIN LIST.,3X,
     1 7HELEMENT,I5,2X,29HDOES NOT HAVE FACE WITH NODES,4I5)
  940 FORMAT(/40H INCREASE SIZE OF ARRAYS MF, TF AND DXYT/
     1 1X,35HIN COMMON BLOCK FIX (ROUTINE FIXX3))
      END
      SUBROUTINE ALTER(IW6,IM,N)
C---------- ROUTINE TO SHIFT ARRAY FORWARD BY ONE PLACE
      DIMENSION IM(N)
C
      IF(N.LE.1)GOTO 100
      NM1=N-1
      IMT=IM(1)
C
      DO 10 K=1,NM1
   10 IM(K)=IM(K+1)
      IM(N)=IMT
      RETURN
  100 WRITE(IW6,900)N
```



```
  900 FORMAT(/1X,45HERROR * ARRAY CONTAINS LESS THAN OR EQUAL TO ,I5,2X,
     1 40HMEMBERS  (ROUTINE ALTER) CALLED BY FIXX3)
      RETURN
      END
```

Appendix F: Common block usage in CRISP

SUBROUTINE	GVAR	DATL	DATW	DEBUGS	DEVICE	ELINF	FIX	FLOW	LABEL	PARS	PRLDI	PRSLD	SAMP	OUT	PRECSN	COUNT	JACB	LOADS
ANGTH	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
ANS	.	X	X	.	X	X	X	X	.	X	X	X	X	.	.	.	.	.
BDATA1	.	X	X	.	.	X	.	.	.	.	.	.	X	.	.	.	.	.
CALDOF	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
CAMCDE	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
CHANGE	.	.	.	.	.	X	.	.	.	.	.	X	.	.	.	.	.	X
CONECT	.	.	.	X	.	X	.	.	.	.	.	.	.	.	.	.	.	.
CPW	.	X	X	.	X	X	X	X	.	X	X	X	.	.	.	.	.	.
CUREDG	.	.	.	.	.	X	.	.	.	.	.	.	.	.	.	.	.	.
DCAM	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
DCON	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
DETJCB	.	.	.	.	.	.	.	.	.	X	.	.	.	.	.	.	.	.
DETMIN	.	.	.	.	.	.	.	.	.	X	.	.	.	.	.	.	.	.
DISTLD	.	.	.	.	.	X	.	X	.	X	.	.	X	.	.	.	.	X
DLIN	.	.	.	.	.	.	.	.	.	X	.	.	.	.	.	.	.	.
DMCAM	.	.	.	.	.	.	.	.	.	.	X	.	.	.	.	.	.	.
EDGLD	.	.	.	.	.	X	.	.	.	X	.	.	.	.	.	.	.	.
EQLBM	.	.	.	.	.	.	.	X	.	X	.	.	.	.	.	.	X	.
EQLIB	.	X	X	.	.	X	.	.	.	X	.	X	.	.	.	.	.	.
EQLOD	.	.	.	.	.	.	.	.	.	X	.	.	.	.	.	X	.	.
EVCAM	.	.	.	.	.	.	.	.	.	X	.	.	.	.	.	.	.	.
FACTOR	.	.	.	.	.	X	X	.	.	X	.	.	.	.	.	.	.	.
FIXX	.	.	.	.	X	.	.	X	.	X	.	.	.	.	.	.	X	.
FORMB2	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.

SUBROUTINE	GVAR DATL	DATW DEBUGS	DEVICE ELINF	FIX FLOW	LABEL PARS	PRLDI PRSLD	SAMP OUT	PRECSN COUNT	JACB LOADS
FORMP	. . . .	X .	. . .	X	. .	. .	. .	X .
FRFXLD	. . . .	. .	X . .	X	. .	. .	. .	. .
FRONTZ	. . . .	X X	X . .	X	. .	. .	. .	. .
FRSLOT	. . . .	. X	. . .	.	. .	. .	. .	. .
GETEQN	. . . .	. .	. . .	.	. .	. .	. .	. .
GPOUT	. . . X	. X	. . .	.	. .	. .	. .	. .
INSITU	. . . .	X .	X . .	X	. X	X X	. .	. .
INSTRS	. X . .	. X	. . .	X	. .	. .	. .	. .
INTPLT	. . . .	. .	. . .	X	. .	. .	. .	. .
JPC	. . . .	. .	. X .	X	. .	. .	. .	. .
LODINC	. . . .	X .	. . .	.	X X	. .	X .	. .
LODLST	. . . .	. .	. . .	.	X .	. .	. .	. .
LSTFSG	. . . .	. .	. . .	X	. .	. .	. .	. .
LSTIFA	. . . .	. .	. . .	.	. .	. .	. .	. .
LSTIFF	. X X .	X X	. X .	X	. .	. .	X .	. .
MAIN	X . . .	. .	. . .	.	. .	. .	. .	. .
MAKENZ	. . . .	. X	. . .	.	. .	. .	. .	. .
MARKZ	. . . .	X X	. . X	.	. .	. .	. .	. .
MAST	. . . X	X .	. . X	X	. .	. X	X .	. .
MAXVAL	. . . .	. X	. . .	X	. .	. .	. .	. .
MIDPOR	. . . .	. X	. . .	X	. .	. .	. .	. .
MIDSID	. . . X	. X	. . .	.	. .	. .	. .	. .
MINIT	. . . .	X .	. . .	X	. .	. .	X .	. .

SUBROUTINE	GVAR DATL	DATW DEBUGS	DEVICE ELINF	FIX FLOW	LABEL PARS	PRLDI PRSLD	SAMP OUT	PRECSN COUNT	JACB LOADS
MLAPZ	. . . .	. X	. . .	.	. .	. .	. .	. .
NUMSH	. . . X	. X	. . .	.	. .	. .	. .	. .
PRINC	. . . .	. .	. . .	X	. .	. .	. .	. .
PRINTF	. . . .	. .	. . .	.	. .	. .	. .	. .
RDCOD	. . . .	. .	. . .	.	. .	. .	. .	. .
RDN	. . . .	. .	. . .	.	. .	. .	. .	. .
RDPROP	. . . .	X .	. . .	.	. .	. X	. .	. .
RDSTRS	. X . .	X X	. X .	X	. .	. .	. .	. .
REACT	. . . .	. .	. . .	.	. .	. .	. .	. .
RESTRN	. . . .	. .	X . .	.	. .	. .	. .	. .
RESTRT	. . . .	X .	X . .	.	. X	. X	. .	. .
SELF	. X X .	. X	. X .	X	. X	. .	. .	. .
SEL1	. . . .	. X	. . .	X	. .	. .	. .	. .
SETNP	. . . .	. .	. . .	.	. .	. .	. .	. .
SFR1	. . . .	. .	. . .	.	. .	. .	. .	. .
SFWZ	. . . X	X X	. . .	.	. .	. .	. .	. .
SHAPE	. . . .	. .	. . .	.	. .	. .	. .	. .
SHFNPP	. . . .	. .	. . .	.	. .	. .	. .	. .
SHFTIB	. . . X	. .	. . .	.	. .	. .	. .	. .
SIDES	. . . .	. .	. . .	.	. .	. .	. .	. .
SORTN2	. . . .	. .	. . .	.	. .	. .	. .	. .
SORT2	. . . .	. .	. . .	.	. .	. .	. .	. .
STORQ	. . . .	. .	. . .	.	. .	. .	. .	. .

SUBROUTINE	GVAR	DATL	DATW	DEBUGS	DEVICE	ELINF	FIX	FLOW	LABEL	PARS	PRLDI	PRSLD	SAMP	OUT	PRECSN	COUNT	JACB	LOADS
STRSEQ	.	.	X	.	.	.	.	X	.	.	.	.	.	.	.	.	.	.
UPARAL	.	.	.	.	X	.	.	.	.	.	.	.	.	.	X	.	.	.
UPOUT	.	X	X	.	X	X	X	X	.	X	.	X	.	X	.	X	X	.
UPOUT2	.	.	.	.	.	X	.	.	.	.	.	.	.	X	.	.	.	.
VARCAM	.	.	.	.	X	.	.	.	.	.	.	.	.	.	.	.	.	.
WRTN	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
ZEROSB	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.

Appendix G: Some notes on running CRISP

G.1 FILES USED

Logical unit no.	Description
1	In restarted analysis this is an unformatted magnetic tape or disk file which contains the results of previous analysis (otherwise set to a dummy file).
2	This is used if this analysis is to be subsequently restarted. It is an unformatted magnetic tape or disk file which contains (if applicable) the results of any previous analysis plus the results of the current analysis (set to a dummy file if thi option is not used).
5	The data input.
6	Printed output.
7	An unformatted scratch disk file.
8	Data output required to produce a plot of the finite element mesh. An unformatted disk file.

G.2 SOIL–STRUCTURE INTERACTION

The single-precision version of CRISP cannot be used for soil–structure interaction problems if the stiffness of the structure is several orders higher than the stiffness of the soil. If sensible answers are not obtained even with a reduced stiffness for the structure, the calculation needs to be carried out in double

precision. Numerical problems with CRISP may be evident by large equilibrium errors or by wildly varying pore pressures in undrained/consolidation analyses. Equilibrium errors in any analysis should be less than 5% (in most cases less than 1%). Therefore any error larger than 5% may indicate numerical problems.

G.3 UNDRAINED/CONSOLIDATION ANALYSIS

Undrained analysis

In undrained analysis an equivalent bulk modulus of water is added to the soil stiffness terms. In such an analysis, if the results appear to be meaningless or if the pore pressures generated fluctuate wildly from integration point to integration point then the analysis should be repeated with a lower bulk modulus of water. In most cases the true undrained behaviour can still be captured with a low value of the bulk modulus of water. However, too low a value would cause the behaviour to be partially drained. Sometimes the use of a finer mesh will improve the results. In some problems oscillations persist and in these cases our experience is that the centroidal values in the triangular elements are most reliable.

Consolidation analysis

If there is any sign of oscillations in the increment in which loading is applied (at the beginning of an analysis) then this is an indication of too small a time step. The analysis should be repeated with a larger time step for the load increment.

G.4 EQUILIBRIUM ERRORS AT *IN SITU* STAGE

If there are significant equilibrium errors at the *in situ* stage when *in situ* stresses have been specified it indicates only errors in input data. It means that the *in situ* stresses are not consistent with the applied boundary loads and displacement fixities. Make sure that either the displacement or the stress boundary condition is specified along the mesh boundary except for any free boundary (for example, the ground surface is free of any fixities or loads).

If the *in situ* stresses include the gravitational effects (as in an analysis of field situation) then PR(8,KM) should be the bulk unit weight of soil consistent with the vertical *in situ* stresses. Also set TGRAVI = 1 to record Q1.

If an element side which should have been restrained is left out inadvertently in the list of fixities it would also result in equilibrium errors.

For an analysis which does not include the effect of earth's gravity (i.e. triaxial test, where it is negligible and the vertical stresses are uniform everywhere at the start), TGRAVI = 0 and the bulk unit weight of soil need not be specified.

G.5 LARGE ANALYSIS

The capacity of the program to analyse a problem with a large number of elements is enhanced by simply increasing the size of array G in routine MAIN.

The size of some arrays which are fixed rather arbitrarily may also have to be increased (the relevant subroutines will issue a message when there is a need). The arrays are as follows.

(a) COMMON/PRSLD/PRESLD(10,100),LEDG(100),NDE1(100), NDE2(100),NLED.
(b) COMMON /PRLDI/ PRSLDI(10,100),LEDI(100),NDI1(100),NDI2(100), ILOD.
(c) COMMON/FIX/DXYT(4,200),MF(200),TF(4,200),NF.
(d) DIMENSION R(500),NDENO(500),NDIR(500) in routine REACT.

The first three common statements appear in a number of routines, and if it is necessary to increase the size of these arrays then all such occurrences have to be changed accordingly.

For example, if the sizes of arrays are increased to cater for up to 400 nodes with fixities, then

```
COMMON /FIX/ DXYT(4,200),MF(200),TF(4,200),NF
```

is replaced by

```
COMMON /FIX/ DXYT(4,400),MF(400),TF(4,400),NF
```

in all routines in which this statement appears. The list of routines in which this statement appears is in Appendix F. Further changes may be necessary, as indicated in Appendix C.

G.6 ANALYSING SMALL PROBLEMS

When testing the program with a small number of elements the core requirement could be reduced by decreasing the size of array G in routine MAIN to about 10000.

```
COMMON /GVAR/ G(10000)

LG = 10000
```

G.7 CONVERSION OF SINGLE-PRECISION VERSION TO DOUBLE PRECISION

Add in the following statement to all the routines:

```
IMPLICIT DOUBLE PRECISION (A-H,O-Z)
```

Convert all REAL constants to double precision (including the ones in routines BDATA1 and MINIT) by adding D0.

Set NP = 2 and comment out NP = 1 in routine MINIT.

```
      NP = 2
CC    NP = 1
```

Remove the CC in the following statements in the following routines.

REAL G – in routines MAST, MAIN, MINIT

REAL A – in routines ANS, CPW, INSITU, LODINC, UPARAL

References

Akin, J.E. & Pardue, R.M. (1975), Element resequencing for frontal solution, *Mathematics of finite elements*, Academic Press, London, 535–541.

Almeida, M.S.S. (1984), Stage constructed embankments on soft clays, PhD thesis, Cambridge University.

Almeida, M.S.S., Britto, A.M. & Parry R.H.G. (1986), Numerical modelling of a centrifuged embankment on soft clay, *Canadian Geotechnical Journal,* 23, 103–114.

Atkinson, J.H. & Bransby, P.L. (1978), *The mechanics of soils*, McGraw-Hill, London.

Bassett, R.H., Davies, M.C.R., Gunn, M.J. & Parry, R.H.G. (1981), Centrifugal models to evaluate numerical methods, Proc. 10th ICSMFE, Stockholm.

Biot, M.A. (1941), General theory of three dimensional consolidation, *J. Appl. Phys.*, **12**, 155–164.

Bishop, A.W. (1966), The strength of soils as engineering materials, (6th Rankine Lecture), *Géotechnique,* **16**, 91–128.

Bjerrum, L. (1973), Geotechnical problems involved in foundations of structures in the North Sea, *Géotechnique,* **23**, 319–358.

Booker, J.R. & Randolph, M.F. (1984), Consolidation of a cross-anisotropic soil medium, *Quarterly Journal of Mechanics and Applied Mathematics,* 37, 479–495.

Booker, J.R. & Small, J.C. (1975), An investigation of the stability of numerical solutions of Biot's equations of consolidation, *Int. J. Solids & Structures,* **11**, 907–911.

Burland, J.B. (1965), The yielding and dilation of clay, Correspondence, *Géotechnique*, **15**, 211–214.

Butterfield, R. (1979), A natural compression law for soils (an advance on e-log p'), *Géotechnique*, **29**, 469–480.

Carter, J.P. (1977), Finite deformation theory and its application to elastoplastic soils, PhD thesis, Sydney University.

Carter, J.P., Small, J.C. & Booker, J.R. (1977), A theory of finite elastic consolidation, *Int. J. Solids & Structures*, **13**, 467–478.

Casagrande, A. (1936), The determination of the pre-consolidation load and its practical significance, Proc. 1st Int. Conf. Soil Mech., Cambridge, Mass., **3**, 60–64.

Cook, R.D. (1981), *Concepts and applications of finite element analysis*, 2nd edition, Wiley, New York.

Coulomb, C.A. (1773), Essai sur une application des regles de maximis et minimis à quelques problems de statique, relatifs à l'architecture, *Mem. Math. Phys.* (divers Savans), Vol. 23. See also Heyman, J. (1972), *Coulomb's memoir on statics: an assay in the history of civil engineering*, Cambridge University Press.

Crandall, S.H. (1956), *Engineering analysis*, McGraw-Hill.

Dafalias, Y.F. & Herrman, L.R. (1982), Bounding surface formulation of soil plasticity, Chapter 10 in *Soil Mechanics – transient and cyclic loads* (ed. by Pande G.N. & Zienkiewicz O.C.), Wiley, 253–282.

Dalton, J.C.P. & Hawkins, P.G. (1982), Fields of stress – some measurements of the *in situ* stress in a meadow in the Cambridgeshire countryside, *Ground Engineering*, **15**, 15–23.

Davis, E.H. & Booker, J.R. (1973), The effect of increasing strength with depth on the bearing capacity of clays, *Géotechnique*, **23**, 551–563.

Day, A.C. (1972), *Fortran techniques*, Cambridge University Press, Cambridge.

Day, A.C. (1978), *Compatible fortran*, Cambridge University Press, Cambridge.

Dean, E.T.R. (1985), Development of Cam-clay – first progress report – the spread work function, Cambridge University Engineering Department.

Drucker, D.C. (1950), Some implications of work hardening and ideal plasticity, *Quart. Appl. Math.*, 7, 411–418.

Drucker, D.C. (1951), A more fundamental approach to stress–strain relations, Proc. 1st U.S. Nat. Cong. for Applied Mechanics, ASME, 487–491.

Drucker, D.C. (1954), Coulomb friction, plasticity, and limit loads, *J. Appl. Mech.*, **21**, 71–74.

Drucker, D.C., Gibson, R.E. & Henkel, D.J. (1957), Soil mechanics and work-hardening theories of plasticity, *A.S.C.E.*, **11**, 338–346.

Drucker, D.C. & Prager, W. (1952), Soil mechanics and plastic analysis or limit design, *Quart. Appl. Math.*, **10**, 157–165.

Galerkin, B.G. (1915), Series solution of some problems of elastic equilibrium of rods and plates, (Russian), *Vestn. Inzh. Tech.*, **19**, 897–908.

Harr, M.E. (1962), *Groundwater and seepage*, McGraw-Hill, New York.

Harr, M.E. (1966), *Foundations of theoretical soil mechanics*, McGraw-Hill, New York.

Henkel, D.J. (1956), Discussion on 'Earth movement affecting L.T.E. railway in deep cutting east of Uxbridge', Proc. I.C.E., Part II, **5**, 320–323, London.

Hill, R. (1950), *The mathematical theory of plasticity*, Oxford University Press.

Hinton, E. & Owen D.R.J. (1977) *Finite Element Programming*, Academic Press.

Irons, B.M. (1968), Roundoff criteria in direct stiffness solutions, *AIAA Journal*, **6**, 1308–1312.

Irons, B.M. (1970), A frontal solution program for finite element analysis, *Int. J. Num. Meth. Eng.*, **12**, 5–32.

Irons, B.M. & Ahmad, S. (1980), *Techniques of finite elements*, Ellis Horwood, Chichester.

Irons, B.M. & Shrive, N. (1983), *Finite element primer*, Ellis Horwood, Chichester.

Jaky, J. (1944), The coefficient of earth pressure at rest, *Magyar Mernok es Epitesz Egylet Kozlonye*.

Jardine, R.J., Symes, M.J. & Burland, J.B. (1984), The measurement of soil stiffness in the triaxial apparatus, *Géotechnique*, **34**, 323–340.

Katzan, H. (1978), *Fortran 77*, Van Nostrand Reinhold, New York.

Koiter, W.T. (1953), Stress–strain relations, uniqueness and variational theorems for elastic–plastic materials with a singular yield surface, *Q. Appl. Math.*, **11**, 350.

Ladd, C.C., Foott, R., Ishihara, K., Schlosser, F. & Poulos, H.G. (1977), Stress-deformation and strength characteristics: state of the art report. Proc. 9th ICSMFE, **2**, 421–494, Tokyo.

Larmouth, J. (1973a), Serious Fortran, *Software – Practice and Experience*, **6**, 87–107.

Larmouth J. (1973b), Serious Fortran – part 2, *Software – Practice and Experience*, **6**, 197–225.

Mair, R.J., Gunn, M.J. & O'Reilly, M.P. (1981), Ground movements around shallow tunnels in soft clay, Proc. 10th ICSMFE, Stockholm.

Maxwell, E.A. (1953), *An analytical calculus*, volume 3, Cambridge University Press, Cambridge.

Milovic, D.M. (1970), Contraintes et déplacements dans une couche élastique d'épaisseur limitée, produits par une foundation circulaire, *le Génie civil*. T.147, No. 5, 281–285.

Monro, D.M. (1982), *Fortran 77*, Edward Arnold, London.

Mroz, Z. & Norris, V.A. (1982), Elastoplastic and viscoplastic constitutive models for soils with application to cyclic loading, Chapter 8 in *Soil Mechanics – transient and cyclic loads* (ed. by Pande G.N. & Zienkiewicz O.C.), Wiley, 173–217.

Naylor, D.J. (1975), Non-linear finite element models for soil, PhD thesis, University College of Swansea.

Naylor, D.J. (1985), A continuous plasticity version of the critical state model, *Int. Jnl. Num. Meth. in Eng.*, **21**, 1187–1204.

Ohta, H. & Wroth, C.P. (1976), Anisotropy and stress reorientation in clay under load, Proc. 2nd Int. Conf. Num. Meth. in Geomechanics, ASCE, 319–328.

Owen, D.R.J. & Hinton, E. (1980), *Finite elements in plasticity: theory and practice*, Pineridge Press, Swansea.

Palmer, A.C. (1973), Contribution to discussion, Proc. Symp. on the Role of Plasticity in Soil Mechanics, Cambridge, 44–45.

Parry, R.H.G. (1970), Overconsolidation in soft clay deposits, *Géotechnique*, **20**, 442–446.

Parry, R.H.G. (1982), Private communication.

Pender, M.J. (1982), A model for the cyclic loading of overconsolidated soil, Chapter 11 in *Soil Mechanics – transient and cyclic loads* (ed. by Pande, G.N. & Zienkiewicz O.C.), Wiley, 283–331.

Potts, D.M. (1981) Private communication.

Poulos, H.G. (1967), Stresses and displacements in an elastic layer underlain by a rough rigid base, *Géotechnique*, **17**, 378–410.

Poulos, H.G. & Davis, E.H. (1974), *Elastic solutions for soil and rock mechanics*, Wiley, New York.

Pyrah, I.C. (1980), The solution of two dimensional consolidation problems, Proc. symposium on Computer applications to geotechnical problems in highway engineering, P.M. Geotechnical Analysis Ltd, Cambridge.

Razzaque, A. (1980), Automatic reduction of frontwidth for finite element analysis, *Int. J. Num. Meth. Eng.*, **15**, 1315–1324.

Roscoe, K.H. & Burland, J.B. (1968), On the generalised stress–strain behaviour of 'wet clay', *Engineering plasticity*, Cambridge University Press.

Roscoe, K.H. & Schofield, A.N. (1963), Mechanical behaviour of an idealised 'wet clay', Proc. 2nd European Conf. Soil Mech., 47–54.

Roscoe, K.H., Schofield, A.N. & Thurairajah, A. (1963), Yielding of clays in states wetter than critical, *Géotechnique*, **13**, 211–240.

Roscoe, K.H., Schofield, A.N. & Wroth, C.P. (1958), On the yielding of soils, *Géotechnique*, 8, 22–53.

Ryder, B.G. (1974), The PFORT verifier, *Software – practice and experience*, **6**, 359–377.

Schofield, A.N. & Wroth, C.P. (1968), *Critical state soil mechanics*, McGraw-Hill, London.

Seneviratne, H.N. & Gunn, M.J. (1985), Predicted and observed time-dependent deformations around shallow model tunnels in soft clay, Proc. 5th Int. Conf. Num. Meth. in Geomechanics, Nagoya, Japan.

Skempton, A.W. (1985), Residual strength of clays in landslides, folded strata and the laboratory, *Géotechnique*, **35**, 3–18.

Sloan, S.W. & Randolph, M.F. (1982), Numerical prediction of collapse loads using finite element methods, *Int. J. Num. Anal. Meth. Geomech.*, **6**, 47–76.

Sloan, S.W. & Randolph, M.F. (1983), Automatic element reordering for finite element analysis with frontal solution schemes, *Int. J. Num. Meth. Eng.*, **19**, 1153–1181.

Taylor, D.W. (1948), *Fundamentals of soil mechanics*, Wiley, New York.

Terzaghi, K. & Frohlich, O.K. (1936), *Theory of settlement of clay layers*, Deutike, Leipzig.

Timoshenko, S.P. & Goodier, J.N. (1970), *Theory of elasticity*, 3rd edition, McGraw-Hill, New York.

Turner, M. J., Clough, R. W., Martin, H. V. & Topp, L. J. (1956), Stiffness and deflection analysis of complex structures, *J. Aero. Sci.*, **23**, 805–823.

Vermeer, P.A. & Verrujit, A. (1981), An accuracy condition for consolidation by finite elements, *Int. J. Num. Anal. Meth. Geomech.*, **5**, 1–14.

Wood, D.M. (1984), On stress parameters, *Géotechnique*, **34**, 282–287.

Wroth, C.P. (1971), Some aspects of the elastic behaviour of overconsolidated clay, Proc. Roscoe memorial symposium, Foulis, 347–361.

Wroth, C.P. (1975), *In-situ* measurement of initial stresses and deformation characteristics, Proc. of the Speciality Conf. in In-situ Measurement of Soil Properties, ASCE, Rayleigh, North Carolina, June, 181–230.

Wroth, C.P. (1984), The interpretation of *in situ* soil tests (24th Rankine Lecture), *Géotechnique*, **34**, 449–489.

Zienkiewicz, O.C. (1967), *The finite element method in structural and continuum mechanics*, McGraw-Hill.

Zienkiewicz, O.C. (1971), *The finite element method in engineering science*, McGraw-Hill.

Zienkiewicz, O.C. (1977), *The finite element method*, McGraw-Hill, London.

Zienkiewicz, O.C., Humpheson, C. & Lewis, R.W. (1975), Associated and non-associated visco-plasticity and plasticity in soil mechanics, *Géotechnique*, **25**, 671–689.

Zytynski, M., Randolph, M.F., Nova, R. & Wroth, C.P. (1978), Short communication: modelling the unloading–reloading behaviour of soils, *Int. J. Num. Anal. Meth. Geomech.*, 2, 87–94.

Subject Index

Program Index

Routines

Variables/Arrays
(page numbers given below refer to explanations as to the purpose of each variable/array)

Author Index