NOTICE

The quality of this microform is heavily dependent upon the quality of the original thesis submitted for microfilming. Every effort has been made to ensure the highest quality of reproduction possible.

If pages are missing, contact the university which granted the degree.

Some pages may have indistinct print especially if the original pages were typed with a poor typewriter ribbon or if the university sent us an inferior photocopy.

Reproduction in full or in part of this microform is governed by the Canadian Copyright Act, R.S.C. 1970, c. C-30, and subsequent amendments.

AVIS

La qualité de cette microforme dépend grandement de la qualité de la thèse soumise au microfilmage. Nous avons tout fait pour assurer une qualité supérieure de reproduction.

S'il manque des pages, veuillez communiquer avec l'université qui a conféré le grade.

La qualité d'impression de certaines pages peut laisser à désirer, surtout si les pages originales ont été dactylographiées à l'aide d'un ruban usé ou si l'université nous a fait parvenir une photocopie de qualité inférieure.

La reproduction, même partielle, de cette microforme est soumise à la Loi canadienne sur le droit d'auteur, SRC 1970, c. C-30, et ses amendements subséquents.

Canada

Pavement Design Using Finite Element Analysis of Subgrades

W.J. Bobesiuk P. Eng.

A Thesis

in

the Department

of

Civil Engineering

Presented in Partial Fulfilment of the Requirements

for the Degree of Master of Engineering at

Concordia University

Montreal, Quebec

November 1992

ISBN 0-315-84620-8

Canada

Abstract

Finite Analysis of Pavement Subgrades

Walter Jerry Bobesiuk P. Eng.

In order to adequately design a pavement system, attention must be paid to the subgrade. This thesis attempts to formulate such a design method based on the expected traffic volumes, the subgrade material properties, and material behaviour.

The methodology described in this thesis involves two main phases. The first phase develops a relationship between the vertical strain produced at the top of the subgrade and a combination of five axle loads and three base thicknesses. This relationship is based on constitutive equations and the finite element method. These fifteen relationships shown as strain contours are for a specific material, and can be used to find the critical strain for a given axle load and base thickness.

The second phase of this thesis deals with the development of a pavement design method using vertical strain contour charts. These strain contour charts have been developed using constitutive equations and the finite element method for different types of soil parameters and behaviors. This method then utilizes equations from the Asphalt Institute design method to determine base thickness and the life of the

subgrade. Numerical examples of the design method have been provided.

A set of computer programs have been developed for the finite element analysis. These programs have been unified under a menu program (FINITE) and are capable of finding the stress-strain distributions in the subgrade.

Acknowledgements

I wish to express my sincere gratitude to Dr. B. Ashtakala for supervising me through all stages of this thesis. His guidance and patience are greatly appreciated. I am thankful to him for funding this research project through his NSERC operating grant.

I would like to thank Jean Miller and Bryan Ekers for allowing the use of their computers for this research project.

Finally, I would like to convey my love and devotion to my wife and family to whom this thesis is dedicated, for their patience and support throughout this work.

Table of Contents

List of Figures

List of Tables

List of Symbols

α_1	2 x shear modulus
α_2	bulk modulus - $\alpha_1/3$
c	cohesion
C	volume of voids in asphalt
C_u	undrained shear strength
CBR	California Bearing Ratio
δ	Kronecker delta
ε	strain (ε_{ij})
ε_c	vertical compressive strain
ε_t	horizontal tensile strain
E	modulus of elasticity (Young's modulus)
EAL	equivalent axle load
f_1	load equivalency factor
G	shear modulus
I_1	first invariant of strain tensor
K	bulk modulus
L_r	load carried by standard crushed rockbase at 0.1 inch piston penetration
L_s	load carried by specimen at 0.1 inch piston penetration
N	number of load repetitions to subgrade failure

N_f	number of load repetitions to fatigue cracking of asphalt
N_i	number of vehicles in each weight class
N_y	load repetitions per year
N_1	number of load repetitions to untreated subgrade failure
N_2	number of load repetitions to treated subgrade failure
ϕ	angle of friction
ϕ_u	shear strength for undrained clay
ψ	soil dilation angle
ν	Poisson's ratio
p	serviceability index
S_1, S_2, S_3	deviatoric stresses
σ	stress (σ_{ij})
σ_i	increment of shear stress
σ_v	Von Mises shear stress
σ_y	yield stress
S	design life
SN	structure number
τ_{oct}	octahedral shear stress
τ_{xz}	xz shear stress

Chapter 1

Introduction

1.1 Flexible Pavement Structure

The flexible pavement structure is composed of an asphalt wearing surface, a granular base, and a subgrade layer which for design purposes is assumed to extend to an infinite depth. In the case of stabilized soils, the base layer may be omitted. Traffic loads are applied to the surface and are transmitted throughout the three courses. The axle load is distributed on the asphalt as a pressure from the contact area of the tires. This stress is then distributed through the asphalt to the base and finally through the subgrade. The actual load does not diminish or disappear. It is merely distributed over a greater area as the depth increases.

1.2 Constitutive Equations and Finite Element Analysis

The stresses and strains produced in the subgrade are dependent on the load applied and the material parameters such as modulus of elasticity. This behaviour is governed by a stress strain relationship or constitutive law. The relationship can be linear elastic (obeying Hooke's law) or elasto-plastic (obeying Von Mises or Mohr Coulomb theories).

The stress-strain distribution throughout the structure can be defined by the finite element method. Until the advent of computer technology, such calculations would be difficult to perform, and as a result engineers were forced to

utilize numerous assumptions which made the results inaccurate. However, now that accurate mapping of the stress/strain distribution is possible, the question of how to apply these results to the design process arises.

1.3 Objectives of the Study

The primary objective of this study is to provide a better understanding of the fundamental relationship between the vertical compressive strain, henceforth called the vertical strain, produced at the surface of the subgrade and the combination of traffic loads and material properties. Beyond the primary objective, the study is specifically aimed at establishing the following:

1. To develop strain contour graphs based on base thicknesses and axle loads. Each of these graphs will be for a specific material with distinct properties.
2. To develop a series of strain contour charts which can be used to design a required base thickness given traffic volumes, soil behaviour, and parameters.

1.4 Structure of the Thesis

Chapter 2 of this thesis outlines the literature review regarding the uses of finite element analysis in geomechanics and highway engineering. It also describes numerous soil properties that are necessary for the study.

Chapter 3 describes the development of constitutive equations and their integration into finite element analysis.

The structure of the three types of analysis (linear elastic, Von Mises, and Mohr Coulomb) are also discussed.

Chapter 4 describes the pavement model and required finite element meshes used in the study. The determination of the soil parameters is also discussed.

Chapter 5 arranges the results of the analysis in a matrix form to show the fundamental relationship between the vertical strain produced at the surface of the subgrade and the combination of axle load and base thickness. Contours are drawn and can be used to find the combination of base thickness and axle load that can produce a given magnitude of vertical strain.

Chapter 6 expands the model to include a broader range of soil parameters. Strain contour charts are created from the large body of vertical strains generated. These charts are incorporated into a design method based on traffic volumes, soil properties, and behaviour.

Chapter 7 illustrates the comprehensive computer work used in this research. The algorithm (FINITE) is discussed and instructions on its use are shown.

Chapter 8 provides conclusions and suggestions for further research.

Appendices A, B, C, and D contain respectively: strain contour charts, sample output, flowcharts, and computer programs.

Chapter 2

Literature Review

2.1 Introduction

The use of finite element analysis for geotechnical applications has been discussed in numerous sources. The object thus far has been either to perform a comparison between the predicted and actual stresses and strains or to calculate the overall effects on the soil due to a given load. This geotechnical analysis can be applied to the pavement structure (such as the subgrade) to find the vertical strain.

2.2 Development of Constitutive Equations

The use of constitutive laws has increased significantly with the development of modern computers. These laws in turn have been applied to finite element, finite difference, and boundary integral methods. Desai and Siriwardane (1983 and 1984) use constitutive laws and together with the finite element analysis for applications to geotechnical problems. They discuss the development of constitutive equations from the generalized Hooke's law expressed by:

$$\sigma = E\varepsilon \qquad (2.1)$$

where σ = stress

ε = strain

E = modulus of elasticity

to linear elastic models expressed by:

$$[\sigma] = [D][\varepsilon] \qquad (2.2)$$

where $[\sigma]$ = stress matrix

$[\varepsilon]$ = strain matrix

$[D]$ = elastic stress-strain matrix

and elasto plastic models expressed by:

$$[d\sigma] = ([D] - [PL])[d\varepsilon] \qquad (2.3)$$

where $[d\sigma]$ = stress increment matrix

$[d\varepsilon]$ = strain increment matrix

$[PL]$ = stress dependent plastic component of the stress-strain relationship

However, these authors state that without adequate soil data the analysis becomes a mathematical exercise. Therefore they discuss the tests based on various types of triaxial set-up to derive the proper parameters. They also provide much of the soil data in their examples. Many of these parameters are used in the computer trials in Chapter 5. These values are also checked against parameters listed in Bowles (1982 and 1984).

While the main focus is on foundation soils, this method can also be applied to highway pavements, where the pavement is assumed to behave as a strip footing.

This application of the finite element analysis to railway embankments is also explored in Desai and Siriwardane (1984). In this case, each layer requires a different model type (linear elastic, cap, cam clay, and variable moduli) and each layer is analyzed separately. The predicted results compare well with the observed values. This approach is used in the analysis of the multi-layered highway pavement structures.

Ashtakala and Poorooshasb (1989), use the concepts of constitutive equations and finite analysis to solve the problem of tensile cracking in pavements. The authors derive a constitutive relationship for the subgrade soil and determine stresses and strains in it using a finite element program called CONPAVE. The results of CONPAVE show the distribution of porewater pressure and ground movement. This approach can be applied to pavement subgrades to plot the distribution of stresses and strains in the continuum as well as the critical stresses and strains for a given load case.

Marchionna et al (1987), also apply finite element analysis to a pavement problem. The authors examine a four layer pavement structure where the asphalt wearing surface is assumed to behave linear elastically while the remaining layers are non linear. Their model evaluates the allowable load repetitions related to different cracking stages of the pavement surface. Their input data is based on falling weight deflectometer tests. Using the tensile strains of the bottom

of the asphalt, they derive the stress-strain distribution. They also evaluate the residual life of the pavement based on the initial asphalt thickness affected by cracking. Thus the fatigue distress model is used to calculate the remaining life of the uncracked portion of the pavement. They also create a diagram where the percentage of residual life is calculated to the percentage of cracked surface in order to predict the life of the overlay before resurfacing must be done. They utilize the method to forecast changes in pavement conditions through time.

2.3 Development of the Finite Element Method

Smith (1982), and Smith and Griffiths (1988), discuss the construction of finite element programs specifically for geotechnical applications. Beginning with the constitutive laws, the authors show how they are placed within the programs. They utilize applications to linear elastic solids as well as Von Mises elasto-plasticity and Mohr Coulomb plasticity. The sample programs provided are set up in standard IBM mainframe building blocks making them easy to modify, if necessary.

Weaver and Johnston (1984), provide additional explanations and intermediate derivations to the equations provided by Smith (1982).

As stated in Desai and Siriwardane (1984), the analysis must be realistic or else the work becomes an exercise in theoretical mathematics. There are numerous tests

that compare the results of finite element analysis with observed values. For example, Heinrichs et al (1989), compare finite element results with real pavement models.

A line of research by Sweere et al (1987), focuses on cyclic loading triaxial testing of unbound base course materials. A full scale test pavement with built-in transducers is constructed to compare the results. Saraf et al (1987), try to determine the effect of tire contact pressure distribution on flexible pavements using finite element programs (ELSYM5 and TEXGAB-3D). Their results indicate that the critical tensile strain is overestimated while the critical compressive strain is underestimated. It has also been found that changes in tire pressure have negligible effects on the critical compressive strain.

The results of various comparisons yield the conclusion that the finite element values compare well with observed effects.

2.4 The Asphalt Institute Method

Oglesby and Hicks (1982), discuss the development of the Asphalt Institute design method. The engineers of the Asphalt Institute acted in an advisory capacity during the AASHO road tests and later took the results to perform an independent analysis. They produced a revised method of design of flexible pavements which were made available in 1963. Ritter and Paquette (1967), state that since this method was designed for use under a wide range of conditions,

it is not considered as accurate as methods developed by local agencies within their own areas.

The Asphalt Institute method uses the laws of mechanics and applies elastic layer theory to pavement design. The method requires the modulus of elasticity (E) and Poisson's ratio (ν) for each material layer. The finite element method has similar requirements but it also includes additional soil parameters such as cohesion, angle of friction, angle of dilation, and yield stress. Thus, while the Asphalt Institute method in effect only considers linear elastic material, the use of finite elements adds the new dimension of soil behaviour to the method.

The traffic load is expressed in terms of 18,000 lbs. (80 kN) single axle load applied on two sets of dual tires.

There are two limiting strains considered by the Asphalt Institute method:

1. The horizontal tensile strain ε_t on the underside of the asphalt layer. In the case of the finite element model for the base, the largest tensile strain would be located on the surface of the mesh directly under the load. This stress can then be inserted into the following equation:

$$N_f = 18.4(C)(4.32\times10^{-3}\varepsilon_t^{-3.29}E^{-0.854}) \qquad (2.4)$$

where N_f represents the number of loads to fatigue cracking, C is the volume of voids in the asphalt, ε_t is the tensile strain, and E represents the modulus of elasticity (Oglesby and Hicks, 1982). However, Ullidtz (1987), notes that this formula can change from agency to agency. For example, in Italy, Autostrade uses:

$$N_f = (\varepsilon_t/47.4\times10^{-4})^{-3.47} \qquad (2.5)$$

which will yield different results.

2. The vertical strain ε_c which is located at the surface of the subgrade directly under the wheel load. The values of ε_c can be inserted into the equation below to find the number of load repetitions to failure (N). The equation to estimate pavement life is given by:

$$N = 1.36\times10^{-9}(\varepsilon_c)^{-4.48} \qquad (2.6)$$

Chapter 3

Theoretical Background

3.1 Constitutive Equations

A constitutive model is defined as a mathematical equation which describes the stress-strain relationship of a material (Desai and Siriwardane, 1984). This model can be used to describe a material based on a Cauchy's linear elastic type of the lowest order equivalent to the generalized Hooke's law) or one based on a elasto-plastic behaviour with an axisymmetric stress-strain relationship as in the case of Von Mises or Mohr Coulomb behaviour. In both cases, a plane strain model is assumed.

3.1.1 Linear Elastic Relationship

The stress (σ_{ij}) and strain (ε_{ij}) are governed by the following equation when initial stress is absent:

$$\sigma_{ij} = \alpha_1 I_1 \delta_{ij} + \alpha_2 \varepsilon_{ij} \qquad (3.1)$$

where: α_1 = 2G where G is the shear modulus

α_2 = K - (2G/3) where K is the bulk modulus

I_1 = first invariant of strain tensor denoting volumetric strain ($\varepsilon_{11} + \varepsilon_{22} + \varepsilon_{33}$)

δ_{ij} = Kronecker delta (where δ_{ij} = 1 for i=j and zero otherwise)

then by substitution the following equation is derived:

$$\sigma_{ij} = KI_1\delta_{ij} + 2G(\varepsilon_{ij} - (I_1/3)\delta_{ij}) \qquad (3.2)$$

For a plane strain model, strains occur in the x and y directions only. Therefore, in this situation, the first invariant becomes:

$$I_1 = \varepsilon_{11} + \varepsilon_{22} \tag{3.3}$$

and by substitution the following equations are defined:

$$\sigma_{11} = \left(\frac{3K + 4G}{3}\right)\varepsilon_{11} + \left(K - \frac{2G}{3}\right)\varepsilon_{22} \tag{3.4}$$

$$\sigma_{22} = \left(\frac{3K + 4G}{3}\right)\varepsilon_{22} + \left(K - \frac{2G}{3}\right)\varepsilon_{11} \tag{3.5}$$

$$\sigma_{33} = \left(K - \frac{2G}{3}\right)\varepsilon_{11} + \left(K - \frac{2G}{3}\right)\varepsilon_{22} \tag{3.6}$$

$$\sigma_{12} = 2G\varepsilon_{12} \tag{3.7}$$

$$\sigma_{23} = \sigma_{13} = 0 \tag{3.8}$$

or in the matrix form:

$$\begin{bmatrix} \sigma_{11} \\ \sigma_{22} \\ \sigma_{12} \end{bmatrix} = \begin{bmatrix} \frac{3K + 4G}{3} & \frac{3K - 2G}{3} & 0 \\ \frac{3K - 2G}{3} & \frac{3K + 4G}{3} & 0 \\ 0 & 0 & 2G \end{bmatrix} \begin{bmatrix} \varepsilon_{11} \\ \varepsilon_{22} \\ \varepsilon_{12} \end{bmatrix} \tag{3.9}$$

In the case or uniaxial stress, $\sigma_{22} = \sigma_{33} = 0$ and

$$\sigma_{11} = \left(\frac{9KG}{3K + G}\right)\varepsilon_{11} = E\varepsilon_{11} \tag{3.10}$$

and

$$\varepsilon_{22} = \varepsilon_{33} = -\left(\frac{3K - 2G}{6K + 2G}\right)\varepsilon_{11} = \nu\varepsilon_{11} \qquad (3.11)$$

where E and ν are known as the modulus of elasticity and Poisson's ratio respectively. The modulus of elasticity can be expressed as the ratio of the stress divided by its corresponding strain. Hence it is also known as the stress-strain modulus (Bowles, 1984). The Poisson's ratio is the proportion of vertical to horizontal stresses in the material. These two parameters can be expressed as:

$$E = \frac{9KG}{3K + G} \qquad (3.12)$$

$$\nu = \frac{3K - 2G}{6K + 2G} \qquad (3.13)$$

and implies that a uniaxial state of stress causes strains in the axial and lateral directions. Thus by substituting the modulus of elasticity and Poisson's ratio into the stress-strain matrix, it can be rewritten as:

$$\begin{bmatrix} \sigma_{11} \\ \sigma_{22} \\ \sigma_{33} \end{bmatrix} = \frac{E(1-\nu)}{(1+\nu)(1-2\nu)} \begin{bmatrix} 1 & \frac{\nu}{1-\nu} & 0 \\ \frac{\nu}{1-\nu} & 1 & 0 \\ 0 & 0 & \frac{1-2\nu}{2(1-\nu)} \end{bmatrix} \begin{bmatrix} \varepsilon_{11} \\ \varepsilon_{22} \\ \varepsilon_{12} \end{bmatrix} \qquad (3.14)$$

This matrix is the constitutive equation for the linear elastic material under a plane strain condition. It can be restated as:

$$\sigma = D\varepsilon \tag{3.15}$$

where D is the constitutive matrix (Desai and Siriwardane, 1984).

3.1.2 Elasto-Plastic Relationship

A material will remain elastic until it reaches a yield point where it begins to deform permanently. At this point, the material begins to produce plastic strains and therefore, is said to behave elasto-plastically.

Elasto plastic soils cause greater analytical problems than the linear elastic routines because solutions are almost always obtained from a step by step linearization process where the increments are small enough for practical purposes (Smith, 1982).

In the case of the two elasto plastic programs used, only the material nonlinearity will be considered. Both programs, as already stated, use incremental solution procedures. As shown in Figure 3.1, repeated trials starting fresh each time yield different load deformation slopes. These slopes change progressively (Newton-Ralphson method) where the stiffness matrix changes continually (Smith, 1982).

The main physical feature of the nonlinearity of

elasto-plastic soils is the irrecoverability of strain. In the theory of plasticity, a yield surface separates stress states which in turn give elastic (recoverable) and plastic (irrecoverable) strains. While this yield surface may move kinematically during cyclical loading, this movement is kept to zero in these cases.

The Von Mises soil possesses a failure surface as shown in Figure 3.2.

The axis of the Von Mises yield cylinder lies on the space diagonal implying that the mean normal stress has $\sigma_1+\sigma_2+\sigma_3$ no effect on yield. The material will yield only when shear stress reaches a critical value.

The sample material is a simple undrained clay with a constant shear strength ($\phi_u = 0$). The undrained shear strength (C_u) is in fact multiplied by two to get the yield stress.

In this case the material flows in an associated manner, that is the vector of plastic strain increment is normal to the yield surface. This type of flow yields mathematical simplifications and with a Von Mises surface serves to describe the plastic behaviour of undrained clays. The direction of the plastic strain vector satisfies:

$d\varepsilon_1 + d\varepsilon_2 + d\varepsilon_3 = 0$ or no volume change.

The Mohr Coulomb materials do not yield and flow according to a no volume change rule. In fact, for the vast majority of soils it is extremely difficult to define yield

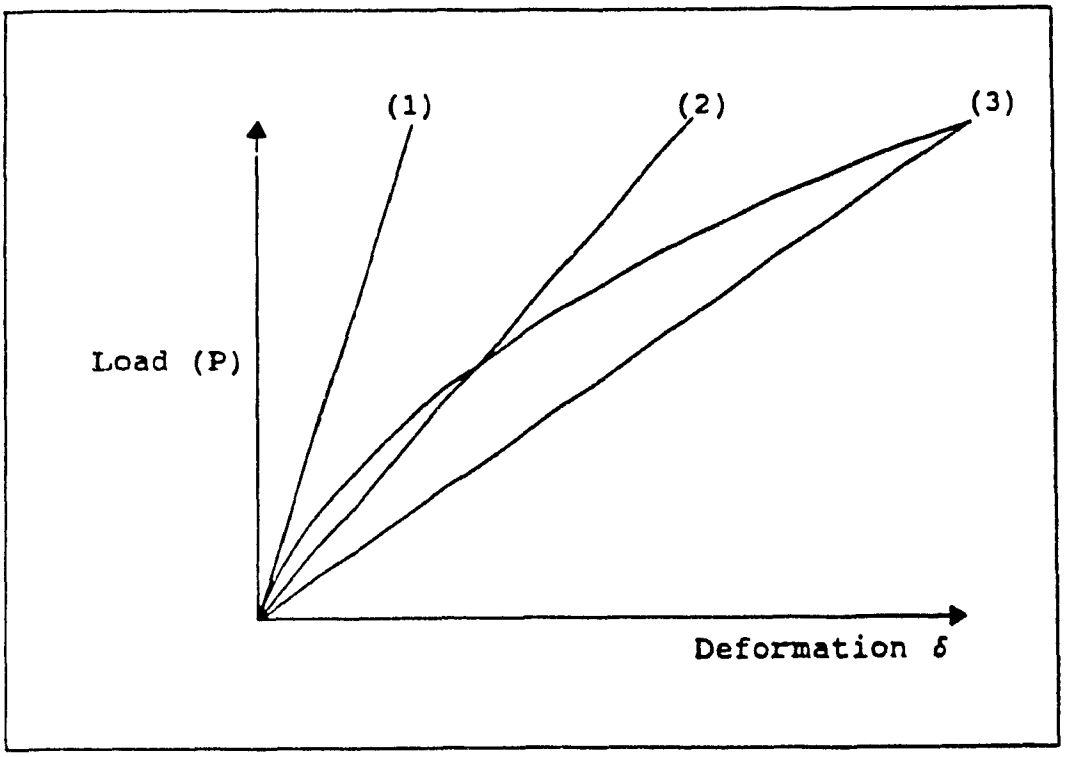


Figure 3.1: Nonlinear Iteration Process

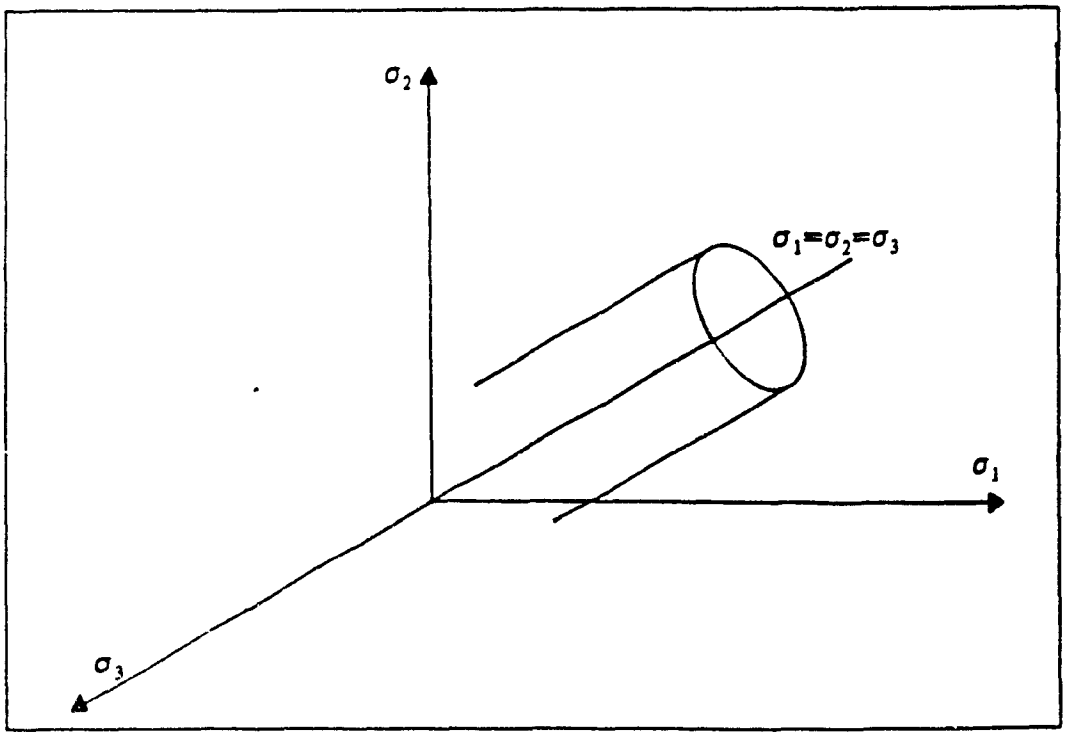


Figure 3.2: Von Mises Failure Cylinder

stress as soils can flow plastically under almost all applied loads. In many cases the detailed undrained volumechanges that occur before failure are not as important as the dependence of soil failures on mean normal stress is adequately represented. The Mohr Coulomb criterion (found from laboratory tests) that the ultimate stresses born by soils in terms of effective stresses fall on the hexagonal surface shown in Figure 3.3 (Smith, 1982).

The cone does not expand linearly for very large values of mean normal stress but does so adequately for moderate loads. Elastic behaviour occurs within the surface while the surface itself indicates plastic flow. The plastic flow can be a crude approximation but two limiting assumptions are used:

1. flow is assumed to be associated where angle of dilation (ψ) is equal to soil friction angle (ϕ) with the Mohr Coulomb surface which implies high volumetric expansion.
2. flow is assumed to be non associated ($\psi = 0$) with a zero plastic volume change.

As stated previously, the second assumption will be utilized. While it implies that the critical state is coincident with the Mohr Coulomb surface (which is not the case), the results are acceptable. In any case, the degree of association attributed to the flow does not have a great deal of influence on ultimate loads.

The stress-strain relationship for both materials can be expressed by:

$$d\sigma = DPL \times d\varepsilon \qquad (3.16)$$

where $d\sigma$ = change in stress

$d\varepsilon$ = change in strain

DPL = variable elasto-plastic matrix

This matrix can be considered to act in the same way as the constitutive matrix in the linear elastic case. However, in this situation, the matrix is a (4 x 4) array due to axisymmetric loading. This matrix is used by both the Von Mises and Mohr Coulomb programs.

Axisymmetric loading is assumed in the elasto-plastic cases because such behaviour is not truly two-dimensional. The elastic and plastic components tend to balance in the direction of the zero total strain (Smith, 1982). Therefore, the axisymmetric assumption is more realistic by having four independent components of strain rather than three.

The matrix DPL can be expressed as a combination of the elastic constitutive matrix and a stress dependent plastic component. The question is:

$$DPL = D - PL \qquad (3.17)$$

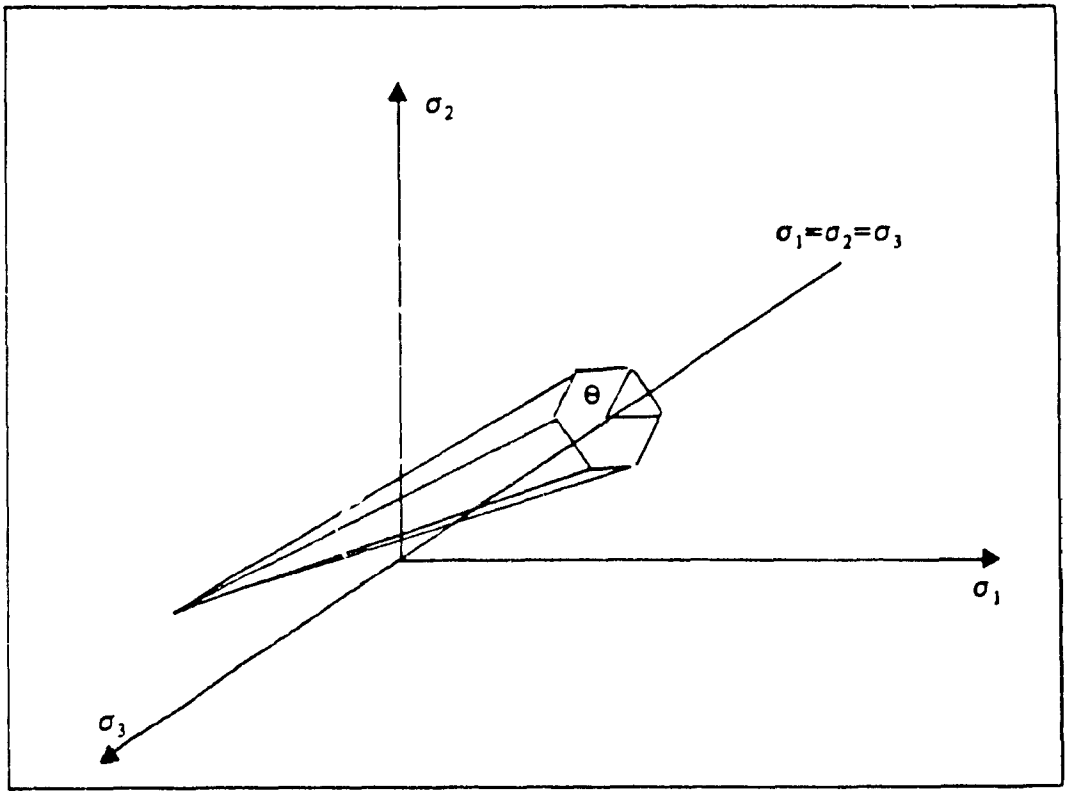


Figure 3.3: Mohr Coulomb Failure Cone

where D = the linear elastic component

PL = the stress dependent plastic component

The matrix can be defined as:

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

while the plastic stress-strain matrix is expressed by:

$$\frac{3E}{2\sigma_i^2(1+\nu)} \begin{bmatrix} S_r^2 & & & \text{Symmetrical} \\ S_z S_r & S_z^2 & & \\ \tau_{rz} S_r & \tau_{rz} S_z & \tau_{rz}^2 & \\ S_\theta S_r & S_\theta S_z & S_\theta \tau_{rz} & S_\theta^2 \end{bmatrix}$$

where S_r = deviatoric stress $(2\sigma_r-\sigma_\theta-\sigma_z)/3$

S_z = deviatoric stress $(2\sigma_\theta-\sigma_r-\sigma_z)/3$

S_θ = deviatoric stress $(2\sigma_z-\sigma_r-\sigma_\theta)/3$

τ_{rz} = xz shear stress

σ_i = Von Mises shear stress

Since the material may be elastic for part of the loading and elasto-plastic for the remainder, a factor of plasticity is entered into the equation (Smith, 1982). The stress-strain matrix is then expressed as:

$$DPL = D - (FAC \times PL) \qquad (3.18)$$

where $FAC = (\sigma_e - \sigma_y) / (\sigma_e - \sigma_v)$

σ_e = increment of shear stress

σ_y = yield stress

σ_v = Von Mises shear stress

These equations are then utilized in finite element analysis to define the material behaviour.

3.2 The Finite Element Method

Finite element analysis is a very efficient means of solving linear elastic or elasto-plastic problems. It is easy to account for boundary conditions and various soil effects. Unlike the classical approach, the finite element method can satisfy the numerous differential equations of equilibrium, stress strain relationships, and compatibility conditions at every point on the continuum including those at the boundaries without assumptions or truncations. It is also more versatile than the finite difference method because it does not require a different equation formulation for boundaries and works better with elements of different sizes (Weaver and Johnston (1984).

The finite element method analyzes a descretized continuum which in the case of pavements represents the base and subgrade layers. The continuum is descretized by dividing it into a finite number of elements of a simple yet arbitrary shape (Weaver and Johnston, 1984). Two popular shapes for analysis are triangles and rectangles. Input at the nodes of each element is related to the output through a matrix of partial differential equations. The matrices of each element (local Matrix) are then summed up into a global matrix representing the entire continuum to be studied.

Finite element analysis, due to the large number of equations to be processed, could not be used until the advent of the digital computer. The finite element approach can be summarized as:

1. divide the continuum into finite elements of a given shape (in the pavement case, rectangles)
2. select nodes where compatibility and equilibrium are enforced and define the boundary conditions at the nodes
3. satisfy the stress-strain relationship at each element
4. sum up the local element relationships into a global matrix for the entire continuum
5. solve the equilibrium equations for nodal displacements
6. calculate the stresses and strains at all

element nodes for elastic soils and at element centres for elasto-plastic soils

The three computer programs, linear elastic, Von Mises, and Mohr Coulomb, use the approach listed above with similar program blocks (with slight variations for soil behaviour). Each program assumes a plane strain model and can be divided into sub-programs which perform, input and initialization, element stiffness integration and assembly, and solution/recovery of stresses. Detailed flowcharts can be found in Appendix C while the program lists are in Appendix D.

3.2.1 Linear Elastic Model

The linear elastic model deals with a plane strain (stress) analysis using four noded rectangular elements. Based on soil properties the element property matrix (or stress-strain matrix) is created. The element geometry is defined by the nodes. In the case of a triangle and rectangle, there are three and four nodes respectively. Each node is defined by its position in space. In the case of a plane strain model the position can be defined in the horizontal (x) and vertical (y) distances from an origin. In the case of the three programs, the origin is defined at the lower left corner of the mesh. Figure 3.4 shows the nodal coordinate scheme for the model.

The nodes are also defined by their ability to move in space. This movement is called the node freedom and defines displacement in the x and y directions for a plane

strain model. For example, nodes at the surface or in the middle of the continuum may move freely in both directions. At the boundaries, the nodes may be restrained in one (roller support) or in both (hinged support) directions. Convention states that zero indicates freedom to move while one indicates restraint. Therefore, the nodal freedom array for a free node would be (0,0), a hinged node (1,1), and (1,0) for a node that can only move vertically. Figure 3.5 illustrates this principle.

Once the stress-strain relationship is defined and the nodal coordinates and freedoms determined, then the local stiffness matrix can be created for each element. This matrix is expressed as:

$$\begin{bmatrix} S_{N11} & S_{N12} & S_{N13} & S_{N14} & S_{N15} & S_{N16} & S_{N17} & S_{N18} \\ \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet \\ S_{N81} & S_{N82} & S_{N83} & S_{N84} & S_{N85} & S_{N86} & S_{N87} & S_{N88} \end{bmatrix}$$

for a rectangular element where the stiffness matrix is based on the element stress strain relationship, nodal coordinates and freedoms.

Each element stiffness matrix in the mesh is added to the global stiffness matrix for the continuum. This global matrix can be expressed as:

$$[G] \times [d] = [l] \qquad (3.19)$$

where [G] = global stiffness matrix

[d] = nodal displacements

[l] = nodal loads

This matrix is then reduced and solved for the nodal displacements using either the Gauss-Jordan or Choleski method.

Once the nodal displacements have been calculated, they are redistributed back to the elements so that local stresses and strains may be recovered.

3.2.2 Elasto-Plastic Models

The setup of the global stiffness matrix for a Von Mises and Mohr Coulomb soil is similar to the linear elastic model except that an axisymmetric stress-strain matrix is used. Elasto-plasticity enters the analysis when the nodal displacements are redistributed to find element stresses and strains. The redistributions in both cases is performed in two nested loops. The outer loop is responsible for load cycles while the inner loop redistributes stresses that exceed the material's elastic limit. The Von Mises method uses a yield stress to indicate this limit while the Mohr Coulomb analysis relies on a visco plastic yield function. In both cases, if the stress does not exceed the yield,then the elastic stress-strain relationship is used. Otherwise an elasto-plastic stress-strain matrix (different for Von Mises

and Mohr Coulomb) must be created. At the end of the inner loop, the stress and strain increments are added to those previously accumulated before proceeding to the next load cycle.

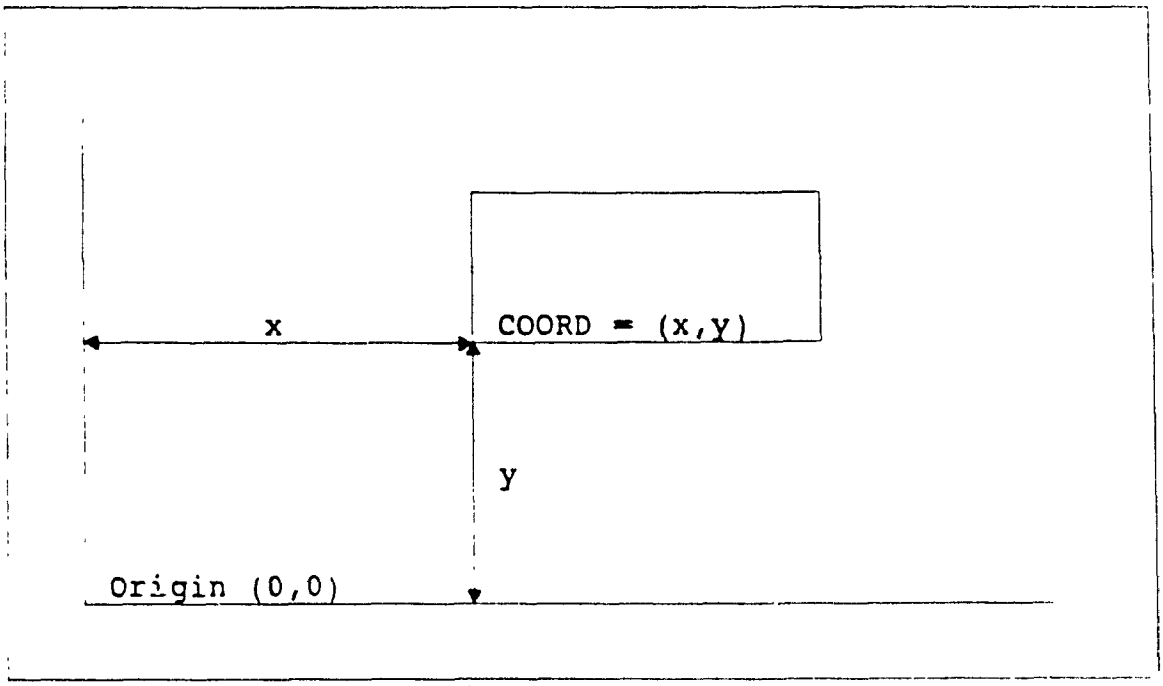


Figure 3.4: Nodal Coordimates

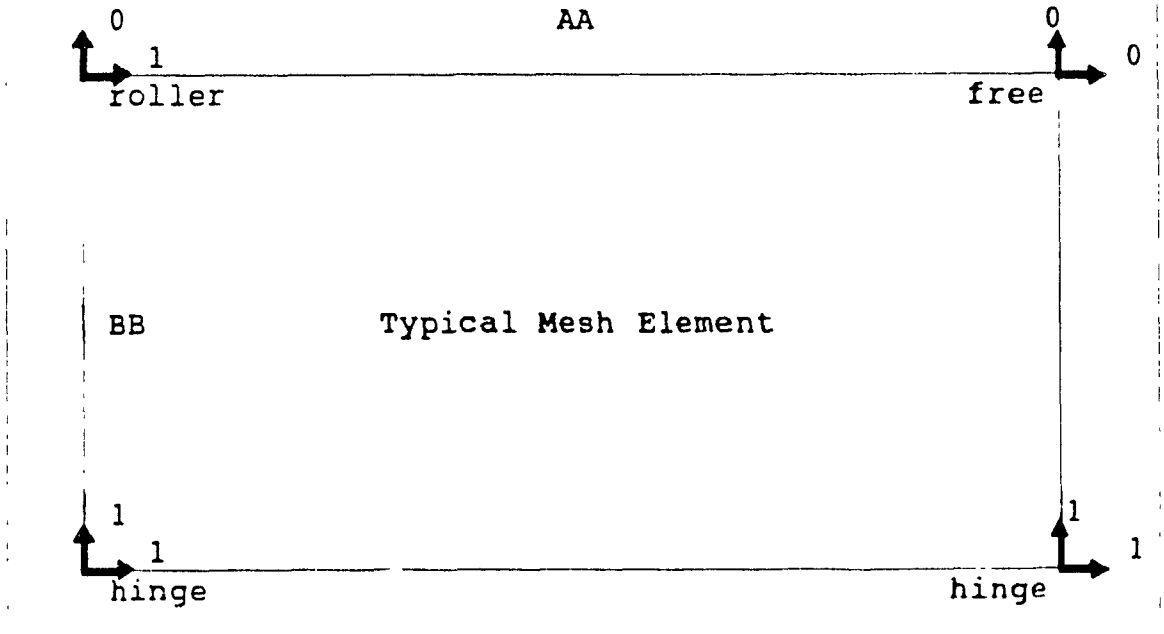


Figure 3.5: Element Node Freedoms

Chapter 4

The Pavement Model

4.1 Introduction

The finite element programs require an accurate model of the pavement structure and reasonable soil parameters to accurately predict the effects of the loads on the material.

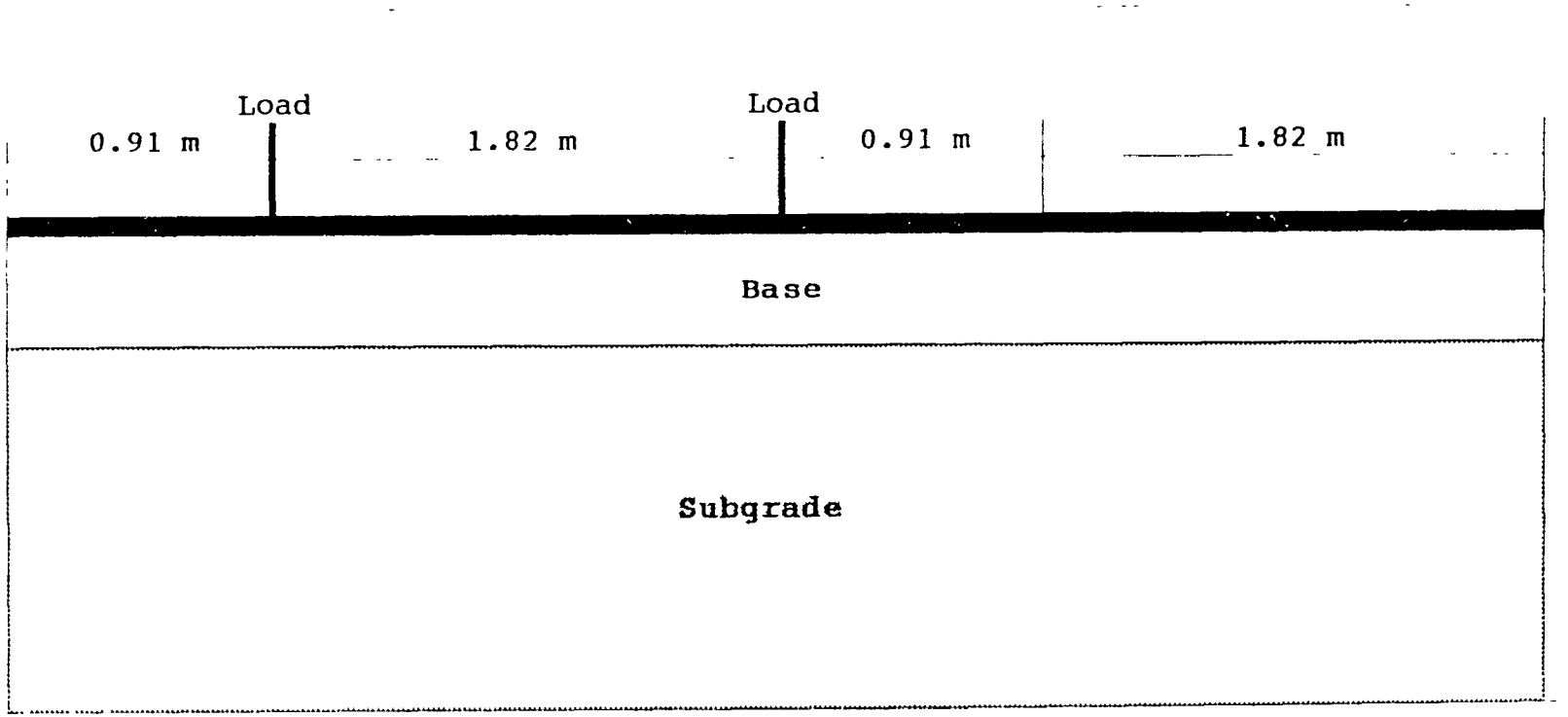
In order to insure that the results are as precise as possible, the finite element mesh must be carefully thought out. A mesh with very large elements will yield inaccurate results while one that is too fine will waste computer memory and time.

Laboratory and field testing provide the parameters that define the material to be studied. Without realistic soil parameters, the analysis becomes an exercise in theoretical mathematics (Desai and Siriwardane, 1984).

4.2 The Mesh Development

The finite element programs used to analyze a typical three layer flexible pavement system consisting of an asphalt wearing surface, granular base, and subgrade. This type of pavement is selected because it is the most commonly used in construction. Use of flexible pavements range from residential streets to major highways. Figure 4.1 shows the structure to be studied with the locations of the wheel loads.

The asphalt surface course is not subjected to



Asphalt Thickness	0.15 m
Base Thickness	0 m (Treated Subgrade) 0.15 m, 0.30 m, 0.45 m
Subgrade Thickness	6.0 m

Figure 4.1: Pavement Structure

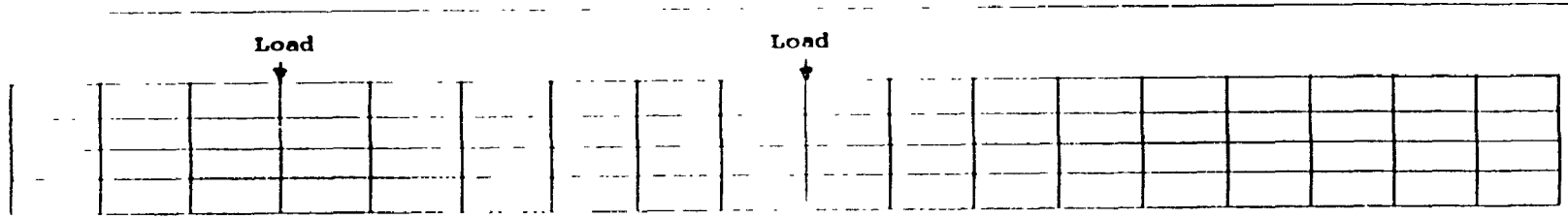
analysis and the wheel loads are assumed to remain unchanged when applied to the base. Desai and Siriwardane (1984) discuss the analysis of a multi-layered railway embankment where each layer possesses a different soil behaviour and consequently must be analyzed separately. This approach is taken with the pavement model where the base and subgrade layers are analyzed separately.

4.3 The Base Mesh

As the base layer is usually thinner and stiffer than the subgrade, stresses tend not to distribute far from the area of the point load and thus require a fine mesh to model the distribution accurately. Figure 4.2 shows a portion of this mesh along with a table of the node freedoms. The entire mesh consists of 72 elements with 95 nodes arranged in an 18 by 4 pattern. The mesh shown in Figure 4.2 is for the 0.45 m base situation.

While the mesh is enclosed by a fixed boundary, in reality, loads exerted on the pavement will be distributed infinitely in all directions. However, each side of the mesh contains roller supports while the bottom is hinged. This situation does not allow stresses to leave the system and thus conservation occurs. Since both sides and bottom are fixed in some way, the behaviour of the outermost elements may not accurately depict soil behaviour.

Consequently, an additional row has been added to the bottom of the mesh to make the base appear to be 0.60 m.



Degrees of Freedom (DOF)

Node	DOF	DOF	Node	DOF	DOF	Node	DOF	DOF	Node	DOF	DOF	Node	DOF	DOF	Node	DOF	DOF
1		1	17	23	24	33	49	50	49	75	76	65			81	125	126
2		2	18	25	26	34	51	52	50			66	101	102	82	127	128
3		3	19	27	28	35			51	77	78	67	103	104	83	129	130
4		4	20			36	53	54	52	79	80	68	105	106	84	131	132
5			21	29	30	37	55	56	53	81	82	69	107	108	85		
6	5	6	22	31	32	38	57	58	54	83	84	70			86	133	134
7	7	8	23	33	34	39	59	60	55			71	109	110	87	135	136
8	9	10	24	35	36	40			56	85	86	72	111	112	88	137	138
9	11	12	25			41	61	62	57	87	88	73	113	114	89	139	140
10			26	37	38	42	63	64	58	89	90	74	115	116	90		
11	13	14	27	39	40	43	65	66	59	91	92	75			91		141
12	15	16	28	41	42	44	67	68	60			76	117	118	92		142
13	17	18	29	43	44	45			61	93	94	77	119	120	93		143
14	19	20	30			46	69	70	62	95	96	78	121	122	94		144
15			31	45	46	47	71	72	63	97	98	79	123	124	95		
16	21	22	32	47	48	48	73	74	64	99	100	80					

Figure 4.2: Finite Element Mesh for the Base Course

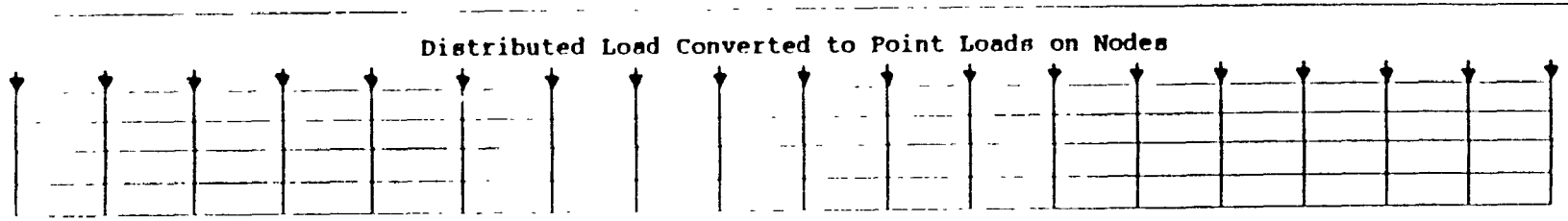
deep. Since the lowest nodes are assumed to move freely with the subgrade surface, the fourth row is added so that the third may move in the x and y directions. Thus the third row stresses are determined and transferred to the subgrade model.

The element geometry was chosen because it has been found that the stresses and strains exerted on each node are most accurately predicted when the element width to height ratio is 2:1. Should the ratio change to 1.5:1 or 3:1, then the calculated values will be greater or smaller than those which actually occur.

4.4 The Subgrade Mesh

The subgrade utilizes two separate models, since the stress distribution in the upper 0.60 m. of the subgrade is different from the other layers. In the first 0.15 m. of the subgrade, stresses are distributed close to the wheel loads line of action. These stresses quickly distribute themselves laterally so that they affect the entire pavement width by 0.60 m. Beyond this level, the stresses and strains are more uniformly distributed and change slightly from layer to layer. The fine mesh, similar in geometry to the base model shown in Figure 4.3 except that all the top nodes are loaded. Since the main thrust of the analysis is to calculate the largest strain on the surface of the subgrade, (vertical strain), a fine mesh extending to a great depth is unnecessary.

A mesh with larger elements has been created to show the stress distribution beyond the depth of 0.60 m. Each



Degrees of Freedom (DOF)

Node	DOF	DOF	Node	DOF	DOF	Node	DOF	DOF	Node	DOF	DOF	Node	DOF	DOF	Node	DOF	DOF
1		1	17	23	24	33	49	50	49	75	76	65			81	125	126
2		2	18	25	26	34	51	52	50			66	101	102	82	127	128
3		3	19	27	28	35			51	77	78	67	103	104	83	129	130
4		4	20			36	53	54	52	79	80	68	105	106	84	131	132
5			21	29	30	37	55	56	53	81	82	69	107	108	85		
6	5	6	22	31	32	38	57	58	54	83	84	70			86	133	134
7	7	8	23	33	34	39	59	60	55			71	109	110	87	135	136
8	9	10	24	35	36	40			56	85	86	72	111	112	88	137	138
9	11	12	25			41	61	62	57	87	88	73	113	114	89	139	140
10			26	37	38	42	63	64	58	89	90	74	115	116	90		
11	13	14	27	39	40	43	65	66	59	91	92	75			91		141
12	15	16	28	41	42	44	67	68	60			76	117	118	92		142
13	17	18	29	43	44	45			61	93	94	77	119	120	93		143
14	19	20	30			46	69	70	62	95	96	78	121	122	94		144
15			31	45	46	47	71	72	63	97	98	79	123	124	95		
16	21	22	32	47	48	48	73	74	64	99	100	80					

Figure 4.3: Finite Element Mesh for the Upper Subgrade

element in this case has a width of 1.82 m. by a 1.00 m. height arranged in a three by six pattern. Figure 4.4 shows the coarse mesh along with a table of node degrees of freedom. This mesh is used only to illustrate the stress distribution below 0.60 m. and is not a part of the remainder of the analysis.

Since each mesh has either a different geometry or soil parameters, each must be analyzed separately. This approach is used in Desai and Siriwardane, (1984) when dealing with the railway embankment problem.

4.5 Determination of Parameters

Laboratory or field testing play a crucial role in defining the soil parameters to be used in finite element analysis. Without reasonable parameters, there will always be a gap between theory and practice (Desai and Siriwardane, 1984). Thus it is necessary to perform laboratory tests on samples which represent the conditions of the region to be analyzed.

The main soil parameters that the three programs require are:

E modulus of elasticity is derived from the triaxial test. The resulting slope of the τ_{oct} vs ε curve when multiplied by ($3/\sqrt{2}$) yields E.

ν Poisson's ratio is the proportion of the

vertical to horizontal stress in the material. Its effects show that the vertical compressive stress is associated with the horizontal stress extending outwards from the wheel load (Lay, 1986). The ratio is derived from the hydrostatic compressive test. The resulting slope of the σ_{all} vs ε curve yields the bulk modulus K which is inserted into following equation:

$$\nu = (K - EK)/2K \qquad (4.1)$$

c soil cohesion is derived from the triaxial test and c can be found from the Mohr's circle where the failure envelope intersects with the y-axis

ϕ internal angle of friction is derived from the triaxial test. This value of ϕ is the slope of the failure envelope on a Mohr's circle. This value can also be found in some cases by pouring the sample into a pile and measuring the angle of the resulting slope

CBR California Bearing Ratio is derived from the plate load test. The resulting load to cause 0.1 inch penetration is compared with that required to penetrate a standard sample of

crushed rock to the same depth and expressed by:

$$CBR = (L_s/L_r) \times 100 \qquad (4.2)$$

where L_s = load carried by specimen at 0.1 in piston penetration

L_r = load carried by standard crushed rockbase at 0.1 in piston penetration

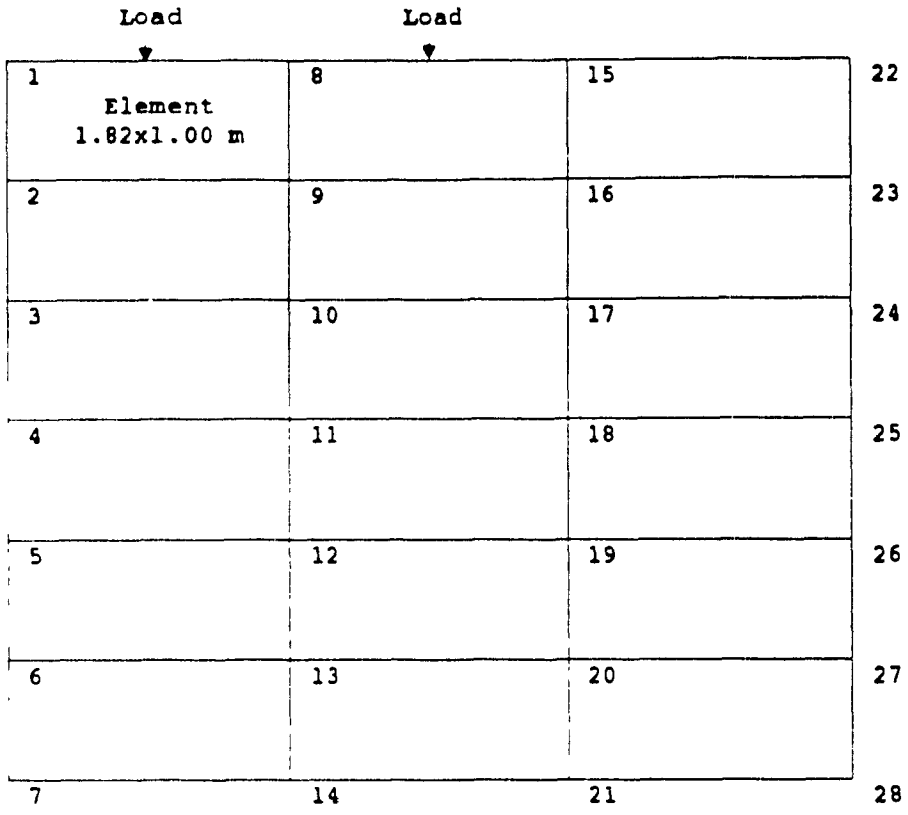
The result is then rated on a scale of 1 to 100. The CBR (Pandey, 1990) can also be related to the modulus of elasticity by:

$$CBR = (E \text{ in MPa})/10 \qquad (4.3)$$

These tests are discussed in greater detail in Desai and Siriwardane (1984), Oglesby and Hicks (1982), and Bowles (1984).

The parameters that are used in the computer trials are shown in Table 4.1. These values are taken from samples presented in Desai and Siriwardane (1984) and Oglesby and Hicks (1982). The parameters are also checked against typical values in Bowles (1982 and 1984) to insure that they are within reasonable limits. Thus there are six cases for the three behaviours and soil types. Unrealistic situations such

as a Von Mises sand are eliminated from the trials. These cases when combined with three base thicknesses and five loading conditions comprise a total of 90 cases.



Degrees of Freedom (DOF)

Node	DOF x	DOF y
1		1
2		2
3		3
4		4
5		5
6		6
7		
8	7	8
9	9	10
10	11	12
11	13	14
12	15	16
13	17	18
14		

Node	DOF x	DOF y
15	19	20
16	21	22
17	23	24
18	25	26
19	27	28
20	29	30
21		
22		31
23		32
24		33
25		34
26		35
27		36
28		

Figure 4.4: Large Element Subgrade Mesh

Table 4.1: Soil Data for Finite Element Analysis

Base

Analysis Type	Gravel
Linear Elastic	E=500 MPa v=0.30

Subgrade

Analysis Type	Sand c=0	Clay o=0	Sand-Clay c=o soil
Linear Elastic	E=190 MPa v=0.30		E=207 MPa v=0.30
Von Mises		E=207 MPa v=0.30 Y.S.=620 kPa	
Mohr Coulomb	E=190 MPa v=0.30 c=0.0 kPa o=35.0	E=207 MPa v=0.30 c=120 kPa o=0.00	E=207 MPa v=0.30 c=70 kPa o=20.0

Chapter 5

Results

5.1 Stress Distribution in the Base

The first phase of pavement analysis involve the calculation of stress distribution through the base layer. The results, found through linear elastic finite element analysis, show how the wheel loads on the asphalt surface vary with the depth of base.

For a granular material with E = 500 MPa and ν = 0.3, stresses are reduced by 13%, 36%, and 50% for 0.15, 0.30, and 0.45 m. of base respectively. For example a standard 80 kN load would be reduced to 70, 51 and 40 kN for base depths of 0.15, 0.30 and 0.45 m. respectively. The decrease in stress directly below the wheel load is balanced by an increase in stresses surrounding the load. Eventually at a certain depth, the stresses will be distributed uniformly over the width of the pavement. Figure 5.1 shows the stress distribution through the various base thicknesses. These stresses are then converted to point loads and applied to the surface of the subgrade mesh.

5.2 Results of Subgrade Analysis

The six soil cases in Table 4.1 are analyzed using the finite element method. The combination of three soil types (sand, clay, and sand-clay), three methods of analysis (linear elastic, Von Mises, and Mohr Coulomb), three base

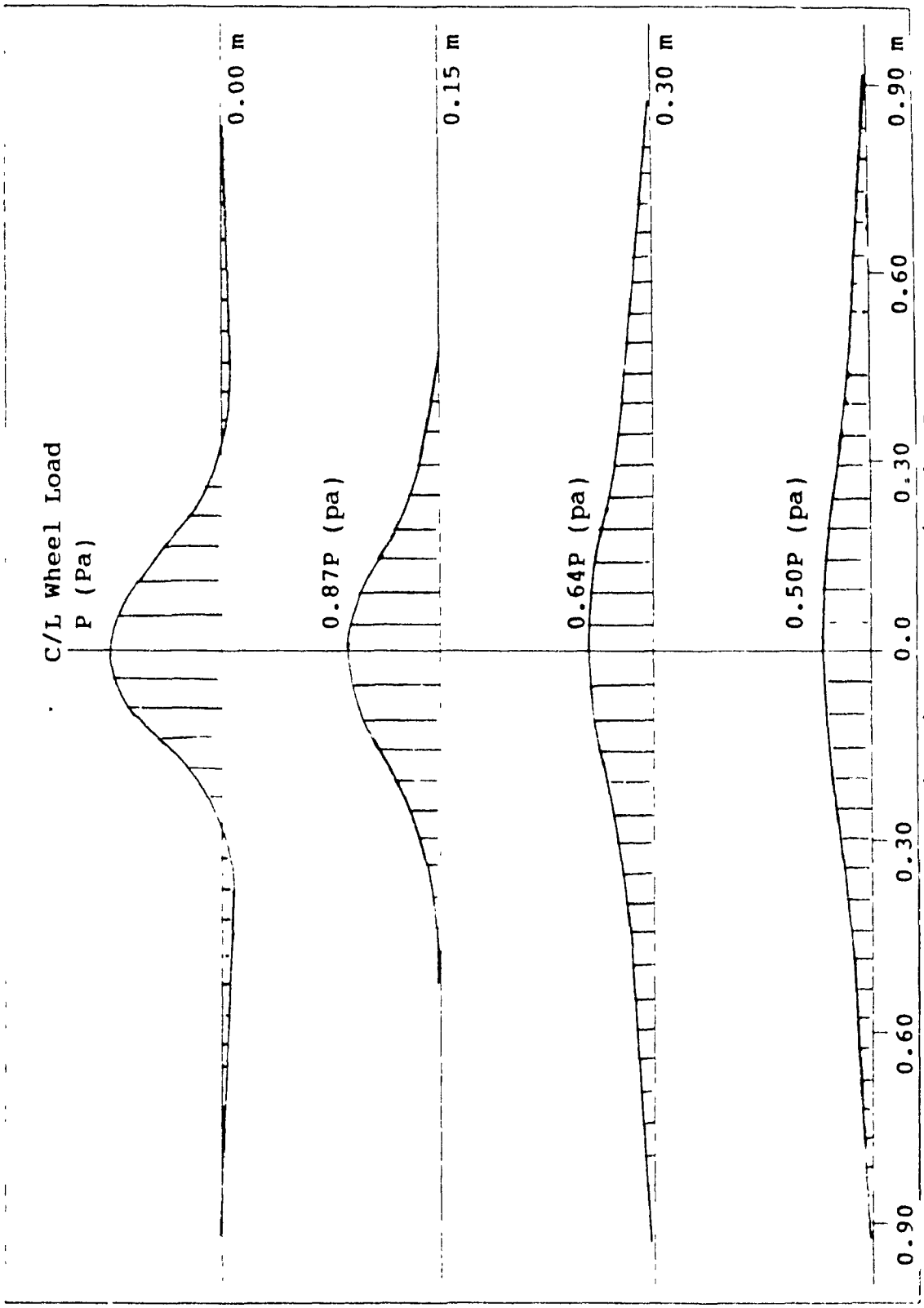


Figure 5.1: Typical Stress Distribution Pattern in the Base

depths, and five truck loads require ninety computer trials to be run. In each case the stress-strain relationship can be derived. Figure 5.2 shows an example of this relationship for the three methods of analysis. Each of the three cases is found to be in the elastic range for the given loads. It should be noted that the two elasto-plastic methods yield the same results in the elastic range and consequently can be plotted on the same curve. However, the linear elastic method produces results which differ from the others and hence possesses a slightly different relationship.

The linear elastic and Mohr Coulomb stress patterns are similar. Initially the stress tend to crest at the centreline of the axle becoming more uniform with an increase in depth. The Von Mises pattern initially crests at the outer wheel and then develops the distribution pattern of the previous two methods.

The stresses and strains in the uppermost 0.60 m. of the subgrade remain closely in line with the point loads rather than being distributed more evenly over the width of the pavements. Due to this non uniform distribution, a mesh with large elements would be insufficient to model the situation adequately. Therefore a mesh with smaller elements must be used for the upper 0.60 m. of the subgrade. The stress distribution at 0.60 m. is more uniform and is similar to the pattern shown on the large element mesh.

Since the main purpose of the finite element

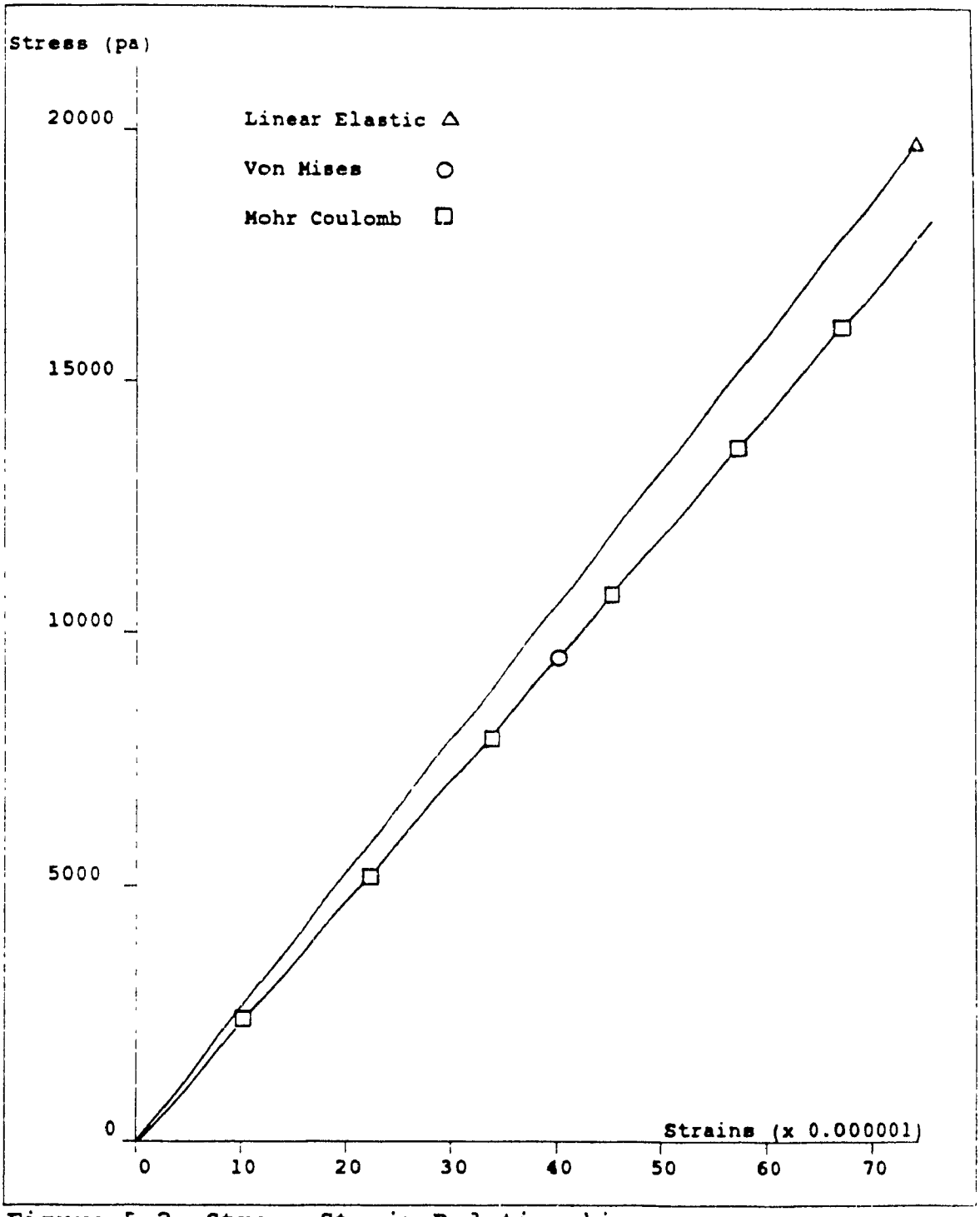


Figure 5.2: Stress-Strain Relationships

analysis is to find the largest strain developed in the topmost row of the subgrade mesh (ε_c), the refined mesh is used for greater accuracy. The large element mesh is only used to illustrate stress distributions beyond the critical 0.60 m. depth and will not be discussed further. The vertical compressive strains (ε_c) for the six soil cases are shown in Table 5.1.

The vertical strains for the two types of analysis (linear elastic and elasto-plastic) are found differently. Linear elastic analysis is static and strains are located at the nodes of each element. The vertical strain is the largest compressive strain at the top of the mesh. This point usually corresponds with the node under the largest load.

The elasto-plastic loading is cyclical and stresses and strains are located at the element centres. In order to find the largest vertical strain at the top of the mesh, the stresses and strains are first plotted to find their relationship. The wheel load is then converted to a stress by dividing the point load by the element width. This stress is then inserted into the relationship to find the corresponding vertical strain.

By assuming two different soil behaviours for each soil type, a comparison can be made. For example, the sand and sand-clay soil cases show that the Mohr Coulomb method yields higher vertical strain values than the linear elastic analysis. The Mohr Coulomb values are found to be an average

of nine percent higher than their linear elastic counterparts. The clay case which compares Von Mises and Mohr Coulomb behaviour indicates that the two methods would yield the same results (at least in the elastic range). However, when the Mohr Coulomb results are compared, the values for the clay and the sand-clay are exactly the same. It can be seen that the values of cohesion and angle of friction (which changed in these two cases) have no effect on the results in the elastic range and the modulus of elasticity and Poisson's ratio are the governing parameters.

5.3 Strain Contours

The values of vertical strain can also be arranged in a matrix form as shown in Figures 5.3 to 5.8. Contours representing certain values of vertical strain can then be plotted. These contours indicate the combined effects of loads and base thickness on the magnitude of ε_c.

A comparison of vertical strains based on loads, base thickness and material properties can be made using the contours. For example, for linear elastic sand shown in Figure 5.3 the vertical strain decreased from 0.000481 to 0.000271 m. when the base thickness is increased from 0.15 to 0.45 m. for a standard 80 kN load. This increase in the base course represents a 44 % decrease in vertical compression strains in the base course. If an 80 kN load were exerted on a sand with a 0.30 m. base then the vertical strain would be

Table 5.1: Vertical Strains

		Sand		Clay		Sand - Clay	
Load (kN)	Base (m)	L/E	M/C	V/M	M/C	L/E	M/C
71	0.15	427*	469	430	430	392	430
	0.30	312	345	316	316	286	316
	0.45	244	267	246	246	224	246
80	0.15	481	528	485	485	442	485
	0.30	351	388	356	356	322	356
	0.45	271	301	277	277	249	277
89	0.15	535	587	539	539	491	539
	0.30	391	432	396	396	359	396
	0.45	306	335	308	308	281	308
98	0.15	589	647	594	594	541	594
	0.30	430	476	437	437	395	437
	0.45	337	369	339	339	301	339
107	0.15	644	706	648	648	591	648
	0.30	470	519	477	477	431	477
	0.45	368	403	370	370	338	370

* Strains x 10^{-6}

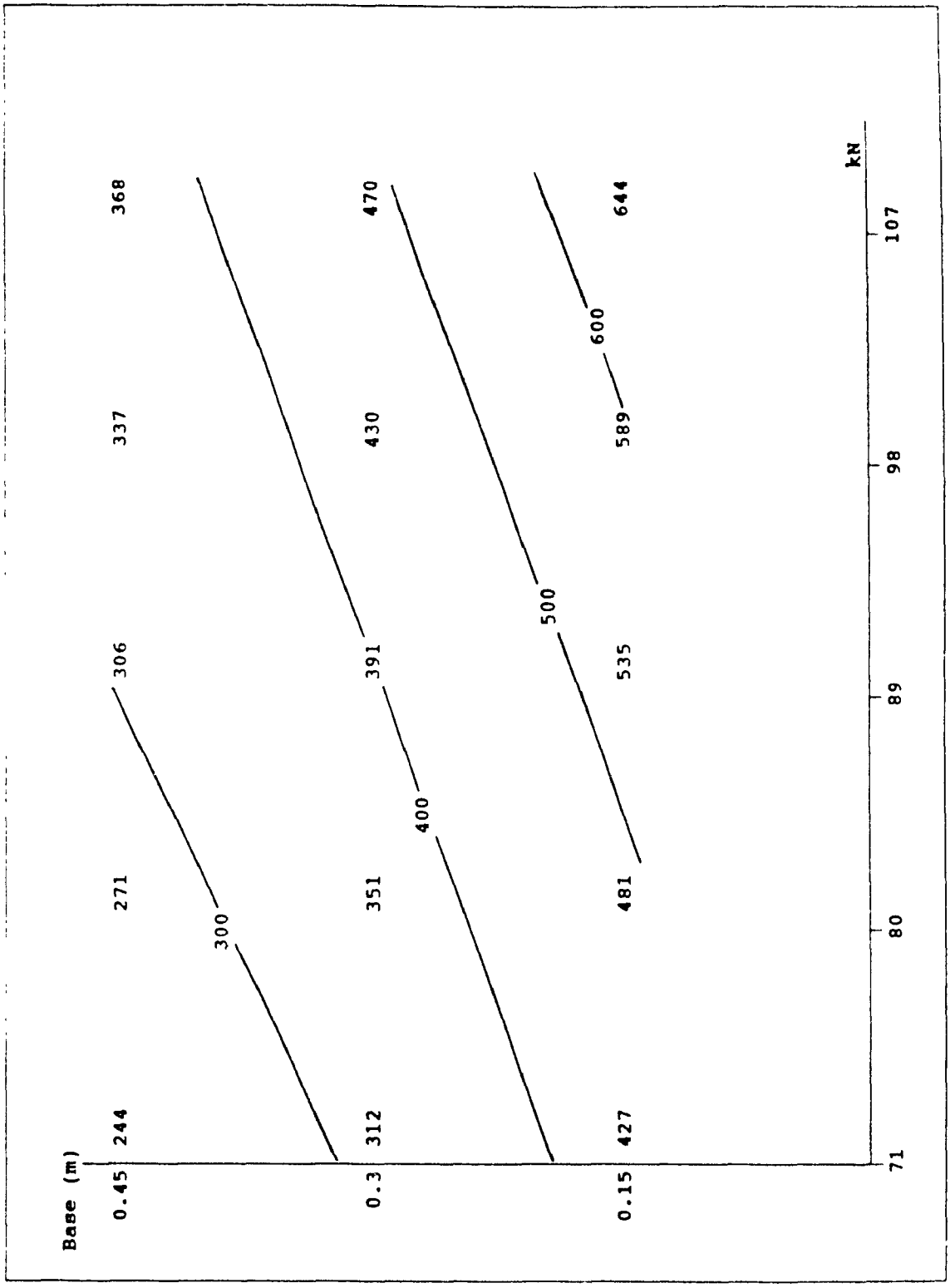


Figure 5.3: Strain Contours (Linear Elastic Sand)

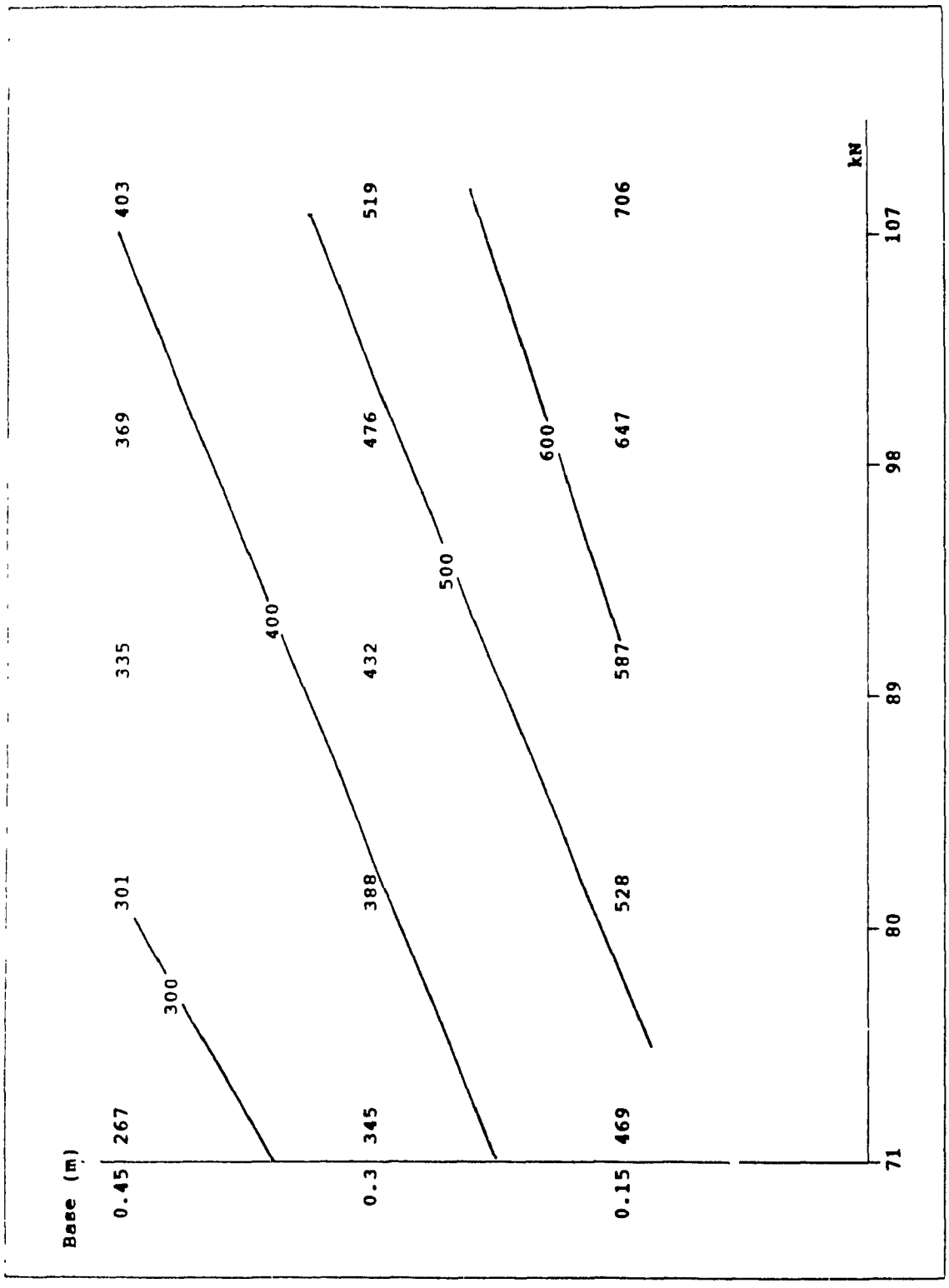


Figure 5.4: Strain Contours (Mohr Coulomb Sand)

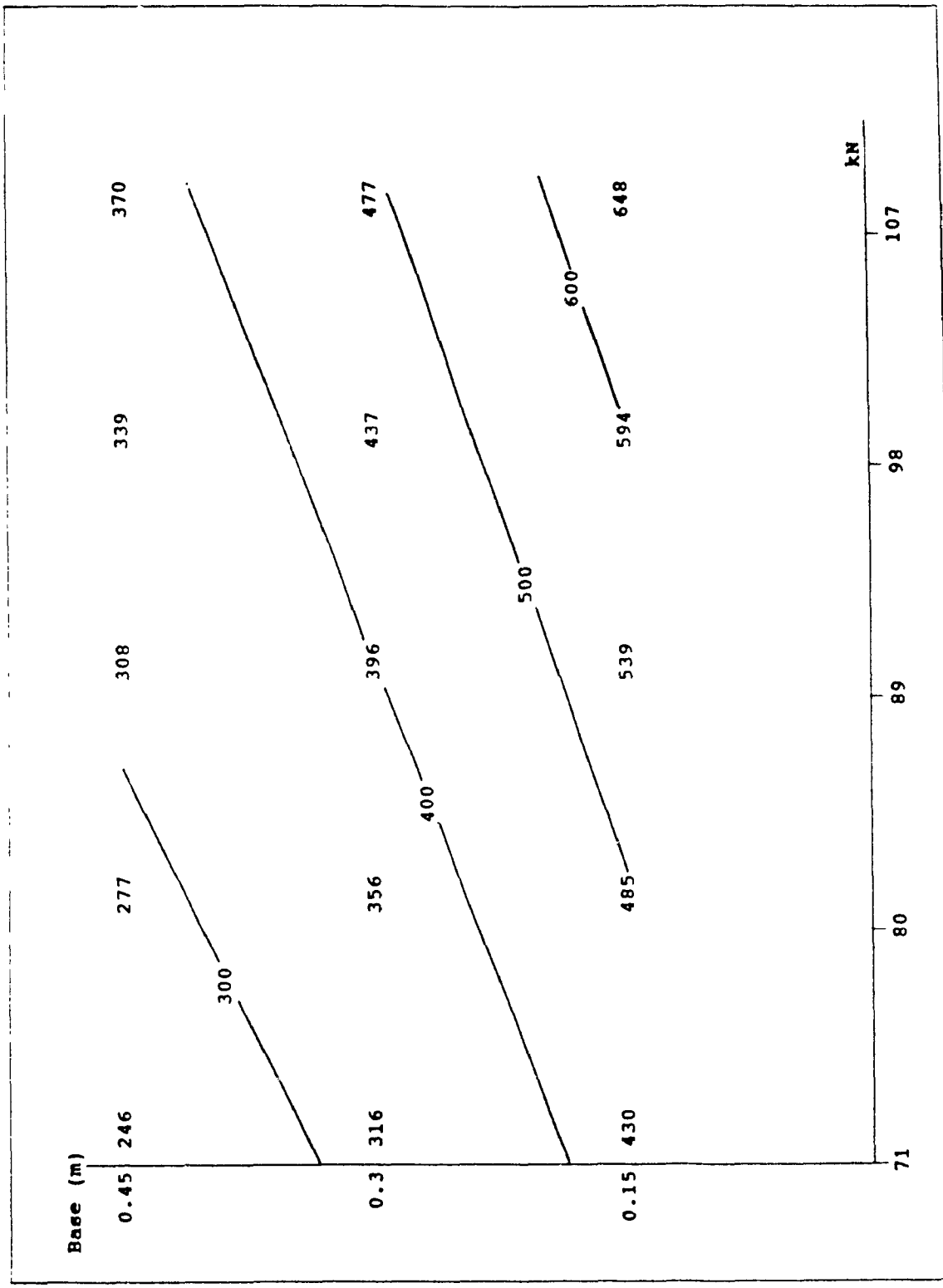


Figure 5.5: Strain Contours (Von Mises Clay)

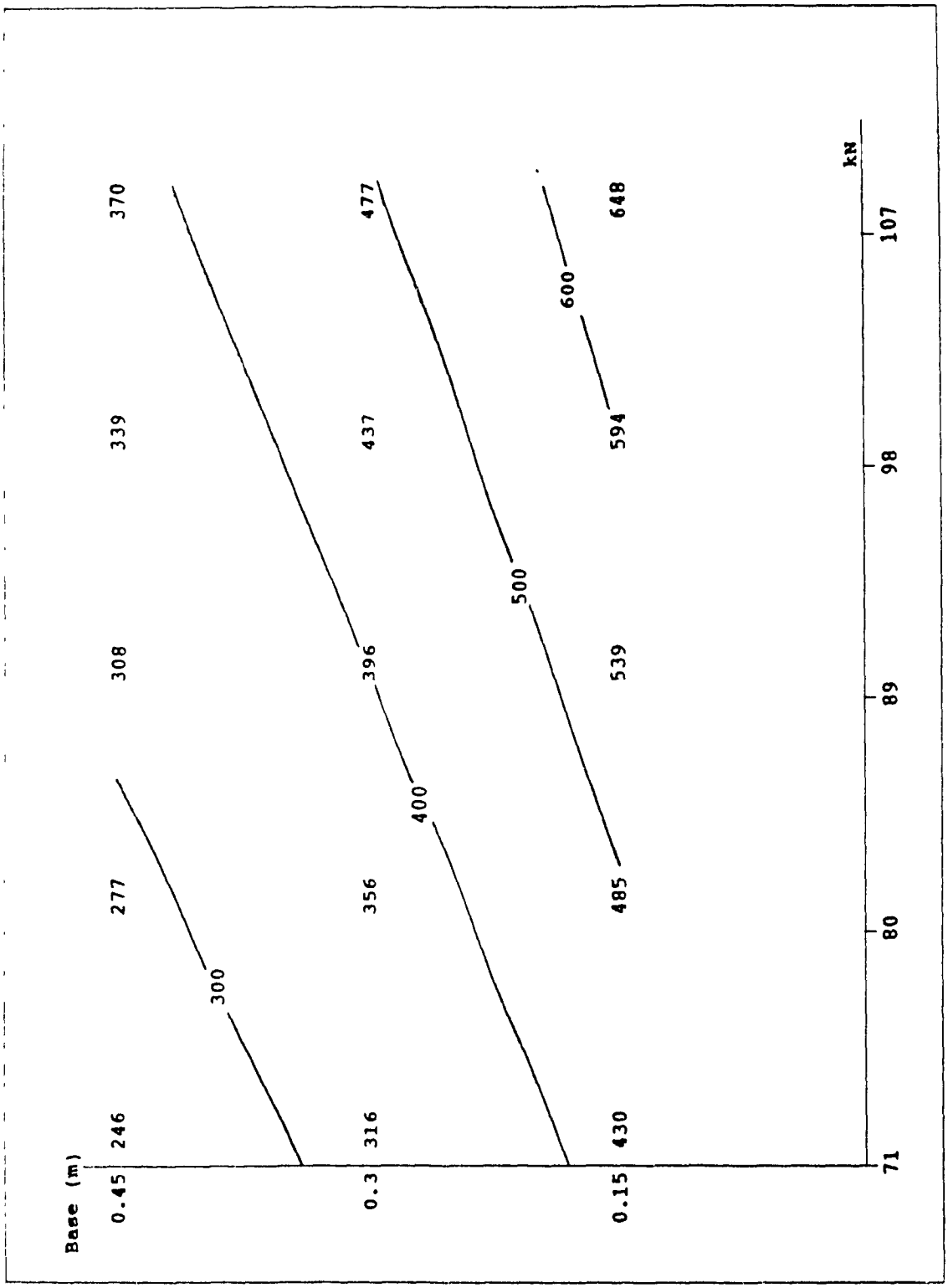


Figure 5.6: Strain Contours (Mohr Coulomb Clay)

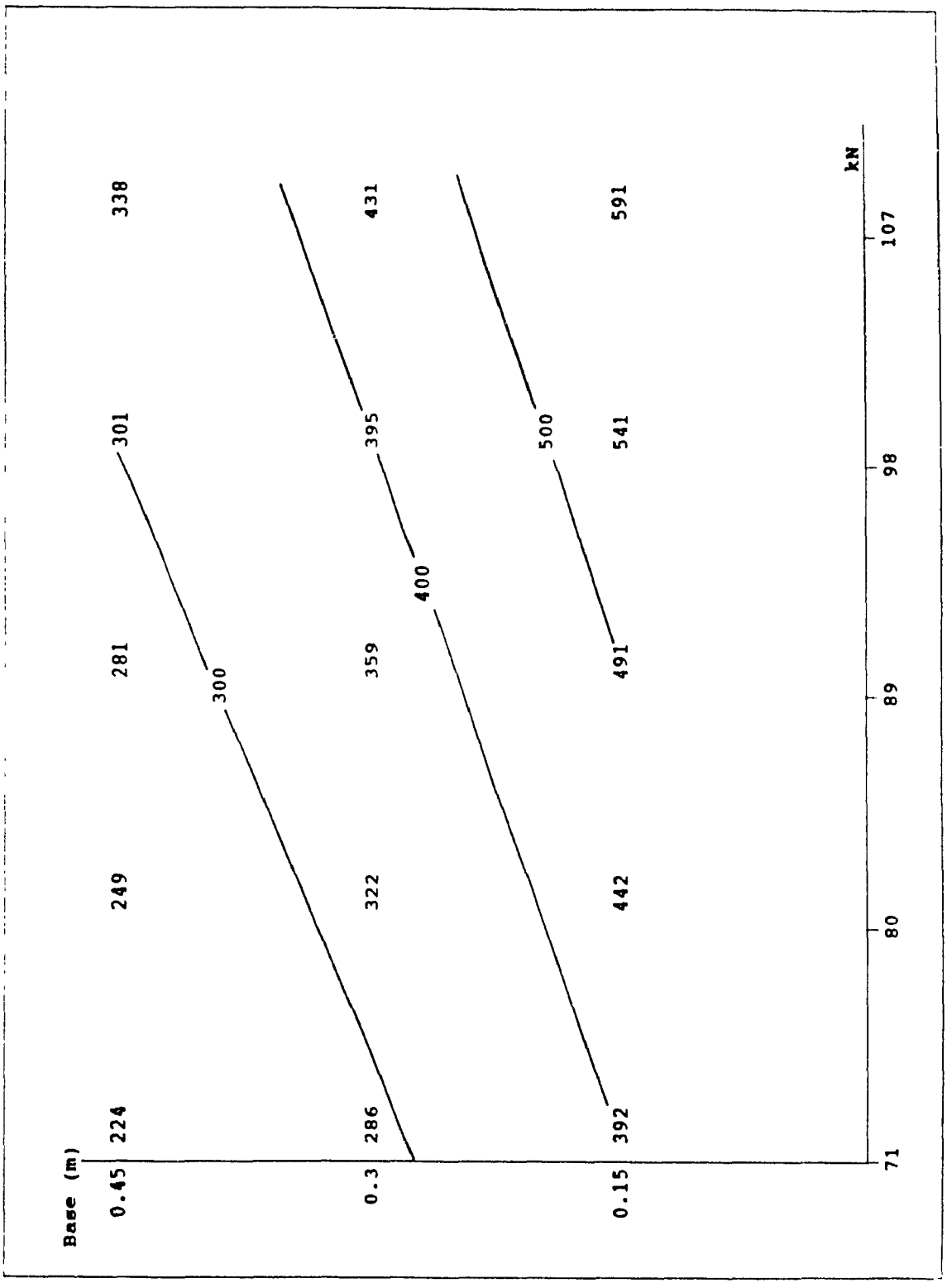


Figure 5.7: Strain Contours (Linear Elastic Sand-Clay)

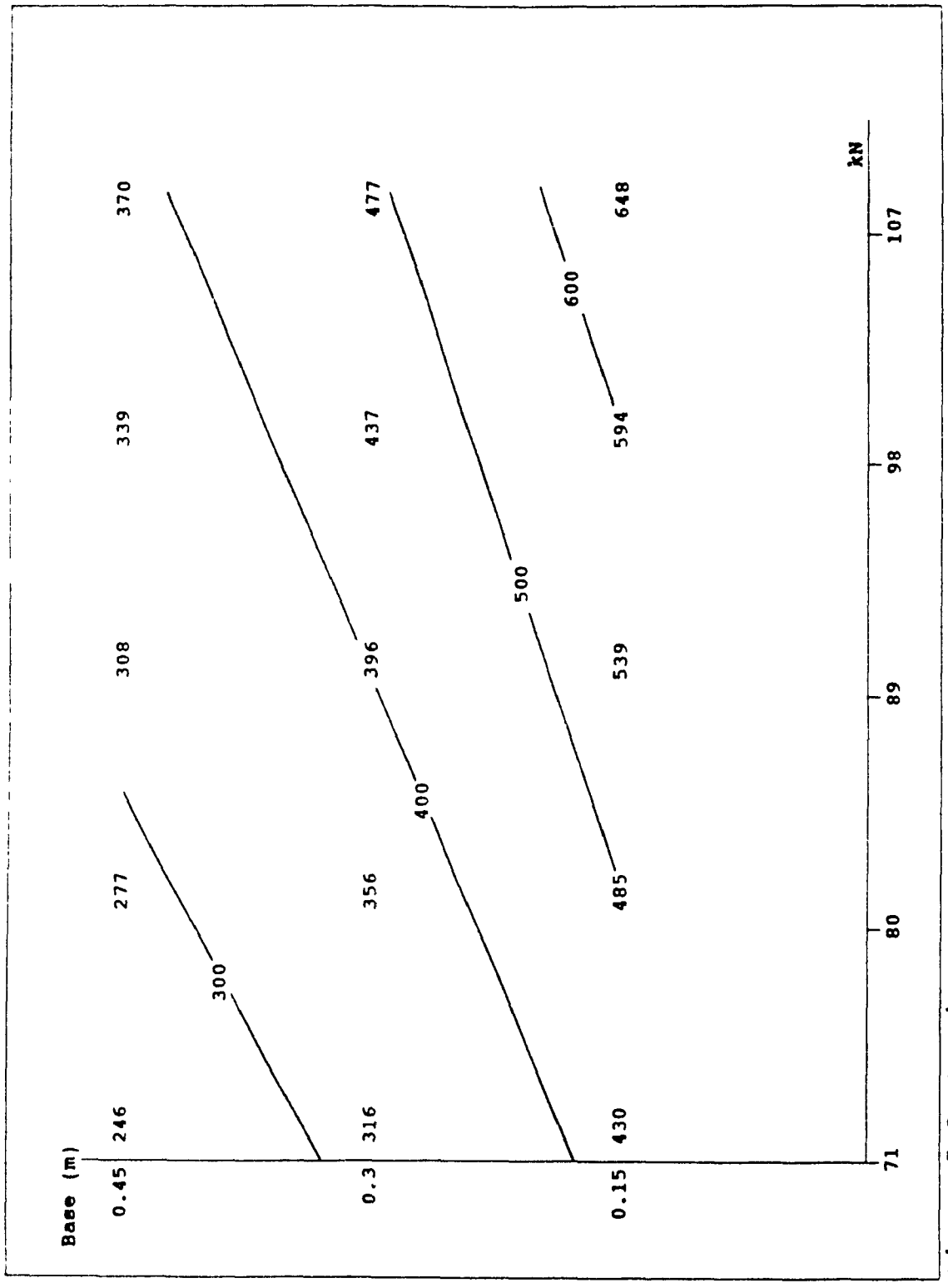


Figure 5.8: Strain Contours (Mohr Coulomb Sand-Clay)

0.000351 and 0.000388 for a subgrade with linear elastic and Mohr Coulomb behaviours. The elasto-plastic behaviour yields more conservative results given identical parameters. The changes in vertical strain are also directly proportional to changes in loadings.

The strain contours represent a fundamental relationship between the applied load and the base thickness in producing the vertical strain at the top of the subgrade layer.

For example, if a horizontal line is drawn in Figure 5.3 corresponding to a base depth of 0.3 m., then the vertical strains for each load case can be found. If a vertical line is drawn corresponding to a load case of 80 kN, then the vertical strain can be found for each base thickness. From these two lines, it can be concluded that the vertical strains decrease as the base thickness increases and increase as the axle load increases. Thus the contours indicate a combination of these two effects in producing a specific magnitude of vertical strain.

The strain contour graphs can be utilized to determine the specific combination of base thickness and axle load that can produce a given magnitude of the vertical strain. These graphs form the basis for the pavement design method developed in the next chapter.

Chapter 6

Paveme ι Design Method

6.1 Introduction

As stated in the previous chapter, the vertical strain (ε) contours created apply only to the three soil types studied. A more general set of contours would have to be formulated if a design method is to encompass as many des.gn types as possible. In the previous chapter the contours are based on loads and base thicknesses. The design curves should rely on base depths, modulus of elasticity, Poisson's ratio, and behaviour types. The different loads are not necessary since loads can be factored to an 80 kN (18,000 lb) standard. These design curves are no longer pertinent to one soil type such as sand or clay. The curves are based soley on behaviour such as Mohr Coulomb and it is the designer's responsibility to match the soil type to the appropriate behaviour. The design method is as follows:

1. Determine the traffic Equivalent Axle Loads (EAL) due to trucks in traffic stream for the service life of the pavement, using the multipliers in AASHTO Tables 6.1 and 6.2
2. Identify the subgrade behaviour
3. Determine the soil parameters
4. Determine the required base thickness
5. Determine the service life of the pavement

6.2 Development of Strain Contour Charts

From the results in chapter 5, it can be seen that the modulus of elasticity (E) and Poisson's ratio (ν) govern the value of critical strain (ε_c) in the elastic range. Changes in the angle of internal friction and cohesion do not play a role in this range. Thus the vertical strains were calculated by varying E from 100 to 1000 MPa and ν from 0.1 to 0.4. These two ranges represent most soils considered in Bowles (1982 and 1984).

The traffic load is kept to a constant 80 kN standard. Different loads can be factored to an 80 kN equivalent axle load (EAL) with AASHTO Tables (Oglesby and Hicks, 1982) and thus an additional dimension to the contours can be eliminated

These values of ε_c could then be arranged in a matrix form where the x-axis would show the different values of E and the California bearing ratio (CBR). The value of CBR as well as the elastic modulus are used because it is felt that engineers using this type of chart would be more familiar with CBR values. A separate contour graph is then drawn for each value of Poisson's ratio which varies from 0.1 to 0.4. These values of CBR and ν serve to encompass almost all the values presented in Bowles (1982 and 1984). Therefore, twelve strain contour charts have been created, taking into account all possible combinations. Figures 6.1 to 6.3 show three examples of the contour graphs. The entire set of graphs are

shown in Appendix A.

6.3 Pavement Design Method

The design method requires that all loads be factored to an 80 kN (18,000 lb) equivalent. This conversion is necessary because the Asphalt Institute equations are based on an 18,000 lb standard load. The soil parameters, modulus of elasticity and Poisson's ratio, are then identified for field or laboratory tests. These values are then combined with the appropriate soil behaviour to find the required base thickness from the design charts.

6.3.1 Determination of Traffic Load Repetitions

The design method initially requires that the daily volume of truck traffic on the pavement be converted to the number of 80 kN (18,000 lb) load repetitions per year. This conversion to equivalent 18,000 lb single axle load applications (EAL) is done by multiplying the number of vehicles in each weight class (N_v) with load equivalency factors (f_l). Thus the traffic loads per year (N_y) can be found by:

$$N_y = (365 \text{ days/year})(\Sigma N_v \times f_L) \qquad (6.1)$$

The load equivalency factor (f_l) is shown in Table 6.1. The values shown in the table are for a present serviceability index p of 2.5. This index represents a rating given to road based on high speed tests as well as on deformations and

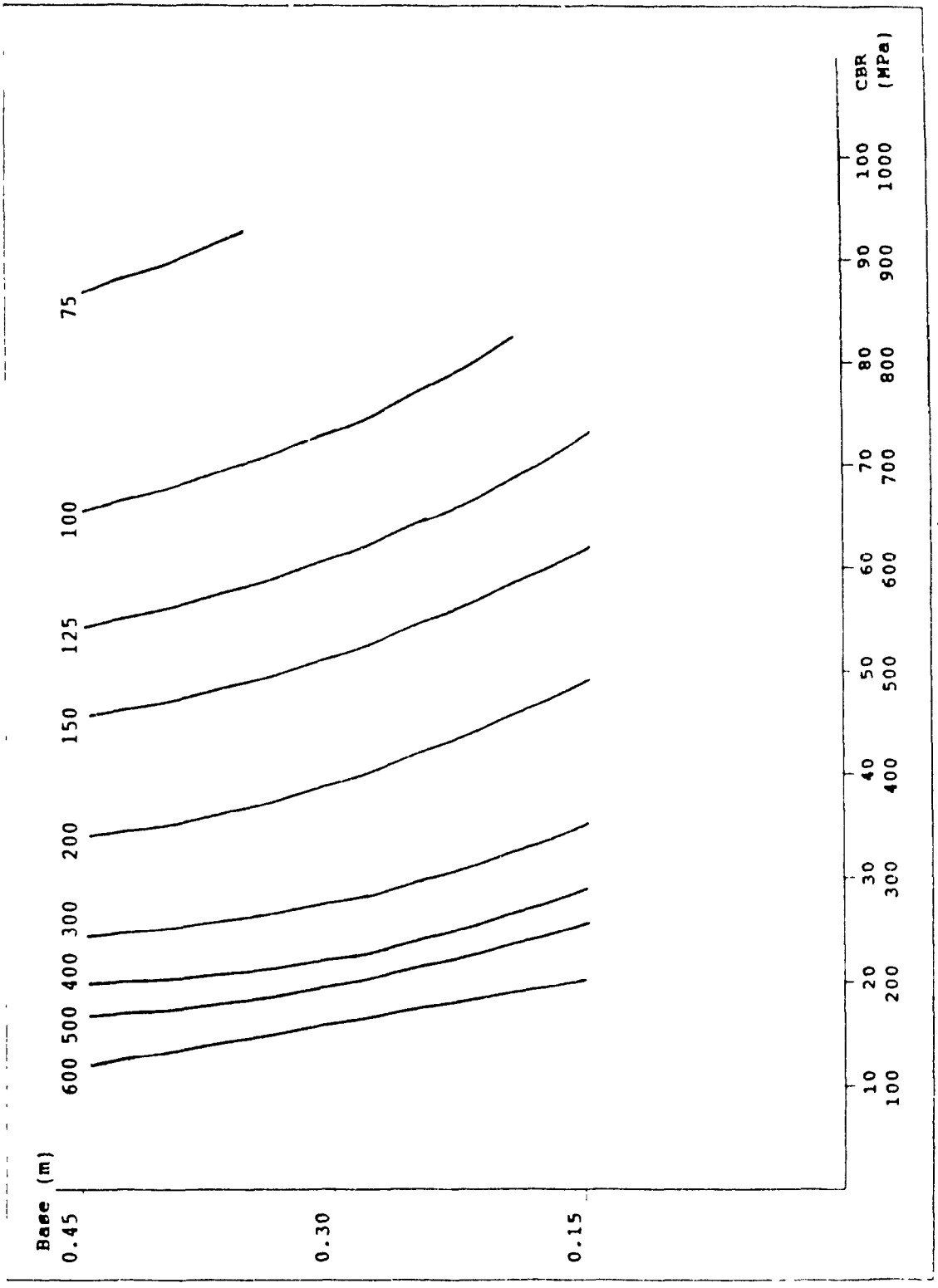


Figure 6.1: Strain Contour Chart (Linear Elastic)

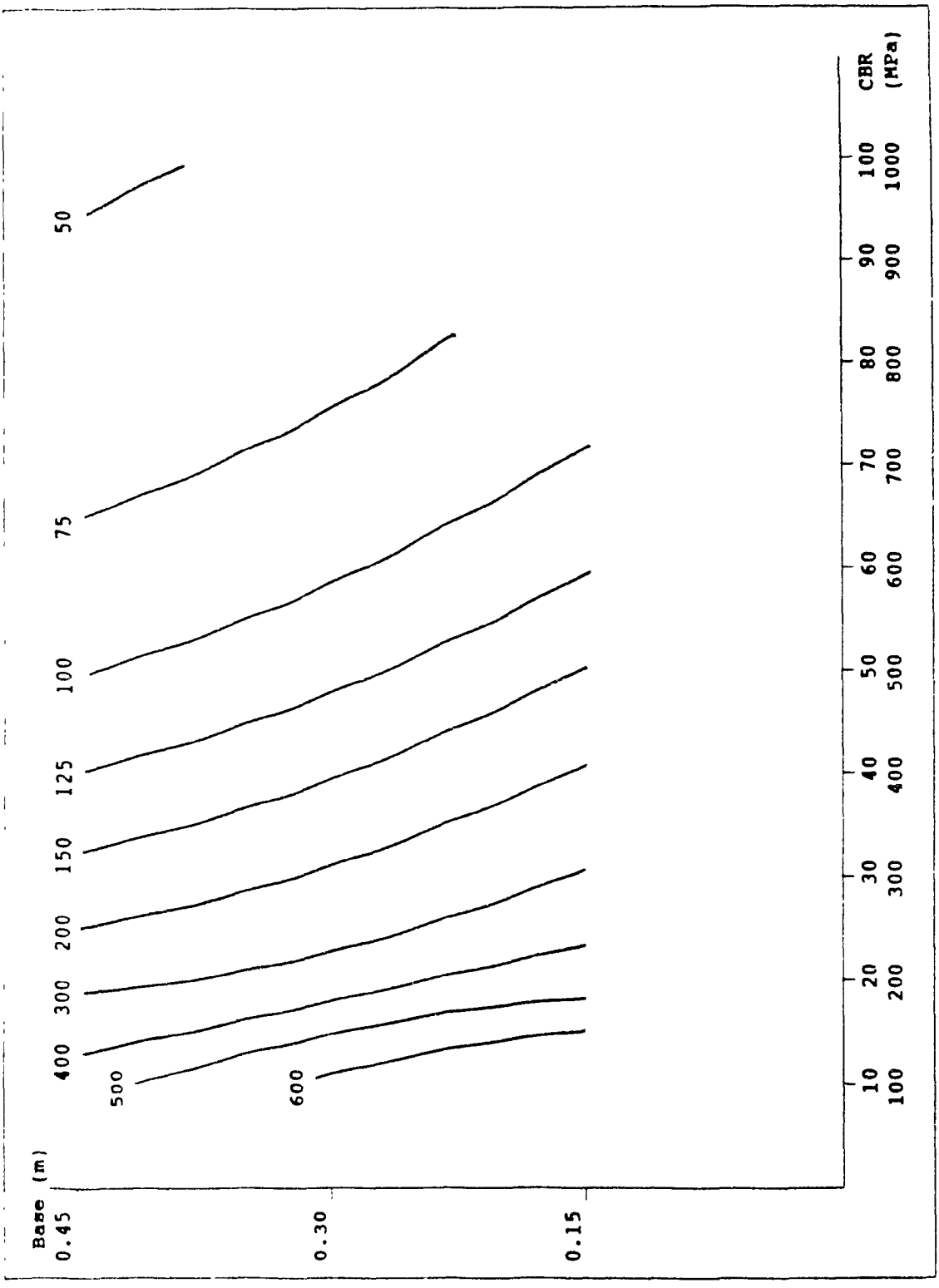


Figure 6.2: Strain Contour Chart (Von Mises)

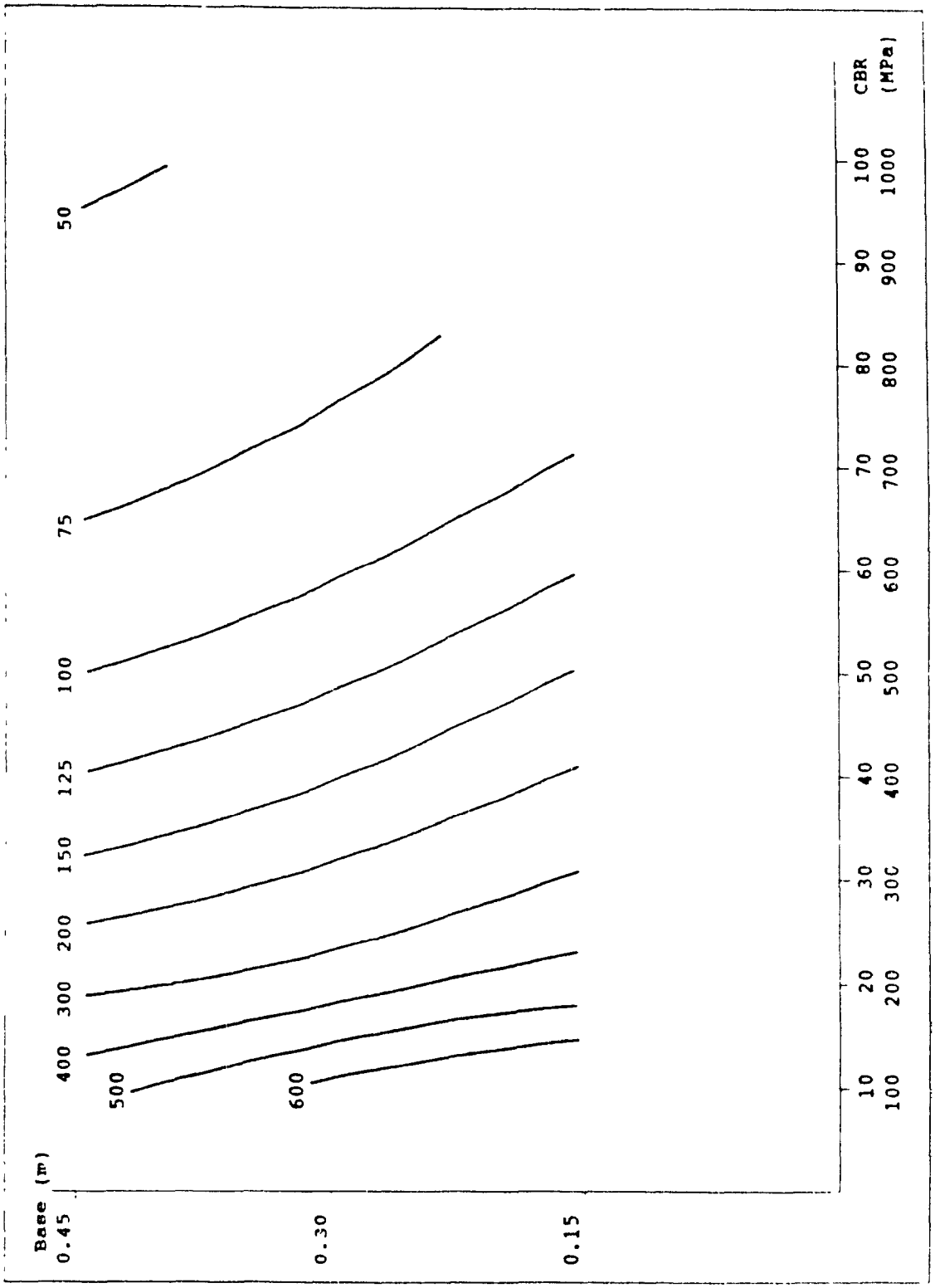


Figure 6.3: Strain Contour Chart (Mohr Coulomb)

deterioration of the pavement. Oglesby and Hicks (1982) give a detailed explanation of the serviceability index and its derivation.

Growth factors in Table 6.2 (Oglesby and Hicks, 1982) may also be used if required. These factors allow the calculation to take into account the annual increase in traffic over the design period. The total number of EAL in the design life of the pavement (N) can be found by:

$$N = S \times N_y \qquad (6.2)$$

where S is the design life (in years) and N_y is the load repetitions per year. Tables 6.1 and 6.2 are taken from Oglesby and Hicks (1982).

6.3.2 Determination of the Subgrade Soil Type

The soil behaviour such as linear elastic, Von Mises or Mohr Coulomb must be selected correctly if results are to be realistic. The behaviour can be determined from laboratory testing where values of parameters such as cohesion and angle of friction can indicate soil type.

6.3.3 Determination of Subgrade Soil Parameters

Using either laboratory or field tests, the parameters of the soil must be found. The governing soil criteria, modulus of elasticity and Poisson's ratio are necessary to adequately model the subgrade.

Table 6.1: Load Equivalence Factors (p = 2.5)

Loads (kN)	Single Axles Structural Number SN			Tandem Axle Sets Structural Number SN		
	1	4	6	1	4	6
9	0.0004	0.0002	0.0002			
26	0.01	0.01	0.01			
44	0.08	0.10	0.08	0.01	0.01	0.01
62	0.33	0.39	0.34	0.03	0.03	0.02
80	1.00	1.00	1.00	0.07	0.09	0.07
98	2.48	2.09	2.30	0.16	0.21	0.17
115	5.33	3.91	4.48	0.33	0.40	0.34
133	10.31	6.83	7.79	0.61	0.70	0.63
151	18.41	11.34	12.51	1.06	1.11	1.08
169	30.90	18.06	18.98	1.75	1.68	1.73
178	39.26	22.50	23.04	2.21	2.03	2.14
195				3.41	2.88	3.16
213				5.08	3.98	4.49

Table 6.2: Growth Factors

Design Period (yr)	Annual Growth Rate (%)					
	0	2	4	6	8	10
1	1.0	1.0	1.0	1.0	1.0	1.0
5	5.0	5.20	5.42	5.64	5.87	6.11
10	10.0	10.95	12.01	13.18	14.49	15.94
15	15.0	17.29	20.02	23.28	27.15	31.77
20	20.0	24.30	29.78	36.79	45.76	57.28

6.3.4 Determination of Base Thickness

The value of N can th n be inserted into the Asphalt Institute equation (Oglesby and Hicks, 1982):

$$\varepsilon_c = (N/1.36 \times 10^9)^{1/4.48} \qquad (6.3)$$

to determine the value of vertical strain ε_c corresponding to N. This ε_c can then be inserted into the appropriate contour chart to find the required base thickness.

For example, a pavement with the following parameters:

1. subgrade CBR = 30
2. subgrade ν = 0.1
3. vertical strain of 0.000241

will require a base thickness depending on each type of subgrade behaviour.

The base thicknesses for the three soil behaviours can then be found from Figures 6.1 to 6.3. These values are

a. linear elastic = 0.40 m.
b. Von Mises = 0.24 m.
c. Mohr Coulomb = 0.24 m.

It should be noted that since the Von Mises and Mohr Coulomb materials behave the same way in the elastic range, their base thicknesses would be identical.

6.3.5 Determination of Service Life

If the base width is specified or assumed, the service life (S) of the pavement can be predicted. For a

given base thickness, CBR, ν, and soil behaviour, the vertical strain can be found from the contours. This value can then be inserted into the following equation:

$$N = 1.36 \times 10^{9}(\varepsilon_{v})^{-4.48} \qquad (6.4)$$

to find the total number of load repetitions to failure. Then, based on the number of 80 kN load repetitions per year (N_{y}), the service life can be calculated as follows:

$$S = N/N_{y} \qquad (6.5)$$

Therefore, given the proper parameters, the base thickness can be designed or the design life calculated by combining the design curves using the Asphalt Institute's equation. The following section lists four numerical examples which show how these principles can be applied to design/analysis situations.

6.4 Numerical Examples

Example A: Determination of Base Thickness

Assume that a material with the following properties:

1. CBR = 30
2. ν = 0.2

should withstand 3000 standard 18,000 lb load applications/day for a service life of fifteen years. If the subgrade behaves like a Mohr Coulomb material, what is the required base depth?

Solution

Given the required daily traffic and service life, the number of load repetitions to failure is:

$$N = (3000 \text{ loads/day}) \times (365 \text{ days/yr}) \times (15 \text{ years})$$

$$= 16,425,000 \text{ load repetitions to failure}$$

and thus the corresponding vertical compressive strain would be:

$$\varepsilon_c = (16,425,000/1.36 \times 10^9)^{-1/4.48}$$

$$= 257 \times 10^{-6} \text{ m/m}$$

This value is inserted into the Mohr Coulomb contour chart shown in Figure 6.4 to find the base thickness. A vertical line is drawn from a CBR value of 30 to intersect a contour line corresponding to an ε_c value of 257 x 10_{-6}. From this point of intersection, a horizontal line is drawn to obtain a value of 0.30 m. for base thickness.

Example B: Determination of Service Life

Assume that a material with the following properties:

1. CBR = 40
2. ν = 0.3

should withstand 4000 load applications/day. If the base thickness is 0.30 m. and the subgrade is linear elastic, then determine the service life of the road.

Solution

Given the base thickness and subgrade properties, the vertical compressive strain (ε_c) value can be found in Figure 6.5. A vertical line is drawn from a CBR value of 40 and a horizontal

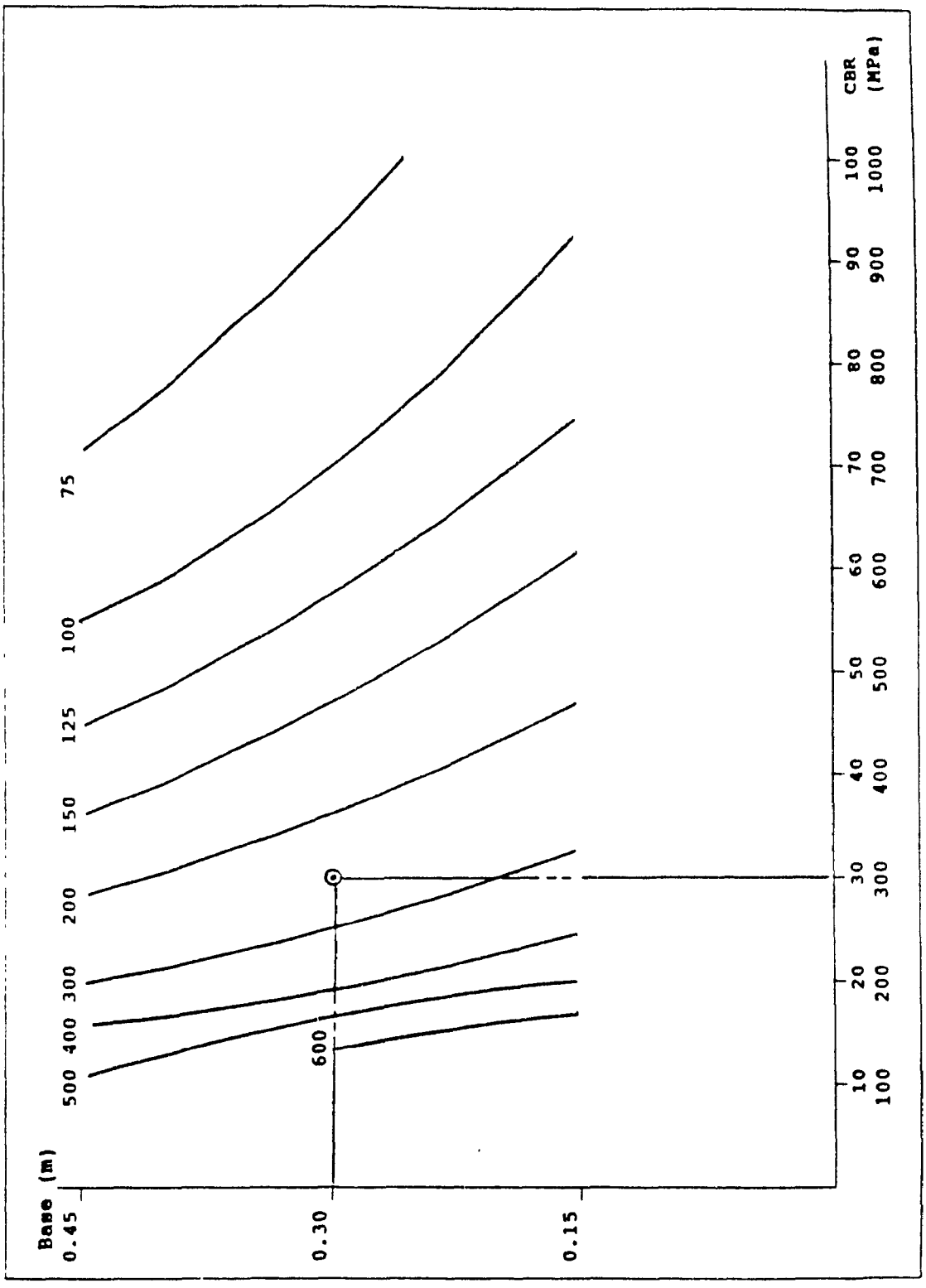


Figure 6.4: Example A

line from a base thickness of 0.30 m. The two lines intersect at a value of ε_c of 175 x 10^{-6}. Then by inserting the value of ε_c = 175 x 10^{-6} into the Asphalt Institute equation:

$$N = 1.36 \times 10^{9}(\varepsilon_c)^{-4.48}$$

$$= 1.36 \times 10^{9}(175 \times 10^{-6})^{-4.48}$$

= 92,199,475 load repetitions to failure

N_1 = 4000 x 365 = 1,460,000 loads/year

The service life can then be calculated by:

$$S = N/N_1$$

= 92,199,475/1,460,000

= 63.15 years

Example C: Determination of Additional Service Life

Assume that a material with the following properties:

1. CBR = 35
2. ν = 0.2
3. Mohr Coulomb behaviour

should withstand 5000 load applications/day for a service life of twenty years. Perform the following:

a. design the required base thickness for the above criteria

b. if the base were increased by 50 percent, what would be the extended service life?

Solution

Given the required daily traffic and service life, the number of load repetitions to failure is:

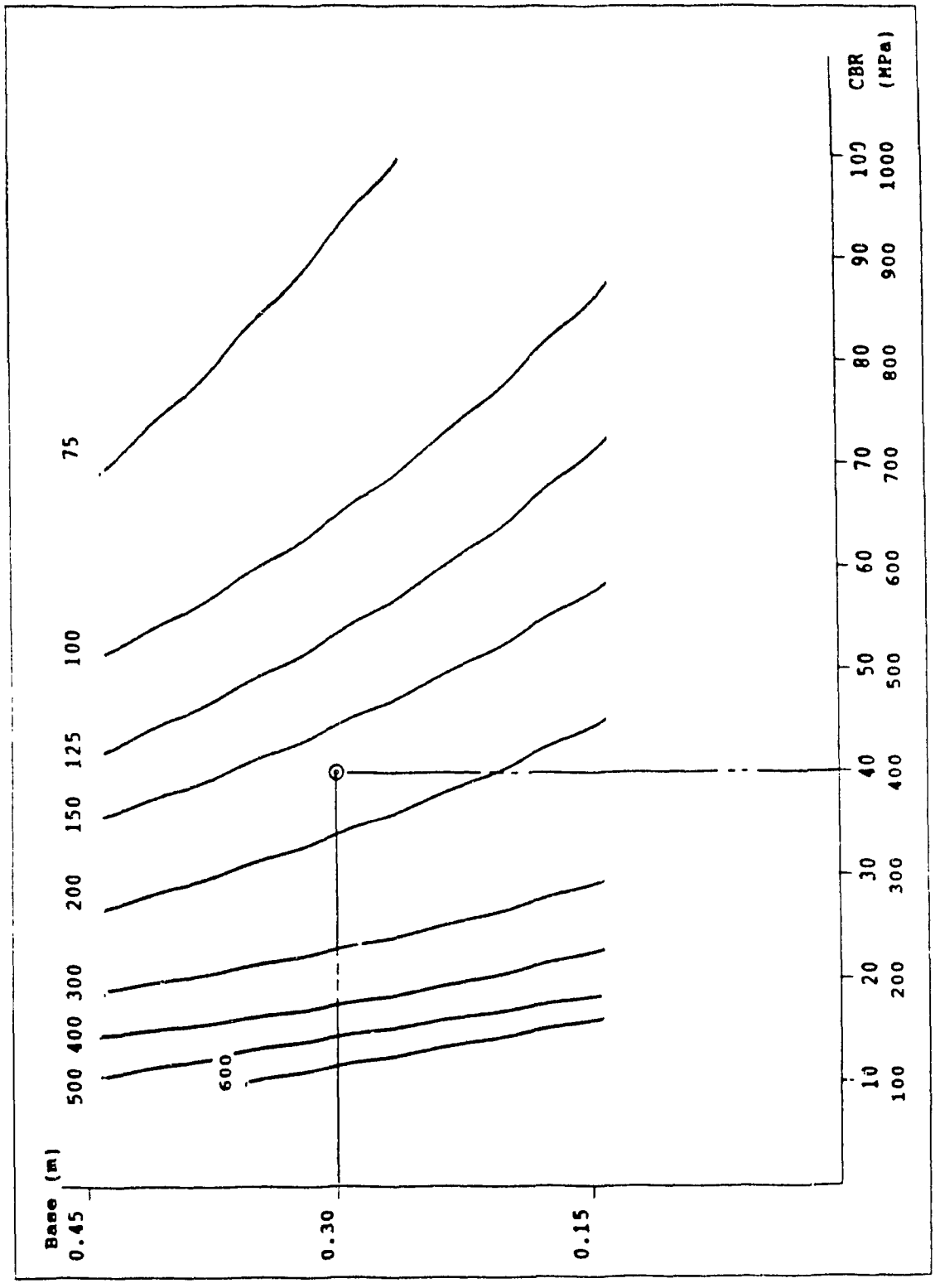


Figure 6.5: Example B

N = (5000 loads/day) x (365 days/yr) x (20 years)

= 36,500,000 load repetitions to failure

and thus the corresponding vertical compressive strain would be:

$\varepsilon_c = (N/1.36 \times 10^9)^{1/4.48}$

$= (36,500,000/1.36 \times 10^9)^{1/4.48}$

$= 0.000215$ m/m

$= 215 \times 10^{-6}$ m/m

This value can then be inserted into the Mohr Coulomb contour chart shown in Figure 6.6 find the base thickness. A vertical line is drawn from a CBR value of 35 to intersect a contour line corresponding to an ε_c value of 215×10^{-6}. From this point of intersection, a horizontal line is drawn to obtain a value of 0.27 m. for base thickness. If the depth is increased by 50 percent to 0.40 m. then the corresponding ε_c would be 1.67×10^{-6} from Figure 6.7. Inserting this value into the equation:

$N = 1.36 \times 10^9(\varepsilon_c)^{-4.48}$

$= 1.36 \times 10^9(0.000167)^{-4.48}$

113,702,265 load repetitions to failure

N_t = 5,000 x 365 = 1,825,000 loads/year

Then given the traffic criteria, the service life would be:

$S = N/N_t$

= 113,702,265/1,825,000

= 62.3 years

Therefore for an increase of 50 percent of base thickness, the

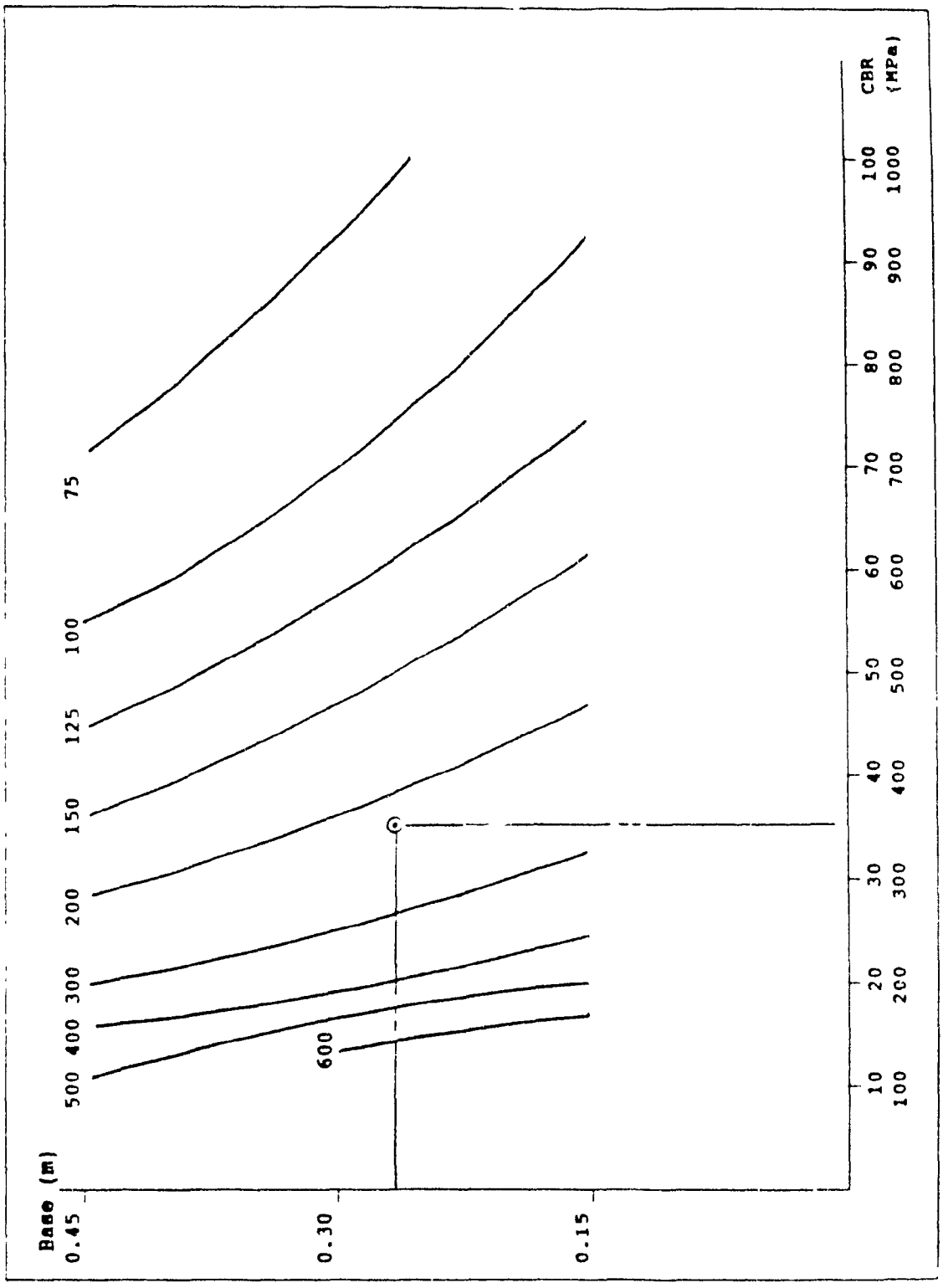


Figure 6.6: Example C (Part 1)

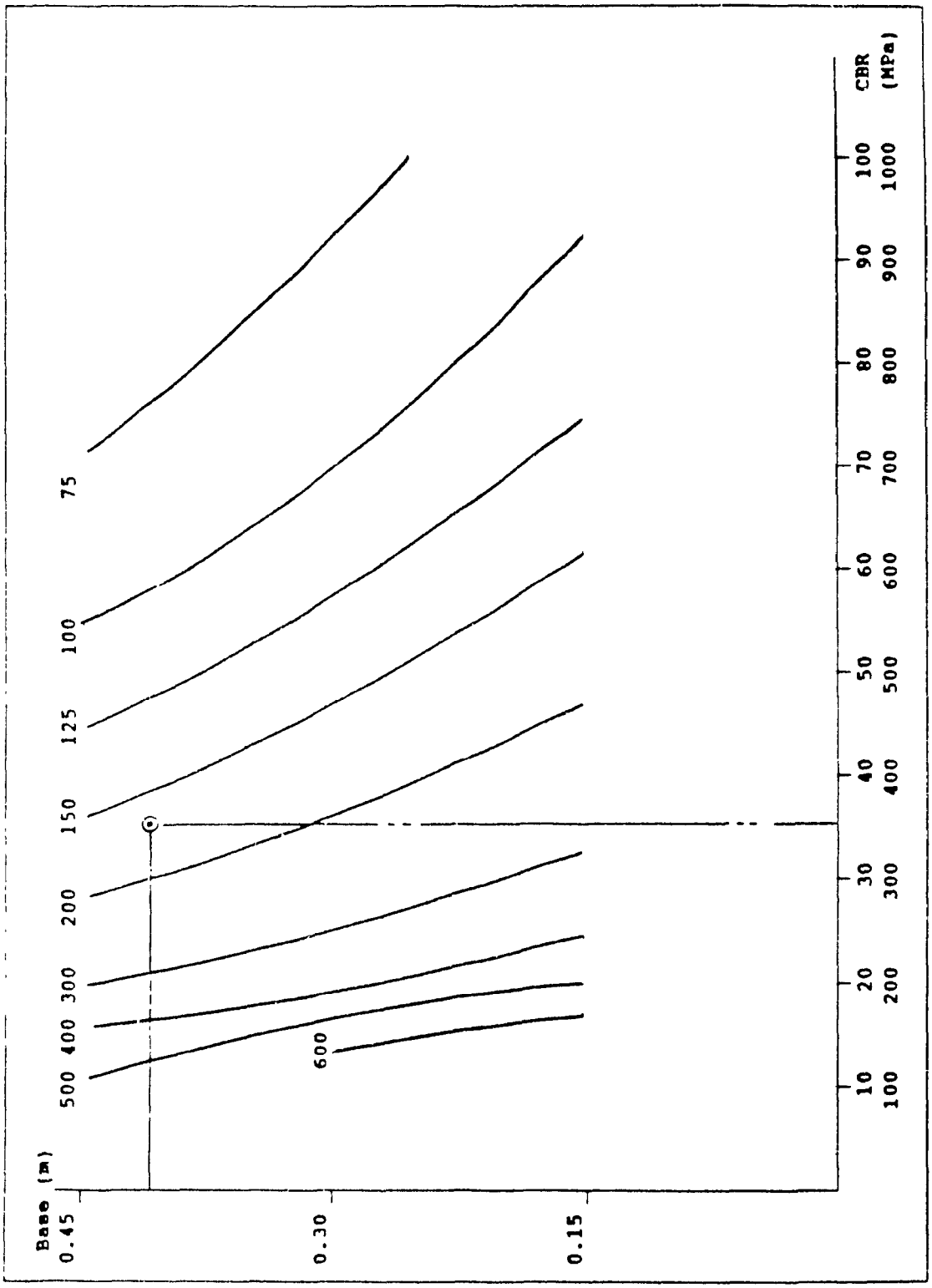


Figure 6.7: Example C (Part 2)

service life is increased 3.1 times.

Example D: Determination of Remaining Service Life

Given a three year old road with the following properties:

1. CBR = 20
2. ν = 0.3
3. Von Mises behaviour

This road has a base thickness of 0.25 m. and is subjected to 1,000 standard 18,000 lb load applications per day. Calculate the remaining service life of the road.

Solution

The value of vertical strain ε_c is found from Figure 6.8. A vertical line is drawn from the CBR value of 20 and a horizontal line from a base thickness of 0.25 m. The intersection point gives the value of $\varepsilon_c = 450 \times 10^{-6}$. Given ε_c, the number of load repetitions to failure is:

$$N = 1.36 \times 10^{9}(\varepsilon_c)^{-4.48}$$

$$= 1.36 \times 10^{9}(0.000450)^{-4.48}$$

$$= 1,340,130 \text{ load repetitions to failure}$$

The number of load repetitions accumulated during the first three years of the road's life is:

$$N_3 = (1000 \text{ loads/day}) \times (365 \text{ days/yr}) \times (3 \text{ years})$$

$$= 1,095,000 \text{ load repetitions}$$

and thus the remaining load repetitions to failure are:

$$N_r = 1,340,130 - 1,095,000$$

$$= 245,130 \text{ load repetitions to subgrade failure}$$

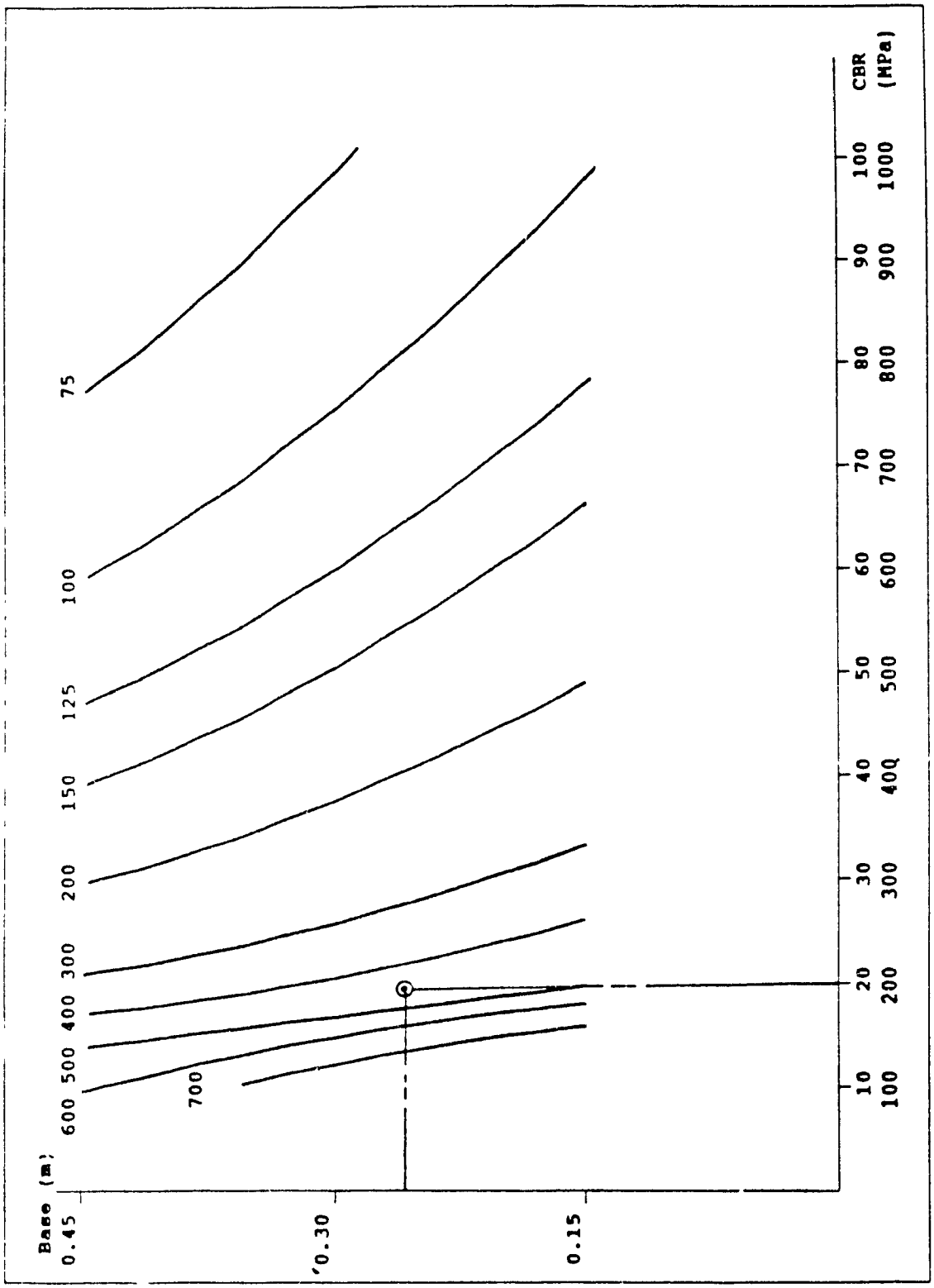


Figure 6.8: Example D

The remaining service life is:

$$S = 245{,}130/(1000 \text{ loads/day} \times 365 \text{ days/year})$$

$$= 0.67 \text{ years}$$

Therefore, the subgrade will fail in 0.67 years.

Chapter 7

Computer Programs

7.1 Introduction

The programs for finite element analysis are written in QuickBasic 4.5 for operation on personal computers. These programs are based on original FORTRAN programs written for mainframe IBM computers (Smith, 1982). Several modifications have been undertaken to suit the requirements of this project. The following sections give flowcharts and explanations of the programs. Computer printouts are given in the appendices.

7.2 Modifications to the Original Programs

The programs are rewritten into QuickBasic 4.5 and then compiled into executable files. The programs have been translated so as to integrate them into a menu driven system. That is, the original programs require all necessary data to be input manually. A menu system allows the program to guide the user through data entry and given some input can automatically generate the remainder. It can also prompt the user as to which program to use or given the soil parameters, select and run the appropriate program automatically.

7.3 Constraints in the Translation

The major constraint with QuickBasic 4.5 (as well as with its parent, BASIC) is the 64K memory limitation. That is, both the data generated and the program itself cannot occupy more than 64 kilobytes of memory.

In order to conserve memory, the main program is broken down into a series of subprograms each chained together. The main cause of memory overload is due to large intermediate matrices. With a series of subprograms, each segment will start with zero memory, create its own intermediate matrices, and then store only the data needed for the next subprogram. During the chaining operation, all intermediate matrices are erased, so that each subprogram can start with zero memory. The linear elastic program is divided into four such subprograms while the elasto-plastic programs are divided in three. Figure 7.1 shows a flow chart of the overall finite element programs. It should be noted that once all three behaviours are integrated with the menu system, the programs essentially become one program.

7.4 Other Modifications

Other changes include the substitution of the Choleski matrix reduction for the Gaussian method because the former is readily available in Smith (1982).

The original Mohr Coulomb program requires that deformation increments be used rather than the actual loads on the mesh. This requirement is not consistent with the other two programs so sections that converted the deformations to loads are modified so that the loads are directly used by the program.

The Mohr Coulomb is also modified to take into account soil failure on the first load cycle. The program as

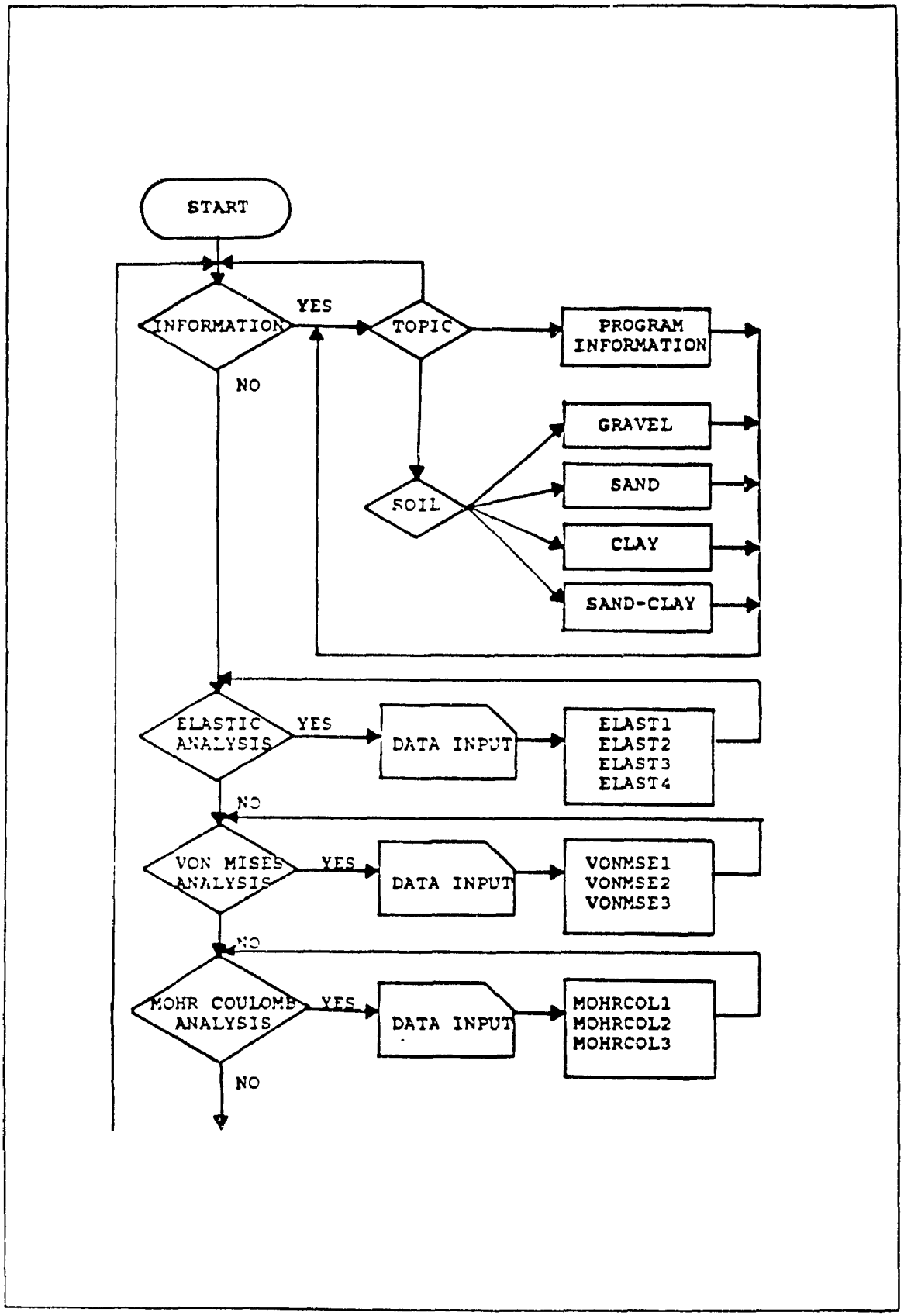


Figure 7.1: Flowchart of Integrated Programs

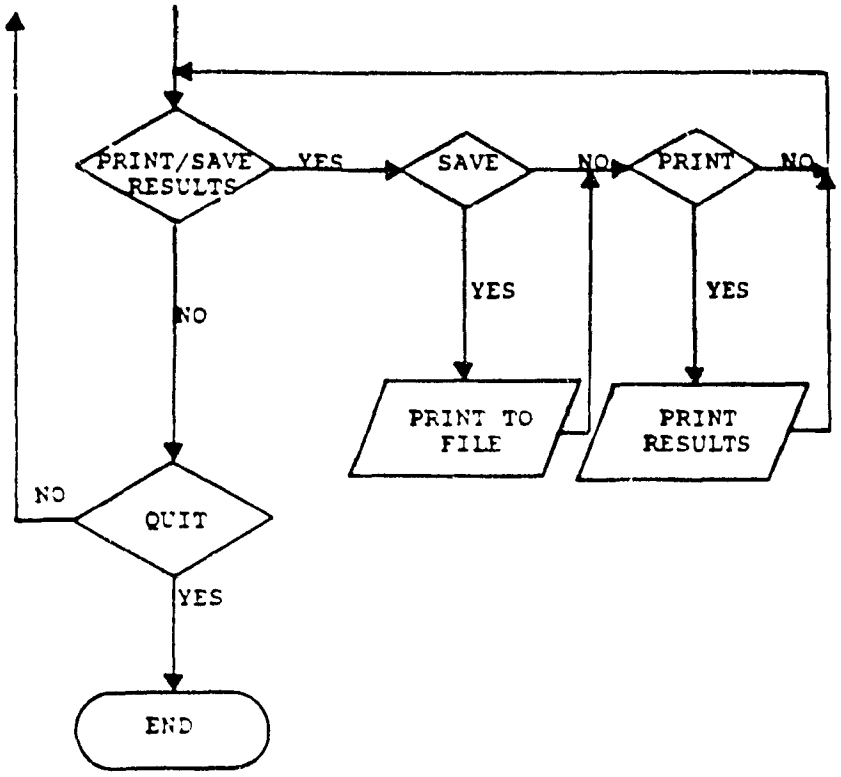


Figure 7.1 (Con't): Flowchart of Integrated Programs

originally written fails to allow for this situation and will "crash" in this event.

7.5 Program Instructions

The program may be started by entering FINITE and pressing the return/enter key. The initial menu screen is shown in Figure 7.2. Each topic in the upper part of the screen can be accessed by pressing the left or right cursor key until the topic cell is highlighted. Once the cell is reached, it may be accessed by pressing the return/enter key.

When the program is started, it initially defaults to the leftmost cell (information). If this topic is selected, then a submenu column will appear directly under the cell. The left right keys are now deactivated and the up/down cursor keys can be used to scan the menu. The program cell will display information about the program and a set of instructions for its operation. The soils cell accesses a second menu column which allows the user to retrieve information about various soil types including typical soil parameters. The final cell in both column menus allows the user to return to the previous menu. Thus, in the soils menu the final cell allows the user to return to the information menu. The final cell or the information menu allows the user to the main menu where the up and down keys are deactivated and the left and right keys can be used. Figure 7.3 shows the information screens.

The next three cells of the main menu allow the user

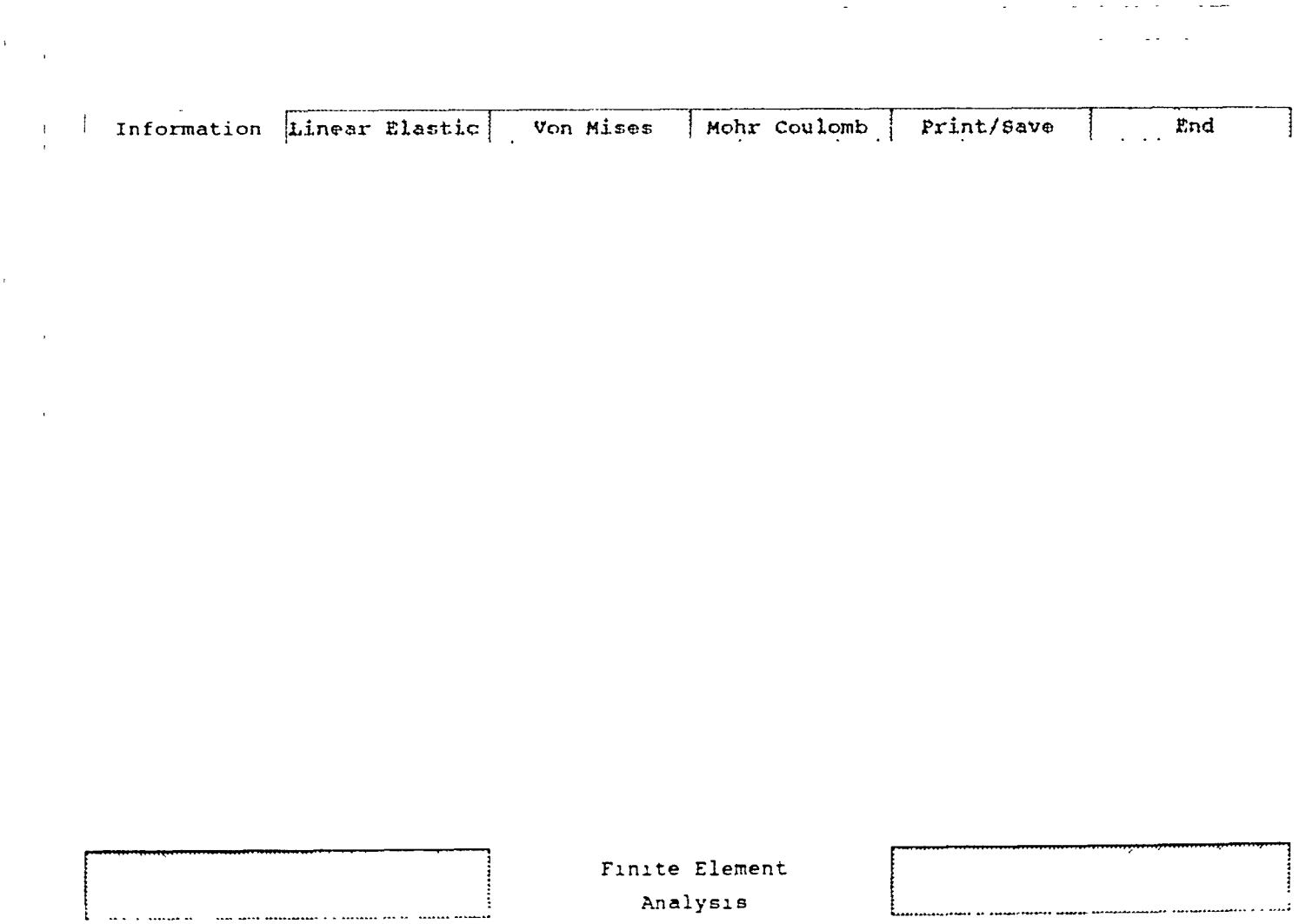


Figure 7.2: Initial Menu Screen

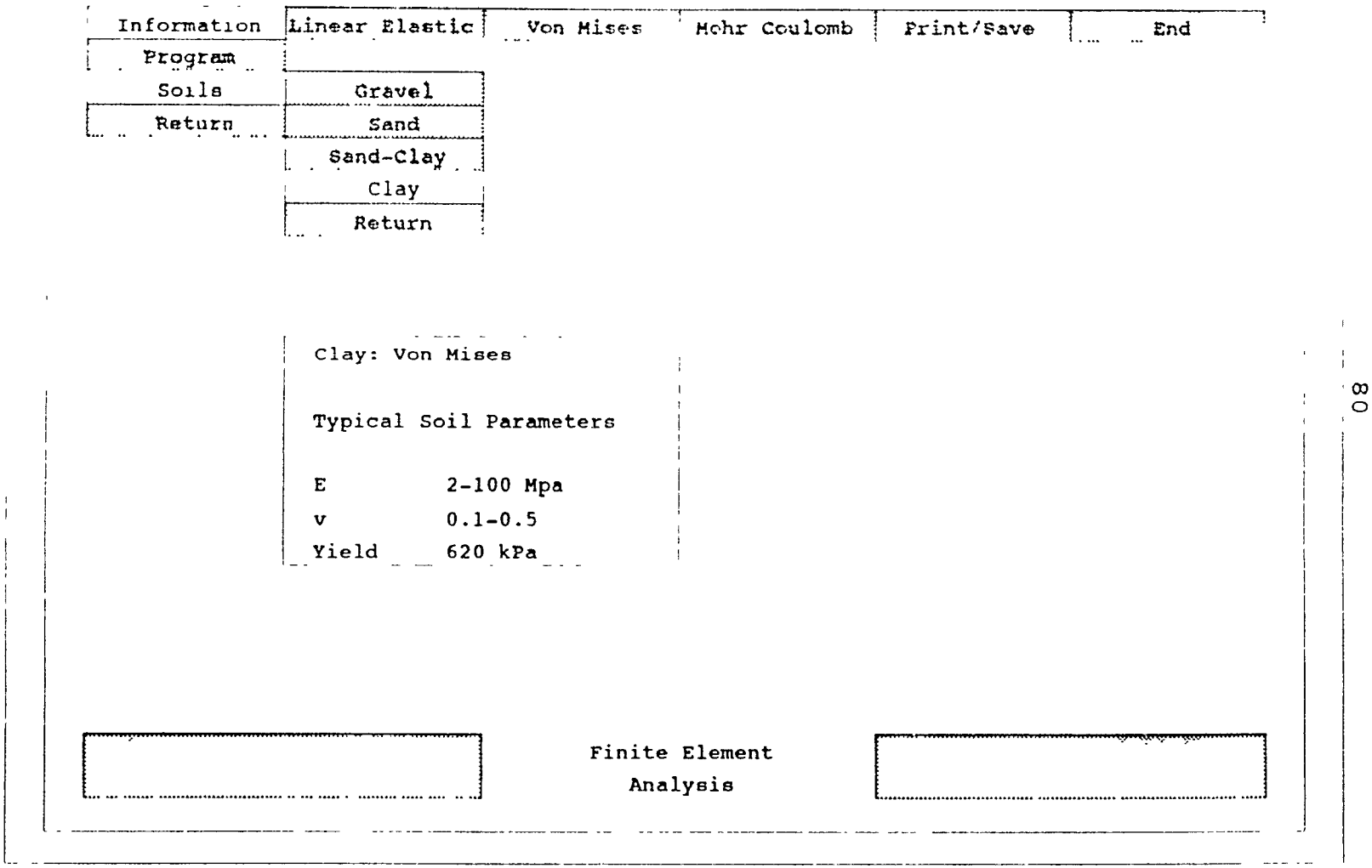


Figure 7.3: Soil Information Screen

to select the appropriate method of analysis. The format of the three cells is identical. The column menu includes a data input, program start and return to main menu cells. The data input cell allows for input directly from the keyboard or from a data file. The keyboard data entry requires that only the general model parameters be given. The more tedious entries such as the restrained node freedoms are generated automatically, saving time and preventing input errors. Figures 7.4 and 7.5 show typical input screens while Figures 7.6 to 7.8 show input requirements. The print save screen allows the user to print the results on paper, save the results to a data file, or both.

Once work is completed, the user may exit to the system by accessing the end cell.

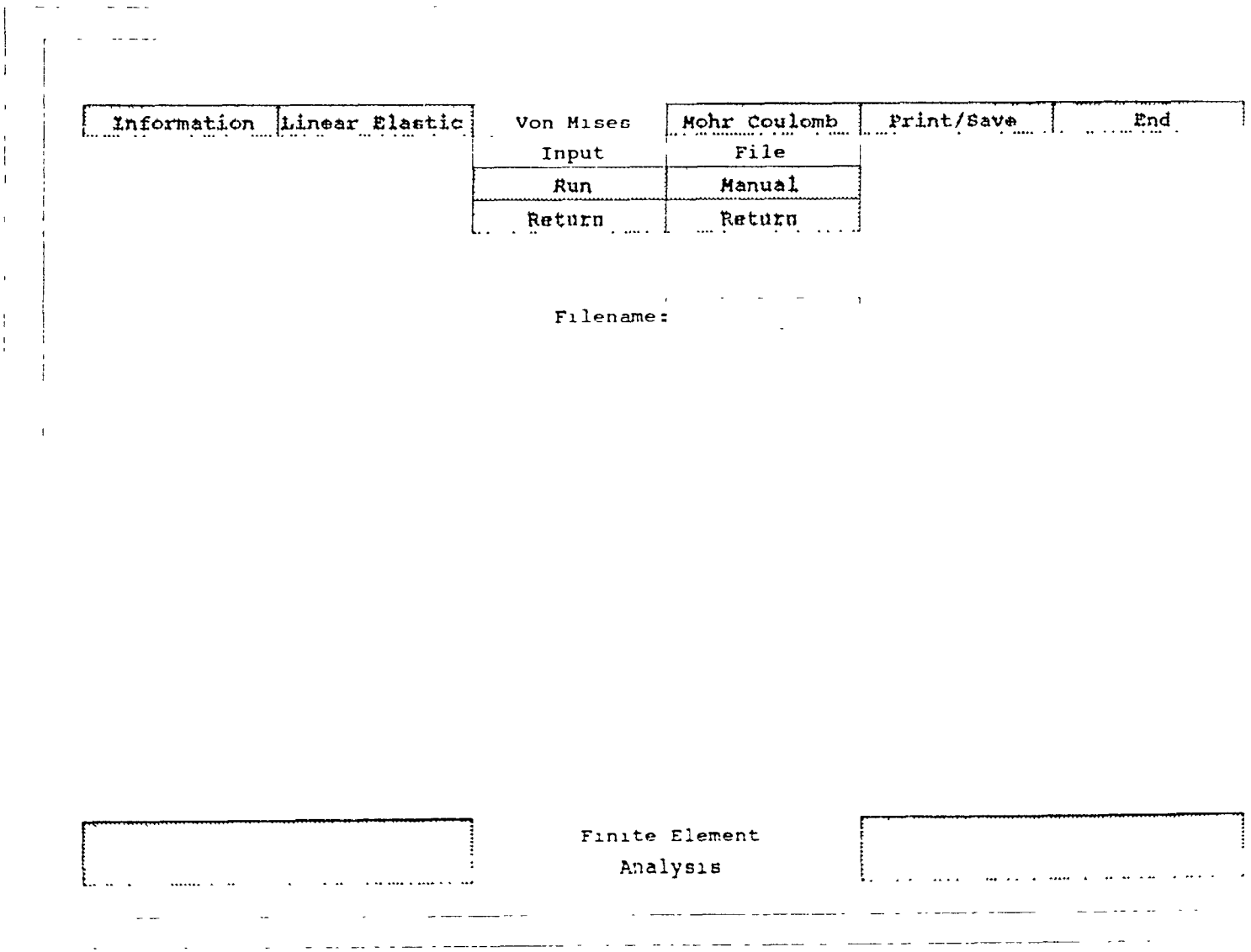


Figure 7.4: Datafile Input Screen

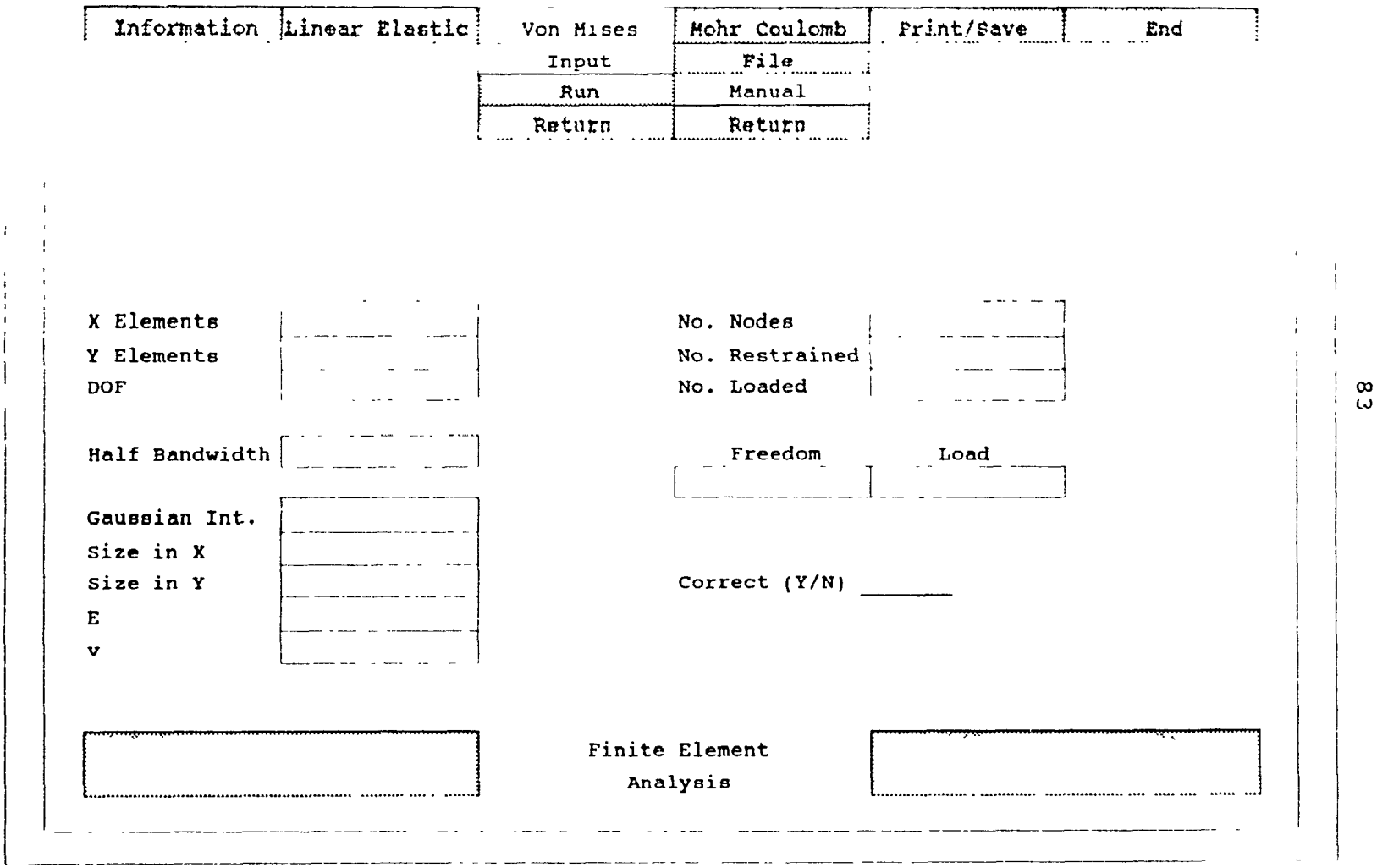
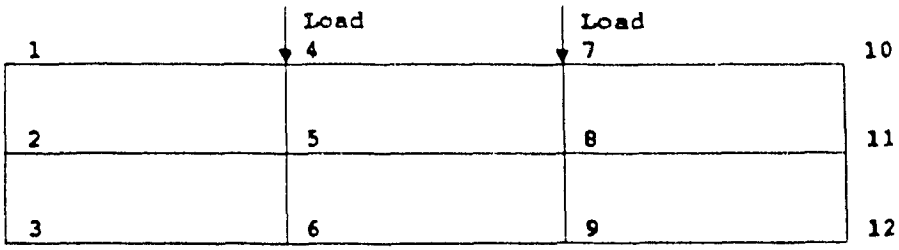


Figure 7.5: Manual Data Input Screen



Degrees of Freedom

Node	x	y	Node	x	y
1		1	7	7	8
2		2	8	9	10
3			9		
4	3	4	10		11
5	5	6	11		12
6			12		

Continuum Data

NXE	# elements in x-direction	3
NYE	# elements in y-direction	2
N	# degrees of freedom	12
W	half bandwidth	7
NN	# nodes	12
RN	# restrained nodes	8
NL	# loaded nodes	2

Element Data

GP	Gaussian integration order	2
AA	element size in x-directic	-
BB	element size in y-directic	-
E	Young's modulus	-
v	Poisson's Ratio	-

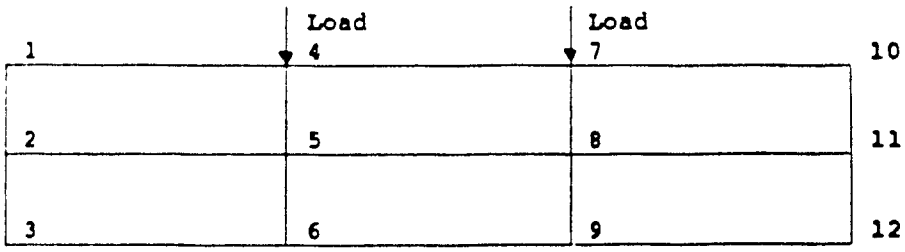
Node Freedom Data (node freedom-x freedom-y)

```
 1  1  0  2  1  0  3  1  1  6  1  1
 9  1  1 10  1  0 11  1  0 12  1  1
```

Load Data

```
 4       Load       8       Load
```

Figure 7.6: Data for Linear Elastic Program



Degrees of Freedom

Node	x	y	Node	x	y
1		1	7	7	8
2		2	8	9	10
3			9		
4	3	4	10		11
5	5	6	11		12
6			12		

Continuum Data

NXE	# elements in x-direction	3
NYE	# elements in y-direction	2
N	# degrees of freedom	12
W	half bandwidth	7
NN	# nodes	12
RN	# restrained nodes	8
NL	# loaded nodes	2
INCS	# load increments	-
ITS	# stress redistribution it	-

Element Data

GP	Gaussian integration order	2
AA	element size in x-directic	-
BB	element size in y-directic	-
v	Poisson's Ratio	-
E	Young's modulus	-
SBARY	von Mises yield stress	-

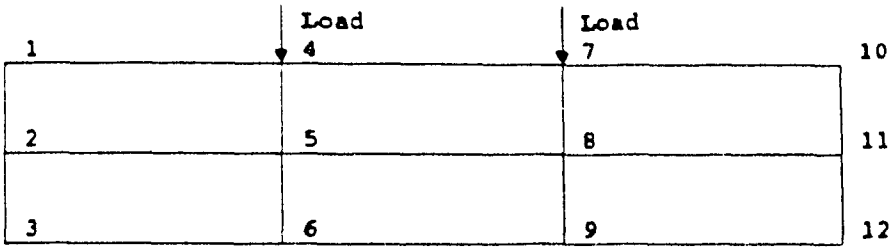
Node Freedom Data (node freedom-x freedom-y)

1	1	0	2	1	0	3	1	1	6	1	1
9	1	1	10	1	0	11	1	0	12	1	1

Load Data

4	Load	8	Load

Figure 7.7: Data for Von Mises Program



Degrees of Freedom

Node	x	y	Node	x	y
1		1	7	7	8
2		2	8	9	10
3			9		
4	3	4	10		11
5	5	6	11		12
6			12		

Continuum Data

NXE	# elements in x-direction	3
NYE	# elements in y-direction	2
N	# degrees of freedom	12
W	half bandwidth	7
NN	# nodes	12
RN	# restrained nodes	8
NL	# loaded nodes	2

Element Data

GP	Gaussian integration order	2
AA	element size in x-directic	-
BB	element size in y-directic	-
v	Poisson's Ratio	-
E	Young's modulus	-
INCS	# load increments	-
ITS	# stress redistribution it	-
COH	soil cohesion	-
PHI	soil friction angle	-
PSI	soil dilation angle	-

Node Freedom Data (node freedom-x freedom-y)

1 1 0 2 1 0 3 1 1 6 1 1
9 1 1 10 1 0 11 1 0 12 1 1

Load Data

4 Load 8 Load

Figure 7.8: Data for Mohr Coulomb Program

Chapter 8

Conclusions

The research work described in this thesis develops a pavement design method based on finite element analysis of the subgrade and equations used by the Asphalt Institute. This chapter summarizes the conclusions developed during the research work and offers some suggestions for further study.

8.1 Conclusions

The conclusions of the study are:

1. A constitutive model in conjunction with the finite element method can be used to predict vertical displacements and strains in the subgrade. The methodology can then be used to predict the thickness of the base and life of the subgrade using the Asphalt Institute design method. The subgrade soil need not be assumed elastic as is done in the Asphalt Institute method. Other assumptions such as elasto plastic behavior of soil can be made as shown in this thesis.
2. A fundamental relationship exists between the vertical strain produced at the surface of the subgrade and the combination of axle load and base thickness. The strain contours developed can be used for finding the combination of base

thickness and axle load that can produce a given magnitude of vertical strain.

3. A fundamental relationship also exists between the vertical strain produced at the surface of the subgrade and the combination of base thickness, modulus of elasticity, and Poisson's ratio. Strain contour charts developed can be used for finding the required base thickness given the traffic volume for the pavement design life and a combination of modulus of elasticity and Poisson's ratio. The life of the subgrade can be determined given a combination of base thickness, modulus of elasticity and Poisson's ratio.
4. The stress strain relationship is identical for the Von Mises and Mohr Coulomb soils in the elastic range. The linear elastic relationship differs slightly.
5. In all three cases, the values of modulus of elasticity and Poisson's ratio governed the critical strain calculated. Changes in cohesion and angle of internal friction parameters did not significantly alter the value of vertical strain and therefore are not considered.
6. A range of loads are not necessary in this analysis, since the design method utilizes the

standard 18,000 lb. axle load as input.

In conclusion, the development of the design contour charts is a significant accomplishment of the research. This design method differs from those presented elsewhere, in that it allows for the direct input of soil type and parameters into the contour charts. Unlike the Asphalt Institute method, base thicknesses can be calculated. This design method is a useful tool for the design of flexible pavements

8.2 Topics for Further Research

The determination of vertical strains on the subgrade using constitutive equations and the finite element method is an accurate means of predicting the behaviour of pavement structures. However, there are numerous refinements and new avenues of research which should be explored.

The finite element programs themselves can be improved. While QuickBasic 4.5 was used to make the programs more user friendly, there are other languages(such as C or C++) that will work just as well but which do not have the 64K memory constraint. Therefore, with these languages, larger matrices can be handled.

The programs can also be written to take into account other soil behaviours such as Drucker-Prager, cap, or variable moduli problems. An extended set of soil behaviours will make the program much more flexible. The program can also be modified so that given the soil criteria, the program can select the appropriate method of analysis automatically.

This modification will tend to reduce the likelihood of the wrong method being chosen.

The finite element method can also be applied to other analysis situations. For example, an analysis can be performed should existing soil parameters change due to infiltration of pollutants or from an accidental toxic spill. Climactic factors such as water infiltration or heat can be included in the analysis program.

References

1. Ashtakala, B., H.B. Poorooshasb, "Prediction of Tensile Cracks in Road Pavements", Mathematical Computer Modelling, 1989, vol 12, No. 1 pp. 55-60.

2. Bowles, Joseph E., Foundation Analysis and Design, (New York: McGraw-Hill Book Company 1982), Personal Library.

3. Bowles, Joseph E., Physical and Geotechnical Properties of Soils, (New York: McGraw-Hill Book Company 1984), Personal Library.

4. Desai C.S., H.. Siriwardane, Constitutive Laws for Engineering Materials With Emphasis on Geologic Materials(New Jersey: Prentice-Hall Inc.,1984), TA 417.6 D47 1984 SEL.

5. Heinrichs, K.W., M.J. Liu, M.I. Darter, S.H. Carpenter, A.M. Ioannides, "Rigid Pavement Analysis and Design", Illinois University-Urban-Champaign Federal Highway Administration FDWA-RD-88-068 NCP 3C1b-1012, June 1988.

6. Lay, M.G., Handbook of Road Technologies - Planning the Pavements, (New York: Gordon and Breach 1986), TE 145 L38 1986v1 SEL.

7. Marchionna, A., M.G. Fornaci, M. Malgarini, "Evaluation of Flexible Pavements and Overlay Design Based on FWD Tests", Proceedings of the 6th International Conference of Structural Design of Asphalt Pavements held in Ann Arbor Michigan, 13-17 July 1987, (University of Michigan), vol 1, pp. 628-637.

8. Oglesby, Clarkson H., R. Gary Hicks, Highway Engineering, (New York: John Wiley and Sons 1982), Personal Library.

9. Pandey, .B., "Fatigue Cracking of Bituminous Pavements", Journal of the Indian Roads Congress, September 1990, vol. 51-2, pp. 353-386.

10. Ritter, Leo J., Radnor J. Paquette, Highway Engineering, (New York: Ronald Press Company 1967), Personal Library.

11. Saraf, C., K. Marshak, H. Chan, R. Connel, W.R. Hudsen, "The Effect of Tire Contact Pressure Distribution on the Design of Flexible Pavements", Proceedings of the 6th International Conference of Structural Design of Asphalt Pavements held in Ann Arbor Michigan 13-17 July 1987, (University of Michigan), vol 1, pp. 180-190.

12. Siriwardane, H.J., C.S. Desai, "Computational Procedures for Non-Linear Three-Dimensional Analysis with Some Advanced Constitutive Laws", International Journal for Numerical and Analytical Methods in Geomechanics, 1983, vol. 7 pp. 143 - 171.

13. Smith, I.M., Programming the Finite Element Method with Applications to Geomechanics, (Chichester: John Wiley and Sons, 1982), Personal Library.

14. Smith, I.M., D.V. Griffiths, Programming the Finite Element Method, (Chichester: John Wiley and Sons, 1988), Personal Library.

15. Sweere, G., A. Penning, E. Vos, "Development of a Structural Design Procedure for Asphalt Pavements with Crushed Rubble Base Course", Proceedings of the 6th International Conference of Structural Design of Asphalt Pavements held in Ann Arbor Michigan 13-17 July 1987, (University of Michigan), vol 1 pp. 34-39.

16. Ullidtz, P., Pavement Analysis, (Amsterdam: Elsevier 1987), TE 250 U45 1987 Sel.

17. Weaver, William, Paul R. Johnston, Finite Elements for Structural Analysis, (New Jersey: Prentice-Hall 1984), Personal Library.

Appendix-A

Strain Contour Charts

This Appendix contains the remaining strain contour charts utilized by the design method which were not included in the body of this thesis.

The strain contour charts shown here include:

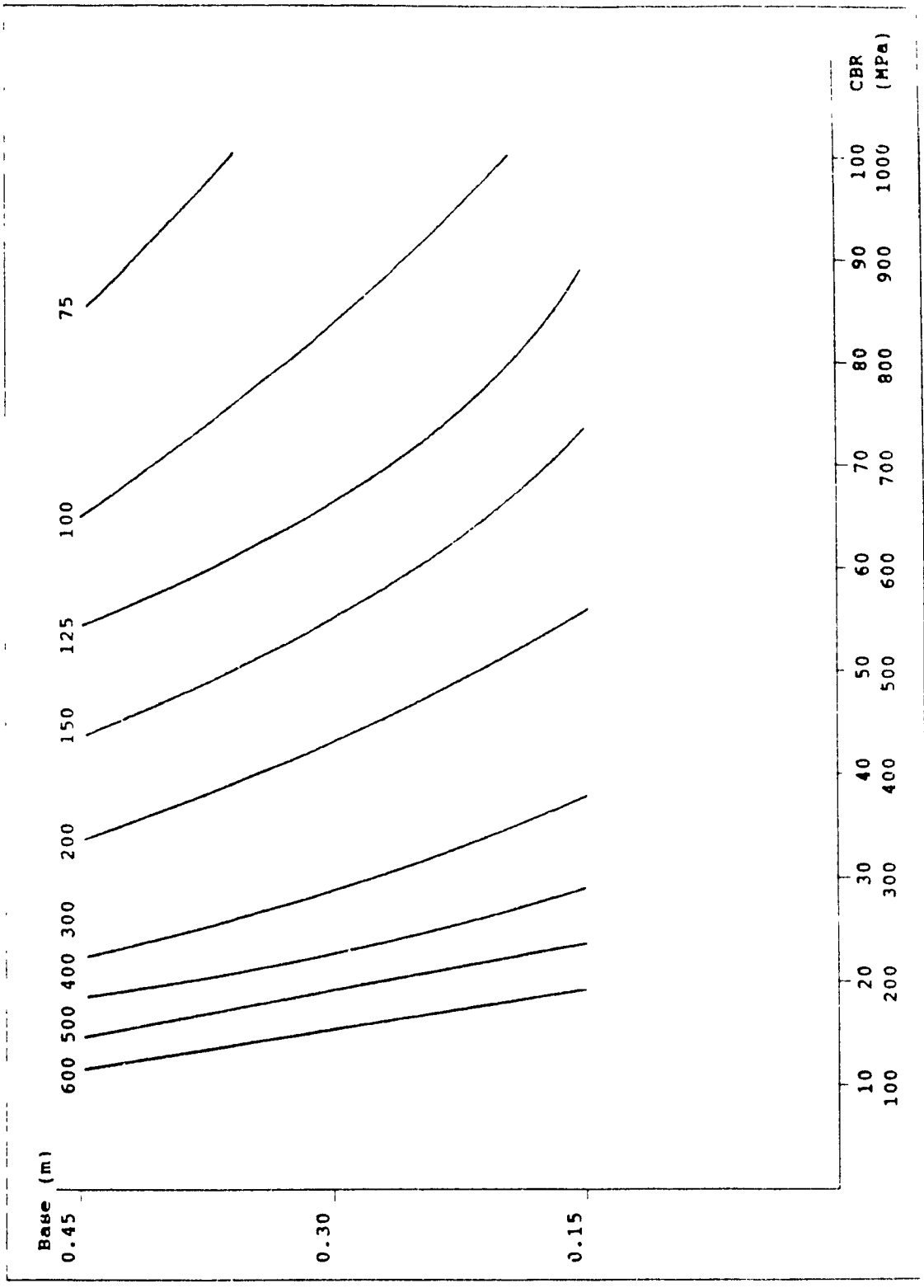


Figure A.1: Strain Contour Chart (Linear Elastic v=0.1)

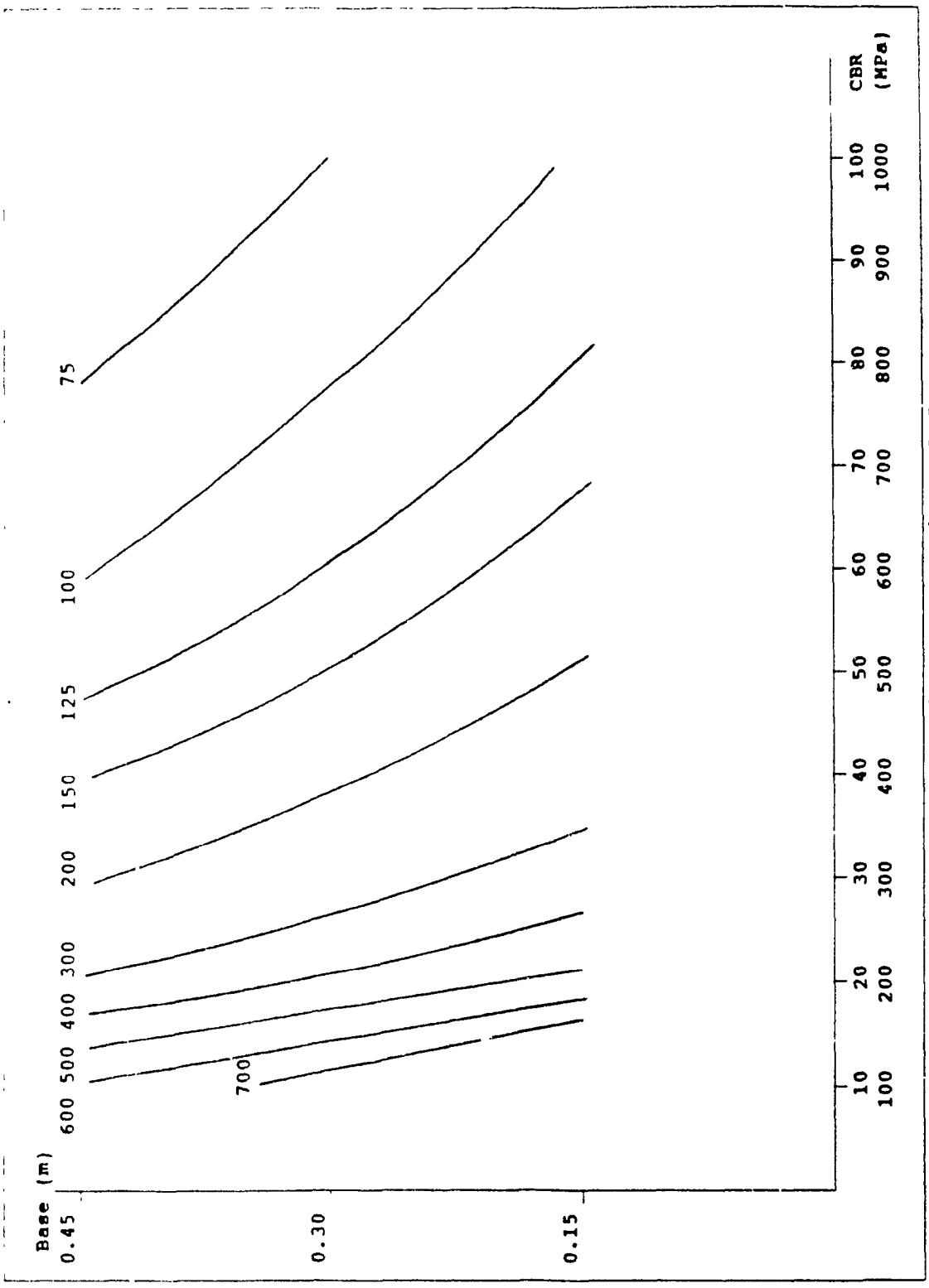


Figure A.2: Strain Contour Chart (Linear Elastic v=0.2)

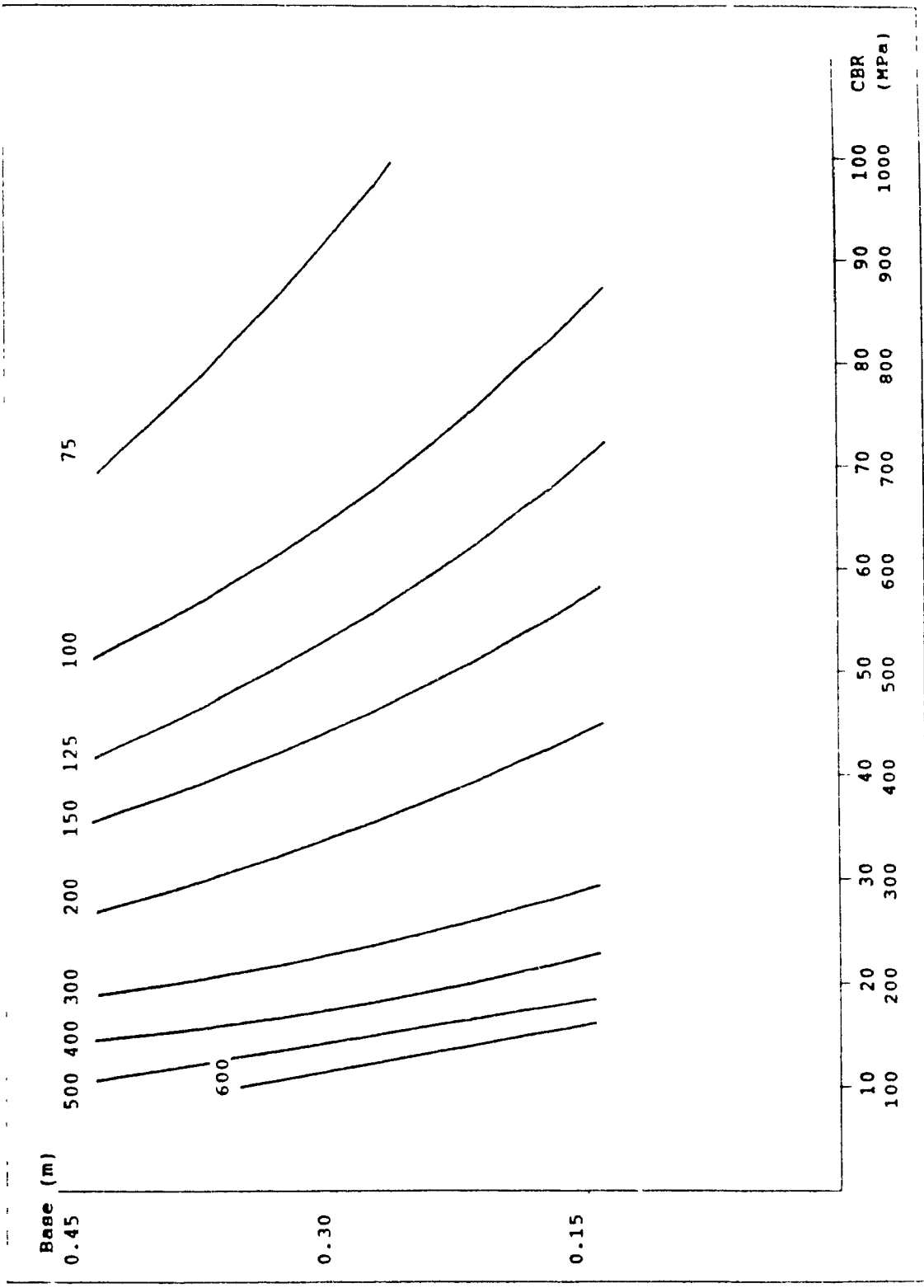


Figure A.3: Strain Contour Chart (Linear Elastic v=0.3)

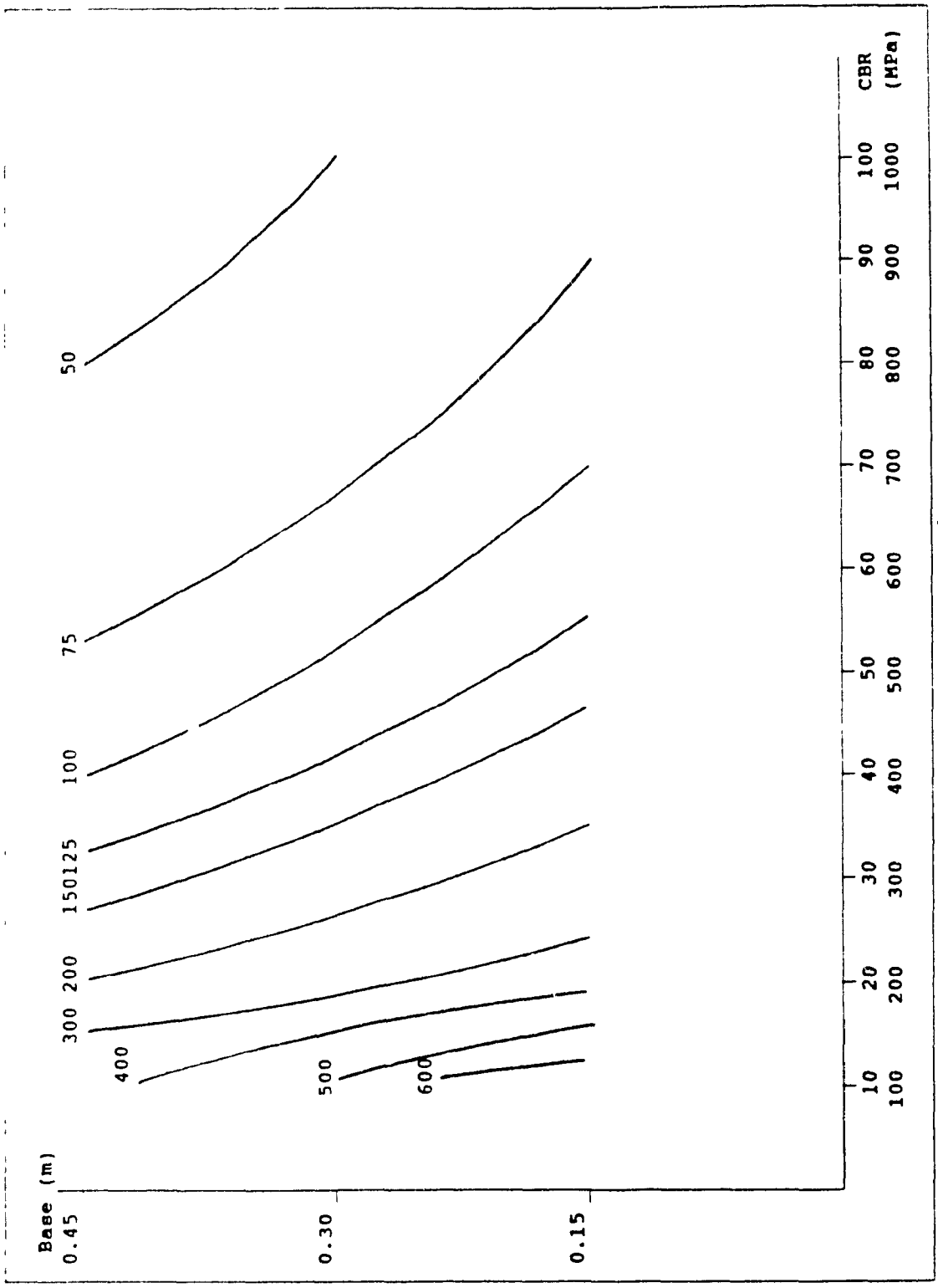


Figure A.4: Strain Contour Chart (Linear Elastic v=0.4)

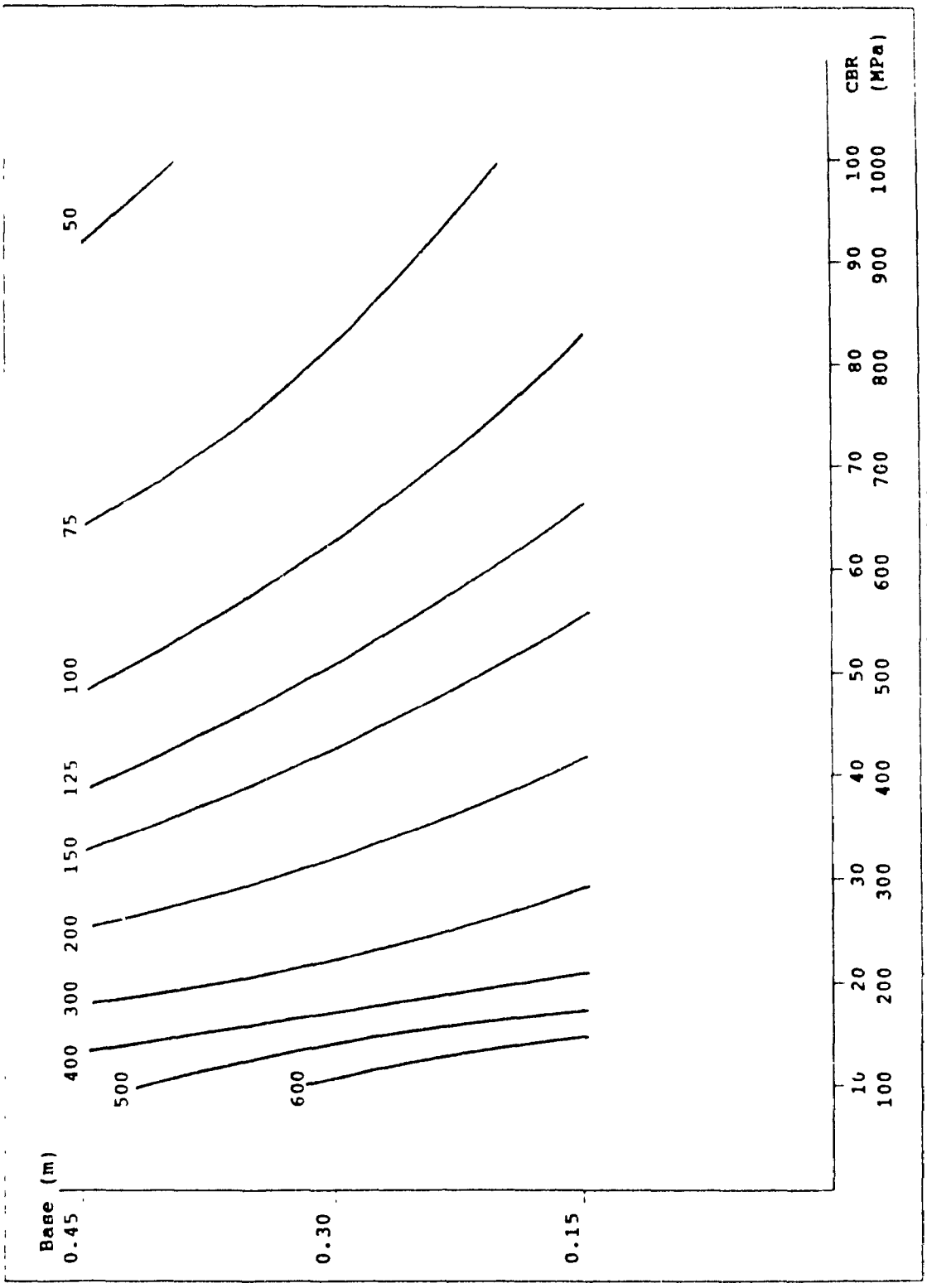


Figure A.5: Strain Contour Chart (Von Mises v=0.1)

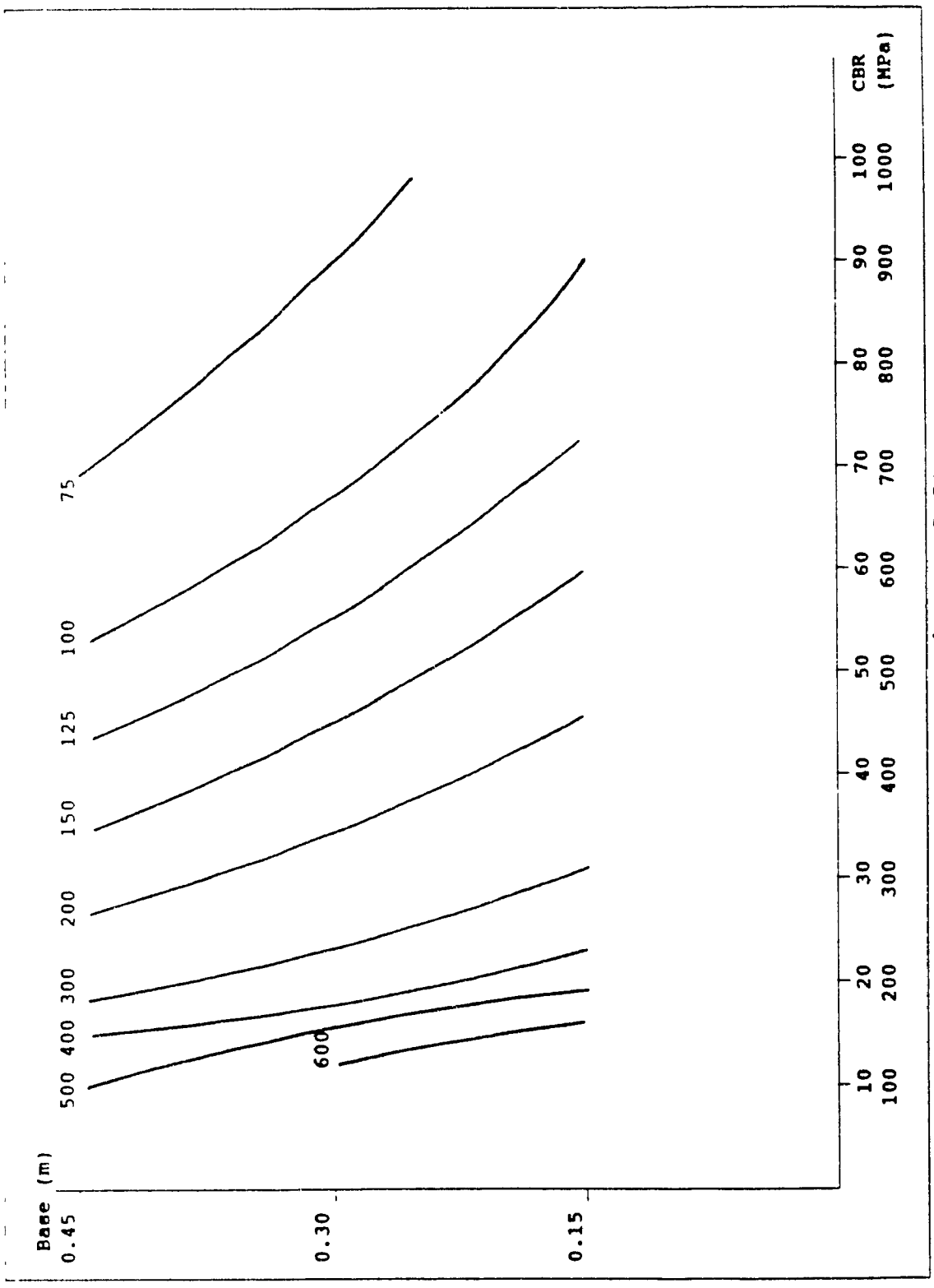


Figure A.6: Strain Contour Chart (Von Mises v=0.2)

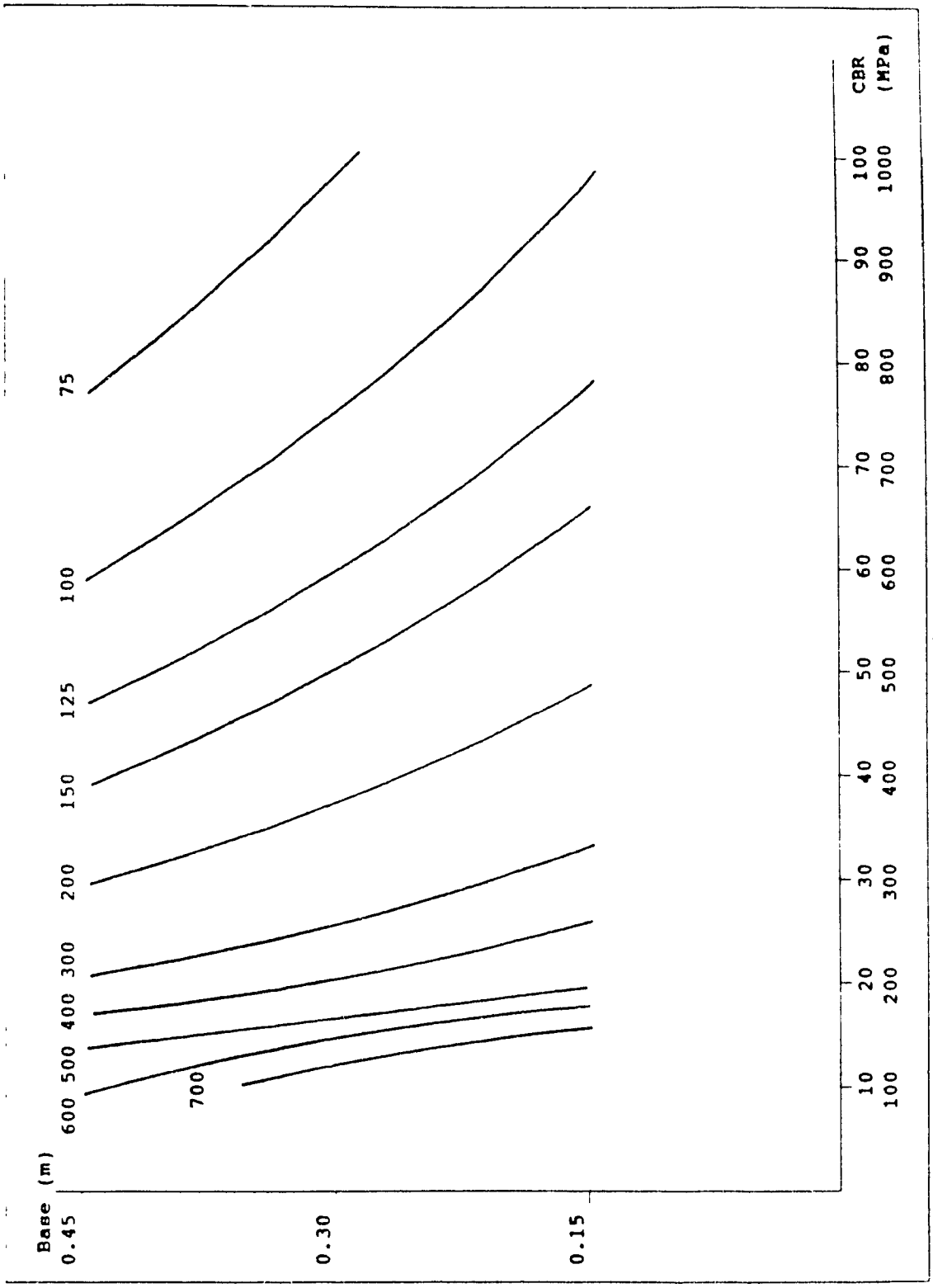


Figure A.7: Strain Contour Chart (Von Mises v=0.3)

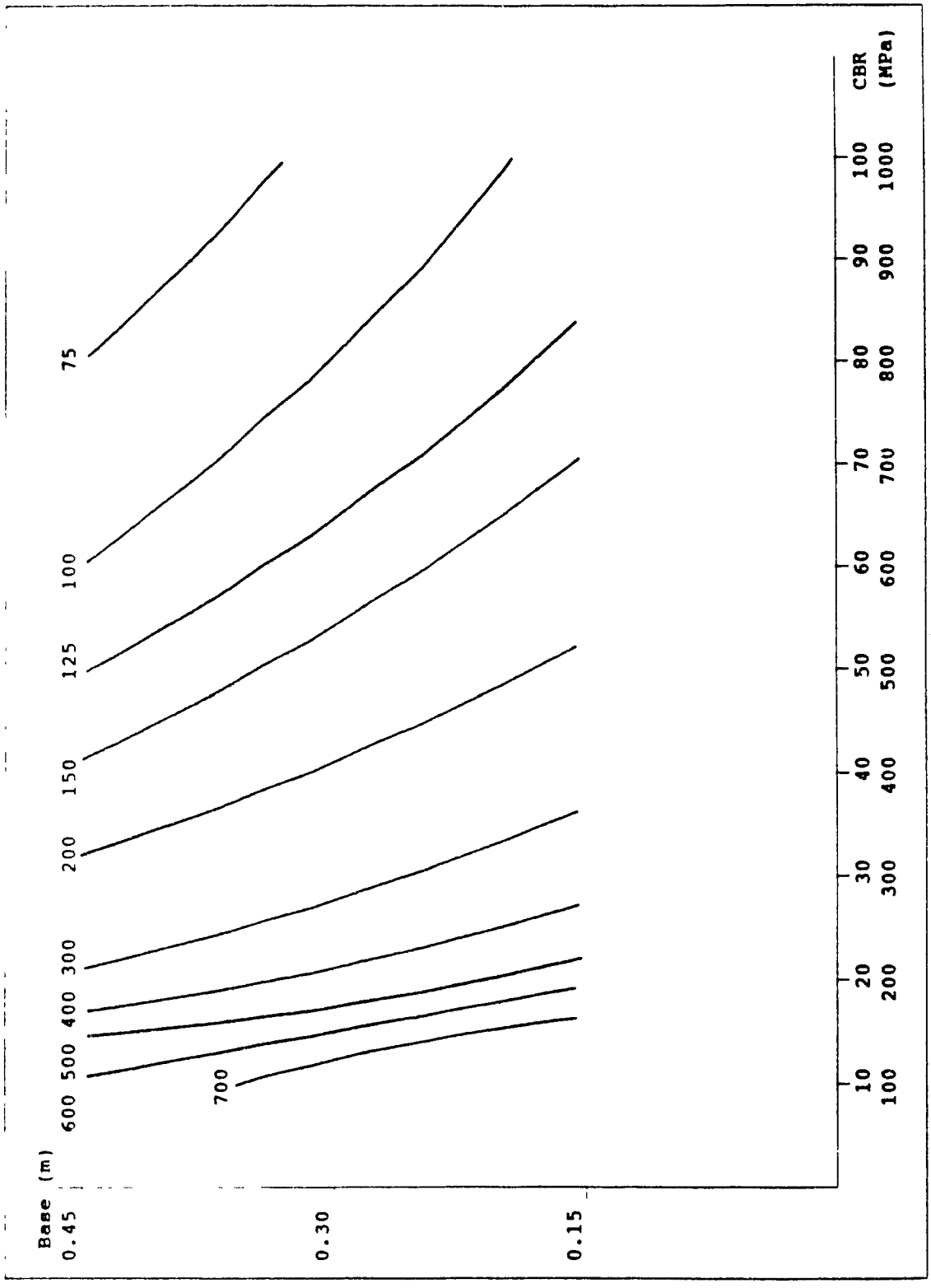


Figure A.8: Strain Contour Chart (Von Mises v=0.4)

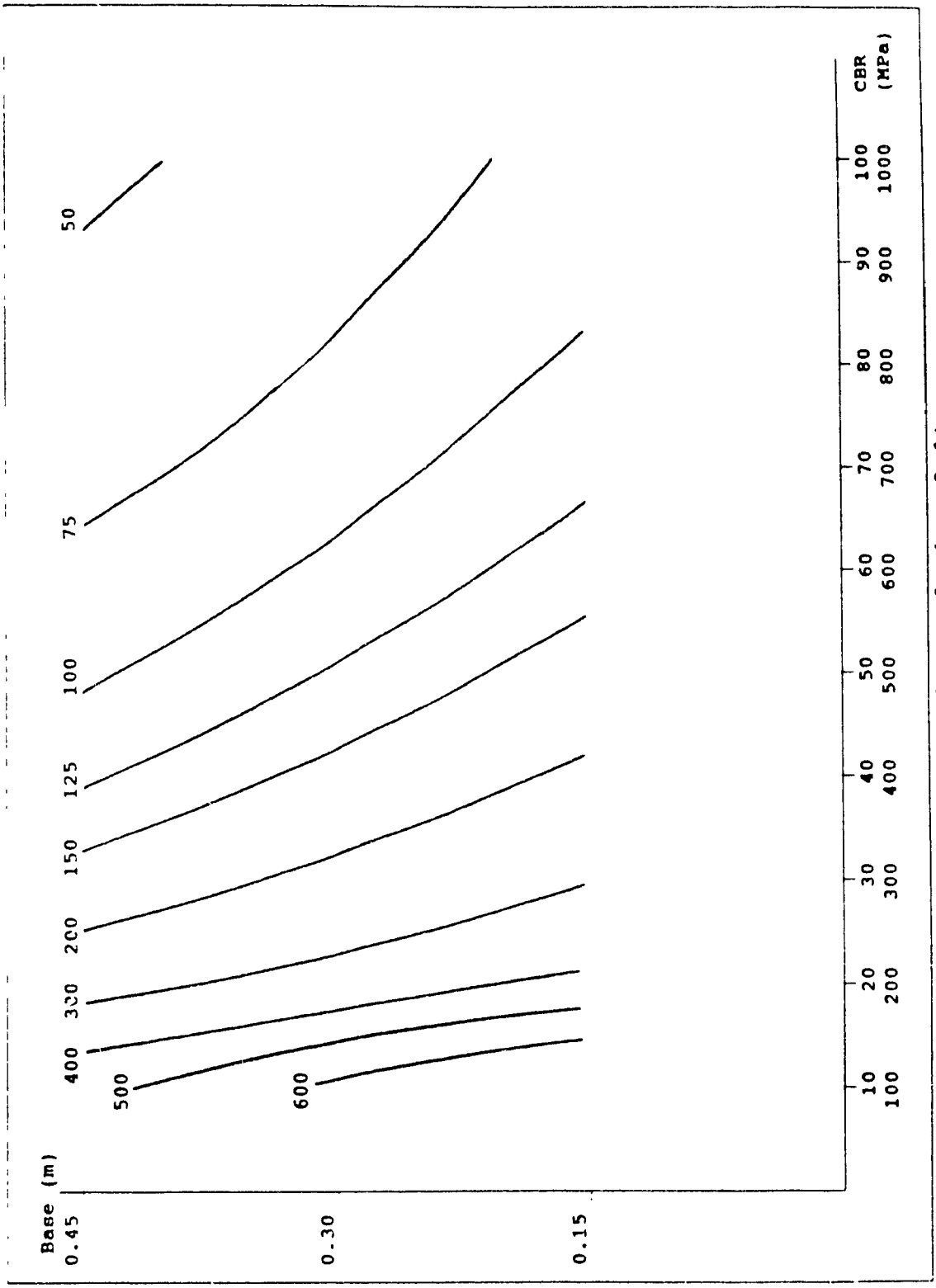


Figure A.9: Strain Contour Chart (Mohr Coulomb v=0.1)

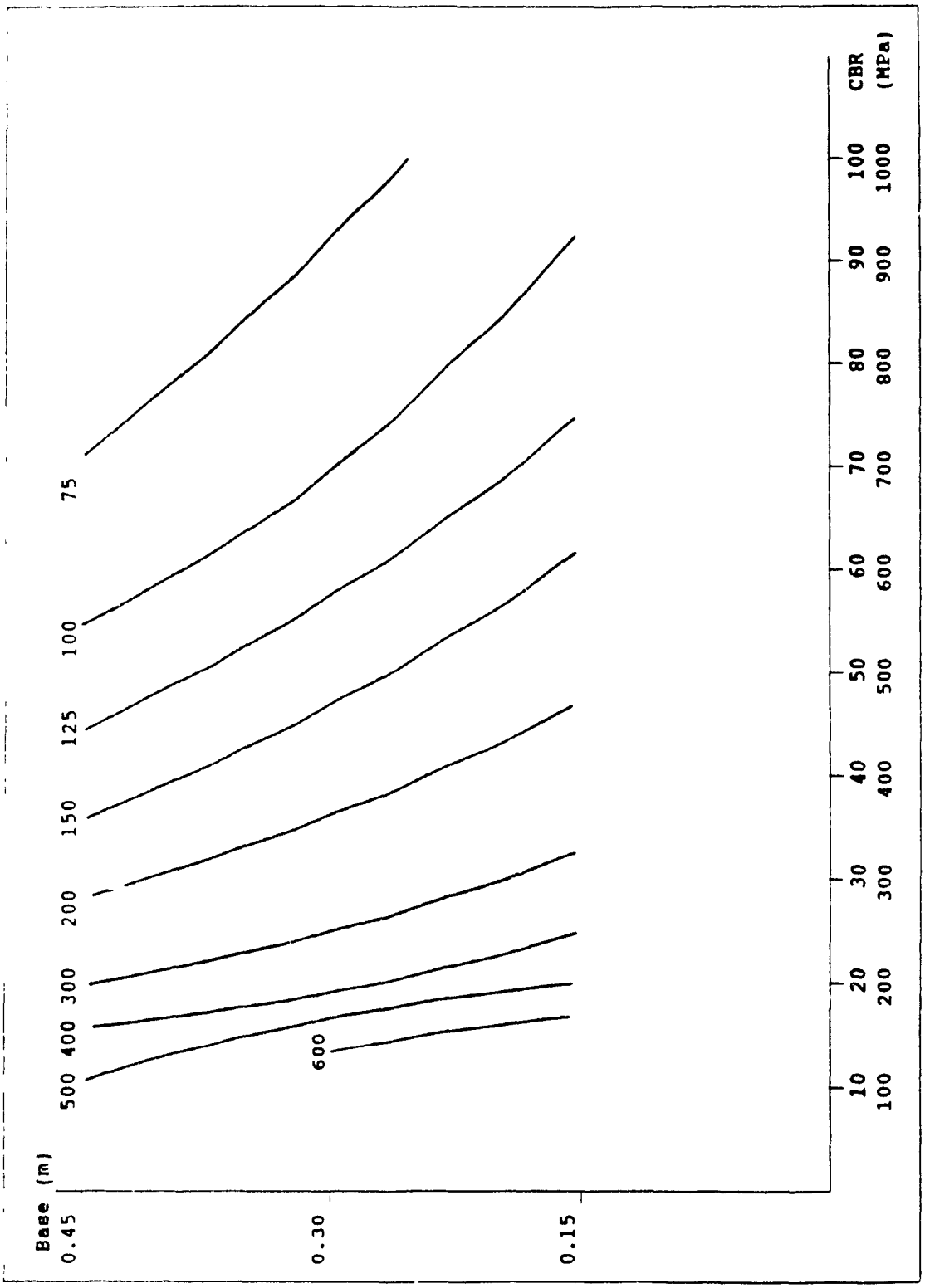


Figure A.10: Strain Contour Chart (Mohr Coulomb v=0.2)

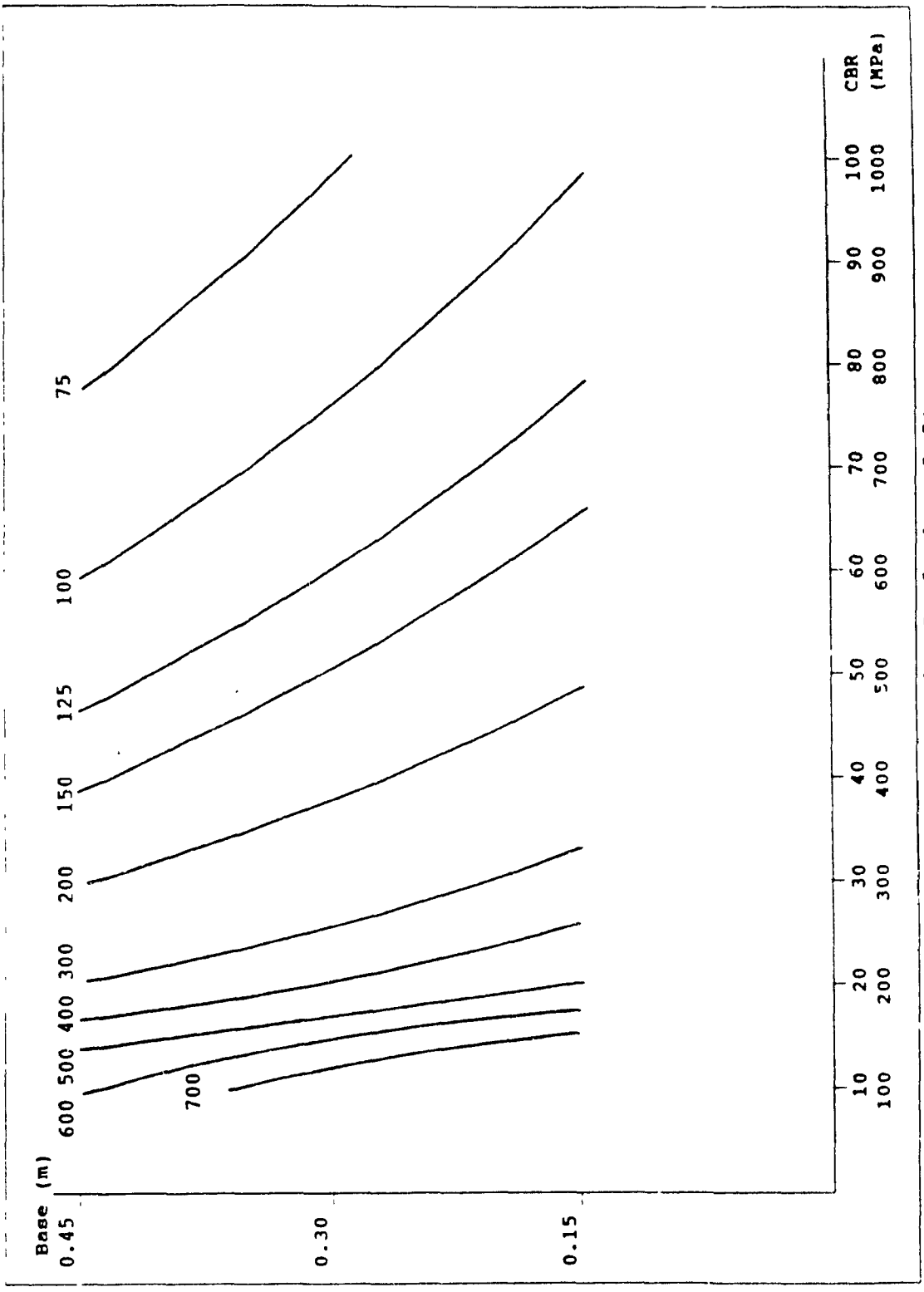


Figure A.11: Strain Contour Chart (Mohr Coulomb v=0.3)

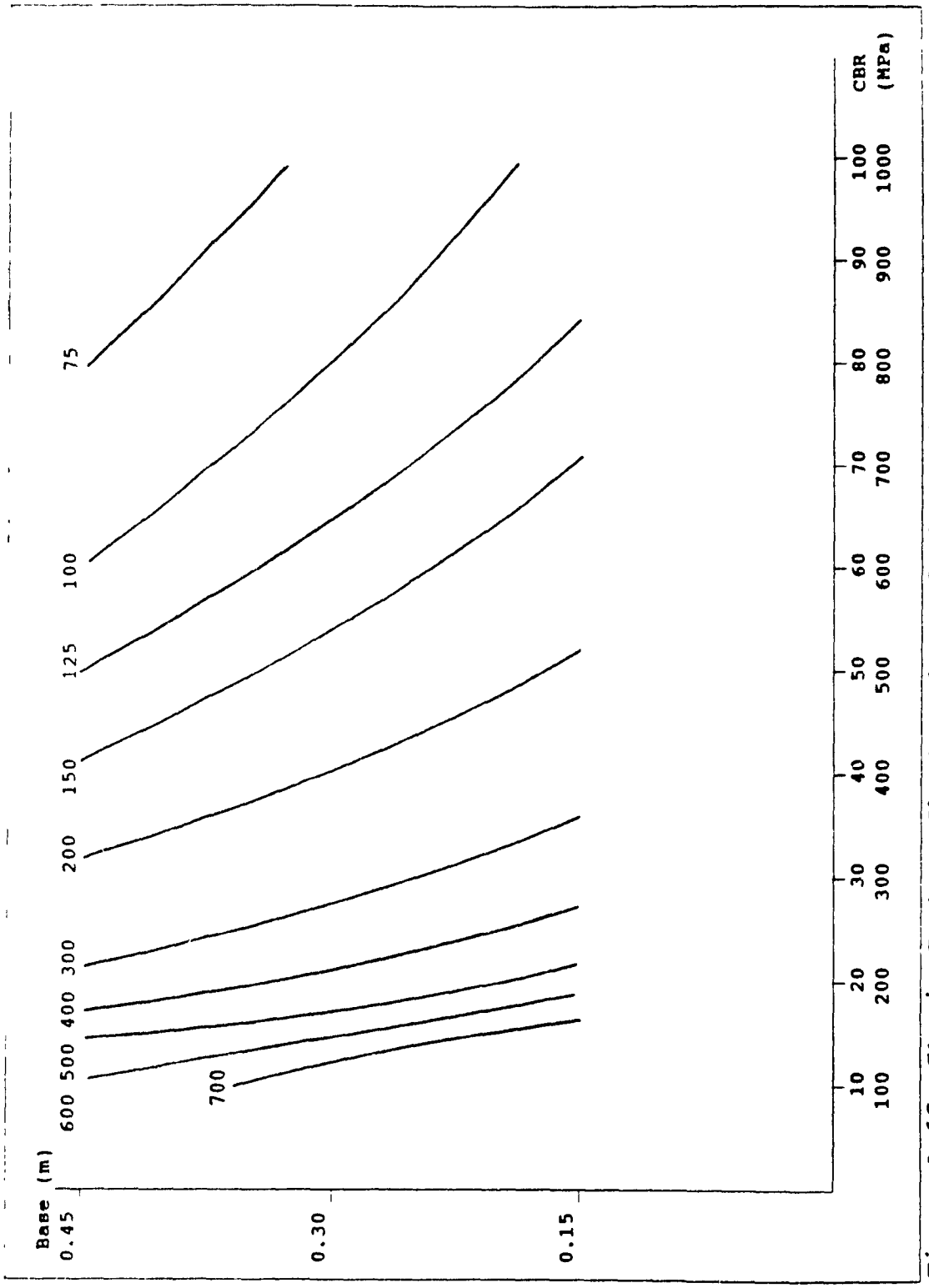


Figure A.12: Strain Contour Chart (Mohr Coulomb v=0.4)

Appendix-B

Sample Output

This Appendix shows a complete list of all data files used in this research project. An example of the output is shown for all three programs (linear elastic, Von Mises, and Mohr Coulomb) as well as the proceedure to find the vertical strain in each case. The information includes:

Table B.1: Base Course and large Element Files

Base Course

	Load				
Gravel Base Course	71 kN	80 kN	89 kN	98 kN	107 kN
For all three depths	BASE.001	BASE.002	BASE.003	BASE.004	BASE.005

Large Element Mesh

Soil Type	Base Depth	Load				
		71 kN	80 kN	89 kN	98 kN	107 kN
Sand Linear Elastic	0.15	STR.C01	STR.C04	STR.C07	STR.C10	STR.C13
	0.30	STR.C02	STR.C05	STR.C08	STR.C11	STR.C14
	0.45	STR.C03	STR.C06	STR.C09	STR.C12	STR.C15
Sand Mohr Coulomb	0.15	STR.C16	STR.C19	STR.C22	STR.C25	STR.C28
	0.30	STR.C17	STR.C20	STR.C23	STR.C26	STR.C29
	0.45	STR.C18	STR.C21	STR.C24	STR.C27	STR.C30
Clay Von Mises	0.15	STR.C31	STR.C34	STR.C37	STR.C40	STR.C43
	0.30	STR.C32	STR.C35	STR.C38	STR.C41	STR.C44
	0.45	STR.C33	STR.C36	STR.C39	STR.C42	STR.C45
Clay Mohr Coulomb	0.15	STR.C46	STR.C49	STR.C52	STR.C55	STR.C58
	0.30	STR.C47	STR.C50	STR.C53	STR.C56	STR.C59
	0.45	STR.C48	STR.C51	STR.C54	STR.C57	STR.C60
Sand Linear Elastic	0.15	STR.C61	STR.C64	STR.C67	STR.C70	STR.C73
	0.30	STR.C62	STR.C65	STR.C68	STR.C71	STR.C74
	0.45	STR.C63	STR.C66	STR.C69	STR.C72	STR.C75
Sand Mohr Coulomb	0.15	STR.C76	STR.C79	STR.C82	STR.C85	STR.C88
	0.30	STR.C77	STR.C80	STR.C83	STR.C86	STR.C89
	0.45	STR.C78	STR.C81	STR.C84	STR.C87	STR.C90

Table B.2: Small Element Mesh Files

Small Element Mesh

Soil Type	Base Depth	Load 71 kN	80 kN	89 kN	98 kN	107 kN
Sand Linear Elastic	0.15	STR.F01	STR.F04	STR.F07	STR.F10	STR.F13
	0.30	STR.F02	STR.F05	STR.F08	STR.F11	STR.F14
	0.45	STR.F03	STR.F06	STR.F09	STR.F12	STR.F15
Sand Mohr Coulomb	0.15	STR.F16	STR.F19	STR.F22	STR.F25	STR.028
	0.30	STR.F17	STR.F20	STR.F23	STR.F26	STR.F29
	0.45	STR.F18	STR.F21	STR.F24	STR.F27	STR.F30
Clay Von Mises	0.15	STR.F31	STR.F34	STR.F37	STR.F40	STR.F43
	0.30	STR.F32	STR.F35	STR.F38	STR.F41	STR.F44
	0.45	STR.F33	STR.F36	STR.F39	STR.F42	STR.F45
Clay Mohr Coulomb	0.15	STR.F46	STR.F49	STR.F52	STR.F55	STR.F58
	0.30	STR.F47	STR.F50	STR.F53	STR.F56	STR.F59
	0.45	STR.F48	STR.F51	STR.F54	STR.F57	STR.F60
Sand Linear Elastic	0.15	STR.F61	STR.F64	STR.F67	STR.F70	STR.F73
	0.30	STR.F62	STR.F65	STR.F68	STR.F71	STR.F74
	0.45	STR.F63	STR.F66	STR.F69	STR.F72	STR.F75
Sand Mohr Coulomb	0.15	STR.F76	STR.F79	STR.F82	STR.F85	STR.F88
	0.30	STR.F77	STR.F80	STR.F83	STR.F86	STR.F89
	0.45	STR.F78	STR.F81	STR.F84	STR.F87	STR.F90

Table B.3: Linear Elastic Material (Design Method Files)

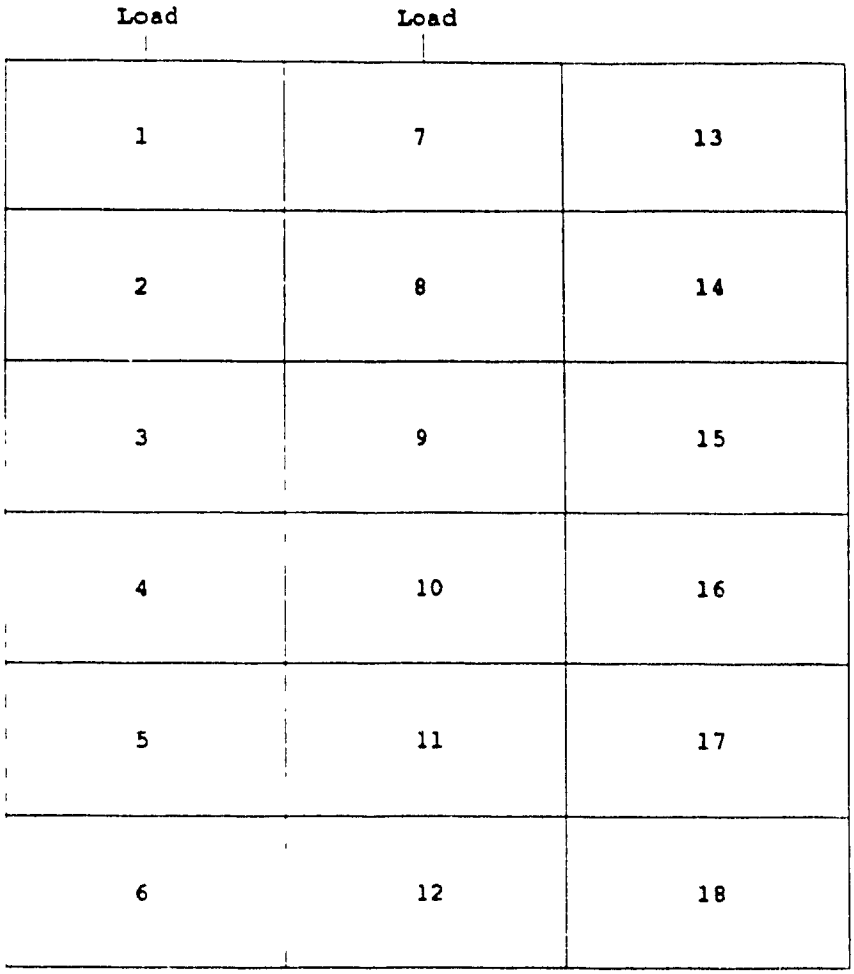
		ν			
	E	0.1	0.2	0.3	0.4
Base=0.15 Subdirectory \LE15	100	STR.DO1	STR.D11	STR.D21	STR.D31
	200	STR.DO2	STR.D12	STR.D22	STR.D32
	300	STR.DO3	STR.D13	STR.D23	STR.D33
	400	STR.DO4	STR.D14	STR.D24	STR.D34
	500	STR.DO5	STR.D15	STR.D25	STR.D35
	600	STR.DO6	STR.D16	STR.D26	STR.D36
	700	STR.DO7	STR.D17	STR.D27	STR.D37
	800	STR.DO8	STR.D18	STR.D28	STR.D38
	90(	STR.DO9	STR.D19	STR.D29	STR.D39
	1000	STR.D10	STR.D20	STR.D30	STR.D40
Base=0.30 Subdirectory \LE30	100	STR.DO1	STR.D11	STR.D21	STR.D31
	200	STR.DO2	STR.D12	STR.D22	STR.D32
	300	STR.DO3	STR.D13	STR.D23	STR.D33
	400	STR.DO4	STR.D14	STR.D24	STR.D34
	500	STR.DO5	STR.D15	STR.D25	STR.D35
	600	STR.DO6	STR.D16	STR.D26	STR.D36
	700	STR.DO7	STR.D17	STR.D27	STR.D37
	800	STR.DO8	STR.D18	STR.D28	STR.D38
	900	STR.DO9	STR.D19	STR.D29	STR.D39
	1000	STR.D10	STR.D20	STR.D30	STR.D40
Base=0.45 Subdirectory \LE45	100	STR.DO1	STR.D11	STR.D21	STR.D31
	200	STR.DO2	STR.D12	STR.D22	STR.D32
	300	STR.DO3	STR.D13	STR.D23	STR.D33
	400	STR.DO4	STR.D14	STR.D24	STR.D34
	500	STR.DO5	STR.D15	STR.D25	STR.D35
	600	STR.DO6	STR.D16	STR.D26	STR.D36
	700	STR.DO7	STR.D17	STR.D27	STR.D37
	800	STR.DO8	STR.D18	STR.D28	STR.D38
	900	STR.DO9	STR.D19	STR.D29	STR.D39
	1000	STR.D10	STR.D20	STR.D30	STR.D40

Table B.4: Von Mises Material (Design Method Files)

		ν			
	E	0.1	0.2	0.3	0.4
Base=0.15 Subdirectory \VM15	100	STR.DO1	STR.D11	STR.D21	STR.D31
	200	STR.DO2	STR.D12	STR.D22	STR.D32
	300	STR.DO3	STR.D13	STR.D23	STR.D33
	400	STR.DO4	STR.D14	STR.D24	STR.D34
	500	STR.DO5	STR.D15	STR.D25	STR.D35
	600	STR.DO6	STR.D16	STR.D26	STR.D36
	700	STR.DO7	STR.D17	STR.D27	STR.D37
	800	STR.DO8	STR.D18	STR.D28	STR.D38
	900	STR.DO9	STR.D19	STR.D29	STR.D39
	1000	STR.D10	STR.D20	STR.D30	STR.D40
Base=0.30 Subdirectory \VM30	100	STR.DO1	STR.D11	STR.D21	STR.D31
	200	STR.DO2	STR.D12	STR.D22	STR.D32
	300	STR.DO3	STR.D13	STR.D23	STR.D33
	400	STR.DO4	STR.D14	STR.D24	STR.D34
	500	STR.DO5	STR.D15	STR.D25	STR.D35
	600	STR.DO6	STR.D16	STR.D26	STR.D36
	700	STR.DO7	STR.D17	STR.D27	STR.D37
	800	STR.DO8	STR.D18	STR.D28	STR.D38
	900	STR.DO9	STR.D19	STR.D29	STR.D39
	1000	STR.D10	STR.D20	STR.D30	STR.D40
Base=0.45 Subdirectory \VM45	100	STR.DO1	STR.D11	STR.D21	STR.D31
	200	STR.DO2	STR.D12	STR.D22	STR.D32
	300	STR.DO3	STR.D13	STR.D23	STR.D33
	400	STR.DO4	STR.D14	STR.D24	STR.D34
	500	STR.DO5	STR.D15	STR.D25	STR.D35
	600	STR.DO6	STR.D16	STR.D26	STR.D36
	700	STR.DO7	STR.D17	STR.D27	STR.D37
	800	STR.DO8	STR.D18	STR.D28	STR.D38
	900	STR.DO9	STR.D19	STR.D29	STR.D39
	1000	STR.D10	STR.D20	STR.D30	STR.D40

Table B.5: Mohr Coulomb Material (Design Method Files)

		ν			
	E	0.1	0.2	0.3	0.4
Base=0.15	100	STR.D01	STR.D11	STR.D21	STR.D31
Subdirectory	200	STR.D02	STR.D12	STR.D22	STR.D32
\MC15	300	STR.D03	STR.D13	STR.D23	STR.D33
	400	STR.D04	STR.D14	STR.D24	STR.D34
	500	STR.D05	STR.D15	STR.D25	STR.D35
	600	STR.D06	STR.D16	STR.D26	STR.D36
	700	STR.D07	STR.D17	STR.D27	STR.D37
	800	STR.D08	STR.D18	STR.D28	STR.D38
	900	STR.D09	STR.D19	STR.D29	STR.D39
	1000	STR.D10	STR.D20	STR.D30	STR.D40
Base=0.30	100	STR.D01	STR.D11	STR.D21	STR.D31
Subdirectory	200	STR.D02	STR.D12	STR.D22	STR.D32
\MC30	300	STR.D03	STR.D13	STR.D23	STR.D33
	400	STR.D04	STR.D14	STR.D24	STR.D34
	500	STR.D05	STR.D15	STR.D25	STR.D35
	600	STR.D06	STR.D16	STR.D26	STR.D36
	700	STR.D07	STR.D17	STR.D27	STR.D37
	800	STR.D08	STR.D18	STR.D28	STR.D38
	900	STR.D09	STR.D19	STR.D29	STR.D39
	1000	STR.D10	STR.D20	STR.D30	STR.D40
Base=0.45	100	STR.D01	STR.D11	STR.D21	STR.D31
Subdirectory	200	STR.D02	STR.D12	STR.D22	STR.D32
\MC45	300	STR.D03	STR.D13	STR.D23	STR.D33
	400	STR.D04	STR.D14	STR.D24	STR.D34
	500	STR.D05	STR.D15	STR.D25	STR.D35
	600	STR.D06	STR.D16	STR.D26	STR.D36
	700	STR.D07	STR.D17	STR.D27	STR.D37
	800	STR.D08	STR.D18	STR.D28	STR.D38
	900	STR.D09	STR.D19	STR.D29	STR.D39
	1000	STR.D10	STR.D20	STR.D30	STR.D40



Linear Elastic Method:

Stresses and strains are measured at Gauss points at each node of the element as shown below. The critical strain for the mesh is located at element 1, Gauss point 1.

Elasto-Plastic Method:

Stresses and strains are measured at the element centres rather than at the Gauss points at the nodes.

Figure B.1: Mesh Used for Sample Output

Table B.6: Linear Elastic Output

Node	G.P.		horizontal	vertical	shear
1	1	strain	0.00001155	0.00007832	0.00001504
		stress	11540.27929	21299.32031	1099.037841
	2	strain	0.00001155	0.00007347	-0.00000833
		stress	11008.06835	20057.49414	-608.421447
	3	strain	-0.00000129	0.00007832	0.00001237
		stress	8256.706054	19892.07421	904.0900268
	4	strain	-0.00000129	0.00007347	-0.00001099
		stress	7724.494628	18650.24804	-803.367858
2	1	strain	-0.00000765	0.00007737	-0.00000883
		stress	6523.767089	18949.11328	-645.020996
	2	strain	-0.00000765	0.00007963	-0.00001706
		stress	6771.920898	19528.13671	-1246.80017
	3	strain	-0.00001217	0.00007737	-0.00000758
		stress	5366.503418	18453.14257	-554.122131
	4	strain	-0.00001217	0.00007963	-0.00001582
		stress	5614.656738	19032.16601	-1155.89892
3	1	strain	-0.00001382	0.00007194	-0.00001733
		stress	4350.300781	16885.10742	-1266.26477
	2	strain	-0.00001382	0.0000775	-0.00001728
		stress	4960.102539	18307.97656	-1262.62268
	3	strain	-0.0000138	0.00007194	-0.00001427
		stress	4357.306640	16888.11132	-1042.89416
	4	strain	-0.0000138	0.0000775	-0.00001422
		stress	4967.106933	18310.97851	-1039.25134
4	1	strain	-0.00001319	0.00006672	-0.00001597
		stress	3939.780273	15619.56542	-1166.92541
	2	strain	-0.00001319	0.00007241	-0.00001302
		stress	4563.067382	17073.90039	-951.365722
	3	strain	-0.00001157	0.00006672	-0.00001284
		stress	4354.317871	15797.22070	-938.615844
	4	strain	-0.00001157	0.00007241	-0.00000989
		stress	4977.606445	17251.56054	-723.055847
5	1	strain	-0.00001009	0.00006252	-0.00001272
		stress	4271.804199	14884.03515	-929.530273
	2	strain	-0.00001009	0.00006726	-0.00000832
		stress	4791.047851	16095.60253	-608.361389

Table B.6: Linear Elastic Output (Con't)

Node	G.P.		horizontal	vertical	shear
5	3	strain	-0.00000768	0.00006252	-0.00001012
		stress	4889.4375	15148.73535	-739.332275
	4	strain	-0.00000768	0.00006726	-0.00000572
		stress	5408.680664	16360.30175	-418.163146
6	1	strain	-0.00000536	0.00005866	-0.00001224
		stress	5059.933593	14415.51757	-894.182067
	2	strain	-0.00000536	0.00006197	-0.0000051
		stress	5423.367675	15263.53027	-372.972900
	3	strain	-0.00000144	0.00005866	-0.00001041
		stress	6062.259277	14845.08593	-761.056091
	4	strain	-0.00000144	0.00006197	-0.00000328
		stress	6425.692871	15693.09765	-239.847076
7	1	strain	0.00000218	0.00004881	-0.00003502
		stress	5908.006835	12723.32617	-2559.50488
	2	strain	0.00000218	0.00007171	-0.00004515
		stress	8418.140625	18580.30273	-3299.17724
	3	strain	-0.00000338	0.00004881	-0.00002244
		stress	4485.563476	12113.70703	-1640.04321
	4	strain	-0.00000338	0.00007171	-0.00003256
		stress	6995.695800	17970.68164	-2379.71508
8	1	strain	-0.00000486	0.0000446	-0.00005697
		stress	3645.529052	10874.24902	-4163.47949
	2	strain	-0.00000486	0.00006797	-0.0000542
		stress	6207.263183	16851.62890	-3960.51318
	3	strain	-0.00000333	0.0000446	-0.00004413
		stress	4035.852539	11041.53027	-3225.11425
	4	strain	-0.00000333	0.00006797	-0.00004136
		stress	6597.587890	17018.91210	-3022.14672
9	1	strain	-0.00000244	0.00004356	-0.00004839
		stress	4150.738769	10873.19433	-3536.10449
	2	strain	-0.00000244	0.00006284	-0.00004672
		stress	6263.962402	15804.04980	-3414.04858
	3	strain	-0.00000152	0.00004356	-0.0000378
		stress	4385.463378	10973.79101	-2762.03051
	4	strain	-0.00000152	0.00006284	-0.00003613
		stress	6498.687988	15904.64746	-2639.97436

Table B.6: Linear Elastic Output (Con't)

Node	G.P.		horizontal	vertical	shear
10	1	strain	-0.00000104	0.00004443	-0.00003466
		stress	4604.617187	11251.00292	-2532.68969
	2	strain	-0.00000104	0.00005922	-0.00003394
		stress	6225.205566	15032.375	-2480.35937
	3	strain	-0.00000065	0.00004443	-0.00002653
		stress	4705.252441	11294.13183	-1939.06872
	4	strain	-0.00000065	0.00005922	-0.00002582
		stress	6325.840332	15075.50390	-1886.73828
11	1	strain	-0.00000045	0.00004579	-0.00002326
		stress	4905.220214	11662.39160	-1700.07812
	2	strain	-0.00000045	0.00005676	-0.00002298
		stress	6107.922363	14468.69726	-1679.17382
	3	strain	-0.00000029	0.00004579	-0.00001724
		stress	4945.422363	11679.62109	-1259.52893
	4	strain	-0.00000029	0.00005676	-0.00001695
		stress	6148.124511	14485.92675	-1238.62402
12	1	strain	-0.00000018	0.00004737	-0.00001823
		stress	5146.002929	12095.83593	-1331.85681
	2	strain	-0.00000018	0.00005474	-0.00001798
		stress	5953.862793	13980.84179	-1314.17761
	3	strain	-0.00000005	0.00004737	-0.00001418
		stress	5180.001464	12110.40722	-1035.93774
	4	strain	-0.00000005	0.00005474	-0.00001393
		stress	598[illegible].861328	13995.41308	-1018.25836
13	1	strain	-0.00001373	0.00000249	-0.00006019
		stress	-3239.43286	-868.649475	-4398.22509
	2	strain	-0.00001373	0.00003024	-0.0000267
		stress	-197.086105	6230.161132	-1951.09460
	3	strain	0.00000467	0.00000249	-0.00004494
		stress	1466.588989	1148.217163	-3283.81201
	4	strain	0.00000467	0.00003024	-0.00001145
		stress	4508.937011	8247.027343	-836.680969
14	1	strain	0.00001251	0.00000719	-0.00004537
		stress	3987.966796	3210.250244	-3315.49658
	2	strain	0.00001251	0.0000283	-0.00003991
		stress	6301.560546	8608.634765	-2916.68725

Table B.6: Linear Elastic Output (Con't)

Node	G.P.		horizontal	vertical	shear
14	3	strain	0.00001551	0.00000719	-0.00003377
		stress	4754.908203	3538.939453	-2468.02563
	4	strain	0.00001551	0.0000283	-0.00002832
		stress	7068.500976	8937.321289	-2069.21582
15	1	strain	0.00001626	0.00001775	-0.00002939
		stress	6104.729492	6321.822265	-2147.78247
	2	strain	0.00001626	0.00003146	-0.00003111
		stress	7608.159179	9829.825195	-2273.48193
	3	strain	0.00001532	0.00001775	-0.00002185
		stress	5863.000488	6218.226074	-1597.07592
	4	strain	0.00001532	0.00003146	-0.00002357
		stress	7366.429199	9726.226562	-1722.77551
16	1	strain	0.00001423	0.00002658	-0.00001797
		stress	6554.012695	8359.051757	-1313.42907
	2	strain	0.00001423	0.00003568	-0.00002164
		stress	7551.3125	10686.08496	-1581.31958
	3	strain	0.00001222	0.00002658	-0.00001297
		stress	6038.837890	8138.261718	-948.118041
	4	strain	0.00001222	0.00003568	-0.00001664
		stress	7036.137695	10465.29492	-1216.00866
17	1	strain	0.00001054	0.00003325	-0.00001026
		stress	6339.277343	9658.274414	-749.638854
	2	strain	0.00001054	0.00003948	-0.00001494
		stress	7022.730957	11252.99902	-1091.71325
	3	strain	0.00000796	0.00003325	-0.00000683
		stress	5681.441406	9376.342773	-499.289733
	4	strain	0.00000796	0.00003948	-0.00001151
		stress	6364.896484	10971.07128	-841.363952
18	1	strain	0.00000554	0.00003913	-0.00000575
		stress	5705.203613	10614.63769	-419.994445
	2	strain	0.00000554	0.00004318	-0.00001312
		stress	6149.627441	11651.62695	-958.882751
	3	strain	0.00000149	0.00003913	-0.00000352
		stress	4668.879882	10170.49902	-257.201873
	4	strain	0.00000149	0.00004318	-0.00001089
		stress	5113.303710	11207.48730	-796.090148

Table B.7: Von Mises Output

Load Increment	Element #	Stress	Strain
1	1	10740.248047	0.0000449672
	2	12360.21875	0.0000517497
	3	12221.75	0.00005117
	4	11301.6025391	0.0000473175
	5	10261.4980469	0.0000429628
	6	9126.5302734	0.0000382109
	7	9830.9453125	0.0000411602
	8	10567.8652344	0.0000442455
	9	9559.7988281	0.0000400249
	10	8542.4931641	0.0000357657
	11	7939.7822266	0.0000332422
	12	7742.5883789	0.0000324166
	13	5319.6054688	0.0000222721
	14	5230.0893555	0.0000218973
	15	4603.7851563	0.0000192751
	16	4521.3344727	0.0000189299
	17	4978.5844727	0.0000208443
	18	5870.7553711	0.0000245797
		2	Iterations
2	1	21480.496094	0.0000899344
	2	24720.4375	0.0001034994
	3	24443.5	0.0001023399
	4	22603.205078	0.000094635
	5	20522.996094	0.0000859256
	6	18253.060547	0.0000764218
	7	19661.890625	0.0000823203
	8	21135.730469	0.000088491
	9	19119.597656	0.0000800498
	10	17084.986328	0.0000715313
	11	15879.5644531	0.0000664845
	12	15485.1767578	0.0000648333
	13	10639.2109375	0.0000445442
	14	10460.1787109	0.0000437946
	15	9207.5703125	0.0000385502
	16	9042.6689453	0.0000378598
	17	9957.1689453	0.0000416886
	18	11741.5107422	0.0000491593
		2	Iterations

Table B.7: Von Mises Output (Con't)

Load Increment	Element #	Stress	Strain
3	1	32220.746094	0.0001349017
	2	37080.65625	0.0001552491
	3	36665.25	0.0001535099
	4	33904.808594	0.0001419525
	5	30784.494141	0.0001288884
	6	27379.589844	0.0001146327
	7	29492.835938	0.0001234805
	8	31703.595703	0.0001327365
	9	28679.396484	0.0001200748
	10	25627.478516	0.000107297
	11	23819.345703	0.0000997267
	12	23227.765625	0.0000972499
	13	15958.8164063	0.0000668163
	14	15690.2675781	0.0000656919
	15	13811.3564453	0.0000578253
	16	13564.0048828	0.0000567897
	17	14935.7539063	0.0000625329
	18	17612.265625	0.0000737389
		2	Iterations
4	1	42960.992188	0.0001798689
	2	49440.875	0.0002069988
	3	48887	0.0002046799
	4	45206.410156	0.00018927
	5	41045.992188	0.0001718512
	6	36506.121094	0.0001528437
	7	39323.78125	0.0001646406
	8	42271.460938	0.000176982
	9	38239.195313	0.0001600997
	10	34169.972656	0.0001430627
	11	31759.128906	0.000132969
	12	30970.353516	0.0001296665
	13	21278.421875	0.0000890884
	14	20920.357422	0.0000875893
	15	18415.140625	0.0000771004
	16	18085.337891	0.0000757196
	17	19914.337891	0.0000833773
	18	23483.021484	0.0000983186
		2	Iterations

Table B.7: Von Mises Output (Con't)

Load Increment	Element #	Stress	Strain
5	1	53701.238281	0.0002248361
	2	61801.09375	0.0002587485
	3	61108.753906	0.0002558499
	4	56508.015625	0.0002365875
	5	51307.492188	0.000214814
	6	45632.644531	0.0001910546
	7	49154.726563	0.0002058008
	8	52839.328125	0.0002212275
	9	47799	0.0002001246
	10	42712.464844	0.0001788284
	11	39698.914063	0.0001662112
	12	38712.9375	0.0001620832
	13	26598.025391	0.0001113605
	14	26150.445313	0.0001094866
	15	23018.925781	0.0000963755
	16	22606.671875	0.0000946495
	17	24892.919922	0.0001042216
	18	29353.777344	0.0001228982
		2	Iterations
6	1	64441.484375	0.0002698033
	2	74161.3125	0.0003104983
	3	73330.507813	0.0003070198
	4	67809.617188	0.000283905
	5	61568.992188	0.0002577768
	6	54759.171875	0.0002292655
	7	58985.675781	0.000246961
	8	63407.195313	0.0002654729
	9	57358.800781	0.0002401495
	10	51254.953125	0.000214594
	11	47638.699219	0.0001994535
	12	46455.523438	0.0001944998
	13	31917.630859	0.0001336326
	14	31380.533203	0.0001313839
	15	27622.710938	0.0001156506
	16	27128.003906	0.0001135794
	17	29871.501953	0.0001250659
	18	35224.535156	0.0001474779
		2	Iterations

Table B.7: Von Mises Output (Con't)

Load Increment	Element #	Stress	Strain
7	1	**75181.734375**	**0.0003147706**
	2	86521.53125	0.000362248
	3	85552.257813	0.0003581898
	4	79111.21875	0.0003312225
	5	71830.492188	0.0003007396
	6	63885.695313	0.0002674764
	7	68816.625	0.0002881211
	8	73975.054688	0.0003097185
	9	66918.601563	0.0002801745
	10	59797.441406	0.0002503597
	11	55578.484375	0.0002326957
	12	54198.113281	0.0002269164
	13	37237.234375	0.0001559047
	14	36610.621094	0.0001532812
	15	32226.496094	0.0001349258
	16	31649.339844	0.0001325093
	17	34850.085938	0.0001459102
	18	41095.289063	0.0001720576
		2	Iterations
8	1	**85921.984375**	**0.0003597378**
	2	98881.75	0.0004139977
	3	97774.015625	0.0004093598
	4	90412.820313	0.00037854
	5	82091.992188	0.0003437024
	6	73012.226563	0.0003056873
	7	78647.570313	0.0003292813
	8	84542.921875	0.000353964
	9	76478.40625	0.0003201994
	10	68339.929688	0.0002861254
	11	63518.269531	0.0002659379
	12	61940.699219	0.0002593331
	13	42556.839844	0.0001781768
	14	41840.710938	0.0001751785
	15	36830.28125	0.0001542009
	16	36170.671875	0.0001514392
	17	39828.667969	0.0001667545
	18	46966.046875	0.0001966372
		2	Iterations

Table B.7: Von Mises Output (Con't)

Load Increment	Element #	Stress	Strain
9	1	96662.234375	0.000404705
	2	111241.96875	0.0004657474
	3	109995.765625	0.0004605297
	4	101714.4375	0.0004258576
	5	92353.484375	0.0003866652
	6	82138.75	0.0003438982
	7	88478.515625	0.0003704415
	8	95110.789063	0.0003982094
	9	86038.203125	0.0003602243
	10	76882.421875	0.000321891
	11	71458.054688	0.0002991802
	12	69683.28125	0.0002917497
	13	47876.445313	0.0002004489
	14	47070.800781	0.0001970758
	15	41434.066406	0.000173476
	16	40692.003906	0.0001703691
	17	44807.253906	0.0001875989
	18	52836.804688	0.0002212168
		2	Iterations
10	1	107402.4843	0.0004496723
	2	123602.1875	0.0005174971
	3	122217.523438	0.0005116996
	4	113016.046875	0.0004731751
	5	102614.984375	0.000429628
	6	91265.273438	0.0003821091
	7	98309.460938	0.0004116016
	8	105678.65625	0.0004424549
	9	95598.007813	0.0004002493
	10	85424.914063	0.0003576567
	11	79397.835938	0.0003324224
	12	77425.867188	0.0003241663
	13	53196.050781	0.000222721
	14	52300.890625	0.0002189731
	15	46037.851563	0.0001927511
	16	45213.335938	0.000189299
	17	49785.835938	0.0002084432
	18	58707.558594	0.0002457965
		2	Iterations

Table B.8: Mohr Coulomb Output

Load Increment	Element #	Stress	Strain
1	1	**2921.5097656**	**0.0000133262**
	2	2585.4499512	0.0000117933
	3	1860.1882324	0.0000084851
	4	1335.1773682	0.0000060903
	5	991.96313477	0.0000045247
	6	728.2713623	0.0000033219
	7	708.41156006	0.0000032314
	8	1171.7718506	0.0000053449
	9	1031.4346924	0.0000047048
	10	804.07250977	0.0000036677
	11	638.90332031	0.0000029143
	12	543.32232666	0.0000024783
	13	221.43247986	0.00000101
	14	337.71130371	0.0000015404
	15	359.5696106	0.0000016401
	16	316.34338379	0.000001443
	17	313.58615112	0.0000014304
	18	357.0390625	0.0000016286
		2	Iterations
2	1	**5904.9160156**	**0.0000280591**
	2	5029.2055664	0.0000269371
	3	3688.5432129	0.0000209251
	4	2561.0803223	0.0000147514
	5	1773.5177002	0.0000099168
	6	1218.338501	0.000006243
	7	1319.9104004	0.0000060206
	8	2613.0153809	0.0000112585
	9	2454.9211426	0.0000104797
	10	1923.9014893	0.0000080691
	11	1458.6368408	0.0000061332
	12	1141.0128174	0.0000051092
	13	291.91040039	0.000002056
	14	707.36090088	0.0000032266
	15	759.66809082	0.0000034652
	16	647.2835083	0.0000029525
	17	637.18823242	0.0000029065
	18	728.78637695	0.0000033243
		10	Iterations

Table B.8: Mohr Coulomb Output (Con't)

Load Increment	Element #	Stress	Strain
3	1	8884.6103516	0.0000416845
	2	7593.1738281	0.0000398133
	3	5625.4052734	0.0000309393
	4	3945.7333984	0.000022002
	5	2737.9133301	0.0000149957
	6	1867.4130859	0.0000095202
	7	1953.9445801	0.0000089127
	8	3879.7854004	0.0000165483
	9	3650.0864258	0.0000154629
	10	2880.4697266	0.0000119495
	11	2204.152832	0.0000091413
	12	1729.5644531	0.0000076753
	13	404.46286011	0.0000033089
	14	981.8994751	0.000004532
	15	1086.7835693	0.0000049573
	16	951.71716309	0.0000043412
	17	948.63659668	0.0000043271
	18	1086.7536621	0.0000049571
		9	Iterations
4	1	11870.08691	0.0000558492
	2	10148.5126953	0.0000529209
	3	7532.8378906	0.0000409864
	4	5302.3564453	0.0000292136
	5	3677.7575684	0.0000199491
	6	2498.092041	0.0000126986
	7	2634.9682617	0.0000120192
	8	5209.8710938	0.000022196
	9	4883.4121094	0.0000206953
	10	3831.7685547	0.0000158555
	11	2932.9299316	0.0000121107
	12	2303.3430176	0.0000101923
	13	505.33529663	0.0000039214
	14	1297.5935059	0.0000063304
	15	1475.6555176	0.0000067311
	16	1265.2683105	0.0000057714
	17	1264.6645508	0.0000057686
	18	1450.3791504	0.0000066158
		8	Iterations

Table B.8: Mohr Coulomb Output (Con't)

Load Increment	Element #	Stress	Strain
5	1	14867.96289	0.0000701313
	2	12728.7880859	0.0000662294
	3	9451.9912109	0.0000511423
	4	6666.6083984	0.0000364998
	5	4626.0219727	0.0000249736
	6	3130.3203125	0.000015893
	7	3323.4753418	0.0000151597
	8	6517.5195313	0.0000277727
	9	6163.5317383	0.0000261268
	10	4815.0786133	0.000019956
	11	3669.2182617	0.0000151357
	12	2877.3476563	0.000012716
	13	624.4520874	0.0000044246
	14	1601.1239014	0.0000079343
	15	1950.9802246	0.0000088992
	16	1605.2808838	0.0000073223
	17	1581.329834	0.0000072131
	18	1814.5085449	0.0000082767
		10	Iterations
6	1	17870.957031	0.0000843216
	2	15299.8632813	0.0000794796
	3	11362.4589844	0.0000612722
	4	8009.9516602	0.0000436833
	5	5552.5170898	0.0000298785
	6	3754.6882324	0.0000190396
	7	4006.9240723	0.0000182376
	8	7841.0322266	0.0000333834
	9	7391.862793	0.0000313216
	10	5762.1567383	0.0000238376
	11	4393.737793	0.0000180917
	12	3447.9284668	0.0000152251
	13	752.75518799	0.0000051353
	14	1937.8519287	0.0000095482
	15	2327.1711426	0.0000106152
	16	1916.6065674	0.0000087424
	17	1898.5609131	0.0000086601
	18	2179.2678223	0.0000099405
		5	Iterations

Table B.8: Mohr Coulomb Output (Con't)

Load Increment	Element #	Stress	Strain
7	1	20868.257813	0.0000984175
	2	17876.244141	0.0000927639
	3	13285.0683594	0.0000714886
	4	9358.0722656	0.0000509113
	5	6480.25	0.0000348028
	6	4378.9648438	0.0000221901
	7	4735.4916992	0.0000212739
	8	9157.2695313	0.0000389444
	9	8620.5146484	0.0000365042
	10	6713.3325195	0.0000277307
	11	5121.5395508	0.0000210607
	12	4020.5	0.0000177446
	13	884.26348877	0.0000059233
	14	2269.3103027	0.0000111007
	15	2693.8869629	0.0000122879
	16	2226.5205078	0.0000101561
	17	2215.3117676	0.0000101049
	18	2543.4399414	0.0000116017
		10	Iterations
8	1	23893.205078	0.0001128779
	2	20461.648438	0.0001061739
	3	15196.3222656	0.000081662
	4	10705.4658203	0.0000581391
	5	7415.2348633	0.0000397679
	6	5007.987793	0.0000253664
	7	5475.703125	0.0000241508
	8	10472.0498047	0.0000444768
	9	9876.9199219	0.0000417925
	10	7684.6162109	0.0000317383
	11	5856.8349609	0.0000240783
	12	4596.0913086	0.0000202826
	13	996.86804199	0.0000067792
	14	2572.6762695	0.0000125386
	15	3106.2858887	0.000014169
	16	2552.8061523	0.0000116444
	17	2531.7902832	0.0000115485
	18	2906.4648438	0.0000132576
		10	Iterations

Table B.8: Mohr Coulomb Output (Con't)

Load Increment	Element #	Stress	Strain
9	1	26917.363281	0.0001272406
	2	23039.865234	0.00011951
	3	17103.59375	0.000091803
	4	12044.0039063	0.0000653118
	5	8340.3056641	0.0000446733
	6	5633.6650391	0.0000285212
	7	6211.3408203	0.0000270277
	8	11793.5439453	0.0000500356
	9	11105.3466797	0.0000469649
	10	8636.1132813	0.0000356349
	11	6584.9311523	0.0000270569
	12	5169.2709961	0.0000228095
	13	1112.0804443	0.0000077105
	14	2892.9282227	0.0000140081
	15	3472.3811035	0.0000158389
	16	2865.3134766	0.0000130699
	17	2848.6550293	0.0000129939
	18	3269.9992676	0.0000149158
		3	Iterations
10	1	29940.851563	0.0001416125
	2	25627.523438	0.0001328938
	3	19019.472656	0.0001019875
	4	13396.5351563	0.0000725652
	5	9280.4892578	0.0000496689
	6	6265.5859375	0.0000317141
	7	6937.0048828	0.0000299149
	8	13098.5195313	0.0000555286
	9	12366.9814453	0.0000522722
	10	9613	0.0000396748
	11	7322.9448242	0.0000300899
	12	5746.2724609	0.0000253543
	13	1225.0366211	0.0000085752
	14	3190.1726074	0.0000154324
	15	3895.7583008	0.0000177701
	16	3195.0053711	0.0000145737
	17	3164.5375977	0.0000144347
	18	3632.4367676	0.000016569
		10	Iterations

Appendix-C

Detailed Flowcharts

The flowcharts for the programs used in the thesis are listed in this Appendix. The flowcharts are:

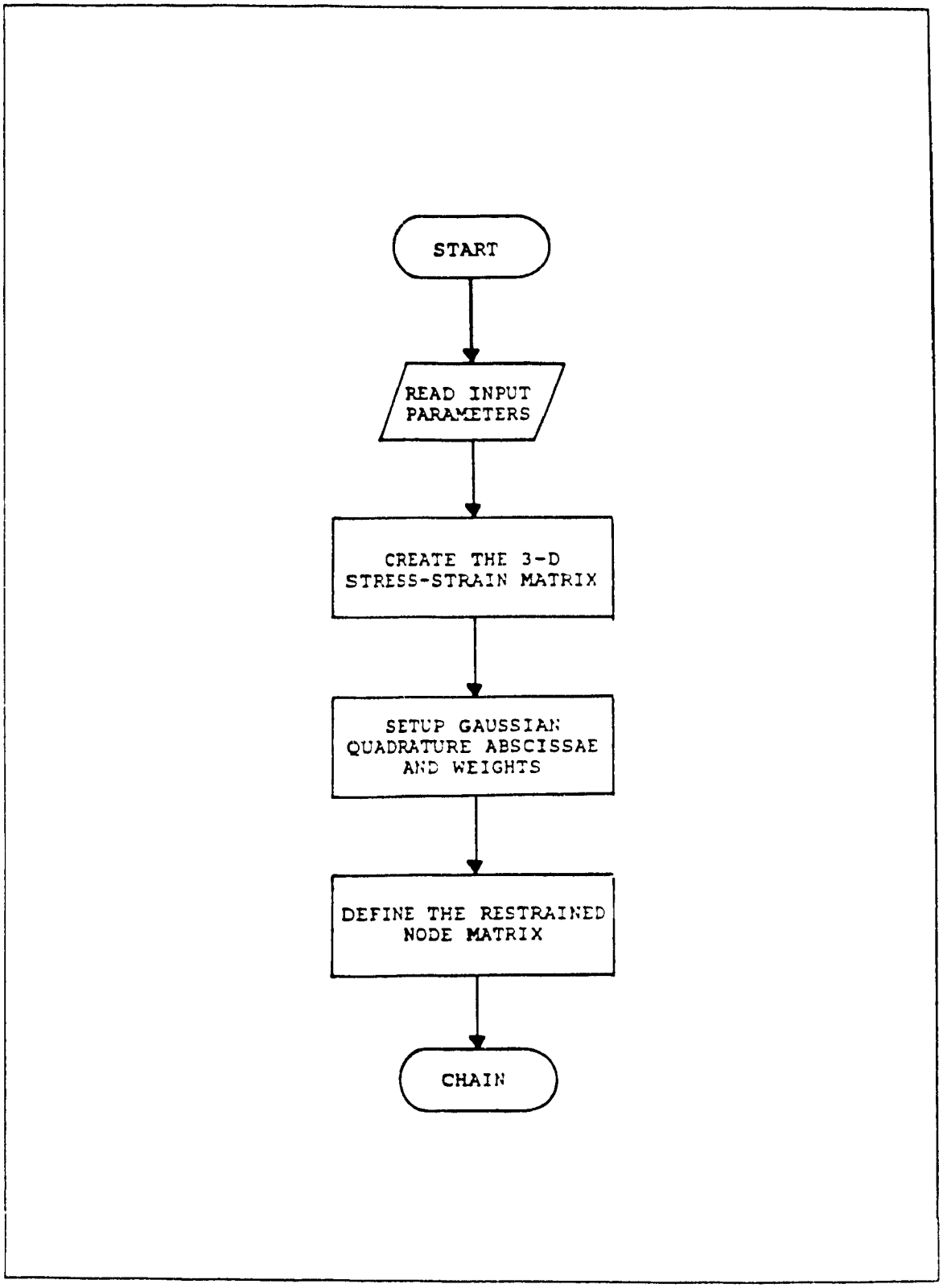


Figure C.1: Flowchart of ELAST1.EXE

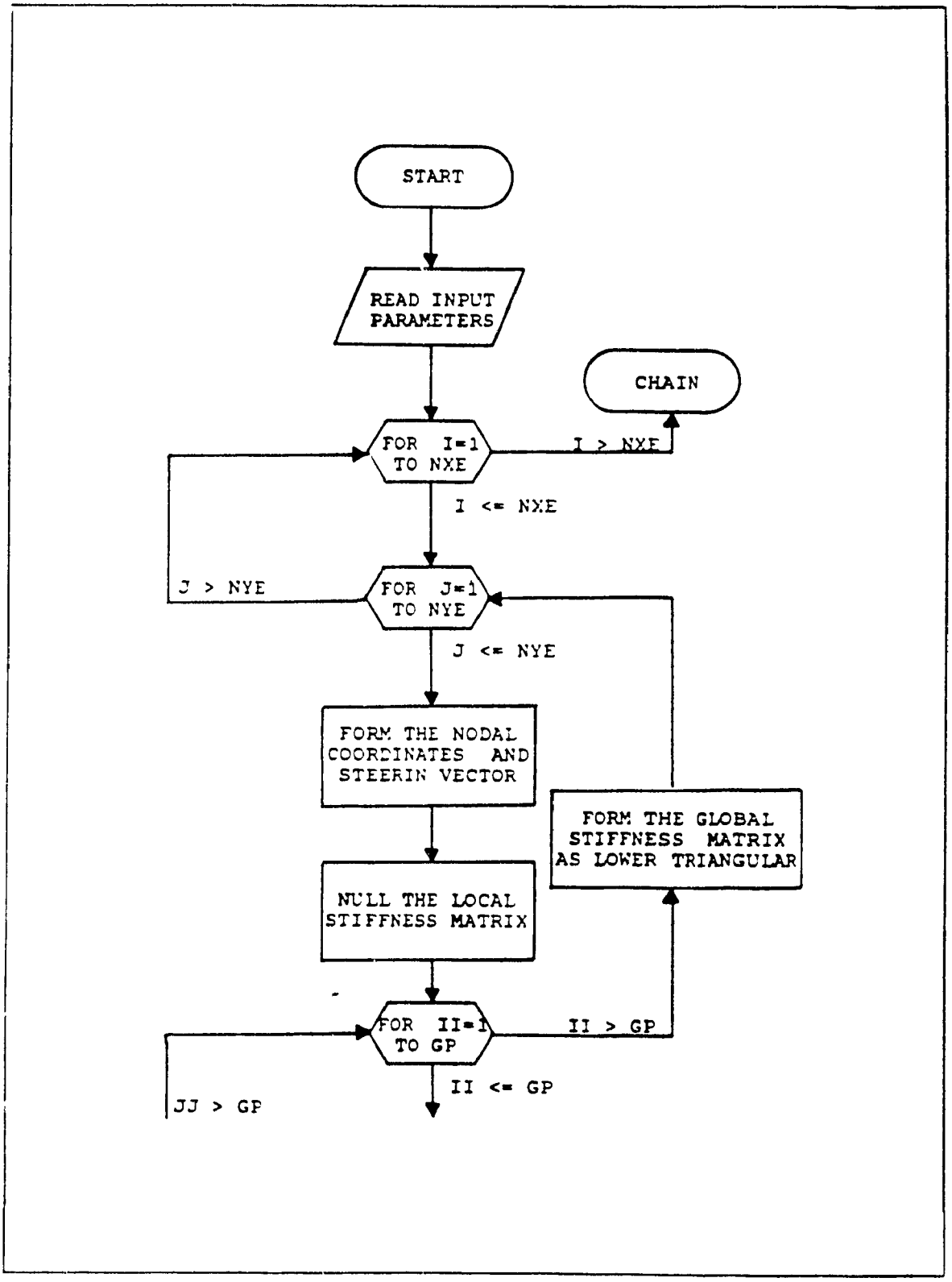


Figure C.2: Flowchart of ELAST2.EXE

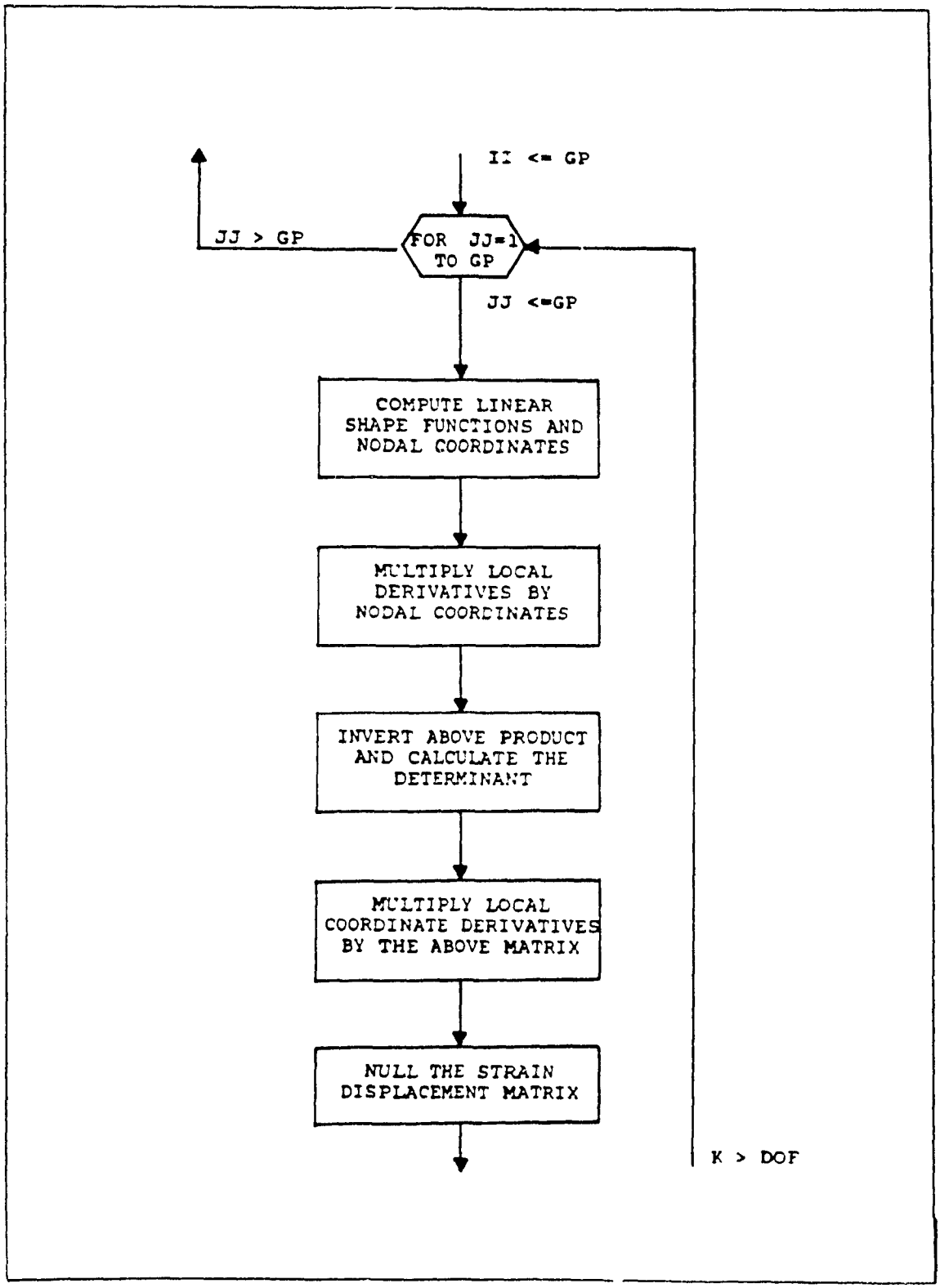


Figure C.2: Flowchart of ELAST2.EXE (Con't)

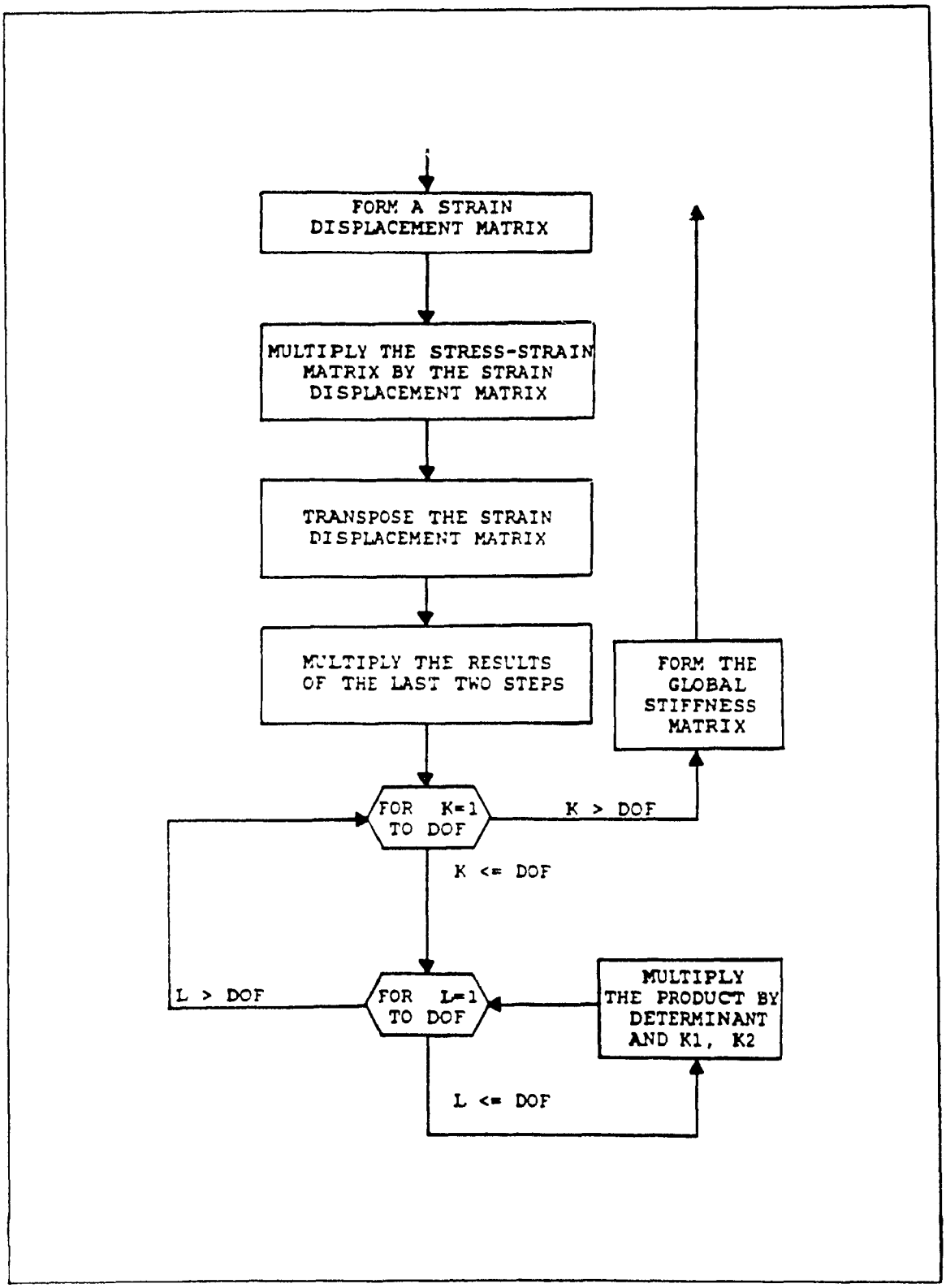


Figure C.2: Flowchart of ELAST2.EXE (Con't)

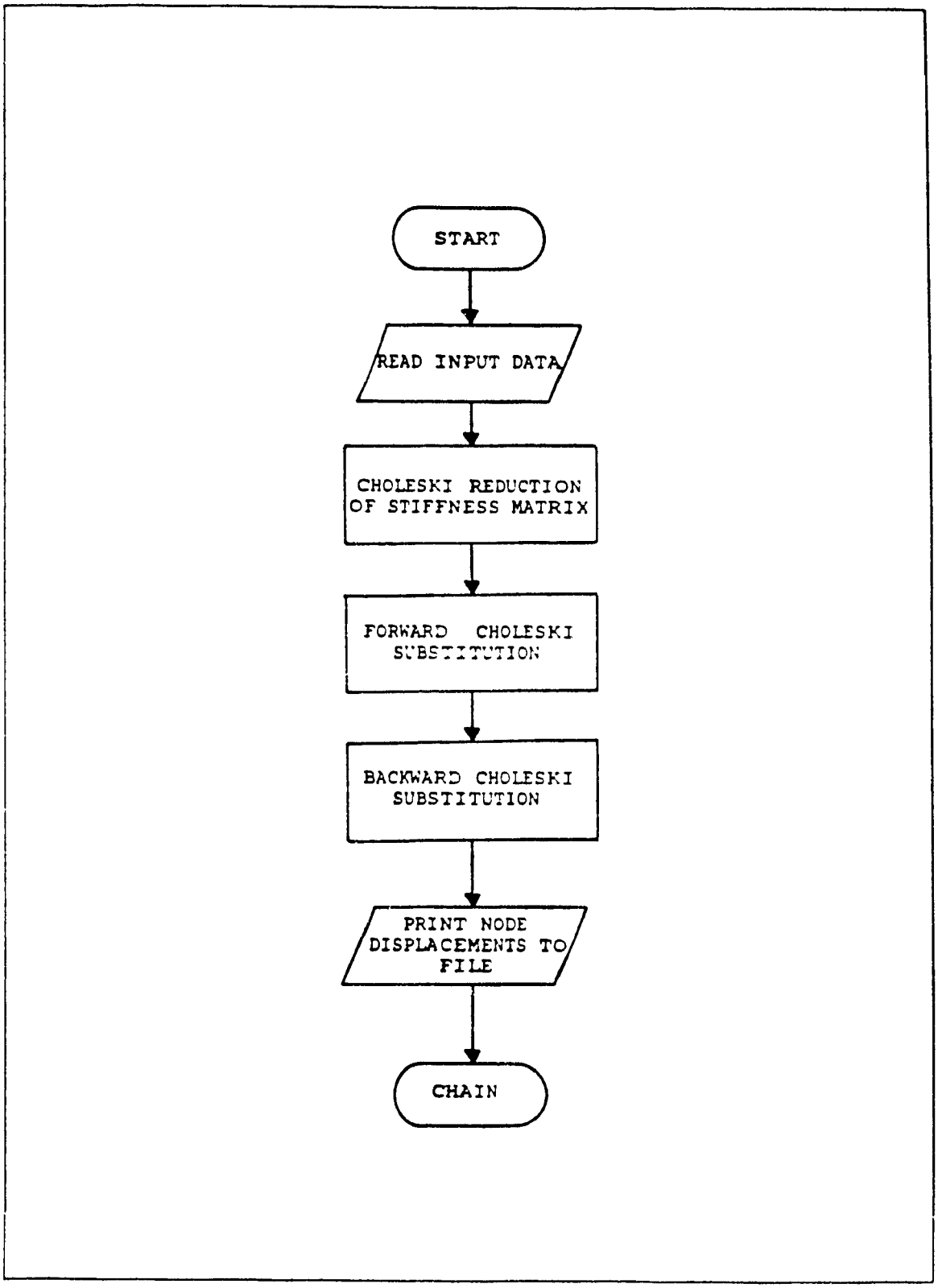


Figure C.3: Flowchart of ELAST3.EXE

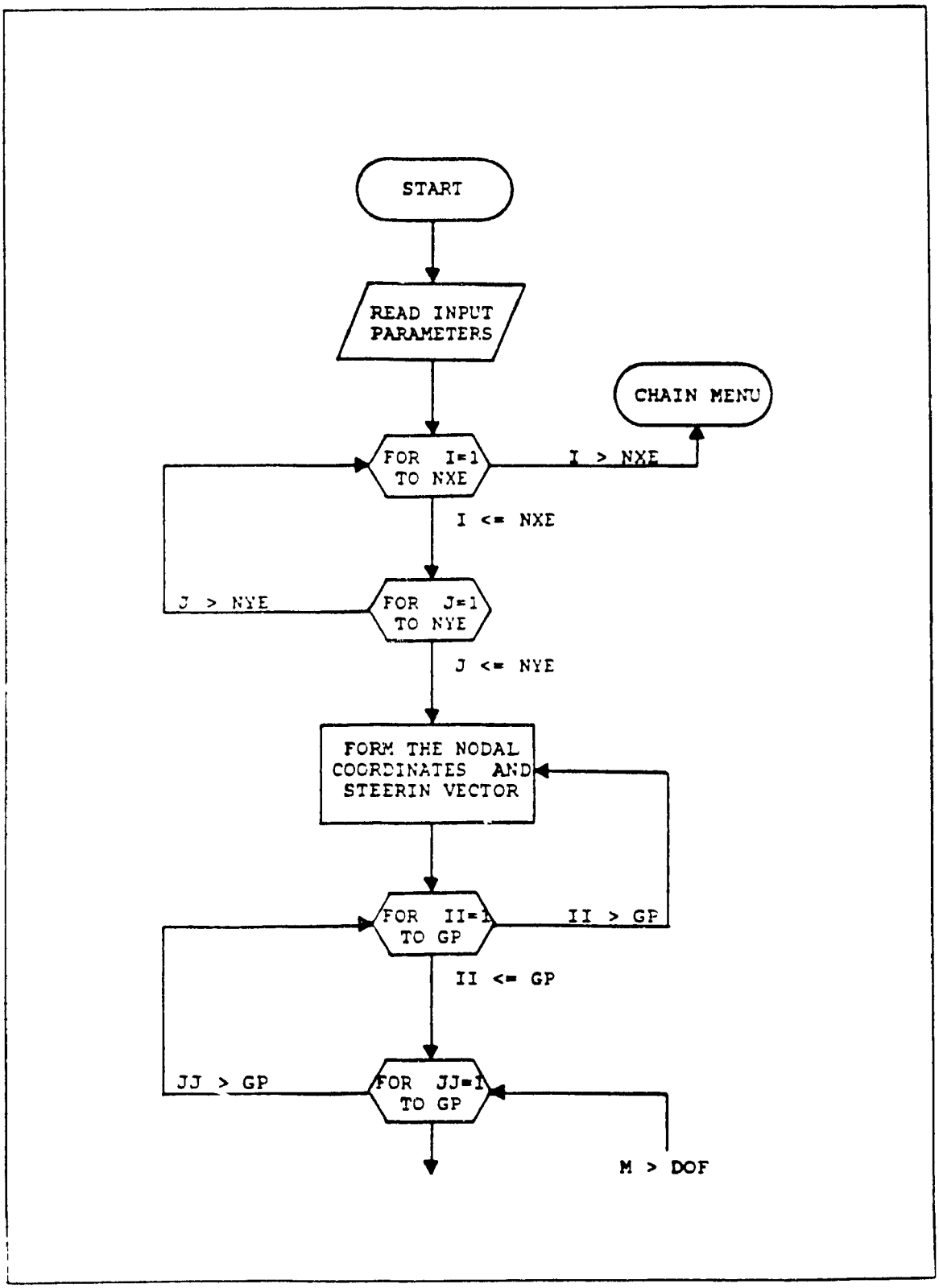


Figure C.4: Flowchart of ELAST4.EXE

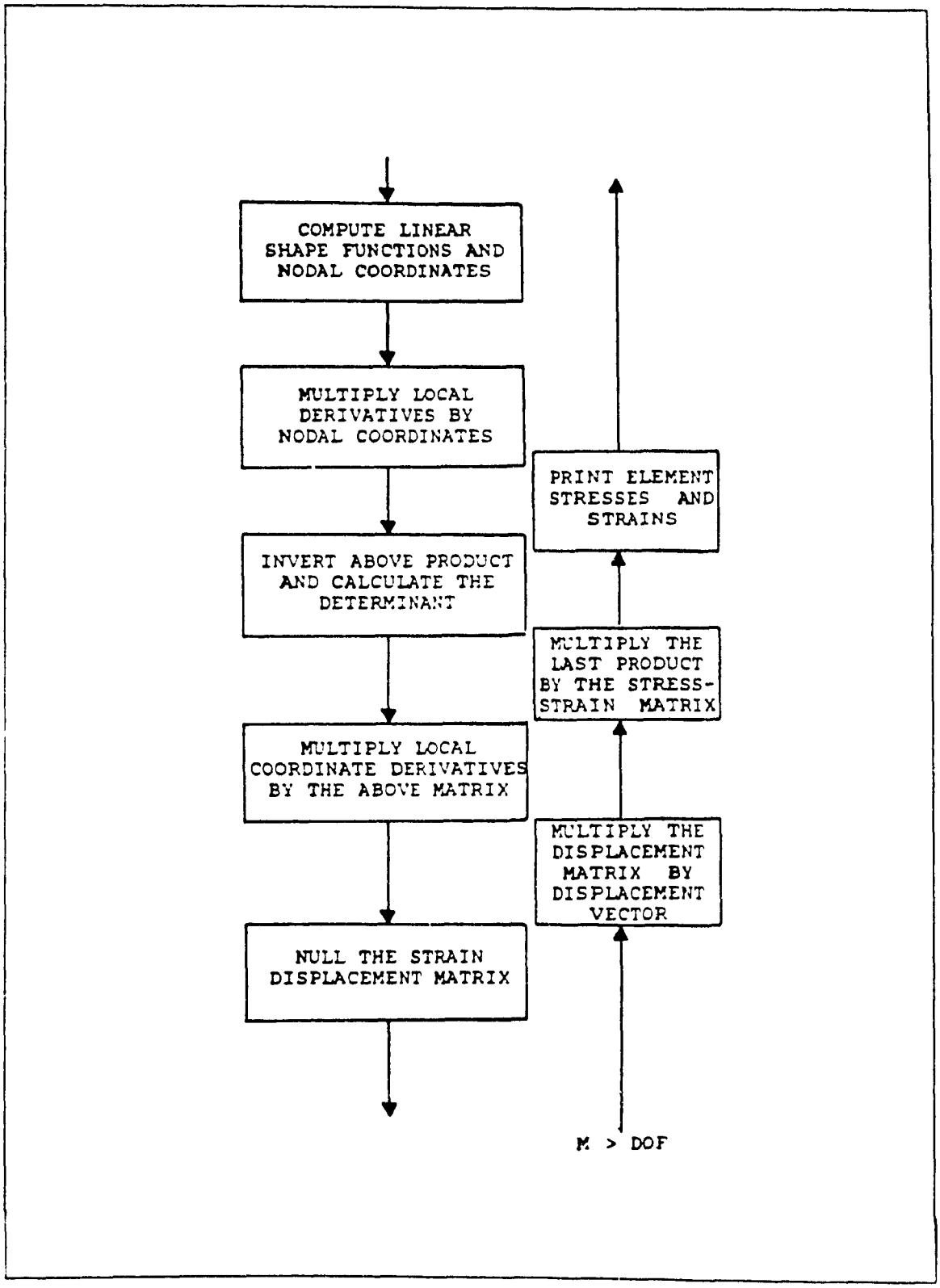


Figure C.4: Flowchart of ELAST4.EXE (Con't)

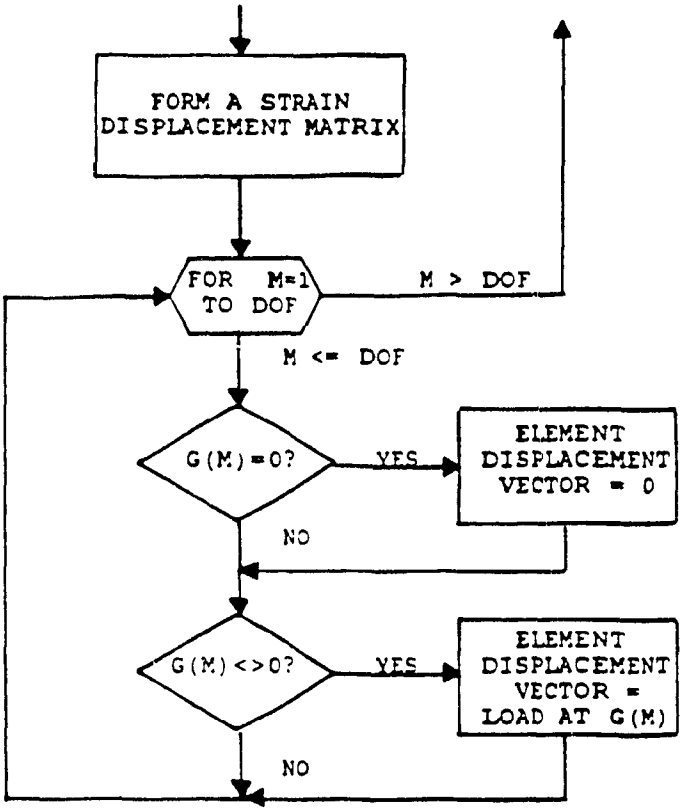


Figure C.4: Flowchart of ELAST4.EXE (Con't)

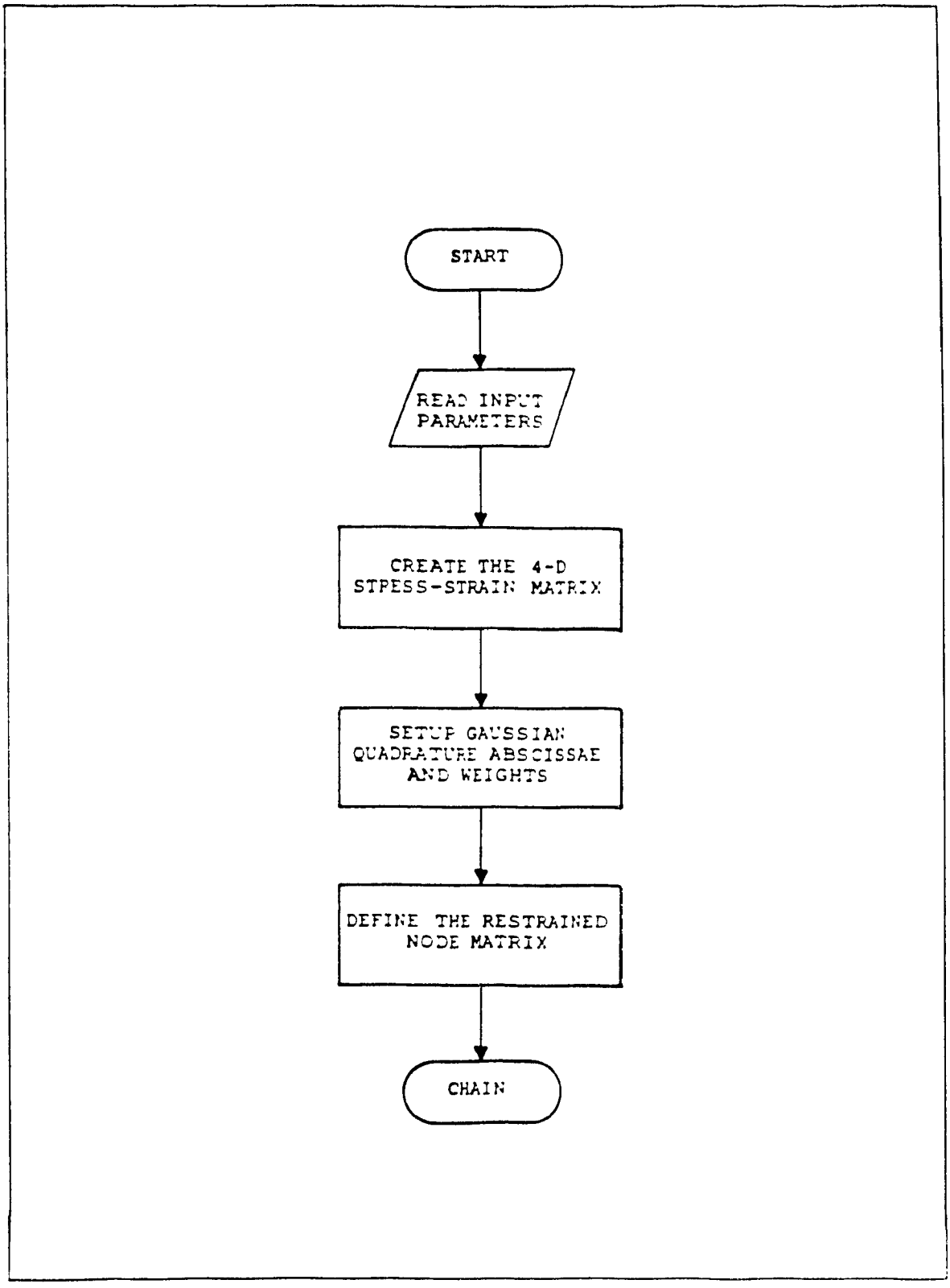


Figure C.5: Flowchart of VONMSE1.EXE

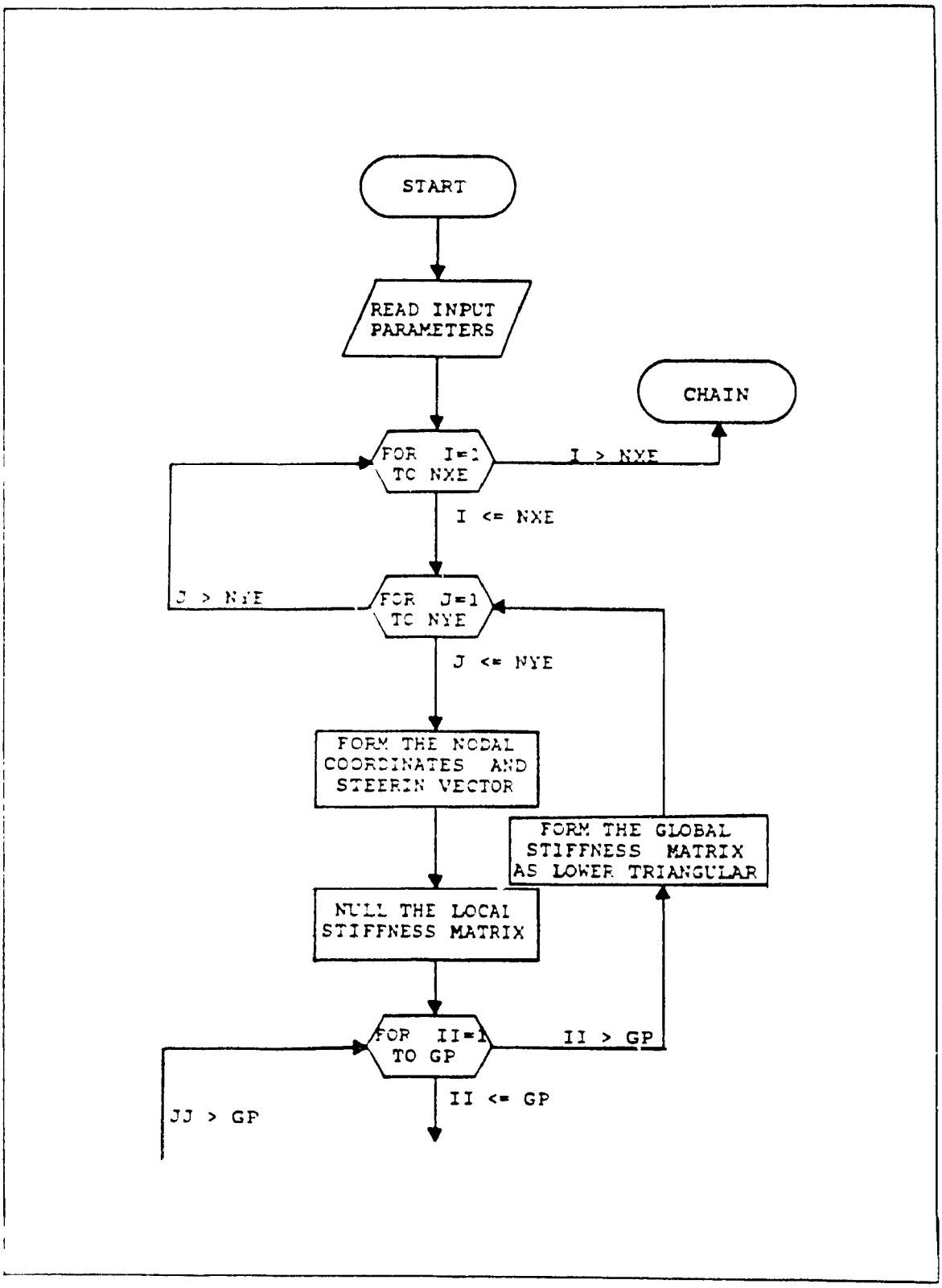


Figure C.6: Flowchart of VONMSE2.EXE

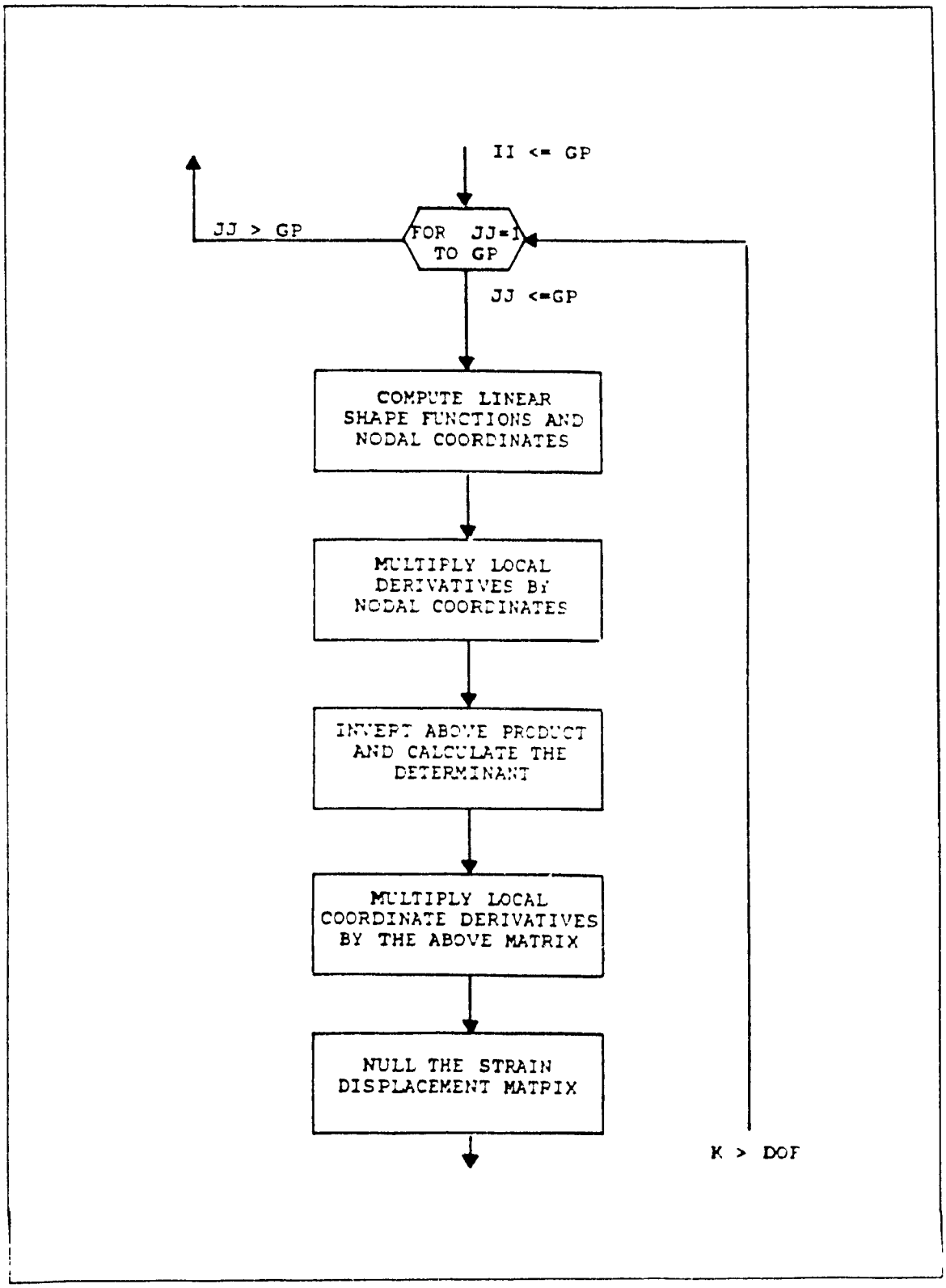


Figure C.6: Flowchart of VONMSE2.EXE (Con't,

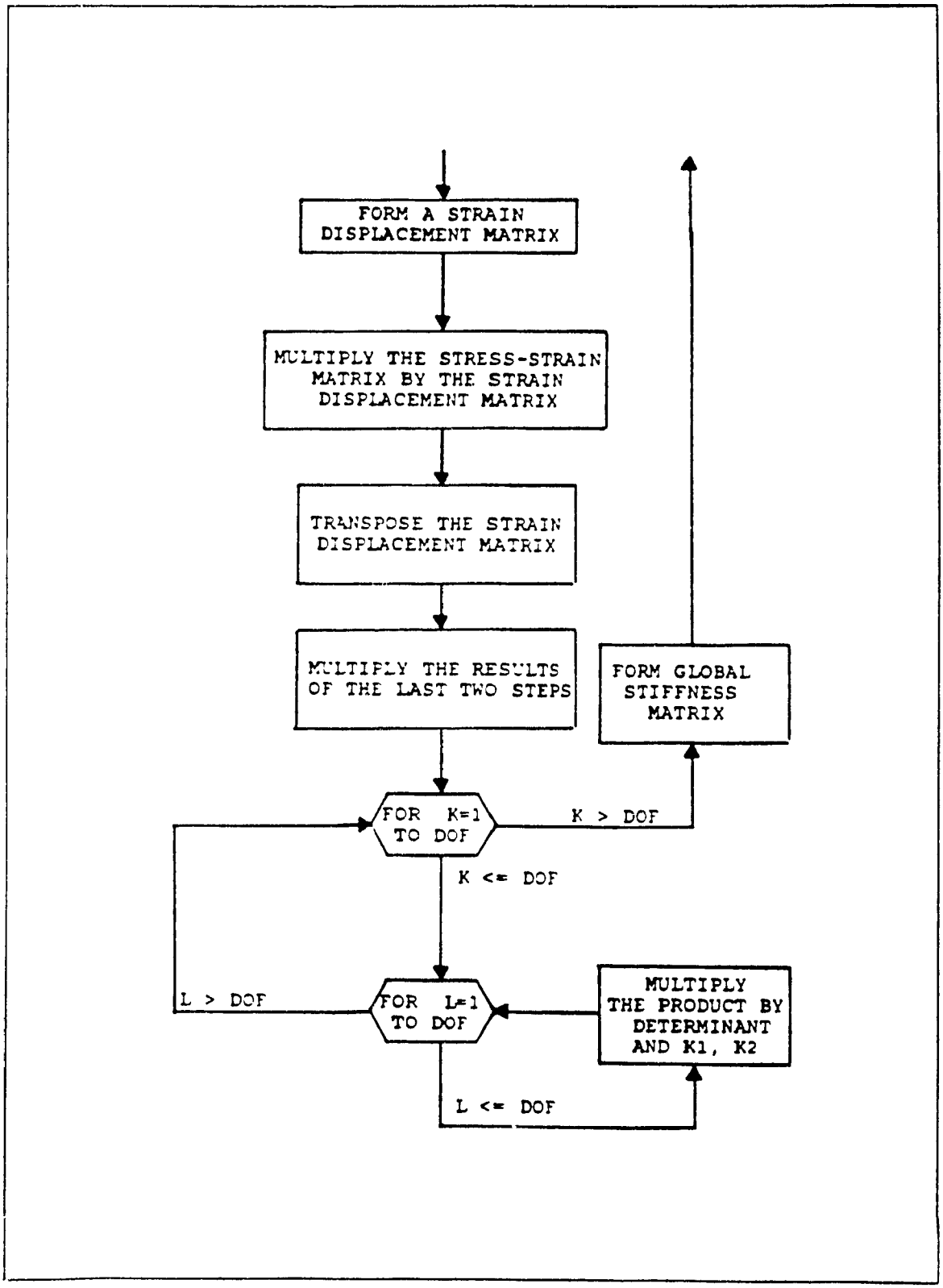


Figure C.6: Flowchart of VONMSE2.EXE (Con't)

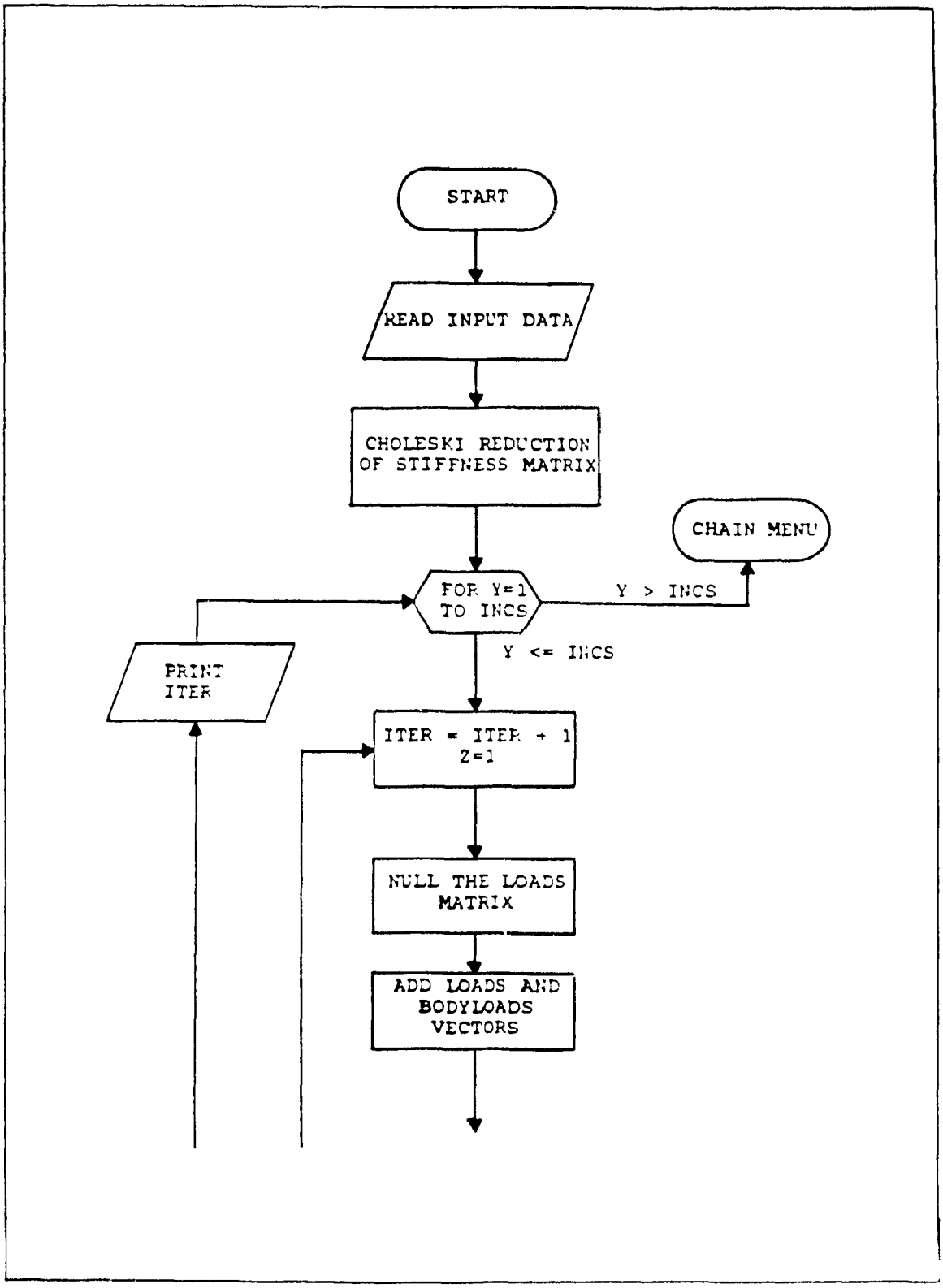


Figure C.7: Flowchart of VONMSE3.EXE

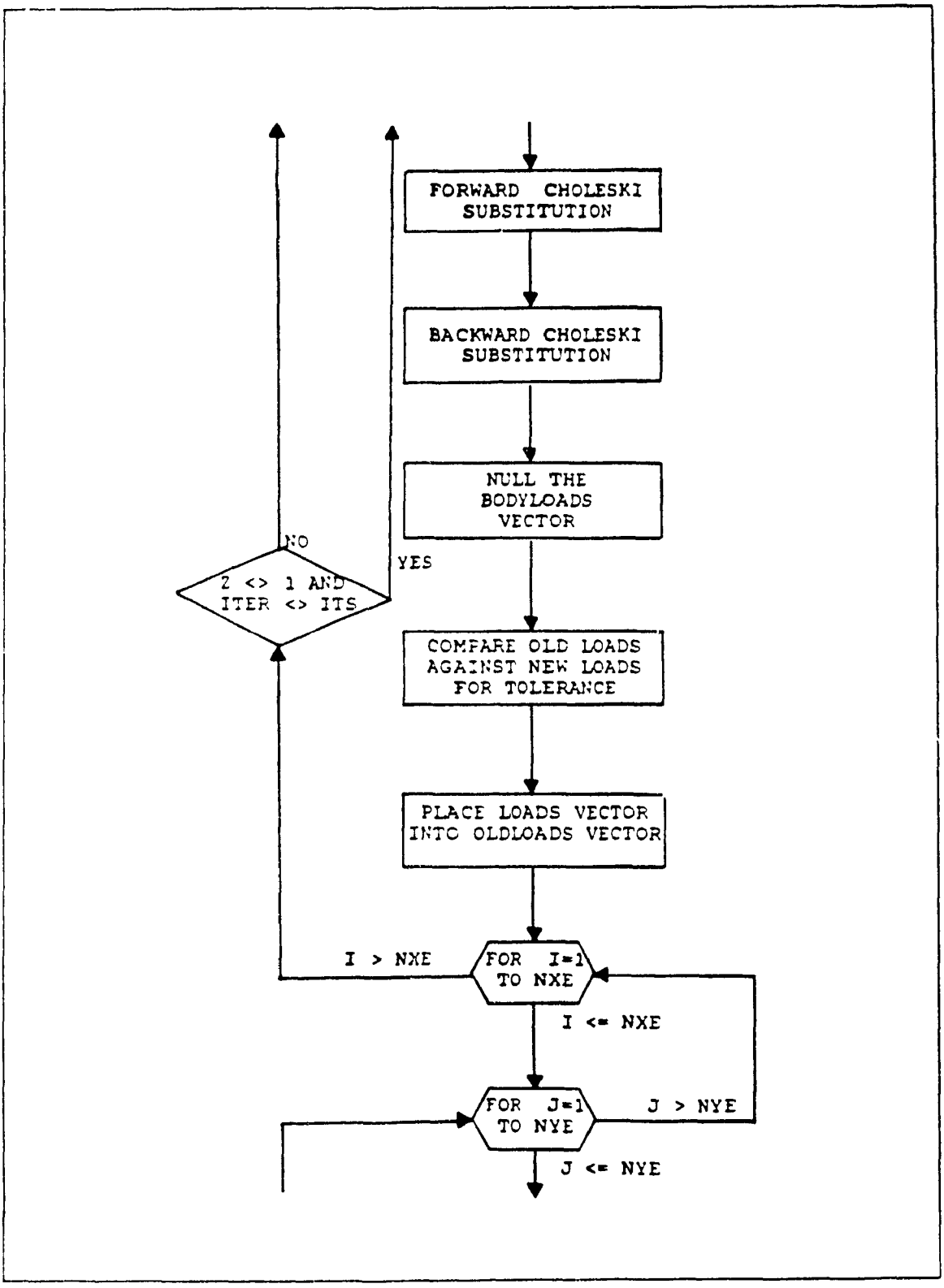


Figure C.7: Flowchart of VONMSE3.EXE (Con't)

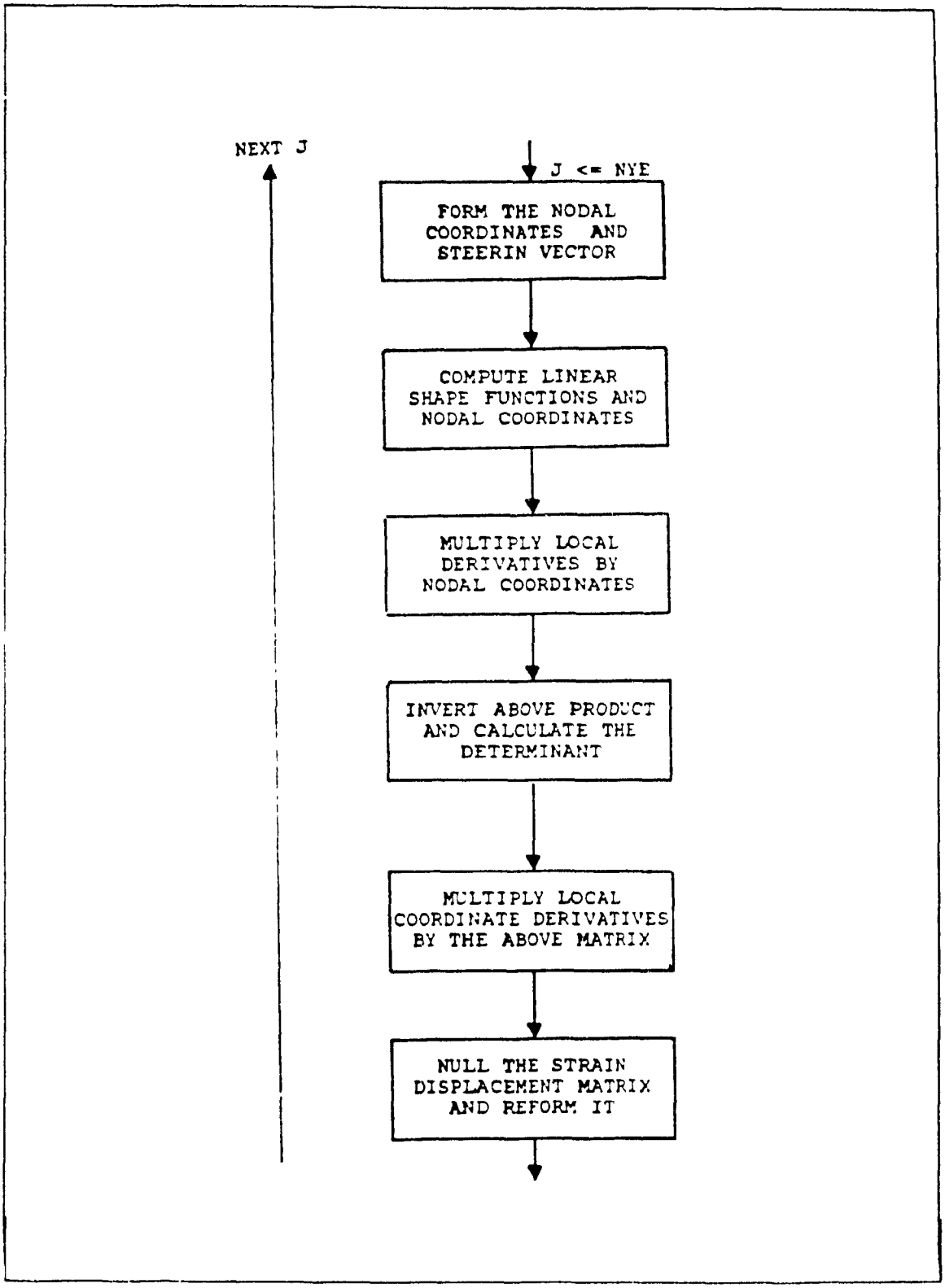


Figure C.7: Flowchart of VONMSE3.EXE (Con't)

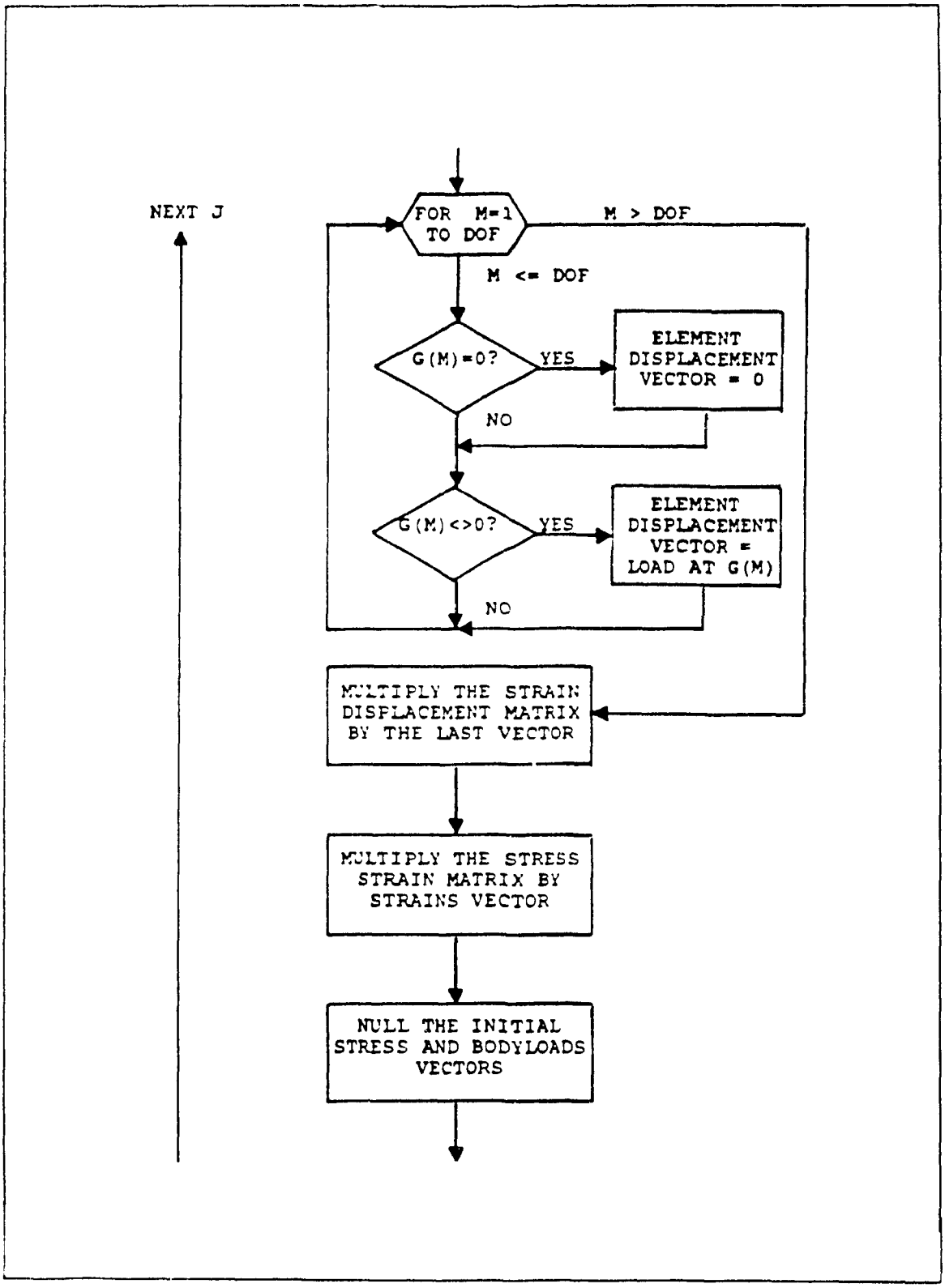


Figure C.7: Flowchart of VONMSE3.EXE (Con't)

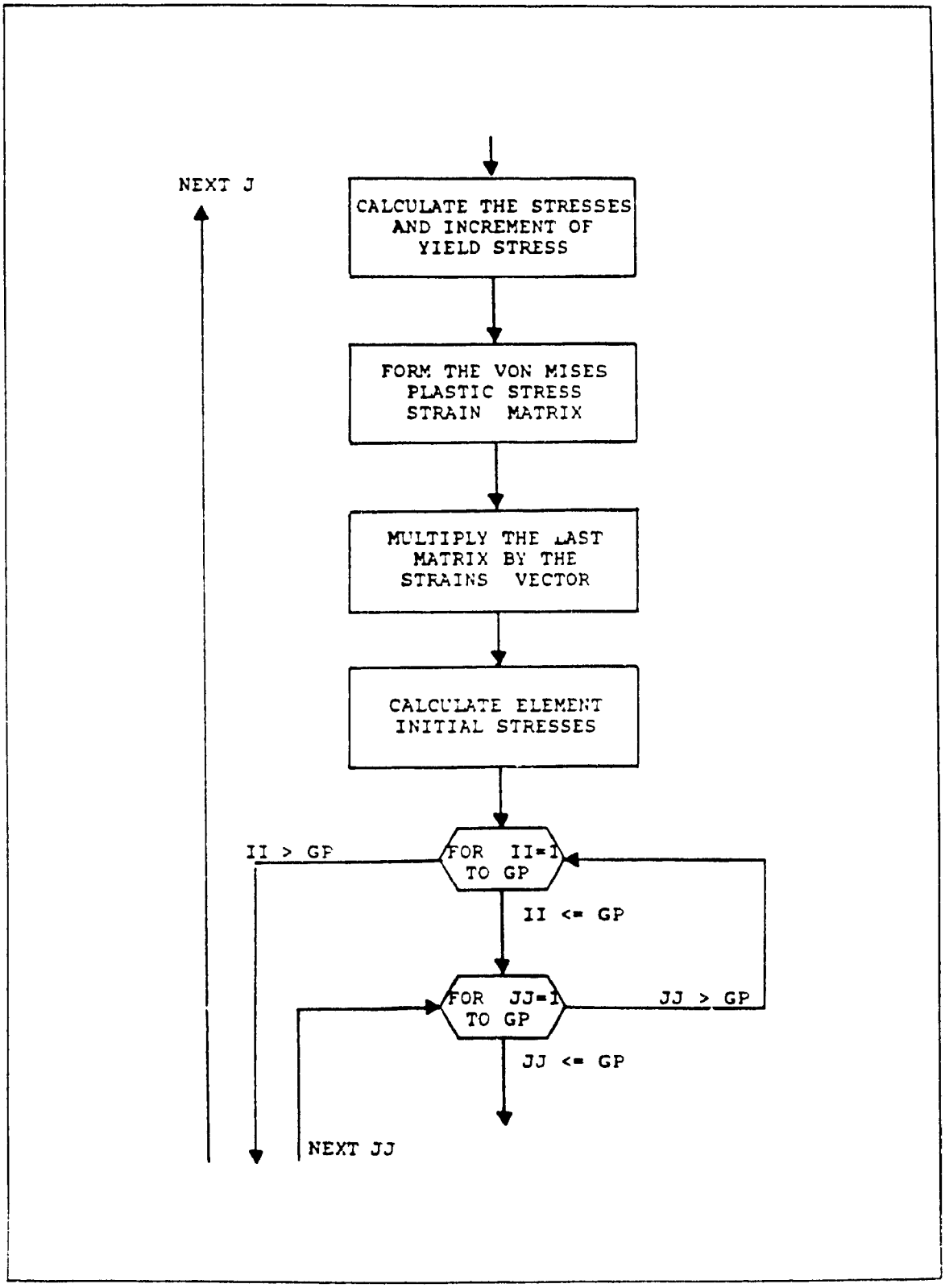


Figure C.7: Flowchart of VONMSE3.EXE (Con't)

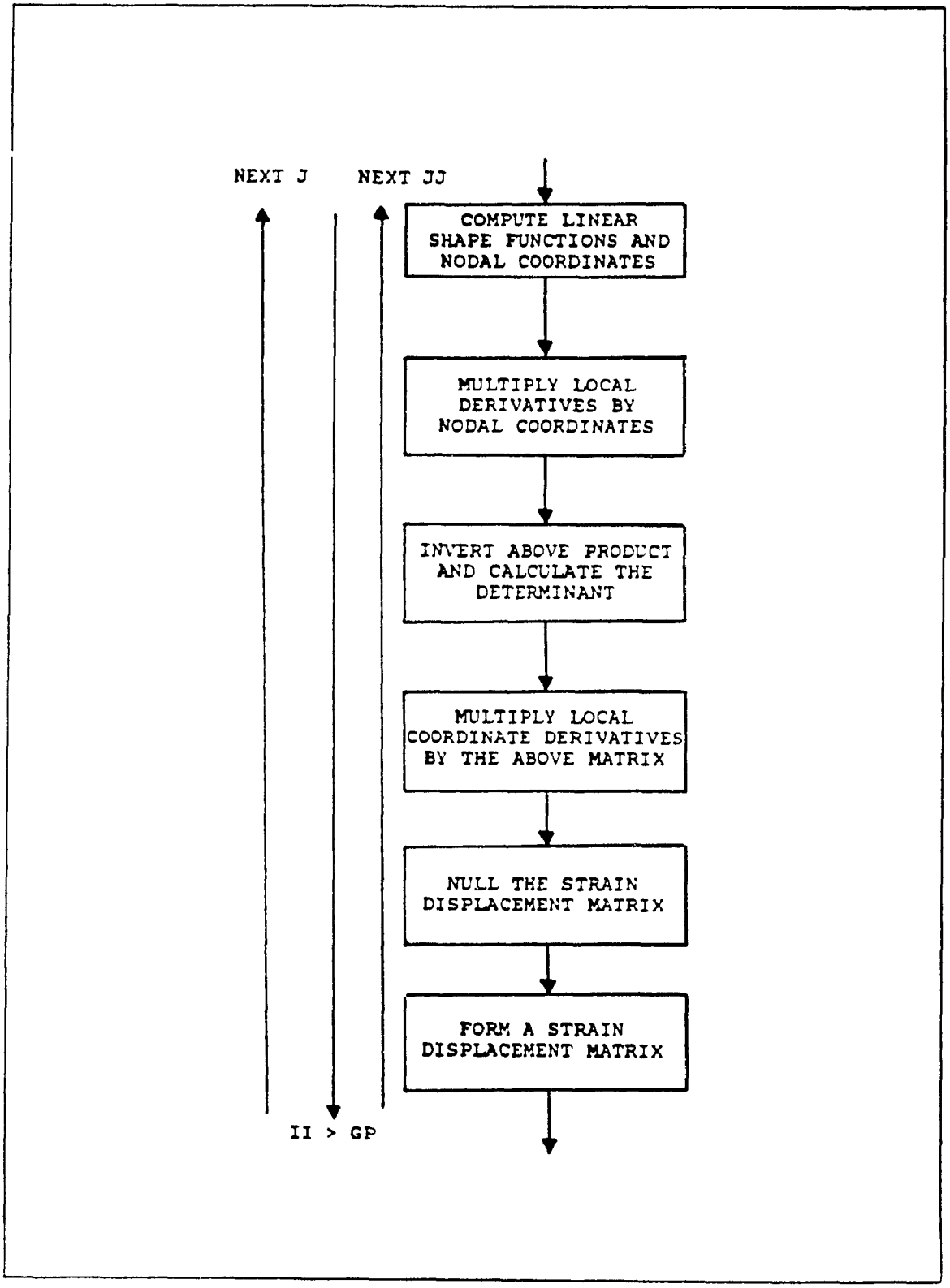


Figure C.7: Flowchart of VONMSE3.EXE (Con't)

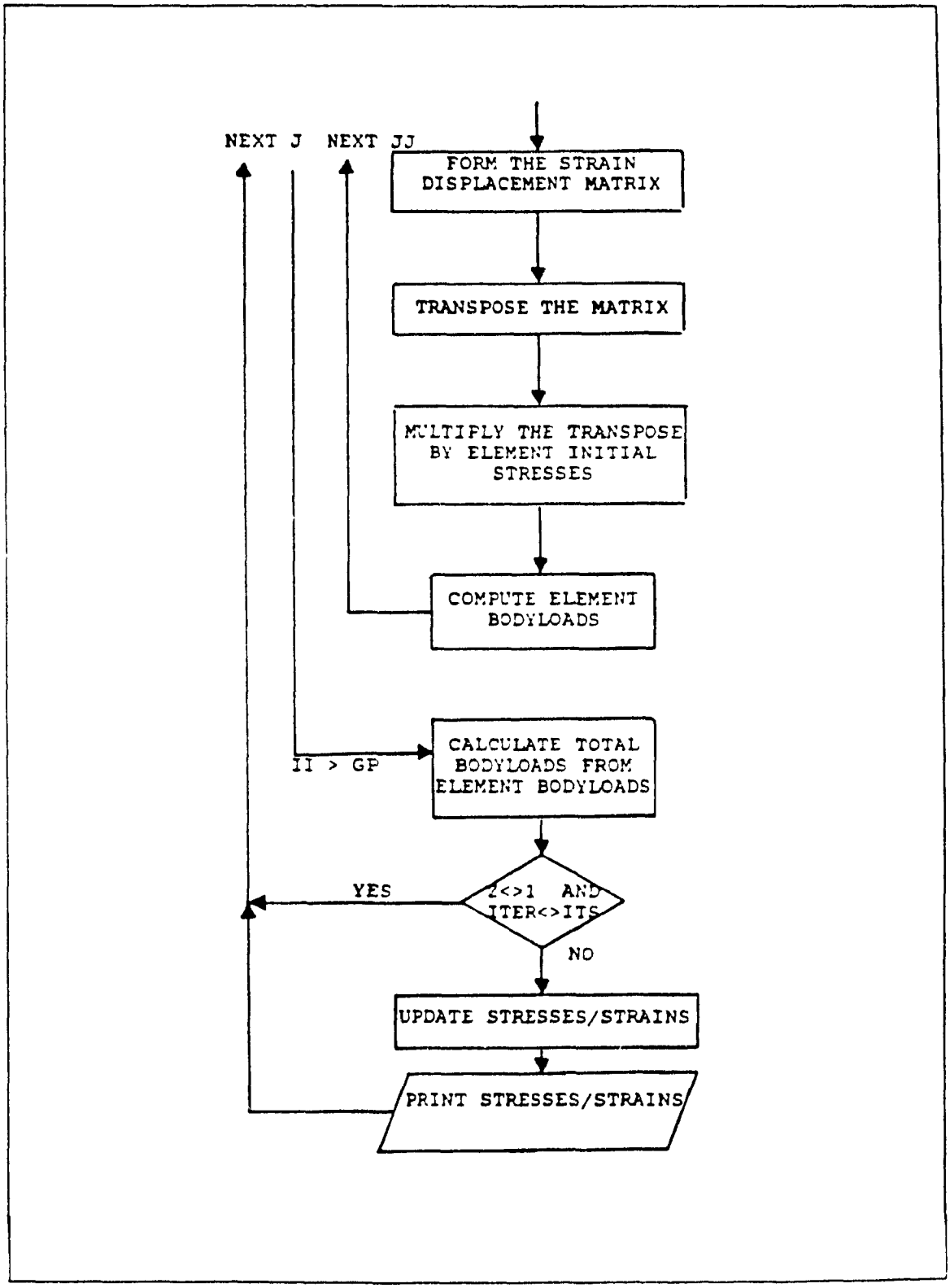


Figure C.7: Flowchart of VONMSE3.EXE (Con't)

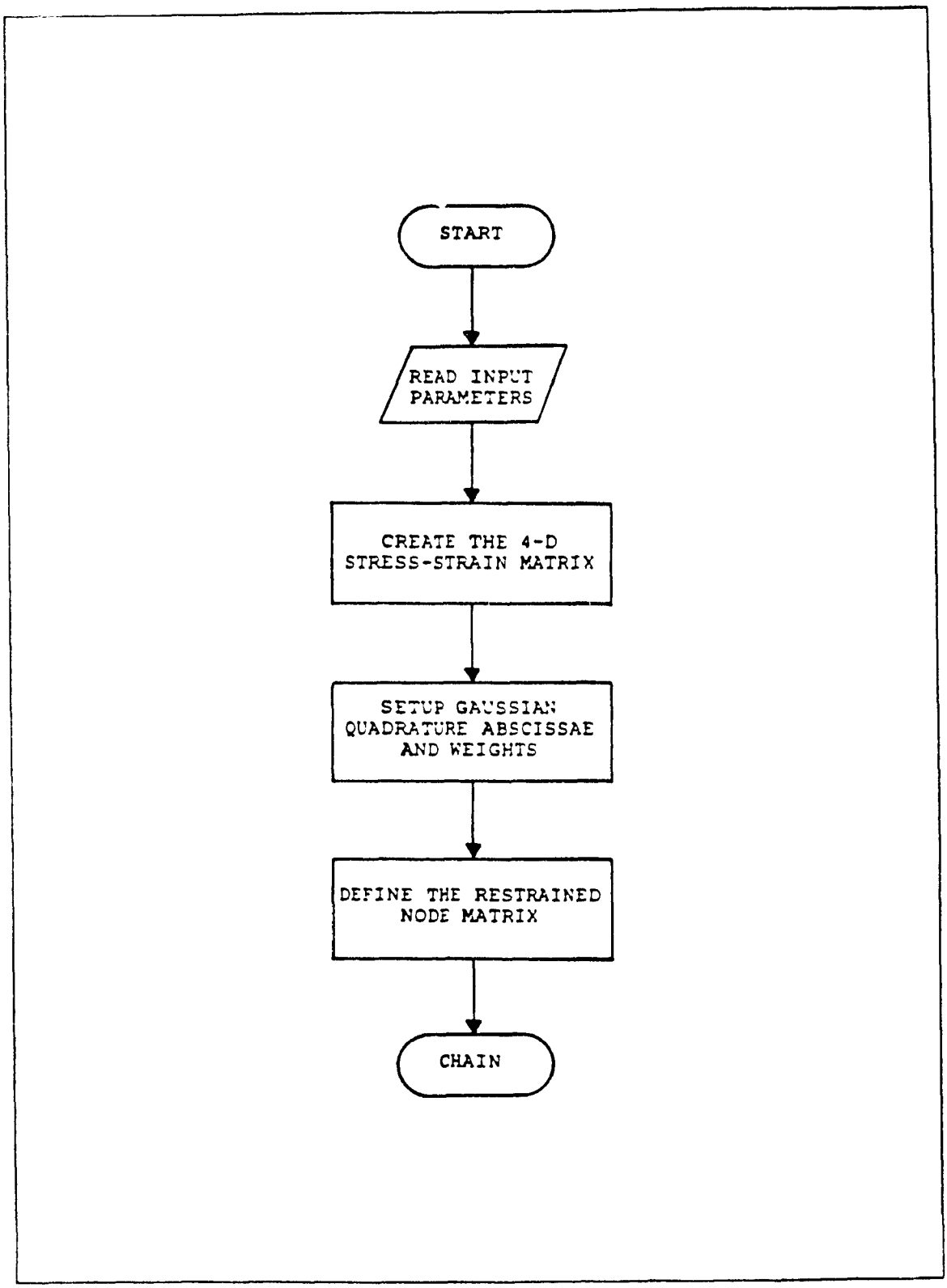


Figure C.8: Flowchart of MOHRCOL1.EXE

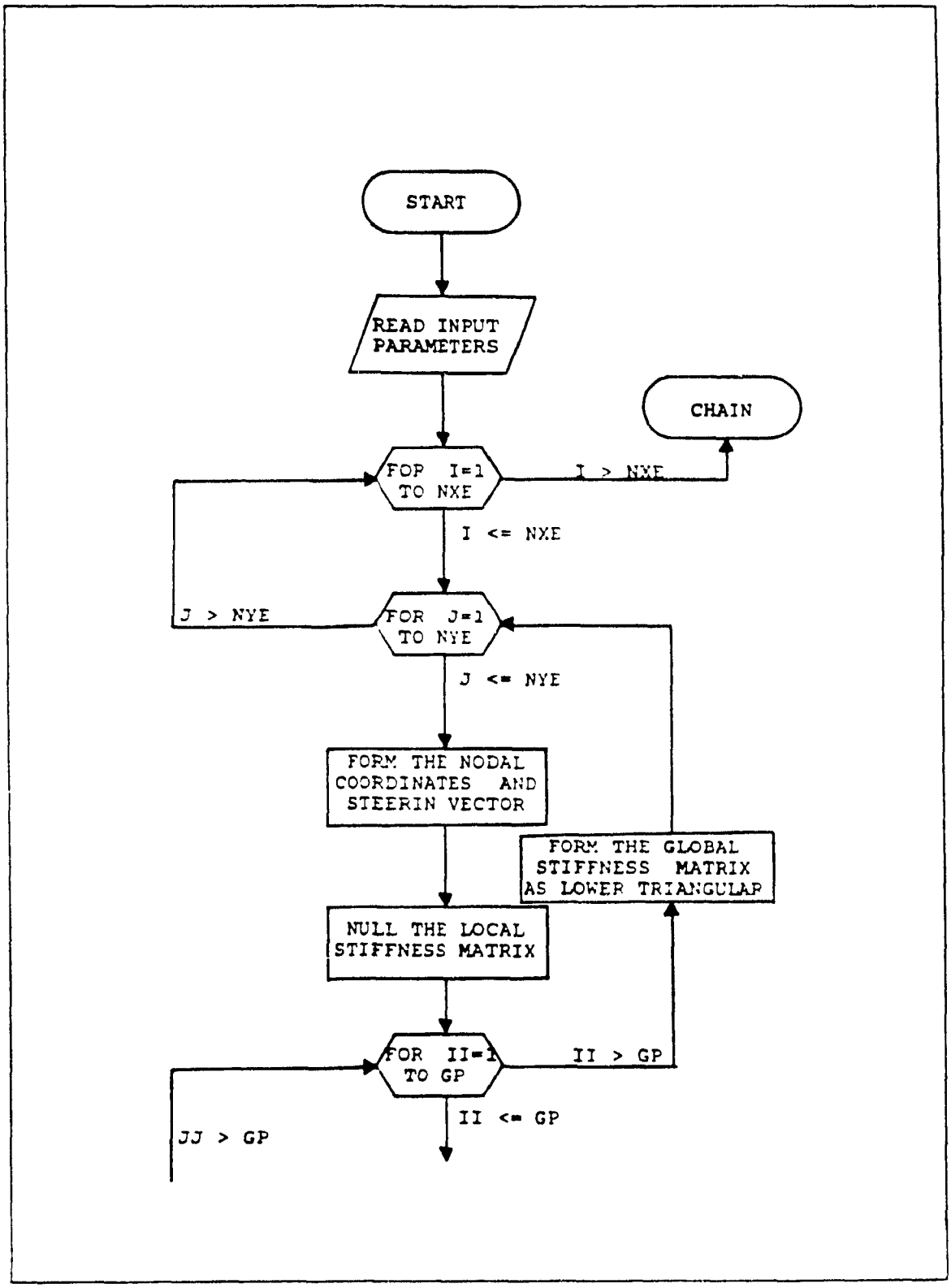


Figure C.9: Flowchart of MOHRCOL2.EXE

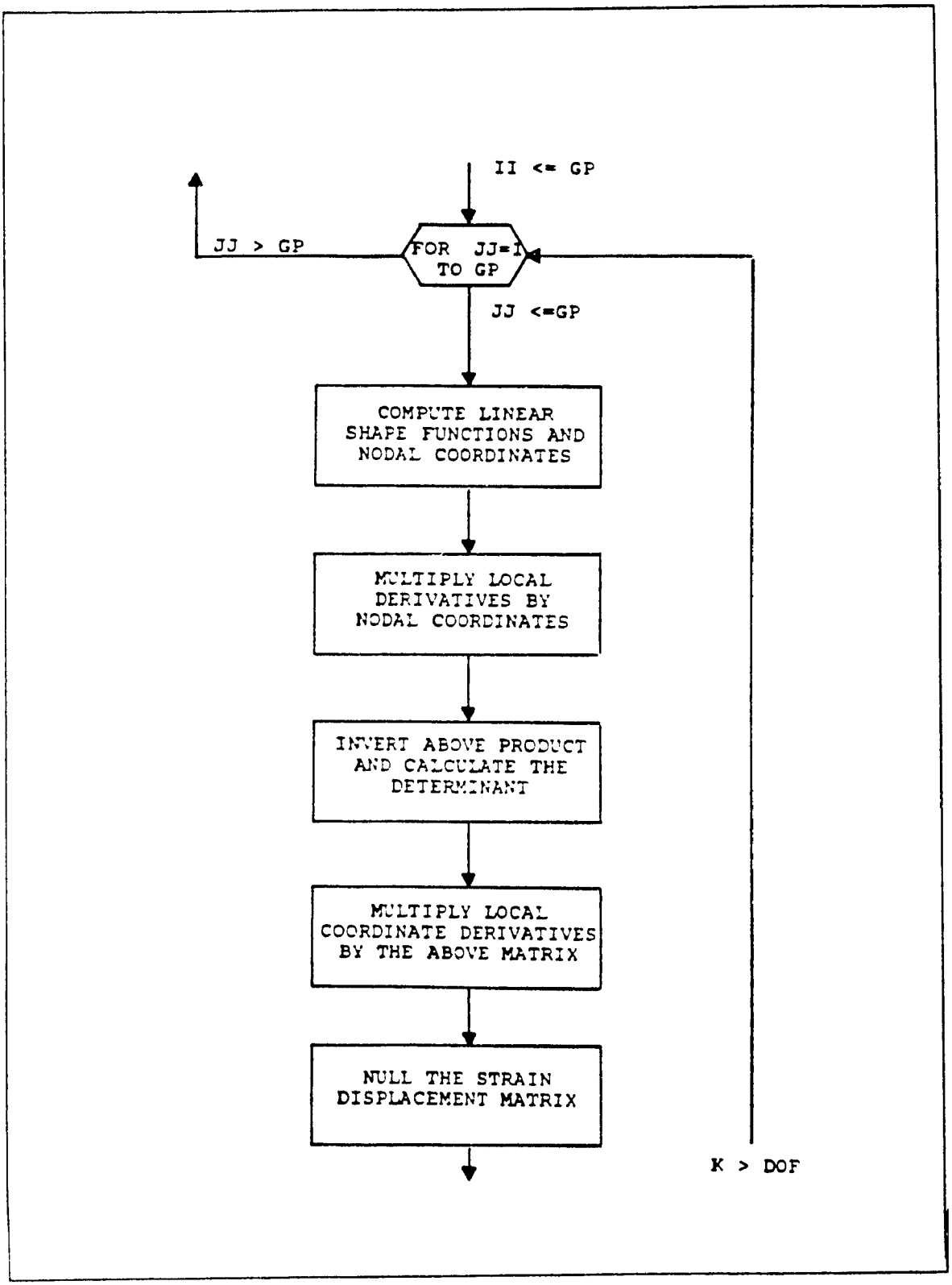


Figure C.9: Flowchart of MOHRCOL2.EXE (Con't)

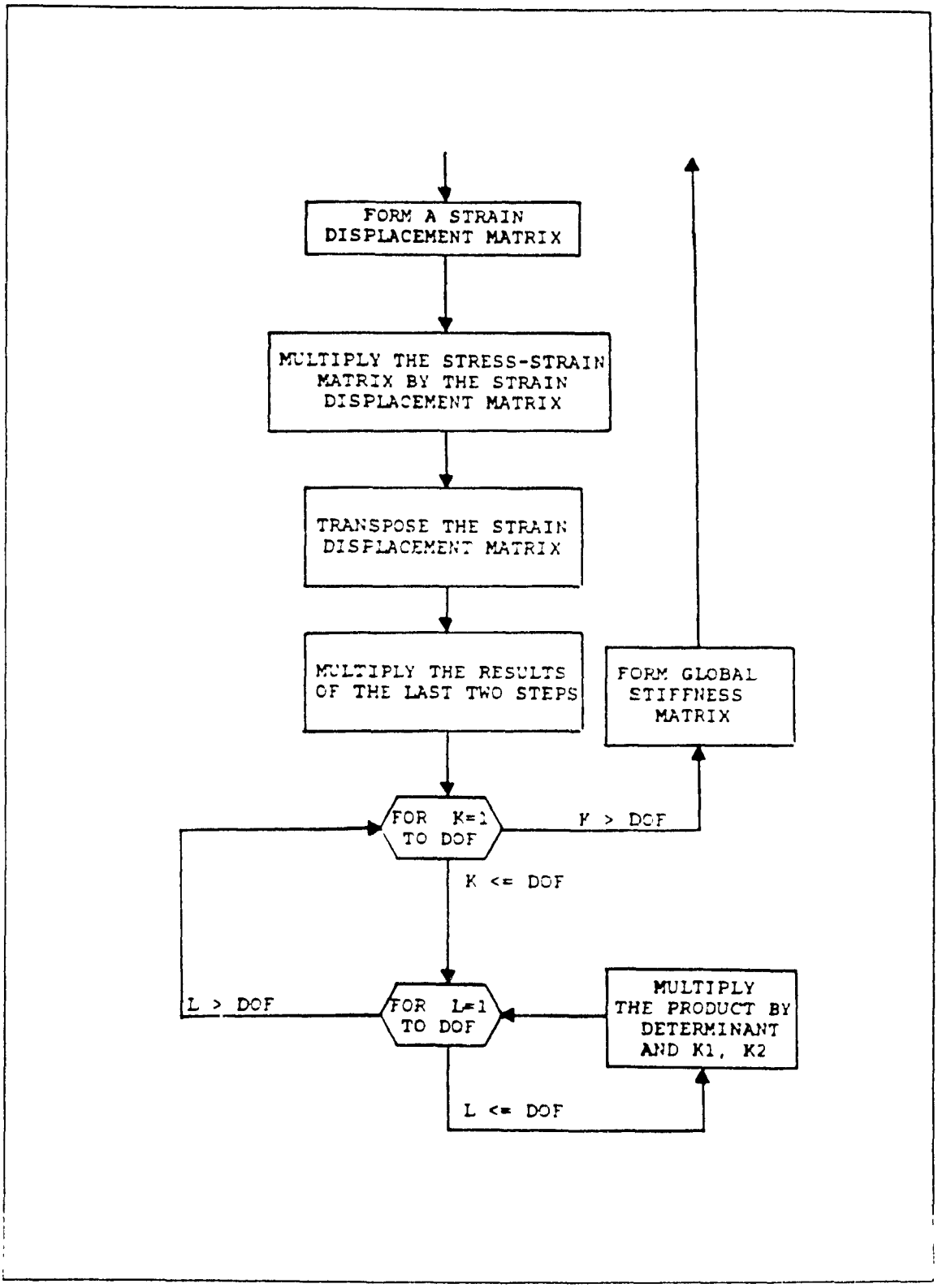


Figure C.9: Flowchart of MOHRCOL2.EXE (Con't)

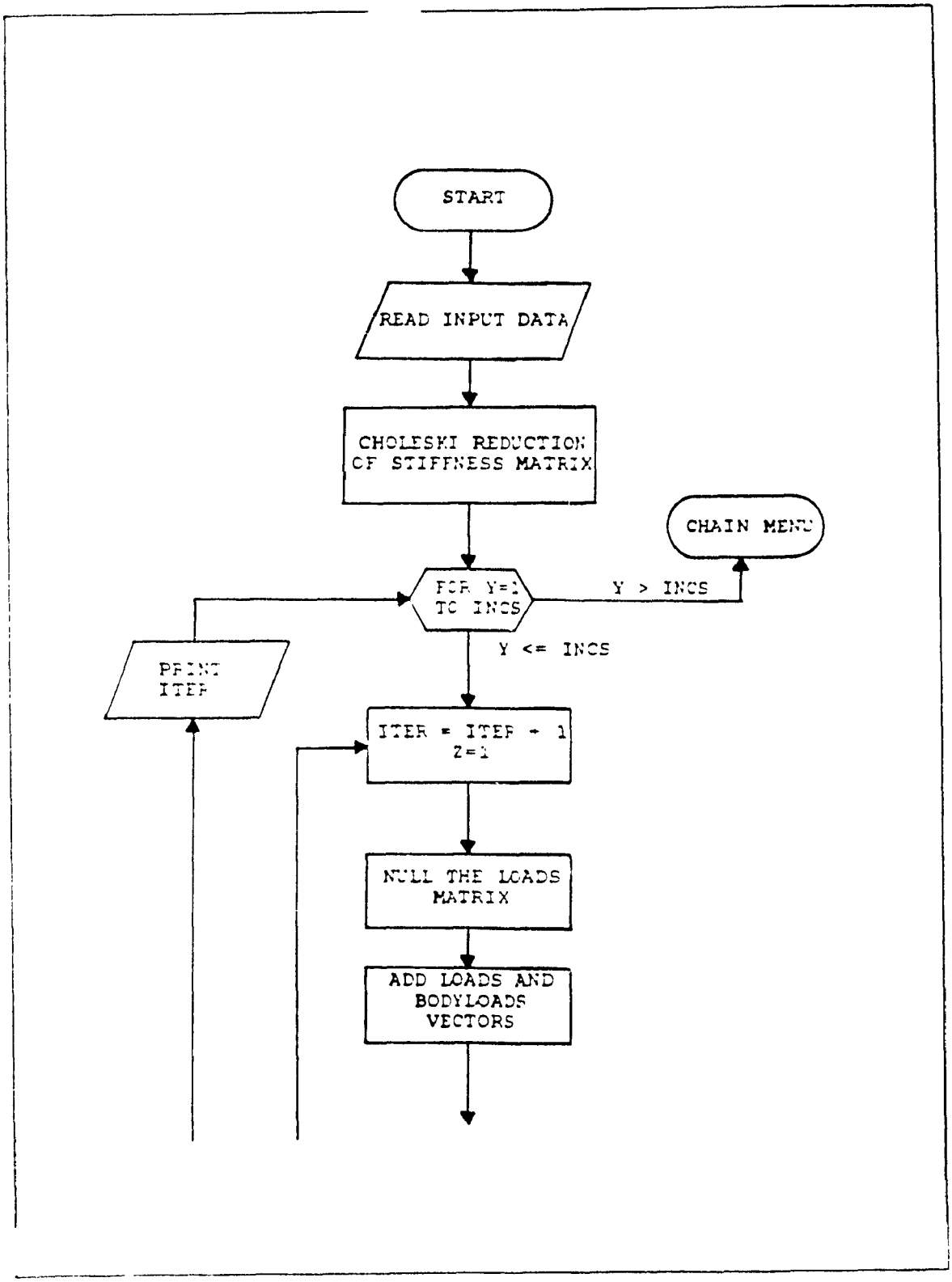


Figure C.11: Flowchart of MOHRCOL3.EXE

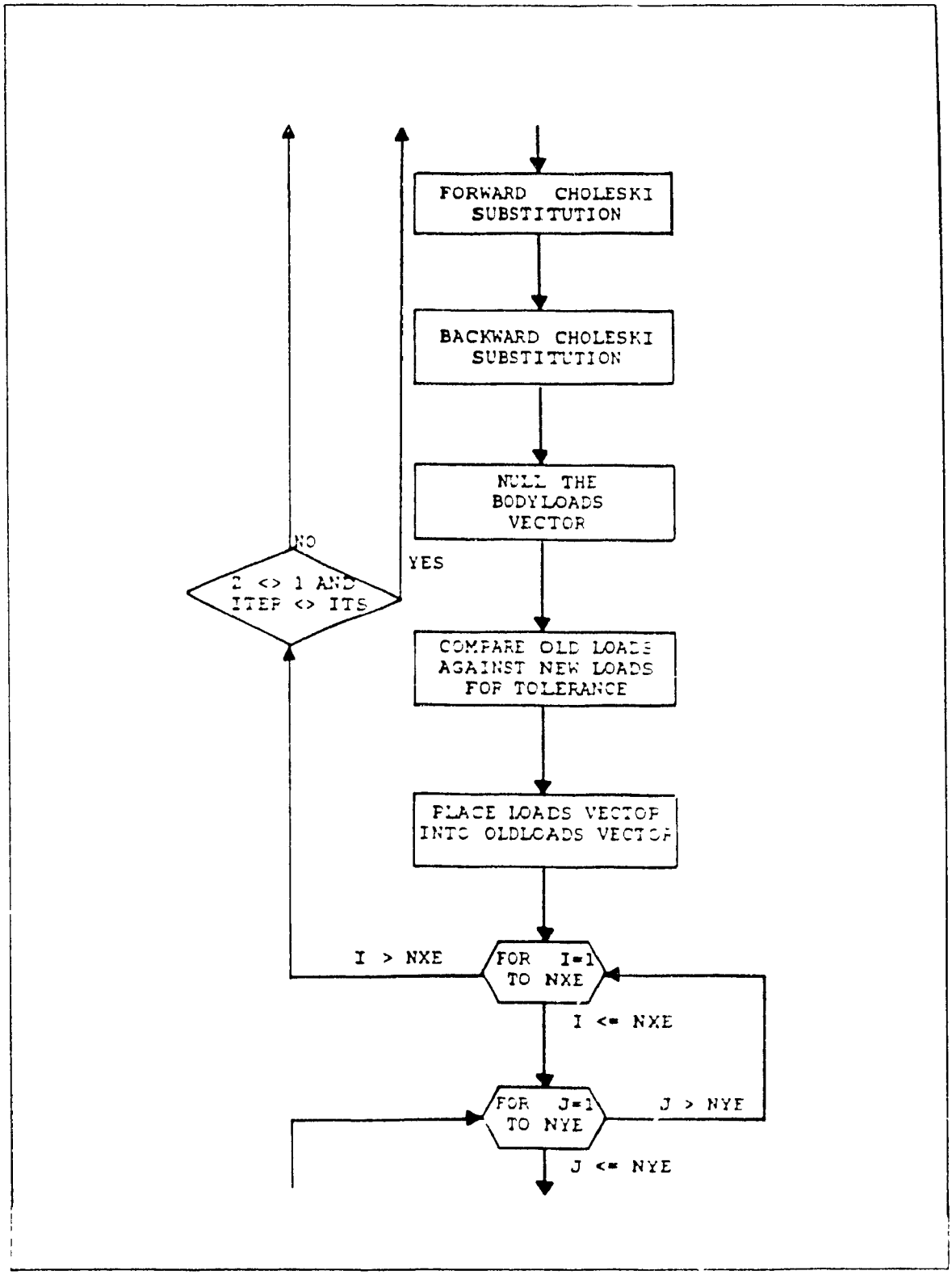


Figure C.10: Flowchart of MOHRCOL3.EXE (Con't)

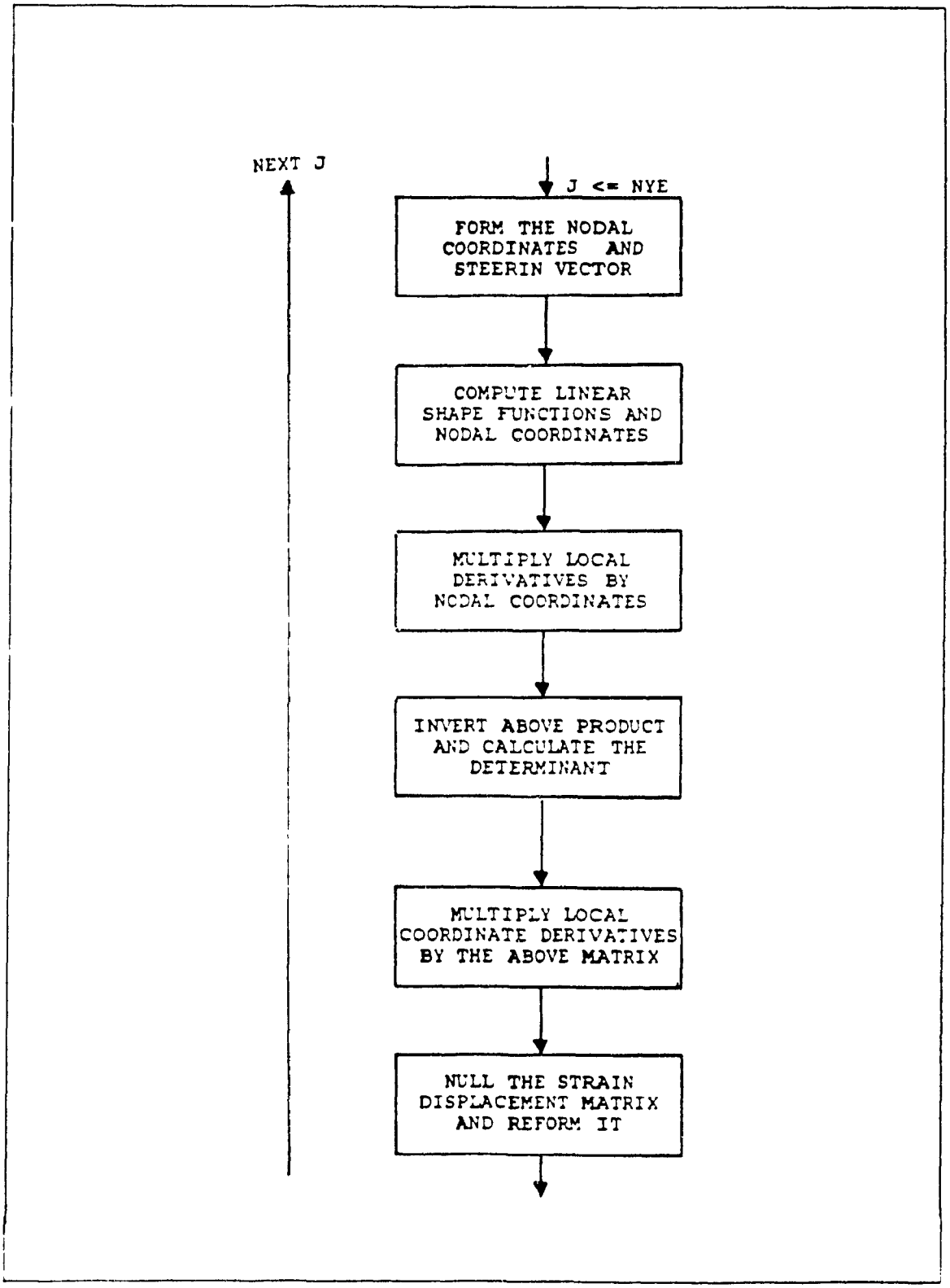


Figure C.10: Flowchart of MOHRCOL3.EXE (Con't)

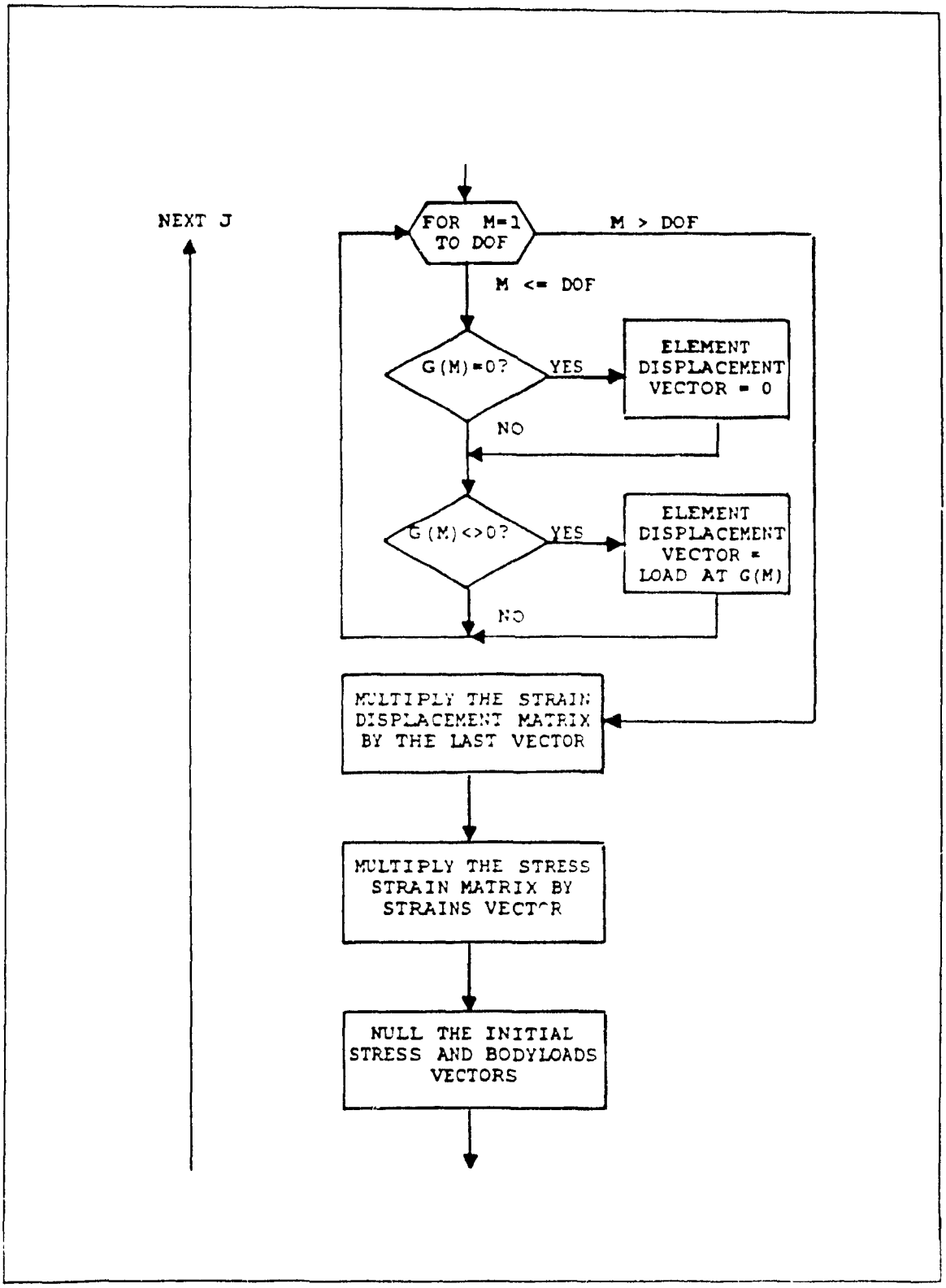


Figure C.10: Flowchart of MOHRCOL3.EXE (Con't)

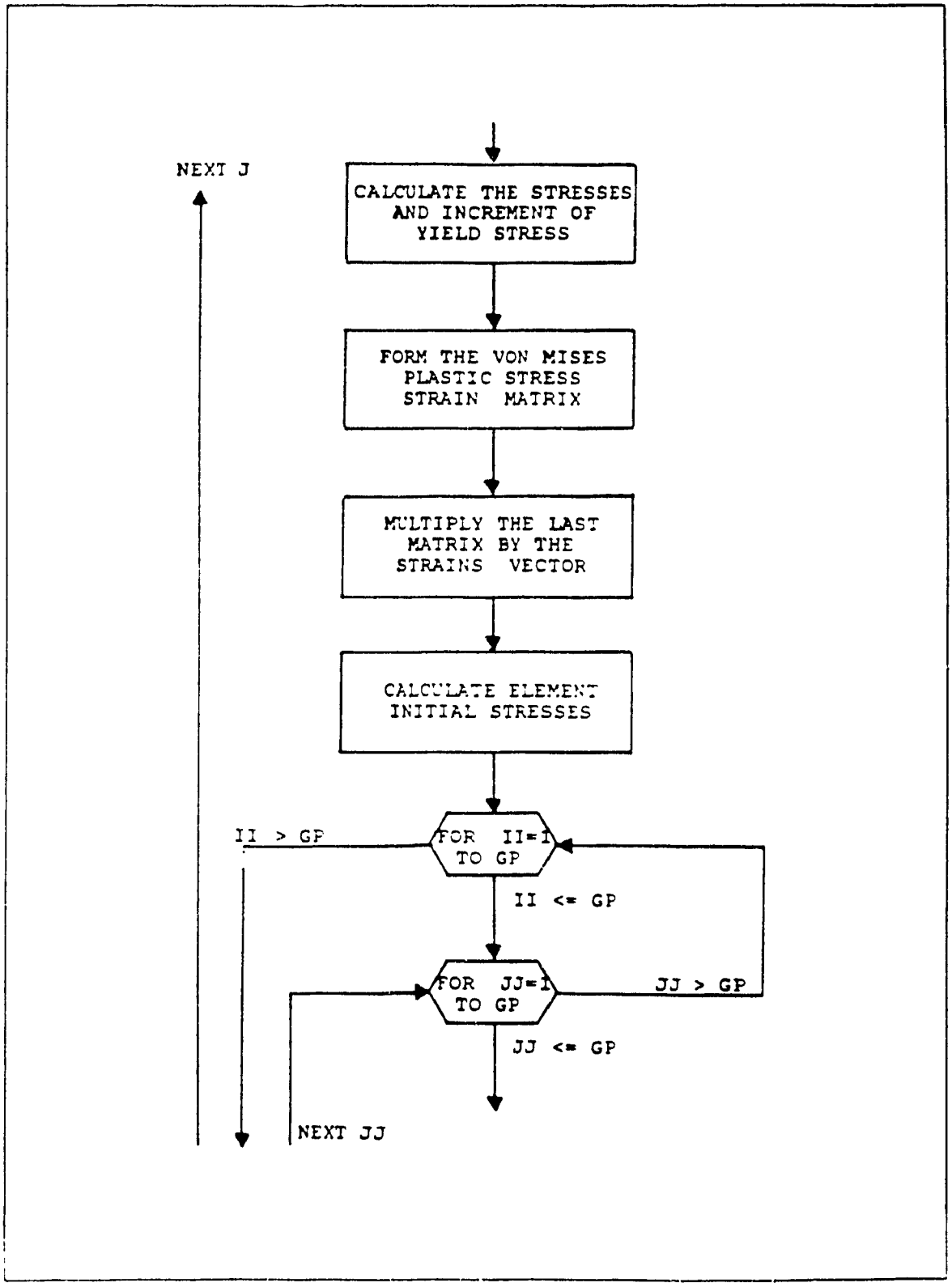


Figure C.10: Flowchart of MOHRCOL3.EXE (Con't)

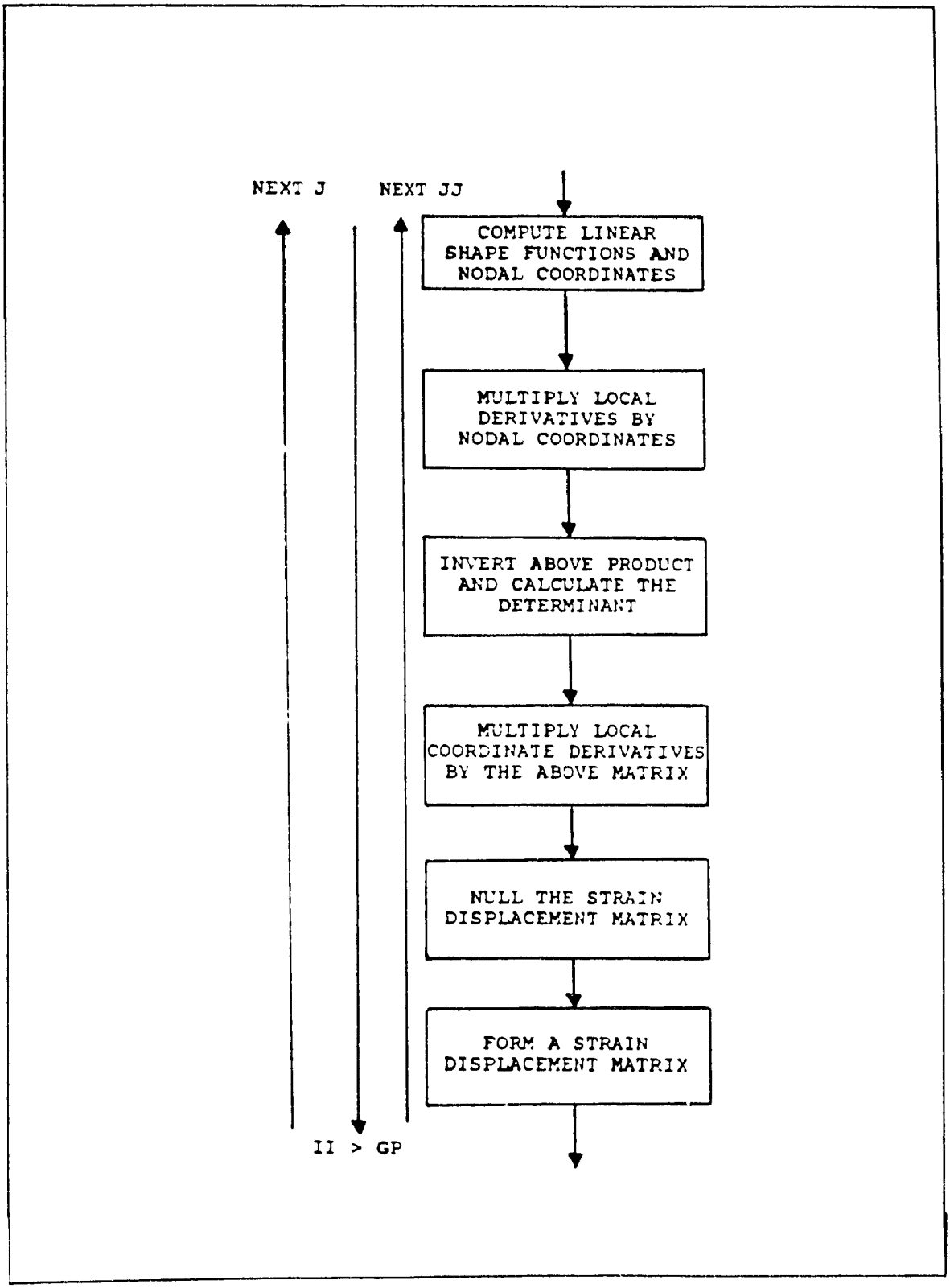


Figure C.10: Flowchart of MOHRCOL3.EXE (Con't)

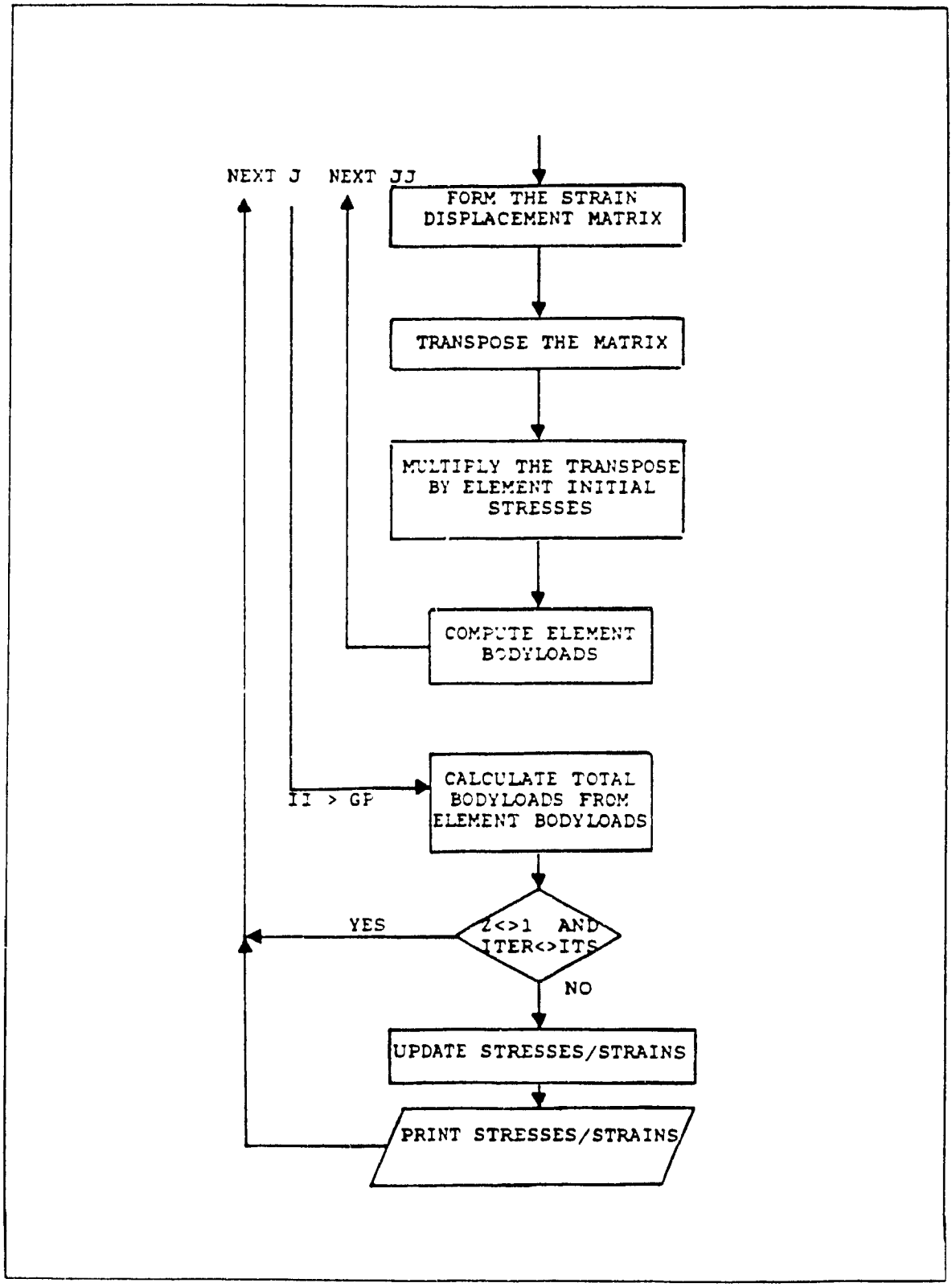


Figure C.10: Flowchart of MOHRCOL3.EXE (Con't)

Appendix-D

Program Listings

The computer programs used in the thesis are listed in this Appendix. These programs were originally written in FORTRAN and have been rewritten and compiled in QuickBasic. The programs are:

```
1.0 ELASTIC ANALYSIS PROGRAM

ELAST1.EXE

CLEAR
REM
REM *** PLANE STRAIN OF ELASTIC SOLID USING 4 NODE ***
REM ***           QUADRILATERAL ELEMENTS           ***
REM
DIM DEE(3, 3), SAMP(7, 2), NF(100, 2)
REM
REM ***                INITIALZE DATA              ***
REM
NODOF = 2:      REM Number of Freedoms per Node
OPEN "INPT.FLE" FOR INPUT AS #1
  INPUT #1, NXE: REM number of elements in x direction
  INPUT #1, NYE: REM number of elements in y direction
  INPUT #1, N:   REM total number of freedoms in mesh
  INPUT #1, W:   REM half bandwidth
  INPUT #1, NN:  REM number of nodes in mesh
  INPUT #1, RN:  REM restrained nodes in mesh
  INPUT #1, NL:  REM number of loaded freedoms
  INPUT #1, GP:  REM Gaussian integration order
  INPUT #1, AA:  REM element size in x direction
  INPUT #1, BB:  REM element size in y direction
  INPUT #1, E:   REM Young's Modulus
  INPUT #1, V:   REM Poisson's Ratio
  REM
  REM *** SET MATRICES TO ZERO ***
  REM
  FOR I = 1 TO 3
      FOR J = 1 TO 3
          DEE(I, J) = 0
      NEXT J
  NEXT I
  REM
  REM *** STRESS STRAIN MATRIX FOR PLANE ELASTIC STRAIN ***
  REM
  V1 = V / (1 - V)
  VV = (1 - 2 * V) * .5 / (1 - V)
  DEE(1, 1) = 1
  DEE(2, 2) = 1
  DEE(3, 3) = VV
  DEE(1, 2) = V1
  DEE(2, 1) = V1
  FOR I = 1 TO 3
      FOR J = 1 TO 3
          DEE(I, J) = DEE(I, J) * E / (2 * (1 + V) * VV)
      NEXT J
  NEXT I
  REM
  REM *** GAUSSIAN QUADRATURE ABSCISSAE AND WEIGHTS ***
  REM
  GP = INT(GP)
  IF GP = 1 THEN GOTO 700
  IF GP = 2 THEN GOTO 100
```

```
        IF GP = 3 THEN GOTO 300
        IF GP = 4 THEN GOTO 400
        IF GP = 5 THEN GOTO 500
        IF GP = 6 THEN GOTO 600
        IF GP = 7 THEN GOTO 200
        GP = 2
100     SAMP(1, 1) = 1 / SQR(3)
        SAMP(2, 1) = -SAMP(1, 1)
200     SAMP(1, 2) = 1
        SAMP(2, 2) = 1
        GOTO 700
300     SAMP(1, 1) = .2 * SQR(15)
        SAMP(2, 1) = 0
        SAMP(3, 1) = -SAMP(1, 1)
        SAMP(1, 2) = 5 / 9
        SAMP(2, 2) = 8 / 9
        SAMP(3, 2) = SAMP(1, 2)
        GOTO 700
400     SAMP(1, 1) = .861136311594053
        SAMP(2, 1) = .339981043584856
        SAMP(3, 1) = -SAMP(2, 1)
        SAMP(4, 1) = -SAMP(1, 1)
        SAMP(1, 2) = .3847854845137454
        SAMP(2, 2) = .652145154862546
        SAMP(3, 2) = SAMP(2, 2)
        SAMP(4, 2) = SAMP(1, 2)
        GOTO 700
500     SAMP(1, 1) = .906179845838664
        SAMP(2, 1) = .538469310105683
        SAMP(3, 1) = 0
        SAMP(4, 1) = -SAMP(2, 1)
        SAMP(5, 1) = -SAMP(1, 1)
        SAMP(1, 2) = .236926885056189
        SAMP(2, 2) = .478628670499366
        SAMP(3, 2) = .568888888888889
        SAMP(4, 2) = SAMP(2, 2)
        SAMP(5, 2) = SAMP(1, 2)
        GOTO 700
        SAMP(1, 1) = .9324695142031521
        SAMP(2, 1) = .661209386466265
        SAMP(3, 1) = .238619186083197
        SAMP(4, 1) = -SAMP(3, 1)
        SAMP(5, 1) = -SAMP(2, 1)
        SAMP(6, 1) = -SAMP(3, 1)
        SAMP(1, 2) = .17132449237917
        SAMP(2, 2) = .360761573048139
        SAMP(3, 2) = .467913934572691
        SAMP(4, 2) = SAMP(3, 2)
        SAMP(5, 2) = SAMP(2, 2)
        SAMP(6, 2) = SAMP(1, 2)
        GOTO 700
600     SAMP(1, 1) = .949107912342759
        SAMP(2, 1) = .741531185599394
        SAMP(3, 1) = .405845151377397
        SAMP(4, 1) = 0
        SAMP(5, 1) = -SAMP(3, 1)
```

```
        SAMP(6, 1) = -SAMP(2, 1)
        SAMP(7, 1) = -SAMP(1, 1)
        SAMP(1, 2) = .12948496616887
        SAMP(2, 2) = .297705391489277
        SAMP(3, 2) = .381830050505199
        SAMP(4, 2) = .417959183673469
        SAMP(5, 2) = SAMP(3, 2)
        SAMP(6, 2) = SAMP(2, 2)
        SAMP(7, 2) = SAMP(1, 2)
700     REM
        REM *** NODE FREEDOM ARRAY FOR > ONE FREEDOM PER NODE ***
        REM
        FOR I = 1 TO NN
            FOR J = 1 TO NODOF
                NF(I, J) = 1
            NEXT J
        NEXT I
        FOR K = 1 TO RN
               INPUT #1, RNODE: REM Restrained Node Number
               FOR L = 1 TO NODOF
                   INPUT #1, RNT:   REM Restrained Node Type
                   IF RNT = 1 THEN NF(RNODE, L) = 0
               NEXT L
        NEXT K
        K = 1
        FOR M = 1 TO NN
            FOR P = 1 TO NODOF
                IF NF(M, P) = 0 THEN GOTO 1300
                NF(M, P) = K
                K = K + 1
1300        NEXT P
        NEXT M
      CLOSE #1
      REM
      REM *** PRINT DATA TO BIN FILE FOR NEXT PHASE ***
      REM
      OPEN "MATRIX.DAT" FOR OUTPUT AS #2
        FOR I = 1 TO 3
            FOR J = 1 TO 3
                PRINT #2, USING "##########.### "; DEE(I, J);
            NEXT J
            PRINT #2, " "
        NEXT I
        PRINT #2, GP
        FOR K = 1 TO GP
            PRINT #2, USING "##########.### "; SAMP(K, 1); SAMP(K, 2)
        NEXT K
        PRINT #2, NN
        FOR L = 1 TO NN
            FOR M = 1 TO NODOF
                PRINT #2, USING "##########.### "; NF(L, M);
            NEXT M
            PRINT #2, " "
        NEXT L
      CLOSE #2
      CHAIN "ELAST2.EXE"
```

```
ELAST2.EXE

CLEAR
DIM DEE(3, 3), SAMP(3, 2), NF(100, 2), COORD(20, 2), KM(20, 20)
DIM FUN(20), DER(20, 8), JAC(20, 2), JAC1(20, 2), DERIV(20, 4)
DIM VOL(20), DBEE(20, 20), BT(20, 3), BTDB(20, 20), G(10)
DIM BEE(20, 20), KB(200, 50)
REM
REM *** ELEMENT STIFFNESS INTEGRATION AND ASSEMBLY ***
REM
REM *** DATA INPUT ***
REM
OPEN "INPT.FLE" FOR INPUT AS #1
  INPUT #1, NXE, NYE, N, W, NN, RN, NL, GP, AA, BB, E, V
CLOSE #1
OPEN "MATRIX.DAT" FOR INPUT AS #1
  FOR I = 1 TO 3
      INPUT #1, DEE(I, 1), DEE(I, 2), DEE(I, 3)
  NEXT I
  INPUT #1, NGP
  FOR J = 1 TO NGP
      INPUT #1, SAMP(J, 1), SAMP(J, 2)
  NEXT J
  INPUT #1, NNF
  FOR K = 1 TO NNF
      INPUT #1, NF(K, 1), NF(K, 2)
  NEXT K
CLOSE #1
T = 2
H = 3
NODOF = 2: DOF = 8
NOD = DOF / NODOF
REM
REM *** NODAL COORDINATES + STEERING VECTOR FOR A RECTANGULAR ***
REM *** MESH OF 8-NODE QUADRILATERAL PLANE ELEMENTS NUMBERING ***
REM ***                  IN THE Y-DIRECTION                    ***
REM
FOR I = 1 TO NXE
    FOR J = 1 TO NYE
        AO = (I - 1) * (NYE + 1) + J
        AL = AO + 1
        AM = I * (NYE + 1) + J
        AN = AM + 1
        G(1) = NF(AL, 1)
        G(2) = NF(AL, 2)
        G(3) = NF(AO, 1)
        G(4) = NF(AO, 2)
        G(5) = NF(AM, 1)
        G(6) = NF(AM, 2)
        G(7) = NF(AN, 1)
        G(8) = NF(AN, 2)
        COORD(1, 1) = (I - 1) * AA
        COORD(1, 2) = (NYE - J) * BB
        COORD(2, 1) = (I - 1) * AA
        COORD(2, 2) = (NYE - J + 1) * BB
```

```
COORD(3, 1) = I * AA
COORD(3, 2) = (NYE - J + 1) * BB
COORD(4, 1) = I * AA
COORD(4, 2) = (NYE - J) * BB
REM
REM *** SET MATRIX TO 0 ***
REM
FOR K = 1 TO DOF
    FOR L = 1 TO DOF
        KM(K, L) = 0
    NEXT L
NEXT K
FOR II = 1 TO GP
    FOR JJ = 1 TO GP
        K1 = SAMP(II, 2)
        K2 = SAMP(JJ, 2)
        REM
        REM *** LOCAL COORDINATE SHAPE FUNCTIONS AND ***
        REM ***      THEIR DERIVATIVES FOR 8-NODE     ***
        REM ***              QUADRILATERAL            ***
        REM
        ETA = SAMP(II, 1)
        XI = SAMP(JJ, 1)
        ETAM = .25 * (1 - ETA)
        ETAP = .25 * (1 + ETA)
        XIM = .25 * (1 - XI)
        XIP = .25 * (1 + XI)
        FUN(1) = 4 * XIM * ETAM
        FUN(2) = 4 * XIM * ETAP
        FUN(3) = 4 * XIP * ETAP
        FUN(4) = 4 * XIP * ETAM
        DER(1, 1) = -ETAM
        DER(1, 2) = -ETAP
        DER(1, 3) = ETAP
        DER(1, 4) = ETAM
        DER(2, 1) = -XIM
        DER(2, 2) = XIM
        DER(2, 3) = XIP
        DER(2, 4) = -XIP
        REM
        REM *** MATRIX MULTIPLY ***
        REM
        FOR K = 1 TO T
            FOR L = 1 TO T
                X = 0
                FOR M = 1 TO NOD
                    X = X + DER(K, M) * COORD(M, L)
                NEXT M
                JAC(K, L) = X
            NEXT L
        NEXT K
        REM
        REM *** INVERT THE JACOBIAN MATRIX ***
        REM
        DET = JAC(1, 1) * JAC(2, 2) - JAC(1, 2) * JAC(2, 1)
        JAC1(1, 1) = JAC(2, 2)
```

```
JAC1(1, 2) = -JAC(1, 2)
JAC1(2, 1) = -JAC(2, 1)
JAC1(2, 2) = JAC(1, 1)
FOR K = 1 TO 2
    FOR L = 1 TO 2
        JAC1(K, L) = JAC1(K, L) / DET
    NEXT L
NEXT K
REM
REM *** MATRIX MULTIPLY ***
REM
FOR K = 1 TO T
    FOR L = 1 TO NOD
        X = 0
        FOR M = 1 TO T
            X = X + JAC1(K, M) * DER(M, L)
        NEXT M
        DERIV(K, L) = X
    NEXT L
NEXT K
REM
REM *** SET MATRIX TO 0 ***
REM
FOR K = 1 TO H
    FOR L = 1 TO DOF
        BEE(K, L) = 0
    NEXT L
NEXT K
REM
REM *** STRAIN DISPLACEMENT MATRIX FOR ***
REM ***      PLANE STRAIN  (STRESS)      ***
REM
FOR K = 1 TO NOD
    KK = 2 * K
    LL = KK - 1
    VOL(LL) = DERIV(1. K)
    BEE(1, LL) = VOL(LL)
    BEE(3, KK) = VOL(LL)
    VOL(KK) = DERIV(2, K)
    BEE(2, KK) = VOL(KK)
    BEE(3, LL) = VOL(KK)
NEXT K
REM
REM *** MATRIX MULTIPLY ***
REM
FOR K = 1 TO H
    FOR L = 1 TO DOF
        X = 0
        FOR M = 1 TO H
            X = X + DEE(K, M) * BEE(M, L)
        NEXT M
        DBEE(K, L) = X
    NEXT L
NEXT K
REM *** TRANSPOSE A MATRIX ***
REM
```

```
                    FOR K = 1 TO H
                        FOR L = 1 TO DOF
                            BT(L, K) = BEE(K, L)
                        NEXT L
                    NEXT K
                    REM
                    REM *** MATRIX MULTIPLY ***
                    REM
                    FOR K = 1 TO DOF
                        FOR L = 1 TO DOF
                            X = 0
                            FOR M = 1 TO H
                                X = X + BT(K, M) * DBEE(M, L)
                            NEXT M
                            BTDB(K, L) = X
                        NEXT L
                    NEXT K
                    QUOT = DET * K1 * K2
                    FOR K
                        FOR L = 1 TO DOF
                            BTDB(K, L) = BTDB(K, L) * QUOT
                        NEXT L
                    NEXT K
                    REM
                    REM *** ADD TWO MATRICES ***
                    REM
                    FOR K = 1 TO DOF
                        FOR L = 1 TO DOF
                            KM(K, L) = KM(K, L) + BTDB(K, L)
                        NEXT L
                    NEXT K
                NEXT JJ
            NEXT II
            REM
            REM *** ASSEMBLES ELEMENT MATRICES INTO GLOBAL MATRIX ***
            REM
            CDMAX = W + 1
            FOR K = 1 TO DOF
                IF G(K) = 0 THEN GOTO 200
                FOR L = 1 TO DOF
                    IF G(L) = 0 THEN GOTO 100
                    CD = G(L) - G(K) + CDMAX
                    IF CD > CDMAX THEN GOTO 100
                    KB(G(K), CD) = KB(G(K), CD) + KM(K, L)
100             NEXT L
200         NEXT K
        NEXT J
    NEXT I
    OPEN "CONSTIT.DAT" FOR OUTPUT AS #2
      FOR I = 1 TO N
        FOR J = 1 TO DOF
            PRINT #2, USING "################.##### "; KB(I, J);
        NEXT J
        PRINT #2, " "
      NEXT I
    CLOSE #2
```

```
      CHAIN "ELAST3.EXE"

      ELAST3.EXE

      REM
      REM  *** CHOLESKI REDUCTION OF A SYMETRICAL BAND ***
      REM
      DIM KB(200, 50), LOADS(200)
      DOF = 8
      OPEN "LOADS.DAT" FOR INPUT AS #1
        INPUT #1, NLOADS
        FOR I = 1 TO NLOADS
            INPUT #1, A, LOADS(A)
        NEXT I
      CLOSE #1
      OPEN "INPT.FLE" FOR INPUT AS #1
        INPUT #1, NXE, NYE, N, W
      CLOSE #1
      OPEN "CONSTIT.DAT" FOR INPUT AS #1
        FOR I = 1 TO N
            FOR J = 1 TO DOF
                INPUT #1, KB(I, J)
            NEXT J
        NEXT I
      CLOSE #1
      FOR I = 1 TO N
          X = 0
          FOR J = 1 TO W
              X = X + (KB(I, J) * KB(I, J))
          NEXT J
          KB(I, W + 1) = SQR(KB(I, W + 1) - X)
          FOR K = 1 TO W
              X = 0
              IF I + K > N THEN GOTO 300
              IF K = W THEN GOTO 200
              L = W - K
100           X = X + (KB(I + K, L) * KB(I, L + K))
              L = L - 1
              IF L = 0 THEN GOTO 200
              GOTO 100
200           A = I + K
              B = (W - K) + 1
              KB(A, B) = (KB(A, B) - X) / KB(I, W + 1)
300       NEXT K
      NEXT I
      REM
      REM *** CHOLESKI FORWARD SUBSTITUTION ***
      REM
      LOADS(1) = LOADS(1) / KB(1, W + 1)
      FOR I = 2 TO N
          X = 0
          K = 1
          IF I <= W + 1 THEN K = (W - I) + 2
          FOR J = K TO W
              XXX = ((I + J) - W) - 1
              X = X + KB(I, J) * LOADS(XXX)
```

```
        NEXT J
        LOADS(I) = (LOADS(I) - X) / KB(I, W + 1)
    NEXT I
    REM
    REM *** CHOLESKI BACKWARD SUBSTITUTION ***
    REM
    LOADS(N) = LOADS(N) / KB(N, W + 1)
    I = N - 1
400 X = 0
    L = I + W
    IF I > N - W THEN L = N
    M = I + 1
    FOR J = M TO L
        XXX = ((W + I) - J) + 1
        X = X + KB(J, XXX) * LOADS(J)
    NEXT J
    LOADS(I) = (LOADS(I) - X) / KB(I, W + 1)
    I = I - 1
    IF I = 0 THEN GOTO 500
    GOTO 400
500 OPEN "DISPL.DAT" FOR OUTPUT AS #2
      FOR I = 1 TO N
          PRINT #2, USING "###############.##### "; LOADS(I)
      NEXT I
    CLOSE #2
    CHAIN "ELAST4.EXE"

    ELAST4.EXE

    REM
    REM *** RECOVER ELEMENT STRAINS AND STRESSES ***
    REM ***  AT ALL GAUSSIAN INTEGRATION POINTS  ***
    REM
    DIM NF(100, 2), DEE(3, 3), SAMP(3, 2), COORD(20, 2), FUN(4)
    DIM JAC(20, 2), JAC1(20, 2), ELD(100), LOADS(100), DERIV(20, 4)
    DIM VOL(20), DBEE(20, 20), G(10), SIGMA(3), EPS(3), DER(20, 8)
    DIM BEE(20, 20)
    REM *** NODAL COORDINATES AND STEERING VECTOR FOR A RECTANGULAR ***
    REM ***  MESH OF 4-NODE QUADRILATERAL PLANE ELEMENTS NUMBERING  ***
    REM ***                  IN THE Y-DIRECTION                     ***
    REM
    OPEN "INPT.FLE" FOR INPUT AS #1
      INPUT #1, NXE, NYE, N, W, NN, RN, NL, GP, AA, BB, E, V
    CLOSE #1
    OPEN "MATRIX.DAT" FOR INPUT AS #1
      FOR I = 1 TO 3
          INPUT #1, DEE(I, 1), DEE(I, 2), DEE(I, 3)
      NEXT I
      INPUT #1, NGP
      FOR I = 1 TO NGP
          INPUT #1, SAMP(I, 1), SAMP(I, 2)
      NEXT I
      INPUT #1, NNF
      FOR I = 1 TO NNF
          INPUT #1, NF(I, 1), NF(I, 2)
      NEXT I
```

```
REM
FOR K = 1 TO T
    FOR L = 1 TO T
        X = 0
        FOR M = 1 TO NOD
            X = X + DER(K, M) * COORD(M, L)
        NEXT M
        JAC(K, L) = X
    NEXT L
NEXT K
REM
REM *** INVERT THE JACOBIAN MATRIX ***
REM
DET = (JAC(1, 1) * JAC(2, 2)) - (JAC(1, 2)
* JAC1(2, 1))
JAC1(1, 1) = JAC(2, 2)
JAC1(1, 2) = -JAC(1, 2)
JAC1(2, 1) = -JAC(2, 1)
JAC1(2, 2) = JAC(1, 1)
FOR K = 1 TO 2
    FOR L = 1 TO 2
        JAC1(K, L) = JAC1(K, L) / DET
    NEXT L
NEXT K
REM
REM *** MATRIX MULTIPLY ***
REM
FOR K = 1 TO T
    FOR L = 1 TO NOD
        X = 0
        FOR M = 1 TO T
            X = X + JAC1(K, M) * DER(M, L)
        NEXT M
        DERIV(K, L) = X
    NEXT L
NEXT K
REM
REM *** SET MATRIX TO 0 ***
REM
FOR K = 1 TO H
    FOR L = 1 TO DOF
        BEE(K, L) = 0
    NEXT L
NEXT K
REM
REM *** STRAIN DISPLACEMENT MATRIX FOR ***
REM ***       PLANE STRAIN (STRESS)     ***
REM
FOR K = 1 TO NOD
    KK = 2 * K
    LL = KK - 1
    VOL(LL) = DERIV(1, K)
    BEE(1, LL) = VOL(LL)
    BEE(3, KK) = VOL(LL)
    VOL(KK) = DERIV(2, K)
    BEE(2, KK) = VOL(KK)
```

```
                        BEE(3, LL) = VOL(KK)
                    NEXT K
                    FOR K = 1 TO DOF
                        IF G(K) = 0 THEN ELD(K) = 0
                        IF G(K) <> 0 THEN ELD(K) = LOADS(G(K))
                    NEXT K
                    FOR L = 1 TO H
                        X = 0
                        FOR M = 1 TO DOF
                            X = X + BEE(L, M) * ELD(M)
                        NEXT M
                        EPS(L) = X
                    NEXT L
                    FOR L = 1 TO H
                        X = 0
                        FOR M = 1 TO H
                            X = X + DEE(L, M) * EPS(M)
                        NEXT M
                        SIGMA(L) = X
                    NEXT L
                    FOR K = 1 TO H
                        PRINT #2, USING "##############.##### ";
                        EPS(K);
                    NEXT K
                    PRINT #2, ""
                    FOR K = 1 TO H
                        PRINT #2, USING "##############.##### ";
                        SIGMA(K);
                    NEXT K
                    PRINT #2, ""
                NEXT JJ
            NEXT II
        NEXT J
    NEXT I
  CLOSE #2
CLOSE #1
CHAIN "SUBFIN.EXE"
```

```
2.0 VON MISE'S ANALYSIS PROGRAM

VONMSE1.EXE

CLEAR
REM
REM *** AXISYMMETRIC STRAIN OF A RECTANGULAR ELASTO-PLASTIC ***
REM ***     (VON MISES'S) SOLID USING 4-NODE QUADRILATERAL    ***
REM ***             ELEMENTS--INITIAL STRESS METHOD           ***
REM
DIM DEE(4, 4), SAMP(7, 2), NF(100, 2), RADIUS(25)
REM
REM ***                INITIALZE DATA                ***
REM
NODOF = 2:      REM Number of freedoms per node
DOF = 8:        REM Degrees of freedom per element
OPEN "INPT.FLE" FOR INPUT AS #1
   INPUT #1, NXE: REM number of elements in x direction
   INPUT #1, NYE: REM number of elements in y direction
   INPUT #1, N:   REM total number of freedoms in mesh
   INPUT #1, W:   REM half bandwidth
   INPUT #1, NN:  REM number of nodes in mesh
   INPUT #1, RN:  REM restrained nodes in mesh
   INPUT #1, NL:  REM number of nodes where solution non zero
   INPUT #1, GP:  REM Gaussian integration order
   INPUT #1, BB:  REM element size in y direction
   INPUT #1, V:   REM Poisson's Ratio
   INPUT #1, E:   REM Young's Modulus
   INPUT #1, SBARY: REM Soil cohesion
   INPUT #1, INCS: REM Number of load increments
   INPUT #1, ITS:  REM Total number of iterations
   PI = 4 * ATN(1)
   NOD = DOF / NODOF
   CDMAX = W + 1
   R = N * CDMAX
   REM
   REM *** STRESS STRAIN MATRIX FOR ELASTIC AXISYMMETRYN ***
   REM
   V1 = V / (1 - V)
   VV = (1 - 2 * V) / (1 - V) * .5
   DEE(1, 1) = 1
   DEE(2, 2) = 1
   DEE(4, 4) = 1
   DEE(3, 3) = VV
   DEE(1, 2) = V1
   DEE(2, 1) = V1
   DEE(1, 4) = V1
   DEE(4, 1) = V1
   DEE(2, 4) = V1
   DEE(4, 2) = V1
   FOR I = 1 TO 4
       FOR J = 1 TO 4
           DEE(I, J) = DEE(I, J) * E * (1 - V) / (1 - 2 * V)
           / (1 + V)
       NEXT J
```

```
        NEXT I
        REM
        REM *** GAUSSIAN QUADRATURE ABSCISSAE AND WEIGHTS ***
        REM
        GP = INT(GP)
        IF GP = 1 THEN GOTO 700
        IF GP = 2 THEN GOTO 100
        IF GP = 3 THEN GOTO 200
        IF GP = 4 THEN GOTO 300
        IF GP = 5 THEN GOTO 400
        IF GP = 6 THEN GOTO 500
        IF GP = 7 THEN GOTO 600
        GP = 2
100     SAMP(1, 1) = 1 / SQR(3)
        SAMP(2, 1) = -SAMP(1, 1)
600     SAMP(1, 2) = 1
        SAMP(2, 2) = 1
        GOTO 700
200     SAMP(1, 1) = .2 * SQR(15)
        SAMP(2, 1) = 0
        SAMP(3, 1) = -SAMP(1, 1)
        SAMP(1, 2) = 5 / 9
        SAMP(2, 2) = 8 / 9
        SAMP(3, 2) = SAMP(1, 2)
        GOTO 700
300     SAMP(1, 1) = .861136311594053
        SAMP(2, 1) = .339981043584856
        SAMP(3, 1) = -SAMP(2, 1)
        SAMP(4, 1) = -SAMP(1, 1)
        SAMP(1, 2) = .3847854845137454
        SAMP(2, 2) = .652145154862546
        SAMP(3, 2) = SAMP(2, 2)
        SAMP(4, 2) = SAMP(1, 2)
        GOTO 700
400     SAMP(1, 1) = .906179845838664
        SAMP(2, 1) = .538469310105683
        SAMP(3, 1) = 0
        SAMP(4, 1) = -SAMP(2, 1)
        SAMP(5, 1) = -SAMP(1, 1)
        SAMP(1, 2) = .236926885056189
        SAMP(2, 2) = .478628670499366
        SAMP(3, 2) = .568888888888889
        SAMP(4, 2) = SAMP(2, 2)
        SAMP(5, 2) = SAMP(1, 2)
        GOTO 700
        SAMP(1, 1) = .9324695142031521
        SAMP(2, 1) = .661209386466265
        SAMP(3, 1) = .238619186083197
        SAMP(4, 1) = -SAMP(3, 1)
        SAMP(5, 1) = -SAMP(2, 1)
        SAMP(6, 1) = -SAMP(3, 1)
        SAMP(1, 2) = .17132449237917
        SAMP(2, 2) = .360761573048139
        SAMP(3, 2) = .467913934572691
        SAMP(4, 2) = SAMP(3, 2)
        SAMP(5, 2) = SAMP(2, 2)
```

```
        SAMP(6, 2) = SAMP(1, 2)
        GOTO 700
500     SAMP(1, 1) = .949107912342759
        SAMP(2, 1) = .741531185599394
        SAMP(3, 1) = .405845151377397
        SAMP(4, 1) = 0
        SAMP(5, 1) = -SAMP(3, 1)
        SAMP(6, 1) = -SAMP(2, 1)
        SAMP(7, 1) = -SAMP(1, 1)
        SAMP(1, 2) = .12948496616887
        SAMP(2, 2) = .297705391489277
        SAMP(3, 2) = .381830050505199
        SAMP(4, 2) = .417959183673469
        SAMP(5, 2) = SAMP(3, 2)
        SAMP(6, 2) = SAMP(2, 2)
        SAMP(7, 2) = SAMP(1, 2)
700     REM
        REM *** NODE FREEDOM ARRAY FOR MORE THAN ONE FREEDOM/NODE ***
        REM
        FOR I = 1 TO NN
            FOR J = 1 TO NODOF
                NF(I, J) = 1
            NEXT J
        NEXT I
        FOR K = 1 TO RN
            INPUT #1, RNODE: REM Restrained Node Number
            FOR L = 1 TO NODOF
                INPUT #1, RNT:   REM Restrained Node Type
                IF RNT = 1 THEN NF(RNODE, L) = 0
            NEXT L
        NEXT K
        K = 1
        FOR M = 1 TO NN
            FOR P = 1 TO NODOF
                IF NF(M, P) = 0 THEN GOTO 800
                NF(M, P) = K
                K = K + 1
800         NEXT P
        NEXT M
    CLOSE #1
    REM
    REM *** PRINT DATA TO BIN FILE FOR NEXT PHASE ***
    REM
    OPEN "MATRIX.DAT" FOR OUTPUT AS #2
    FOR I = 1 TO 4
        FOR J = 1 TO 4
            PRINT #2, USING "############.### "; DEE(I, J);
        NEXT J
        PRINT #2, " "
    NEXT I
    PRINT #2, GP
    FOR K = 1 TO GP
        PRINT #2, USING "############.### "; SAMP(K, 1); SAMP(K, 2)
    NEXT K
    PRINT #2, NN
    FOR L = 1 TO NN
```

```
      FOR M = 1 TO NODOF
          PRINT #2, USING "####.### "; NF(L, M);
      NEXT M
      PRINT #2, " "
NEXT L
CLOSE #2
CHAIN "VONMSE2.EXE"

VONMSE2.EXE

CLEAR
DIM DEE(4, 4), SAMP(7, 2), NF(100, 2), COORD(20, 2), KM(20, 20)
DIM FUN(20), DER(20, 8), JAC(20, 2), JAC1(20, 2), DERIV(20, 4)
DIM VOL(20), DBEE(20, 20), BT(20, 4), BTDB(20, 20), G(10)
DIM BEE(20, 20), KB(200, 50)
REM
REM *** ELEMENT STIFFNESS INTEGRATION AND ASSEMBLY ***
REM
REM *** DATA INPUT ***
REM
OPEN "INPT.FLE" FOR INPUT AS #1
   INPUT #1, NXE, NYE, N, W, NN, RN, NL, GP, BB, V, E, SBARY, INCS,
   FOR I = 1 TO RN
       INPUT #1, DUMMY1$, DUMMY2$, DUMMY3$
   NEXT I
CLOSE #1
OPEN "MATRIX.DAT" FOR INPUT AS #1
   FOR I = 1 TO 4
       INPUT #1, DEE(I, 1), DEE(I, 2), DEE(I, 3), DEE(I, 4)
   NEXT I
   INPUT #1, NGP
   FOR J = 1 TO NGP
       INPUT #1, SAMP(J, 1), SAMP(J, 2)
   NEXT J
   INPUT #1, NNF
   FOR K = 1 TO NNF
       INPUT #1, NF(K, 1), NF(K, 2)
   NEXT K
CLOSE #1
T = 2
H = 4
NODOF = 2: DOF = 8
NOD = DOF / NODOF
PI = 4 * ATN(1)
REM
REM *** NODAL COORDINATES AND STEERING VECTOR FOR RECTANGULAR ***
REM *** MESH OF 8-NODE QUADRILATERAL PLANE ELEMENTS NUMBERING ***
REM ***                  IN THE Y-DIRECTION                    ***
REM
FOR I = 1 TO NXE
    FOR J = 1 TO NYE
        AO = (I - 1) * (NYE + 1) + J
        AL = AO + 1
        AM = I * (NYE + 1) + J
        AN = AM + 1
        G(1) = NF(AL, 1)
```

```
G(2) = NF(AL, 2)
G(3) = NF(AO, 1)
G(4) = NF(AO, 2)
G(5) = NF(AM, 1)
G(6) = NF(AM, 2)
G(7) = NF(AN, 1)
G(8) = NF(AN, 2)
COORD(1, 1) = (I - 1) * AA
COORD(1, 2) = (NYE - J) * BB
COORD(2, 1) = (I - 1) * AA
COORD(2, 2) = (NYE - J + 1) * BB
COORD(3, 1) = I * AA
COORD(3, 2) = (NYE - J + 1) * BB
COORD(4, 1) = I * AA
COORD(4, 2) = (NYE - J) * BB
REM
REM *** SET MATRIX TO 0 ***
REM
FOR K = 1 TO DOF
    FOR L = 1 TO DOF
        KM(K, L) = 0
    NEXT L
NEXT K
FOR II = 1 TO GP
    FOR JJ = 1 TO GP
        K1 = SAMP(II, 2)
        K2 = SAMP(JJ, 2)
        REM
        REM *** LOCAL COORDINATE SHAPE FUNCTIONS AND ***
        REM ***     THEIR DERIVATIVES FOR 8-NODE     ***
        REM ***             QUADRILATERAL            ***
        REM
        ETA = SAMP(II, 1)
        XI = SAMP(JJ, 1)
        ETAM = .25 * (1 - ETA)
        ETAP = .25 * (1 + ETA)
        XIM = .25 * (1 - XI)
        XIP = .25 * (1 + XI)
        FUN(1) = 4 * XIM * ETAM
        FUN(2) = 4 * XIM * ETAP
        FUN(3) = 4 * XIP * ETAP
        FUN(4) = 4 * XIP * ETAM
        DER(1, 1) = -ETAM
        DER(1, 2) = -ETAP
        DER(1, 3) = ETAP
        DER(1, 4) = ETAM
        DER(2, 1) = -XIM
        DER(2, 2) = XIM
        DER(2, 3) = XIP
        DER(2, 4) = -XIP
        REM
        REM *** MATRIX MULTIPLY ***
        REM
        FOR K = 1 TO T
            FOR L = 1 TO T
                X = 0
```

```
            FOR M = 1 TO NOD
                X = X + DER(K, M) * COORD(M, L)
            NEXT M
            JAC(K, L) = X
        NEXT L
    NEXT K
    REM
    REM *** INVERT THE JACOBIAN MATRIX ***
    REM
    DET = JAC(1, 1) * JAC(2, 2) - JAC(1, 2)
    * JAC(2, 1)
    JAC1(1, 1) = JAC(2, 2)
    JAC1(1, 2) = -JAC(1, 2)
    JAC1(2, 1) = -JAC(2, 1)
    JAC1(2, 2) = JAC(1, 1)
    FOR K = 1 TO 2
        FOR L = 1 TO 2
            JAC1(K, L) = JAC1(K, L) / DET
        NEXT L
    NEXT K
    REM
    REM *** MATRIX MULTIPLY ***
    REM
    FOR K = 1 TO T
        FOR L = 1 TO NOD
            X = 0
            FOR M = 1 TO T
                X = X + JAC1(K, M) * DER(M, L)
            NEXT M
            DERIV(K, L) = X
        NEXT L
    NEXT K
    REM
    REM *** SET MATRIX TO 0 ***
    REM
    FOR K = 1 TO H
        FOR L = 1 TO DOF
            BEE(K, L) = 0
        NEXT L
    NEXT K
    REM
    REM *** STRAIN DISPLACEMENT MATRIX FOR ***
    REM ***       AXISYMMETRIC STRAIN       ***
    REM
    SUM = 0
    FOR K = 1 TO NOD
        SUM = SUM + FUN(K) * COORD(K, 1)
    NEXT K
    FOR M = 1 TO NOD
        K = 2 * M
        L = K - 1
        BEE(1, L) = DERIV(1, M)
        BEE(3, K) = BEE(1, L)
        VOL(K) = DERIV(2, M)
        BEE(2, K) = VOL(K)
        BEE(3, L) = VOL(K)
```

```
                BEE(4, L) = FUN(M) / SUM
                VOL(L) = BEE(1, L) + BEE(4, L)
            NEXT M
            REM
            REM *** MATRIX MULTIPLY ***
            REM
            FOR K = 1 TO H
                FOR L = 1 TO DOF
                    X = 0
                    FOR M = 1 TO H
                        X = X + DEE(K, M) * BEE(M, L)
                    NEXT M
                    DBEE(K, L) = X
                NEXT L
            NEXT K
            REM
            REM *** TRANSPOSE A MATRIX ***
            REM
            FOR K = 1 TO H
                FOR L = 1 TO DOF
                    BT(L, K) = BEE(K, L)
                NEXT L
            NEXT K
            REM
            REM *** MATRIX MULTIPLY ***
            REM
            FOR K = 1 TO DOF
                FOR L = 1 TO DOF
                    X = 0
                    FOR M = 1 TO H
                        X = X + BT(K, M) * DBEE(M, L)
                    NEXT M
                    BTDB(K, L) = X
                NEXT L
            NEXT K
            QUOT = DET * K1 * K2 * 2 * PI * SUM
            FOR K = 1 TO DOF
                FOR L = 1 TO DOF
                    BTDB(K, L) = BTDB(K, L) * QUOT
                NEXT L
            NEXT K
            REM
            REM *** ADD TWO MATRICES ***
            REM
            FOR K = 1 TO DOF
                FOR L = 1 TO DOF
                    KM(K, L) = KM(K, L) + BTDB(K, L)
                NEXT L
            NEXT K
        NEXT JJ
    NEXT II
    REM
    REM *** ASSEMBLES ELEMENT MATRICES INTO GLOBAL MATRIX ***
    REM
    CDMAX = W + 1
    FOR K = 1 TO DOF
```

```
                IF G(K) = 0 THEN GOTO 200
                FOR L = 1 TO DOF
                    IF G(L) = 0 THEN GOTO 100
                    CD = G(L) - G(K) + CDMAX
                    IF CD > CDMAX THEN GOTO 100
                    KB(G(K), CD) = KB(G(K), CD) + KM(K, L)
100             NEXT L
200         NEXT K
        NEXT J
    NEXT I
    REM
    REM  *** CHOLESKI REDUCTION OF A SYMETRICAL BAND ***
    REM
    FOR I = 1 TO N
        X = 0
        FOR J = 1 TO W
            X = X + (KB(I, J) * KB(I, J))
        NEXT J
        KB(I, W + 1) = SQR(KB(I, W + 1) - X)
        FOR K = 1 TO W
            X = 0
            IF I + K > N THEN GOTO 500
            IF K = W THEN GOTO 400
            L = W - K
300         X = X + (KB(I + K, L) * KB(I, L + K))
            L = L - 1
            IF L = 0 THEN GOTO 400
            GOTO 300
400         A = I + K
            B = (W - K) + 1
            KB(A, B) = (KB(A, B) - X) / KB(I, W + 1)
500     NEXT K
    NEXT I
    OPEN "CONSTIT.DAT" FOR OUTPUT AS #2
       FOR I = 1 TO N
           FOR J = 1 TO DOF
               PRINT #2, USING "##############.##### "; KB(I, J);
           NEXT J
           PRINT #2, " "
       NEXT I
    CLOSE #2
    CHAIN "VONMSE3.EXE"

VONMSE3.EXE

CLEAR
REM
REM  *** INCREMENT THE LOADS ***
REM
DIM DEE(8, 4), DPL(8, 4), PL(8, 4), SAMP(3, 2), COORD(8, 2)
DIM DERIV(8, 4), BEE(8, 8), ELD(8), VOL(8), EPS(4), SIGMA(4)
DIM DF(8), ELOAD(8), NVL(20), STORKB(20), NO(20), G(8), KB(16, 8)
DIM SR(20, 20), ST(20, 20), SZ(20, 20), TRZ(20, 20), ER(20, 20)
DIM JAC(8, 2), JAC1(8, 2), DER(8, 4), FUN(4), ELSO(4), SPL(4)
DIM LOADS(20), BDYLDS(20), OLDLDS(20), ET(20, 20), EZ(20, 20)
DIM GRZ(20, 20), NF(20, 2)
```

```
DOF = 8
OPEN "INPT.FLE" FOR INPUT AS #1
   INPUT #1, NXE, NYE, N, W, NN, RN, NL
   INPUT #1, GP, BB, V, E, SBARY, INCS, ITS
CLOSE #1
DOF = 8
NODOF = 2
NOD = DOF / NODOF
T = 2
H = 4
OPEN "CONSTIT.DAT" FOR INPUT AS #1
   FOR I = 1 TO N
     FOR J = 1 TO DOF
         INPUT #1, KB(I, J)
     NEXT J
   NEXT I
CLOSE #1
OPEN "MATRIX.DAT" FOR INPUT AS #1
   FOR I = 1 TO 4
       INPUT #1, DEE(I, 1), DEE(I, 2), DEE(I, 3), DEE(I, 4)
   NEXT I
   INPUT #1, NGP
   FOR J = 1 TO NGP
       INPUT #1, SAMP(J, 1), SAMP(J, 2)
   NEXT J
   INPUT #1, NNF
   FOR K = 1 TO NNF
       INPUT #1, NF(K, 1), NF(K, 2)
   NEXT K
CLOSE #1
OPEN "LOADS.DAT" FOR INPUT AS #1
   INPUT #1, NLDS
   FOR I = 1 TO NLDS
       INPUT #1, NO(I), VVL(I)
   NEXT I
CLOSE #1
PI = 4 * ATN(1)
OPEN "STRAINS.DAT" FOR OUTPUT AS #3
   REM
   REM *** INCREMENT THE LOADS ***
   REM
   FOR Y = 1 TO INCS
       REM
       REM *** ITERATE TO REDISTRIBUTE EXCESS ELEMENT STRESSES ***
       REM
       FOR Z = 1 TO ITS
           FOR I = 1 TO N
               LOADS(I) = 0
           NEXT I
           FOR I = 1 TO NL
               LOADS(NO(I)) = VVL(I) * PI
           NEXT I
           FOR I = 1 TO N
               LOADS(I) = LOADS(I) + BDYLDS(I)
           NEXT I
           REM
```

```
                REM *** CHOLESKI FORWARD SUBSTITUTION ***
                REM
                LOADS(1) = LOADS(1) / KB(1, W + 1)
                FOR I = 2 TO N
                    X = 0
                    K = 1
                   IF I <= W + 1 THEN K = (J - I) + 2
                   FOR J = K TO W
                       XXX = ((I + J) - W) - 1
                       X = X + KB(I, J) * LOADS(XXX)
                   NEXT J
                   LOADS(I) = (LOADS(I) - X) / KB(I, W + 1)
                NEXT I
                REM
                REM *** CHOLESKI BACKWARD SUBSTITUTION ***
                REM
                LOADS(N) = LOADS(N) / KB(N, W + 1)
                I = N - 1
100             X = 0
                L = I + W
                IF I > N - W THEN L = N
                M = I + 1
                FOR J = M TO L
                    XXX = ((W + I) - J) + 1
                    X = X + KB(J, XXX) * LOADS(J)
                NEXT J
                LOADS(I) = (LOADS(I) - X) / KB(I, W + 1)
                I = I - 1
                IF I = 0 THEN GOTO 200
                GOTO 100
                REM
                REM *** SET VECTOR TO 0 ***
                REM
200             FOR I = 1 TO N
                    BDYLDS(I) = 0
                NEXT I
                REM
                REM *** INSPECT ALL ELEMENTS ***
                REM
                FOR I = 1 TO NXE
                    FOR J = 1 TO NYE
                        AO = (I - 1) * (NYE + 1) + J
                        AL = AO + 1
                        AM = I * (NYE + 1) + J
                        AN = AM + 1
                        G(1) = NF(AL, 1)
                        G(2) = NF(AL, 2)
                        G(3) = NF(AO, 1)
                        G(4) = NF(AO, 2)
                        G(5) = NF(AM, 1)
                        G(6) = NF(AM, 2)
                        G(7) = NF(AN, 1)
                        G(8) = NF(AN, 2)
                        COORD(1, 1) = (I - 1) * AA
                        COORD(1, 2) = (NYE - J) * BB
                        COORD(2, 1) = (I - 1) * AA
```

```
COORD(2, 2) = (NYE - J + 1) * BB
COORD(3, 1) = I * AA
COORD(3, 2) = (NYE - J + 1) * BB
COORD(4, 1) = I * AA
COORD(4, 2) = (NYE - J) * BB
REM
REM ***STRAINS AT ELEMENT 'CENTRES' ***
REM
II = 2
JJ = 2
SAMP(2, 1) = 0
REM
REM *** LOCAL COORDINATE SHAPE FUNCTIONS ***
REM *** AND THEIR DERIVATIVES FOR 8-NODE ***
REM ***           QUADRILATERAL          ***
REM
ETA = SAMP(II, 1)
XI = SAMP(JJ, 1)
ETAM = .25 * (1 - ETA)
ETAP = .25 * (1 + ETA)
XIM = .25 * (1 - XI)
XIP = .25 * (1 + XI)
FUN(1) = 4 * XIM * ETAM
FUN(2) = 4 * XIM * ETAP
FUN(3) = 4 * XIP * ETAP
FUN(4) = 4 * XIP * ETAM
DER(1, 1) = -ETAM
DER(1, 2) = -ETAP
DER(1, 3) = ETAP
DER(1, 4) = ETAM
DER(2, 1) = -XIM
DER(2, 2) = XIM
DER(2, 3) = XIP
DER(2, 4) = -XIP
REM
REM *** MATRIX MULTIPLY ***
REM
FOR K = 1 TO T
    FOR L = 1 TO T
        X = 0
        FOR M = 1 TO NOD
            X = X + DER(K, M) * COORD(M, L)
        NEXT M
        JAC(K, L) = X
    NEXT L
NEXT K
REM
REM *** INVERT THE JACOBIAN MATRIX ***
REM
DET = JAC(1, 1) * JAC(2, 2) - JAC(1, 2)
* JAC(2, 1)
JAC1(1, 1) = JAC(2, 2)
JAC1(1, 2) = -JAC(1, 2)
JAC1(2, 1) = -JAC(2, 1)
JAC1(2, 2) = JAC(1, 1)
FOR K = 1 TO 2
```

```
        FOR L = 1 TO 2
                JAC1(K, L) = JAC1(K, L) / DET
        NEXT L
    NEXT K
    REM
    REM *** MATRIX MULTIPLY ***
    REM
    FOR K = 1 TO T
        FOR L = 1 TO NOD
            X = 0
            FOR M = 1 TO T
                X = X + JAC1(K, M) * DER(M, L)
            NEXT M
            DERIV(K, L) = X
        NEXT L
    NEXT K
    REM
    REM *** SET MATRIX TO 0 ***
    REM
    FOR K = 1 TO H
        FOR L = 1 TO DOF
            BEE(K, L) = 0
        NEXT L
    NEXT K
    REM
    REM *** STRAIN DISPLACEMENT MATRIX FOR ***
    REM ***       AXISYMMETRIC STRAIN       ***
    REM
    SUM = 0
    FOR K = 1 TO NOD
        SUM = SUM + FUN(K) * COORD(K, 1)
    NEXT K
    FOR M = 1 TO NOD
        K = 2 * M
        L = K - 1
        BEE(1, L) = DERIV(1, M)
        BEE(3, K) = BEE(1, L)
        VOL(K) = DERIV(2, M)
        BEE(2, K) = VOL(K)
        BEE(3, L) = VOL(K)
        BEE(4, L) = FUN(M) / SUM
        VOL(L) = BEE(1, L) + BEE(4, L)
    NEXT M
    FOR M = 1 TO DOF
    IF G(M) = 0 THEN ELD(M) = 0
    IF G(M) <> 0 THEN ELD(M) = LOADS(G(M))
    NEXT M
    REM
    REM *** MULTIPLY A MATRIX BY A VECTOR ***
    REM
    FOR M = 1 TO H
        X = 0
        FOR MM = 1 TO DOF
            X = X + BEE(M, MM) * ELD(MM)
        NEXT MM
        EPS(M) = X
```

```
NEXT M
REM
REM *** ELASTIC STRESSES ARE SIGMA ***
REM
REM
REM *** MULTIPLY A MATRIX BY A VECTOR ***
REM
FOR M = 1 TO H
    X = 0
    FOR MM = 1 TO H
        X = X + DEE(M, MM) * EPS(MM)
    NEXT MM
    SIGMA(M) = X
NEXT M
REM
REM *** SET VECTOR TO 0 ***
REM
FOR M = 1 TO H
    ELSO(M) = 0
NEXT M
REM
REM *** SET VECTOR TO 0 ***
REM
FOR M = 1 TO DOF
    BLD(M) = 0
NEXT M
S1 = SIGMA(1) + SR(I, J)
S2 = SIGMA(2) + ST(I, J)
S3 = SIGMA(2) + SZ(I, J)
S4 = SIGMA(3) + TRZ(I, J)
DSBAR = SQR(.5 * ((S1 - S2) * (S1 - S2)
+ (S2 - S3) * (S2 - S3) + (S3 - S1) * (S3 - S1)
+ 6 * S4 * S4))
IF DSBAR < SBARY THEN GOTO 400
S1 = SR(I, J)
S2 = ST(I, J)
S3 = SZ(I, J)
S4 = TRZ(I, J)
REM
REM *** STRESS-STRAIN MATRIX FOR VON MISES ***
REM ***              PLASTICITY              ***
REM
SP = (S1 + S2 + S3) / 3
SR1 = S1 - SP
ST1 = S2 - SP
SZ1 = S3 - SP
SBAR = SQR(.5 * ((S1 - S2) * (S1 - S2)
+ (S2 - S3) * (S2 - S3) + (S3 - S1) * (S3 - S1)
+ 6 * S4 * S4))
IF SBAR > SBARY THEN FAC = 1
IF SBAR <= SBARY THEN FAC = (DSBAR - SBARY)
/(DSBAR - SBAR)
PL(1, 1) = SR1 * SR1
PL(1, 2) = SR1 * SZ1
PL(2, 1) = PL(1, 2)
PL(1, 3) = SR1 * S4
```

```
PL(3, 1) = PL(1, 3)
PL(1, 4) = SR1 * ST1
PL(4, 1) = PL(1, 4)
PL(2, 2) = SZ1 * SZ1
PL(2, 3) = SZ1 * S4
PL(3, 2) = PL(2, 3)
PL(2, 4) = SZ1 * ST1
PL(4, 2) = PL(2, 4)
PL(3, 3) = S4 * S4
PL(3, 4) = ST1 * S4
PL(4, 3) = PL(3, 4)
PL(4, 4) = ST1 * ST1
FOR K = 1 TO 4
    FOR L = 1 TO 4
        PL(K, L) = PL(K, L) * 3 * E / (2 * SBAR
        * SBAR * (1 + V))
        DPL(K, L) = DEE(K, L) - FAC * PL(K, L)
    NEXT L
NEXT K
REM
REM *** ELASTO-PLASTIC STRESSES ARE SPL ***
REM
REM
REM *** MULTIPLY A MATRIX BY A VECTOR ***
REM
FOR K = 1 TO H
    X = 0
    FOR L = 1 TO H
        X = X + DPL(K, L) * EPS(L)
    NEXT L
    SPL(K) = X
NEXT K
FOR M = 1 TO H
    ELSO(M) = SIGMA(M) - SPL(M)
NEXT M
REM
REM *** INTEGRATE ALL EXCESS STRESSES TO ***
REM ***          FIND NODAL LOADS          ***
REM
FOR II = 1 TO GP
    FOR JJ = 1 TO GP
        K1 = SAMP(II, 2)
        K2 = SAMP(JJ, 2)
        REM
        REM *** LOCAL COORDINATE SHAPE ***
        REM ***   FUNCTIONS AND THEIR  ***
        REM *** DERIVATIVES FOR 8-NODE ***
        REM ***     QUADRILATERAL      ***
        REM
        ETA = SAMP(II, 1)
        XI = SAMP(JJ, 1)
        ETAM = .25 * (1 - ETA)
        ETAP = .25 * (1 + ETA)
        XIM = .25 * (1 - XI)
        XIP = .25 * (1 + XI)
        FUN(1) = 4 * XIM * ETAM
```

```
FUN(2) = 4 * XIM * ETAP
FUN(3) = 4 * XIP * ETAP
FUN(4) = 4 * XIP * ETAM
DER(1, 1) = -ETAM
DER(1, 2) = -ETAP
DER(1, 3) = ETAP
DER(1, 4) = ETAM
DER(2, 1) = -XIM
DER(2, 2) = XIM
DER(2, 3) = XIP
DER(2, 4) = -XIP
REM
REM *** MATRIX MULTIPLY ***
REM
FOR K = 1 TO T
    FOR L = 1 TO T
        X = 0
        FOR M = 1 TO NOD
            X = X + DER(K, M)
            * COORD(M, L)
        NEXT M
        JAC(K, L) = X
    NEXT L
NEXT K
REM
REM *** INVERT THE JACOBIAN MATRIX ***
REM
DET = JAC(1, 1) * JAC(2, 2) - JAC(1, 2)
* JAC(2, 1)
JAC1(1, 1) = JAC(2, 2)
JAC1(1, 2) = -JAC(1, 2)
JAC1(2, 1) = -JAC(2, 1)
JAC1(2, 2) = JAC(1, 1)
FOR K = 1 TO 2
    FOR L = 1 TO 2
        JAC1(K, L) = JAC1(K, L) / DET
    NEXT L
NEXT K
REM
REM *** MATRIX MULTIPLY ***
REM
FOR K = 1 TO T
    FOR L = 1 TO NOD
        X = 0
        FOR M = 1 TO T
            X = X + JAC1(K, M)
            * DER(M, L)
        NEXT M
        DERIV(K, L) = X
    NEXT L
NEXT K
REM
REM *** SET MATRIX TO 0 ***
REM
FOR K = 1 TO B
    FOR L = 1 TO DOF
```

```
                    BEE(K, L) = 0
                NEXT L
            NEXT K
            REM
            REM *** STRAIN DISPLACEMENT MATRIX ***
            REM ***  FOR AXISYMMETRIC STRAIN   ***
            REM
            SUM = 0
            FOR K = 1 TO NOD
                SUM = SUM + FUN(K) * COORD(K, 1)
            NEXT K
            FOR M = 1 TO NOD
                K = 2 * M
                L = K - 1
                BEE(1, L) = DERIV(1, M)
                BEE(3, K) = BEE(1, L)
                VOL(K) = DERIV(2, M)
                BEE(2, K) = VOL(K)
                BEE(3, L) = VOL(K)
                BEE(4, L) = FUN(M) / SUM
                VOL(L) = BEE(1, L) + BEE(4, L)
            NEXT M
            REM
            REM *** TRANSPOSE A MATRIX ***
            REM
            FOR K = 1 TO H
                FOR L = 1 TO DOF
                    BT(L, K) = BEE(K, L)
                NEXT L
            NEXT K
            REM
            REM *** MULTIPLY MATRIX BY A VECTOR ***
            REM
            FOR M = 1 TO DOF
                X = 0
                FOR MM = 1 TO H
                    X = X + BT(M, MM) * ELSO(MM)
                NEXT MM
                ELOAD(M) = X
            NEXT M
            QUOT = DET * K1 * K2 * 2 * PI * SUM
            FOR K = 1 TO DOF
                BLD(K) = BLD(K) + ELOAD(K) * QUOT
            NEXT K
        NEXT JJ
    NEXT II
    REM
    REM *** COMPUTE TOTAL BODYLOADS VECTOR ***
    REM
    FOR M = 1 TO DOF
        IF G(M) = 0 THEN GOTO 300
        BDYLDS(G(M)) = BDYLDS(G(M)) + BLD(M)
300 NEXT M
400 IF Z <> ITS THEN GOTO 500
    REM
    REM *** UPDATE ELEMENT STRESSES AND STRAINS ***
```

```
                        REM
                        SR(I, J) = SR(I, J) + SIGMA(1) - ELSO(1)
                        ER(I, J) = ER(I, J) + EPS(1)
                        SZ(I, J) = SZ(I, J) + SIGMA(2) - ELSO(2)
                        EZ(I, J) = EZ(I, J) + EPS(2)
                        TRZ(I, J) = TRZ(I, J) + SIGMA(3) - ELSO(3)
                        GRZ(I, J) = GRZ(I, J) + EPS(3)
                        ST(I, J) = ST(I, J) + SIGMA(4) - ELSO(4)
                        ET(I, J) = ET(I, J) + EPS(4)
                        S1 = SR(I, J)
                        S2 = ST(I, J)
                        S3 = SZ(I, J)
                        S4 = TRZ(I, J)
                        SBAR = SQR(.5 * ((S1 - S2) * (S1 - S2)
                        + (S2 - S3) * (S2 - S3) + (S3 - S1) * (S3 - S1)
                        + 6 * S4 * S4))
                        S1 = ER(I, J)
                        S2 = ET(I, J)
                        S3 = EZ(I, J)
                        S4 = GRZ(I, J)
                        EBAR = SQR(2 * ((S1 - S2) * (S1 - S2)
                        + (S2 - S3) * (S2 - S3) + (S3 - S1) * (S3 - S1)
                        + 3 * S4 * S4 / 2)) / 3
                        PRINT #3, USING "#########.############ "; SBAR;
                        PRINT #3, USING "#########.############ "; EBAR
500                 NEXT J
                NEXT I
            NEXT Z
        NEXT Y
        CHAIN "SUBFIN.EXE"
```

3.0 MOHR-COULOMB ANALYSIS PROGRAM

MOHRCOL1.EXE

```
CLEAR
REM
REM *** AXISYMMETRIC STRAIN OF A RECTANGULAR ELASTO-PLASTIC ***
REM ***     (MOHR COULOMB) SOLID USING 4-NODE QUADRILATERAL  ***
REM ***              ELEMENTS--INITIAL STRESS METHOD         ***
REM
DIM DEE(4, 4), SAMP(7, 2), NF(100, 2)
REM
REM ***                      INITIALZE DATA                  ***
REM
NODOF = 2:      REM Number of freedoms per node
DOF = 8:        REM Degrees of freedom per element
OPEN "INPT.FLE" FOR INPUT AS #1
   INPUT #1, NXE:  REM number of elements in x direction
   INPUT #1, NYE:  REM number of elements in y direction
   INPUT #1, N:    REM total number of freedoms in mesh
   INPUT #1, W:    REM half bandwidth
   INPUT #1, NN:   REM number of nodes in mesh
   INPUT #1, RN:   REM restrained nodes in mesh
   INPUT #1, NL:   REM number of nodes where solution is non zero
   INPUT #1, GP:   REM Gaussian integration order
   INPUT #1, AA:   REM element size in x direction
   INPUT #1, BB:   REM element size in y direction
   INPUT #1, V:    REM Poisson's Ratio
   INPUT #1, E:    REM Young's Modulus
   INPUT #1, INCS: REM Number of load increments
   INPUT #1, ITS:  REM Total number of iterations
   INPUT #1, COH:  REM Soil cohesion
   INPUT #1, PHI:  REM Soil friction angle
   INPUT #1, PSI:  REM Soil dilation angle
   NOD = DOF / NODOF
   CDMAX = W + 1
   R = N * CDMAX
   REM
   REM *** STRESS STRAIN MATRIX FOR ELASTIC AXISYMMETRYN ***
   REM
   V1 = V / (1 - V)
   VV = (1 - 2 * V) / (1 - V) * .5
   DEE(1, 1) = 1
   DEE(2, 2) = 1
   DEE(4, 4) = 1
   DEE(3, 3) = VV
   DEE(1, 2) = V1
   DEE(2, 1) = V1
   DEE(1, 4) = V1
   DEE(4, 1) = V1
   DEE(2, 4) = V1
   DEE(4, 2) = V1
   FOR I = 1 TO 4
       FOR J = 1 TO 4
           DEE(I, J) = DEE(I, J) * E * (1 - V) / (1 - 2 * V)
           / (1 + V)
```

```
            NEXT J
          NEXT I
          REM
          REM *** GAUSSIAN QUADRATURE ABSCISSAE AND WEIGHTS ***
          REM
          GP = INT(GP)
          IF GP = 1 THEN GOTO 700
          IF GP = 2 THEN GOTO 100
          IF GP = 3 THEN GOTO 200
          IF GP = 4 THEN GOTO 300
          IF GP = 5 THEN GOTO 400
          IF GP = 6 THEN GOTO 500
          IF GP = 7 THEN GOTO 600
          GP = 2
100       SAMP(1, 1) = 1 / SQR(3)
          SAMP(2, 1) = -SAMP(1, 1)
600       SAMP(1, 2) = 1
          SAMP(2, 2) = 1
          GOTO 700
200       SAMP(1, 1) = .2 * SQR(15)
          SAMP(2, 1) = 0
          SAMP(3, 1) = -SAMP(1, 1)
          SAMP(1, 2) = 5 / 9
          SAMP(2, 2) = 8 / 9
          SAMP(3, 2) = SAMP(1, 2)
          GOTO 700
300       SAMP(1, 1) = .861136311594053
          SAMP(2, 1) = .339981043584856
          SAMP(3, 1) = -SAMP(2, 1)
          SAMP(4, 1) = -SAMP(1, 1)
          SAMP(1, 2) = .3847854845137454
          SAMP(2, 2) = .652145154862546
          SAMP(3, 2) = SAMP(2, 2)
          SAMP(4, 2) = SAMP(1, 2)
          GOTO 700
400       SAMP(1, 1) = .906179845838664
          SAMP(2, 1) = .538469310105683
          SAMP(3, 1) = 0
          SAMP(4, 1) = -SAMP(2, 1)
          SAMP(5, 1) = -SAMP(1, 1)
          SAMP(1, 2) = .236926885056189
          SAMP(2, 2) = .478628670499366
          SAMP(3, 2) = .568888888888889
          SAMP(4, 2) = SAMP(2, 2)
          SAMP(5, 2) = SAMP(1, 2)
          GOTO 700
          SAMP(1, 1) = .9324695142031521
          SAMP(2, 1) = .661209386466265
          SAMP(3, 1) = .238619186083197
          SAMP(4, 1) = -SAMP(3, 1)
          SAMP(5, 1) = -SAMP(2, 1)
          SAMP(6, 1) = -SAMP(3, 1)
          SAMP(1, 2) = .17132449237917
          SAMP(2, 2) = .360761573048139
          SAMP(3, 2) = .467913934572691
          SAMP(4, 2) = SAMP(3, 2)
```

```
        SAMP(5, 2) = SAMP(2, 2)
        SAMP(6, 2) = SAMP(1, 2)
        GOTO 700
500     SAMP(1, 1) = .949107912342759
        SAMP(2, 1) = .741531185599394
        SAMP(3, 1) = .405845151377397
        SAMP(4, 1) = 0
        SAMP(5, 1) = -SAMP(3, 1)
        SAMP(6, 1) = -SAMP(2, 1)
        SAMP(7, 1) = -SAMP(1, 1)
        SAMP(1, 2) = .12948496616887
        SAMP(2, 2) = .297705391489277
        SAMP(3, 2) = .381830050505199
        SAMP(4, 2) = .417959183673469
        SAMP(5, 2) = SAMP(3, 2)
        SAMP(6, 2) = SAMP(2, 2)
        SAMP(7, 2) = SAMP(1, 2)
700     REM
        REM *** NODE FREEDOM ARRAY FOR > ONE FREEDOM PER NODE ***
        REM
        FOR I = 1 TO NN
            FOR J = 1 TO NODOF
                NF(I, J) = 1
            N ?T J
        NEXT I
        FOR K = 1 TO RN
            INPUT #1, RNODE: REM Restrained Node Number
            FOR L = 1 TO NODOF
                INPUT #1, RNT:    REM Restrained Node Type
                IF RNT = 1 THEN NF(RNODE, L) = 0
            NEXT L
        NEXT K
        K = 1
        FOR M = 1 TO NN
            FOR P = 1 TO NODOF
                IF NF(M, P) = 0 THEN GOTO 800
                NF(M, P) = K
                K = K + 1
800         NEXT P
        NEXT M
    CLOSE #1
    REM
    REM *** PRINT DATA TO BIN FILE FOR NEXT PHASE ***
    REM
    OPEN "MATRIX.DAT" FOR OUTPUT AS #2
    FOR I = 1 TO 4
        FOR J = 1 TO 4
            PRINT #2, USING "############.### "; DEE(I, J);
        NEXT J
        PRINT #2, " "
    NEXT I
    PRINT #2, GP
    FOR K = 1 TO GP
        PRINT #2, USING "############.### "; SAMP(K, 1); SAMP(K, 2)
    NEXT K
    PRINT #2, NN
```

```
FOR L = 1 TO NN
    FOR M = 1 TO NODOF
        PRINT #2, USING "####.### "; NF(L, M);
    NEXT M
    PRINT #2, " "
NEXT L
CLOSE #2
CHAIN "MOHRCOL2.EXE"

MOHRCOL2.EXE

CLS
CLEAR
DIM DEE(4, 4), SAMP(7, 2), NF(100, 2), COORD(20, 2), KM(20, 20)
DIM FUN(20), DER(20, 8), JAC(20, 2), JAC1(20, 2), DERIV(20, 4)
DIM VOL(20), DBEE(20, 20), BT(20, 4), BTDB(20, 20), G(10)
DIM BEE(20, 20), KB(200, 50)
REM
REM *** ELEMENT STIFFNESS INTEGRATION AND ASSEMBLY ***
REM
REM *** DATA INPUT ***
REM
OPEN "INPT.FLE" FOR INPUT AS #1
   INPUT #1, NXE, NYE, N, W, NN, RN, NL, GP, AA, BB, V, E, INCS
   INPUT #1, ITS, COH, PHI, PSI
   FOR I = 1 TO RN
       INPUT #1, DUMMY1$, DUMMY2$, DUMMY3$
   NEXT I
CLOSE #1
OPEN "MATRIX.DAT" FOR INPUT AS #1
   FOR I = 1 TO 4
       INPUT #1, DEE(I, 1), DEE(I, 2), DEE(I, 3), DEE(I, 4)
   NEXT I
   INPUT #1, NGP
   FOR J = 1 TO NGP
       INPUT #1, SAMP(J, 1), SAMP(J, 2)
   NEXT J
   INPUT #1, NNF
   FOR K = 1 TO NNF
       INPUT #1, NF(K, 1), NF(K, 2)
   NEXT K
CLOSE #1
T = 2
H = 4
NODOF = 2: DOF = 8
NOD = DOF / NODOF
PI = 4 * ATN(1)
REM
REM *** NODAL COORDINATES AND STEERING VECTOR FOR RECTANGULAR ***
REM *** MESH OF 8-NODE QUADRILATERAL PLANE ELEMENTS NUMBERING ***
REM ***                  IN THE Y-DIRECTION                    ***
REM
FOR I = 1 TO NXE
    FOR J = 1 TO NYE
        AO = (I - 1) * (NYE + 1) + J
        AL = AO + 1
```

```
AM = I * (NYE + 1) + J
AN = AM + 1
G(1) = NF(AL, 1)
G(2) = NF(AL, 2)
G(3) = NF(AO, 1)
G(4) = NF(AO, 2)
G(5) = NF(AM, 1)
G(6) = NF(AM, 2)
G(7) = NF(AN, 1)
G(8) = NF(AN, 2)
COORD(1, 1) = (I - 1) * AA
COORD(1, 2) = (NYE - J) * BB
COORD(2, 1) = (I - 1) * AA
COORD(2, 2) = (NYE - J + 1) * BB
COORD(3, 1) = I * AA
COORD(3, 2) = (NYE - J + 1) * BB
COORD(4, 1) = I * AA
COORD(4, 2) = (NYE - J) * BB
REM
REM *** SET MATRIX TO 0 ***
REM
FOR K = 1 TO DOF
    FOR L = 1 TO DOF
        KM(K, L) = 0
    NEXT L
NEXT K
FOR II = 1 TO GP
    FOR JJ = 1 TO GP
        K1 = SAMP(II, 2)
        K2 = SAMP(JJ, 2)
        REM
        REM *** LOCAL COORDINATE SHAPE FUNCTIONS AND ***
        REM ***     THEIR DERIVATIVES FOR 8-NODE     ***
        REM ***              QUADRILATERAL           ***
        REM
        ETA = SAMP(II, 1)
        XI = SAMP(JJ, 1)
        ETAM = .25 * (1 - ETA)
        ETAP = .25 * (1 + ETA)
        XIM = .25 * (1 - XI)
        XIP = .25 * (1 + XI)
        FUN(1) = 4 * XIM * ETAM
        FUN(2) = 4 * XIM * ETAP
        FUN(3) = 4 * XIP * ETAP
        FUN(4) = 4 * XIP * ETAM
        DER(1, 1) = -ETAM
        DER(1, 2) = -ETAP
        DER(1, 3) = ETAP
        DER(1, 4) = ETAM
        DER(2, 1) = -XIM
        DER(2, 2) = XIM
        DER(2, 3) = XIP
        DER(2, 4) = -XIP
        REM
        REM *** MATRIX MULTIPLY ***
        REM
```

```
FOR K = 1 TO T
    FOR L = 1 TO T
        X = 0
        FOR M = 1 TO NOD
            X = X + DER(K, M) * COORD(M, L)
        NEXT M
        JAC(K, L) = X
    NEXT L
NEXT K
REM
REM *** INVERT THE JACOBIAN MATRIX ***
REM
DET = JAC(1, 1) * JAC(2, 2) - JAC(1, 2)
* JAC(2, 1)
JAC1(1, 1) = JAC(2, 2)
JAC1(1, 2) = -JAC(1, 2)
JAC1(2, 1) = -JAC(2, 1)
JAC1(2, 2) = JAC(1, 1)
FOR K = 1 TO 2
    FOR L = 1 TO 2
        JAC1(K, L) = JAC1(K, L) / DET
    NEXT L
NEXT K
REM
REM *** MATRIX MULTIPLY ***
REM
FOR K = 1 TO T
    FOR L = 1 TO NOD
        X = 0
        FOR M = 1 TO T
            X = X + JAC1(K, M) * DER(M, L)
        NEXT M
        DERIV(K, L) = X
    NEXT L
NEXT K
REM
REM *** SET MATRIX TO 0 ***
REM
FOR K = 1 TO H
    FOR I = 1 TO DOF
        BEE(K, L) = 0
    NEXT L
NEXT K
REM
REM *** STRAIN DISPLACEMENT MATRIX FOR ***
REM ***       AXISYMMETRIC STRAIN       ***
REM
SUM = 0
FOR K = 1 TO NOD
    SUM = SUM + FUN(K) * COORD(K, 1)
NEXT K
FOR M = 1 TO NOD
    K = 2 * M
    L = K - 1
    BEE(1, L) = DERIV(1, M)
    BEE(3, K) = BEE(1, L)
```

```
                VOL(K) = DERIV(2, M)
                BEE(2, K) = VOL(K)
                BEE(3, L) = VOL(K)
                BEE(4, L) = FUN(M) / SUM
                VOL(L) = BEE(1, L) + BEE(4, L)
            NEXT M
            REM
            REM *** MATRIX MULTIPLY ***
            REM
            FOR K = 1 TO H
                FOR L = 1 TO DOF
                    X = 0
                    FOR M = 1 TO H
                        X = X + DEE(K, M) * BEE(M, L)
                    NEXT M
                    DBEE(K, L) = X
                NEXT L
            NEXT K
            REM
            REM *** TRANSPOSE A MATRIX ***
            REM
            FOR K = 1 TO H
                FOR L = 1 TO DOF
                    BT(L, K) = BEE(K, L)
                NEXT L
            NEXT K
            REM
            REM *** MATRIX MULTIPLY ***
            REM
            FOR K = 1 TO DOF
                FOR L = 1 TO DOF
                    X = 0
                    FOR M = 1 TO H
                        X = X + BT(K, M) * DBEE(M, L)
                    NEXT M
                    BTDB(K, L) = X
                NEXT L
            NEXT K
            QUOT = DET * K1 * K2 * 2 * PI * SUM
            FOR K = 1 TO DOF
                FOR L = 1 TO DOF
                    BTDB(K, L) = BTDB(K, L) * QUOT
                NEXT L
            NEXT K
            REM
            REM *** ADD TWO MATRICES ***
            REM
            FOR K = 1 TO DOF
                FOR L = 1 TO DOF
                    KM(K, L) = KM(K, L) + BTDB(K, L)
                NEXT L
            NEXT K
        NEXT JJ
    NEXT II
    REM
    REM *** ASSEMBLES ELEMENT MATRICES INTO GLOBAL MATRIX ***
```

```
            REM
            CDMAX = W + 1
            FOR K = 1 TO DOF
                IF G(K) = 0 THEN GOTO 200
                FOR L = 1 TO DOF
                    IF G(L) = 0 THEN GOTO 100
                    CD = G(L) - G(K) + CDMAX
                    IF CD > CDMAX THEN GOTO 100
                    KB(G(K), CD) = KB(G(K), CD) + KM(K, L)
100             NEXT L
200         NEXT K
        NEXT J
    NEXT I
    REM
    REM  *** CHOLESKI REDUCTION OF A SYMETRICAL BAND ***
    REM
    FOR I = 1 TO N
        X = 0
        FOR J = 1 TO W
            X = X + (KB(I, J) * KB(I, J))
        NEXT J
        KB(I, W + 1) = SQR(KB(I, W + 1) - X)
        FOR K = 1 TO W
            X = 0
            IF I + K > N THEN GOTO 500
            IF K = W THEN GOTO 400
            L = W - K
300         X = X + (KB(I + K, L) * KB(I, L + K))
            L = L - 1
            IF L = 0 THEN GOTO 400
            GOTO 300
400         A = I + K
            B = (W - K) + 1
            KB(A, B) = (KB(A, B) - X) / KB(I, W + 1)
500     NEXT K
    NEXT I
    OPEN "CONSTIT.DAT" FOR OUTPUT AS #2
       FOR I = 1 TO N
           FOR J = 1 TO DOF
               PRINT #2, USING "##############.##### "; KB(I, J);
           NEXT J
           PRINT #2, " "
       NEXT I
    CLOSE #2
    CHAIN "MOHRCOL3.EXE"

    MOHRCOL3.EXE

    CLEAR
    REM
    REM  *** INCREMENT THE LOADS ***
    REM
    DIM DEE(8, 4), DPL(8, 4), PL(8, 4), SAMP(3, 2), COORD(8, 2)
    DIM DERIV(8, 4), BEE(8, 8), ELD(8), VOL(8), EPS(4), SIGMA(4)
    DIM DF(8), ELOAD(8), VVL(30), STORKB(30), NO(30), G(8), KB(30, 8)
    DIM SR(30, 30), ST(30, 30), SZ(30, 30), TRZ(30, 30), ER(30, 30)
```

```
    DIM JAC(8, 2), JAC1(8, 2), DER(8, 4), FUN(4), ELSO(4), SPL(4)
    DIM ET(30, 30), EZ(30, 30), GRZ(30, 30), NF(30, 2)
    DIM LOADS(30), BDYLDS(30), OLDLDS(30)
    DOF = 8
    OPEN "INPT.FLE" FOR INPUT AS #1
       INPUT #1, NXE, NYE, N, W, NN, RN, NL
       INPUT #1, GP, AA, BB, V, E, INCS, ITS
       INPUT #1, COH, PHI, PSI
    CLOSE #1
    DOF = 8
    NODOF = 2
    NOD = DOF / NODOF
    T = 2
    H = 4
    PI = 4 * ATN(1)
    PHI = PHI * PI / 180
    PSI = PSI * PI / 180
    SNPH = SIN(PHI)
    CSPH = COS(PHI)
    SQ3 = SQR(3)
    OPEN "CONSTIT.DAT" FOR INPUT AS #1
       FOR I = 1 TO N
         FOR J = 1 TO DOF
             INPUT #1, KB(I, J)
         NEXT J
       NEXT I
    CLOSE #1
    OPEN "MATRIX.DAT" FOR INPUT AS #1
       FOR I = 1 TO 4
           INPUT #1, DEE(I, 1), DEE(I, 2), DEE(I, 3), DEE(I, 4)
       NEXT I
       INPUT #1, NGP
       FOR J = 1 TO NGP
           INPUT #1, SAMP(J, 1), SAMP(J, 2)
       NEXT J
       INPUT #1, NNF
       FOR K = 1 TO NNF
           INPUT #1, NF(K, 1), NF(K, 2)
       NEXT K
    CLOSE #1
    OPEN "LOADS.DAT" FOR INPUT AS #1
       INPUT #1, NLDS
       FOR I = 1 TO NLDS
           INPUT #1, NO(I), VVL(I)
       NEXT I
    CLOSE #1
    OPEN "STRAINS.DAT" FOR OUTPUT AS #3
       REM
       REM *** INCREMENT THE LOADS ***
       REM
       FOR Y = 1 TO INCS
           ITER = 0
           REM
           REM *** ITERATE TO REDISTRIBUTE EXCESS ELEMENT STRESSES ***
           REM
50         ITER = ITER + 1
```

```
            Z = 1
                FOR I = 1 TO N
                    LOADS(I) = 0
                NEXT I
                FOR I = 1 TO NL
                    LOADS(NO(I)) = VVL(I)
                NEXT I
                FOR I = 1 TO N
                    LOADS(I) = LOADS(I) + BDYLDS(I)
                NEXT I
                REM
                REM *** CHOLESKI FORWARD SUBSTITUTION ***
                REM
                LOADS(1) = LOADS(1) / KB(1, W + 1)
                FOR I = 2 TO N
                    X = 0
                    K = 1
                   IF I <= W + 1 THEN K = (W - I) + 2
                   FOR J = K TO W
                        XXX = ((I + J) - W) - 1
                        X = X + KB(I, J) * LOADS(XXX)
                   NEXT J
                   LOADS(I) = (LOADS(I) - X) / KB(I, W + 1)
                NEXT I
                REM
                REM *** CHOLESKI BACKWARD SUBSTITUTION ***
                REM
                LOADS(N) = LOADS(N) / KB(N, W + 1)
                I = N - 1
100             X = 0
                L = I + W
                IF I > N - W THEN L = N
                M = I + 1
                FOR J = M TO L
                    XXX = ((W + I) - J) + 1
                    X = X + KB(J, XXX) * LOADS(J)
                NEXT J
                LOADS(I) = (LOADS(I) - X) / KB(I, W + 1)
                I = I - 1
                IF I = 0 THEN GOTO 200
                GOTO 100
                REM
                REM *** SET VECTOR TO 0 ***
                REM
200             FOR I = 1 TO N
                    BDYLDS(I) = 0
                NEXT I
                REM
                REM *** ITERATION TOLERANCE ***
                K2 = 0
                FOR I = 1 TO N
                    IF ABS(LOADS(I)) > K2 THEN K2 = ABS(LOADS(I))
                NEXT I
                FOR I = 1 TO N
                    K1 = ABS((LOADS(I) - OLDLDS(I)) / K2)
                    IF K1 > .001 THEN Z = 0
```

```
        OLDLDS(I) = LOADS(I)
    NEXT I
    IF ITER = 1 THEN Z = 0
    REM
    REM *** INSPECT ALL ELEMENTS ***
    REM
    FOR I = 1 TO NXE
        FOR J = 1 TO NYE
            AO = (I - 1) * (NYE + 1) + J
            AL = AO + 1
            AM = I * (NYE + 1) + J
            AN = AM + 1
            G(1) = NF(AL, 1)
            G(2) = NF(AL, 2)
            G(3) = NF(AO, 1)
            G(4) = NF(AO, 2)
            G(5) = NF(AM, 1)
            G(6) = NF(AM, 2)
            G(7) = NF(AN, 1)
            G(8) = NF(AN, 2)
            COORD(1, 1) = (I - 1) * AA
            COORD(1, 2) = (NYE - J) * BB
            COORD(2, 1) = (I - 1) * AA
            COORD(2, 2) = (NYE - J + 1) * BB
            COORD(3, 1) = I * AA
            COORD(3, 2) = (NYE - J + 1) * BB
            COORD(4, 1) = I * AA
            COORD(4, 2) = (NYE - J) * BB
            REM
            REM ***STRAINS AT ELEMENT 'CENTRES' ***
            REM
            II = 2
            JJ = 2
            SAMP(2, 1) = 0
            REM
            REM *** LOCAL COORDINATE SHAPE FUNCTIONS ***
            REM *** AND THEIR DERIVATIVES FOR 8-NODE ***
            REM ***            QUADRILATERAL          ***
            REM
            ETA = SAMP(II, 1)
            XI = SAMP(JJ, 1)
            ETAM = .25 * (1 - ETA)
            ETAP = .25 * (1 + ETA)
            XIM = .25 * (1 - XI)
            XIP = .25 * (1 + XI)
            FUN(1) = 4 * XIM * ETAM
            FUN(2) = 4 * XIM * ETAP
            FUN(3) = 4 * XIP * ETAP
            FUN(4) = 4 * XIP * ETAM
            DER(1, 1) = -ETAM
            DER(1, 2) = -ETAP
            DER(1, 3) = ETAP
            DER(1, 4) = ETAM
            DER(2, 1) = -XIM
            DER(2, 2) = XIM
            DER(2, 3) = XIP
```

```
DER(2, 4) = -XIP
REM
REM *** MATRIX MULTIPLY ***
REM
FOR K = 1 TO T
    FOR L = 1 TO T
        X = 0
        FOR M = 1 TO NOD
            X = X + DER(K, M) * COORD(M, L)
        NEXT M
        JAC(K, L) = X
    NEXT L
NEXT K
REM
REM *** INVERT THE JACOBIAN MATRIX ***
REM
DET = JAC(1, 1) * JAC(2, 2) - JAC(1, 2)
* JAC(2, 1)
JAC1(1, 1) = JAC(2, 2)
JAC1(1, 2) = -JAC(1, 2)
JAC1(2, 1) = -JAC(2, 1)
JAC1(2, 2) = JAC(1, 1)
FOR K = 1 TO 2
    FOR L = 1 TO 2
            JAC1(K, L) = JAC1(K, L) / DET
    NEXT L
NEXT K
REM
REM *** MATRIX MULTIPLY ***
REM
FOR K = 1 TO T
    FOR L = 1 TO NOD
        X = 0
        FOR M = 1 TO T
            X = X + JAC1(K, M) * DER(M, L)
        NEXT M
        DERIV(K, L) = X
    NEXT L
NEXT K
REM
REM *** SET MATRIX TO 0 ***
REM
FOR K = 1 TO H
    FOR L = 1 TO DOF
        BEE(K, L) = 0
    NEXT L
NEXT K
REM
REM *** STRAIN DISPLACEMENT MATRIX FOR ***
REM ***       AXISYMMETRIC STRAIN      ***
REM
SUM = 0
FOR K = 1 TO NOD
    SUM = SUM + FUN(K) * COORD(K, 1)
NEXT K
FOR M = 1 TO NOD
```

```
        K = 2 * M
        L = K - 1
        BEE(1, L) = DERIV(1, M)
        BEE(3, K) = BEE(1, L)
        VOL(K) = DERIV(2, M)
        BEE(2, K) = VOL(K)
        BEE(3, L) = VOL(K)
        BEE(4, L) = FUN(M) / SUM
        VOL(L) = BEE(1, L) + BEE(4, L)
    NEXT M
    FOR M = 1 TO DOF
    IF G(M) = 0 THEN ELD(M) = 0
    IF G(M) <> 0 THEN ELD(M) = LOADS(G(M))
    NEXT M
    REM
    REM *** MULTIPLY A MATRIX BY A VECTOR ***
    REM
    FOR M = 1 TO H
        X = 0
        FOR MM = 1 TO DOF
            X = X + BEE(M, MM) * ELD(MM)
        NEXT MM
        EPS(M) = X
    NEXT M
    REM
    REM *** ELASTIC STRESSES ARE SIGMA ***
    REM
    REM
    REM *** MULTIPLY A MATRIX BY A VECTOR ***
    REM
    FOR M = 1 TO H
        X = 0
        FOR MM = 1 TO H
            X = X + DEE(M, MM) * EPS(MM)
        NEXT MM
        SIGMA(M) = X
    NEXT M
    REM
    REM *** SET VECTOR TO 0 ***
    REM
    FOR M = 1 TO H
        ELSO(M) = 0
    NEXT M
    REM
    REM *** SET VECTOR TO 0 ***
    REM
    FOR M = 1 TO DOF
        BLD(M) = 0
    NEXT M
    S1 = SIGMA(1) + SR(I, J)
    S2 = SIGMA(4) + ST(I, J)
    S3 = SIGMA(2) + SZ(I, J)
    S4 = SIGMA(3) + TRZ(I, J)
    REM
    REM *** INVARIANTS OF THE STRESS TENSOR ***
    REM
```

```
                    DSBAR = SQR(.5 * ((S1 - S2) * (S1 - S2)
                    + (S2 - S3) * (S2 - S3) + (S3 - S1) * (S3 - S1)
                    + 6 * S4 * S4))
                    DEV1 = (2 * S1 - S2 - S3) / 3
                    DEV2 = (2 * S2 - S3 - S1) / 3
                    DEV3 = (2 * S3 - S1 - S2) / 3
                    MEAN = (S1 + S2 + S3) / 3
                    J3 = DEV1 * DEV2 * DEV3 - DEV2 * S4 * S4
                    IF DSBAR <> 0 THEN GOTO 300
                    LODE = 999.999
                    GOTO 600
300                 LODE = -13.5 * J3 / (DSBAR * DSBAR * DSBAR)
                    IF LODE >= 1 THEN GOTO 400
                    IF LODE <= -1 THEN GOTO 500
                    ADJ = SQR(1 - LODE * LODE)
                    LODE = ATN(LODE / ADJ) / 3
                    GOTO 600
400                 LODE = PI / 6
                    GOTO 600
500                 LODE = -PI / 6
600                 SNTH = SIN(LODE)
                    CSTH = COS(LODE)
                    NEWF = -SNPH * MEAN + DSBAR * (CSTH / SQR(3)
                    - SNTH * SNPH / 3) - COH * CSPH
                    IF NEWF < 0 THEN GOTO 1900
                    S1 = SR(I, J)
                    S2 = ST(I, J)
                    S3 = SZ(I, J)
                    S4 = TRZ(I, J)
                    REM
                    REM *** INVARIANTS OF THE STRESS TENSOR ***
                    REM
                    DSBAR = SQR(.5 * ((S1 - S2) * (S1 - S2)
                    + (S2 - S3) * (S2 - S3) + (S3 - S1) * (S3 - S1)
                    + 6 * S4 * S4))
                    DEV1 = (2 * S1 - S2 - S3) / 3
                    DEV2 = (2 * S2 - S3 - S1) / 3
                    DEV3 = (2 * S3 - S1 - S2) / 3
                    MEAN = (S1 + S2 + S3) / 3
                    J3 = DEV1 * DEV2 * DEV3 - DEV2 * S4 * S4
                    IF DSBAR <> 0 THEN GOTO 700
                    LODE = 999.999
                    GOTO 1000
700                 LODE = -13.5 * J3 / (DSBAR * DSBAR * DSBAR)
                    IF LODE >= 1 THEN GOTO 800
                    IF LODE <= -1 THEN GOTO 900
                    ADJ = SQR(1 - LODE * LODE)
                    LODE = ATN(LODE / ADJ) / 3
                    GOTO 1000
800                 LODE = PI / 6
                    GOTO 1000
900                 LODE = -PI / 6
1000                SNTH = SIN(LODE)
                    CSTH = COS(LODE)
                    F = -SNPH * MEAN + DSBAR * (CSTH / SQR(3) -
                    SNTH * SNPH / 3) - COH * CSPH
```

```
                REM *** STRESS STRAIN MATRIX FOR MOHR ***
                REM ***      COULOMB PLASTICITY       ***
                REM
                SBAR = SQR(.5 * ((S1 - S2) * (S1 - S2)
                + (S2 - S3) * (S2 - S3) + (S3 - S1) * (S3 - S1)
                + 6 * S4 * S4))
                DEV1 = (2 * S1 - S2 - S3) / 3
                DEV2 = (2 * S2 - S3 - S1) / 3
                DEV3 = (2 * S3 - S1 - S2) / 3
                MEAN = (S1 + S2 + S3) / 3
                J3 = DEV1 * DEV2 * DEV3 - DEV2 * S4 * S4
                IF SBAR <> 0 THEN GOTO 1050
                THETA = 999.999
                GOTO 1300
1050            THETA = -13.5 * J3 / (SBAR * SBAR * SBAR)
                IF THETA >= 1 THEN GOTO 1100
                IF THETA <= -1 THEN GOTO 1200
                ADJ = SQR(1 - THETA * THETA)
                THETA = ATN(THETA / ADJ) / 3
                GOTO 1300
1100            THETA = PI / 6
                GOTO 1300
1200            THETA = -PI / 6
1300            SRD = (2 * S1 - S2 - S3) / 3
                STD = (2 * S2 - S3 - S1) / 3
                SZD = (2 * S3 - S1 - S2) / 3
                ROOT3 = SQR(3)
                PRAD = PHI
                IF ABS(THETA) <= .523424 THEN GOTO 1400
                F1 = SIN(PRAD)
                F2 = .25 / SBAR * (3 - SIN(PRAD))
                F3 = 0
                GOTO 1500
1400            F1 = SIN(PRAD)
                F2 = COS(THETA) * ROOT3 * .5 / DSBAR
                * ((1 + SIN(THETA) / COS(THETA) * SIN(3*THETA)
                + SIN(PRAD)/ROOT3*(SIN(3 * THETA)/COS(3 * THETA)
                - SIN(THETA)/COS(THETA)))
                F3 = (ROOT3 * SIN(THETA) + SIN(PRAD)
                * COS(THETA)) * 1.5 / (SBAR * SBAR
                * COS(3 * THETA))
1500            IF ABS(THETA) <= .523424 THEN GOTO 1600
                Q1 = SIN(PRAD)
                Q2 = .25 / SBAR * (3 - SIN(PRAD))
                Q3 = 0
                GOTO 1700
1600            Q1 = SIN(PRAD)
                Q2 = COS(THETA) * ROOT3 * .5 / DSBAR
                * ((1 + SIN(THETA) / COS(THETA) * SIN(3*THETA)
                + SIN(PRAD)/ROOT3*(SIN(3 * THETA)/COS(3 * THETA)
                - SIN(THETA)/COS(THETA)))
                Q3 = (ROOT3 * SIN(THETA) + SIN(PRAD)
                * COS(THETA)) * 1.5 / (SBAR * SBAR
                * COS(3 * THETA))
1700            DF(1) = F1 / 3 + F2 * SRD + F3 / 3 * (SRD * SRD
                + 2 * SZD * STD * S4 * S4)
```

```
DF(2) = F1 / 3 + F2 * SZD + F3 / 3 * (SZD * SZD
+ 2 * SRD * STD * S4 * S4)
DF(3) = 2 * F2 * S4 - 2 * F3 * S4 * STD
DF(4) = F1 / 3 + F2 * STD + F3 / 3 * (2 * SZD
* SRD + STD * STD - 2 * S4 * S4)
DQ(1) = Q1 / 3 + Q2 * SRD + Q3 / 3 * (SRD * SRD
+ 2 * SZD * STD * S4 * S4)
DQ(2) = Q1 / 3 + Q2 * SZD + Q3 / 3 * (SZD * SZD
+ 2 * SRD * STD * S4 * S4)
DQ(3) = 2 * Q2 * S4 - 2 * Q3 * S4 * STD
DQ(4) = Q1 / 3 + Q2 * STD + Q3 / 3 * (2 * SZD
* SRD + STD * STD - 2 * S4 * S4)
REM
REM *** MULTIPLY A MATRIX BY A VECTOR ***
REM
FOR M = 1 TO 4
    FOR MM = 1 TO 4
        X = X + DEE(M, MM) * DF(MM)
    NEXT MM
    DDF(M) = X
NEXT M
REM
REM *** MULTIPLY A MATRIX BY A VECTOR ***
REM
FOR M = 1 TO 4
    X = 0
    FOR MM = 1 TO 4
        X = X + DEE(M, MM) * DQ(MM)
    NEXT MM
    DDQ(M) = X
NEXT M
VVL = 0
FOR K = 1 TO 4
    VVL = VVL + DDF(K) * DQ(K)
NEXT K
FOR K = 1 TO 4
    FOR L = 1 TO 4
        DDQF(K, L) = DDQ(K) * DF(L)
        DDQF(K, L) = DDQF(K, L) / VVL
    NEXT L
NEXT K
FOR K = 1 TO 4
    FOR L = 1 TO 4
        X = 0
        FOR M = 1 TO 4
            X = X + DDQF(K, M) * DEE(M, L)
        NEXT M
        PL(K, L) = X
    NEXT L
NEXT K
IF F > 0 THEN FAC = 1
IF F <= 0 THEN FAC = NEWF / (NEWF - F)
FOR K = 1 TO H
    FOR L = 1 TO H
        DPL(K, L) = DEE(K, L) - FAC * PL(K, L)
    NEXT L
```

```
NEXT K
REM
REM *** ELASTO-PLASTIC STRESSES ARE SPL ***
REM
REM
REM *** MULTIPLY A MATRIX BY A VECTOR ***
REM
FOR K = 1 TO H
    X = 0
    FOR L = 1 TO H
        X = X + DPL(K, L) * EPS(L)
    NEXT L
    SPL(K) = X
NEXT K
FOR M = 1 TO H
    ELSO(M) = SIGMA(M) - SPL(M)
NEXT M
SAMP(2, 1) = -1 / SQR(3)
REM
REM *** INTEGRATE ALL EXCESS STRESSES TO ***
REM ***          FIND NODAL LOADS          ***
REM
FOR II = 1 TO GP
    FOR JJ = 1 TO GP
        K1 = SAMP(II, 2)
        K2 = SAMP(JJ, 2)
        REM
        REM *** LOCAL COORDINATE SHAPE ***
        REM ***  FUNCTIONS AND THEIR   ***
        REM *** DERIVATIVES FOR 8-NODE ***
        REM ***     QUADRILATERAL      ***
        REM
        ETA = SAMP(II, 1)
        XI = SAMP(JJ, 1)
        ETAM = .25 * (1 - ETA)
        ETAP = .25 * (1 + ETA)
        XIM = .25 * (1 - XI)
        XIP = .25 * (1 + XI)
        FUN(1) = 4 * XIM * ETAM
        FUN(2) = 4 * XIM * ETAP
        FUN(3) = 4 * XIP * ETAP
        FUN(4) = 4 * XIP * ETAM
        DER(1, 1) = -ETAM
        DER(1, 2) = -ETAP
        DER(1, 3) = ETAP
        DER(1, 4) = ETAM
        DER(2, 1) = -XIM
        DER(2, 2) = XIM
        DER(2, 3) = XIP
        DER(2, 4) = -XIP
        REM
        REM *** MATRIX MULTIPLY ***
        REM
        FOR K = 1 TO T
            FOR L = 1 TO T
                X = 0
```

```
            FOR M = 1 TO NOD
                X = X + DER(K, M)
                * COORD(M, L)
            NEXT M
            JAC(K, L) = X
        NEXT L
    NEXT K
    REM
    REM *** INVERT THE JACOBIAN MATRIX ***
    REM
    DET = JAC(1, 1) * JAC(2, 2) - JAC(1, 2)
    * JAC(2, 1)
    JAC1(1, 1) = JAC(2, 2)
    JAC1(1, 2) = -JAC(1, 2)
    JAC1(2, 1) = -JAC(2, 1)
    JAC1(2, 2) = JAC(1, 1)
    FOR K = 1 TO 2
        FOR L = 1 TO 2
            JAC1(K, L) = JAC1(K, L) / DET
        NEXT L
    NEXT K
    REM
    REM *** MATRIX MULTIPLY ***
    REM
    FOR K = 1 TO T
        FOR L = 1 TO NOD
            X = 0
            FOR M = 1 TO T
                X = X + JAC1(K, M)
                * DER(M, L)
            NEXT M
            DERIV(K, L) = X
        NEXT L
    NEXT K
    REM
    REM *** SET MATRIX TO 0 ***
    REM
    FOR K = 1 TO H
        FOR L = 1 TO DOF
            BEE(K, L) = 0
        NEXT L
    NEXT K
    REM
    REM *** STRAIN DISPLACEMENT MATRIX ***
    REM ***  FOR AXISYMMETRIC STRAIN   ***
    REM
    SUM = 0
    FOR K = 1 TO NOD
        SUM = SUM + FUN(K) * COORD(K, 1)
    NEXT K
    FOR M = 1 TO NOD
        K = 2 * M
        L = K - 1
        BEE(1, L) = DERIV(1, M)
        BEE(3, K) = BEE(1, L)
        VOL(K) = DERIV(2, M)
```

```
                    BEE(2, K) = VOL(K)
                    BEE(3, L) = VOL(K)
                    BEE(4, L) = FUN(M) / SUM
                    VOL(L) = BEE(1, L) + BEE(4, L)
                NEXT M
                REM
                REM *** TRANSPOSE A MATRIX ***
                REM
                FOR K = 1 TO H
                    FOR L = 1 TO DOF
                        BT(L, K) = BEE(K, L)
                    NEXT L
                NEXT K
                REM
                REM *** MULTIPLY A MATRIX BY VECTOR ***
                REM
                FOR M = 1 TO DOF
                    X = 0
                    FOR MM = 1 TO H
                        X = X + BT(M, MM) * ELSO(MM)
                    NEXT MM
                    ELOAD(M) = X
                NEXT M
                QUOT = DET * K1 * K2 * 2 * PI * SUM
                FOR K = 1 TO DOF
                    BLD(K) = BLD(K) + ELOAD(K) * QUOT
                NEXT K
            NEXT JJ
        NEXT II
        REM
        REM *** COMPUTE TOTAL BODYLOADS VECTOR ***
        REM
        FOR M = 1 TO DOF
            IF G(M) = 0 THEN GOTO 1800
            BDYLDS(G(M)) = BDYLDS(G(M)) + BLD(M)
1800     NEXT M
1900     IF Z <> 1 AND ITER <> ITS THEN GOTO 2000
        REM
        REM *** UPDATE ELEMENT STRESSES AND STRAINS ***
        REM
        SR(I, J) = SR(I, J) + SIGMA(1) - ELSO(1)
        ER(I, J) = ER(I, J) + EPS(1)
        SZ(I, J) = SZ(I, J) + SIGMA(2) - ELSO(2)
        EZ(I, J) = EZ(I, J) + EPS(2)
        TRZ(I, J) = TRZ(I, J) + SIGMA(3) - ELSO(3)
        GRZ(I, J) = GRZ(I, J) + EPS(3)
        ST(I, J) = ST(I, J) + SIGMA(4) - ELSO(4)
        ET(I, J) = ET(I, J) + EPS(4)
        S1 = SR(I, J)
        S2 = ST(I, J)
        S3 = SZ(I, J)
        S4 = TRZ(I, J)
        SBAR = SQR(.5 * ((S1 - S2) * (S1 - S2)
        + (S2 - S3) * (S2 - S3) + (S3 - S1) * (S3 - S1)
        + 6 * S4 * S4))
        S1 = ER(I, J)
```

```
                    S2 = ET(I, J)
                    S3 = EZ(I, J)
                    S4 = GRZ(I, J)
                    EBAR = SQR(2 * ((S1 - S2) * (S1 - S2)
                    + (S2 - S3) * (S2 - S3) + (S3 - S1) * (S3 - S1)
                    + 6 * S4 * S4))
                    PRINT #3, USING "#########.############ "; SBAR;
                    PRINT #3, USING "#########.############ "; EBAR
2000            NEXT J
            NEXT I
            IF Z <> 1 AND ITER <> ITS THEN GOTO 50
            PRINT #3, USING "###"; ITER
            NEXT Y
            CHAIN "SUBFIN.EXE"
```