411

COMPUTER PROGRAM DEVELOPMENT FOR THE ANALYSIS OF INELASTIC BEAM AND SOIL BEHAVIOUR IN GEOTECHNICAL DESIGN

Prepared By : C.T.Howie
Postgraduate student in the
Department of Civil Engineering
University of Cape Town

Prepared For : The Department of Civil Engineering
University of Cape Town

February 1992

Thesis prepared in partial fulfillment of the requirements for the degree of MSc(Eng) in Civil Engineering.

DECLARATION

I, Cameron Telfer Howie, hereby declare that this thesis is my own work and that it has not been submitted for a degree at any other university.

Signed by candidate

Signature Removed

C.T.Howie
April 1993

ACKNOWLEDGEMENTS

I would especially like to thank my supervisor, Dr. F. Scheele, for his invaluable support and inspiration during the course of this research, and his tireless devotion to extract the maximum potential from his students.

In addition, I extend many thanks to Mr. P. Day, of Jones and Wagener Inc., Johannesburg, for performing the field tests on the soldier piles, the results of which play a large part in corroborating the methods presented.

Thanks also to Mr. T. Smith for use of the results of a laboratory test conducted by him at this univeristy.

SYNOPSIS

Computer-aided engineering requires the correct implementation of design methods in computer programs so as to play a beneficial role in engineering practice. This thesis describes the development of a computer program to analyse geotechnical engineering problems based on the principles of beam-soil interaction where the beam is supported by a single or two-layer soil system.

In 1867, a foundation model was proposed by Winkler in which the elastic foundation beneath a horizontal beam could be viewed as a series of independent springs. Foundation reaction to beam deflection is, therefore, linear. A stiffness matrix, for use in matrix methods of structural analysis, has been developed to define this beam-soil interaction, and such a method can be incorporated into a computer program. Furthermore, an iterative technique was created to allow for inelastic soil response when using the elastic stiffness matrix. However, such a technique did not consider realistic soil behaviour, and has limitations is used for practical design.

This research work describes how use can be made of the pressure-displacement response relationship for a soil to bring greater realism to beam-soil modelling and analysis. Such a relationship is commonly determined in geotechnical design procedures through a plate load test in the field. In addition, the iterative technique is extended to include non-linear beam behaviour as well, and plastic hinging of the beam material is incorporated to enable limitation of inelastic response.

While previous research has only considered foundations of a single soil only, a procedure to model a two-layered system is developed. Two-layered foundations are required for proper modelling of soldier pile support systems, an area of structural design in geotechnics chosen to demonstrate realistic design potential for the computer program. The two-layered principle is based on the derivation of a control parameter to differentiate between response from just the upper soil layer, and a combined response from both soil layers. The procedure is relatively simple, and no extra information is required other than the two pressure-displacement relationships for the individual soil layers.

A desktop computer program is described which incorporates the inelastic analysis features, as well as the two-layered soil system. The program makes use of a graphical user interface to offer the user an easy, interactive environment for analysing beam-on-soil foundation problems. As such, the program can be used directly, or for further research into beam-soil interaction.

The program is applied in the analysis of both field and laboratory tests to ascertain its accuracy in predicting beam-soil interaction. The laboratory test measures the deflection of a horizontal beam on a single soil foundation medium, where the beam is loaded by a single jack at approximately mid-span. Computer predictions for such a test were in very close agreement with the laboratory observations, despite the small magnitude of beam displacements, and the fact the beam-soil system suffered a bearing capacity failure which affected the beam deflection.

The field test was performed to investigate the performance of a flexible soldier pile under high anchor loading. Results of the computer analyses again show the program's predictions to be in very close agreement with the field measurements. Currently, the program does not include the facility to model soil layers behind a soldier pile, but the method developed in this thesis can easily incorporate multiple pressure-displacement curves for different soils.

Final conclusions drawn express a need for more research into soldier pile systems before the techniques of this work can be used for routine design. Nevertheless, the development of the program has made a significant contribution to advancing the use of computer-aided design in this field of geotechnical engineering.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

GLOSSARY

abstract data type	Computer programming term used to describe a conceptually single unit of data which comprises many primitive data types (integer, real number, character).
algorithm	A finite number of steps required to solve a given problem. An algorithm must terminate for any input specified.
beam element	A finite subsection of a beam, whose boundaries are defined by nodal points.
beam modulus	Parameter describing the measure of beam displacement under an applied stress. Elastic behaviour is characterised by the Young's Modulus, denoted by E.
combination depth	Minimum beam deflection into the upper layer of a two-layer soil system, at which the lower soil layer is compressed by a set amount.
data structure	A data type which groups together simpler abstract/primitive data types into a new, more complex abstract data type.
elastic analysis	Analysis in which the material follows a linear stress-strain relationship.
isotropic	Material exhibiting elastic behaviour in all directions.
k_error	User-specified parameter which controls the degree of equilibrium achieved in an iterative analysis. The smaller the error, the closer the beam-soil system will be to equilibrium.
linked list	Abstract data type in which each element is connected to its successor by a logical 'link'. The link is achieved through the use of a *pointer*.
matrix reduction	Process used to remove zero-valued boundary conditions.
midpoint displacement	Displacement associated with the midpoint of a beam element. This displacement is derived by averaging the displacements at the left and right nodal ends of an element.
node	Points in a beam which define the ends of beam elements. For a beam of N elements, there will be $N+1$ nodes.

outfile	Term derived from '**out**put **file**', meaning the data file to which computer program output will be written for permanent storage.
plate load test	Field test in which the load-settlement behaviour of a soil is determined.
pointer	Computer program variable which contains the memory location (address) of a particular data structure/primitive data type.
stiffness matrix	Symmetric matrix of terms describing an element's stiffness relative to bending moments and shear forces applied to its end points.
stress redistribution	Process of redistribution of stress in a material, due to local yielding, away from points of maximum local stress.
subgrade modulus	A conceptual relationship between a pressure applied to the surface of the soil (subgrade) and displacement of the subgrade material.
UDL	Uniformly distributed load ($F.L^{-1}$) acting on a beam.

LIST OF SYMBOLS

BEAM

b - width of beam
L - length of beam
E - beam modulus
I - moment of inertia
G - shear modulus
$[k]_e$ - element stiffness matrix
$[k]_G$ - global stiffness matrix for the beam
V - vertical displacement of the end of a beam element
θ - rotational displacement of the end of a beam element
M - internal moment at the end of a beam element, or
- externally applied moment to the end of a beam element
S - internal shear force at the end of a beam element
U - global displacement vector
F - global load vector
P - applied concentrated load (F)
w,W - applied distributed load ($F.L^{-1}$)
δ - beam element displacement in direction of support
δm - displacement at the midpoint of a beam element

SOIL

q - pressure ($F.L^{-2}$)
d - soil displacement
k_0 - subgrade modulus ($F.L^{-3}$)
k - subgrade modulus ($F.L^{-2}$)
γ - dry soil density
σ - vertical soil stress
d_i - combination depth for two layer soil system
z_i - depth to soil interface between two soil layers
q_U - pressure response from upper soil layer (2 layer system)
q_L - pressure response from lower soil layer (2 layer system)

OTHER

K - unit of computer storage, equal to 1024 bytes (or 8192 bits).
∇ - the Laplace operator
T - constant tension field

CHAPTER 1

INTRODUCTION

Computer-aided engineering is becoming increasingly commonplace, and civil engineering research is endeavouring to find ways of incorporating existing design methods in computer applications. In geotechnical engineering, computer-based matrix methods for structural analysis can be developed for analysis and design of shallow foundations (continuous footings, plates), deep foundations (piles and piers), lateral support systems (sheet and soldier pile walls), buried structures and pavement systems. The *matrix* method of structural analysis is popular with computer programmers since matrix techniques are fairly simple to implement on today's desktop computers. The popular *finite element* method, for instance, is founded on matrix methods.

The aim of this research work is the development of a computer-based tool for beam-soil interaction modelling, which can be used by the engineering profession in the design of respective problems, for example in the design of a lateral support system.

Beam-on-soil foundation problems have been widely researched. However, in the relevant literature and daily design practice, there is little evidence of practically-oriented solution methods. Generally, these solutions go directly to the mathematics of the problem and usually attempt to present results in terms of influence values, tabulated or charted, so that the designer can, in reasonable time, determine a solution.

This thesis extends the existing theory of beam on elastic foundations to allow modelling of inelastic soil and beam behaviour. The extensions focus on the modelling of realistic behaviour for both the beam and the soil, while attempts are made to prepare the solutions in such a way that they are readily accessible to the designer in real-time.

A specific application of the research is undertaken to investigate the appropriateness of the developed method in the analysis and design of soldier pile support systems in deep excavations. At present, such systems are designed using widely divergent methods. The application of Terzaghi and Peck's (1967) earth pressure diagrams results in uneconomical, even wasteful pile sections. This has led many designers to use beam on elastic foundation methods to assess bending moments and shear forces in the piles. In recent years, further savings have been achieved by using plastic design methods. Development of a computer program capable of non-linear modelling and analysis of non-linear soil and pile (beam) behaviour will provide the engineer in the field with a highly advanced design tool.

Below is a chapter by chapter summary of the work presented in this thesis:

Chapter 2 gives an overview of several prominent foundation models which have been proposed by past researchers. The Winkler model is explained in greater detail since it is the model on which the theoretical work of this research is based. Other models described are : Pasternak and Filonenko-Borodich, and Ohde's half-space model.

Chapter 3 is a summary of the technique of structural analysis using stiffness matrices. Since the topic is well known, only a limited discussion on issues pertinent to solving simple beam problems is presented. A worked example is included to illustrate the theory.

Chapter 4 presents the stiffness matrix for the Winkler foundation model. The respective various extensions of the stiffness matrix method of structural analysis to include a soil foundation beneath the beam (assuming the foundation characteristics are those of the Winkler model) are pointed out. Numerical solution techniques for a non-linear analysis, using stiffness matrices, are explained.

Chapter 5 describes how the solution method for horizontal beam-on-soil problems can be applied to the soldier pile support model. Modifications to the model, necessary for an improved simulation of soldier pile foundation conditions, are also presented.

Chapter 6 covers the basic routines that would be required in a computer implementation of the theory of the preceding chapters. Variables and algorithms are explained in as general a manner as possible. The chapter is intended as a reference for future researchers wishing to extend the methods of this work.

Chapter 7 provides a step by step description of the program 'BSF' (Beam on Soil Foundation) which is a computer implementation of the theory from the above chapters. Detailed guidance for the operation of the program is given with the aid of an example problem.

Chapter 8 assesses the suitability of the analysis method in predicting the beam-soil interactions for a beam on a horizontal foundation and a soldier pile support system (vertical beam on vertical foundation). The computer predictions are compared with actual laboratory and field tests.

The thesis concludes with the discussion of the effectiveness of the theory and computer implementation in achieving greater accuracy and efficiency in the analysis of beams on soil foundations in the various configurations of this research work.

Appendix A is an influence chart for determining the vertical stress in an elastic, isotropic half-space beneath a line load applied at the surface.

CHAPTER 2

THEORETICAL FOUNDATION MODELS

Basic analysis of steel or concrete structures supported by an elastic foundation (or subgrade) comprises the determination of stresses and displacements resulting from the imposition of loads upon the structure. Various theoretical foundation models have been proposed by researchers for the purpose of simulating beam-soil interaction behaviour in numerical methods for analysis and design purposes. A selection of well-documented models is briefly presented below.

2.1 WINKLER FOUNDATION MODEL

The essential feature in the analysis of beams supported by a deformable medium (foundation) is the presence of reactive forces from the foundation. The magnitudes of these forces are related to the deformations of the supporting medium. Generally, the supporting medium is considered to be soil.

The functional relationship existing between the reactive force and the deformation of the soil may be of many different forms. The simplest form is known as 'Winkler's hypothesis' (1867), the principle characteristics of which are:

- the reactive force at any point is dependent on the deformation at that point only, and
- the magnitude of the reactive force is directly proportional to the deformation.

Thus the foundation is modelled as a system of many closely spaced, independent springs with a linear force-deformation behaviour. Further assumptions related to Winkler's model are:

- there is no friction between the structure and soil at the interface, and
- the interaction between the soil and the structure exists even under *negative* deformations (ie where the beam tends to separate off the soil).

Hetenyi (1946) notes that Winkler's model is too simplistic to accurately model the behaviour of soil. As such, the application of theory relating to beams on elastic soil foundations making use of Winkler's model can only be considered as a practical approximation. However, he agrees that the spring model is appropriate for many soils given their elastic nature.

The deflection of the beam into the soil medium will produce reaction forces in the supporting medium (the reaction coming from the springs beneath the beam). Following the assumptions of the Winkler model, the intensity of the foundation reaction force, P, at any

point is proportional to the deflection of the beam, δ, at that point.

$$P = k\delta \tag{2.1}$$

This implies that the supporting medium is elastic, and thus follows Hooke's law. The concept is illustrated in figure 2.1.

The reaction forces are taken to act vertically, and oppose the beam deflection. Hetenyi, following the second above-mentioned assumption relating to Winkler's model, considered the supporting medium as being in a state of tension if the beam lifts upwards and tries to separate from the soil. This fictitious tension keeps the beam in contact with the soil to comply with Winkler's hypothesis.

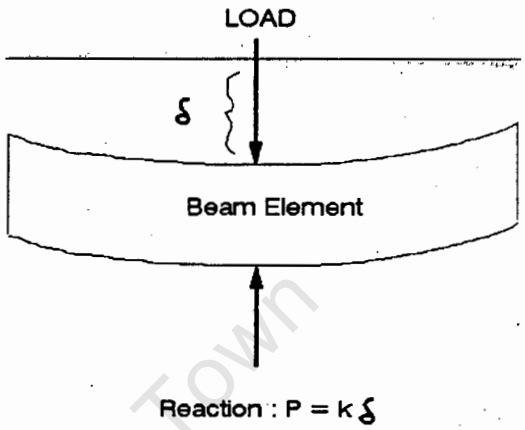


Figure 2.1 : Schematic of foundation response of Winkler model

The compressibility of the subgrade in Winkler's model is characterised by the *coefficient of subgrade reaction*, k_0 (also known as the *subgrade modulus*). By definition, k_0 is the pressure which, when applied to the surface of the subgrade, will cause a unit displacement of the subgrade material.

$$k_0 = Force \;/\; Area \;/\; Deflection$$
$$= N/m^3$$

Where a portion of a beam (called a beam *element*) of width b, deflects under loading by an amount δ, the associated foundation reaction, P, is calculated as

$$P = bk_0\delta \tag{2.2}$$

Winkler's model is widely used in the analysis of linear elastic foundations and numerous investigators (Hayashi, 1921; Hetenyi, 1946 and others) have presented numerical solutions to the differential equation for the deflection of a beam on elastic subgrade with a variety of boundary conditions:

$$EI\frac{\delta^4 y}{\delta x^4} + ky = q \tag{2.3}$$

where

$$k = bk_0 \tag{2.4}$$

with

E = Young's Modulus of the beam material
I = Moment of inertia for the beam
x = distance measured along the beam
y = deflection of the beam in the vertical direction

2.2 PASTERNAK FOUNDATION MODEL

Extending Winkler's model, Pasternak (1954) assumed the existence of shear interactions between the spring elements. This is achieved by connecting the springs to an incompressible plate as shown in figure 2.2. The plate only deforms under transverse shear. The foundation material is considered homogeneous and isotropic, hence $G_x = G_y = G$ (where G is the shear modulus).

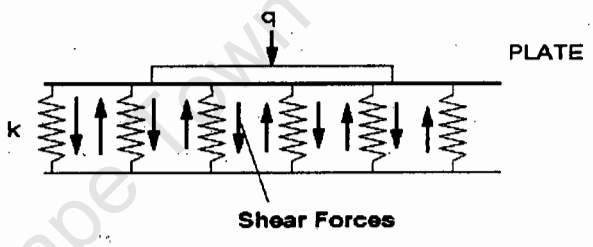


Figure 2.2 : Schematic of Pasternak foundation model

Therefore,

$$q = ky - G\nabla^2 y \tag{2.5}$$

where ∇^2 is the Laplace operator.

2.3 FILONENKO-BORODICH FOUNDATION MODEL

Filonenko-Borodich (1940) assumed the top ends of the springs are connected to a stretched elastic membrane subjected to a constant tension field, *T*, as shown in figure 2.3. This membrane allows for a degree of interaction between the spring elements by forming a common connection between them.

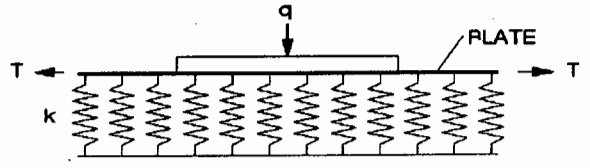


Figure 2.3 : Filonenko-Borodich foundation model

The condition of equilibrium of a membrane element yields the load-displacement relation

$$q = ky - T\nabla^2 y \qquad (2.6)$$

From equation (2.6) it can be seen that the intensity of T characterises the interaction of the spring elements.

2.4 OHDE HALF-SPACE FOUNDATION MODEL

It is clear that the Pasternak and Filonenko-Borodich models do not essentially improve the unrealistic 'spring approach' adopted in Winkler's model. For the studies of deflections and stresses in railroad tracks and ties, Zimmermann (1930) applied the Winkler method successfully. However, for foundation analysis, a more accurate method was developed by Ohde (1942).

Ohde's model considers the foundation as an elastic isotropic halfspace. The compressibility of the soil is characterised by its coefficient of volume change, the shape of the beam foundation, and the magnitude of loading. In a conventional settlement analysis, a realistically shaped settlement curve for a unit pressure influence line is evaluated. In a second step of analysis, variation of the foundation pressure is carried out in such a way that the settlement curve and beam deflection are identical. Thus the unknown foundation pressure is identified and stresses and moment in the beam are determined. For the purposes of accuracy, the beam is generally subdivided into 10 elements. However, for ease in applications involving nondimensional solutions, solution tables are prepared. Kany (1974) expanded the method for stratified soil systems or soil systems where the Young's Modulus increases linearly with soil depth.

2.5 MODEL CHOSEN FOR THIS THESIS

This research thesis adopted the Winkler foundation model for defining beam-soil interaction behaviour because of its popular acceptance and widespread use, even today. The fact that Eisenberger et al (1985) developed a stiffness matrix from equation (2.3) for use in the matrix method of structural analysis (as will be discussed in chapters 3 and 4) was of significant importance in this work.

CHAPTER 3

SUMMARY OF STIFFNESS METHOD IN STRUCTURAL ANALYSIS

Many books have been written on the subject of analysing structures with the use of matrix methods. An early reference on the subject may be found in Gere el al (1965). This chapter outlines the stiffness matrix method where it pertains to the solving of horizontal beam problems, given that such problems are fundamental to the beam-soil structures in this thesis. Here, the beam is not supported by any foundation and horizontal forces and displacements are not considered.

3.1 BEAM ELEMENT STIFFNESS MATRIX AND ASSOCIATED VECTORS

Consider a beam section IJ, as shown in figure 3.1, which is in equilibrium under some loading (not shown).

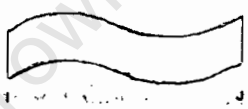


Figure 3.1 : Element IJ

Distributed and concentrated loads are assumed to act only in the transverse direction (thus axial displacements are zero). If no horizontal loads on the beam are permitted, then the ends of the section will be displaced amounts v_I, θ_I, v_J, θ_J, defined in the beam's local coordinate system as shown in figure 3.2.

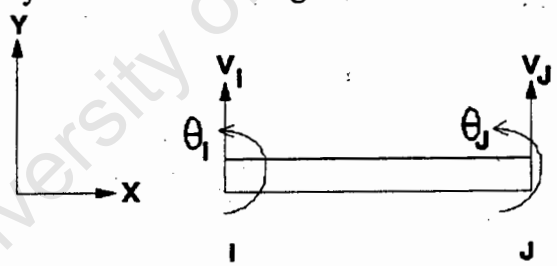


Figure 3.2 : Shear and bending displacements

Therefore, there are two degrees of freedom at each end of an element. The corresponding forces and moments at the ends of element IJ are S_I, S_J, M_I, and M_J, as shown in figure 3.3.

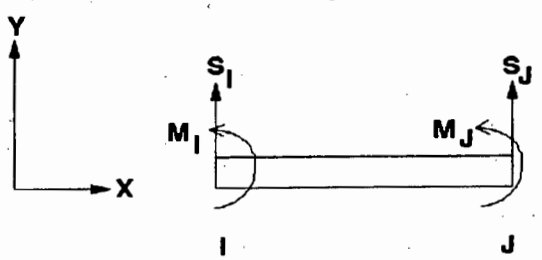


Figure 3.3 : Internal shear forces and moments at the ends of the element IJ

In this thesis, the sign convention for bending moments and shear forces is given in figure 3.4.

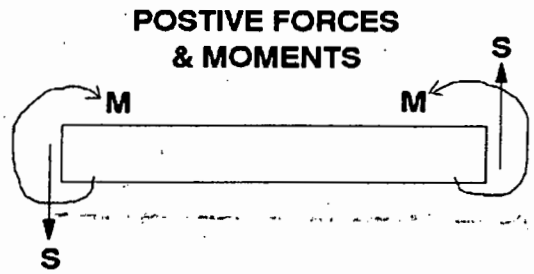


Figure 3.4 : Sign convention for moments and shear forces

The *displacement vector*, u_e, and *load vector*, f_e, for the element IJ are defined as

$$u_e = \begin{bmatrix} v_I \\ \theta_I \\ v_J \\ \theta_J \end{bmatrix} \qquad f_e = \begin{bmatrix} P_{IY} \\ M_I \\ P_{JY} \\ M_J \end{bmatrix} \tag{3.1}$$

The relationship between the end displacements (linear and rotational movements) and the corresponding forces for any beam element can be given in matrix form as

$$[k]_e u_e = f_e \tag{3.2}$$

where $[k]_e$ is termed the beam *element stiffness matrix* (in local coordinates). The terms for the matrix $[k]_e$ are

$$[k]_e = \begin{bmatrix} \frac{12EI}{L^3} & \frac{6EI}{L^2} & \frac{-12EI}{L^3} & \frac{6EI}{L^2} \\ \frac{6EI}{L^2} & \frac{4EI}{L} & \frac{-6EI}{L^2} & \frac{2EI}{L} \\ \frac{-12EI}{L^3} & \frac{-6EI}{L^2} & \frac{12EI}{L^3} & \frac{-6EI}{L^2} \\ \frac{6EI}{L^2} & \frac{2EI}{L} & \frac{-6EI}{L^2} & \frac{4EI}{L} \end{bmatrix} \tag{3.3}$$

where L is the length of the element, E its elastic modulus, and I the moment of interia. These terms are derived from considerations of unit translations and rotations at the end of

a beam element.

In solving a beam problem the whole member is divided into logical elements. The ends of elements are taken at

- points of concentrated load application (point loads)
- end points of distributed loads
- hinges within the beam
- changes in beam cross-section or material properties
- points of physical support

Therefore, in the example of figure 3.5, the beam AF would be subdivided into the elements AB, BC, CD, DE, EF.

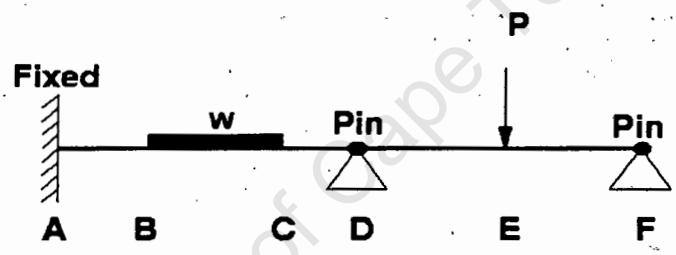


Figure 3.5 : Beam AF with points of appropriate subdivision into elements

The known displacements for beam AF are

$$v_A = \theta_A = v_D = v_F = 0 \qquad (3.4)$$

since the fixed support at A rules out any displacement and rotation of the left end of element AB; and the pinned supports at D and F prohibit any vertical displacement of elements CD, DE and EF at the support points D and F.

The unkown displacements for AF are : v_B, θ_B, v_C, θ_C, θ_D, v_E, θ_E, θ_F.

The ends of elements are termed *nodes*. Therefore, in figure 3.5, the nodal points are A, B, C, D, E and F. When working with elements, a *hinge* is generally defined as a very short element with a low elastic modulus and moment of inertia.

The nature of the unkown displacements at any node depends on the type of beam support, if any, at that node. The possible nodes and associated displacements for a beam with two degrees of freedom are given in figure 3.6.

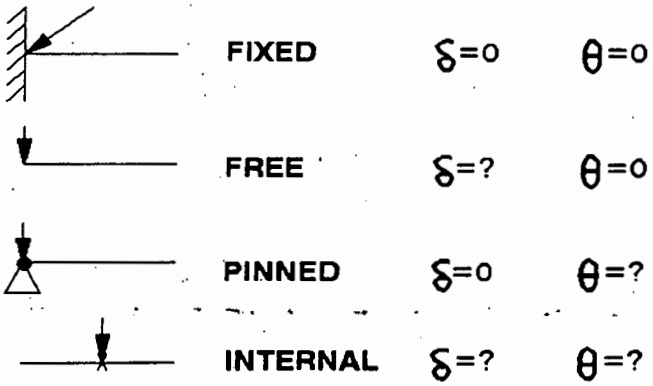


Figure 3.6 : Node conditions and associated displacements in a beam with 2 degrees of freedom

In order to assemble a global load vector, it is necessary to use the equivalent point load application since loads can only be considered if they act at element nodes. Thus a uniformly distributed load is conveniently split into two concentrated (point) forces and moments as shown in figure 3.7.

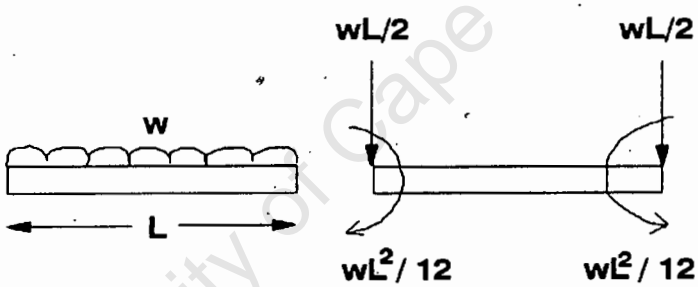


Figure 3.7 : Equivalent concentrated loading at end points for a distributed load *w*

3.2 GLOBAL STIFFNESS MATRIX AND ASSOCIATED VECTORS

Considering the entire beam, the element displacement vectors can be combined into a *global displacement vector*, *U*. Similarly, a *global load vector*, *F*, comprises the elements' load vectors. For a beam of *N* nodes

$$U = \begin{bmatrix} v_1 \\ \theta_1 \\ \vdots \\ v_N \\ \theta_N \end{bmatrix} \qquad F = \begin{bmatrix} P_1 \\ M_1 \\ \vdots \\ P_N \\ M_N \end{bmatrix} \tag{3.5}$$

where P_i and M_i refer to the applied point load and applied moment, respectively, at a given node i. In the case of beam AF of figure 3.4

$$U = \begin{bmatrix} v_A=0 \\ \theta_A=0 \\ v_B \\ \theta_B \\ v_C \\ \theta_C \\ v_D=0 \\ \theta_D \\ v_E \\ \theta_E \\ v_F=0 \\ \theta_F \end{bmatrix} \qquad F = \begin{bmatrix} P_A=0 \\ M_A=0 \\ P_B=-\alpha \\ M_B=-\beta \\ P_C=-\alpha \\ M_C=+\beta \\ P_D=0 \\ M_D=0 \\ P_E=-P \\ M_E=0 \\ P_F=0 \\ M_F=0 \end{bmatrix} \tag{3.6}$$

where

$$\alpha = \frac{wL}{2} \qquad \beta = \frac{wL^2}{12} \tag{3.7}$$

A *global stiffness matrix*, for the whole beam, is also composed of the element stiffness matrices. For readability, if the 16 terms in an element stiffness matrix are denoted

$$[k]_e = \begin{bmatrix} a & b & -a & b \\ b & c & -b & d \\ -a & -b & a & -b \\ b & d & -b & c \end{bmatrix} \tag{3.8}$$

where

$$a = \frac{12EI}{L^3} \quad ; \quad b = \frac{6EI}{L^2} \quad ;$$

$$c = \frac{4EI}{L} \quad ; \quad d = \frac{2EI}{L} \tag{3.9}$$

then global stiffness matrix $[K]_G$ for a beam of N nodes is assembled as

$$[K]_G = \begin{bmatrix} a_1 & b_1 & -a_1 & b_1 & & & & \\ b_1 & c_1 & -b_1 & d_1 & & & & \\ -a_1 & -b_1 & a_1+a_2 & -b_1+b_2 & -a_2 & b_2 & & \\ b_1 & d_1 & -b_1+b_2 & c_1+c_2 & -b_2 & d_2 & & \\ & & -a_2 & -b_2 & a_2+a_3 & -b_2+b_3 & & \\ & & b_2 & d_2 & -b_2+b_3 & c_2+c_3 & & \\ & & & & & & \ddots & \\ & & & & & & a_N & -b_N \\ & & & & & & -b_N & c_N \end{bmatrix} \tag{3.10}$$

where the columns and rows are tied to the associated displacements as shown in figure 3.8, and the subscripts identify the element stiffness matrices from which the terms are taken.

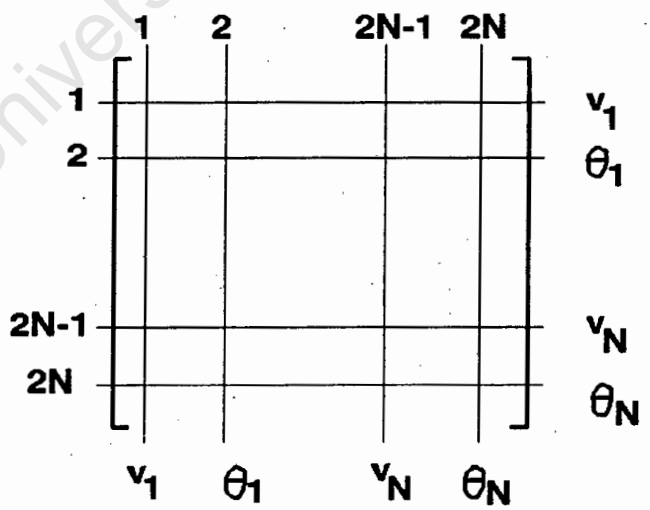


Figure 3.8 : Column-row displacement association

3.3 SOLVING FOR UNKOWN DISPLACEMENTS

Where a displacement, v_i, or θ_i, is known to be zero, the associated row and column can be removed from $[K]_G$. This *matrix reduction* for beam AF of figure 3.4 will lead to the modified global matrix:

$$[K]_G = \begin{bmatrix} & & & & & & & \\ & & & & & & & \\ & & & & & & & \\ & & & & & & & \\ & & & & & & & \\ & & & & & & & \\ & & & & & & & \\ & & & & & & & \end{bmatrix} \begin{matrix} v_B \\ \theta_B \\ v_C \\ \theta_C \\ \theta_D \\ v_E \\ \theta_E \\ \theta_F \end{matrix} \qquad (3.11)$$

$$v_B \quad \theta_B \quad v_C \quad \theta_C \quad \theta_D \quad v_E \quad \theta_E \quad \theta_F$$

The associated elements in the global vectors U and F must also be removed to retain compatibility. The unkown nodal displacements of the beam elements are determined by solving the following equation

$$[K]_G U = F \qquad (3.12)$$

using common solution methods (for example the Gauss reduction method).

3.4 BENDING MOMENTS AND SHEAR FORCES

To calculate the bending moments and shear forces for the beam, each element stiffness matrix, $[k]_e$, is multiplied by the associated displacement values solved in equation (3.12). For an element IJ, the shear force, S_{IJ} and bending moment, M_{IJ}, at its ends are determined by

$$\begin{bmatrix} S_I \\ M_I \\ S_J \\ M_J \end{bmatrix} = [k]_{e_{IJ}} u_{e_{IJ}} \qquad (3.13)$$

If there is a uniformly distributed load acting on the element, then the external moments and shear forces must be removed to arrive at the correct internal values. Applied moments and shear forces for such a load are given in figure 3.7. Therefore, these distributed loads would

have to be removed taking the sign convention (figure 3.4) into account. According to this convention, the final internal forces for element IJ are

$$\begin{bmatrix} S_I \\ M_I \\ S_J \\ M_J \end{bmatrix} = \begin{bmatrix} S_I - \alpha \\ M_I - \beta \\ S_J + \alpha \\ M_J - \beta \end{bmatrix} \tag{3.14}$$

where

$$\alpha = \frac{w_{IJ} L_{IJ}}{2} \qquad \beta = \frac{w_{IJ} L_{IJ}^2}{12} \tag{3.15}$$

Now the bending moment and shear force diagrams for the beam can be drawn and the internal forces evaluated for each element in the beam. A worked example is given below to demonstrate the method presented in this chapter.

3.5 WORKED EXAMPLE

The beam is

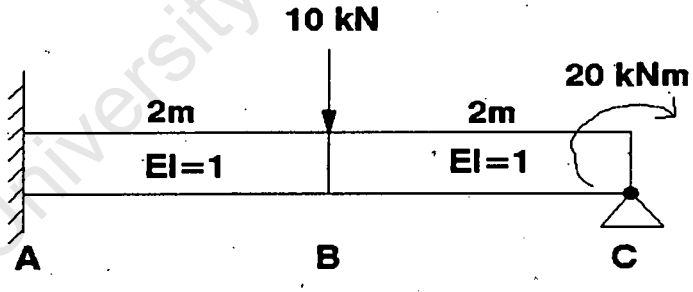


Figure 3.9 : Example problem

Therefore

$$U = \begin{bmatrix} v_B \\ \theta_B \\ \theta_C \end{bmatrix} \qquad F = \begin{bmatrix} -10 \\ 0 \\ 20 \end{bmatrix}$$

$$[K]_G = \begin{bmatrix} 3 & 0 & 1.5 \\ 0 & 4 & 1 \\ 1.5 & 1 & 2 \end{bmatrix} \tag{3.16}$$

Solving equation (3.12) for the values of equations (3.16) yields

$$\begin{bmatrix} v_B \\ \theta_B \\ \theta_C \end{bmatrix} = \begin{bmatrix} 4.17 \\ 3.75 \\ -15.00 \end{bmatrix} \tag{3.17}$$

To solve for moments and shear forces

$$\begin{bmatrix} S_A \\ M_A \\ S_B \\ M_B \end{bmatrix} = [k]_{e_{AB}} \begin{bmatrix} 0 \\ 0 \\ 4.17 \\ 3.75 \end{bmatrix} = \begin{bmatrix} -0.63 \\ -2.50 \\ 0.63 \\ 1.25 \end{bmatrix} \tag{3.18}$$

Similarly

$$\begin{bmatrix} S_B \\ M_B \\ S_C \\ M_C \end{bmatrix} = \begin{bmatrix} -10.63 \\ -1.25 \\ 10.63 \\ -20.00 \end{bmatrix} \tag{3.19}$$

The resulting bending moment and shear force diagrams are given in figure 3.10.

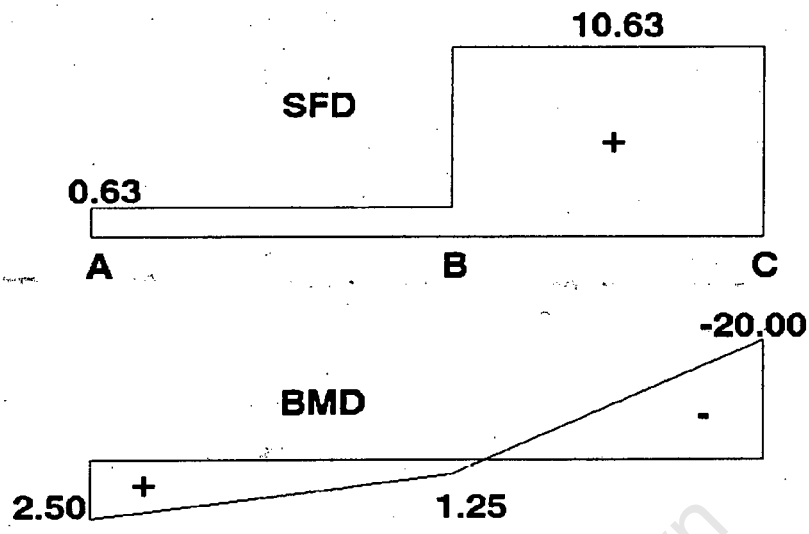


Figure 3.10 : Bending moment (BMD) and shear force (SFD) diagrams of beam example

CHAPTER 4

STIFFNESS MATRIX FOR WINKLER FOUNDATION

In chapter 3, the stiffness method for solving displacements, bending moments and shear forces for a horizontal beam was presented (where there was no foundation support beneath the beam). This chapter shows the derivation of a stiffness matrix for beams resting on an elastic Winkler soil foundation as described in Eisenberger et al (1985).

4.1 STIFFNESS MATRIX FOR BEAM ON ELASTIC WINKLER FOUNDATION

The general solution of the differential equation for the deflection of a beam on elastic foundation (equation 2.3 in chapter 2) can be written in terms of four functions, using the sign convention as given in figure 4.1.

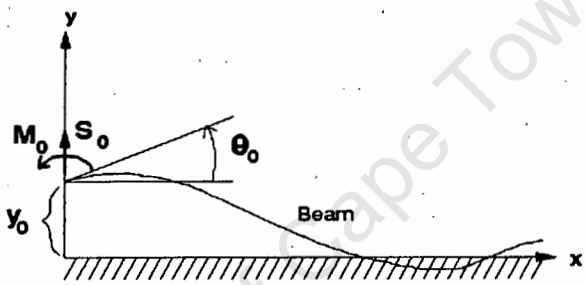


Figure 4.1 : Sign convention for beam on elastic foundation

$$y(x) = y_0 F_1(\lambda x) + \frac{1}{\lambda}\theta_0 F_2(\lambda x) - \frac{1}{\lambda^2 EI} M_0 F_3(\lambda x) + \frac{1}{\lambda^3 EI} S_0 F_4(\lambda x) \qquad (4.1)$$

where

$$\lambda = \left(\frac{k}{4EI}\right)^{0.25}$$

$$\alpha = x\,\lambda$$

$$F_1(\alpha) = \cosh(\alpha)\cos(\alpha)$$

$$F_2(\alpha) = \frac{1}{2}(\cosh(\alpha)\sin(\alpha) + \sinh(\alpha)\cos(\alpha)) \qquad (4.2)$$

$$F_3(\alpha) = \frac{1}{2}\sinh(\alpha)\sin(\alpha)$$

$$F_4(\alpha) = \frac{1}{4}(\cosh(\alpha)\sin(\alpha) - \sinh(\alpha)\cos(\alpha))$$

and y_0, θ_0, M_0, and S_0 are the values at x = 0.

The 16 terms for the stiffness matrix are derived from equation (4.1) and its derivative defining the slope along the beam. In keeping with the principle of the element equation compilation and its respective stiffness matrix, the terms relate to unit translations and rotations at the end of a beam element. The matrix is

$$[k]_e = \begin{bmatrix} a & b & c & d \\ b & e & -d & f \\ c & -d & a & -b \\ d & f & -b & e \end{bmatrix} \tag{4.3}$$

where

$$\beta = \sinh^2(\lambda L) - \sin^2(\lambda L)$$

$$a = \frac{k}{\lambda\beta}[\cosh(\lambda L)\sinh(\lambda L) + \cos(\lambda L)\sin(\lambda L)]$$

$$b = \frac{k}{2\lambda^2\beta}[\sinh^2(\lambda L) + \sin^2(\lambda L)]$$

$$c = -\frac{k}{\lambda\beta}[\sinh(\lambda L)\cos(\lambda L) + \cosh(\lambda L)\sin(\lambda L)]$$

$$d = \frac{k}{\lambda^2\beta}[\sinh(\lambda L)\sin(\lambda L)]$$

$$e = \frac{k}{2\lambda^3\beta}[\sinh(\lambda L)\cosh(\lambda L) - \sin(\lambda L)\cos(\lambda L)]$$

$$f = \frac{k}{2\lambda^3\beta}[\cosh(\lambda L)\sin(\lambda L) - \sinh(\lambda L)\cos(\lambda L)]$$

and L is the length of the beam element.

This stiffness matrix can be incorporated into the matrix method for analysing beams as presented in chapter 3. The matrix is applicable to elastic foundation behaviour only (in keeping with Winkler's hypothesis of linear soil response). The following sections describe iterative techniques used to allow for non-linear response from the soil foundation and beam material. In addition, consideration will be given to limitation of inelastic response by modelling plastic hinging in the beam.

4.2 NON-LINEAR SOIL BEHAVIOUR

To model non-linear soil behaviour using the stiffness matrix for an elastic Winkler foundation, Yankelevsky et al (1989) proposed an iterative analysis which utilises a theoretical piecewise linear force-displacement approximation curve for both compressive and tensile behaviour of the beam-soil interaction system.

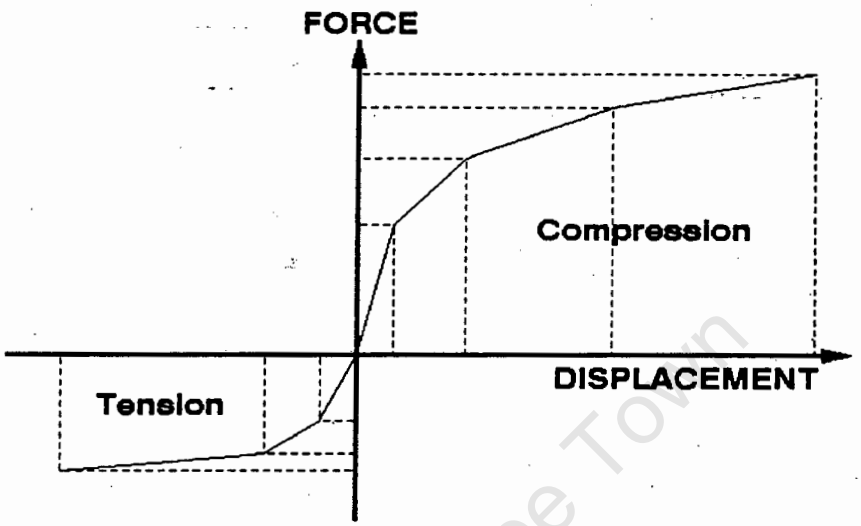


Figure 4.2 : Diagram of a piecewise linear force-displacement relationship for a soil

The parameter, k, (referred to by Yankelevsky as the 'foundation' modulus) associated with a beam element is dependent on the displacement of that element. This parameter has no connection whatsoever with the subgrade modulus, k_s, from geotechnical engineering - it is simply the spring coefficient fundamental to Winkler's foundation model. The modulus is determined by referring to the displacement interval of the piecewise curve and taking k (kN/m) to be the slope of the curve for that interval. The need for the inclusion of a tensile contact pressure arises from Winkler's assumption that the beam and soil remain in contact even if the beam lifts off the soil (see chapter 2).

Employing the conventional limit equilibrium approach, an initial analysis (first iteration) is performed assuming a constant foundation modulus for each element. The resulting deflections identify the portions of the foundation in tension or compression. The appropriate foundation modulus for each portion is assessed to determine if it is compatible with the displacement of that portion of the beam. If there is a region where the beam has displaced by an amount incompatible with the stiffness assigned to it, that region is subdivided at the points of incompatibility (transition points) and each subdivision assigned an appropriate stiffness value. A new stiffness matrix and load vector are formed and another solution obtained. The iterative process continues until the locations of transition points do not change by more than a predetermined amount (specified by the user).

While this non-linear, iterative approach of evaluating the foundation modulus relates to a rather theoretical force-displacement relationship for the soil, a more direct assessment of the subgrade reaction upon loading can be incorporated which greatly simplifies the analytical procedure. Making use of the relationship between soil pressure and displacement as

established in a plate load test, a common field test to determine the compressibility of the soil (and thus modulus of subgrade reaction), the relevant stiffness can be determined directly for any given element displacement without the need for a piecewise linear approximation. From figure 4.3,

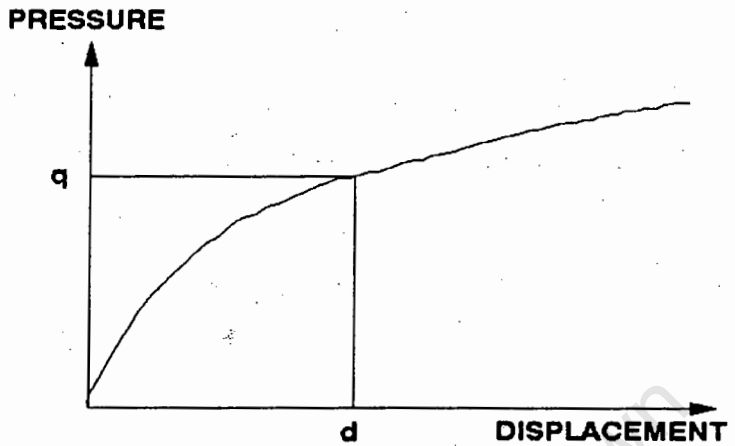


Figure 4.3 : Pressure-displacement relationship of a plate load test

given pressure, q, and displacement, d, the subgrade modulus, is

$$k = \frac{q}{d} \times b \qquad (4.4)$$

where b is the width of the beam. Hereafter, the *initial* subgrade modulus will be referred to as the *elastic* subgrade modulus since a piecewise approximation of a plate load curve is used for simplification purposes, and the first portion of such a curve can be considered as an elastic behaviour region. The use of the plate load curve is fundamental to the method of this thesis, and provides a strong practical input to the analytical modelling of beam-soil problems (unlike Yankelevsky's approach).

To implement this technique into the matrix method of structural analysis, the following iterative method is used:

1. The plate load curve is approximated using a linear piecewise curve.

2. The first iteration is performed assigning the initial or elastic subgrade modulus to each beam element.

3. Having solved equation (3.12), the following 5 steps are considered for each beam element:

   3.1 The displacements at both ends of the element are averaged to give a *midpoint displacement*.

   3.2 For a displacement less than 0, the element has lifted off the soil, and the elastic subgrade modulus is assigned to it, if its present value is not that already.

Otherwise, for a displacement greater than 0, steps 3.3 to 3.5 are followed:

3.3 With reference to the curve of step 1, the pressure associated with the midpoint displacement is determined.
3.4 Using equation (4.4), the associated subgrade modulus is calculated.
3.5 Should this modulus differ from the one presently assigned to the element by an amount greater than a predetermined value, *k_error*, a new modulus is assigned to the element.

4. If any new modulus values were assigned to elements in either step 3.2 or 3.5 then another iteration is performed and the above procedure repeated starting from step 3.

5. If no such changes are made in step 3, then the beam and soil foundation are considered to be in equilibrium (within an error of *k_error*).

Note that no tension pressure-displacement curve is used since such a curve does not exist in reality. This overcomes the impractical condition in Yankelevsky's method which requires that a tension curve be available. A pressure-displacement response curve derived from a plate load test therefore allows realistic and practical modelling of soil behaviour.

Note that the decision made in step 3.2 above, to set k values to the elastic subgrade modulus when elements break away from the soil interface is based on the following observation:

If these elements had their k parameters set to 0 upon breaking away, then the stiffness matrix for those elements returns to the basic stiffness matrix for a normal beam element (equation 3.3) where there is no soil support considered. However, while this seems to be the correct approach to take, investigation of analyses revealed that these elements deflected unrealistically relative to those neighbouring elements in the beam which have positive k values (being in contact with the soil). These 'weak' sections of the beam led to very high local beam deflections and it was decided to reset the k values to the elastic modulus of the supporting soil. This is in keeping with Winkler's assumption that the beam never separates from the soil, and promotes a greater continuity of strength in those portions of the beam where some elements are in contact with the soil while others are not.

4.3 NON-LINEAR BEAM BEHAVIOUR

In a similar manner to that for modelling non-linear soil behaviour, the stiffness of the beam material can be adjusted for each element to reflect non-linear behaviour of the beam material by altering the beam's modulus in the stiffness matrix of the appropriate element. The stress-strain relationship for structural steel is generally known and a piecewise linear approximation of the stress-strain characteristics of the beam material can be established for the purpose of analysis.

During each iteration, in addition to the consideration of the soil response for each beam element, the stresses (in the beam flanges due to bending) at the ends of each element are evaluated. Reference is made to the piecewise linear stress-strain characteristic for the beam

material. These stresses, like the displacement values, are averaged to obtain a midpoint stress value for each element. An element's beam modulus, E, can then be adjusted, if necessary, by assigning the appropriate modulus as derived from the stress-strain curve. The next iteration will consequently make use of the updated beam modulus when assembling the stiffness matrix for the element.

Some structural steels, like Grade 300W steel (SABS 0162-1, 1992), have a bilinear stress-strain relationship, as shown in figure 4.4, which does not even require a piecewise approximation - the beam modulus for an element is either truly elastic (Young's Modulus), or the internal stress has exceeded yielding stress; this would lead to plastic hinging of the beam, which is described in the following section.

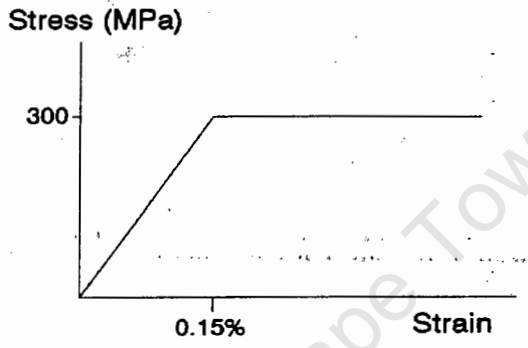


Figure 4.4 : Stress-strain relationship for Grade 300W steel

4.4 PLASTIC HINGES IN THE BEAM

When the element midpoint stress (calculated in section 4.3) has exceeded the yield stress for the beam material, plastic deformation has taken place. A plastic hinge is inserted at the appropriate end of the element for subsequent iterations.

As mentioned in chapter 3, a hinge can be modelled using a very short beam element, say 1mm long, with a low elastic modulus and possibly a low moment of inertia as well. In the diagram of figure 4.5 it is shown how, between 2 elements i and $i+1$ (of length $L1$ and $L2$ respectively), a hinge is inserted should the stress at the common node exceed the yield stress of the beam material. The length of the hinge is 1mm, and the elements have had their lengths adjusted accordingly to maintain the original beam length.

In this research work, beam hinges are assigned a length of 0.5mm with an elastic modulus of 25000 kPa. This modulus was chosen to ensure that magnitudes of certain numbers did not get too large during analysis where E is in the denominator of an expression (which might occur for values of E significantly lower than 25000 kPa). The moment of inertia is left unchanged, the same as that of the beam prior to hinging.

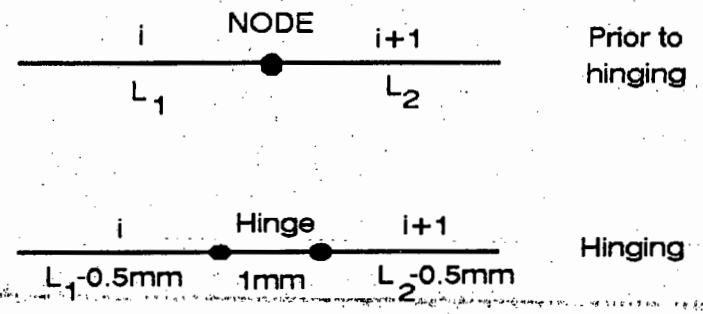


Figure 4.5 : Logical insertion of a hinge between 2 elements

CHAPTER 5

MODELLING OF A SOLDIER PILE SUPPORT SYSTEM

5.1 THE SOLDIER PILE MODEL

Soldier piles, in the context of lateral support systems, can be considered as the equivalent of a horizontal beam on a soil foundation rotated by 90°. The model, however, requires the following modifications:

- Due to an increase in overburden stress with depth, an increase in the soil modulus along the beam may be considered.

- The soldier piles, usually comprising one or two steel sections, are often installed in predrilled holes which are then backfilled with a low strength soil-cement mixture. The soil foundation thus consists of a two-layer system (backfill and in-situ material), each layer having different properties.

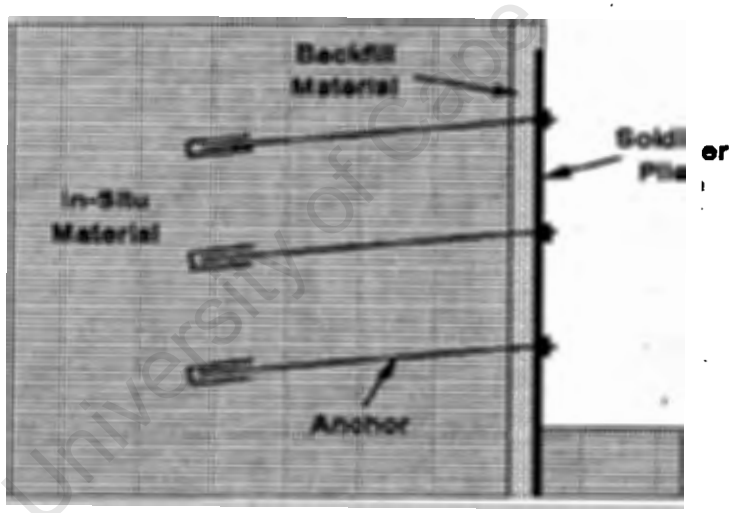


Figure 5.1 : Schematic showing section of a soldier pile lateral support system in elevation

The stiffness of the in-situ soil can readily be determined by means of plate load tests in which the test assembly and thus load is applied normal to the excavation face. However, the pressure displacement relationship of the backfill material is impractical to establish, and an engineering estimate of the relationship is required.

5.2 BEAM ON TWO-LAYERED SOIL FOUNDATION

In order to model the backfill and in-situ material, a two-layer soil foundation model must be developed. In a single-layer system, the foundation response is determined quite simply by referring to the pressure-displacement relationship (as observed in a plate load test) for the soil. For a soil foundation consisting of two layers, an initial beam deflection will cause the upper layer to deform and respond, with no deformation or response coming from the lower layer. Once the contact pressure between the beam and upper layer is sufficiently high to develop significant vertical stress in the lower layer, this layer will also deform and consequently respond forcefully. Therefore, to model a two-layered system, a means of differentiating between response from the upper layer and a combined response from both the layers is required.

The method adopted to model a two-layer system makes use of a pressure-displacement curve for both the soil layers assuming a *combination depth* to differentiate between a one- and two-layer response. When the beam deflects an amount less than this combination depth, only the upper layer responds, otherwise both layers will respond. The determination of this depth requires the following iterative procedure:

1. For any given beam and loading configuration (as in the example of figure 5.2), the beam is freed of all loads and supports so that no applied loads act on it, and the left and right end conditions are both made *free* (figure 5.3). This beam can thus be represented by a single beam element. The depth from the beam to the interface between the two soil layers is termed z_i ('i' for 'interface').

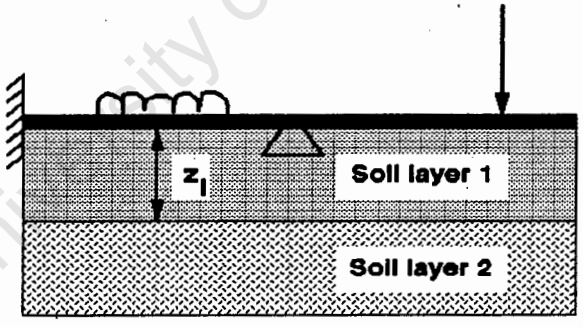


Figure 5.2 : Example problem with beam on a two-layered soil foundation

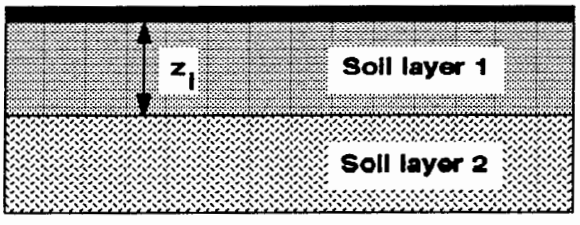


Figure 5.3 : Schematic of example problem with load and end conditions removed

2. A uniformly distributed load (UDL), equal to the beam's self weight, is applied to the beam.

3. A further uniformly distributed load of arbitrary magnitude, W (kN/m), is applied to the beam (as shown in figure 5.4).

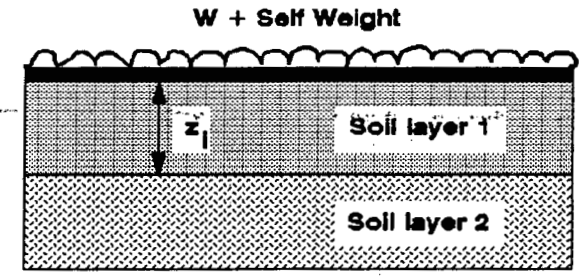


Figure 5.4 : Self weight and arbitrary line load applied to beam

A line load along the entire length of the beam has been chosen because it is the most general loading configuration, and the most easily defined (alternatively, one could use any combination of concentrated and distributed loads at this stage).

4. The single beam element is divided into, say, 30 elements (as shown in figure 5.5), and an iterative (inelastic) analysis of the beam resting on only the upper layer is carried out, ie. the presence of the lower layer material is ignored for the time being.

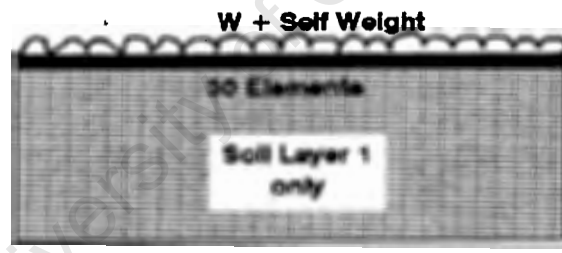


Figure 5.5 : Beam configuration for determination of the combination depth

5. The foundation response profile beneath the beam as a result of the applied UDL (W + self weight) is then calculated. Since the peak portion of the profile will always be slightly curved, the peak region is averaged out to a constant value acting beneath a central portion of the beam, of length L_{PEAK} as shown in figure 5.6.

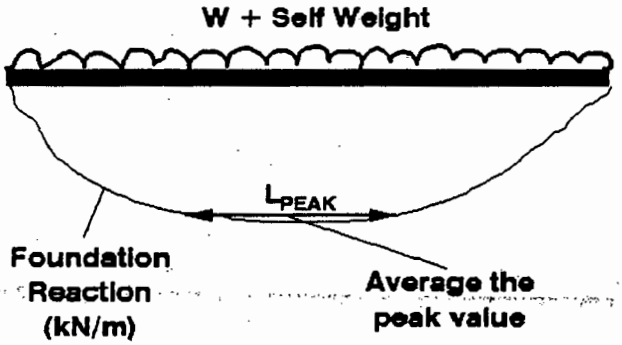


Figure 5.6 : Generalised profile of foundation response due to uniform loading

6. Using elastic half space considerations, the vertical stress σ_{zi} at depth z_i due to the peak foundation response acting over the length L_{PEAK} is determined (see figure 5.7). Here use can be made of the influence chart attached in Appendix A.

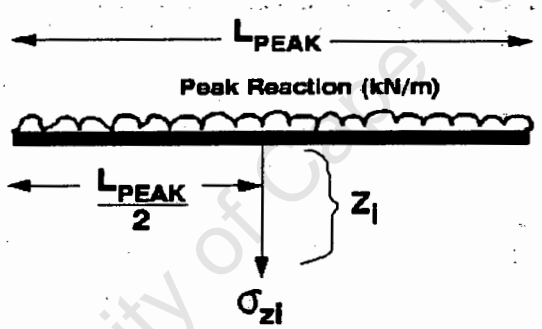


Figure 5.7 : Configuration for vertical stress determination at depth z_i

7. The amount of deflection associated with σ_{zi} that would occur in the second soil layer, assuming no discontinuity in response due to the layered system, is established by referring to the pressure-displacement response curve for the second soil layer.

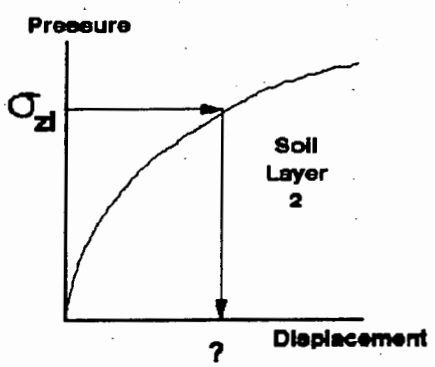


Figure 5.8 : Determination of displacement in second layer due to σ_{zi}

8. Upon changing the magnitude of W, steps 3 to 7 are repeated until the deflection obtained in step 7 is the minimum amount practically significant (say 0.5mm) in the problem under consideration.

9. The minimum value of W has then been found for which the second layer of soil would just respond. The maximum deflection of the beam under W (and the beam's self weight) when resting on the upper soil layer only is termed d_i, the *combination depth*. Since the load W is only used to generate displacement of the beam in this procedure, it is no longer needed for analytical purposes once d_i has been determined.

The advantage of using a single line load, of magnitude W, acting along the entire beam is that there is only one loading parameter to change with each iteration, namely W. If a different loading configuration had been used, say with a few concentrated loads and a UDL acting over some portion of the beam, there would be several load values to change with each iteration.

The combination depth is employed in the analysis as a measure to decide if one or two layers of the soil foundation are activated by (and thus responding to) beam deflection. During the iterative process of the non-linear analysis method, the soil modulus is checked for each beam element as before. If the *midpoint displacement* for an element, δ_m, is less than d_i, then only the response curve of the upper soil layer is required. However, if δ_m is greater than d_i, then an adjusted soil response from the lower layer, associated with a displacement δ_m-d_i, is added to the reaction from the upper layer.

An adjustment needs to be incorporated since the soil responses are actually located at different horizons, a distance z_i apart. In figure 5.9, a schematic for the responses of both soil layers for a displacement δ_m-d_i is given. The response from the upper soil layer, q_U, occurs at the beam-soil horizon, while the response from the lower layer, q_L, is at the horizon between the two soil layers. Given that these responses are spring reactions to the beam deflection (Winkler's model), the effect of the reaction from the lower horizon on the beam-soil horizon must reflect the attenuation of the response as it passes through the upper layer. This adjustment of q_L allows for the calculation of the combined response, q_U plus the adjusted q_L, to be incorporated in the iterative procedure achieve 'equilibrium' between loading on the beam and foundation reaction. If the magnitude of z_i is such that the attenuation of soil response from the lower horizon is practically insignificant, this adjustment procedure can be ignored. An example where the upper layer is relatively thin may be found in the following section.

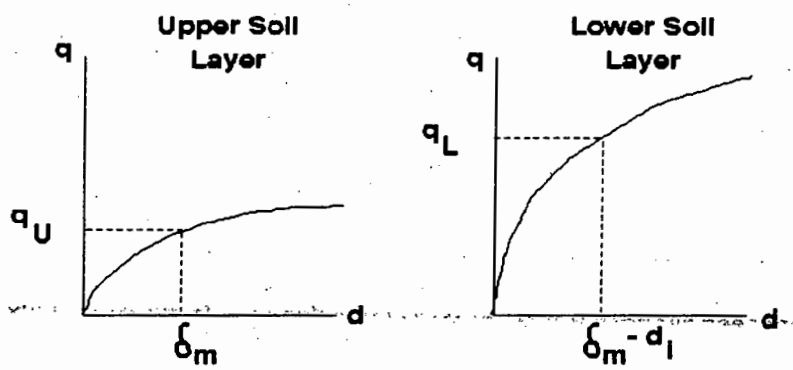


Figure 5.9 : Evaluation of combined foundation response when $\delta_m > d_i$

5.3 SOLDIER PILE ON TWO-LAYERED SYSTEM

In the case of the soldier pile model, the beam-on-soil-foundation model has been rotated by 90° so the soil *depth* is now the horizontal distance from the soldier into the soil medium. The depth of the backfill material, z_i, will have to be evaluated based on the dimensions of the piling equipment employed; this depth is generally very limited.

When using the described method to evaluate the combination depth between the backfill (upper layer) and in-situ (lower layer) material, the pressure-displacement response of the backfill material will have to be estimated while the response of the in-situ material will be determined in a plate load field test.

Loss of contact, or even separation, of soldier pile and excavation face is ruled out. The bottom end of the pile is generally embedded below the excavation level. The free top end of the pile resists the lateral earth pressure and only highly cohesive materials can at this end theoretically separate from the soldier beam. A tensile pressure-displacement relationship is therefore not considered in the non-linear beam-foundation analysis.

The matrix method and techniques of chapters 3 and 4 can be used to analyse soldier pile support systems. The soldier can be visualised as a horizontal beam and broken down into elements in the discussed manner. Since only two degrees of freedom are considered, the self-weight of the soldier becomes an axial force and will thus be ignored.

CHAPTER 6

COMPUTER IMPLEMENTATION OF ANALYSIS METHOD

In this chapter, the basic components and features constituted in the computer program BSF are described. This chapter is not a user manual - instead its purpose is to illustrate the basic algorithms which embody the principles of the previous chapters. Future researchers who wish to use/extend the features of BSF will, therefore, have a reference to the algorithmic structure of the key routines necessary for a computer program analysis of the work described in this thesis.

The algorithms and programming structures are presented using high-level abstraction characteristic of common imperative programming languages such as Pascal and C (the program was written using the C++ language, which is a superset of the C language). Since only the general concepts are covered, most of (if not all) the techniques used should be portable to a number of different programming platforms (for example, the Fortran language).

A brief description of the format used to present the various routines follows:

Arrays and matrices are assumed to start at index 0 (first value = *array_name*[0]) and not index 1 (first value = *array_name*[1]). The former syntax is used by the C language while the latter is characteristic of languages like Pascal and BASIC. For example, the array

```
number = array [4] of integers
```

would be represented by the integers:

```
number[0], number[1], number[2], number[3]
```

Comments within pseudo-code are contained within a pair of braces {}. To enhance readability, variables such as α and ß are used to break long statements into several shorter ones. For example:

```
sum = (3*x + 76*sin(value)) / ((2*x + 2*y) - cos(value))
```

might be broken up as

```
α = (3*x + 76*sin(value))
ß = (2*x + 2*y) - cos(value)
sum = α / ß
```

When a particular routine is to be invoked, and that routine requires a variable(s) to be passed to it, the format used will be:

```
routine_name ( variable_1, variable_2,... variable_N )
```

for example:

```
square_root ( 10 )
```

passes the value '10' to a routine called `square_root`. Similarly:

```
let x = 10
square_root ( x )
```

also passes the value '10' (by means of variable x) to the same routine.

The shorthand '+=' is taken from the C language where A += B is equivalent to A = A + B. Similarly, A -= B is equivalent to A = A - B. For multiplication and division, the operators '*=' and '/=' respectively, are also permissable.

6.1 GLOBAL VARIABLES AND DATA STRUCTURES

6.1.1 Element Information

The main data structure of the program is the abstract data type used to hold the information for beam elements. Borrowing terminology from the C language, this *structure* is called the `element structure`.

```
'element' structure
     real : length        (length of element)
     real : E             (beam modulus; usually Young's Modulus)
     real : I             (moment of inertia)
     real : depth         (depth of beam section)
     real : width         (width of beam section)
     real : k             (subgrade modulus)
     real : force_l       (shear at L end)
     real : force_r       (shear at R end)
     real : moment_l      (applied moment at L end)
     real : moment_r      (applied moment at R end)
     real : udl           (foundation response)
     integer : left_end   (code for L end condition)
     integer : right_end  (code for R end condition)
     element → : next     (pointer to next element)
end of 'element' structure
```

where *real* indicates a 'real number', and L and R represent the 'left' and 'right' ends of the element. Note that the symbol '*E*' was used to refer to an element's beam modulus since this will mostly be the elastic modulus of the beam material, hence *E*. To refer to a parameter within the structure, a '.' is used. For example, to set the value for the `udl` parameter of the 5th element of the beam to '101.5', the following syntax is used

```
element[5].udl = 101.5
```

When referring to elements, the syntax used in presenting algorithms will not start at index 0 (as is the case with arrays); instead it will follow the more readable format of `element[`*n*`].parameter` when referring to element *n*'s information (thus using an index

starting at '1').

The integer codes to indicate the *end conditions* are as follows

```
0     internal end
1     pinned end
2     fixed end
3     free end
```

The appropriate schematic of these end conditions and corresponding degrees of freedom is shown in figure 3.6. The *pointer* `element→next` is a variable which holds the address (location in memory) of the next logical element. Thus if a variable called `first_element` holds the address of the left-most beam element, then the entire beam can be held in memory as a chain of logical elements, each one connected to its neighbouring element on the right. A diagram of such a structure (called a *linked list*) is shown in figure 6.1. To signal the end of the list, the last 'next' link is set to 0 (called 'null').

If the program wanted to access the information of, say, the 3rd element in the beam, it would refer to the variable `first_element` to find the start of the list. From there is could follow the 'next' links until it arrived at the third element. It would then be at the right place in memory to directly access the information for this element.

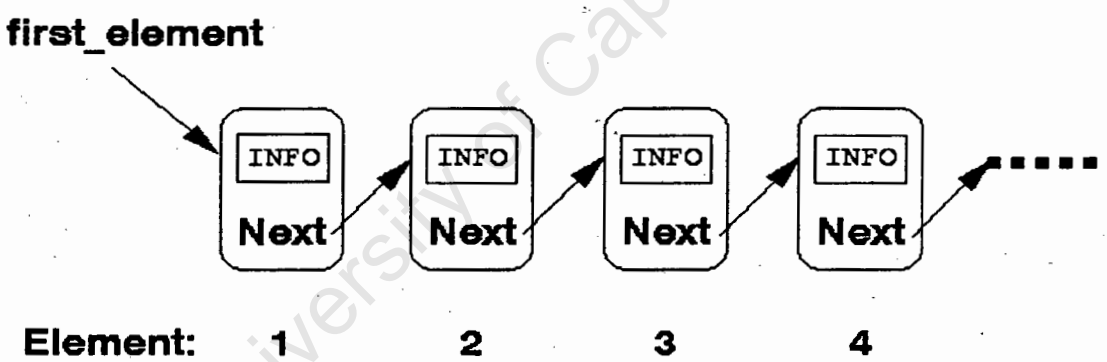


Figure 6.1 : Schematic of 'linked list' structure of beam element information

6.1.2 Beam Matrix

Assuming that `MAX_ELEMENTS` denotes the maximum number of elements that can define a beam as permitted by the program, there must be space in memory for `MAX_ELEMENTS+1` beam nodes (*n* elements have *n+1* nodes). Since each node has two degrees of freedom, v and θ, the matrix and arrays required for analysis must be dimensioned in the program to the array size

```
MAXSIZE = 2 * (MAX_ELEMENTS + 1)
```

The beam matrix is stored in the global 2-dimensional array

```
beam_matrix = matrix [MAXSIZE][MAXSIZE] of real numbers
```

which will hold the global stiffness matrix of equation (3.10). However, since the stiffness matrix for a beam element is symmetrical, significant saving in memory space required can be obtained by storing the beam matrix as follows:

```
beam_matrix = matrix [MAXSIZE][4] of real numbers
```

ie. the matrix has MAXSIZE rows, but only 4 columns. Less memory space also enables faster computations when performing an analysis because there is less information to have to interpret.

The compaction method for matrix information is shown in figure 6.2. Only 4 columns are required to store a single stiffness matrix (which is 4 rows by 4 columns). The fact that all element stiffness matrices lie on the main diagonal, and that the rest of the matrix contains 0 values, enables compaction of the global beam matrix. The Gauss reduction algorithm for such a matrix is complicated by the compaction method, but nevertheless still fairly short. This algorithm will be presented later.

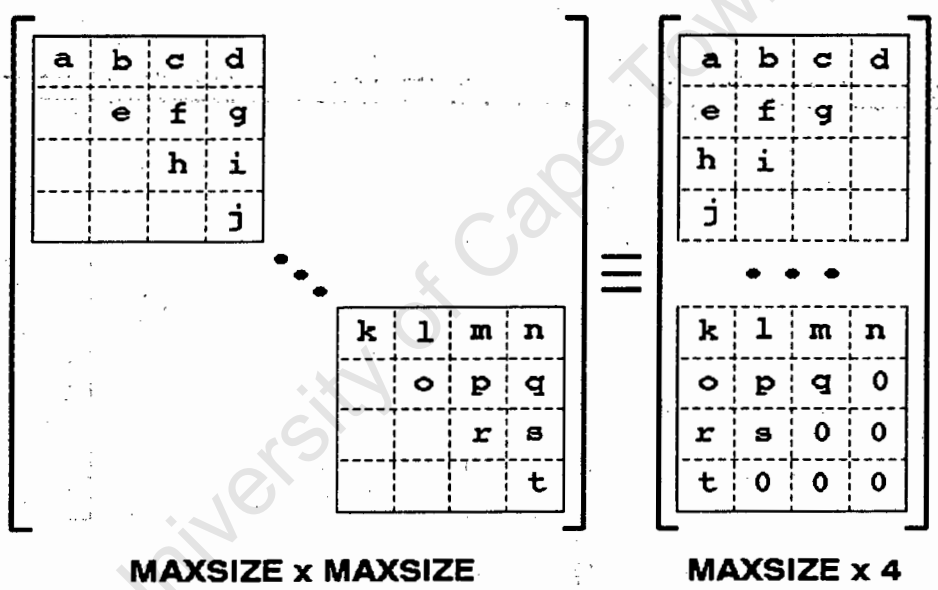


Figure 6.2 : Matrix storage compaction

6.1.3 Load, Displacement And Force Vectors

The load and displacement vectors are stored in the arrays

```
displacement = array [MAXSIZE] of real numbers
load_vector = array [MAXSIZE] of real numbers
```

These relate to equation (3.5). The internal force vector, containing the bending moments and shear forces for the elements is stored in

```
force_vector = array [MAXSIZE] of real numbers
```

in accordance with equation (3.13).

6.1.4 Global Variables

The following variables are required globally (ie. accessible to all routines within the program).

`num_elements`	integer. Actual number of elements used in program at any time
`first_element`	element→. Address of first element in memory. Allows a linked list of elements to be set up to represent the beam.
`order`	integer. Equal to 2*(num_elements+1). Indicates the size of the beam matrix and associated arrays for the specific number of elements in the program at any one time.

The pressure-displacement information for the in-situ and backfill material is held the arrays

```
insitu_displ = array [MAXPOINTS] of real numbers
insitu_press = array [MAXPOINTS] of real numbers
backfill_displ = array [MAXPOINTS] of real numbers
backfill_press = array [MAXPOINTS] of real numbers
```

where the `MAXPOINTS` parameter holds the maximum number of points permitted to define such a relationship. The actual number of points defining these curves is held in the two associated variables (the exact number of points must be noted if there are less than `MAXPOINTS` defining a curve):

`insitu_points`	number of points defining in-situ curve
`backfill_points`	number of points defining backfill curve

Similarly, the stress-strain relationship for the beam material is held in the arrays

```
beam_stress = array [MAXPOINTS] of real numbers
beam_strain = array [MAXPOINTS] of real numbers
```

and the actual number of points is stored in

`beam_points`	number of points defining beam material stress-strain relationship.

6.2 ASSEMBLING STIFFNESS MATRICES AND ASSOCIATED VECTORS

Once the elements and their parameters have been entered into the program, and a linked list of the elements built, the pressure-displacement relationships for the in-situ and backfill material must be entered, as well as the stress-strain data for the beam material. The program will then have the entire problem defined: the beam defined by elements and a stress-strain relationship, and the soil layer(s) by pressure-displacement curves.

6.2.1 Element Stiffness Matrix Compilation

The following routine compiles the stiffness matrix for a given element number 'i' using a 4x4 matrix of real numbers which will later be inserted into the global beam matrix.

routine : **compile_stiffness_matrix**
given : element number 'i'
local data :
    s_matrix = matrix [4][4] of real numbers
algorithm :
    {This algorithm is more easily described in writing than pseudo-code}
    set the values of the 16 entries in s_matrix according to the formulae for the stiffness matrix for a beam on an elastic Winkler foundation given in equation (4.3). Use the values associated with the parameters of the given element 'i'.

    Since the global beam matrix is compacted as discussed in section 6.1.2, the algorithm need only assemble the required terms in the upper triangular portion of the matrix, ie.
    s_matrix[0][0], [0][1], [0][2], [0][3]
    s_matrix[1][1], [1][3]

6.2.2 Beam Stiffness Matrix Compilation

routine : **compile_beam_matrix**
algorithm :

```
for i = 1 to num_elements
    compile_stiffness_matrix (i)
    beam_matrix [2*i][0] += s_matrix[0][0];
    beam_matrix [2*i][1] += s_matrix[0][1];
    beam_matrix [2*i][2] += s_matrix[0][2];
    beam_matrix [2*i][3] += s_matrix[0][3];
    beam_matrix [2*i+1][0] += s_matrix[1][1];
    beam_matrix [2*i+1][1] -= s_matrix[0][3];
    beam_matrix [2*i+1][2] += s_matrix[1][3];
    beam_matrix [2*i+2][0] += s_matrix[0][0];
    beam_matrix [2*i+2][1] -= s_matrix[0][1];
    beam_matrix [2*i+3][0] += s_matrix[1][1];
next i
```

ie. the entries in the global beam stiffness matrix are assembled from the stiffness matrix values for each element while looping through the elements from first to last.

6.2.3 Load Vector Compilation

The global load vector is compiled by looping through the elements from first to last and referring to the parameters: force_l, force_r, moment_l, and moment_r for each

element. In addition, if there is a foundation response beneath the element (indicated by a positive *udl* parameter value for the element), then the equivalent shear and moments (see figure 3.6) are factored into the load vector as well.

```
routine : compile_load_vector
local data :
     real : shear
     real : moment
algorithm :
     for i = 1 to num_elements
          shear = (element[i].udl * element[i].length) / 2
          moment = (element[i].udl * element[i].length^2) / 12
          load_vector [2*i] += (element[i].force_l + shear)
          load_vector [2*i+1] += (element[i].moment_l + moment)
          load_vector [2*i+2] += (element[i].force_r + shear)
          load_vector [2*i+3] += (element[i].moment_r + moment)
     next i
```

6.3 SOLVING FOR UNKNOWN DISPLACEMENTS

Having assembled the matrix and vectors, equation (3.12) can now be solved to determine the unknown displacements of the beam. The algorithm presented employs the Gauss reduction method. However, before any reduction can take place, the columns and rows in the beam matrix which correspond to fixed displacements (ie. displacements known to be 0) must be removed as mentioned under *matrix reduction* in section 3.3.

6.3.1 Removing Known Displacements Indices

In order for a known displacement index *i* in the beam matrix to be removed (and consequently its associated rows and columns as shown in figure 3.8) correctly when using the compacted storage method for the beam matrix, it is necessary to first scan the entire list of elements and compile a separate list of all displacements indices which must be removed before any modifications are actually made to the `beam_matrix`. In the routine below, the list of indices to be removed is kept in the local variable `displ_list`.

In the algorithm, the left end of the first element is checked for any displacements known to be zero. Thereafter, all the elements have their right ends checked.

```
routine : matrix_reduction
local data :
     displ_list = list of integers
algorithm :
     let displ_list = empty
     if left end of first element is
          pinned :  add '0' to displ_list
          fixed :   add '0' and '1' to displ_list
```

```
for i = 1 to num_elements
     if right end of element[i] is
          pinned :  add '2*i+2' to displ_list
          fixed :   add '2*i+2' and '2*i+3' to displ_list
next i
```

Thus the `displ_list` holds all the displacement indexes ready for removal from the `beam_matrix`. Although the displacements held in the list should be removed as described in section 3.3, this would represent a complex operation given the compact storage method for the beam matrix. A simpler method of achieving the same goal is to go ahead with the Gauss reduction of the matrix, but ignore those indices held in `displ_list`. Therefore, a routine called `skip(i)` is called to determine if the displacement index *i* should be ignored in the Gauss reduction process.

routine : **skip**
given : `displacement index 'i'`
algorithm :

```
     scan displ_list
     if 'i' is in the list return TRUE
     otherwise return FALSE
```

6.3.2 Gauss Reducing The Global Stiffness Matrix

The algorithm for the Gauss reduction routine is described below. Note that `beam_matrix` is shortened to `bm`.

routine : **gauss_reduce_matrix**
local data :

```
     integer : stop      {column where to stop looping}
     integer : endrow    {row where Gauss normalising stops}
     real : factor       {row multiplication factor}
     real : sum          {used for row summation}
```

algorithm :

{ NB: arrays and matrices start at index 0, thus value of variables like 'order' must have 1 deducted }

```
     for i = 0 to order-1

          if skip(i) is FALSE then

          endrow = 3
          {if near bottom of matrix, adjust endrow accordingly}
          if ( i > order-4 ) then endrow = order-i-1

          { normalise rows below this one. Only normalise those rows where cells in
          the matrix contain non-zero entries }

          for j = i+1 to i+endrow
```

```
            if bm[i][j-i] ≠ 0 and skip(j) is FALSE then
            factor = bm[i][j-i] / bm[i][0]

            for k = 0 to endrow-j+i
                 if skip(j) is FALSE
                 bm[j][k] -= bm[i][j-i+k] * factor
                 load_vector[j] -= load_vector[i] * factor
            next k
       next j
next i
```

The unknown displacements can now be solved for and stored in the array `displacement`. Below `displacement` is shortened to `d`, and `beam_matrix` to `bm`:

routine : **solve_displacements**
algorithm :

```
if skip(order-1) is TRUE then d[order-1] = 0
else if ( bm[order-1][0] ≠ 0 ) then
     d[order-1] = load_vector[order-1] /
     bm[order-1][0]
else error : more than 1 solution to matrix

{ now work backwards, substituting for a known displacement where possible }

for i = order-2 to 0

if skip(i) is TRUE then d[i] = 0
else
     sum = 0
     stop = 3
     if ( i > order-4 ) then stop = order-1-i

     for j = 1 to stop
          sum += bm[i][j] * d[i+j]
          if ( bm[i][0] ≠ 0 ) then
               d[i] = ( load_vector[i]-sum ) / bm[i][0]
          else
               d[i] = 0
     next j
next i
```

To solve for the internal bending moments and shear forces within the beam, the following routine compiles the shear forces and bending moments of the entire beam into one array `force_vector` as per equation (3.13).

routine : **solve_internal_forces**
local data :

```
vec = array [4] of real numbers
sol = array [4] of real numbers
real : length   { length of element }
real : udl      { foundation response to element deflection }
real : shear
real : moment
```

algorithm :

```
for i = 0 to 3
     vec[i] = displacement[i]
next i
```

{ as is generally the case, the algorithm works on the first element, then loops through the remaining elements, with reference to their right end values only }

```
{ compile stiffness matrix for first element }
compile_element_matrix (1)
sol = s_matrix * vec

{ sign convention means shear & moment on left end of first element are reversed }
force_vector[0] = -sol[0]
force_vector[1] = -sol[1]
force_vector[2] = sol[2]
force_vector[3] = sol[3]

{ remove any distributed load if acting on first element }
force_vector[0] += element[1].force_l;
force_vector[1] -= element[1].moment_r;
force_vector[2] -= element[1].force_l;
force_vector[3] -= element[1].moment_r;

{ add any foundation distributed reaction effect }
udl = element[1].udl
length = element[1].length
shear = (udl * length) / 2
moment = (udl * length²) / 12
force_vector[0] -= shear
force_vector[1] += moment
force_vector[2] += shear
force_vector[3] += moment

for i = 2 to num_elements
     compile_element_matrix (i)
     length = element[i].length
     udl = element[i].udl
     for j = 0 to 3
          vec[j] = displacement[2*i+j]
     next j
     sol = s_matrix * vec
     force_vector[2*i+2] = sol[2]
     force_vector[2*i+3] = sol[3];

     { add any original UDL effect as before }
     force_vector[2*i+2] -= element[i].force_l
     force_vector[2*i+3] -= element[i].moment_r
     { add effect of foundation udl }
     force_vector[2*i+2] += (udl * length) / 2
     force_vector[2*i+3] += (udl * length²) / 12
next i
```

6.4 NON-LINEAR ANALYSIS

Section 6.3 is sufficient to perform an analysis of a beam on an elastic Winkler foundation where the beam too is elastic in behaviour. For non-linear analyses, including the allowance for plastic hinging in the beam, the following routines are required.

6.4.1 Curve Interval Determination

Pressure-displacement curves are stored as piecewise linear approximations, and interpolation is necessary to 'read off' required values from such curves. The interpolation considers the various slopes of the line segments defining the curve. Since the information for the pressure-displacement relationship curves for the in-situ and backfill material is stored in arrays of *load* and *displacement* values, an algorithm is needed to determine the displacement interval for a given element's midpoint displacement so that the corresponding pressure interval is available for interpolation calculations. This concept is illustrated in figure 6.3.

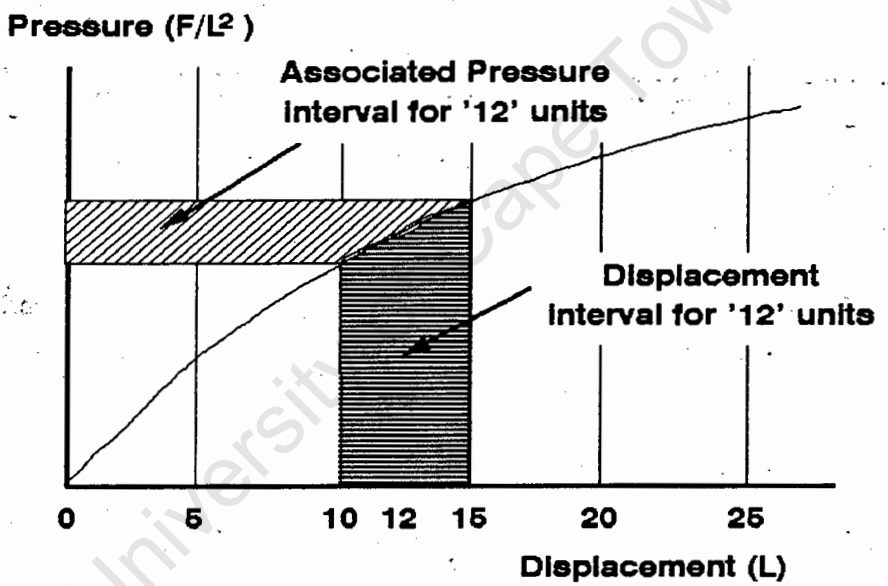


Figure 6.3 : Conceptual schematic showing pressure and displacement intervals for interpolating pressure value associated with displacement of '12' units

The routine below is a general routine which will work for any displacement information array of information.

routine : **determine_curve_interval**
given :

```
displacement = array[NUM_ENTRIES] of real numbers
real : d                { actual given displacement }
integer : interval      { required interval }
```

algorithm :

{ initialise interval. Since arrays start at index 0, 1 must be subtracted from the number of values (entries) in the array }

```
interval = NUM_ENTRIES - 1

while ( d <= displacement [interval] and interval >= 0 )
     let interval = interval - 1

{ correct the while-loop overshoot on last adjustment }
let interval = interval + 1
```

6.4.2 Calculating Element Subgrade Modulus And Foundation Reaction

For any given element, with a midpoint displacement (displacement in the middle of the element) of δ, the following routine is applied to determine the subgrade modulus and foundation reaction (as a distributed load) taking account of the appropriate pressure-displacement response for both the in-situ and backfill material. The algorithm is considers the *combination depth* (see section 5.2), denoted `comb_d`. The procedure for determining the foundation response given δ and `comb_d` was covered in chapter 5. For clarity, the following shorthand is used below:

```
insitu_displ ≡ id
insitu_press ≡ ip
backfill_displ ≡ bd
backfill_press ≡ bp

routine : calculate_k_and_foundation_udl
given :
     real : δ          {element midpoint displacement}
     real : width      {element width}
local data :
     real : k          {subgrade modulus}
     real : udl        {foundation reaction}
     real : pressure
     integer : i       {curve interval (see section 6.4.1)}
     real : α, ß
algorithm :
     pressure = 0
     if ( δ > comb_d ) then
          { determine in-situ response }
          let i = determine_curve_interval (δ) for in-situ soil
          { Linear interpolation to find 'pressure'}
          if ( i = 0 ) then
               pressure = ((δ-comb_d) / id[0]) * ip[0]
          else
               α = ((δ-comb_d)-id[i-1])/(id[i] - id[i-1])
               ß = ip[i] - ip[i-1]
               pressure = (α * ß) + ip[i-1]

     { backfill response }
     if ( δ > bd[backfill_entries-1] ) then
          pressure += bp[backfill_entries-1]
     else
```

```
        let i = determine_curve_interval (δ) for backfill
        if ( i = 0 ) then
             pressure += (δ / bd[0]) * bp[0]
        else
             α = (δ-bd[i-1]) / (bd[i]-bd[i-1])
             ß = bp[i]-bp[i-1]
             pressure += (α * ß) + bp[i-1]

    let udl = pressure * width
    let k = udl / δ
```

6.4.3 Check Routine For Element Subgrade Modulus

As described in section 4.2, in every iteration the subgrade modulus, *k*, of each element is inspected for possible alteration so as to arrive at an equilibrium condition between the loaded beam and the foundation response. The element's midpoint displacement, δ, is sent to the **calculate_k_and_foundation_udl** routine which computes the *k* value associated with the displacement. If this *k* value differs from the *k* value, presently associated with the given element, by an amount greater than *k_error* (user-specified amount), then the *k* parameter is changed to the new value.

The routine is called by the Master routine (see section 6.4.7) which expects this routine to return TRUE if the *k* parameter of the given element underwent a change. Otherwise, if there was no change, this routine returns FALSE.

```
routine : new_k
given :
     element number 'i'
     real : δ          {element midpoint displacement}
local data :
     width = element[i]->width
algorithm :

     calc_k_and_udl (δ,width);
     element[i]->udl = calculated udl
     if ( |element[i]->k - calculated k| >= k_error )
          then set element[i]->k = calculated k
          return TRUE
     else
          return FALSE
```

6.4.3.1 Routine to facilitate closure of iterative analysis

The principle of the iterative analysis is to repeatedly adjust the *k* parameters in the beam elements and then re-solve equation (3.12) until 'equilibrium' is achieved between the loaded beam and the responding foundation. The 'equilibrium' is actually an equilibrium state within a reasonable degree of error (*k_error*). Consequently, alterations of the k parameters must employ a method which enables closure of the iterative method. The method used in BSF

focuses on avoiding overadjustment of k parameters:

In addition to storing all the parameters of section 6.1.1 in the `element structure`, the following two real numbers are added to that structure: `prev_displ`, and `prev_prev_displ`. The former is the previous midpoint displacement for the given element, from the previous iteration (the value is 0 if the present iteration being performed by the program is the first iteration). The latter is the midpoint displacement from 2 iterations previous to the current one (0 if the current iteration is the 2nd iteration).

Using these two parameters, it is possible to trace the recent history of an element's displacement behaviour. In so doing, it is possible to converge on a suitable k parameter value which will prevent overcorrections from occurring. Let δ_2 be the value of the `prev_displ`, and δ_1 the value of the `prev_prev_displ` for an element i. Let δ_3 be the current midpoint displacement for the element. Three options can be distinguished:

1. $\delta_3 < \delta_2 < \delta_1$: The beam is deflecting normally, the element displacement is getting progressively deeper into the soil as the soil modulus decreases under the loading.

2. $\delta_2 < \delta_3 < \delta_1$: As for (1) since the most recent displacement δ_3 is lying between the previous two, indicating a convergence to a value between δ_1 and δ_2.

3. $\delta_2 < \delta_1 < \delta_3$: Possible instability developing. The foundation reaction to the δ_2 displacement of element i was too great, ie. δ_2 was too deep into the soil foundation. If the k parameter of element i was altered according to the most recent displacement, δ_3, then the next iteration will lead to very large displacement for i, likely to be even greater than δ_2. Instead the analysis should be aiming for a displacement between δ_1 and δ_2. This is achieved by ignoring δ_3 and by calculating a k parameter which would lead to a displacement (in the next iteration) between the displacements δ_1 and δ_2.

In `BSF`, this stability method has been incorporated into the routine **new_k** using the following algorithm:

```
routine : new_k
given :
     element number 'i'
     real : d          {element midpoint displacement}
local data :
     width = element i's width
     real : k1         {k for prev_prev_displ}
     real : udl1       {udl for prev_prev_displ}
     real : k2         {k for prev_displ}
     real : udl2       {udl for prev_displ}
algorithm :
     if ( d > prev_prev_displ and d > prev_displ and
          prev_displ < prev_prev_displ )
          { unstable tendency }
```

```
        then
             calc_k_and_udl ( prev_prev_displ,width )
             let k1 = calculated k
             let udl1 = calculated udl
             calc_k_and_udl ( prev_displ,width )
             let k2 = calculated k
             let udl2 = calculated udl
             set element[i].k = (k1 + k2) / 2
             set element[i].udl = (udl1 + udl2) / 2
        return TRUE
   else
        calc_k_and_udl (d,width);
        set element[i].udl = calculated udl
        if ( |element[i].k - calculated k| >= k_error )
             then set element[i].k = calculated k
             return TRUE
        else
             return FALSE
```

6.4.4 Check Routine For Element Beam Modulus

In a similar manner to altering an element's subgrade modulus and foundation response in accordance with its midpoint deflection, the element's beam modulus, *E*, can be altered to model inelastic beam behaviour. The routine makes use of **solve_internal_forces** to determine the bending moments at the ends of any given element.

These bending moments are used to determine stresses are at the element ends. Averaging these stress values derives a *midpoint stress* for the given element. If this midpoint stress exceeds the maximum inelastic stress of the beam material, plastic hinging would occur (as discussed in section 6.4.5). If the stress exceeds the maximum elastic stress of the beam material, and the beam material curve has an inelastic range as defined in the global variables `beam_stress[]` and `beam_strain[]` (not applicable in the case of Grade 300W steel), then the element's *E* parameter is changed to the inelastic modulus as defined by the beam material stress-strain curve.

The routine is called by the Master routine (section 6.4.7) which expects this routine to return TRUE if the *E* parameter of the given element underwent a change from the elastic modulus value to the inelastic value; otherwise, this routine returns FALSE.

```
routine : new_E
given : element number 'i'
algorithm :
     calculate moments at left and right end of the element
     calculate bending stress due to these moments
     average the stresses to arrive at element 'midpoint' stress
     if midpoint_stress > beam_stress[0] {yield stress}
          then
          element[i].E  =  (beam_stress[1]-beam_stress[0])  /
               (beam_strain[1]-beam_strain[0])
          return TRUE
```

```
        { thus the element beam modulus has been set to the
        inelastic range }
    otherwise return FALSE
```

6.4.5 Check Routine For Plastic Hinging

At the end of every iteration during the analysis, it is necessary to determine whether any hinges have developed in the beam. This is effectively accomplished by solving for the bending moment distribution of the beam. The extreme values are investigated with regard to yield stress of the beam material, and a hinge is inserted at those locations along the beam where bending stresses caused yielding of the beam material.

Since the algorithm will proceed from left to right along the bending moment distribution, hinges should only be inserted after the entire beam has been examined, otherwise insertion of hinges prior to complete examination of the distribution would complicate the linked list structure (see figure 6.1) for the beam elements. Therefore, a list of node numbers where hinges will be introduced is compiled and used for hinge insertion only after the routine is completed.

routine : **check_for_hinges**
given :

```
    real : depth                    {element depth}
    real : I                        {element I}
```

local data :

```
    hinge_list = list of integers {node numbers}
    real : bstress                  {bending stress}
    moment = array[order/2] of real numbers
```

algorithm :

```
    use solve_internal_forces to determine the shear forces and
    bending moments within the beam
    let moment[] = bending moments extracted from force_vector[]

    let hinge_list = empty
    for each node 'i' in the beam where such a node is
    i)   first node in beam, or
    ii)  last node in beam, or
    iii) node where moment[] peaks

        bstress = moment[i] * depth/2 / I
        { check if stress > yield stress }
        if ( stress > beam_stress[1] )
        then
            add 'i' to hinge_list

    scan hinge_list and for every node number in the list,
    insert a hinge into the linked list structure of the beam
    elements (see figure 6.1)
```

6.4.6 Pre-analysis Initialisation Of Variables

Prior to a new analysis being performed, it is necessary to reset key parameter values (*E*, *k*, and *udl*). The implications of resetting these parameters is discussed in section 7.4.5.2. The routine takes, as its only argument, the element number (the left-most element is number 1) at which the initialisation process will be performed. From that element, to the final element at the extreme right end of the beam, the parameters are reset as described below:

```
routine : pre_analysis
given : integer : start_element
algorithm :
     for i = start_element to last element
          element[i].E = elastic modulus of beam material
                         stress-strain curve
          element[i].k = elastic modulus of the backfill
                         pressure-displacement curve
          element[i].udl = backfill_press[0] * element[i].width
     next i
```

6.4.7 The Master Routine

The 'master' routine is the central controlling routine for non-linear analysis, and it requires all the above routines in performing the iterations necessary for a correct inelastic solution as described in chapter 4. The routine terminates when no elements have their subgrade modulus or beam modulus altered, and when no new hinges have been created as a result of bending stresses in the beam. Such a termination condition satisfies the 'equilibrium' between beam loading and foundation response, within a user-specified error.

```
routine : perform_nonlinear_analysis
local data :
     real : comb_d              { combination depth }
     real : displ_mid           { element midpoint displacement }
     boolean : busy_analysing   { control variable }
algorithm :
     { preanalysis initialisation }
     for every element in the beam
          let E = beam_stress[0] / beam_strain[0]
          let α = backfill_press[0] / backfill_displ[0]
          let k = α * element width
          let udl = backfill_press[0] * element width

     ask user for value of comb_d
     solve_internal_forces
     busy_analysing = TRUE

     do the following routines while busy_analysing = TRUE...
```

```
compile_beam_matrix
matrix_reduction
gauss_reduce_matrix
solve_displacements
solve_internal_forces

busy_analysing = FALSE
for each element 'i' in beam
          displ_mid = midpoint displacement
          if ( displ_mid >= 0 ) then
               { set k to elastic value }
               α = (backfill_press[0] / backfill_displ[0])
               k = α * element width
               if ( udl under element ≠ 0 )
                    {turn off foundation response}
                    udl = 0
                    busy_analysing = TRUE
          else
               calculate_k_and_udl (displ_mid)
               if  new_k(i)  returns  a  TRUE  signal  then
                    busy_analysing = TRUE
          if  new_E(i)  returns  a  TRUE  signal  then
                    busy_analysing = TRUE

check_for_hinges
if any new hinges then busy_analysing = TRUE
```

end of 'do'

CHAPTER 7

COMPUTER PROGRAM "BSF"

The computer program BSF (Beam on Soil Foundation) is an implementation of the methods of the previous chapters to solve the type of problems described therein. The development of this program is central to this research work and this chapter presents the program in an organised and complete way. The following points were addressed in this chapter:

- Program files
- Program specifications
- Graphical user interface
- Instructions for use
- Program validation

The computer program is some 7300 lines long and has been coded in the powerful C++ language, which is rapidly becoming the language of choice for today's leading software applications on desktop computers. Due to length requirements, no source code listing has been included in this document. It should be noted that several routines in the program are based on previous work done (Howie, 1991).

7.1 PROGRAM FILES

When programming in the C language (or its superset language, C++) it is common practice to break a large program into separate *module* files. These files contain the *source* code which is later compiled to form the final program.

The division of a program into modules makes it easier to group related functions and routines into a certain module. For example, all the routines required to assemble a stiffness matrix could be written in the module file `MATRIX`. Should the programmer wish to examine the code relating to the stiffness matrix, it is clearly identified.

By convention, module files which contain C language source code are given a '.C' extension. Thus, a 'matrix' module containing source code for matrix operations would actually be referred to as `MATRIX.C`. Those modules containing C++ source code are given the extension '.CPP'. When modules of source code are compiled and linked to create an executable file (the actual 'program'), the file is given the extension '.EXE' (for EXEcutable code). Files with a '.H' extension are called *header* files. These contain important definitions needed by the compiler when it compiles the .CPP files.

7.2 PROGRAM SPECIFICATION

7.2.1 Program Files

BSF was written using Borland's Turbo C++ Version 1. The main part of the program is 5372 lines long and comprises the files listed in table 7.1

Table 7.1

FILE	LINES	DESCRIPTION
THESIS.H	137	Global definitions
ACKNOWDG.CPP	38	Acknowledgement windows
EDITOR.CPP	310	Beam element editor
ELDIVIDE.CPP	185	Division of beam element into subelements
ENDFORCE.CPP	165	Solution of internal bending moments and shear forces
ERRORS.CPP	78	Error display windows
FILES.CPP	226	Input/output of data files
GETKEY.CPP	31	Obtaining user key selection
GLOBLVAR.CPP	53	Global variable declarations
GRAPHS.CPP	262	X-Y Graph display windows
INPUT.CPP	602	Input of beam & foundation information
LOADING.CPP	212	Element load specification
MATRIX.CPP	218	Compilation of beam matrix and associated vectors
MEMORY.CPP	91	Memory allocation and initialisation
MENU_C.CPP	209	*Curve* menu driver
MENU_E.CPP	197	*Element* menu driver
MENU_O.CPP	71	*Outfile* menu driver
MENU_S.CPP	248	*Solve* menu driver
MENU_V.CPP	299	*View* menu driver
MENU_X.CPP	108	*Export* menu driver
OUTFILE.CPP	272	Routines to write solution output to data files
REDUCE.CPP	106	Global reduction of beam matrix
SETUP.CPP	224	Main screen display setup
SOLVE.CPP	580	Inelastic solution
TABLES.CPP	450	Tabular display routines

The above files form the core of BSF. The graphical user interface source code files are listed in

table 7.2. The graphical user interface features will be discussed in section 7.3.

Table 7.2

FILE	LINES	DESCRIPTION
GRTOOLS.H	324	Definitions for interface tools
GRMOUSE.CPP	331	Mouse device routines
GRGUI.CPP	173	*Icon* class definitions
GRMENU.CPP	332	*Pull-down Menu* class definitions
GRFRAME.CPP	143	*Window Frame* class definitions
XYGRAPH.CPP	123	X-Y graph routine
TICKS_F.CPP	73	Automatic axis scaling routine for X-Y graphs
GRTABLE.CPP	164	*Table Window* class definitions
ENTRYBOX.CPP	193	*Entry Box* input routine
ENTRYWIN.CPP	77	*Entry Window* frame routine

The files of tables 7.1 and 7.2 make up the program `BSF.EXE`, or `BSF` for short. The graphical user interface for `BSF` uses 3D-icons which require that the pictures assigned to them be specified in a *binary* file containing a bitmap image. An icon image is created by drawing its picture on the screen using a separate program, and then saving the image generated to a binary file. Upon execution of `BSF`, this binary file is then linked to an icon *object* within `BSF` itself. The icon object uses the image from the file and stores it in memory from where it can be displayed when required. The bitmap image files for the icons are listed in table 7.3.

Table 7.3

ICON File	DESCRIPTION OF ICON
ELEMENT.DAT	Horizontal beam under an applied point load. Activates `Element` menu
CURVE.DAT	Pressure-displacement curve. Activates `Curve` menu
CALCULTR.DAT	Calculator. Activates `Solve` menu
BINOCULR.DAT	Binoculars. Activates `View` menu
OUTFILE.DAT	Arrow pointing to a folder. Activates `Outfile` menu
EXPORT.DAT	Hand with index finger pointing. Activates `Export` menu
QUITICON.DAT	Button with the word 'Quit' on it. Exits `BSF` and returns to DOS

The '.DAT' extension is commonly used for data files.

The icons of BSF have been designed to appear 3-dimensional, and an added enhancement to the interface is that an icon, when activated, appears to 'click in' like a button on a control panel, and at the same time a small green light on the icon glows light green to denote activation of the icon. Since this 'activated' image requires a different bitmap for the green light, the icon bitmap '.DAT' files of table 7.3 have accompanying '.GRN' (for GReeN) image files of the same name. These files are also linked to icons in BSF.

All the '.DAT' and '.GRN' icon files must be placed in the same *working directory* as BSF for the program to run.

The files listed in table 7.4 belong to the Borland Turbo C++ package. Borland has given permission for these files to be distributed with programs developed using the Borland C++ package.

Table 7.4

FILE	DESCRIPTION
EGAVGA.BGI	Driver for EGA/VGA monitors
LITT.CHR	Small font data file. This font is used for all characters in BSF expect the main screen icon headers, and the options in pull-down menus
TRIP.CHR	Triplex font data file. This font is used for options in pull-down menus

Like the icon image files, these files must also be in the same directory as BSF. All the files required to run the program are listed in table 7.5.

Table 7.5

FILE	DISK SPACE (bytes)
BSF.EXE	156 752
LITT.CHR	5 131
TRIP.CHR	16 677
EGAVGA.BGI	5 554
All the '.DAT' icon image files	9 702
All the '.GRN' icon image files	8 940

7.2.2 Hardware Required

The program will run on any IBM or IBM-compatible using a DOS-compatible operating system. Minimum memory is 640K, and a VGA monitor is required.

A mouse is recommended but not essential. If the mouse is not a 'Logitech' make of mouse, then the user might find the mouse pointer not appearing on the screen. In such an event, the mouse driver must still be loaded otherwise those portions of the program which expect a mouse to be connected to the computer will be left in infinite loops while they try desperately to find the mouse. The user will have to make do with keyboard entries only.

7.2.3 Program Features

The program allows the user to define a beam of elements as well as any applied load(s) acting on the beam. The definition can be edited and saved to a data file for permanent storage. A *subdivide* feature allows the user to specify the minimum number of beam elements without affecting accuracy of the output (since data is output for each nodal point in the beam - the more elements, the more information available to the user). Having defined the beam using the minimum elements necessary, the user has the option of subdividing the simple beam definition into one of many smaller elements. Due to memory limitations, a maximum of 155 elements has been specified, which should prove more than sufficient for almost every possible beam-soil problem.

A pressure-displacement relationship for any foundation material can be defined. As with the beam elements, it is possible to save the foundation information to a file, although there is no editor provided for the foundation parameters. Foundation curves are read off the disk and into memory to represent the different soil layers in an analysis. Stress-strain curves for beam materials can also be defined with this feature. At any time, the user can view the various curves both in tabular and graphical format.

The program allows both direct (non-iterative, elastic) and indirect (iterative, inelastic and plastic) analyses. The former can be used to analyse common beam problems where there is no foundation at all, or where there is an elastic foundation supporting the beam. The indirect analysis uses the iterative method as described in chapter 5. In such a solution, the user has the option of defining the foundation as a two-layered model (see chapter 6), or just a single medium of soil material. Various options are available during the iterative solution to allow the user to view key parameters and to specify the accuracy of the iterative process. If the user wishes to terminate an iterative analysis, this can be achieved by pressing the ESCape key.

Upon completion/termination of an analysis, the program allows the viewing of subgrade modulus, elastic modulus, foundation response, bending moments and shear forces, and nodal displacement for every element defining the beam. Such values may be viewed in tabular or graphical format.

An *outfile* option enables the user to save the solution parameters to a specified text file. Such a file could be printed on a typical line-printer, incorporated into word processor

documents, or just stored on a floppy disk as a record.

An *export* option allows exporting the information to *spreadsheets* where more sophisticated graphical display and printing facilities are available for formal presentations, if such features are necessary.

The program itself has a graphical user interface consisting of 3D icon buttons, pull-down menus with 3D borders, and entry and message windows. Entry windows are for accepting input from the user, while message windows are used to display error messages and program status. Tables and graphs are also displayed in windows with 3D borders.

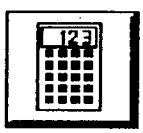
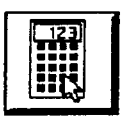
The look and feel of the interface is thus similar to those of most commercial programs available today. The interface allows the use of a *mouse* to select options displayed on the screen. If no mouse is available, options can be selected with appropriate keys.

7.3 GRAPHICAL USER INTERFACE

A graphical user interface (GUI) is the standard adopted for most of today's leading computer programs. The interface is characterised by the user making use of a *mouse*

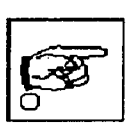
to select various options displayed on the screen. Such options can be in the form of *icons* or *menu options*.

7.3.1 Icons

Icons pictorially represent a predefined function. For example, the icon  might be linked with a 'calculation' routine. To select the icon, the user would move the mouse until the *on-screen pointer* (usually an arrow) logically linked to the mouse's movements, is displayed over the icon  .

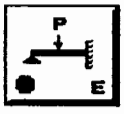
Then, if the left (by popular convention) mouse button is pressed, the icon will be activated, and perform its function. If the icon has been made to resemble a 3D 'button', it will look a little different on the screen after activation, usually as if it has been pushed into screen, as a real button might appear upon being pressed. The icons in BSF are modelled on this principle. In addition, for added realism, they also have a little green light on them which glows when the icon is selected, as shown below

Normal icon: 

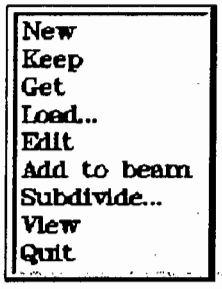
Selected icon: 

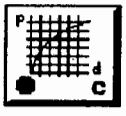
If no mouse is available, then most GUIs (that for BSF included) will allow the user to select an icon with a certain key as well. The mouse, therefore, provides the user with an on-screen pointing device with which to select program options.

With the exception of the 'Quit' icon, the icons in BSF are all linked to pulling down menus of options. The icons and their functions are given in figure 7.1.



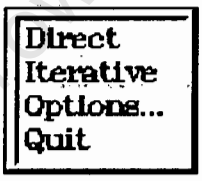
Elements : pulls down menu





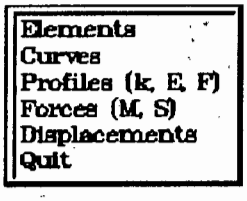
Curves : pulls down menu

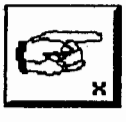
Solve : pulls down menu



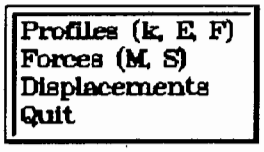
View : pulls down menu

Outfile : pulls down menu





Export : pulls down menu



quits BSF

Figure 7.1 : BSF icons and their functions

7.3.2 Pull-down Menus

The menus of figure 7.1 are called pull-down menus because they are not displayed on the screen until their controlling icon is selected with the mouse, at which point they are displayed below the icon as if having been 'pulled down'. A typical pull-down menu is shown in figure 7.2.

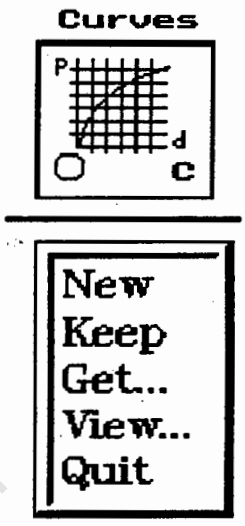


Figure 7.2 : Pull-down menu

The options on these menus are also selected with the mouse pointer. The user would move the pointer over the desired option, and then press the left button on the mouse. If an option has a '...' suffix (ex. option `View` in the menu of figure 7.2), then selecting that option will pull down a *child* menu as shown in figure 7.3. Most programs, `BSF` included, allow the user to press a certain key to activate an option. In figure 7.2, the user could press 'N' to select the 'New curve' option (the 'N' key is not highlighted in the figure due to black and white printing limitations). The 'Quit' option in a menu will close the menu window and cause the icon 'button' to click back out again, ready for activation if required. The 'Quit' option in a second pull-down menu will cause that menu to close, and return control to its *parent* menu.

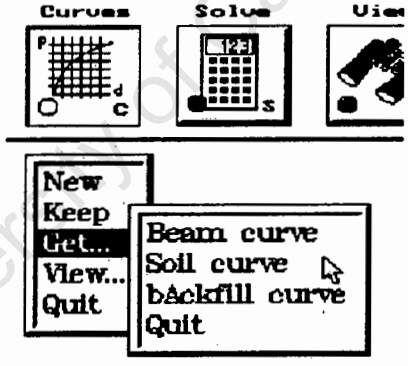


Figure 7.3 : Parent and child menus

7.3.3 Message Windows

Like most graphical user interfaces, the interface of `BSF` makes use of message windows such as

Figure 7.4 : Example of a message window

to display information to the user. Such messages are colour-coded depending on their nature; for example (figure 7.5), error messages have a red window to alert the user that an

error has occurred.

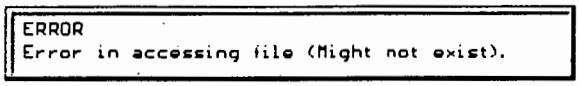


Figure 7.5 : Example of an Error message window

Message windows can be removed, once read, by pressing a key on the keyboard or any button on the mouse.

7.3.4 Entry Windows And Entry Boxes

Entry windows are used by BSF to prompt the user for a particular parameter value, an example is given in figure 7.6.

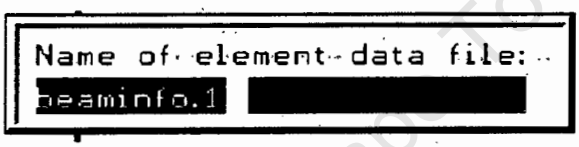


Figure 7.6 : Example of an Entry window

The dark portion of the window is where the cursor is displayed for the user to type the value/word required. This shall be referred to as an **Entry Box**. Entry Boxes in BSF have several advanced editing keys for use when information is typed in them:

- The **ESC**ape key can be used at any time to erase the current entry and start again
- The **backspace** (←) and **Delete** keys can be used in the traditional manner
- The **Home** key moves the cursor to the beginning of the input line, while the **End** key moves it to the end
- If the user types in characters which are longer than the width of the entry box, the characters are automatically *scrolled* within the box to maintain a neat output display.

7.3.5 Table Windows

The graphical user interface in BSF also makes use of tables to display information in a tabular format such as that of figure 7.7. The user can press the ↓ and ↑ keys to scroll the display up and down the table window one line at a time. The **PageUp** and **PageDn** keys move the table whole windows at a time. Accordingly, if the table window has 10 *display rows* but 60 actual rows of information associated with it, then only 1/6 of the table will be displayed in the window at any time (the small number of display rows is to keep the size

of the table small enough so as not to block out large parts of the screen). The scroll keys can then be used to adjust which of the actual rows are within the window display. The ESCape key (or any button on the mouse) can be used to close the window when the user has finished viewing the information.

Element	Distance	Moment (kNm)
011	1.833	-0.514
012	2.000	-0.825
013	2.167	-1.217
014	2.333	-1.698
015	2.500	-2.269
016	2.607	-2.684
017	2.713	-3.136
018	2.820	-3.623
019	2.927	-4.144
020	3.033	-4.696

Figure 7.7 : Example of a Table window

7.3.6 Graphic Windows

X-Y graphs are displayed by BSF in graphic windows like that of figure 7.8.

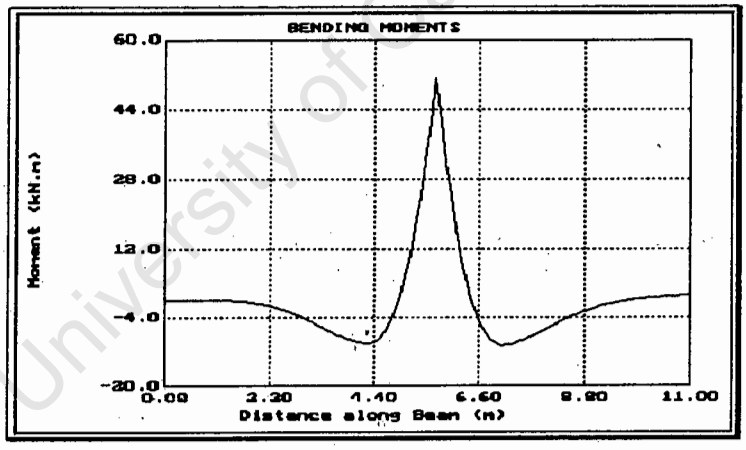


Figure 7.8 : Example of a Graphic window

As with table windows, a graphic window is closed by the user pressing ESCape or a mouse button.

7.4 INSTRUCTIONS FOR USE

In this section, all the features of the program are described, starting with the entry of the problem-defining information, and ending with the interpretation of output from an analysis. The various features have been described using reference to a common example problem, defined below, which is solved and interpreted during the course of the chapter.

7.4.1 Example Problem

Consider the beam foundation on a two-layered soil system as shown in figure 7.9. The results of plate load tests for the two soil layers are plotted in the pressure-displacement diagrams shown in figures 7.10A and 7.10B.

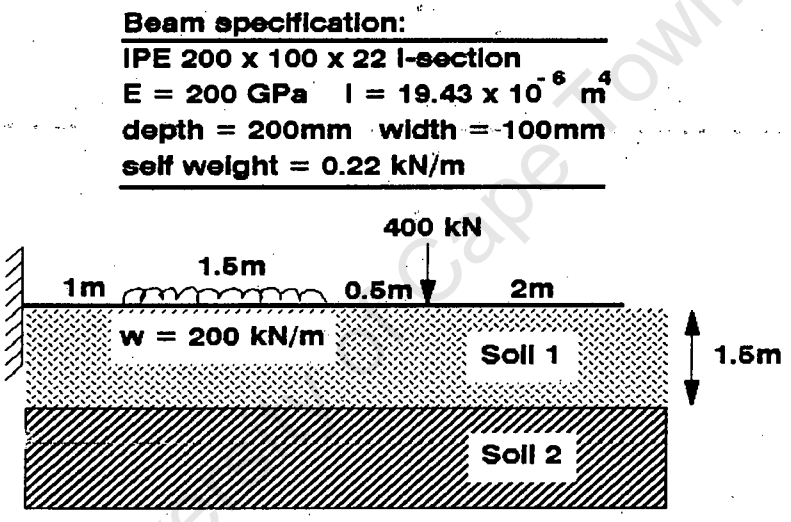


Figure 7.9 : Schematic of the example problem

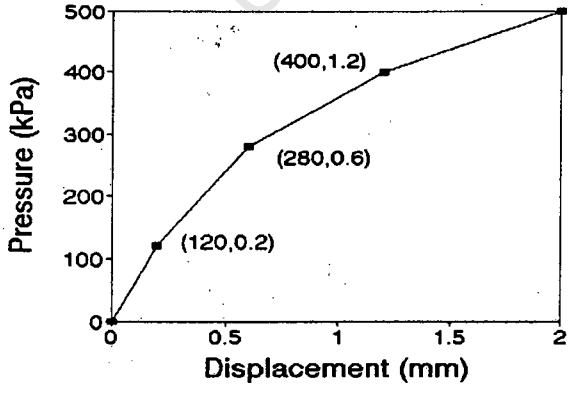


Figure 7.10A : Pressure-displacement curve for upper soil layer

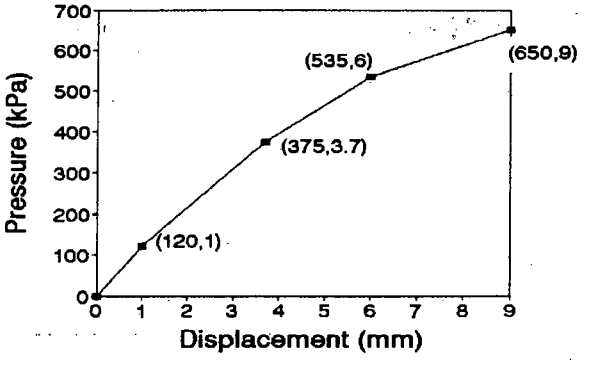


Figure 7.10B : Pressure-displacement curve for lower soil layer

7.4.2 Running BSF

To run BSF, the user will load the files as listed in table 7.5 (section 7.2.1) into a working directory and type

```
BSF
```

after which the program will run and the screen display will be as shown in figure 7.11. This display shows the different icons and their options : Elements, Curves, Solve, View, Outfile, Export, and Quit.

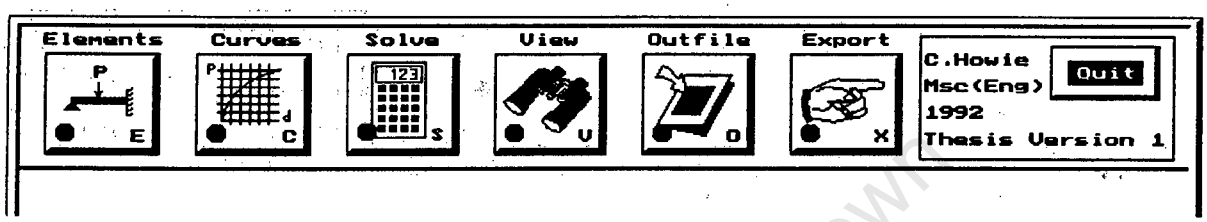
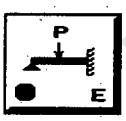


Figure 7.11 : Header of main screen display

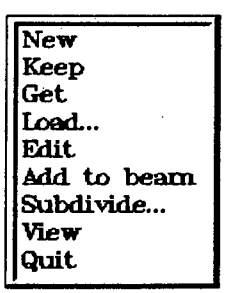
The layout of the icons follows a logical sequence for using the different options. The user will usually start with the option on the very left, Elements, and proceed from left to right, finally ending with the Outfile or even the Export option after the analysis is completed. This logical sequence will also be used to structure the various sections of this chapter.

7.4.3 Beam Elements

ICON:



MENU:



All the options for beam elements can be accessed with selection of the Element icon. As described in section 7.3.1, icons can be activated with the mouse, or by a certain keypress. As displayed on the Element icon, the 'e' or 'E' key can be used to select the icon. Once the Element icon has been activated, the element pull-down menu will be displayed.

7.4.3.1 Defining Beam Elements : Option 'New'

The **New** option is used to define a beam of elements; 'new' because such a definition will reset the computer's memory where the previous definition was stored. By convention, the first element in a beam is the left-most one, and elements are numbered in increasing order from left to right.

Upon selection of the **New** option, the program will display a large entry window which will prompt the user for all the necessary information defining the beam elements. Initially the window is blank, with only the

```
Length of the beam (m)
```

prompt and its associated Entry Box displayed. As explained in section 7.3.4, the user enters the length of the beam in the Entry Box. If the value entered is incompatible with that expected by the program, for example the user entered in a negative length for the beam, then an appropriate error message is displayed in an Error Window like that of section 7.3.3.

Systematically, the program will prompt the user for the

- length of the beam (m)
- number of elements defining the beam (maximum of 155 given memory limitations).
- major element parameters:
    - elastic modulus, E (kPa)
    - moment of inertia, I (m^4)
    - initial subgrade modulus, k (kPa)
    - depth of the element section, d (m)
    - width of the element section, w (m)
- element end-conditions of the type:
    - (1) Internal
    - (2) Pinned
    - (3) Fixed
    - (4) Free; or Spring (kN/m)

Note that the input routine will prompt the user for the elastic beam modulus as this is the standard beam modulus value. With respect to end-conditions, the user need only press the

keys marked '1' through '4' to specify the type of end-condition.

The end-conditions of elements are as described in chapter 3. A new addition is the concept of a *spring* end, which requires the user to enter a stiffness constant for the spring, denoted *spring_k* (kN/m). This parameter, in kN/m, is the response from the spring if that end of the element is deflected in the same direction as the sign of the value of spring_k. Thus if spring_k is positive, then positive displacement of the element end will invoke the spring reaction. For example, a spring end with *spring_k* = 100 kN/m, if deflected upwards (positive in BSF) an amount 0.011 metres, will invoke a spring response of 1.1 kN downwards.

The program will prompt the user for the number (1 to 4) which corresponds to the left end of the first element (the left-most) of the beam. Thereafter, since the *right* end of element *i* is clearly the *left* end of element *i+1*, the program only prompts the user for the right end-conditions of all the beam elements. This way every node (and consequently element end-condition) is considered. If the user presses '4' (a *free end* or *spring*) for any prompted end-condition, the program will ask for the value of the spring constant to which the user would respond with either '0' (if the end-condition is a free end), or a non-zero value (if the end-condition is a spring as defined above).

To facilitate easy entry of information, the program first prompts the user as to whether or not any of the element parameters *E* through *w* have a constant value for the entire beam. If so, the user can enter the constant value for an element parameter only once, without having to enter it for every element in the beam.

If the length of beam elements is not constant for all elements, then the user enters each element's length, with the exception of the last element, the length of which is calculated automatically from the total length of the beam.

Naturally, all element parameters must have a positive value. Wherever an incorrect value is entered by the user, an error message will be displayed and the Entry Box for that parameter will be cleared. Subsequently, the program will prompt the user to re-enter in an appropriate value. The only exception to this 'positive value' requirement is the subgrade modulus parameter, *k*. This is due to the following facts:

- When carrying out an **inelastic** analysis, BSF automatically assigns the initial (elastic) modulus of the soil curve to every element in the beam prior to the first iteration, and any user-defined value for *k* is ignored (the user could enter in 0, -1, 999 etc, without having any effect).

- However, when carrying out an **elastic** analysis, the program will use the user-defined value. A value of 0 in such an instance would not be erroneous for the following reason:

- Where the user might wish to solve a simple beam problem without any soil support, the user can enter in a zero value for *k* for every element, and then select an elastic solution. In so doing, no soil is modelled or considered in the analysis.

When the program prompts for element parameters and end-conditions, it automatically avoids prompting the user for values already known. Such values occur when the parameter in question has a constant value already defined, or when the left end-condition of an element is already known (which occurs when the preceding element's right condition is specified by the user - elements are numbered in increasing order from left to right).

In the case of the example beam problem, the following input values apply. The final window display, as it would look after all values have been entered, is given thereafter in figure 7.12.

```
Length of beam : 5 (m)
Number of elements : 4
Constant length elements ? No
Constant E for all elements ? Yes. Value > 200e6 (kPa)
Constant I for all elements ? Yes. Value > 19.43e-6 (m⁴)
Constant k for all elements ? Yes. Value > 1 (kPa), say
Constant d for all elements ? Yes. Value > 0.2 (m)
Constant w for all elements ? Yes. Value > 0.1 (m)
Length of element 1 : 1 (m)
Length of element 2 : 1.5 (m)
Length of element 3 : 0.5 (m)
Length of element 4 : Automatically calculated as 2 (m)
Left-condition of element 1 : 3 (fixed)
Right-condition of element 1 : 1 (internal)
Right-condition of element 2 : 1 (internal)
Right-condition of element 3 : 1 (internal)
Right-condition of element 4 : 4 followed by 0 kN/m (∴ free)
```

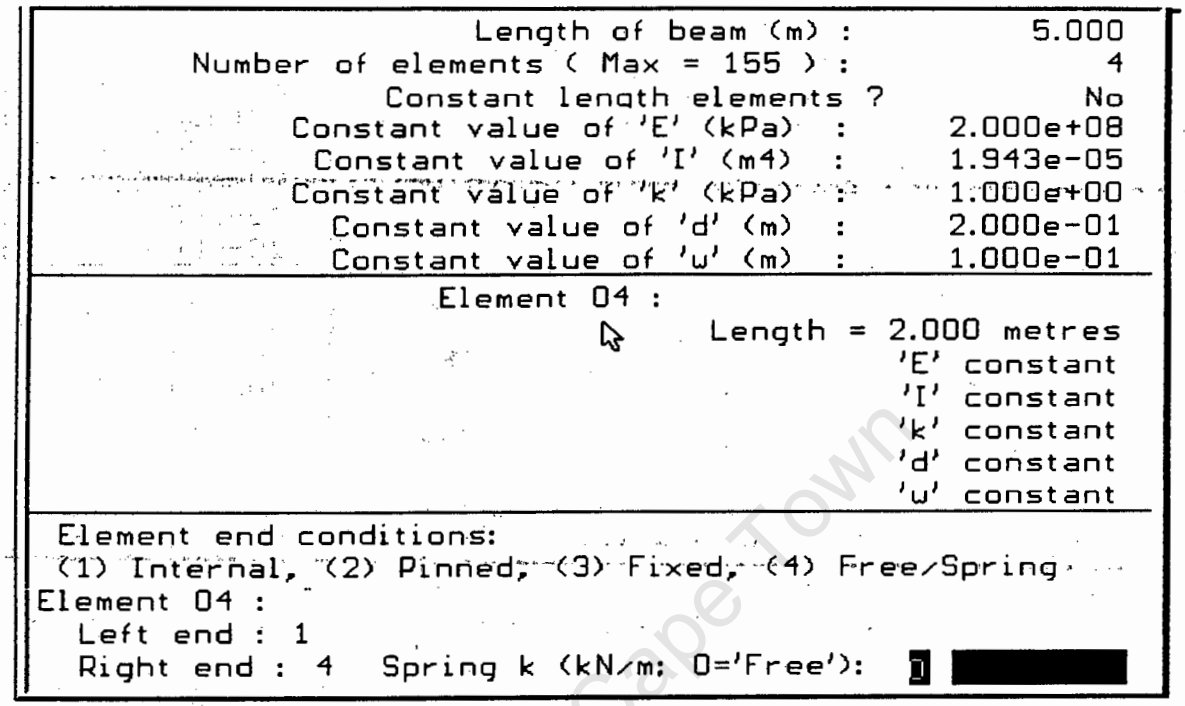
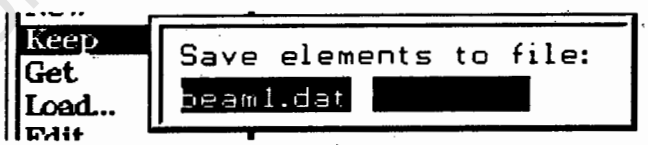


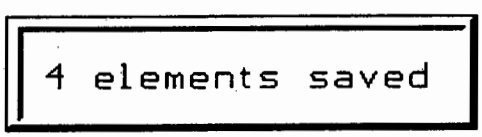
Figure 7.12 : Post-input window display

7.4.3.2 Saving Element Information : Option 'Keep'

Using the entry box:



the program asks the user to enter the file's name under which the element information will be saved, whereafter the information is saved to a text file of the specified name, and the user reminded of the number of elements written to the file with the acknowledgement window



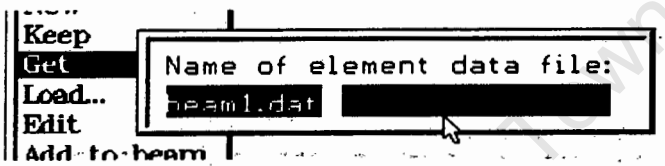
For the rest of this chapter, it will be assumed that the information for the example beam problem has been saved to a file called `beam.dat`. In specifying the file name, the user can use a full path name, for example

```
c:\bsf\data\beam.dat
```

A specified beam data file may be retrieved at any time, for editing or viewing purposes by using the following function:

7.4.3.3 Retrieving An Element File : Option 'Get'

The **Get** option asks the user for the name of the element data file to be retrieved



If the file requested cannot be found, an error message will be displayed stating that there is a fault with that file, or that it does not exist. Such an error would occur if the user misspelled the filename asked for.

When a user follows the entry procedure of 7.4.3.1 and defines a beam of elements, that information resides in the computer's memory, regardless of whether or not the user makes use of the **Keep** option. However, if an element file is successfully retrieved, then the information in that file *overwrites* any existing element information which may have been present in the computer's memory prior to the user selecting **Get**. This overwritten information is then lost, and would have to be re-entered using the **New** option, unless it had been saved to a data file (in which case it could be retrieved with **Get**).

7.4.3.4 Specification Of Beam Loading : Option 'Loading'

Upon selecting this option the program will display the **Loading** submenu

The **Specific** option allows the user to enter the loading information for a single specified element. The **All** option will result in the program going through all the elements in the beam, one after the other (from the left end - element 1), asking for loading information for each one. If the user wishes to load a group of elements with a non-uniform range of distributed loads, as, for instance, that shown in figure 7.13, then the **Group** option can be

used. The program will prompt the user for the first element in the group (No. 4 in figure 7.13), and the UDL magnitude at that element. The same is done for the last element in the group (No. 12), and the program automatically loads these elements with the approximate non-uniform loading - approximate because BSF requires a distributed load to be constant over the length of an element. Thus the effective non-uniformity in the load pattern will be a 'stepped' pattern over the element group.

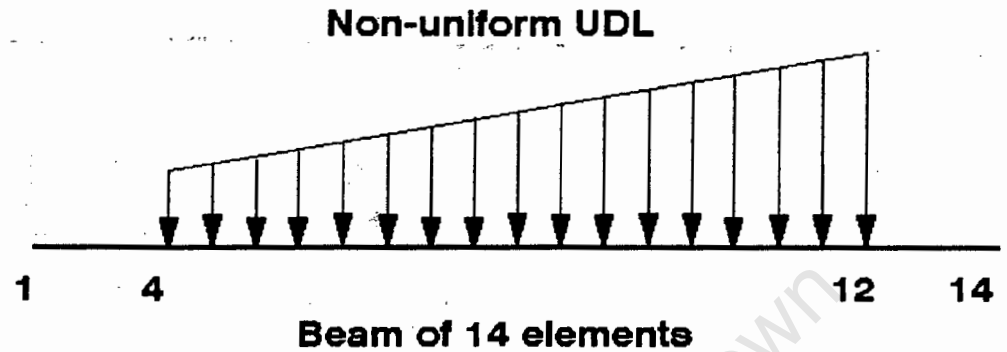


Figure 7.13 : Example of non-uniform UDL defined with the **Group** option

Finally, the **self Weight** option allows the user to specify the self weight of the entire beam (kN/m) whereafter the program will automatically apply this distributed load to every element in the beam without the user having to do so manually. In the case of the example beam problem, the user would specify a self weight of '0.22' kN/m (see figure 7.9).

In addition, elements 2 and 4 need to be loaded. Element 2 has a distributed load of 200 kN/m acting on it, while element 4 has a point load of -400 kN acting down on its left end. Note that this point load could also be viewed as acting on the right end of element 3. However, program convention in BSF is that, where point loads are concerned, the load is linked with the element whose left end it acts on. The loads for the example beam would be entered as follows:

Selecting the **Specific** option from the menu will allow the user to specify element 2 as the element for which loading information will be entered. Thereafter, the program will prompt the user to specify if a UDL is acting on the given element or not. The user would respond with 'y' (yes). The program would then require the user to enter in the magnitude of the distributed load ('200' kN/m). Note that, in keeping with the method of specifying elements covered in chapter 3, an element must be defined such that any distributed load acts over the entire length of the element. Also, distributed loads must be constant in magnitude.

The program will ask for the point loads acting on the left and right ends of element 2, as well as the magnitude of moments applied to the element's ends. The user would enter '0' in response to all these prompts since there are no such loads applied to element 2:

```
Point load at left end (kN)          > 0
Point load at right end (kN)         > 0
Moment at left end (kNm)             > 0
Moment at right end (kNm)            > 0
```

However, since a distributed load has already been specified for the element, the equivalent shear and moment forces, as shown in figure 3.6, will be displayed on the screen even though the user entered '0' for the different loads; ie. by entering 0 for point loads and moments, the user is really specifying that no additional load/moment is acting at either end, over and above that occurring as a result of the distributed load. This fact is illustrated in figure 7.14 below. On the other hand, if the user enters a non-zero value, then the equivalent shear/moment is added to the specified force to produce a combined shear/moment value.

For the above input, the final window display (as the user enters the 'right' moment value) is

```
If point load between elements i and i+1, enter
the load at LEFT end of element i+1.
     Shear force : upwards = positive
     End moments : anticlockwise = positive

                    Element 02

 UDL (kN/m: + = acting downwards) :     200.000
      Point load at left end (kN) :    -150.000
     Point load at right end (kN) :    -150.000
           Moment at left end (kN) :     -37.500
          Moment at right end (kN) :
```

Figure 7.14 : Example of a load definition window

To load element 4, the user would again select the **Specific** option and enter '4'. Since point loads acting downwards are negative, the entire loading response for element 4 is

```
Distributed load acting on element (y/n) ? No.
Point load at left end (kN)          > -400
Point load at right end (kN)         > 0
Moment at left end (kNm)             > 0
Moment at right end (kNm)            > 0
```

The user would have to save the new element information, to `beam.dat` (or any other file) by using the **Keep** option. If the information is not saved, then the loading information would be lost should execution of BSF be terminated.

7.4.3.5 Editing Element Information : Option 'Edit'

An edit feature is provided whereby the user can change a single parameter value for any specified element. For example, consider that the user wishes to find a solution to the example beam where the point load at the left end of element 4 is 500 kN and not 400 kN. Instead of entering in a whole new beam and then specifying loading on it, use could be made of the **Edit** option to simply change the 400 to 500, and then save the information to

a new file, say `beam2.dat`. Then `beam.dat` would contain all the original information of figure 7.9, and `beam2.dat` could be used for a solution with a 500 kN load acting downwards at on element 4.

After the **`Edit`** option is selected the user specifies the number of the element which is to be edited. The number given must be greater than 0 and less than or equal to the number of elements currently in memory (in this case 4). If not, an error message is displayed reminding the user of the number of elements available, and the user is reprompted for the element number. Having entered '4', the screen would show the editor display of figure 7.15.

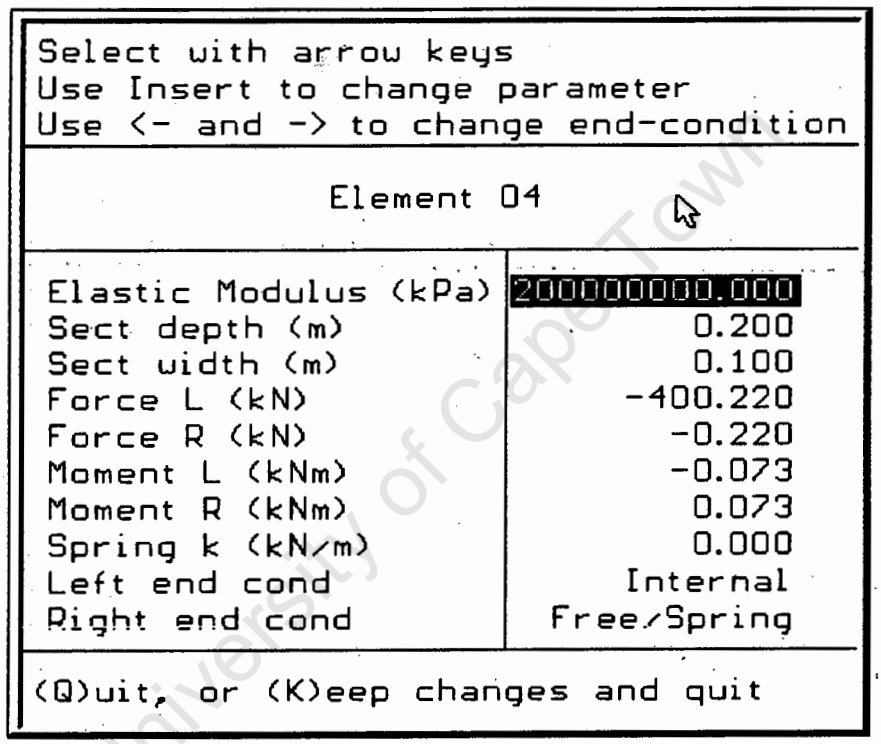


Figure 7.15 : Example of the element editor screen display

The editor makes use of a highlighted *cursor* to allow the user to select which parameter will be changed. In figure 7.15, this cursor is shown over the `section width` parameter. The cursor can be moved from one parameter to another with the ↓ and ↑ cursor movement keys. A *wrap-around* feature is provided which means that if the cursor is moved up from the top parameter (the element `length`) it will appear at the bottom row of the editor (the element's `right` end-condition), and vice versa if moved off the bottom of the editor window.

To change a parameter value, the user must press the **Insert** key, but only when the cursor has been moved onto that parameter. However, with end-conditions, the user moves the cursor onto either the left or right end-condition and makes use of the ← and → keys to *toggle* the particular condition over the range of values : `internal`, `pinned`, `fixed`, or `free/spring`. A reminder of this method is displayed in the editor window, above the

element parameters.

Therefore, to change the applied load in the worked example, the user would move the cursor down onto the `Force_1` parameter, and then press **Insert**. The program would prompt the user for the new value (of 500 kN). In order to preserve the self weight shear component of -0.220 kN, the user would actually enter a value of -500.220 kN to effect a change in point load from -400 kN to -500 kN. The screen display would be as shown in figure 7.16.

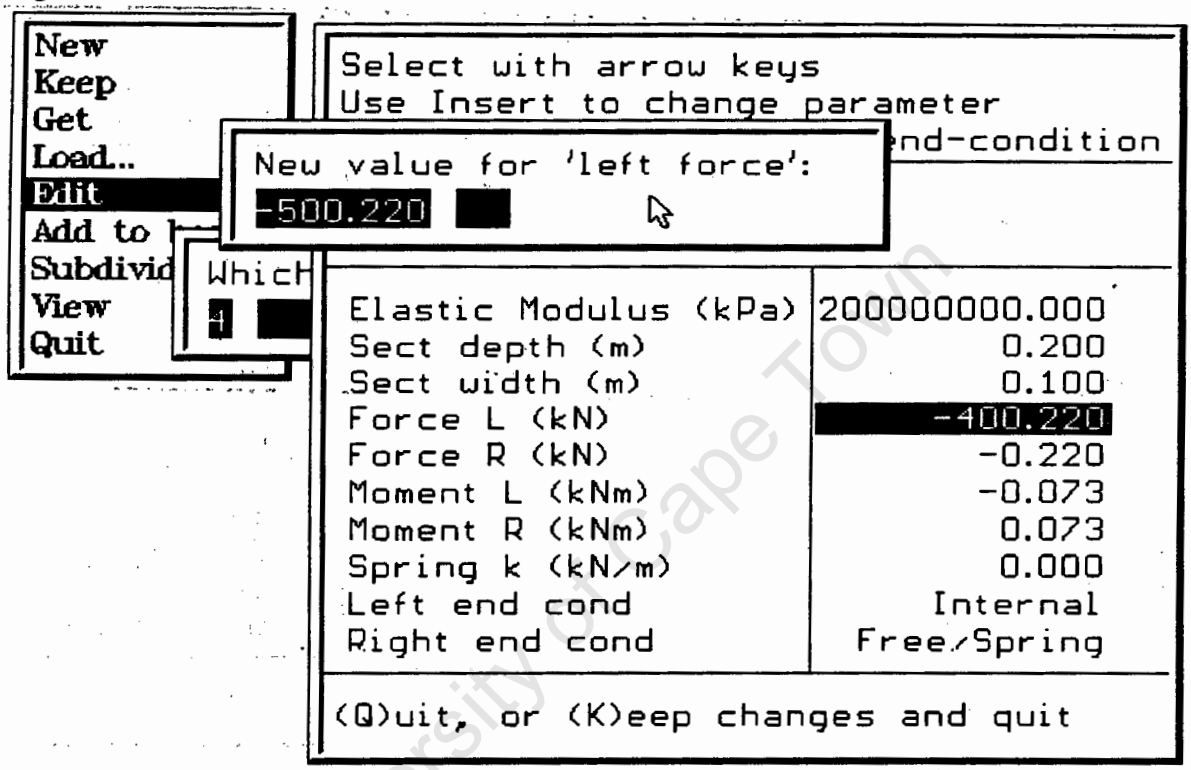


Figure 7.16 : Example showing specification of new value for an element parameter

When the user has edited all parameters as required, the editing session is complete. The user then has two options (again these are displayed on the screen, below the editor window) :

- Pressing 'q' or 'Q' will 'quit' the editor without effecting any changes on the specified element. In this case, it would mean the load at the left of element 4 would still be the old value of -400 kN.

- If the user presses 'k' or 'K' to 'keep' the alteration(s) made, then the new value of -200 kN would be stored in memory for element 4. The user could then use the `Keep` option and specify `beam2.dat` as the file for saving the information to, as described earlier.

The editing process uses a temporary storage area for edited values. If the user wishes to cancel the editing process without the changes taking effect (by pressing 'Q'), the use of a temporary area means the actual memory contents for the element being edited remain intact. If the user wishes to 'keep' the changes, then the information in the temporary area

overwrites the actual memory contents for the element.

7.4.3.6 Adding On More Beam Elements : Option 'Add to beam'

If the user wishes to add on more beam elements, this option allows the user to specify the length of beam (metres) to be added to the right end of the existing beam, and how many elements will define this new beam segment. The parameters like *E*, *I* and so on are set the same values as the last element at the very right end of the previous beam definition, and all end-conditions for the new segment are made *internal*. If the user wishes to change the parameter values of the new elements, then the editor described in section 7.4.3.5 will have to be used.

7.4.3.7 Subdividing Elements : Option 'Subdivide'

Since output values such as bending moments, displacements etc are only given for the **nodes** at the ends of elements, the example beam of only 4 elements will offer a coarse solution for any output values considered. Subdividing the four elements into an equivalent beam of, say, 100 elements will produce much more accurate results, as well as produce smoother graphs of such output as the displaced shape of the beam.

However, it would be very inefficient for the user to enter all the details for a beam of 100 elements! Therefore BSF allows the user to subdivide some, or all of the beam elements into equivalent subelements. The `Subdivide` option of the element menu will call up the submenu:

The `Specific` option allows the user to specify a single element, by number, which is to be subdivided. Thereafter, the user would enter in how many subelements are required. BSF would then automatically divide the specified element up; this would naturally change the numbering of beam elements. For example, if element 3 in the example beam was to be subdivided into 4 elements, the former beam of 4 elements of length 1, 1.5, 0.5 and 2 metres would become an equivalent beam of 7 elements of lengths 1, 1.5, 0.125, 0.125, 0.125, 0.125, and 2 metres. Element 3, originally 0.5 metres in length, has been subdivided into 4 elements, each of length 0.125 metres. The screen display would appear as in figure 7.17.

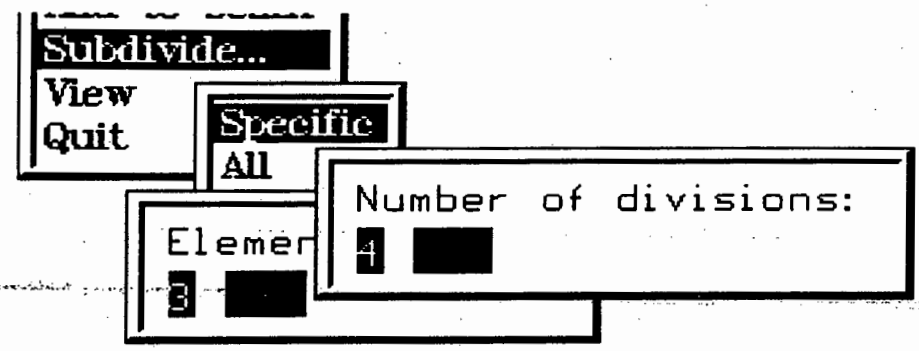


Figure 7.17 : Example showing subdivision window

The **All** option will allow all the existing elements (4) to be divided into a constant specified number of subelements.

In both the **Specific** and **All** options, the user cannot specify subdivisions which would result in there being more elements than 155 (imposed limit for **BSF** program). In such an event, an error message would be displayed, and the subdivision request ignored.

7.4.3.8 Viewing Element Information : Option 'View'

At any time, the user may use this option to view the beam element information currently residing in memory. The program will display the parameters of element 1 in a viewing window. The user can use the ↓ and ↑ keys to move through the various elements in the beam, one at a time. The **PgDn** and **PgUp** keys move the display through the list of elements 5 at a time. The figure 7.18 shows the display for element 4 of the example beam problem:

Element 4:

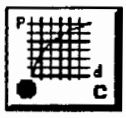
Length (m)	=	2.000
Modulus, E (kPa)	=	200000000
Moment Inertia, I (m4)	=	1.94300e-05
Depth of Section (m)	=	0.200
Width of Section (m)	=	0.100
k Value (kPa)	=	0
L Shear Force (kN)	=	-250.220
R Shear Force (kN)	=	-0.220
L Moment (kNm)	=	-0.073
R Moment (kNm)	=	0.073
Left End	=	Internal
Right End	=	Free/Spring
Spring k (kN/m)	=	0.00

Figure 7.18 : Example showing the 'viewing' window for element parameters

7.4.3.9 Quit Element Menu : Option 'Quit'

This option will close the `element` menu and return the user to the icon-level where the element icon will 'pop out' to its original unselected shape. This invites the user to select any of the six icons in the main display header of figure 7.11.

7.4.4 Curve Data

ICON: 

MENU: 

All the options for defining pressure-displacement of soil or stress-strain curves of beam material can be accessed with the selection of the `Curve` icon. As displayed on the `Curve` icon, the 'c' or 'C' key can be used to select the icon. Once the `Curve` icon has been activated, the `curve` pull-down menu will be displayed.

7.4.4.1 Defining Curves : Option 'New'

The **New** option is used to define a piecewise linear curve of n points ($n <= 20$), where each point is an (x, y) pair of values representing either (pressure, displacement) or (stress, strain). Note that beam material stress-strain curves must be defined by using only 2 points; these points define the elastic modulus, inelastic modulus and yield stress values. Two points define two slopes (elastic and inelastic moduli) and the stress of the 2nd point is the yield stress. In the case of Grade 300W steel (see figure 4.4), the beam material has no inelastic range, then the two points must be arranged to define a single slope (the elastic modulus).

Once the option has been selected, an Entry Window will open up and prompt the user for the number of points which will be used to define the curve - this number cannot be greater than 20 for memory limitation reasons.

With all curves, the first point (0,0) is implicit. Thus in the example curve shown in figure 7.10A, the curve definition would require 4 (pressure, displacement) points : (120 kPa, 0.2mm), (280, 0.6), (400, 1.2), and (500, 2). Since there is no inelastic range for Grade 300W steel, the 2 points which define the beam material must lie on the slope of the elastic modulus, the second point being the yield stress for the steel. ie `BSF` does not allow the single point (300, 0.0015) to define this modulus since it insists that 2 points are used for the material stress-strain curve.

Once the user has specified the number of points which will be used to define the piecewise linear approximation to the desired curve, the pairs of values are read in. Standard units are kPa for pressure and millimeters for displacement. If the curve is a stress-strain curve (which is required for the beam material), then stress is entered in kPa and strain has no dimension.

An automatic feature of the curve entry procedure is that for any parameter value i, be it pressure, displacement, strain or stress, the magnitude entered must be greater than that for the preceding value i-1. ie. for any two consecutive points, $(\alpha_1,ß_1)$ and $(\alpha_2,ß_2)$, the condition $(\alpha_2 > \alpha_1)$ and $(ß_2 > ß_1)$ must be satisfied. This feature ensures that any curve entered will tend toward a maximum value at the final point entered, avoiding the possible entry of a

curve where the peak value is not the final point. If such a curve were entered, the inelastic solution algorithm used by BSF would be vulnerable to instability when a problem is analysed.

7.4.4.2 Saving A Defined Curve : Option 'Keep'

As with saving element information described in section 7.4.3.2, the **Keep** option prompts the user for the name of the file to which the curve-defining points must be saved. Like the element data file, the file format used by BSF is an ASCII text file, for which the user can specify a full path name.

If the user does not save the data curve to a file, then the curve cannot be attached to a descriptor described in section 7.4.4.3 below. Thus the **New** option alone is useless for analysis purposes - it simply allows the user to define a new data curve. The **Keep** option must be used in conjunction with the **Get** option, described below, in order to link the defined data to desired soil or beam material.

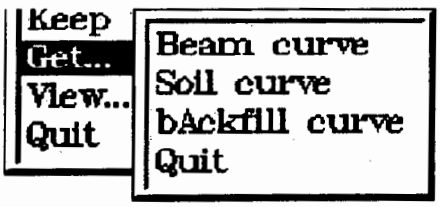
For the rest of this chapter, it will be assumed that the user has saved the two curves of figures 7.10 A and B, each of 4 points, to the curve data files : `soil1.dat` and `soil2.dat` respectively. In addition, the Grade 300W steel stress-strain curve is assumed to be defined in the data file `steel.dat`.

7.4.4.3 Retrieving A Curve File : Option 'Get'

The **Get** option works the same way with curve files as with element files - the user specifies the data filename which holds the points defining a given curve. The points for that curve are then read into the computer's memory. However, unlike the element data which defines a single beam, several curves are used by BSF:

If a *direct* (non-iterative) analysis (see section 7.4.5) is required, no soil curves are needed, nor is a beam material stress-strain required. However, if an *indirect* (iterative) solution is required, then two soil curves must be defined prior to an analysis : the in-situ soil curve, called the *soil curve*, and the *backfill soil curve* (so termed because of the influence of soldier piles on the design of BSF). If a problem has only one soil layer, then the user can define a very deep *backfill* layer, and a fictitious second layer to satisfy the presence of an in-situ soil layer (the program always expects two soil layers, even if only one will be used). When the analysis proceeds, the program will, naturally, never need to refer to the *soil* curve information since the backfill layer is too deep. This will correctly model a single-layer foundation.

Therefore, once the **Get** option is selected from the menu, the following submenu will be displayed



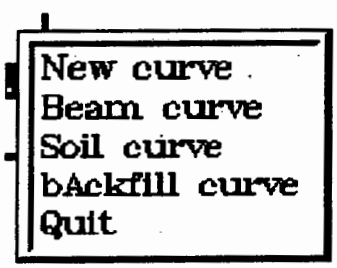
which allows the user to specify whether the curve to be retrieved will be associated with the backfill, in-situ, or beam material. Having selected either the `beam`, `soil`, or `backfill` curve descriptor, the program will prompt for the name of the curve data file. This file is then read in, and the data associated with the chosen curve descriptor.

Note that while the `Get` option requires the curve data being read be assigned a descriptor, any curve can be read in to any descriptor. ie when the user enters in curve data points as described in section 7.4.4.1, the data is not specifically linked with any type of curve at that stage. It could be describing a pressure-displacement curve for the backfill material, or even a stress-strain curve for the beam material. Only when the data is retrieved it is linked with a specific descriptor. It is thus possible to read in any curve as being representative of any material in the problem. The same curve could even be read twice, say for the backfill and in-situ material.

7.4.4.4 Viewing Curve Points : Option 'View'

If the user wishes to view the points which currently define any of the beam, soil, or backfill curves, the `View` option offers two alternatives : the information can be displayed graphically, or in a tabular format. In the former case, a Graphic Window (see section 7.3.6) is used, while the latter uses a Table Window (section 7.3.5).

Once the `View` option has been selected, the following submenu is displayed to allow the user to specify which curve data is to be viewed:



The `New` curve refers to data points entered in with the `New` option from the main `curve` menu. Even if new points have been saved with the `Keep` option, this data still resides in memory until the user defines a new set of data points. Thus the memory-resident information can be viewed to confirm correct entry, and to detect any anomalies that may be present in the data points.

Once the curve has been specified, another submenu 

pops up to ask whether the information should be displayed in graphical or tabular form. Figure 7.19 shows the soil curve 1 of figure 7.10A in graphical form, while figure 7.20 shows the same information in tabular form:

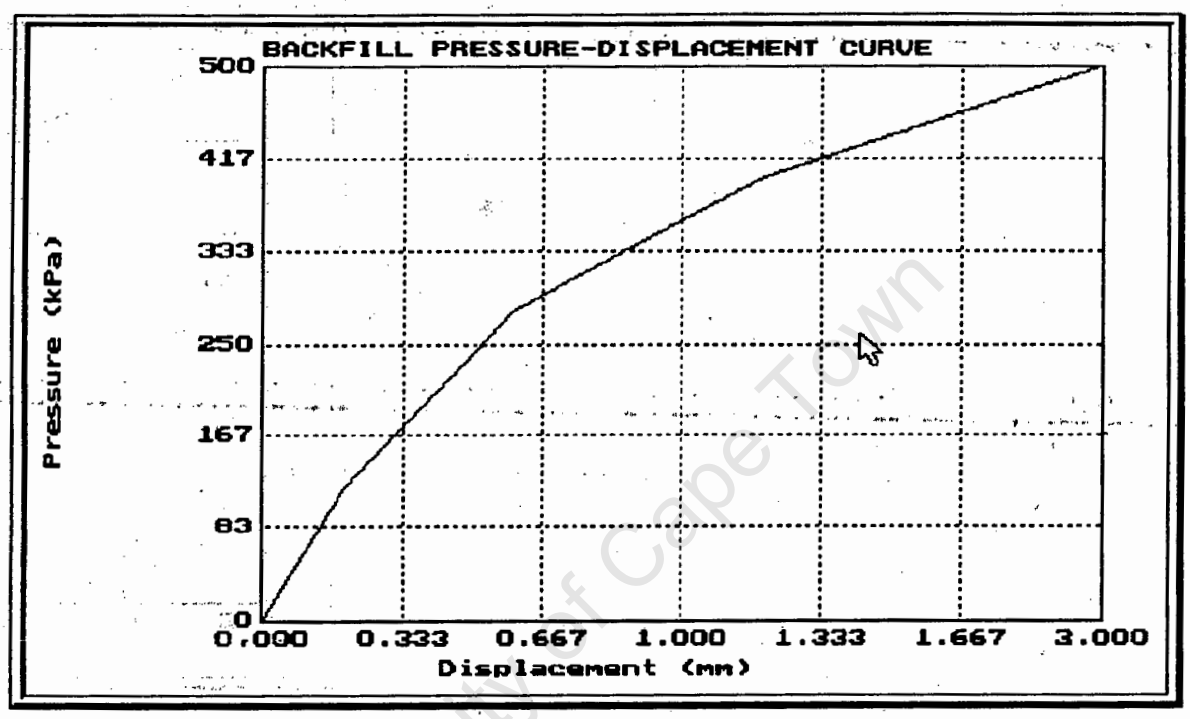


Figure 7.19 : Graphical display of soil curve 1 in example problem

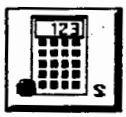
Entry	Pressure (kPa)	Displ (m)
001	120.000	0.0002
002	280.000	0.0006
003	400.000	0.0012
004	500.000	0.0020

Figure 7.20 : Tabular display of soil curve 1 in example problem

7.4.4.5 Quit Curve Menu : Option 'Quit'

This option will close the `curve` menu and return the user to the icon-level where the curve icon will 'pop out' to its original unselected shape. This invites the user to select any of the six icons in the main display header of figure 7.11.

7.4.5 Solving The Defined Problem

ICON: 

MENU:

The analysis options for solving the beam-soil problem defined in section 7.4.3 and 7.4.4 is accessed with selection of the `Solve` icon. As displayed on the `Solve` icon, the 's' or 'S' key can be used to select the icon. Once the `Solve` icon has been activated, the `solve` pull-down menu will be displayed.

7.4.5.1 Elastic Analysis : Option 'Direct'(Non-iterative)

The `Direct` option is used to obtain an elastic analysis of the problem currently defined using sections 7.4.3 and 7.4.4. The term 'direct' is used because no iterative algorithm is involved - the solution being obtained by directly solving equation (3.12) in chapter 3. The program routines for the analysis were discussed in section 6.3.

No curve data, either for the backfill or in-situ material, is required; nor is a stress-strain curve required for the beam material since the analysis will be an elastic one.

As explained in section 7.4.3.1, the value of the k parameter in the beam elements will dictate the soil response, should one be required. Zero values for k mean that no soil is modelled; positive values model the subgrade modulus of an elastic single-layer soil foundation beneath the beam.

Once the `Direct` option has been selected, the solution will be calculated by the program (which will appear to pause while the calculations are in progress) whereafter the user can use the options described in sections 7.4.6 to 7.4.8 to examine and save the output of the analysis.

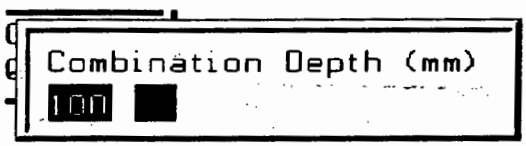
7.4.5.2 Inelastic Analysis : Option 'Iterative'

An inelastic analysis of the defined problem requires the iterative analysis algorithm (see pseudo-code in section 6.4). Various options which control the nature of the iterations and key analysis parameters are presented in section 7.4.5.3 below. The user will usually ensure that these options are set to the required values before requesting an inelastic analysis with the `Iterative` option.

Once the `Iterative` option is selected, the program will check to see that there is both a backfill and an in-situ curve defined and stored in memory (see section 7.4.4.3), as well as a stress-strain curve for the beam material. If any of these three curves are not defined and

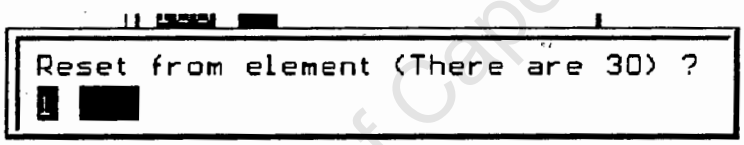
in memory, then the program will display an error message stating which curve(s) still needs to be retrieved into memory, and then return the user to the `solve` menu.

If the curves are in memory, then the program proceeds by prompting the user for the *combination depth* (see section 5.2), in millimetres, which controls the interaction between the two soil layers: the *backfill* layer (upper layer) and *soil* layer (lower layer).



Should the user not wish to model two soil layers in the problem, the *combination depth* could be set to a very large value. This would rule out any effect from the lower (*soil*) layer, effectively modelling a single soil layer supporting the beam even though two soil layers have been specified (this is compulsory in BSF).

After specification of the combination depth, the program prompts for the beam element at which the *reset process* will begin (element numbering starts at '1' from the left end of the beam).



The reasons for this reset process are given below:

When an analysis is completed, the displacements, soil modulus, beam modulus, and foundation response values are kept in memory until a new analysis is requested. Therefore, if the user wishes to retain the output values of an analysis *i* as partial input to a new analysis *i+1*, then the re-initialisation of all parameters would prevent such an intention. However, by requesting the element reference number from which such reset/initialisation will be performed, BSF allows the user to preserve the values from the *i* analysis for use in analysis *i+1*. Although such a feature will rarely be used, it has been built into BSF because of anticipated use of the program in further research. For the example problem of figure 7.9, this feature is not required, and so the user can specify that the reset option must begin from element 1 - the left-most element of the defined beam.

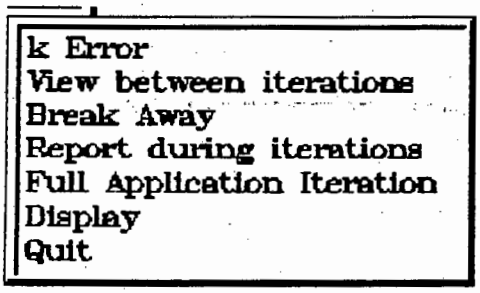
After initialising the elements, the program employs the algorithm of section 6.4.7 to arrive at an inelastic solution of the defined problem, where both inelastic soil and beam behaviour are considered, as well as possible plastic hinging in the beam. During the iterative process, the user can abort at any time by pressing the **ESC**ape key. A slight delay may occur before this happens since the program only reacts to the ESC key after finishing the current processing task.

The interaction which takes place with the user during the iterative procedure is controlled

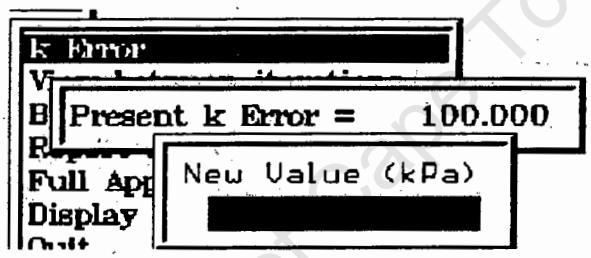
by the options in section 7.4.5.3.

7.4.5.3 Solution Menu : 'Options'

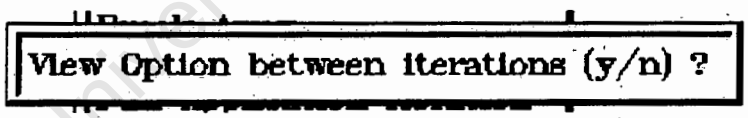
Selection of this option results in the following submenu being displayed:



Selecting the `k_error` option leads to the program prompting the user for the magnitude of the error (kPa) to be used in the algorithm described in section 6.4.3:



The `View between iterations` option prompts the user if this feature is to be turned 'on' or 'off' for the iterative solution:



If the user replies 'yes' (by pressing 'y' or 'Y'), then after each iteration, the program will display, in a Graphic Window, the following:

- *k* values for all elements (see section 7.4.6.1)
- Foundation reaction (*UDL*) beneath all elements (see section 7.4.6.3)
- Displaced shape of the beam (see section 7.4.6.6)

The data is presented as an X-Y graph with the distance along the beam being marked off on the X-axis, and the corresponding parameter value on the Y-axis.

The `Break away` option, if set 'on' will result in elements which have displaced a positive amount (ie. the element has moved up and away from the soil interface) will have their *k*

parameters set to 0 and not the elastic modulus of the *backfill* material (which is usually the case). The reason for normally assigning the elastic modulus is discussed in detail in section 4.2. However, this option has been built into BSF for future research possibilities.

The **Report during iteration** option, if 'on', will result in the *iteration window* being displayed and updated during each iteration:

```
The beam is 11.000 metres long.
It has been discretised into 72 elements.
-------------------------------------------
Iteration 2 (72 elements) [ESC exits]
0 Excessive SOIL displacements
0 Excessive BEAM displacements
Max Downward displ = -6.727 (mm) [26]
Max Upward displ =  4.506 (mm)
```

The above example of an iteration window shows

- the number of the current iteration, followed by the present number of elements in the beam (hinge formation will lead to more elements, 1 per hinge).
- whether any excessive displacements have occurred (these occur when an element displacement is detected which is greater than the largest deflection given for the soil curve in question).
- the current peak displacements (in positive and negative directions) which have been detected for the present iteration.

The 'excessive deflection' feature indicates to the user that the beam is deflecting an amount greater than defined by the user for a particular soil curve. The user should thus contemplate redefining that soil curve such that the largest displacement reading is greater than the largest beam deflection anticipated.

The 'peak displacement' information informs the user of the degree of displacement being measured, at which point the user can determine if the solution is proceeding as expected or not. If not, the user can abort the iterations by pressing the **ESC**ape key, a reminder of which is also displayed in the window. The reporting option will also display an Acknowledgement Window if a plastic hinge has formed in the beam.

The reading in [] to the right of the maximum downward displacement is the number of beam elements that have undergone a parameter change (either E or k) during the iteration shown.

On occasion it is possible that a defined problem will take a while to reach equilibrium if 100% of the defined loading on the beam is used from the very first iteration. Consider a small beam under very heavy loading on a two-layer foundation. The initial deflections will be very large, causing large reactions from the foundation which might 'propel' the beam away from the soil interface. This would lead to many of the elements having their k parameters, as discussed above, set to the elastic modulus of the *backfill* layer. To avoid this

situation, and to facilitate a smoother simulation of beam loading, and consequent solution, the user can specify a certain iteration number, say iteration number 5, at which point the applied loading will have reached 100% of the defined magnitude. In the preceding iterations, the defined loads on the beam will be linearly increased so as to arrive at 100% of their defined magnitudes by the specified iteration. The iteration at which 100% loading will be reached, called the *full application iteration*, can be set with the **Full application iteration** option. This feature models a gradually increasing load, as might often occur in reality, where loads are never truly instantaneous.

Finally, the **Display** option will produce the following window showing the current status of the various options:

```
k Error =     100.000 (kPa)
View Option = OFF
Break Away = OFF
Reporting = ON
Full Application =    1 iter
```

7.4.5.4 Solving The Example Beam Problem

In this section, the example problem of figure 7.9 is analyzed using the inelastic algorithm.

However, before the iterations can in fact begin, the user has to determine the *combination depth* for the two soil layers. This is done as follows:

Step 1
The soil curves (here assumed located in the files `soil1.dat` and `soil2.dat`) must be read in for the *backfill* and *soil* material respectively. The stress-strain curve for the beam material (Grade 300W steel) must be read in (assumed file here is `steel.dat`) and assigned to the *beam* 'curve'.

Step 2
A new beam is defined using the input methods of section 7.4.3.1. This beam is identical to that shown in figure 7.9 except that the ends are *free* ends. Since there is no loading on the beam, it can be defined using 1 element 5 metres long. As with `beam.dat`, the user can use the **Self Weight** feature of section 7.4.3.4 to apply the self weight of 0.22 kN/m to the beam. This information can then be saved to the data file `freebeam.dat` with the **Keep** option of section 7.4.3.2.

Step 3
As described in section 5.2, a line load of W kN/m must be applied to the beam. This is done by also using the **Self Weight** option (for efficiency), where the load specified is

W. Assume the first trial value of *W* is 50 kN/m. The beam information should not be saved to `freebeam.dat` since, if the first trial load of 50 kN/m is not correct, the user can save a lot of time by simply respecifying the `freebeam.dat` file, and applying a different *W*, without having to go through all the stages of defining the beam again.

Step 4

The user would then subdivide the beam into, say, 30 elements using the **`Subdivide, All`** feature of section 7.4.3.7. This is done by specifying 'all' beam elements to be subdivided into 30 elements each. Since the free beam was defined as a single element, this divided into 30 subelements gives 30 elements for the beam (the user could also have achieved this by specifying that element 1 be subdivided into 30 elements).

Step 5

The user would select the `solve` menu and the **`Iterative`** option. In response to the prompt for the *combination depth* value, the user could enter 100mm to ensure no influence from the *soil* layer. As described in chapter 5, the combination depth is determined by imagining that the second soil layer does not exist. The *reset* process would be set to start at element 1.

Step 6

Once the iterative solution is complete (on this occasion it would take only 1 iteration), the user would select the `view` icon (described in section 7.4.6) and the **`Foundation R`** (foundation response) option. A graphical window display (selected by choosing the **`Graph`** option display) of the response profile beneath the beam would reveal a peak response of approximately 29 kN over a length of about 5 m, as shown in figure 7.21.

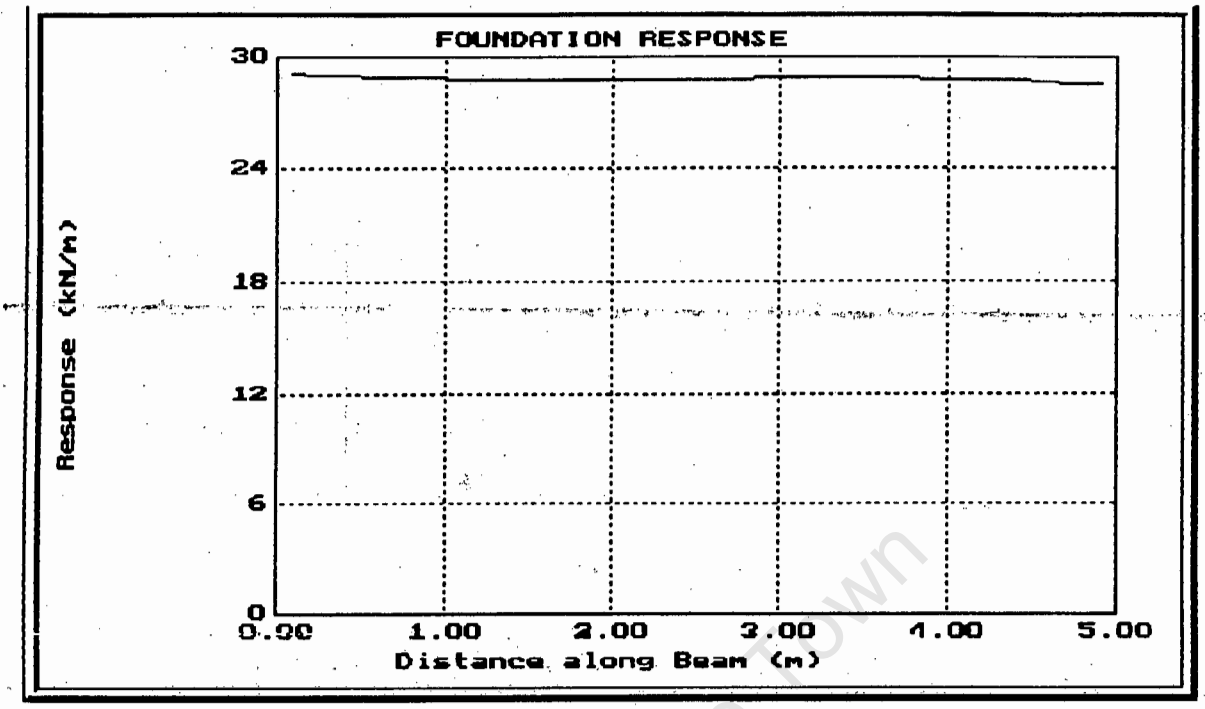


Figure 7.21 : Foundation response of example problem associated with step 6

Using elastic methods of determining vertical stress at a depth Z (see Appendix A), the stress from this peak of 29 kN over 5 m, at depth 1.5 m (depth to the actual interface between the two soil layers - if the second layer were present) would be:

$$\sigma_{zi} = 2 \times \frac{J}{1.5} \times \frac{29}{0.1}$$

$$J = 0.319$$

$$\therefore \sigma_{zi} = 123\ kPa$$

Step 7

To determine if this vertical stress is sufficient to cause a 'significant' displacement, and hence response, from the second soil layer, a *significant stress* must be calculated for the second layer. This is done by multiplying the peak 'elastic' stress (ie. assuming the first linear portion of the curve is the 'elastic' range) for this material (120 kPa) by the ratio of the chosen significant displacement to the displacement associated with the peak elastic stress. Assuming the smallest significant displacement of the second layer is 0.5mm, the stress which would cause such a deflection is clearly

$$significant\ \sigma = \frac{0.5}{1} \times 120\ kPa$$
$$= 60\ kPa$$

Step 8

Since the 123 kPa of step 7 is significantly greater than 60 kPa, the initial 50 kN/m chosen for *W* was too high. Reading in the `freebeam.dat` file again, and repeating steps 3 to 6 with a new value for *w* of 25 kN/m would lead to a vertical stress of 55 kPa at depth 1.5 metres; this is very close to the desired 60 kPa.

Step 9

The peak displacement for the $W = 25$ kN/m is obtained by again selecting the `view` icon and the **D**isplacements option. The peak displacement reading is 0.22mm. This is the desired *combination depth*.

Having determined the combination depth using the 'free beam', the user would read in the actual example beam with the `Get` option of section 7.4.3.3 and specifying the `beam.dat` file.

The user would then select the `solve` icon and check that the various analysis parameters are set as desired by selecting `Options, Display`. If satisfied, the user could then select the `Iterative` option. Assuming a *k_error* of 100 kPa (this is 0.2% of the k value of the elastic region of the soil curve in figure 7.10A), the example analysis will take 2 iterations to complete if the all four elements are subdivided into 25 subelements (making a total of 100 elements). The output of this analysis is covered in sections 7.4.6 through to 7.4.8.

Note that the point load applied to the left end of element 4, namely -400 kN, does not lead to any plastic hinge in the beam. For a plastic hinge to form, the user could increase this load (by using the procedure described in section 7.4.3.5) to -500 kN. If this is done, a plastic hinge will form beneath the concentrated load. In sections 7.4.6 to 7.4.8, the solutions of these two analyses will be considered, the former being referred to as the '400 kN solution', the latter as the '500 kN solution'.

7.4.5.5 Quit Solve Menu : Option 'Quit'

This option will close the `solve` menu and return the user to the icon-level where the solve icon will 'pop out' to its original unselected shape. The user can then select any of the six icons in the main display header of figure 7.11.

7.4.6 Viewing The Solution

ICON:

MENU:

The options for viewing the output information from an elastic (direct) or inelastic (iterative) analysis can be accessed with selection of the `View` icon. As displayed on the `View` icon, the 'v' or 'V' key can be used to select the icon. Once the `View` icon has been activated, the `view` pull-down menu will be displayed.

All the options on the view menu are output parameters which can be viewed in a Table or Graphic Window. Upon selecting a particular option in the menu, the program will ask the user for the format of display required : **`Table`** or **`Graph`**.

The sections 7.4.6.1 through 7.4.6.6 demonstrate the view display options by using the '400 kN' and '500 kN' analysis output. While the Graphic Window format is more informative when viewing a solution, in instances where the user needs to know the exact output parameter value, the Table format can be used to get the desired value. Most parameters are tabulated to 3 decimal places of accuracy.

7.4.6.1 The k Profile

The k Profile is a display of all the beam elements' *k* parameter (subgrade modulus) values. The figure 7.22 shows the k profile of the '400 kN' solution:

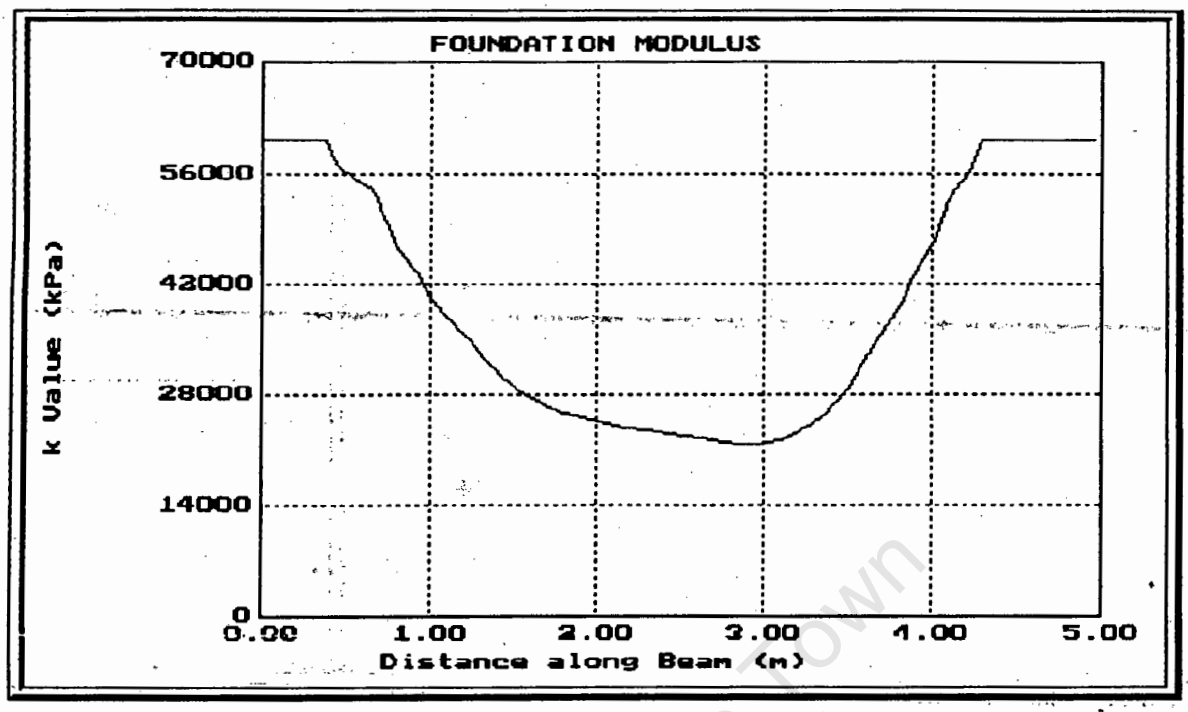


Figure 7.22 : k profile of '400 kN' solution of the example problem

The slight dents in the curved portion of the profile are due to the double soil layer effect; most of the beam has been displaced such as to get a response from the lower layer as well. The table in figure 7.23 shows the same information but in tabular form:

Element	Distance	k (kPa)
011	0.350	60000
012	0.383	60000
013	0.417	57846
014	0.450	56989
015	0.483	56305
016	0.517	55751
017	0.550	55297
018	0.583	54920
019	0.617	54605
020	0.650	54338

Figure 7.23 : Tabular display of k profile of '400 kN' solution

The second 10 beam elements (No's 11-20) are listed, along with the distance to the midpoint of each element (measured from the left end of the beam, in metres) and the *k* parameter value for the element (kPa). All *k* parameter values are tabulated as integer values.

7.4.6.2 The E Profile

The E profile is similar to the k profile above except that the elements' *E* (beam modulus) parameter is displayed. Since no hinges appear in the '400 kN' solution, the '500 kN' solution is used to illustrate the E profile feature. The program uses an *E* value of 25000 kPa to model a hinge. Thus the marked drop in the *E* value at 3.0 metres along the beam (from the left end) is the position where the hinge formed, as shown in figure 7.24.

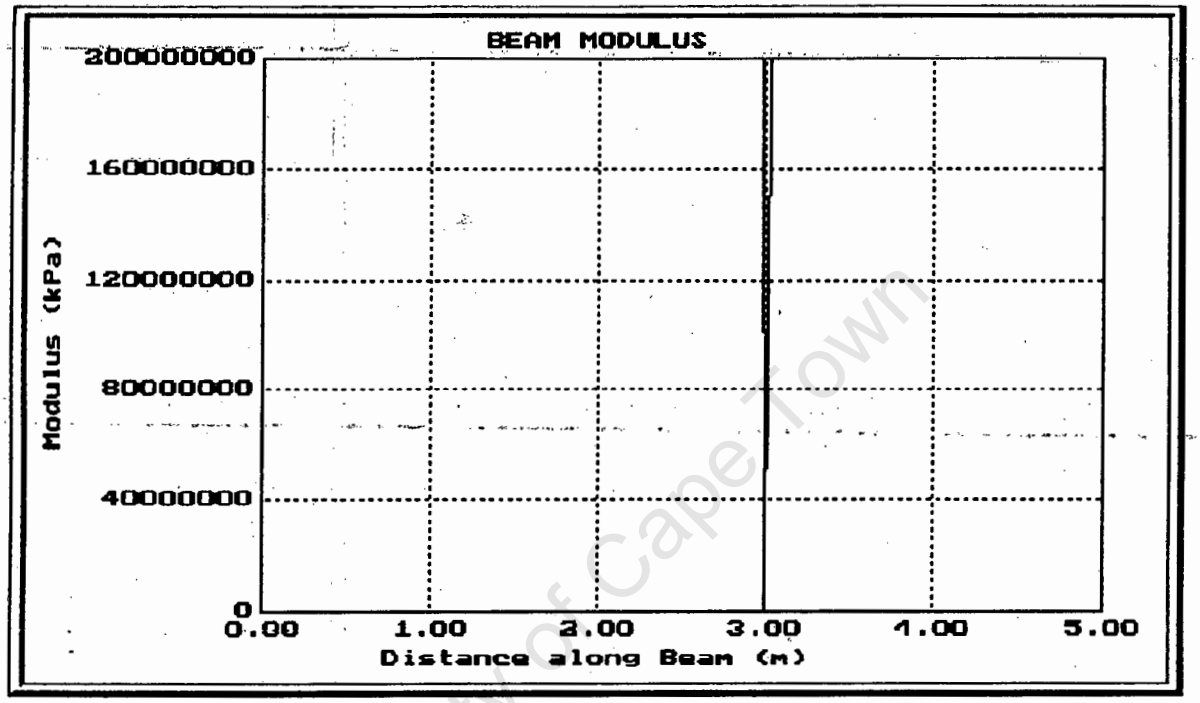


Figure 7.24 : E profile of the '500 kN' solution

The elastic *E* modulus of 200e6 kPa is constant in all the elements except for the one which hinged. The steel stress-strain curve used for the analysis is bilinear (had no inelastic range). Therefore most of the graph line is not clearly visible because it lies at the top of the display window along the 200e6 kPa level.

7.4.6.3 The Foundation Response

The Foundation Response option is selected by choosing the `Foundation R` option from the `view` menu. The figure 7.25 below shows the Graphic Window display of the foundation response beneath the example beam of 7.4.5 where the applied point load was 400 kN.

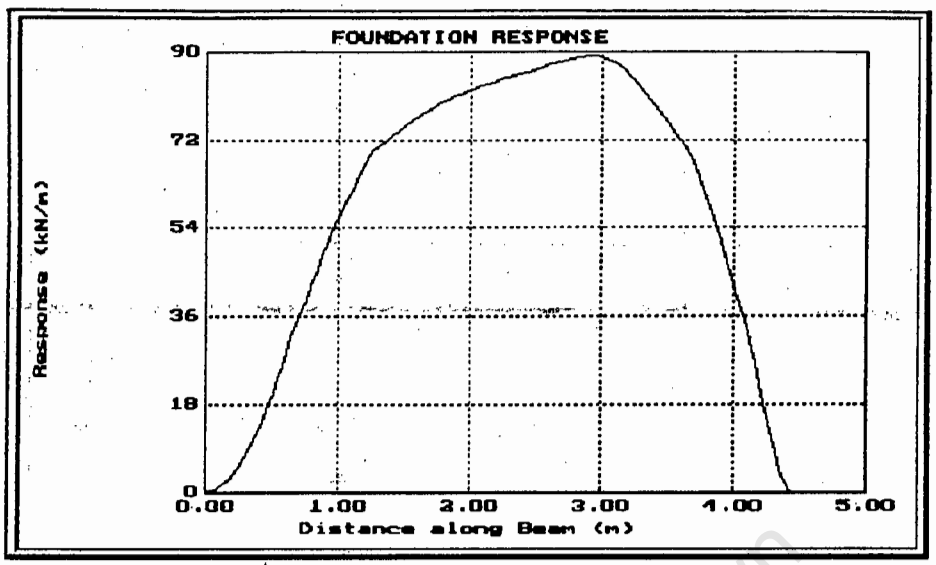


Figure 7.25 : Foundation response profile of '400 kN' solution

7.4.6.4 Bending Moments

The bending moments are solved using the `solve_internal_forces` routine of section 6.3.2. The bending moment diagram in figure 7.26 is for the '400 kN' solution of the example problem:

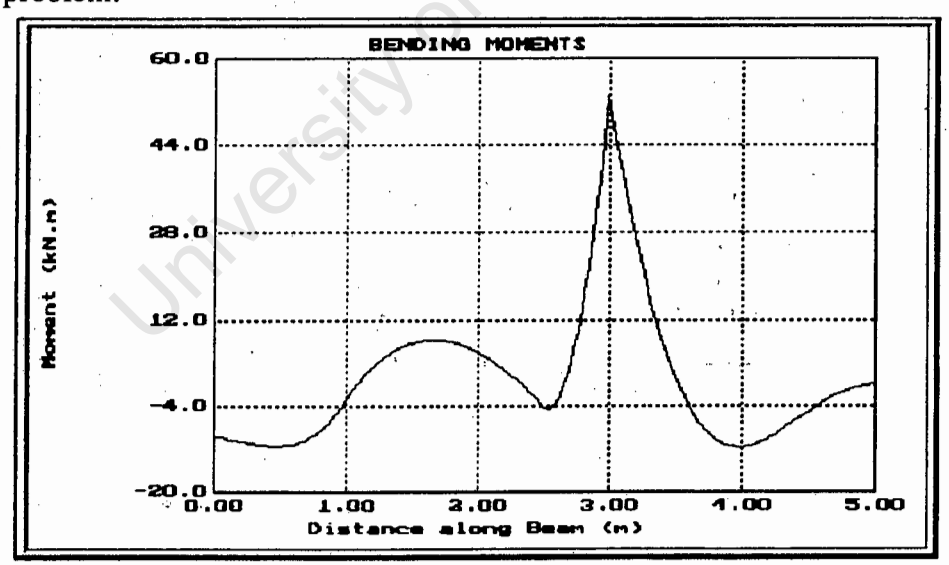


Figure 7.26 : Bending moment diagram of '400 kN' solution

The peak moment at 3.0 metres along the beam, beneath the point load of 400 kN, shows the beam beginning to approach a plastic hinge stress state (which occurs when the load is increased to 500 kN in the subsequent analysis).

7.4.6.5 Shear Forces

The shear forces are also calculated using the `solve_internal_forces` routine. Figure 7.27 shows the shear forces for the '400 kN' solution:

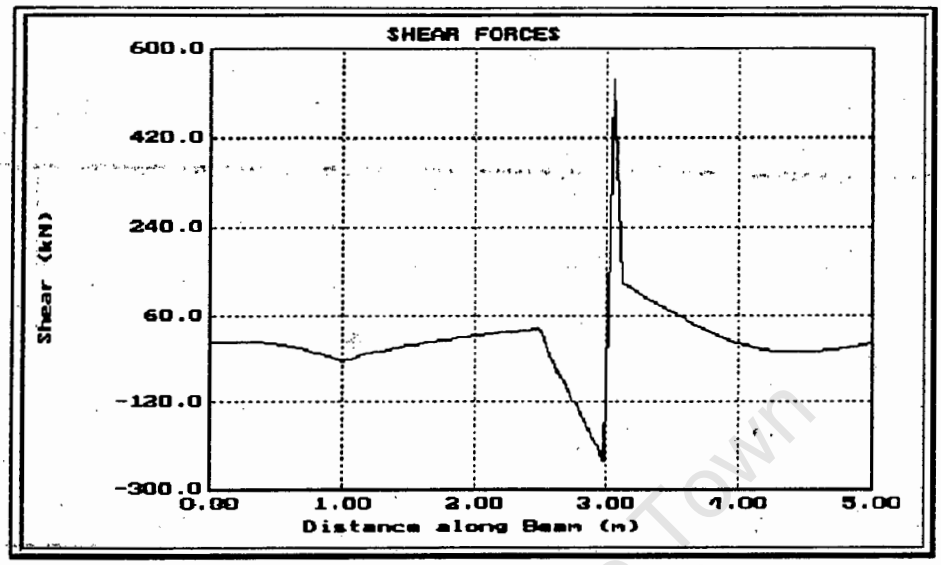


Figure 7.27 : Shear forces of the '400 kN' solution

7.4.6.6 Beam Displacements

The Graphic Window display of beam displacements shows the deflected shape of the beam. The Tabular Window format shows the element displacements and rotations (radians, positive angle is anticlockwise) for the first 9 beam elements. Figure 7.28 shows the displaced shape of the example beam after the '500 kN' analysis.

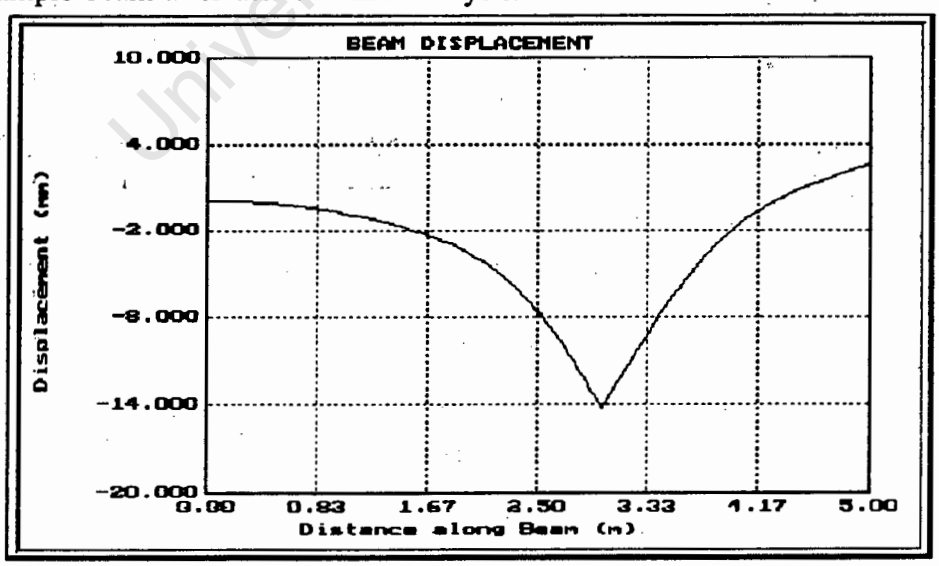


Figure 7.28 : Displaced shape of example beam ('500 kN' solution)

Note the effect on the displaced shape caused by the hinge at 3.0m. The original beam is always at the 0 mm displacement level. The displacements and rotations, for the first 9 elements of the same solution, are listed in tabular form in figure 7.29.

Element	Distance	Displace (mm)	Rotation
001-Left	0.000	0.000	0.00000
001	0.033	-0.000	-0.00003
002	0.067	-0.002	-0.00006
003	0.100	-0.005	-0.00010
004	0.133	-0.009	-0.00014
005	0.167	-0.014	-0.00018
006	0.200	-0.020	-0.00022
007	0.233	-0.029	-0.00027
008	0.267	-0.038	-0.00032
009	0.300	-0.050	-0.00037

Figure 7.29 : Displacements and rotations for '500 kN' solution

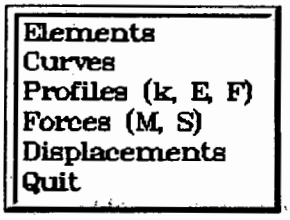
7.4.6.7 Quit View Menu

This option will close the `view` menu and return the user to the icon-level where the view icon will 'pop out' to its original unselected shape. This invites the user to select any of the six icons in the main display header of figure 7.11.

7.4.7 Saving The Solution

ICON: 

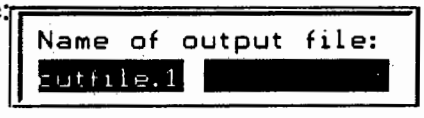
MENU:



The options for saving the output information from a direct (elastic; non-iterative) or indirect (inelastic; iterative) analysis can be accessed with selection of the `Outfile` icon. This icon shows an arrow pointing to a folder (or file) indicating a flow of *out*put information to a *file* - hence the term '*outfile*'. As displayed on the `Outfile` icon, the 'o' or 'O' key can be used to select the icon. Once the `Outfile` icon has been activated, the `outfile` pull-down menu will be displayed.

The different options on the menu indicate which output values will be written to an outfile specified by the user. The outfile is an ASCII text file to which the output information is written in a neat and presentable format. The outfile may be printed directly from the DOS command line, or can be incorporated into a word-processor document for further editing and inclusion in a report (for example).

Once the user has selected the output information to be saved to an outfile, the program will prompt the user for the name of the outfile:



The user-specified filename can, as before, include a full pathname.

In sections 7.4.7.1 through to 7.4.7.5, portions of typical outfiles are given to show the format of output data written to a file. The information of the example outfiles comes from the output information of the '400 kN' solution of the example beam problem (section 7.4.5.5).

Note that successive specification of a particular outfile results in the corresponding output data being appended to the file each time, not overwriting the information previously saved to the file. For example, if the user specified the filename `'solution.400'` for the output of the '400 kN' analysis of section 7.4.5.5 and then used the options of sections 7.4.7.1 to 7.4.7.5 to save output information to this outfile, all the output would be sent to `solution.400`, each option appending its data to the previous contents of the file. The fact that data is appended, and does not overwrite previous file contents, allows the user the option of building up a single outfile of information containing all the data from an analysis using sections 7.4.7.1 to 7.4.7.5.

7.4.7.1 The Beam Elements : Option 'Elements'

All the parameter values for the beam elements can be saved with this option. The output format is:

```
ELEMENT INFORMATION...

The beam length is 5.000 metres.

Element 1:
                length (m) = 0.033
                   E (kPa) = 200000000
                    I (m4) = 1.943000e-05
                 depth (m) = 0.200
                 width (m) = 0.100
                   k (kPa) = 60000
              force L (kN) = -0.004
              force R (kN) = -0.004
            moment L (kNm) = -0.000
            moment R (kNm) = 0.000
     Foundation udl (kNm) = 0.042
                  Left end = Fixed
                 Right end = Internal
       Displacement L (m) =   0.0000
       Displacement R (m) =  -0.0000
  Spring Constant (kN/m) =       0.00

Element 2:
                length (m) = 0.033
                   E (kPa) = 200000000
                    I (m4) = 1.943000e-05
                 depth (m) = 0.200
                 width (m) = 0.100
                   k (kPa) = 60000
              force L (kN) = -0.004
              force R (kN) = -0.004
            moment L (kNm) = -0.000
            moment R (kNm) = 0.000
     Foundation udl (kNm) = 0.210
                  Left end = Internal
                 Right end = Internal
       Displacement L (m) =  -0.0000
       Displacement R (m) =  -0.0000
  Spring Constant (kN/m) =       0.00

... etc up to the last defined beam element
```

7.4.7.2 The Soil And Beam Curves : Option 'Curves'

The backfill, soil, and beam curves can be saved for reference purposes with this option. The curves of the worked example in this chapter are shown in outfile-format below:

BACKFILL CURVE ...

POINT	STRESS (kPa)	DISPL (mm)
01	120.00000	00.20000
02	280.00000	00.60000
03	400.00000	01.20000
04	500.00000	02.00000

SOIL CURVE ...

POINT	STRESS (kPa)	DISPL (mm)
01	120.00000	01.00000
02	375.00000	03.70000
03	535.00000	06.00000
04	650.00000	09.00000

BEAM CURVE ...

POINT	STRESS (kPa)	STRAIN
01	280000.00000	0.00140
02	300000.00000	0.00150

7.4.7.3 The Material Profiles : Option 'Profiles (k,E,F)'

Using this option, the *k*, *E*, and *UDL* parameters of the beam elements can be saved in a more convenient format than that available in section 7.4.7.1 where they cannot be isolated from all the other element parameters. The distance measurements are to the midpoints of elements since these parameters are defined for the middle of beam elements.

ELEMENT PARAMETERS (kPa, kPa, kN/m)...

ELEMENT	DISTANCE	k VALUE	E VALUE	FOUNDATION
000	000.017	60000	200000000	0.0
001	000.050	60000	200000000	0.2
002	000.083	60000	200000000	0.5
003	000.117	60000	200000000	1.1
004	000.150	60000	200000000	1.7
005	000.183	60000	200000000	2.6
006	000.217	60000	200000000	3.7
007	000.250	60000	200000000	4.9
008	000.283	60000	200000000	6.3
009	000.317	60000	200000000	7.9
010	000.350	60000	200000000	9.8

...etc

7.4.7.4 The Internal Beam Forces : Option 'Forces (M,S)'

The bending moments and shear forces in the beam can be saved to an outfile. The first row in the outfile table is the data for the **left** end of element 1, hence the '01-Left' in the *Element* column. Since moments and shears occur at element nodes, the distance measurements are to the ends of beam elements:

ELEMENT FORCES...

ELEMENT	DISTANCE	MOMENT (kNm)	SHEAR (kN)
01-Left	000.000	-9.664	5.435
001	000.033	-9.845	5.443
002	000.067	-10.026	5.443
003	000.100	-10.207	5.424
004	000.133	-10.386	5.378
005	000.167	-10.563	5.289
006	000.200	-10.735	5.151
007	000.233	-10.900	4.941
008	000.267	-11.056	4.655
009	000.300	-11.199	4.275
010	000.333	-11.327	3.802

...etc

7.4.7.5 The Beam Displacements : Option 'Displacements'

The element end-displacement and end-rotation (radians) can be saved to an outfile with this option. As with bending moments and shear forces, the distance values are to the ends of beam elements (nodes).

DISPLACEMENTS (mm & rad)...

ELEMENT	DISTANCE	DISPLACEMENT	ROTATION
01-Left	000.000	0.000	0.00000
001	000.033	-0.001	-0.00008
002	000.067	-0.006	-0.00017
003	000.100	-0.013	-0.00026
004	000.133	-0.023	-0.00034
005	000.167	-0.036	-0.00043
006	000.200	-0.052	-0.00053
007	000.233	-0.071	-0.00062
008	000.267	-0.093	-0.00071
009	000.300	-0.118	-0.00081
010	000.333	-0.147	-0.00090

...etc

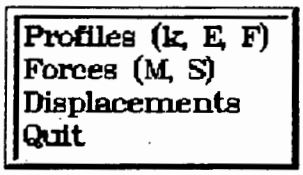
7.4.7.6 Quit Outfile Menu : Option 'Quit'

This option will close the `outfile` menu and return the user to the icon-level where the outfile icon will 'pop out' to its original unselected shape. This invites the user to select any of the six icons in the main display header of figure 7.11.

7.4.8 Exporting The Solution

ICON:

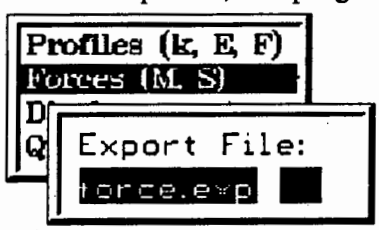
MENU:



The options for exporting the output information from an elastic (direct) or inelastic (iterative) solution can be accessed with selection of the `Export` icon. As displayed on the `Export` icon, the 'x' or 'X' key can be used to select the icon. Once the `Export` icon has been activated, the `export` pull-down menu will be displayed.

The different options on the menu indicate which output values will be written to an export file specified by the user. The export file is an ASCII text file to which the output information is written as *comma-separated* columns of values; each column contains one set of data values, for example : beam element displacements. The export file can then be exported to a spreadsheet where the data can be graphed with more extensive functions than available in the program `BSF`.

Once the user has selected the output information to be exported, the program will prompt the user for the name of the export file:



The user-specified filename, can as before, include a full pathname.

Successive specification of a particular export file will result in the corresponding output data overwriting the previous file contents. The first portion of a typical export file is given below to illustrate the comma-separated format. At the top of each column is a short description of the data to clearly identify the various columns when exported to a spreadsheet.

```
DIST,k (kPa),E (kPa), Foundation Response (kNm)
-----------------------------------------------
0.000,60000.008,200000000.000,0.042
0.017,60000.008,200000000.000,0.042
0.050,60000.004,200000000.000,0.210
0.083,60000.000,200000000.000,0.548
0.117,60000.004,200000000.000,1.060
0.150,60000.004,200000000.000,1.748
0.183,60000.004,200000000.000,2.616
0.217,60000.004,200000000.000,3.667
0.250,60000.000,200000000.000,4.904
0.283,60000.000,200000000.000,6.329
```

```
0.317,60000.000,200000000.000,7.944
...etc
```

As before, the **Quit** option will close the **export** menu and return the user to the icon-level where the export icon will 'pop out' to its original unselected shape. This invites the user to select any of the six icons in the main display header of figure 7.11.

7.5 PROGRAM VALIDATION

The addition of soil effect in the stiffness matrix method is new to structural design, and cannot be found in common structural textbooks on the subject matter. Since BSF can solve simple beam problems where there is no soil foundation, a simple beam problem (Brohn, 1984) has been solved to verify the correctness of the different routines and features in the program. The schematic for the problem is given in figure 7.30 below.

7.5.1 Example Of Structural Analysis Problem (No Foundation)

In using BSF to analyse the beam problem, the elastic option was used since there is no soil support - hence a *k* parameter of 0 kPa was assigned to all elements. The 4 elements in the beam data file were each subdivided into 30 to discretize the beam into 120 elements prior to analysis. This results in the smooth and accurate output displays that follow.

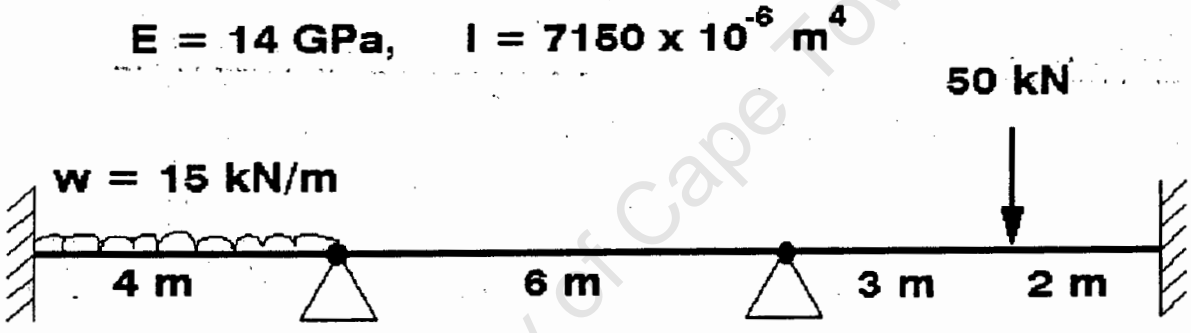


Figure 7.30 : Example problem in structural analysis

The following three figures show Graphic Windows from BSF displaying the beam displacement (figure 7.31), bending moment (figure 7.32), and shear force (figure 7.33) diagrams of the example problem.

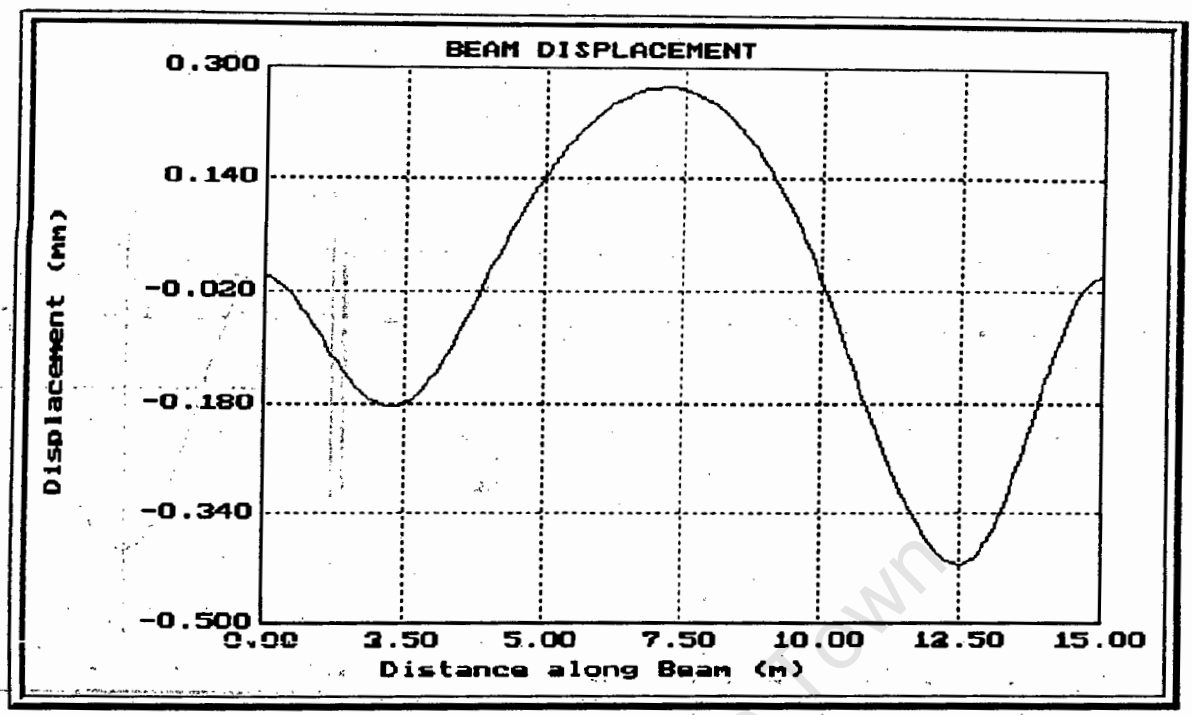


Figure 7.31 : Displacement of example beam

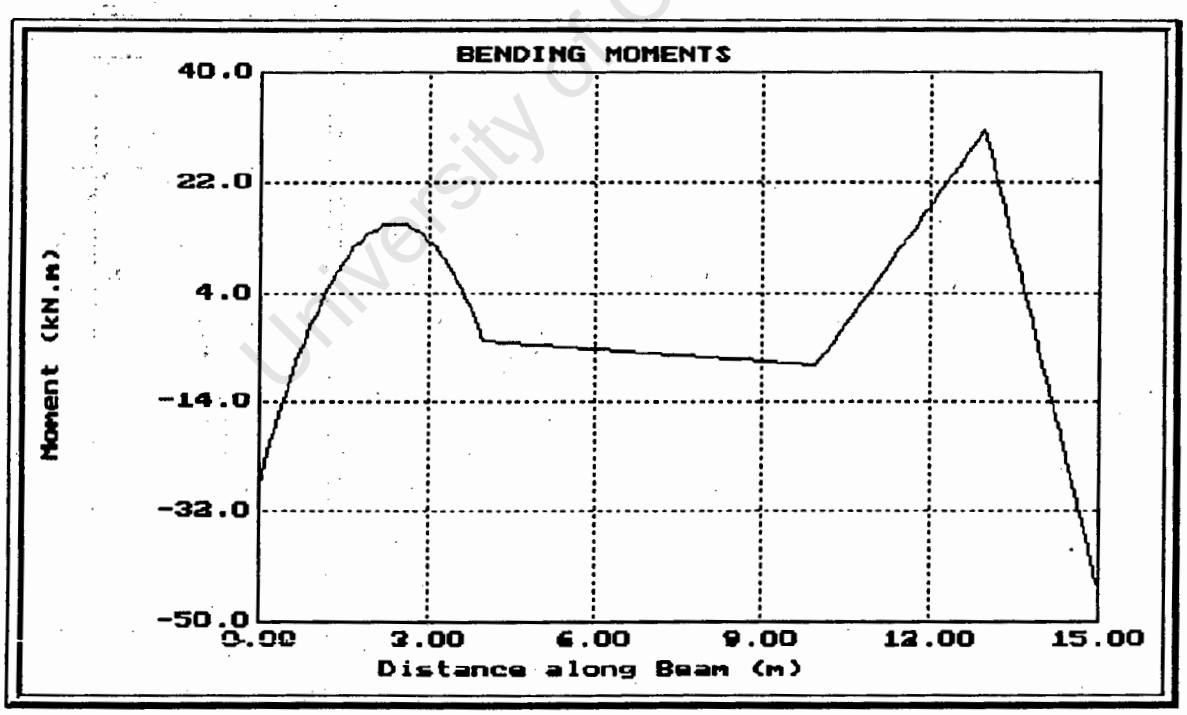


Figure 7.32 : Bending moments in example beam

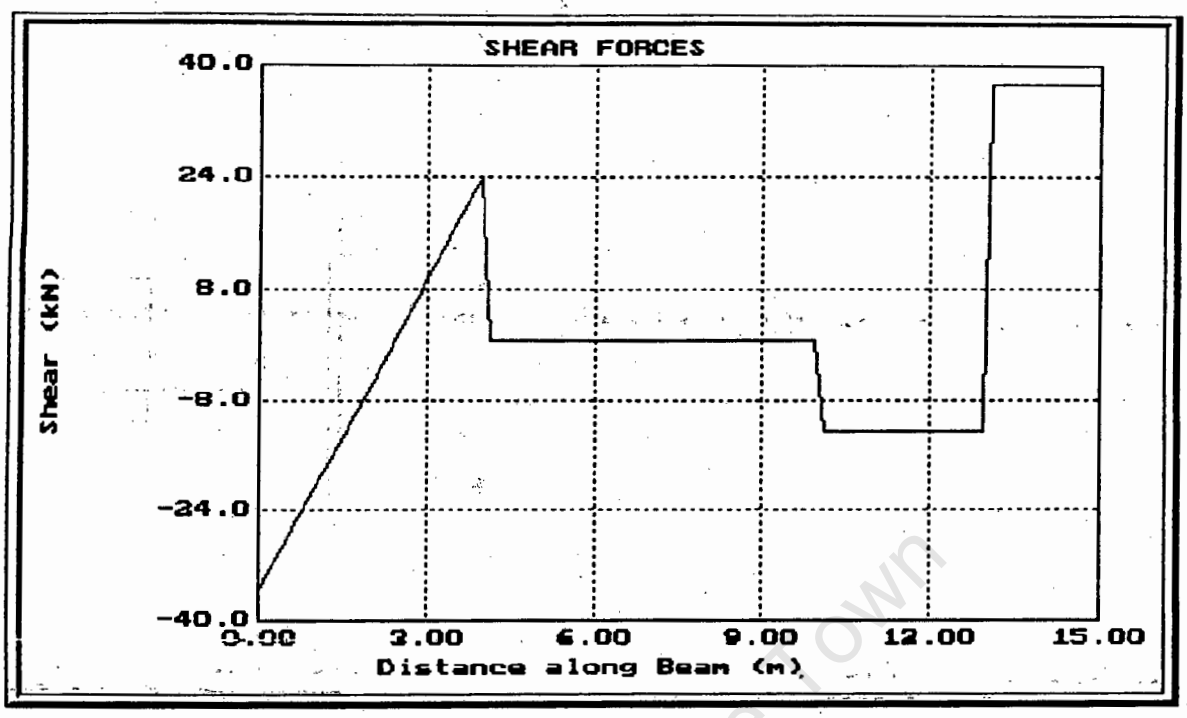


Figure 7.33 : Shear forces in example beam

7.5.2 Example Of Beam Foundation Problem

To determine the correctness of the code relating to foundation routines, the beam analysed by Hetenyi (1946, p. 47) was solved using BSF. This beam is shown in figure 7.34. The bending moment diagram, given in figure 7.35, agrees with that produced by Eisenberger et al (1985). The elastic analysis option was used to arrive at the solution since there are no soil curves - only a single k parameter value of 13791.86 kPa. The four elements of the beam data file were subdivided into 30 subelements. This meant the beam had been discretized into an accurate 120 elements prior to analysis.

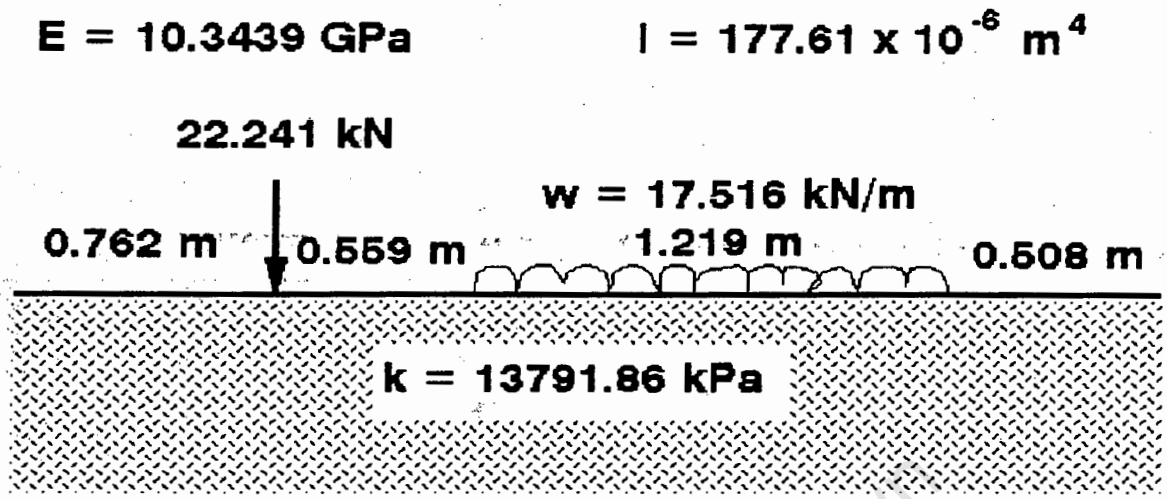


Figure 7.34 : Beam on foundation problem (Hetenyi, 1946, p. 47)

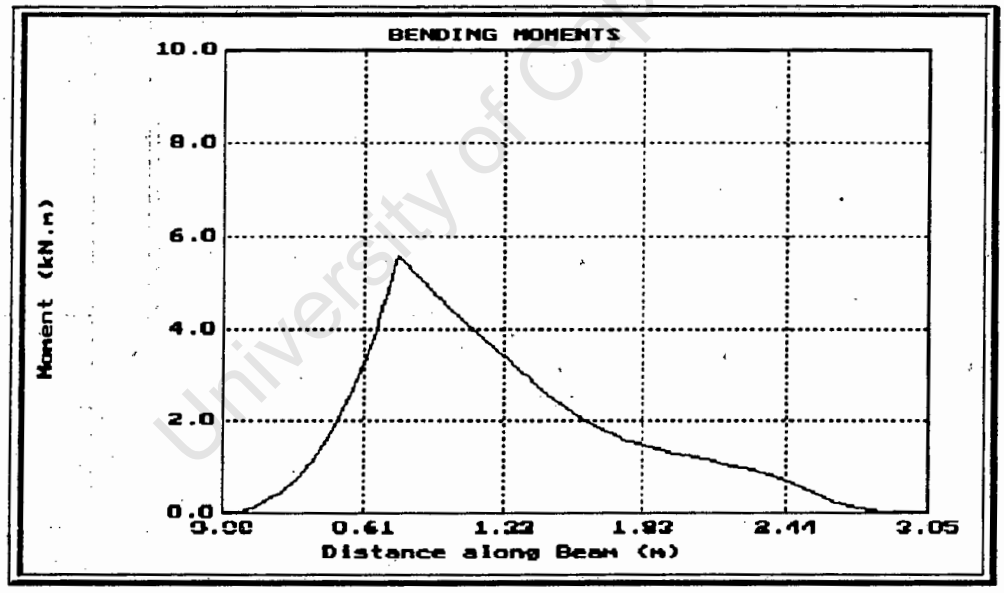


Figure 7.35 : Bending moment diagram of beam problem (Hetenyi, 1946, p. 47)

Other verification tests have been conducted by the author using extreme values, for example checking that error messages are displayed for invalid data entry, or that the user cannot enter in more information than the preset limit allows for.

CHAPTER 8

APPLICATIONS OF BSF : LABORATORY AND FIELD TESTS

This chapter aims to show the suitability of the BSF program in modelling and analysing actual laboratory and field tests (these tests are also used by Howie et al, 1994). In doing so, assessments can be made regarding the accuracy of the methods of this thesis, as well as the practical uses for such a program.

8.1 LABORATORY TEST ON HORIZONTAL BEAM

8.1.1 The Laboratory Test

A series of load tests on horizontal beams placed on a soil foundation was performed in the laboratory. The results of one such test are presented below.

The test was conducted using a 2,7m I-beam (100mm x 50mm, $I = 0{,}32 \times 10^{-6} m^4$) with free end conditions resting on a uniform sand which was compacted to a uniform dry density $\gamma = 15{,}2 kN.m^{-3}$. The natural moisture content of the soil was 6,75%. A plate load test conducted on the soil using a 300mm plate gave the pressure-displacement response shown in figure 8.1 with an ultimate soil bearing capacity of 294kPa at 2.3mm displacement.

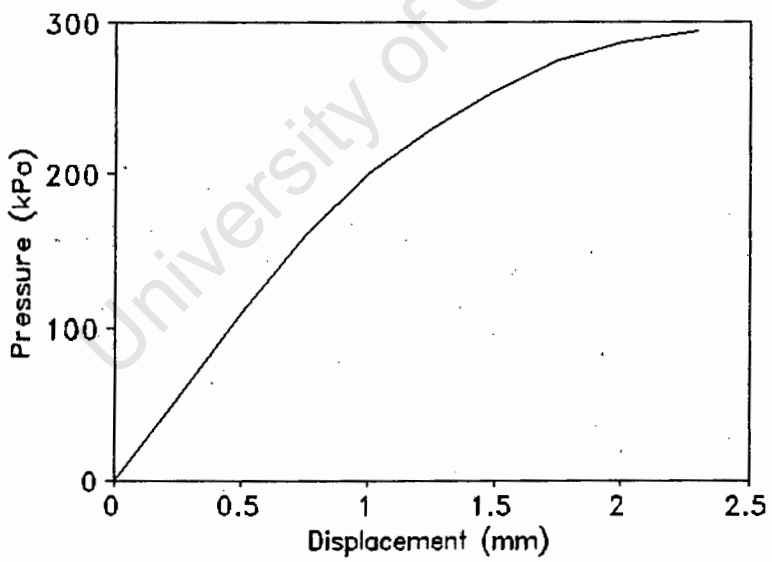


Figure 8.1 : Pressure-displacement relationship of sand foundation in the laboratory

The steel beam used had a yield stress of 300MPa (Grade 300W steel, SABS 0162-1, 1992). The approximate bilinear stress-strain relationship is that shown in figure 4.4.

Figure 8.2 shows the steel beam on the sand foundation including dimensions, locations of the linear variable displacement transducers (LVDT), as well as the position of the jack used

to apply a vertical load to the beam.

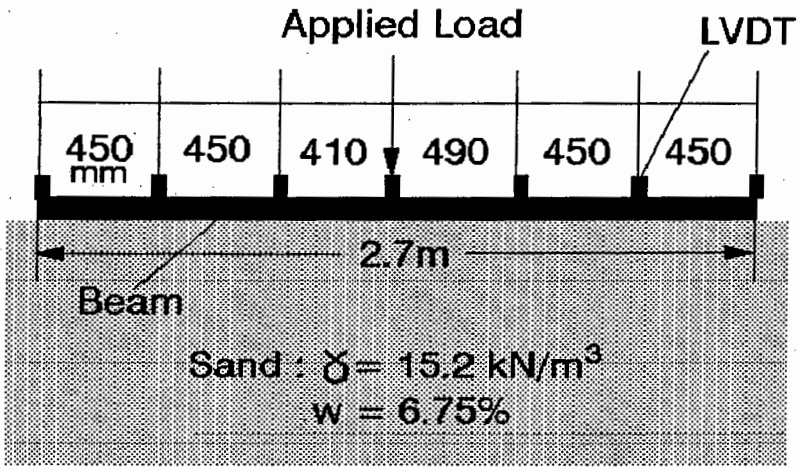


Figure 8.2 : Schematic of beam on sand foundation in laboratory test

The load, applied at about mid-span, was increased continuously by means of an electronically controlled hydraulic jack. The highest load measured before the bearing capacity of the soil was exceeded was 36.8kN. The displacements at ultimate bearing capacity were relatively small, insufficient to cause formation of a plastic hinge within the beam.

8.1.2 Computer Analysis Of Laboratory Test

The above beam-soil system was analyzed at the maximum recorded load using the BSF program. The beam was defined using 6 elements. The comparison of observed deflections compared with those predicted by the analysis is given in figure 8.3. Although the observed displacements are small and may have been affected by the beam 'bedding in' to the sand foundation, a reasonably good agreement between measured and predicted beam deflections was obtained.

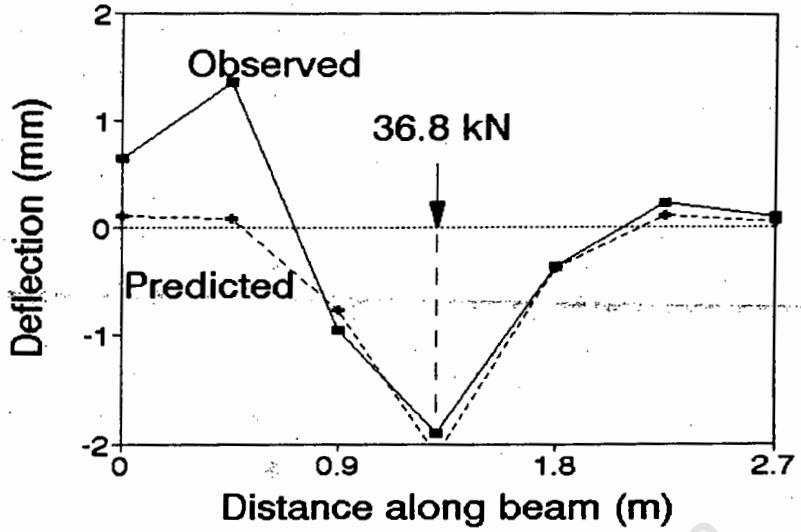


Figure 8.3 : Deflections of test beam at ultimate applied load of 36.8kN

The beam section used was the smallest commercially available. The applied load was incapable of causing a hinge within the beam due to a classical bearing capacity failure in the sand foundation. In spite of the low loads applied and consequent small deflections, accurate predictions of the elastic beam behaviour were obtained.

8.2 FIELD TESTS ON SOLDIER PILE

Given the encouraging verification of the computer program's ability to accurately predict the laboratory test of the horizontal beam, this section applies the program in analysing an actual field test on a soldier pile; the field testing of flexible soldier piles was done to investigate their performance and hinging behaviour under anchor loading. A computer analysis of such a test moves application of the program beyond the theoretical realm of horizontal beams on foundations towards 'real world' problems, thereby offering the possibility of improved design methods for soldier pile support systems.

8.2.1 The Field Tests

A series of tests were carried out on working soldier piles on the perimeter of a 13m deep excavation in the Johannesburg city centre. Two slender piles, with moments of inertia of less than half of that of the adjacent soldiers, were installed in non-critical locations at the periphery of the excavation. Each pile was supported by five prestressed ground anchors with load capacities of 450 or 600kN. In the field tests, the load on the middle anchor was increased progressively while the deflection of the pile was monitored. The results of the test on one of the soldiers is described in detail in this section, followed by a computer analysis of the test for comparative purposes.

The soldier pile concerned had a total length of 13m and consisted of 2 I-beams (160mm x

82mm, I = 8.7 x $10^{-6}m^4$) placed side-by-side and connected at both flanges with tie-plates. The soldier pile was installed in a 600mm diameter auger hole which was subsequently backfilled. The soldier pile penetrated 2m below the bottom of the excavation.

The in-situ material into which the pile was installed was a decomposed residual andesite in the form of a moist, red brown, clayey silt with a firm to stiff consistency. The residual soil has a bulk density of 18 $kN.m^{-3}$, a moisture content 31.7% (average), and the plasticity index was 12.6%. The average effective angle of internal friction was 27° and an effective cohesion of 20 to 40kPa was observed. (In view of the jointed nature of the material, the cohesion is generally ignored in the design of the lateral support.) A plate load test of the in-situ material produced the pressure-displacement response curve shown in figure 8.4. The backfill material response curve, also shown in figure 8.4, was estimated - the initial slope corresponds to a rule-of-thumb strength of 10 MPa.

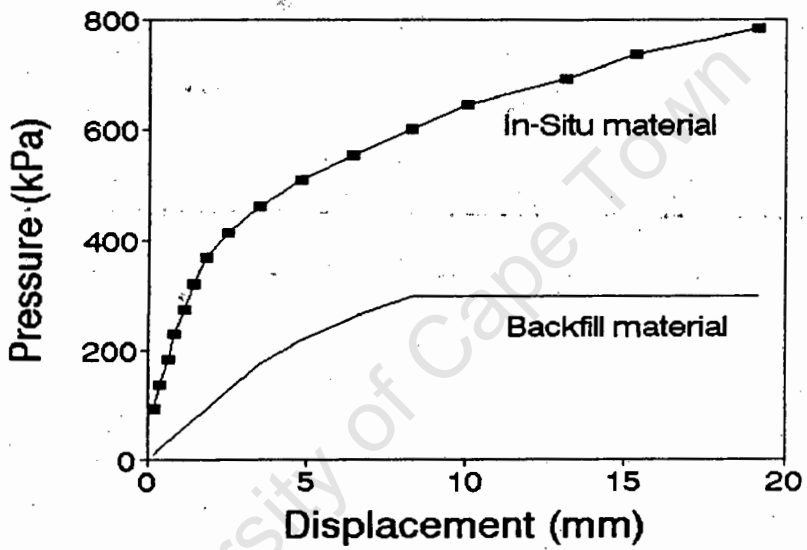


Figure 8.4 : Pressure-displacement relationship of backfill and in-situ material

In figure 8.5, a simplified section through the anchored wall is shown. The soldier pile was tied back at five levels using pressure grouted, prestressed ground anchors with working loads of 450 and 600 kN. All anchors were inclined at 10° to the horizontal. At the commencement of the test all anchors had been stressed to the point where the soldier started bedding into the retained soil. A lift-off test was undertaken to determine the actual load on the middle anchor, which was found to be 417 kN. Movements of the wall were not recorded during excavation or initial stressing of the anchors. However, the movement of a measurement peg, positioned 9m behind the crest at the top of the pile, was recorded to be 6mm towards to the excavation in the horizontal direction.

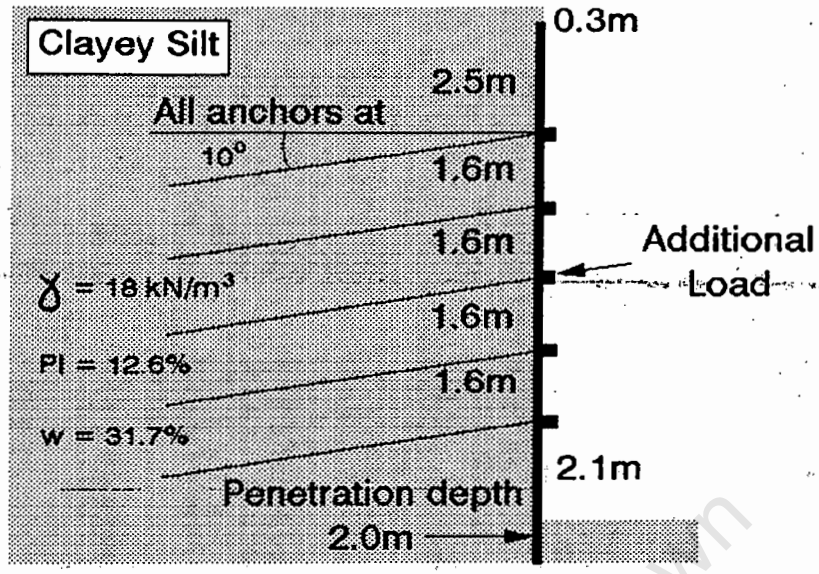


Figure 8.5 : Schematic of the soldier pile support system

In the experiment the load of the central anchor (as indicated in figure 8.5) was increased in increments to 615 kN, 668 kN, and 719 kN (giving additional loads of 198, 271, and 302 kN, respectively) while the deflections were monitored along the total length of the pile. The observed deflections resulting from the application of the additional load are shown in figure 8.6. The deflections were measured relative to a stable datum and are accurate to the nearest millimetre.

From these observations, it appears that the load increment of 198 kN was insufficient to result in the formation of a plastic hinge. However, hinging appears to have occurred for the subsequent load increments of 271 and 302 kN.

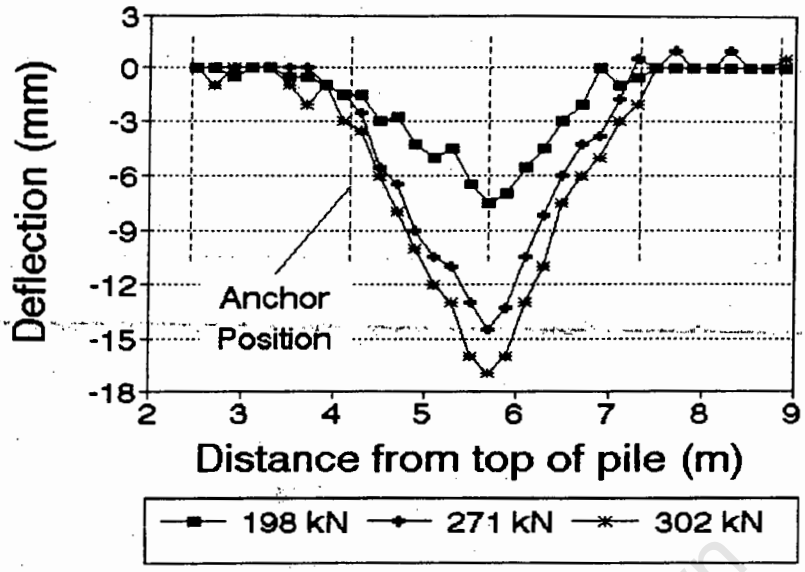


Figure 8.6 : Observed soldier pile deflections from field test

Another test pile in a similar configuration with the same boundary conditions was also loaded to plastic hinging. However, due to eccentric loading of the soldier pile one I-beam hinged independently causing the loading jack to shift out of alignment. Only the first load step prior to hinging could be recorded.

8.2.2 Computer Analysis Of Field Test

Using BSF, it is possible to simulate the field test on the soldier pile, and thus compare the computer output with that observed. Such a comparison will indicate whether or not the methods of this thesis are suitable for soldier pile analysis and design.

Note that no initial conditions for the in-situ soil, backfill material, or soldier pile in terms of loading and deflection were determined. Consequently, the extra loads of 198, 271, and 302 kN are assumed to be acting on a soldier pile system where there is no initial stress in either the soldier or the supporting media. This is an unrealistic assumption and attempts are made to establish initial conditions for the support system based on earth pressure theory in section 8.3.

The input information for the analysis was as follows:

Using the geometry of the I-beams serving as the soldier pile, the thickness of the backfill layer, z_i, (considering the predrilled auger hole diameter of 600mm) was approximately 160mm. The combination depth for the backfill and in-situ materials of figure 8.4 was evaluated as roughly 8mm. This was established by following the step-by-step description in section 7.4.5.5 of the method used to determine combination depth.

With regard to the pile, the base of the pile was modelled as a *fixed-end* condition, the top as a *free* end. The pile was discretised into 150 elements of equal length. The pressure-displacement relationships of the two foundation materials were those shown in

figure 8.4. The stress-strain characteristics of the steel soldier sections were in accordance with the Grade 300W stress-strain curve of figure 4.4.

The predicted pile deflections for the three additional load increments are presented in figure 8.7.

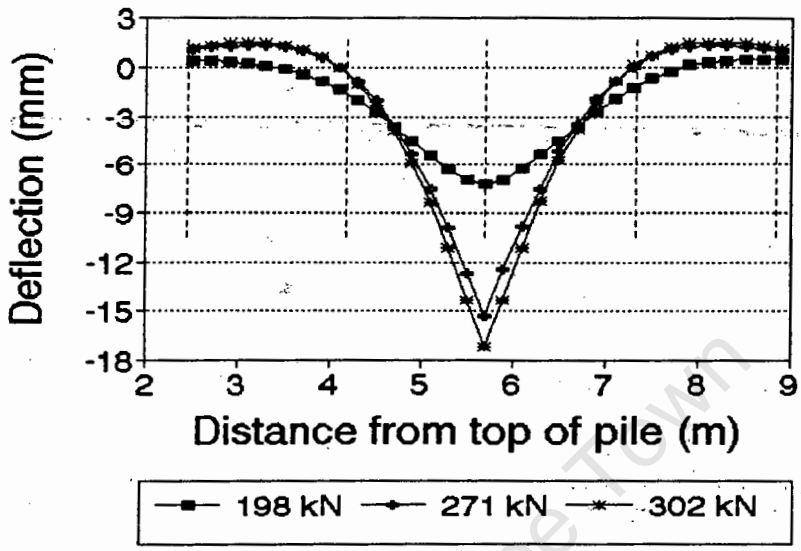


Figure 8.7 : Predicted soldier pile displacements using BSF

The computer solution predicted hinging of the beam (the maximum fibre stress of the steel section reached the yield stress) occurring for displacements greater than or equal to 10 mm. Thus, the 198 kN load increment did not cause a hinge in the pile. Hinging clearly developed for both additional loads of 271 kN and 302 kN as the deflected shapes of the soldier pile at these load levels indicate.

8.2.3 Comparison Of Observed And Predicted Results

The computer analysis of the field test is in good agreement with the observed deflections providing confidence in the use of BSF as a design tool for analysing soldier piles. The theoretical analysis of the soldier piles produced predicted peak deflections very close to those observed in the field for all three load increments. Note that if the softer *backfill* layer was not included in the analysis, and if a single-layer system of only the in-situ material was considered, then the pile deflections under the 271 and 302 kN loads would be quite close to the field observations for these loads. However, the pile movement under the 198 kN load would be too low (about 3mm) because the softer backfill layer would not be there to facilitate the observed deflection of approximately 7.5mm.

Minor differences in deflections are apparent further from the point of load application. The computer analysis predicts small deflections towards the excavation which should, realistically, be prevented by the restraining action of the anchors. Although no measurements were taken of change in loading of the anchors, these changes are likely to have been small as a result of the limited movements observed at these points and the significant free length of the anchors used. The restraining effect of the anchors was therefore ignored in the analyses.

The influence of end conditions had negligible effect on the pile deflection. This is due to the length of the pile, and the very localised nature of the deflection in the vicinity of the applied load.

8.3 SIMULATION OF INITIAL CONDITIONS AFTER FULL EXCAVATION

The analysis of the soldier pile within a lateral support system loaded to plastic hinging (section 8.2.2) was grossly simplified by neglecting any initial conditions in terms of beam deflections and stress (moments) in the solder pile prior to loading. The assumption was made that the system was in a state of equilibrium where lateral earth pressure and anchor loads were in 'perfect' balance : the peak lateral pressures have been redistributed due to local yielding in the soil and the lock-off loads of the anchors dropped off to residual values.

In the following section, an attempt is made to establish the initial conditions at that stage of excavation and wall installation by using the BSF program and to incorporate these into the subsequent computer analysis of beam hinging. Clearly, this is a hypothetical procedure since the lateral earth pressure distribution and magnitude, as well as the actual working loads of the anchors, required as input information, are unknown. The only reliable data for the analysis is the load-displacement relationship of the in-situ soil since the plate load test was performed at the excavation face.

Since field observations of wall deflections do not exist, the outcome of the computer simulation cannot be substantiated and may, therefore, be of only limited validity. However, it will offer insight into the capability of the computer program in modelling typical pile systems for design and analysis purposes.

Again, the soldier pile system, as shown in elevation in figure 8.5, was investigated with the following assumptions:

lateral earth pressure coefficient : $k = (k_a+k_0)/2 = 0.46$
distribution : Terzaghi and Peck (1967) pressure diagram for clay
traffic load : 10 kN/m^2
bulk density of in-situ material : $\gamma = 18$ kN/m^3
pile spacing : 2.6m

A schematic of the pressure distribution behind the wall is shown in figure 8.8. The soldier pile is loaded at five anchor levels with the indicated loads. The loads are assumed to be in close agreement with the actual residual loads (the central anchor load was found to be 417 kN prior to the field test).

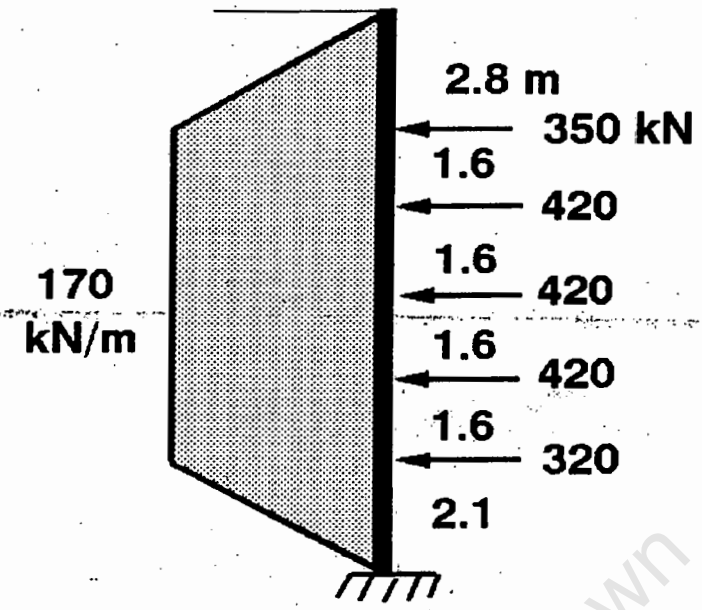


Figure 8.8 : Schematic for simulation of initial conditions after full excavation

The pressure-displacement relationships of the soil materials used in the analysis are as shown in figure 8.4. The *k_error* was kept at 100 kPa, as previously, and the pile was subdivided into 120 elements.

8.3.1 Results Of Computer Simulation

The results of the analysis are presented in the figures 8.9 through to 8.13 below.

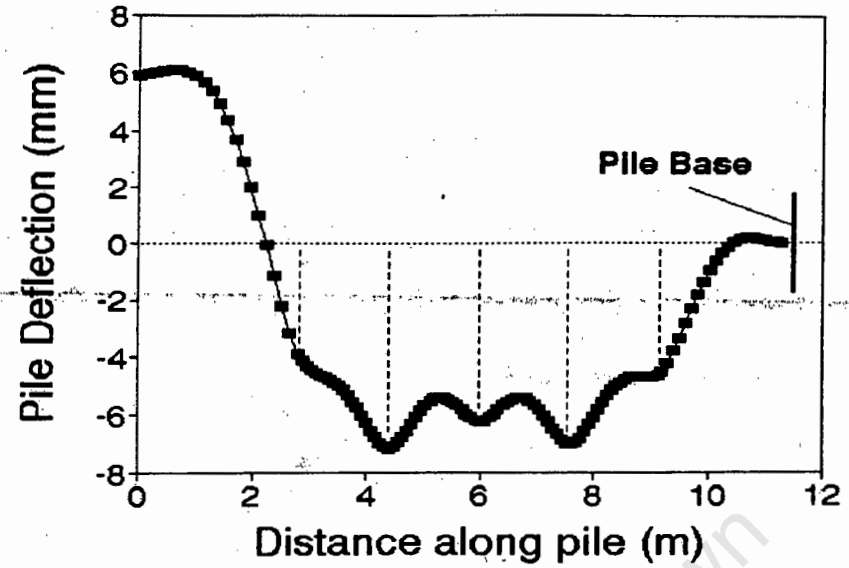


Figure 8.9 : Pile deflection

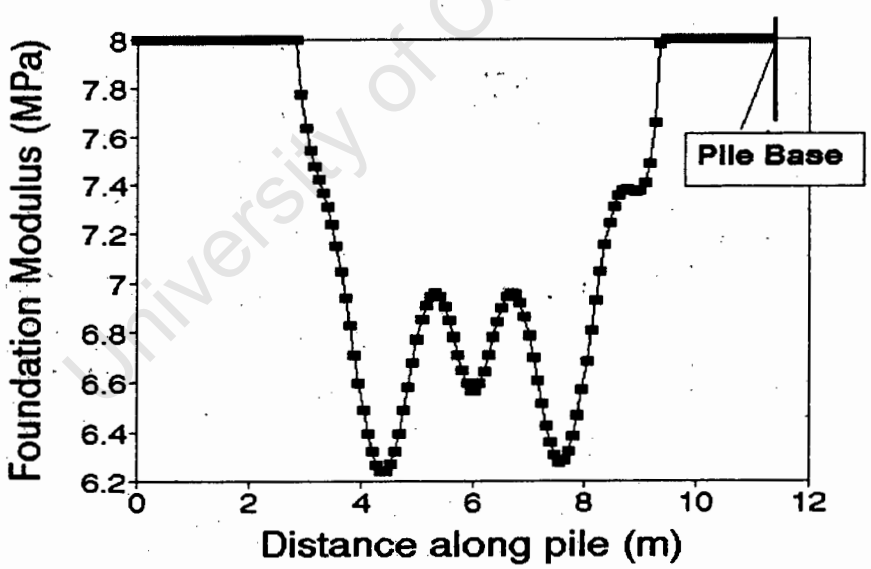


Figure 8.10 : Subgrade modulus profile

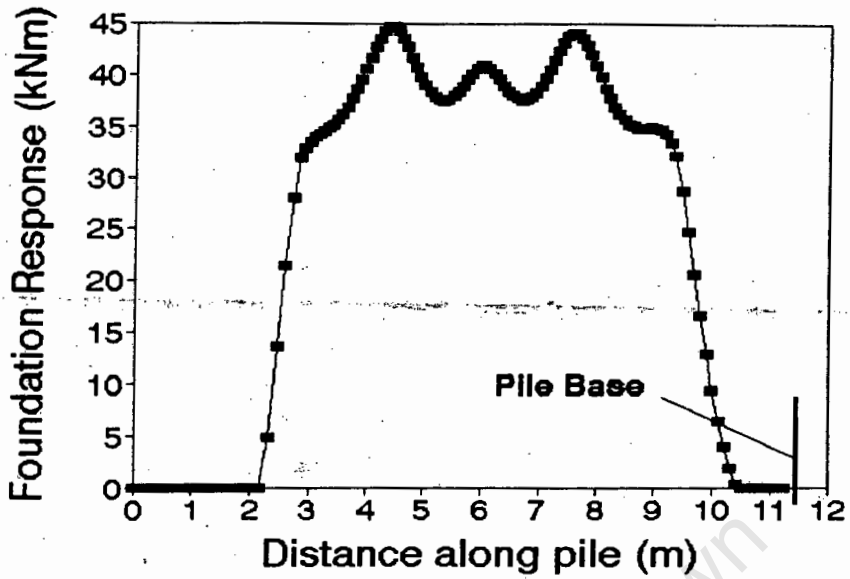


Figure 8.11 : Foundation response profile

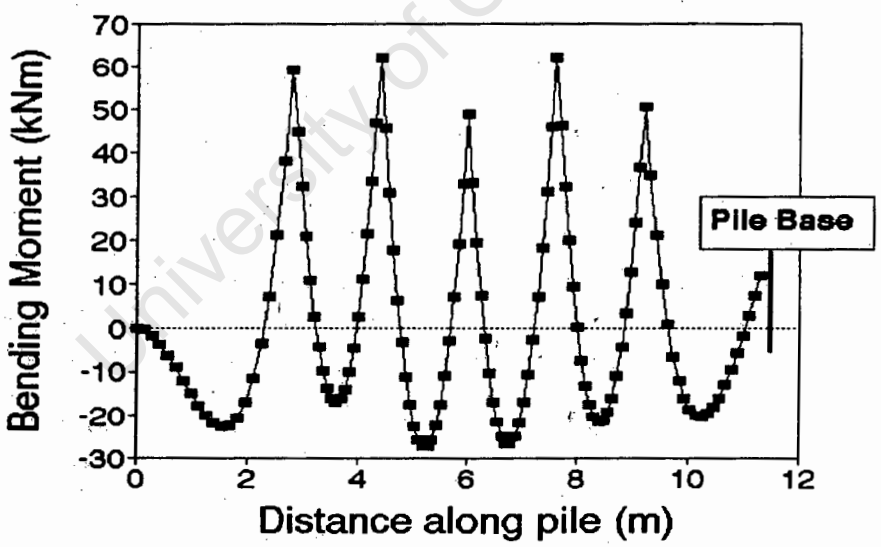


Figure 8.12 : Bending moments in pile

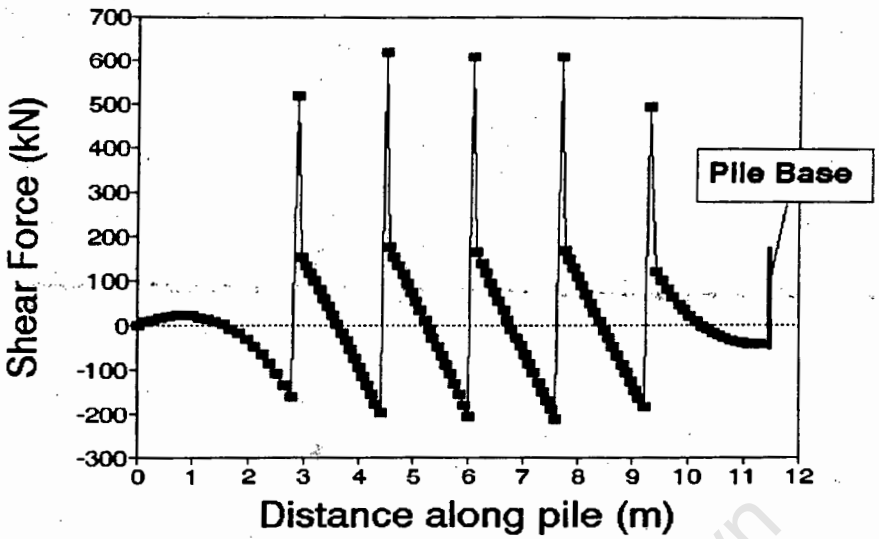


Figure 8.13 : Shear forces in pile

8.3.2 Discussion Of Simulation Results

The designer and practical engineer is primarily interested in wall deflections. Predictions of familiar patterns documented in numerous field observations (for example : Burland, 1979; Ostermayer, 1983) will provide confidence in the computer-based design tool. Typically, the wall bulges out towards the excavation, with the smallest measure at the top of the wall, and from there a linearly-increasing outward deflection of approximately 0.8mm deflection per vertical metre excavation, reaching a peak deflection roughly 2/3 down the excavation face.

Since the program BSF is based on Winkler's spring model, the simulation can only model beam (or wall) behaviour on a soil foundation which is characterised by a subgrade reaction determined in a plate load test conducted at the surface of the soil in the field. Undoubtedly, the deflection predicted by the analysis of the model is a component of the outward deflection of the wall observed by Burland, Ostermayer et al. However, the overriding deflection stems from a wedge or block movement of the entire embankment, and such movement is clearly beyond the scope of the fundamental assumptions of the theory adopted for implementation within the program.

The predicted movement at the top end of the wall (6mm in the horizontal direction) happened, in the case of the analysis in 8.3.1, to agree with the field measurement. Nevertheless, this is likely a coincidence given that the rest of the pile deflection does not bulge out towards the excavation as would be expected.

Moment and shear force distributions, along the wall, are accurately predicted in the simulation. The investigation was then carried further to exhibit beam hinging. The central anchor load was increased from 420 kN to 615 kN, for comparison with the analysis of 8.2.2. At this load level, 6 hinges formed in the pile, producing pile deflections as shown in figure 8.14.

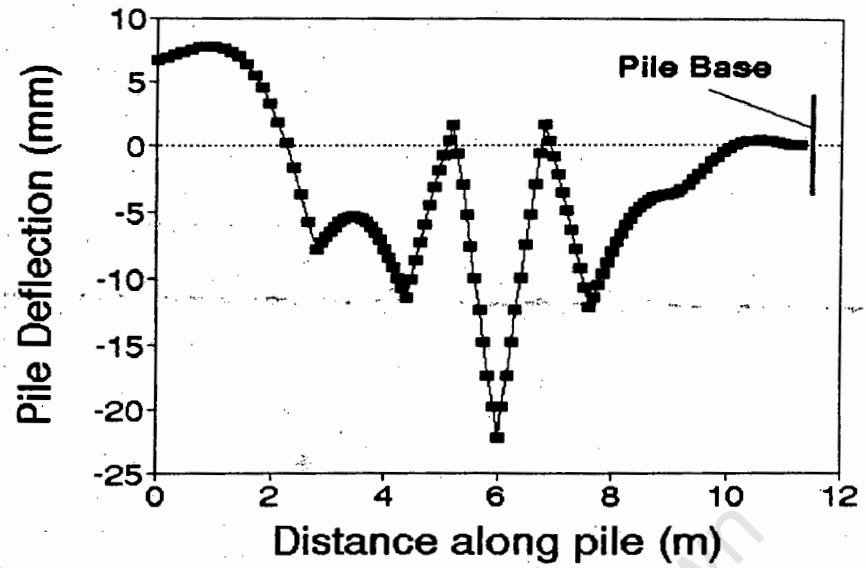


Figure 8.14 : Pile deflection under 615 kN load

This indicates local failure of the pile, and possible collapse of the wall - a situation which was fortunately not experienced in the field. Such a result supports the earlier assumption, made in section 8.2, that redistribution of stresses and system relaxation must have occurred to such an extent that despite the neglection of the initial conditions, the computer predictions still agreed closely with the field observations in that analysis (section 8.2.2).

8.4 SUMMARY OF PROGRAM APPLICATION

In all the above case studies, the BSF program was shown to produce predictions of beam-soil interaction (in both horizontal and vertical models) which were very similar to the observations in the laboratory and field. Close agreement between program output and observed measurements provides confidence for future application of the methods of this thesis. Although there are numerous simplifications in the underlying theory adopted and the program analysis assumptions, the objectives to produce a practical method appear to have been met, underlined by the inclusion of realistic data (plate load tests) in the analysis.

CONCLUSIONS

The ever-increasing demand for computer-aided engineering requires researchers to develop methods, suitable for computer implementation, which provide more powerful analysis and design facilities than are presently available in practice. In geotechnical engineering, structural analysis methods can be applied in several areas, for example : foundations and lateral support systems. Most computer programs for structural analysis make use of matrix-based methods because they are easy to code on conventional computers.

This thesis has considered the development of a computer program, using matrix-based methods, to model and analyze both linear and non-linear beam-soil interaction. Imperative in the approach adopted was the need to focus on practical design considerations. While several researchers have described techniques to model horizontal beams on soil foundations, their methods have included impractical, or insufficient, considerations of beam-soil interaction behaviour. For example, in many cases there was no consideration for inelastic behaviour of the soil; none of the researchers referenced considered inelastic beam behaviour. In addition, no researchers modelled the soil behaviour on parameters measured in the field.

In striving for practical merit for the computer program, assessment of its suitability was concentrated in the area of soldier pile support system modelling and analysis. This was mainly due to the information available from important field tests, and the fact that the soldier pile problem has direct practical significance with regard to making use of this research. Present design methods of soldier pile systems are endeavouring to model beam-soil interaction, and the facility of a computer-based tool for such a purpose will greatly improve design methods in this area of geotechnical engineering.

In order to achieve a practically viable analysis method, an existing matrix-based technique for structural analysis was extended, using an iterative procedure, to allow consideration of non-linear behaviour for both the beam and the soil in beam-on-soil foundation problems. Fundamental to the matrices used was the widely known Winkler foundation model. While such a model offers a simplistic approach to defining beam-soil interaction, it lends itself more easily to computer program development since a stiffness matrix for such interaction modelling can readily be derived. Moreover, research using the model has not produced spurious results.

To base the method on realistic characteristics, the soil behaviour was modelled on a plate load test (done in the field), and beam behaviour on a stress-strain curve for the beam material (easily available for design purposes). Provision was also made for possible plastic hinging in the beam, where structural hinges could be used to reflect local yielding of the beam material.

The computer program of this thesis, BSF, was coded using the powerful C++ language, and its interface was based on graphical methods for improved interaction with the user. The program runs on the popular IBM/IBM-compatible desktop platform, and offers a number of flexible options to enable analysis of beams on both horizontal and vertical (soldier pile) foundations. Various features have also been incorporated to enable future research on beam-

soil interaction to make use of the program.

In assessing the accuracy of the program in predicting beam-soil interaction, both a laboratory and field test were analysed. The laboratory test had measured the deflection of a horizontal beam on soil foundation. Computer prediction of beam-soil interaction made by BSF was in close agreement with that observed in the laboratory. The field test of a soldier pile offered the opportunity of measuring the correctness of the methods described for modelling a vertical beam on a vertical foundation. Here again, a simplified model for the lateral support system led to computer predictions being in very close agreement with field observations.

Future research and modifications could consider layered soil systems behind soldier piles. Although there is, at present, no feature within BSF to enable this, the algorithms used could easily be altered to allow discrete soil pressure-displacement curves for the different subsections (representing the discontinuities between soil layers) of the pile. Examinations of element subgrade modulus values would then be done with reference to the appropriate soil curve for the given subsection under inspection.

In the case of soldier pile support systems, more research is required to examine the nature of block movement of the entire wall. In addition, research needs to be done into the redistribution of stresses within the support system, including the effects of creep. The results of such research would be useful for continued investigation into computer-aided modelling of soldier pile support systems.

Nevertheless, the computer-based facility to model and investigate the interaction of beam-soil problems, as provided by this work, offers the practising engineer a far more detailed approach to analysing beam-soil interaction than is presently available. In doing so, this research stands to make a noteworthy contribution to improved engineering design through the use of computer technology.

BIBLIOGRAPHY

Brohn, D. (1984). *Understanding Structural Analysis*. Granada Dobbs Ferry, New York. pp. 271,273.

Burland, J.B., Simpson, B. and St.John, H.D. (1979). *Movements around excavations in London clay*. Proc - 7th Euro.Conf.Soil Mech.Found.Eng., Brighton, Vol. 1.

Eisenberger, M. and Yankelevsky, D.Z. (1985). Exact stiffness matrix for beams on elastic foundation. *Computers and Structures*. Vol. 21, No. 6, pp. 1355-1359.

Filonenko-Borodich, M.M. (1940). Some approximate theories of the Elastic Foundation. *Uch. Zap. Mos. Gos. Uni. Mekh.* No. 46, pp. 3-18.

Gere, J.M. and Weaver, W.Jr.D. (1965). *Analysis of framed structures*. Van Nostrand Comp., Inc., Princeton, N.J.

Hayashi, K. (1921). *Theory of beams on elastic foundation*. Springer Verlag, Berlin [in German].

Hetenyi, M. (1946). *Beams on Elastic Foundation*. University of Michigan Press, Ann Arbor, Michigan.

Howie, C.T. (1991). *Beam analysis on inelastic foundation*. BSc Thesis, University of Cape Town.

Howie, C.T., Scheele, F., and Day, P.W. (1994). *Soldier pile analysis using nonlinear beam-foundation theory*. XIII International Conference on Soil Mechanics and Foundation Engineering. New Delhi, India. Paper accepted for publication.

Kany, M. (1974). *Berechnungen von Flachengrundungen*. Band 1 und 2, 2. Aufl., W. Ernst u. Sohn, Berlin.

Kerr, A.D. (1964). Elastic and viscoelastic foundation models. ASME *J. of Applied Mechanics*. Vol. 31, pp. 491-498.

Ohde, J. (1942). *Berechnungen der Sohldruckverteilung unter Grundungskorpern*. Der Bauingenieur, 23, 102-105.

Ostermayer, H. and Mager, W.D. (1983). *Messungen au verankerten konstruktionen*. Sympos., Messtechnik im Erd-und Grundbau, Munich. DGEG, Essen.

Pasternak, P.L. (1954). On a new method of analysis of an elastic foundation by means of two foundation constants. *Gos. Izd. Lit. po Stroit i Arkh*, Moscow.

SABS 0162-1, 1992. Draft Code. Council of the South African Bureau of Standards.

Terzaghi, K and Peck, R.B. (1967). *Soil mechanics in engineering practice*. John Wiley Sons, N.Y.

Vallabhan, C.V.G. and Das, Y.C. (1991). A refined model for beams on elastic foundations. *Int. J. Solids Structures*, Vol. 27, No. 5, pp. 629-637.

Yankelevsky, D.Z, Eisenberger, M. and Adin, M.A. (1989). Analysis of beams on nonlinear Winkler foundation. *Computers and Structures*. Vol. 31, No. 2, pp. 287-292.

Zimmermann , H. (1930). *Die Berechnung des Eisenbahn-Oberbaues*. 2. Aufl., W. Ernst u. Sohn, Berlin.

INFLUENCE CHART FOR LIMITED LINE LOAD