IMPROVED CHARTS FOR THE METHOD OF FRAGMENTS

by

Jennifer E. Huggins

ProQuest LLC.
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106 – 1346

A thesis submitted to the Faculty and the Board of Trustees of the Colorado School of Mines in partial fulfillment of the requirements of the degree Master of Science (Engineering).

Golden, Colorado

Date 5/19/10

Signed: _____
Jennifer E. Huggins

Signed: _____
Dr. D.V. Griffiths
Thesis Advisor

Golden, Colorado

Date 5/19/10

Signed: _____
Dr. Terry Parker
Professor and Director
Engineering Division

ABSTRACT

The Method of Fragments has been used by engineers to determine quantities of seepage beneath and exit gradients due to confined flow beneath civil engineering structures such as dams for several decades. Fragments have been classified into various types based upon the arrangement of impermeable boundary conditions. Previously, charts have been developed for various fragments types for the quick computation of fragment form factors and exit gradients to assist engineers in design.

With the increased computer power now available, this project uses a finite element approach in order to corroborate previously established charts for form factor and exit gradients, and to develop new exit gradient charts. An existing, non-commercial finite element code (Smith and Griffiths 2004) was modified and used to solve the Laplace equation for steady-state seepage flows.

Of the two types of fragments explored, the "Type A" and the "Type C" fragments, only the "Type C" fragment had existing published exit gradient charts. These charts were expanded and additional new exit gradient charts, not found in any other literature, were developed for special cases of the "Type A" fragment.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

The author wishes to thank her advisor, D.V. Griffiths, as well as her thesis committee members Professors Panos Kiousis and Mike Mooney, for all of their thoughtful insight and consultation.

Tuition costs for this project were paid by the U.S. Bureau of Reclamation as authorized by John LaBoon.

For my husband.

INTRODUCTION

The Method of Fragments is a simple, approximate way of solving problems of confined seepage flow under civil engineering structures such as dams and weirs. The Method of Fragments divides are area of porous foundation beneath a dam with multiple cutoffs into rectangular fragments. Each fragment has a form factor which can be used to calculate the rate of flow passing through and these form factors are read from charts based upon the fragment's dimensions. One type of fragment also has a chart for determining the exit gradient.

Prior to the advent of modern computing techniques, fragment charts were developed using the mathematical technique of conformal mapping and several are documented in Polubarinova-Kochina (1962). This project, however, uses the finite element method to solve the problem of steady-state flow through a two-dimensional mesh of four-node quadrilateral elements in order to solve for fragment form factors and exit gradients.

This project was undertaken in order to examine exit gradient data for the "Type C" fragment. The existing charts for "Type C" fragment included in Appendix A, exhibit peak values for the non-dimensional form of the exit gradient. The author was curious as to whether these peaks might hint at some best or worst case design scenarios for vertical cutoff wall length and this project was initiated in order to investigate.

After the investigation of the "Type C" exit fragment was concluded, the scope of the project broadened to include the "Type A" fragment. To the author's best knowledge, there are no charts for values of exit gradient for the "Type A" fragment currently documented. The finite element approach was continued and the work with the "Type C" fragment was expanded upon and modified in order to create exit gradient charts for the "Type A" fragment.

Various facets of the analyses conducted for each of the two fragment types of interest include mesh sensitivity studies as well as studies of the effect of anisotropic soil material properties upon form factor and exit gradient results. The results of these studies are documented herein and finished; expanded charts for the exit gradient of each of the two fragments are also included.

CHAPTER 1

OVERVIEW OF THE METHOD OF FRAGMENTS

The Method of Fragments is a way of solving problems of confined seepage flow under civil engineering structures such as dams and weirs. It originated in Russia with Pavlovsky in 1922 (Polubarinova-Kochina 1962) and was introduced in the United States by Harr in the 1960s with the publication by McGraw-Hill of his text book entitled Ground Water and Seepage. The Method of Fragments was further refined by Griffiths in the 1980s with his paper entitled Rationalized Charts of the Method of Fragments published in Geotechnique.

Previous to the introduction of the method of fragments, problems of confined seepage were solved using methods such as conformal mapping or the sketching of flow nets. Both of these approaches can be rather difficult and time consuming whereas the method of fragments offers an array of readily available solutions.

In his book published in 1922, N.N. Pavlovsky considered the problem of seepage beneath a dam as being represented as the flow of water through the polygonal region of the permeable soil upon which the structure was founded (Polubarinova-Kochina 1962). The region is described by the straight line contours of the foundation footprint of the structure and the boundary of the impervious bedrock below the porous foundation material. The Christoffel-Schwartz formula can then be used for the conformal mapping of the polygon into the half plane.

As an alternative to conformal mapping, Davidenkoff and Franke used an electrical analogue method as a means of studying the problem to seepage through cofferdams and other problems defined by two fragments (Azizi 1999). Unfortunately, the published works of Davidenkoff and Franke could not be obtained because they were only found in a library in Europe that has a policy against lending materials outside of Europe and Africa.

The method of fragments is not the most accurate way to solve confined flow problems; however, it is a very good approximation and simple to employ. Harr (1962) developed equations for form factors for six fragment shapes. Griffiths (1984) further simplified matters with the derivation of rationalized charts for form factors for three fragment types. Each

fragment is a rectangular shape the types of which are distinguished by their impermeable boundary conditions.

The method of fragments is similar to sketching a flow net in many ways. Flow nets are sketched to partition an area of interest into flow channels and further divided with equipotential lines. The shape factor for the problem is then determined by dividing the number of flow channels by the number of equipotential drops. In the method of fragments, an area of interest is separated into rectangular areas corresponding to the boundary conditions of applicable fragments. Similar to flow nets, each fragment has a form factor. The form factor is based upon the dimensions and boundary conditions of the fragments. The key assumption which makes the method of fragments an approximation is that equipotential lines are coincident with the vertical boundaries.

1.1 Fragment Types

The three fragment types (Griffiths, 1984) are known by the letters A, B and C and shown in Figure 1.1. The heavy lines denote impermeable boundary conditions. The two fragments of most interest in the studies conducted as part of this project are the "Type A" and the "Type C" fragments.

The "Type A" fragment is defined by an impermeable boundary along the bottom and an impermeable boundary at one corner. The vertical cutoff wall is of length s and the horizontal cutoff wall is of length b. The "Type A" fragment is not defined by its length.

The "Type B" fragment is defined by impermeable boundaries all along the top and the bottom and a vertical cutoff at each end. The vertical cutoffs are of dimensions s' and s'' and the length of the fragment is L.

The "Type C" fragment is defined by impermeable boundaries extending along the entire vertical dimension of one side and the bottom with one vertical cutoff on the opposite side. Like in the "Type A" fragment, the "Type C" fragment's vertical cutoff is of length s. Its length L can be infinite as well.

1.2 Application of the Method of Fragments

The method of fragments is a simple, if approximate, method for solving problems of confined flow under civil engineering structures. The area of porous foundation material beneath a structure with multiple vertical cutoffs can be partitioned into fragments and the form factors

determined for each individual fragment. Figure 1.2 shows a dam with an upstream clay blanket on a porous foundation and an impermeable layer below.

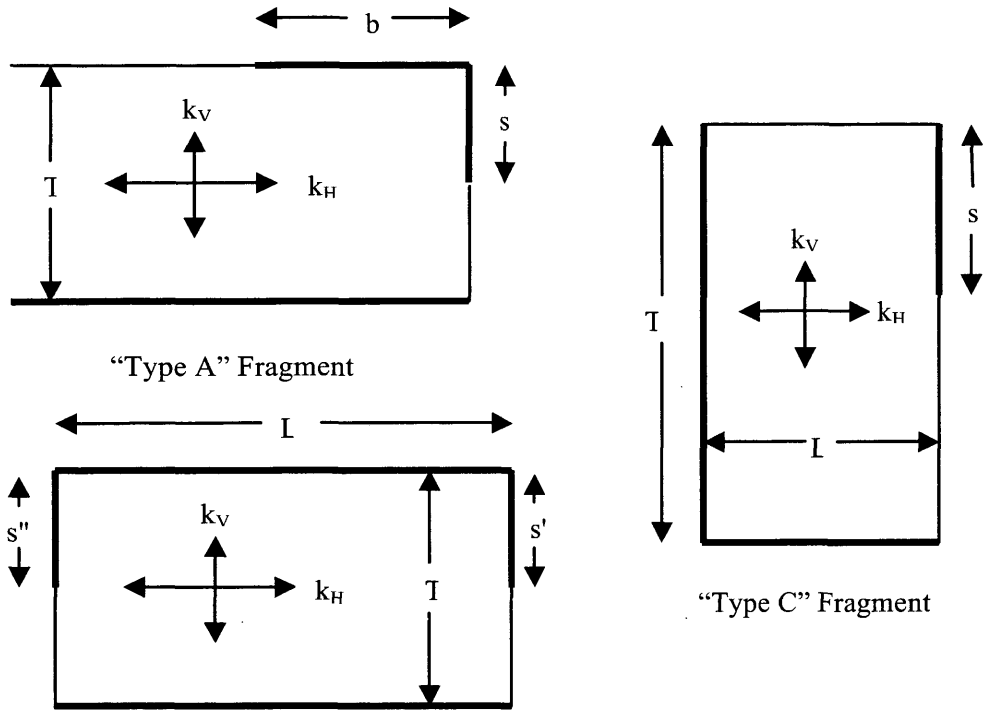


Figure 1.1 The three fragment types.

The flow rate can be determined by equation (1.1) where Q is the flow rate, $\bar{k}$ is the effective permeability, H is the total head loss and $\Sigma\Phi$ is the sum of the form factors for all fragments (Holtz and Kovacs 1981). The form factor, Φ, is easily read from the charts when the dimensions of the fragment are known. It is the reciprocal of the shape factor that one would derive from a flow net; it could be represented by the number of equipotential drops divided by the number of flow channels.

$$Q = \frac{\bar{k}H}{\sum \Phi} \tag{1.1}$$

The number of flow channels in each fragment is constant in a confined flow problem but the number of equipotential drops can vary from fragment to fragment. As the number of flow channels is in the denominator for the form factor, Φ, the principle of superposition can be

applied to obtain a form factor for the entire problem. For example the form factor for the problem shown in Figure 1.2 would be determined by summing the form factors for each of the three fragments. Knowing the total head loss, form factors and soil permeability equation (1.1) can be applied to determine the rate of seepage.

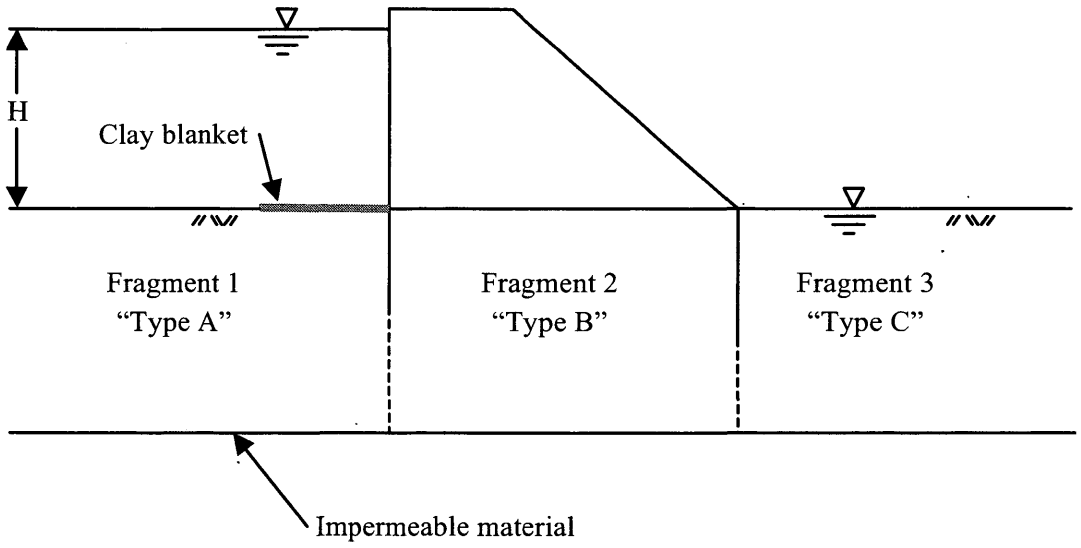


Figure 1.2 Gravity dam on a porous foundation, shown divided into fragments.

1.3 Anisotropy

The modified permeability is used to scale all lengths in the horizontal direction in the sketching of a flow net when the soil permeability differs in the horizontal and vertical directions. The modified permeability is determined using equation (1.2).

$$\bar{k} = \sqrt{k_H k_V} \tag{1.2}$$

The charts used for the method of fragments employ an anisotropy factor called R. Horizontal dimensions are multiplied by the factor as they are plugged into the chart to determine form factors or exit gradient. R can be calculated using the following equation (1.3).

$$R = \sqrt{\frac{k_V}{k_H}} \tag{1.3}$$

1.4 Exit Gradients

Head loss in an individual fragment is proportional to the form factor of the fragment. For example, the head loss in the third, or exit, fragment in the problem shown in Figure 1.2 can be calculated using equation (1.4).

$$h_3 = \frac{\Phi_3}{\sum_1^3 \Phi} H \tag{1.4}$$

The exit hydraulic gradient can be estimated from the head loss in the exit fragment. For the problem shown in Figure 1.2, the head loss across the exit fragment can be determined as shown above. Knowing the length of the downstream cutoff wall, s, the depth to the impermeable foundation layer, T, and considering the length of Fragment 3 as being infinite, one can use the charts found in Appendix A to determine the value of $i_e s/h$. In this case, h is equal to h_3 and evaluation of the ratio will yield a value for i_e, the exit gradient.

1.5 Example Problem

The example problem shown in Figure 1.3 is a concrete gravity dam situated on a porous foundation 12 meters deep from the ground surface to bedrock. The total head for the problem is 12 meters and soil is isotropic with $k_H = k_V = 10^{-7}$ m/s. This example problem will be solved using the Method of Fragments and the charts found in Appendix A in order to determine the rate of seepage through the foundation as well as the exit gradient at the downstream toe of the dam.

The process of solving a problem by the Method of Fragments can be broken down into several steps. The first is to divide the problem into fragments at the locations of vertical cutoffs. After the problem has been separated into fragments and the dimensions of each have been determined, the form factor for each individual fragment can then be read from the charts. Having determined the form factor, Φ, for each fragment, their sum can be taken in order to compute the seepage rate using equation (1.1).

If the exit fragment is a "Type C" fragment, the value of the $i_e s/h$ can be read from the exit gradient chart in Appendix A. In the expression $i_e s/h$, i_e is the exit gradient, s is the length of the vertical cutoff as shown in Figure 1.1, and h is the head loss through the exit fragment which can be computed using equation (1.4). Since s and h are known or computed values, the

expression can be resolved in order to determine the value of the exit gradient. The exit gradient is of interest for determining the factors of safety against instability or piping.

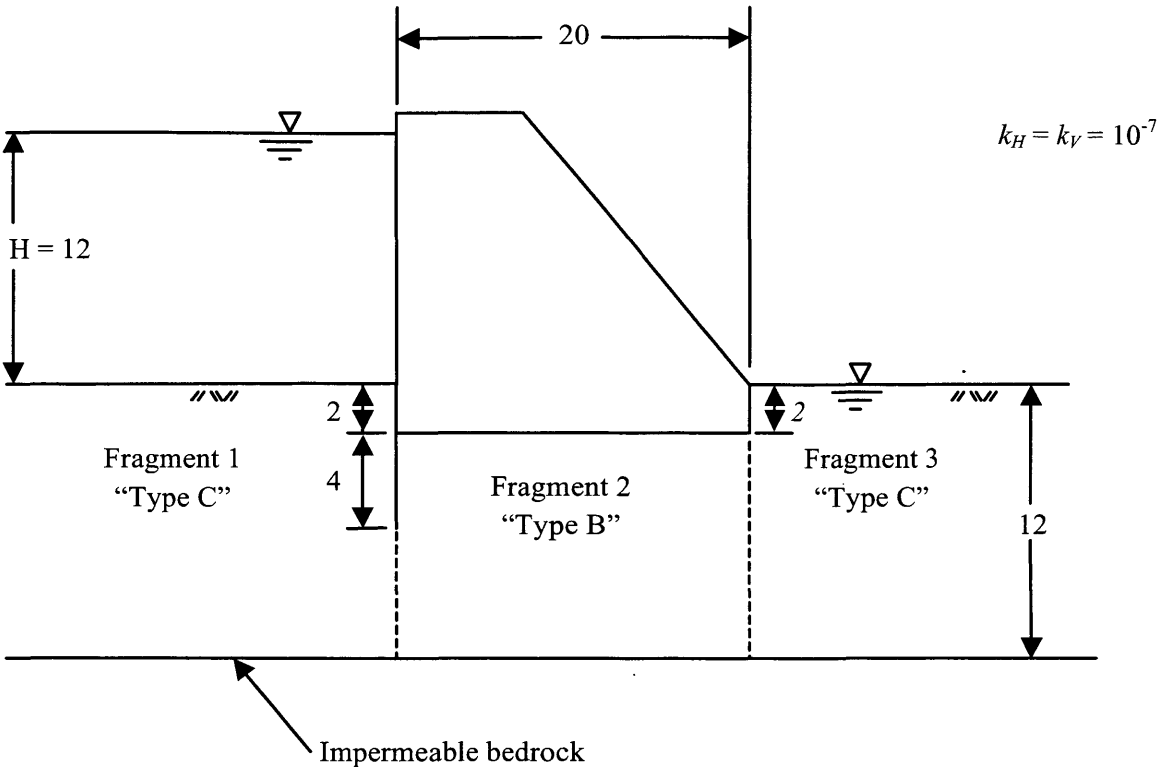


Figure 1.3 Example problem showing a concrete gravity dam on a porous isotropic soil foundation, $k_H = k_V = 10^{-7}$ m/s, divided into three fragments.

As shown, this example problem can be divided into three fragments. Fragment 1 is a "Type C" fragment with a vertical cutoff, s, of 6 meters and a fragment depth, T, of 12 meters. Its length is infinite. Because k_H is equal of k_V, $R = 1$, this is true for all three of the fragments. Given that $s/T = 0.5$ and $L = \infty$, the form factor read from the chart for fragment 1 is $\Phi_1 = 1.0$.

Fragment 2 is a "Type B" fragment with one vertical cutoff and the dimensions, in the terms used by the charts, are $s' = 4$ meters, $s'' = 0$ meters and $T = 10$ meters. Looking at the chart for form factor for the "Type B" fragment found in Appendix A, one can calculate the dimensionless parameters C_1 and C_2 using equations shown on the chart and repeated below as equations (1.5) and (1.6), respectively. Selecting the curve for $C_2 = 1.6$ and given that $C_1 = 0.6$, the value for Φ_2 is read as 2.11.

$$C_1 = \left(1 - \frac{s'}{T}\right)\left(1 - \frac{s''}{T}\right) = \left(1 - \frac{4}{10}\right) = 0.6 \qquad (1.5)$$

$$C_2 = \frac{LR - (s' + s'')}{T} = \frac{20(1) - 4}{10} = 1.6 \tag{1.6}$$

Fragment 3, the exit fragment, is of "Type C" with infinite length, a vertical cutoff of $s = 2$ meters and a depth of $T = 12$ meters. Since $s/T = 0.167$ and $L = \infty$, Φ_3 is read from the chart as 0.507. Likewise, $i_e s/h$ is found to have a value of 0.632.

Now that each of the form factors has been found, equations (1.2) can be substituted into equation (1.1) to get equation (1.7) in order to determine the rate of seepage as shown below.

$$Q = \frac{\bar{k}H}{\sum_1^3 \Phi} = \frac{\sqrt{k_v k_h}H}{\Phi_1 + \Phi_2 + \Phi_3} = \frac{10^{-7}(12)}{1.0 + 2.11 + 0.507} = \frac{1.2 \times 10^{-6}}{3.617} = 3.32 \times 10^{-7} m^3/s/m \tag{1.7}$$

Because the exit fragment is a "Type C" fragment and $i_e s/h$ was read from the chart as 0.632, the exit gradient can be computed and a factor of safety against instability can be determined. First, the head loss through the exit fragment must be calculated using equation (1.4).

$$h_3 = \frac{\Phi_3}{\sum_1^3 \Phi} H = \frac{0.507}{3.617}(12) = 1.68 \tag{1.4}$$

Now the exit gradient can be expressed as shown in equation (1.8).

$$i_e = \frac{0.632 h_3}{s} = \frac{(0.632)(1.68)}{2} = 0.53 \tag{1.8}$$

The factor of safety against instability is defined as a ratio of the computed exit gradient to the critical exit gradient as shown in equation (1.9). As a rule of thumb, the critical exit gradient, i_{cr}, can be taken as 1.0.

$$FS = \frac{i_{cr}}{i_e} = \frac{1}{0.53} = 1.88 \tag{1.9}$$

Suppose evaluation of this facility deemed the seepage quantity to be too large. A modification will be made to the structure in order to decrease the quantity of seepage. This modification will be discussed as a continuation of this example problem later on in Chapter 5.

CHAPTER 2

OVERVIEW OF THE FINITE ELEMENT METHOD

Having modern computing tools very near to hand, a good way to solve problems of steady-state seepage is by using the finite element method. This is a luxury not available so widely nor so inexpensively to our predecessors in the 1960s. In this project an existing, non-commercial finite element code (Smith and Griffiths 2004) was modified and used repeatedly to solve the steady-state seepage problem of the "Type A" and "Type C" fragments and the results were used to develop and improve charts for form factor and exit gradient estimation. Specific modifications made to the code will be discussed in the following chapter.

2.1 Steady-State Problems

Two-dimensional problems of steady-state seepage are governed by Laplace's equation (2.1).

$$k_x \frac{\partial^2 h}{\partial x} + k_y \frac{\partial^2 h}{\partial y} = 0 \tag{2.1}$$

Where k_x and k_y are the soil permeabilities in x- and y-directions, respectively, and h is the potential or total head.

2.2 Element Type and Formulation

Problems of steady-state seepage in two dimensions are best solved using four-node quadrilateral elements. Each node has one degree of freedom, the potential. The solution of these problems involves finding the total head at any point in the mesh. Constructing the element conductivity matrix begins with a trial solution given in the form of equation (2.2), where N_i are the shape functions for 4-node elements and h_i are the potentials at each of the four nodes. The trial solution allows for an approximate solution of the problem. Initially unknown parameters such as shape function will be optimized by means of the weighted residual so that $\tilde{h}$ is as accurate as possible.

$$\tilde{h} = \sum_{i=1}^{4} N_i h_i \tag{2.2}$$

The trial solution can be rewritten as equation (2.3) and then substituted into the governing equation, in this instance the Laplace equation (2.1), to obtain equation (2.4).

$$\tilde{h} = \begin{bmatrix} N_1 & N_2 & N_3 & N_4 \end{bmatrix} \begin{Bmatrix} h_1 \\ h_2 \\ h_3 \\ h_4 \end{Bmatrix} \tag{2.3}$$

$$k_x \frac{\partial^2}{\partial x^2} \begin{bmatrix} N_1 & N_2 & N_3 & N_4 \end{bmatrix} \begin{Bmatrix} h_1 \\ h_2 \\ h_3 \\ h_4 \end{Bmatrix} + k_y \frac{\partial^2}{\partial y^2} \begin{bmatrix} N_1 & N_2 & N_3 & N_4 \end{bmatrix} \begin{Bmatrix} h_1 \\ h_2 \\ h_3 \\ h_4 \end{Bmatrix} = R \tag{2.4}$$

In equation (2.4) the term R represents the error that is introduced by the approximate solution, also called the residual. Now the residual is weighted using Galerkin's Method (Smith and Griffiths 2004) which can be symbolically described in one dimension by equation (2.5). The objective in doing so is to find the best trail function, ψ_i, so that R is minimized. In this case, the trial functions are the shape functions for the 4-node quadrilateral element.

$$\int_{xA}^{xB} R\psi_i dx = 0 \tag{2.5}$$

Having applied Galerkin's Method , equation (2.4) can now be written as shown in equation (2.6)

$$\begin{aligned} k_x \iint_{area} \begin{Bmatrix} N_1 \\ N_2 \\ N_3 \\ N_4 \end{Bmatrix} \frac{\partial^2}{\partial x^2} \begin{bmatrix} N_1 & N_2 & N_3 & N_4 \end{bmatrix} dxdy \begin{Bmatrix} h_1 \\ h_2 \\ h_3 \\ h_4 \end{Bmatrix} + \\ k_y \iint_{area} \begin{Bmatrix} N_1 \\ N_2 \\ N_3 \\ N_4 \end{Bmatrix} \frac{\partial^2}{\partial x^2} \begin{bmatrix} N_1 & N_2 & N_3 & N_4 \end{bmatrix} dxdy \begin{Bmatrix} h_1 \\ h_2 \\ h_3 \\ h_4 \end{Bmatrix} = \begin{Bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{Bmatrix} \end{aligned} \tag{2.6}$$

It is important to note that a typical term in equation (2.6) can be approximated as shown in equation (2.7)

$$\int\int_{area} N_i \frac{\partial^2 N_j}{\partial y^2} dxdy \approx -\int\int_{area} \frac{\partial N_i}{\partial y} \frac{\partial N_j}{\partial y} dxdy \tag{2.7}$$

After substituting the approximate term in equation (2.7) into equation (2.6), it can then be integrated by parts and after the terms associated with the boundaries of no flow have been discarded, the governing differential equation can be expressed as the matrix equation seen in (2.8). This is the element conductivity relationship where $[k_c]$ is the element conductivity matrix, $\{h\}$ are nodal potential values and $\{q\}$ are nodal flux values.

$$[k_c]\{h\} = \{q\} \tag{2.8}$$

The more generic form of the element conductivity matrix is expressed by equation (2.9). In equation (2.9), where $[T]$ is the matrix associated with the partial derivatives of the shape functions as shown in equation (2.10) and $[K]$ are the permeabilities a equation (2.11).

$$[k_c] = \int\int_{area} [T]^T [K][T] dxdy \tag{2.9}$$

$$[T] = \begin{bmatrix} \frac{\partial N_1}{\partial x} & \frac{\partial N_2}{\partial x} & \frac{\partial N_3}{\partial x} & \frac{\partial N_4}{\partial x} \\ \frac{\partial N_1}{\partial y} & \frac{\partial N_2}{\partial y} & \frac{\partial N_3}{\partial y} & \frac{\partial N_4}{\partial y} \end{bmatrix} \tag{2.10}$$

$$[K] = \begin{bmatrix} k_x & 0 \\ 0 & k_y \end{bmatrix} \tag{2.11}$$

For the case of a 4-node quadrilateral element shown in Figure 2.1, the shape functions can be written in terms of Cartesian coordinates as shown in equations (2.12) to (2.15) (Smith and Griffiths 2004).

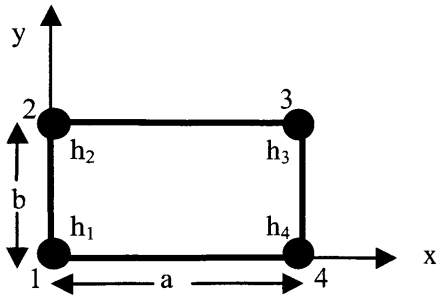


Figure 2.1 A rectangular 4-node quadrilateral element with clockwise node numbering.

$$N_1 = \left(1-\frac{x}{a}\right)\left(1-\frac{y}{b}\right) \tag{2.12}$$

$$N_2 = \left(1-\frac{x}{a}\right)\left(\frac{y}{b}\right) \tag{2.13}$$

$$N_3 = \left(\frac{x}{a}\right)\left(\frac{y}{b}\right) \tag{2.14}$$

$$N_4 = \left(\frac{x}{a}\right)\left(1-\frac{y}{b}\right) \tag{2.15}$$

Integrating equation (2.9) analytically leads to the expression for element conductivity seen in equation (2.16).

$$[k_c] = \frac{k_x}{6}\frac{b}{a}\begin{bmatrix} 2 & 1 & -1 & 2 \\ 1 & 2 & -2 & -1 \\ -1 & -2 & 2 & 1 \\ 2 & -1 & 1 & 2 \end{bmatrix} + \frac{k_y}{6}\frac{a}{b}\begin{bmatrix} 2 & -2 & -1 & 1 \\ -2 & 2 & 1 & -1 \\ -1 & 1 & 2 & -2 \\ 1 & -1 & -2 & 2 \end{bmatrix} \tag{2.16}$$

For the special case of the isotropic, square element where $a = b$ and $k_x = k_y = k$, the element conductivity can be simplified to the form seen in equation (2.17).

$$[k_c] = \frac{k}{6}\begin{bmatrix} 4 & -1 & -2 & -1 \\ -1 & 4 & -1 & -2 \\ -2 & -2 & 4 & -1 \\ -1 & -2 & -1 & 4 \end{bmatrix} \tag{2.17}$$

2.3 Global System of Equations

Following the application of Galerkin's Weighted Residual Method to Laplace's equation, the element conductivity matrices are assembled and boundary conditions included, arriving at the global conductivity relationship shown as equation (2.18).

$$[K_c]\{H\} = \{Q\} \tag{2.18}$$

Where $[K_c]$ is the global conductivity matrix of soil permeability, $\{H\}$ are the nodal potential vales and $\{Q\}$ are the nodal flux values. This system of equations can be solved directly by the finite element code using Gaussian elimination in order to determine the nodal potential values.

CHAPTER 3

THE FINITE ELEMENT PROGRAMS

The finite element programs used in the analyses documented herein were modified version of Program 7.2 found in Programming the Finite Element Method, 4th edition (Smith and Griffiths 2004). Program 7.2 was written specifically for solving steady-state problems in two dimensions using four-node quadrilateral elements.

Modifications were made in order to simplify the input files and customize the output files. Three separate programs were developed with this basis; one for the "Type C" fragment, and two for the "Type A" fragment. Two were required for the "Type A" fragment in order to account for the special case where the length of the horizontal cutoff wall, b, was of length zero. These three programs include subroutines to establish nodal point coordinates and boundary conditions. Previously the nodal coordinates and fixed freedoms had to be entered by hand into each input file. With the new versions of the code, these are generated automatically based on the input values of fragment dimensions L, s, T, b and k_H and k_V as shown in Figure 1.2.

Program 7.2 could also be used to generate flow nets. It was limited slightly in that only the equipotential or flow lines could be plotted with one run of the program. For example, running the program to find the desired results for the steady-state problem would give the output contours for the equipotential lines. In order to get the contours for the flow lines, boundary conditions had to be reversed in the input file and Program 7.2 run again (Pavlovsky 1933). The resulting Postscript files then had to be combined and edited in order to obtain the entire flow net.

All three of the updated versions of the code have the capacity to plot the entire flow net, with the additional convenience of adding heavy lines to indicate impermeable boundaries, in a single run of the program.

3.1 General Program Structure

The discussion for the remainder of this chapter will cover only the new versions of the program. The interested reader can further pursue Program 7.2 in Programming the Finite Element Method, 4th edition (Smith and Griffiths 2004).

The programs used in these analyses for both the "Type A" and "Type C" fragments follow the same general structures. They key differences lie in the settings of the boundary conditions. The codes can be broken down into several basic steps. First, the nodal point coordinates are assigned beginning with number one at the top left corner of the fragment as shown below in Figure 3.1. Nodes are number sequentially from left to right, top to bottom beginning with 1 and ending with *nn*, where *nn* is the total number of nodes in the mesh. Note that the number of x nodes is equal to the number of x elements plus one.

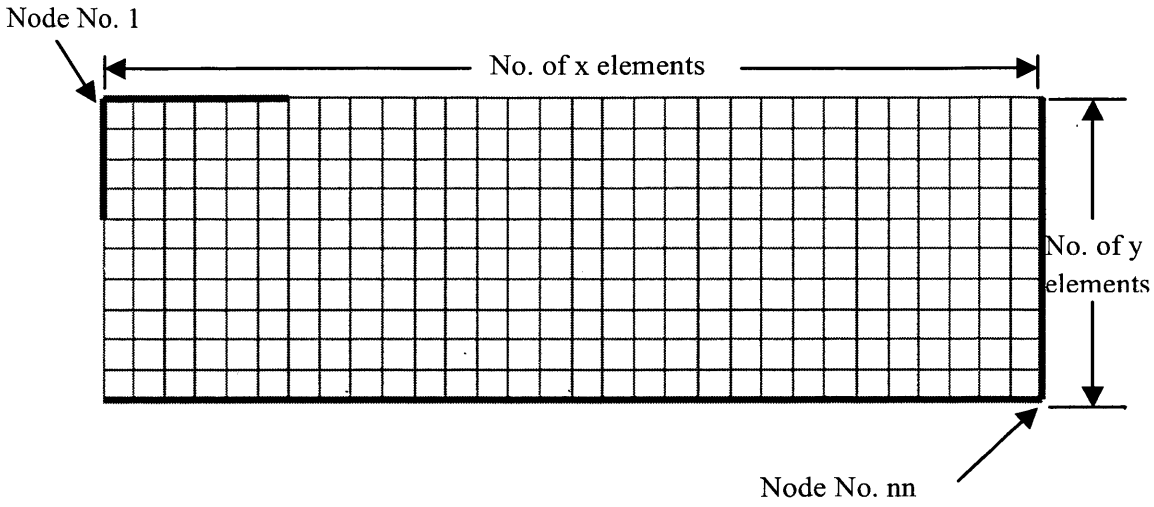


Figure 3.1 Typical mesh node numbering scheme showing a "Type A" fragment.

The nodal coordinates for these analyses were set up for square elements of one unit length on each side.

Next the program loops the elements to find the global array size and assembles the global conductivity matrix. Following that, boundary conditions are specified for the problem of interest and the solution for the equipotential lines. This phase will be discussed in more detail for the specific fragment types. After the boundary condition values are specifically assigned to the desired nodes, the global system of equations is solved and the results written to the output file.

The program then begins its second phase during which it solves for the flow line contours. It follows all of the steps described above, solving Laplace's equation, this time for the plotting of the flow lines. The key difference here is the assignment of boundary conditions, essentially they are reversed. Assigned nodal potentials are described in the following sections.

It should be noted that solving the Laplace equation twice in order to plot the flow net contours is not the only approach which generates a good flow net. The interested reader is referred to Kiousis (2004) for the Least Squares Method of Finite Element Flow Net Evaluation.

3.2 The "Type C" Program

Since these modeling efforts were first undertaken in order to examine the possibility of a critical exit gradient value for the "Type C" fragment, it was the code for these analyses which was developed first. In general, this is the reason for discussing "Type C" fragments before "Type A" fragments.

The "Type C" fragment was modeled using the program called "type_c.f95". A complete listing of the code can be found in Appendix B.

3.2.1 Required Input Parameters

The first line of input is the same used in all programs. It reads "plane' x'". This line denotes that the problem is one to be solved in place "strain" with the node numbering progressing in the x-direction. A sample input file is shown below in Figure 3.2.

```
plane' x'
100    200
1
1.0  1.0
0

60
1.0  0.0

10
```

Figure 3.2 Sample input file for a "Type C" fragment.

One of the motivations to modify Program 7.2 was to simplify the required input parameters. Rather than containing input for each of the nodal coordinates, the type_c.f95 program establishes nodal coordinates based on square, four-node quadrilateral elements with sides of one unit length. So, the second line of program input contains the number of elements in the x-direction followed by the number of elements in the y-direction.

Since the analyses were done for only one soil type, the third line of the program, *np_types*, is always set to a value of one regardless of fragment type.

The fourth line of input is where soil permeabilities are read. The first entry is for k_x, also called k_H, and the second is for k_y, also called k_V. For simplicity, these values are set to 1.0 for analyses run with isotropic soil properties and varied as simple ratios of k_H to k_V for studies of anisotropy. These studies are discussed in further detail in the following chapters.

The fifth line of the input file is where nodal loads are entered. This line is zero for all analyses conducted regardless of fragment type. This is because values of nodal flux are not being assigned.

The sixth line of input is for entry of the cutoff wall length, s. It is entered in terms of number of elements and is applied during the establishment of boundary conditions, which is discussed in further detail the following section.

Following the cutoff wall length, the boundary conditions for potential, or head, are entered. For easy analysis of results the upstream head, h_1, is always set to a value of 1.0 while the downstream head, h_2, is always set to 0.0.

3.2.2 Assigning Boundary Conditions

In the first phase of the program, the equipotential lines are plotted and the problem of interest is solved, namely that of determining nodal potential values and flow rate through the fragment. In the second phase of the program, the contours for the flow lines are determined. Each of these problems has an opposing set of boundary conditions. Figure 3.3 below shows the boundary conditions for the "Type C" fragment. In the figure, the solid lines represent impermeable boundaries and the dashed lines represent areas of fixed potential. Note that the fixed potential is constant for the entire line.

The boundary condition $h_2 = 0.0$ in Figure 3.3 (a) applies to all nodes along the top of the mesh. The boundary condition $h_1 = 1.0$ in Figure 3.3 (a) applies to all nodes between the bottom of the cutoff wall and the bottom of the fragment. The nodes at the bottom of the cutoff wall and the bottom of the fragment are not included.

In Figure 3.3 (b) all the nodes in the cutoff wall, with the exception of the very top one, are set to $h_1 = 1.0$. The boundary condition $h_2 = 0.0$ is applied to each of the nodes along the bottom of the fragment and along the right-hand side, excluding the node at the top of the mesh.

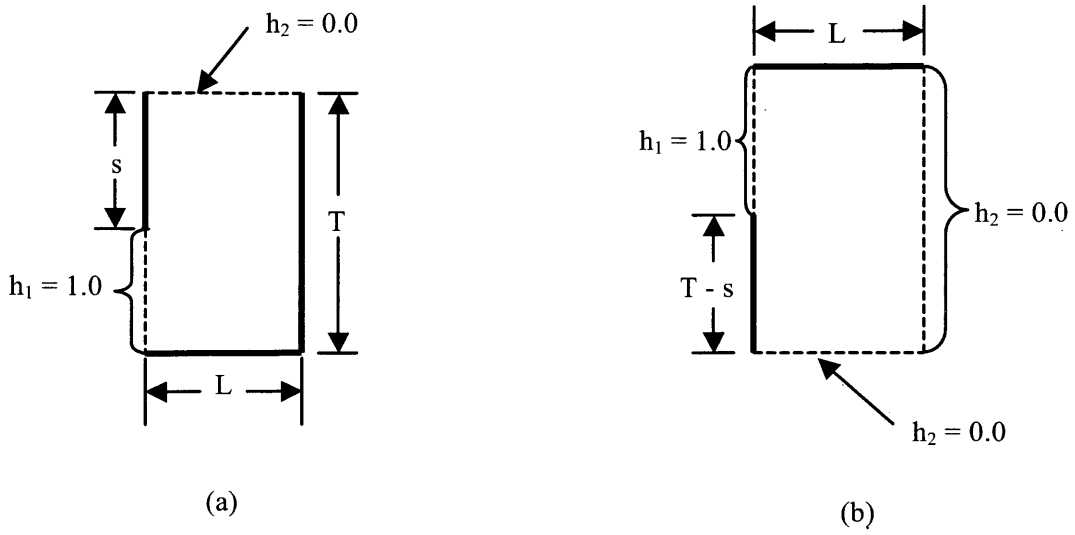


Figure 3.3 Boundary conditions for the "Type C" fragment and program type_c.f95: (a) boundary conditions for equipotential lines; (b) boundary conditions for flow lines.

3.2.3 Program Outputs

A sample of the program output file is shown below in Figure 3.4. It provides a summary of *L/T* and *s/T* ratios as well as values for the form factor, Φ, and the dimension less form of the exit gradient, $i_e s/h$.

```
   L/t        s/T     Form Factor     i_e*s/h
0.5000     0.3000      1.0451         0.6760
```

Figure 3.4 Sample output file for a "Type C" fragment.

The computations for *L/T* and *s/T* are easy to understand. *L/T* is equal to the number of elements in the x-direction of the mesh divided by the number of elements in the y-direction of the mesh. *s/T* is equal to the input s value divided by the number of y-elements in the mesh.

The form factor is calculated by equation (1.1), rearranged as shown below in equation (3.1).

$$\Phi = \frac{\bar{k}H}{Q} = \frac{\bar{k}(h_1 - h_2)}{Q} \tag{3.1}$$

The dimensionless form of the exit gradient, $i_e s/h$ is calculated as shown in equation (3.2).

$$\frac{i_e s}{h} = \frac{i_e s}{h_1 - h_2} \tag{3.2}$$

The values for i_e is the difference between the nodal potentials at the top of the mesh and one element deep. Note that the potential at the top of the mesh is always fixed to the value $h_2 = 0.0$ for all analyses, so it is not necessary to compute i_e within the program, rather we can assume it to be the potential at the node number (number of x elements + 2).

A key component of the program outputs is the flow net. The flow nets are written as Postscript files and processed using GSview software. The Postscript files are written during the subroutine countour1. The subroutine was called twice by the program, as opposed to only one time by Program 7.2, to write contours for both the equipotential and the flow lines. The boundary lines are also written and the subroutine was modified in order to plot the impermeable boundaries as dark lines, the equipotential lines in the color brown and flow lines in the color blue. Figure 3.5 shows a sample flow net output by the program type_c.f95.

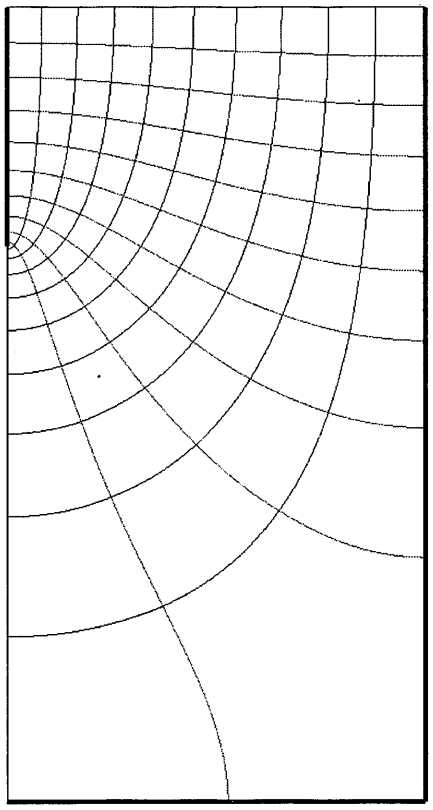


Figure 3.5 Flow net output for a typical "Type C" fragment.

3.3 The "Type A" Program

The "Type A" fragment is similar to the "Type C" fragment with two key differences. The first is that the "Type A" fragment contains a horizontal cutoff wall of length b in addition to the vertical cutoff of length s, as seen in Figure 1.1. The addition of an impermeable boundary meant that the boundary conditions needed to be established in a slightly different way than those in the "Type C" fragment.

The second difference is that the "Type A" fragment is not limited in its length and none of the charts for this sort of fragment directly use the dimensional parameter L for overall fragment length. However, since these analyses are being conducted with a finite element model, the fragments need to be discretized but still of adequate length to simulate an infinite shape. A sensitivity study was conducted to investigate the matter of how long a fragment is required to be and is discussed later on in this chapter.

The "Type A" fragment was modeled using the programs called "type_a.f95". A complete listing of each of the codes can be found in Appendix B.

3.3.1 Required Input Parameters

Much of the input file is the same as that required for the "Type C" fragment. The only changes come in the second line of the file. Since the number of required x-elements was a matter for study, it was not a good candidate for direct input data itself, although with the addition of the horizontal cutoff, the parameter b made an excellent candidate for direct input. So the second line of the input file for the "Type A" fragment analyses contains the desired length of the horizontal cutoff, b, in elements, a floating point variable called α, and the desired fragment depth, T. Note that T is again equal to the number of y-elements as it was for the "Type C" fragment. A sample input file is shown in Figure 3.6.

As mentioned previously, the number of x-elements is not entered directly into the file; rather it is calculated within the code. The number of x-elements is determined by multiplying b by α. In this manner, only α needs to be changed in order to change the entire fragment length. We can use the expression in equation (3.3) to determine the fragment length, L.

$$L = b\alpha \tag{3.3}$$

```
plane' x'

60   5    100
1
1.0   1.0
0

30
1.0   0.0

10
```

Figure 3.6 Sample input file for a "Type A" fragment.

3.3.2 Sensitivity Study

The analyses of the "Type A" fragment require modeling the typically "infinite" length fragment using "finite" elements. A sensitivity study was conducted in order to ensure that the model be able to adequately capture the "infinite" nature of the "Type A" fragment. To conduct this study the parameters for the fragment height as well as those for the horizontal and vertical cutoffs were set to fixed values while the length of the fragment was varied by changing α. For all analyses done in these efforts, T was set to equal 100 elements and both s and b were set to 60 elements. The input parameter α was varied so that the first fragment studied was 61 elements in length and increased gradually until the total fragment length was 300 elements, or $L/T = 3.0$. Table 3.1 shows a listing of input parameters and results for the analyses run as part of the sensitivity study.

Figure 3.7 shows the plotted results of the sensitivity study for form factor, Φ, and exit gradient, i_e, each plotted against L/T. It can be seen that the value of the form factor from one run to the next changes very little beginning at $L/T = 2.4$ From $L/T = 2.4$ to $L/T = 2.55$ the value of the form factor changes less than one tenth of one percent and the curve is observed to be rather flat from that point on. The same trend holds true for the exit gradient.

From these results, it was concluded that in conducting analyses of the "Type A" fragment with α set so that L/T is greater than 2.4 is a satisfactory way of modeling the "infinite" fragment.

3.3.3 Assigning Boundary Conditions

The overall structure of the programs used for analyses of the "Type A" fragment is the same as that used in analyses for the "Type C" fragment, the only differences lie in the assignment of

boundary conditions and the output of the exit gradient. The boundary conditions are much the same for the "Type A" fragment with the noted exception of the horizontal cutoff. The boundary conditions for the "Type A" fragment are shown in Figure 3.8.

Table 3.1 Sensitivity study results for "Type A" fragment mesh.

α	L	L/T	Φ	i_e
1.017	61	0.61	3.9159	0.2338
1.034	62	0.62	3.4271	0.1819
1.051	63	0.63	3.1695	0.1603
1.067	64	0.64	2.9900	0.1473
1.084	65	0.65	2.8518	0.1383
1.100	66	0.66	2.7395	0.1316
1.117	67	0.67	2.6451	0.1263
1.134	68	0.68	2.5638	0.1219
1.151	69	0.69	2.4926	0.1183
1.167	70	0.70	2.4294	0.1152
1.184	71	0.71	2.3727	0.1124
1.201	72	0.72	2.3215	0.1100
1.217	73	0.73	2.2748	0.1079
1.234	74	0.74	2.2320	0.1060
1.25	75	0.75	2.1927	0.1042
1.50	90	0.90	1.8328	0.0885
1.75	105	1.05	1.6685	0.0809
2.00	120	1.20	1.5805	0.0765
2.25	135	1.35	1.5302	0.0738
2.50	150	1.50	1.5003	0.0721
2.75	165	1.65	1.4822	0.0710
3.00	180	1.80	1.4711	0.0704
3.25	195	1.95	1.4643	0.0700
3.50	210	2.10	1.4600	0.0697
3.750	225	2.25	1.4574	0.0696
4.000	240	2.40	1.4558	0.0695
4.250	255	2.55	1.4547	0.0694
4.500	270	2.70	1.4541	0.0694
4.750	285	2.85	1.4537	0.0693
5.000	300	3.00	1.4535	0.0693

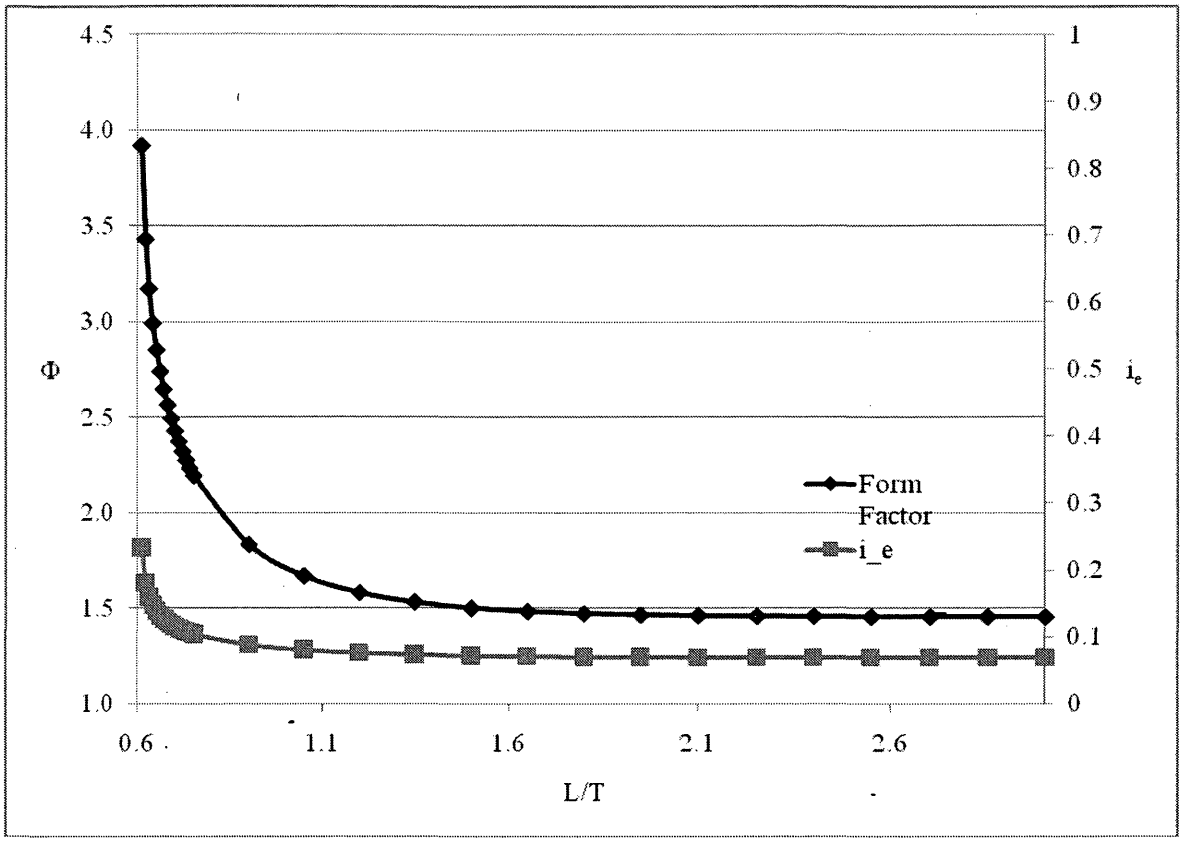
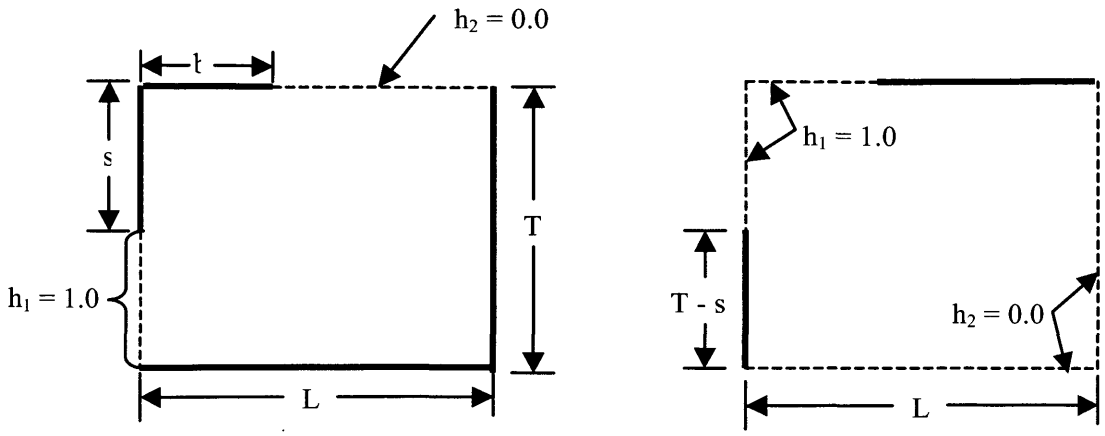


Figure 3.7 Sensitivity study results for the "Type A" fragment.

The boundary conditions for the "Type A" fragment are similar to those for the "Type C" fragment but not exactly the same. For example, for plotting the equipotential lines, as shown in Figure 3.8 (a), the node number one at the top left-hand corner is not a fixed boundary condition when b is equal to zero.

The boundary condition $h_2 = 0.0$ in Figure 3.8 (a) applies to all nodes along the top of the mesh starting from node $(b + 2)$ and ending at node number L. The boundary condition $h_1 = 1.0$ in Figure 3.8 (a) applies to all nodes between the bottom of the cutoff wall and the bottom of the fragment. The nodes at the bottom of the cutoff wall and the bottom of the fragment are not included.

In Figure 3.8 (b) the boundary conditions are reversed as was the case for the "Type C" fragment.



(a) (b)

Figure 3.8 Boundary Conditions for the "Type A" fragment and program type_a.f95: (a) boundary conditions for equipotential lines; (b) boundary conditions for flow lines.

3.3.4 Program Outputs

A sample of the program output file is shown below in Figure 3.9. It provides a summary of *bR/T* and *s/T* ratios as well as values for the form factor, Φ, the exit gradient, i_e, and the total head, *h*, in addition to the separate overall fragment dimensions.

bR/T	s/T	Form Factor	i_e	h	b	s	T	nxe
0.6000	0.3000	1.1395	0.0625	1.0000	60	30	100	300

Figure 3.9 Sample output file for a "Type A" fragment.

The computation for the form factor, Φ, is performed the same way for the "Type A" fragment as for the "Type C" fragment. The value for i_e is the difference between the nodal potentials at the top of the mesh one element from the end of the horizontal cutoff and the nodal potential directly below that. Since the nodal potential at the top of the mesh is set to 0.0, the exit gradient can be taken as the potential at the node below. Figure 3.10 shows an example "Type A" fragment mesh and where the exit gradient is taken.

Flow nets are plotted in the same way that those for the "Type C" fragment were. An example output flow net is shown below in Figure 3.11.

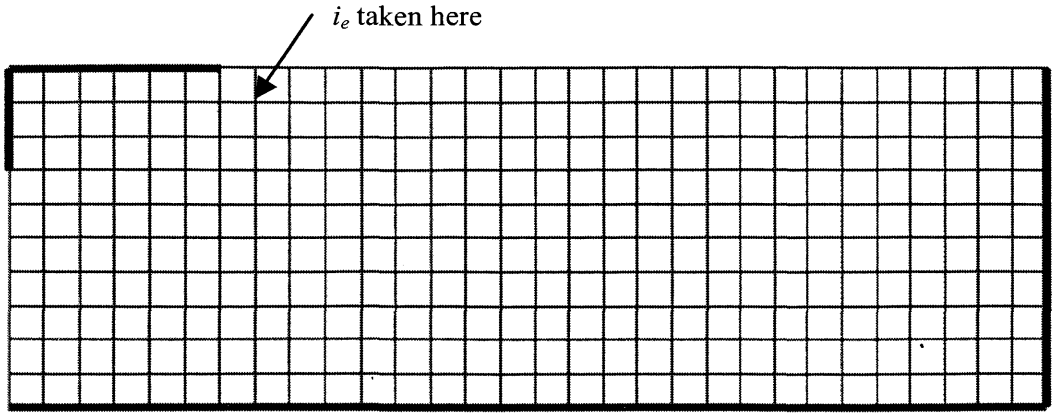


Figure 3.10 Example of the location of the exit gradient output for the "Type A" fragment

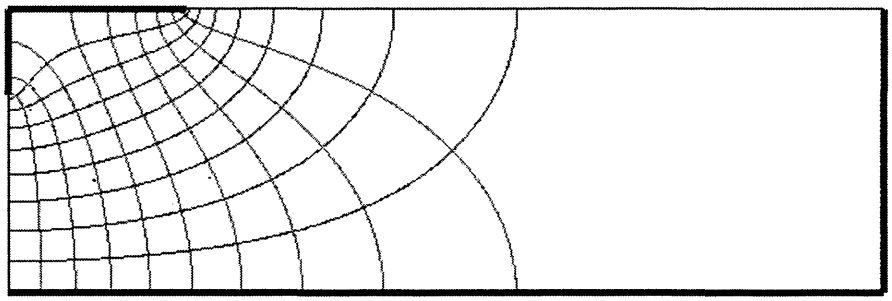


Figure 3.11 Flow net output for a typical "Type A" fragment.

CHAPTER 4

THE ANALYSES OF THE "TYPE C" FRAGMENT

The "Type C" fragment includes impermeable boundaries along the bottom and for the whole of one side with a vertical cutoff on the opposing side. As discussed in chapter 3, nodal potentials were applied as boundary conditions to model flow beneath the vertical cutoff, *s*, as shown below in Figure 4.1.

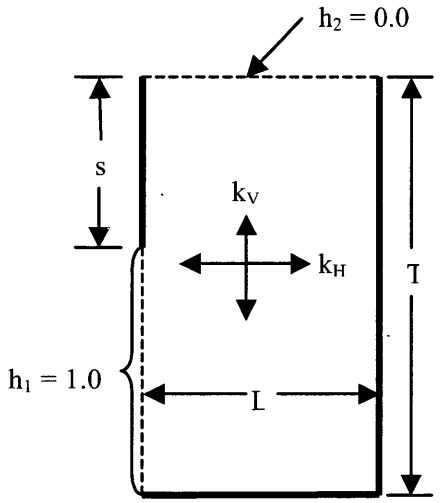


Figure 4.1 The "Type C" Fragment and established boundary conditions.

In Figure 4.1, the heavy solid lines denote impermeable boundaries and the dashed lines illustrate fixed nodal potential values. The solid permeabilities in the horizontal and vertical directions are given by k_H and k_V, respectively.

4.1 Input Parameters and Sensitivity Study

Charts previously developed for the method of fragments (Griffiths 1984) give form factors and exit gradients for suites of curves for a given *LR/T* ratio plotted against the ratio *s/T*. These charts can be seen in Appendix A. Recall the *L* is the fragment length, *R* is the anisotropy factor as defined in equation (1.3) in Chapter 1, *T* is the total fragment depth and *s* is the length of the cutoff wall.

Suites of files were set up for constant values of T for all studies. Initially studies were run with values of $T = 100$ elements. The value of s was varied from 10 to 90 elements for values of L including 10, 15, 20, 25, 30, 35, 40, 50, 60, 80 and 100. Values of L, R and T were chosen in order to match the existing charts seen in Appendix A. Note that $R = 1.0$ in these analyses, as they are conducted for soils of isotropic material properties.

A sensitivity study was conducted in order to examine the effects of a refined mesh. This study used double the number of elements setting the value of T to 200 elements and holding it constant for all studies. Doubling the number of elements is useful in determining whether the results are sensitive to the mesh density and also allows for more accurate results for smaller s/T ratios. In this series of studies, s/T ranges from 0.02 to 0.98 whereas in the initial analyses s/T was at a minimum of 0.10 with a maximum value of 0.98. For this study, as in the last, values of L were chosen in order to create curves matching those in charts already published and R is equal to 1.0.

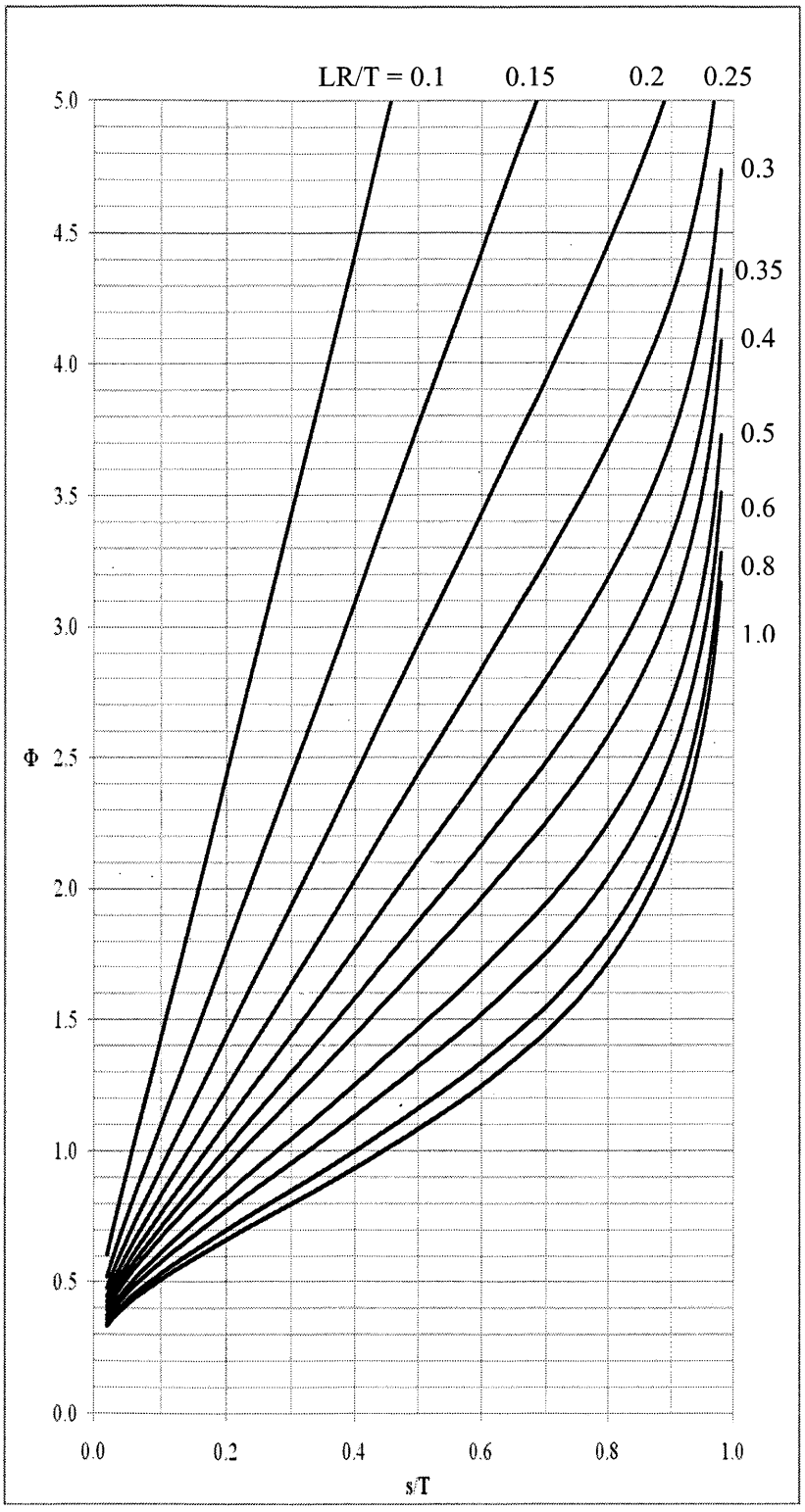
4.2 The Form Factor Chart

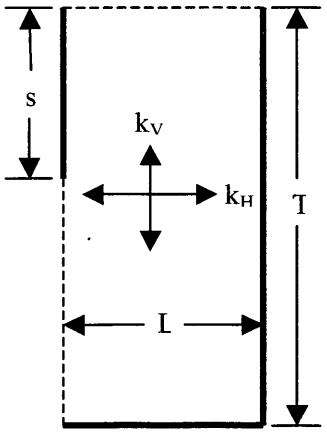
Form factor charts compare favorably to those already produced (Griffiths 1984). This was true regardless of whether T was equal to 100 elements or 200 elements. Figure 4.2 shows the finished form factor charts for $T = 200$ elements.

4.3 Examining Anisotropy

Anisotropic soils are those which have different values for permeability in the horizontal and vertical directions. As discussed in Chapter 1, the charts account for anisotropy with the factor R, computed using equation (1.3). Recall from Chapter 3 that the permeabilities in the x- and y-directions are input parameters that can be specified by the user.

A simple study was undertaken in order to validate the code's performance when anisotropic soil properties are specified. Two curves were examined to check that the code was dealing with anisotropy correctly. The plots for form factor and exit gradient were compared for $LR/T = 0.4$ and $LR/T = 0.6$ for the isotropic condition, previously run, and the anisotropic condition.





$$R = \sqrt{\frac{k_V}{k_H}}$$

Figure 4.2 Form Factor Chart for the "Type C" fragment.

The anisotropic condition was chosen for easy scaling of the input parameters. It is convenient to select $R = 2.0$ and adjust the values of L and T accordingly to match the desired LR/T ratio. In order for R to be equal to 2.0, it is easiest to keep k_H equal to 1.0 and set k_V to 4.0. applying these values to equation (1.3) will yield an R value of 2.0. Table 4.3 shows in input parameters used to study anisotropy.

Table 41 Input Parameters for Anisotropy Studies.

k_h	k_v	R	T	s_{min}	s_{max}	L	R	LR/T
1.0	4.0	2.0	200	4	196	20	1.0	0.10
1.0	4.0	2.0	200	4	196	30	1.0	0.15

Resulting form factor plots were virtually impossible to distinguish from those obtained using $k_H = k_V$. A comparison of form factors for isotropic and anisotropic soil is shown in Figure 4.3.

Comparisons of the exit gradients for anisotropic and isotropic soils showed fairly good agreement of the results, although this agreement was generally stronger for s/T values greater than 0.10. Figure 4.4 shows the exit gradient data. The exit gradient curves for the case $R = 2.0$ do not match those for $R = 1.0$ as closely for smaller values of s/T. They give slightly higher values for $i_e s/h$ but they do follow the same general shape and trends as do the curves for $R = 1.0$.

4.4 The Exit Gradient Charts

In the initial study, for $T = 100$ elements, charts were developed for the dimensionless form of the exit gradient similar to those found in Appendix A. These charts plot $i_e s/h$ against s/T for suites of curves for constant LR/T ratios. The initial study was conducted with values of s/T ranging from 0.1 to 0.9 as shown in Table 4.1. These exit gradient plots shown in Figure 4.5 are a near exact match to those found in Appendix A.

As mentioned previously in this chapter, running the analyses with a finer mesh, $T = 200$ elements, allowed for accurate analyses using smaller values of the vertical cutoff length, s. The entire suite of curves was analyzed again, this time for the values of s/T ranging from from 0.02 to 0.98. In this way the curves in the exit gradient chart could be extended as shown in Figure 4.6. It is interesting to note that each of these curves has nearly the same $i_e s/h$ value for $s/T =$ 0.02. Values of $i_e s/h$ vary from 0.6738 to 0.6786, a difference that is virtually impossible to visibly distinguish on the chart.

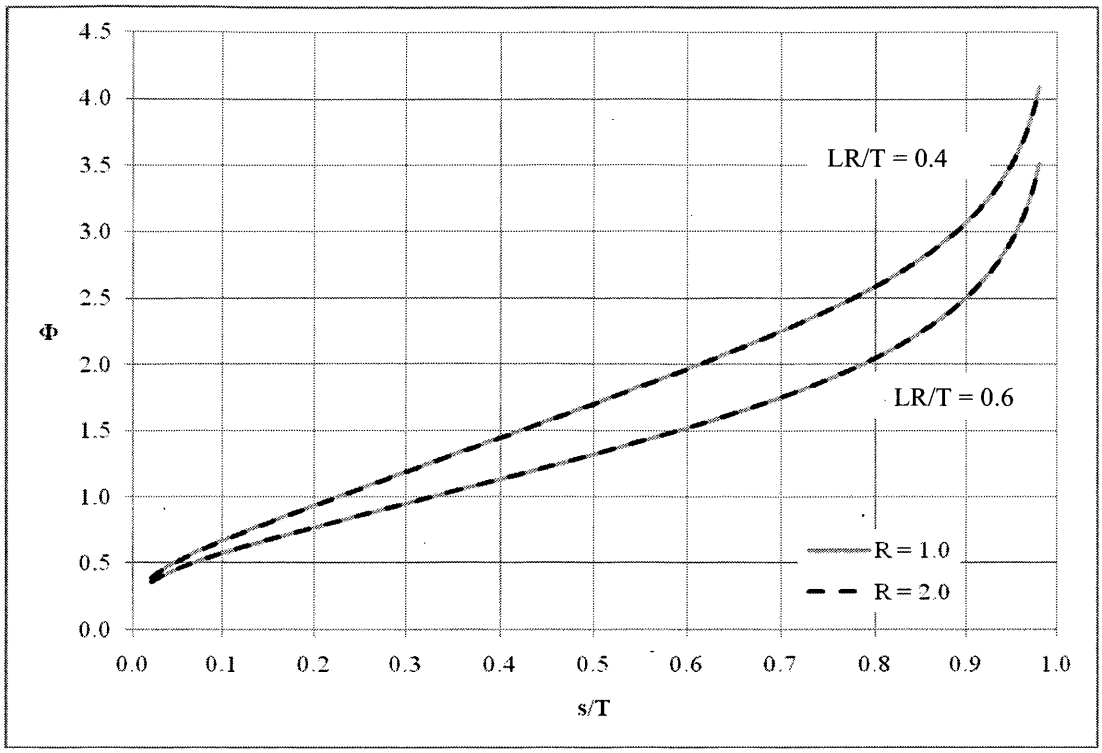


Figure 4.3 Comparison of Form Factor Plots for $R = 1.0$ and $R = 2.0$.

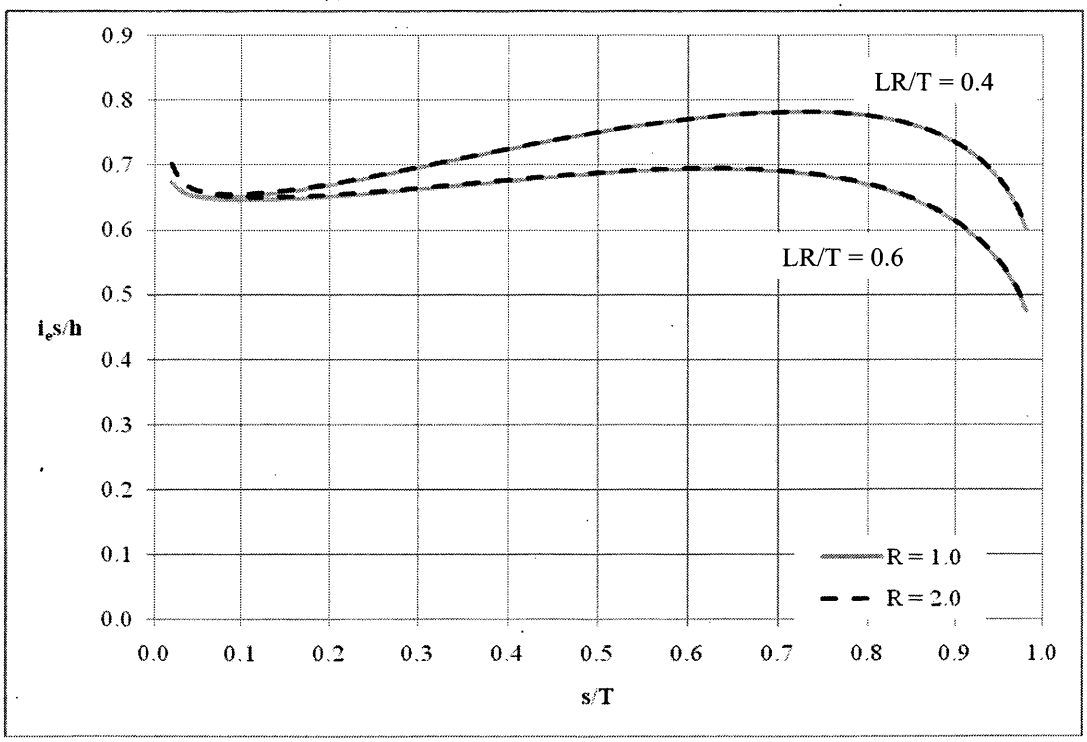


Figure 4.4 Comparison of Exit Gradient Plots for $R = 1.0$ and $R = 2.0$.

In order to examine the exit gradient, it was plotted separately against *s/T*, rather than in its dimensionless form, and shown in Figure 4.7. Of interest in this plot is the overall similarity of the curve shapes. Exit gradient is seen to drop drastically as *s/T* approaches 0.1, at which point the curves gradually flatten and exit gradient continues to deccrease but less rapidly.

4.5 Conclusions Regarding the "Type C" Fragment

The initial suite of studies conducted yielded good results with form factor and exit gradient charts matching very well to those previously developed and found in Appendix A. Good results for form factors and exit gradients were obtained for the both initial mesh and the refined mesh. Whether *T* was set to 100 elements or to 200 elements, results for isotropic soils matched on another closely.

The resutls from the anisotropy study matched the form factors extremely well. The plots for exit gradient, seen in Figure 4.4 were not quite as accurate for *s/T* values less than 0.1. Altogether, the code handles anisotropic properties well.

Closer examination of the exit gradient revealed that it is reduced most quickly when *s/T* is less than 0.1. As *s/T* increases, exit gradient continues to drop but at a much slower rate as shown in Figure 4.7.

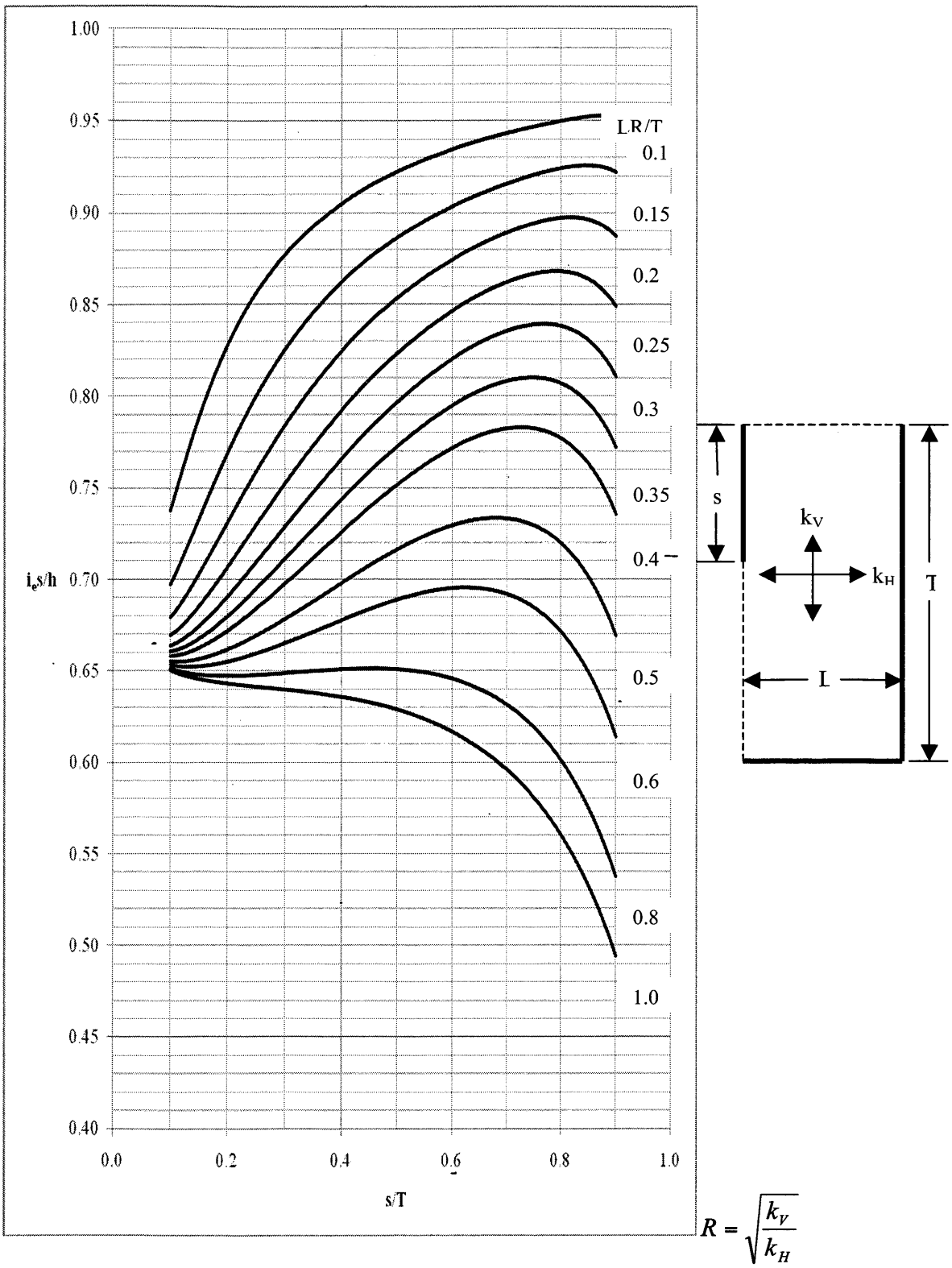


Figure 4.5 Exit gradient chart for the "Type C" fragment.

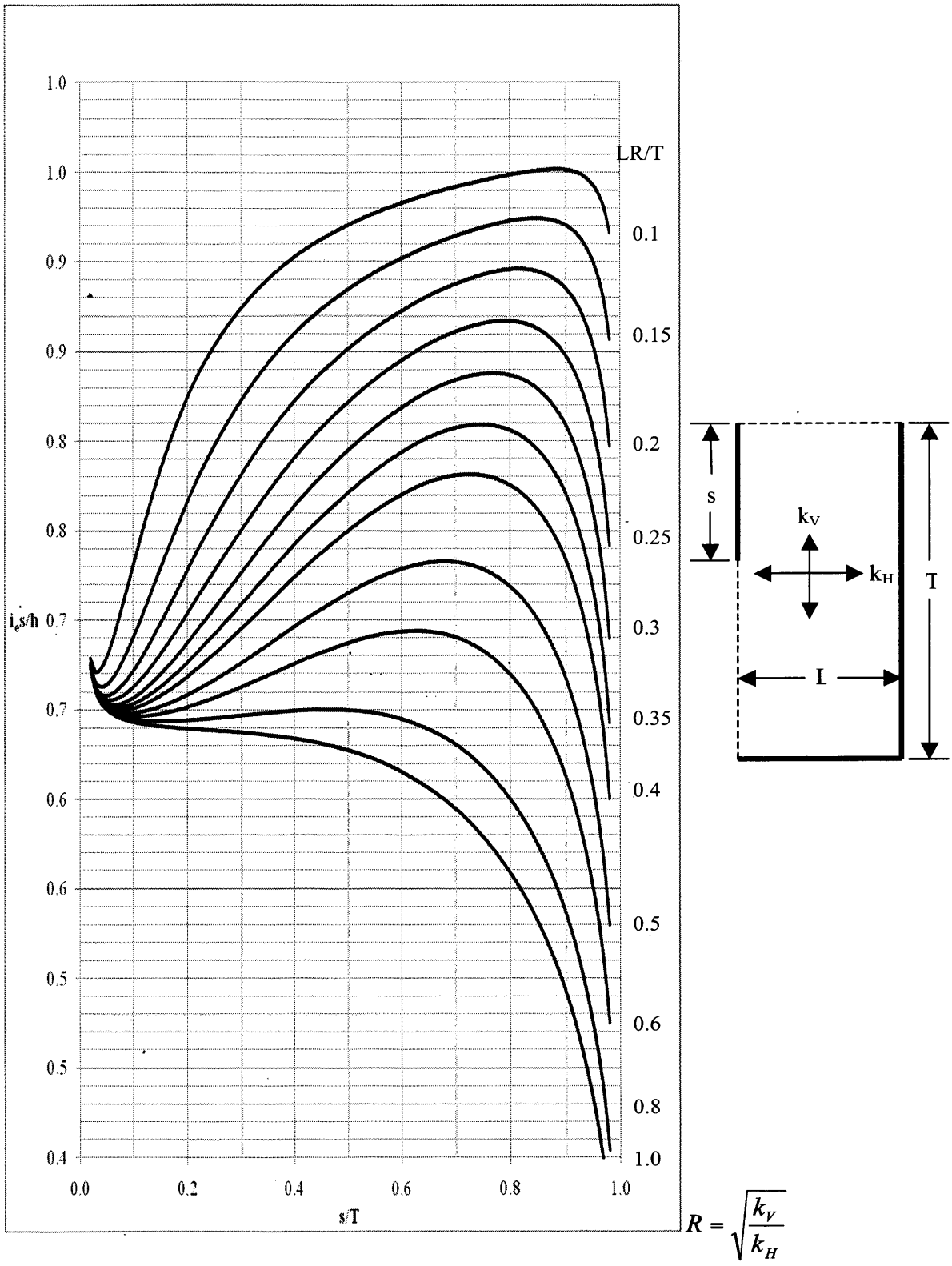


Figure 4.6 Extended exit gradient chart for the "Type C" fragment.

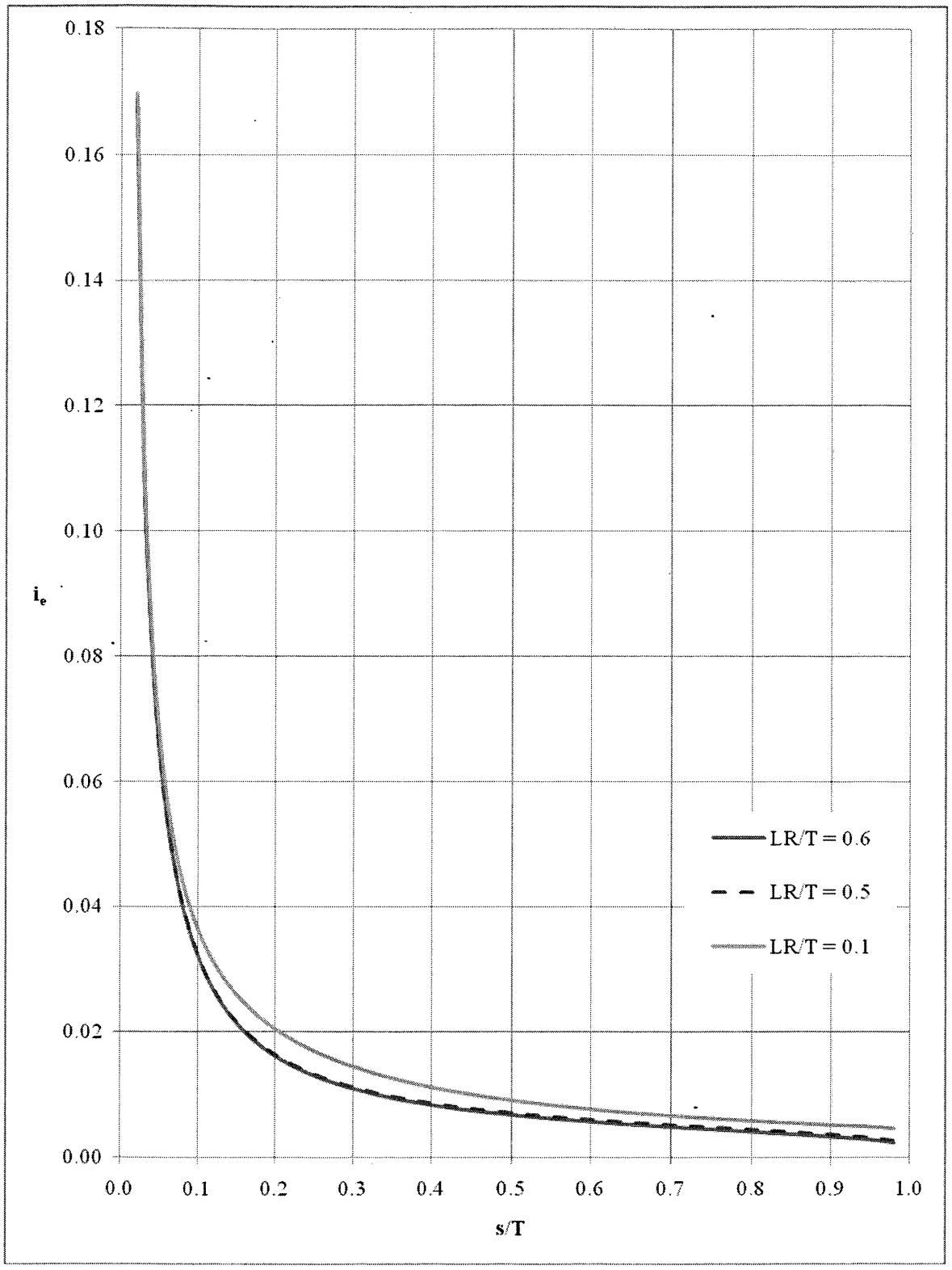


Figure 4.7 "Type C" fragment exit gradient vs. s/T for selected LR/T ratios.

CHAPTER 5

THE ANALYSES OF THE "TYPE A" FRAGMENT

The "Type A" fragment includes impermeable boundaries along the bottom and for the whole of one side with both a horizontal and a vertical cutoff on the opposing side. As discussed in Chapter 3, nodal potentials were applied as boundary conditions to model flow beneath the vertical cutoff, s, as shown below in Figure 5.1.

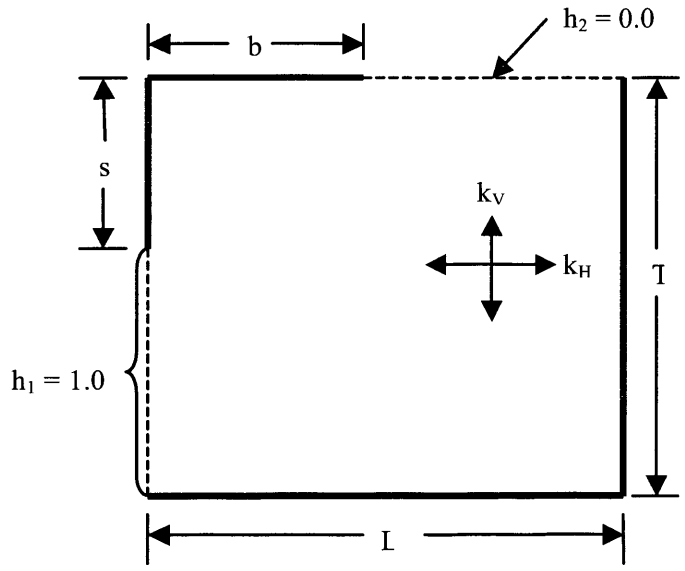


Figure 5.1 The "Type A" fragment and established boundary conditions.

In Figure 5.1, the heavy solid lines denote impermeable boundaries and the dashed lines illustrate fixed nodal potential values. The solid permeabilities in the horizontal and vertical directions are given by k_H and k_V, respectively.

5.1 Input Parameters

Charts previously developed for the method of fragments (Griffiths 1984) give form factors for suites of curves for a given *bR/T* ratio plotted against the ratio *s/T*. These charts can be found in Appendix A. The dimension b represents the horizontal cutoff, *R* is the anisotropy factor

as defined in equation (1.3) in Chapter 1, T is the total fragment depth and s is the length of the vertical cutoff wall.

Suites of files were set up for constant values of T equal to 100 elements for all analyses. The value of s was varied from 2 to 98 elements for the b values shown in Table 5.1. Note that in Table 5.2, values of α were varied in such a way as to achieve a fragment length of 300 elements.

Table 5.1 Input parameters for the suite of analyses for the "Type A" fragment.

s_{min}/T	s_{max}/T	b		L	R	bR/T
0.02	0.98	20	15.000	300	1.0	0.20
0.02	0.98	40	7.500	300	1.0	0.40
0.02	0.98	60	5.000	300	1.0	0.60
0.02	0.98	80	3.750	300	1.0	0.80
0.02	0.98	100	3.000	300	1.0	1.00
0.02	0.98	120	2.500	300	1.0	1.20
0.02	0.98	140	2.150	301	1.0	1.40
0.02	0.98	160	1.875	300	1.0	1.60

Values of b, R and T were chosen in order to match the existing charts seen in Appendix A. Note that R = 1.0 in these analyses, as they are conducted for soils of isotropic material properties.

5.2 The Form Factor Chart

Form factor chart compare favorably to those already produced (Griffiths 1984). Figure 5.2 shows the finished form factor charts for the "Type A" fragment.

5.3 The Exit Gradient Charts

Plotting rationalized chrats for the "Type C" fragment is simpler than plotting those for the "Type A" fragment. The "Type C" fragment has only one cutoff wall and the length of percolation is readily apparent as the length of that cutoff, s, and the dimensionless form of the exit hydraulic gradient can be taken as $i_e s/h$. However, in the case of the "Type A" fragment there is a horizontal cutoff, b, in addition to the vertical cutoff.

The addition of a horizontal cutoff, often referred to in design practice as an apron or as a blanket, increases the length of percolation. It has been suggested based on emperical evidence that adding a vertical cutoff to a design which already includes an apron can decrease the required

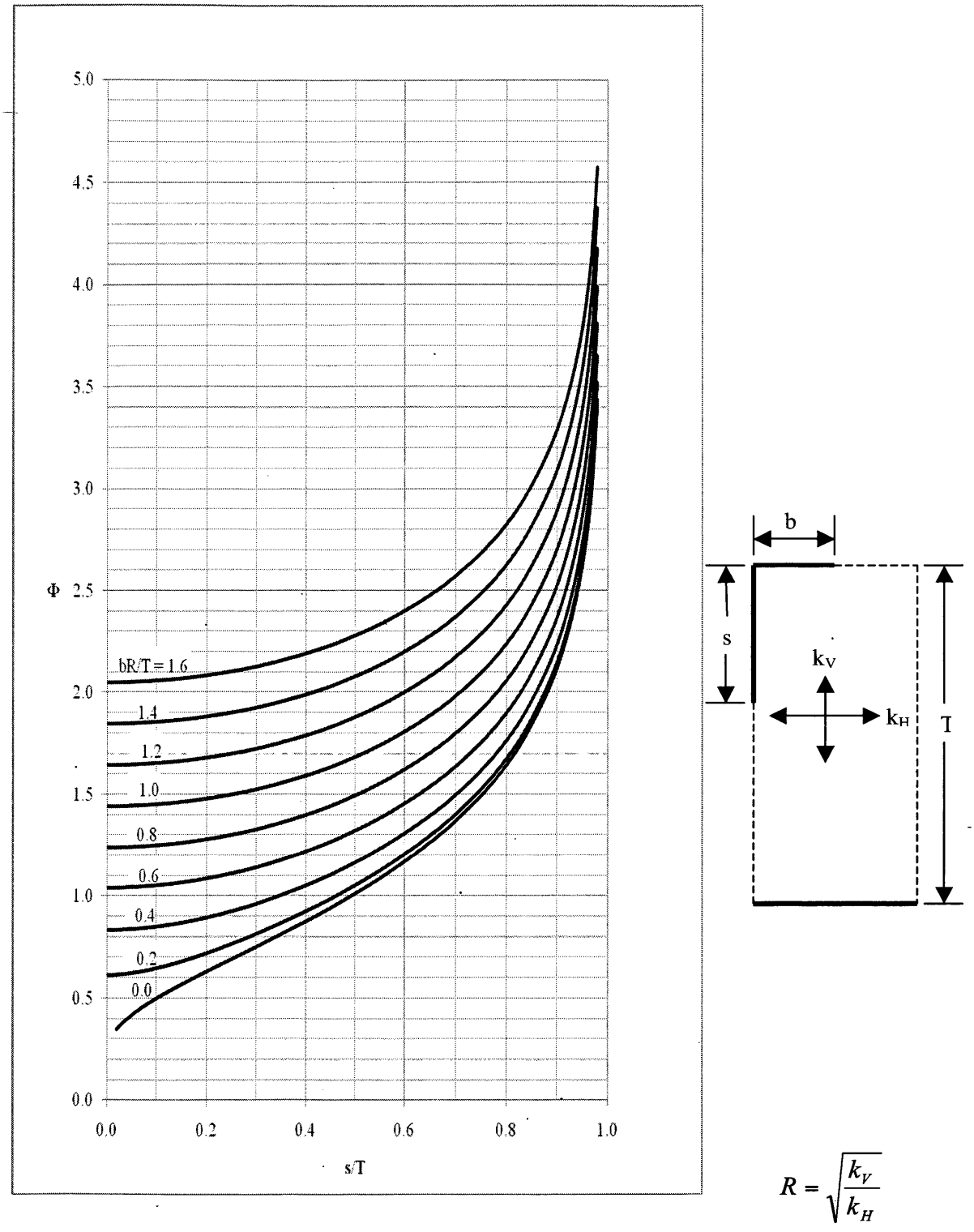


Figure 5.2 Form factor chart for the "Type A" fragment.

length of the apron by two times the cutoff wall's depth (Bligh 1910). Decreasing the apron length and adding a vertical cutoff maintins the same length of percolation as having the longer apron.

Similarly, with a "Type A" fragment the length of percolation can be taken as $s + b$. The dimensionless form of the fragment exit gradient can then be plotted as $i_e(s + b)/h$. In the case of an anisotropic soil, the factor R would be applied to the horizontal dimensions and fragment exit gradient would be plotted as $i_e(s + Rb)/h$.

The chart for the "Type A" fragment exit gradient for isotropic soil is shown in Figure 5.3. Note that this chart was developed specifically for the case that the fragment depth, T, is equal to 100 units and as such, the chart is not general. Exit gradient plots presented later in this chapter are only valid for situations in which the fragment depth is 100 units in length. Clearly this is a significant drawback that needs further investigation.

5.4 Example Problem Using the "Type A" Fragment

As mentioned in Chapter 1, the seepage quantity resulting from the computation of the example problem was deemed too large by engineers evaluating the dam. In order to reduce the quantity, designers are proposing a modification to the facility which will allow the reservoir to continue storing water during construction. The addition of an impermeable clay blanket at the upstream toe of the dam is meant to reduce the seepage to 80% of what it was before the modifaction.

Since the only change to the facility is the blanken at the downstream toe of the dam, nothing changes with regard to fragments 1 and 2, they still have the same form factors . The only change is to fragment 3, which is now become a "Type A" fragment rather than a "Type C" fragment.

Recall that the form factors for fragments 1 and 2 are $\Phi_1 = 1.0$ and $\Phi_2 = 2.11$. Fragment 3 is a "Type A" fragment with $s/T = 0.167$ and $bR/T = 1.0$ giving it a form factor of $\Phi_3 = 1.46$ from the chart. Now the flow rate can be recomputed using equation (5.1).

$$Q = \frac{\bar{k}H}{\sum_1^3 \Phi} = \frac{\sqrt{k_v k_h}H}{\Phi_1 + \Phi_2 + \Phi_3} = \frac{10^{-7}(12)}{1.0 + 2.11 + 1.46} = \frac{1.2\times10^{-6}}{4.57} = 2.63\times10^{-7}\, m^3/s/m \qquad (5.1)$$

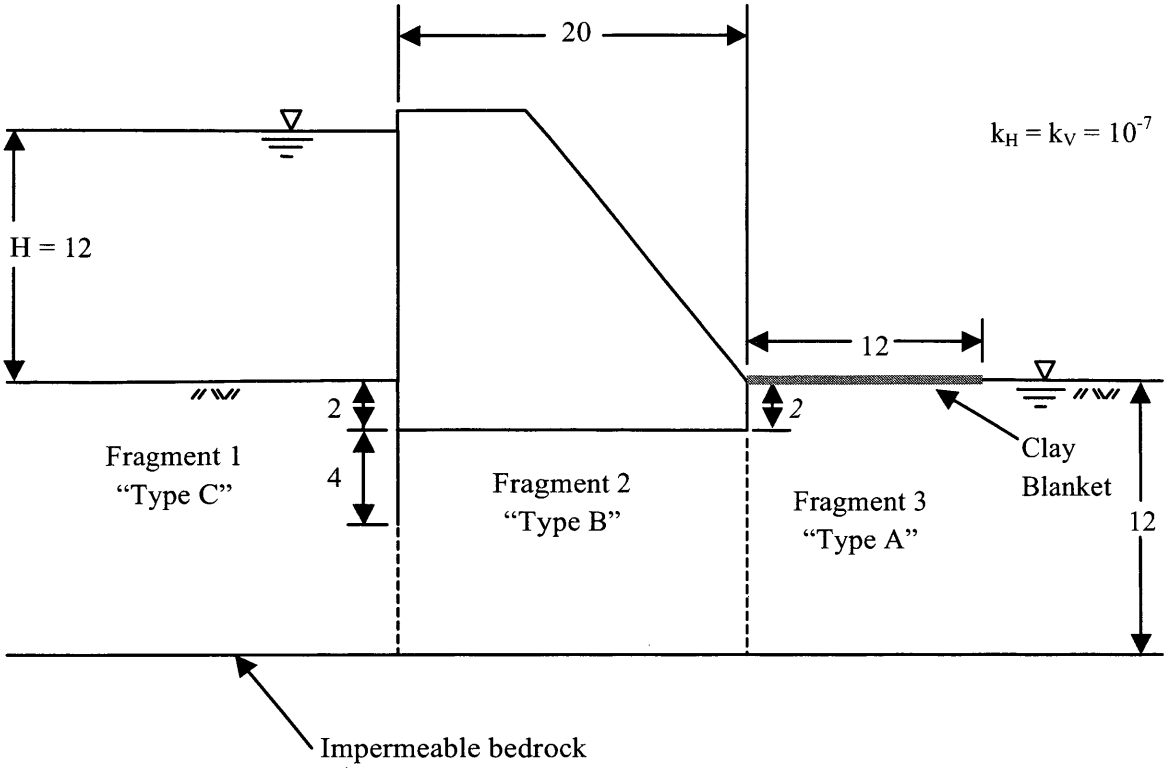


Figure 5.3 The modified example problem showing downstream clay blanket.

Note that with the addition of the clay blanket, the seepage has been reduced 20% from 3.32×10^{-7} $m^3/s/m$ to 2.63×10^{-7} $m^3/s/m$.

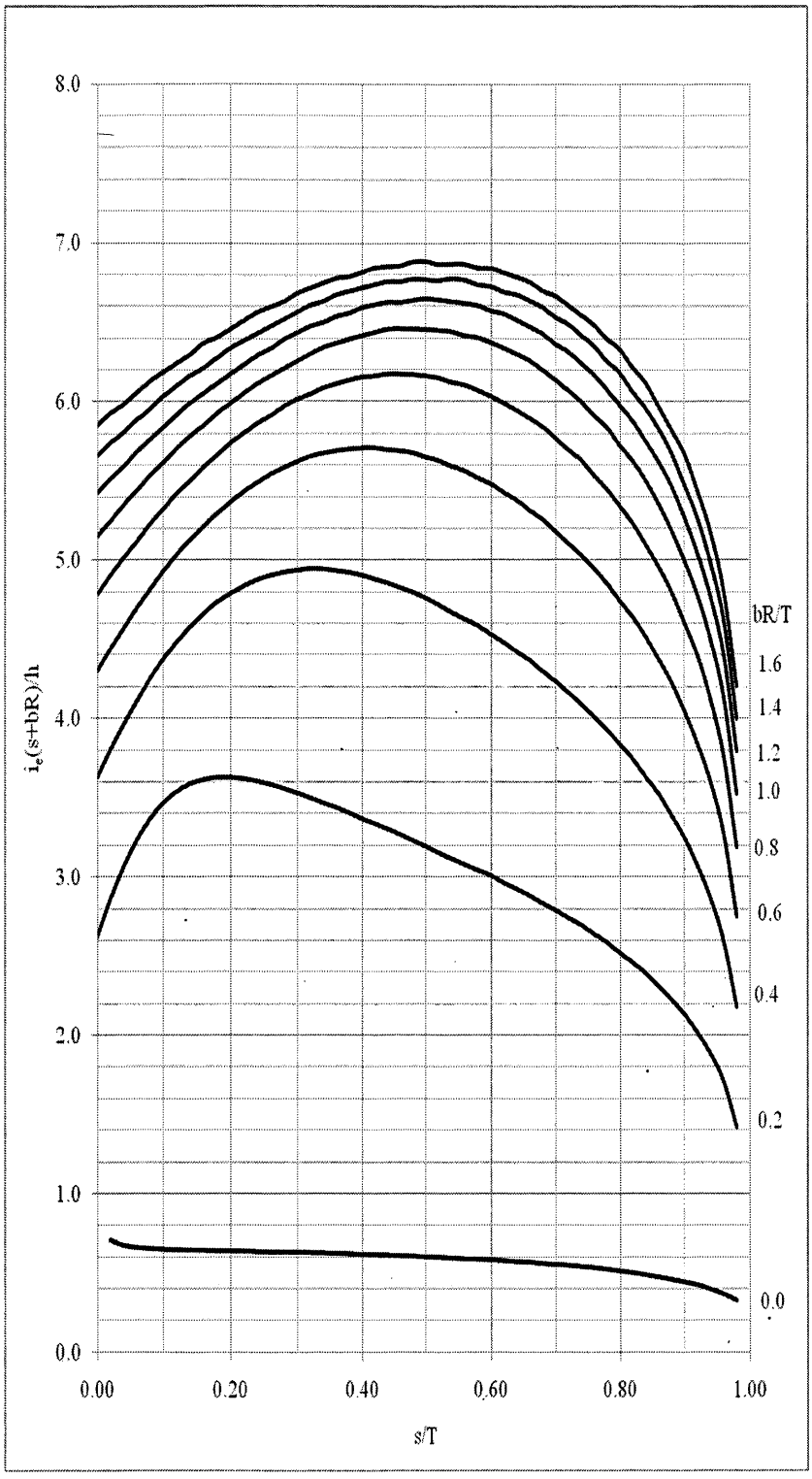
5.5 Examining Anistropy

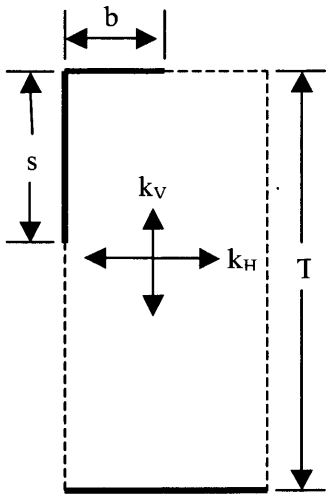
Anisotropic soils are those which have different values for permeability in the horizontal and vertical directions. As discussed in Chapter 1, the charts account for anisotropy with the factor *R*, computed using equation (1.3). Recall from Chapter 3 that the permeabilities in the x- and y-directions are input parameters that can be specified by the user.

Studies of anisotropic material properties are somewhat more involved for the "Type A" fragment than those of the "Type C" fragment. Initial studies showed very good agreement among the form factor results; however, those for exit gradient did not compare quite as well. So the suite of studies was expanded and eventually analyses were conducted for the entire suite of curves for three sets of anisotropic soil properties.

5.5.1 Input Parameters

The soil properties were chosen based upon the advice of a U.S. Bureau of Reclamation Technical Specialist. According to David R. Gillette, k_H is typically four to 20 times k_V. With





$$R = \sqrt{\frac{k_V}{k_H}}$$

Figure 5.4 Exit gradient chart for the "Type A" fragment with isotropic soil properties.

this in mind, suites of input files were set up to run for $k_H = 4k_V$, $k_H = 9k_V$, and $k_H = 16k_V$. Tables 5.2 to 5.4 shows the input parameters for the anisotropy studies conducted. Values of *s/T* ranged from 0.0 to 0.98 as in isotropic studies and studies were conducted for minimum *LR/T* values of 2.4.

Table 5.2 Input parameters for analyses with $k_H = 4\ k_V$.

Soil Properties	***bR/T***	***b***	***k_h***	***k_v***	***R***	***T***	***α***	***L***	***LR/T***
$k_H = 4\ k_V$	0	0	4.0	1.0	0.5	100	N/A	480	2.4
$k_H = 4\ k_V$	0.2	40	4.0	1.0	0.5	100	15	600	3.0
$k_H = 4\ k_V$	0.4	80	4.0	1.0	0.5	100	6	480	2.4
$k_H = 4\ k_V$	0.6	120	4.0	1.0	0.5	100	4	480	2.4
$k_H = 4\ k_V$	0.8	160	4.0	1.0	0.5	100	3	480	2.4
$k_H = 4\ k_V$	1.0	200	4.0	1.0	0.5	100	2.4	480	2.4
$k_H = 4\ k_V$	1.2	240	4.0	1.0	0.5	100	2	480	2.4
$k_H = 4\ k_V$	1.4	280	4.0	1.0	0.5	100	2	560	2.8
$k_H = 4\ k_V$	1.6	320	4.0	1.0	0.5	100	2	640	3.2

Table 5.3 Input parameters for analyses with $k_H = 9\ k_V$.

Soil Properties	***bR/T***	***b***	***k_h***	***k_v***	***R***	***T***	***α***	***L***	***LR/T***
$k_H = 9\ k_V$	0	0	9.0	1.0	0.333	100	N/A	720	2.4
$k_H = 9\ k_V$	0.2	60	9.0	1.0	0.333	100	12.0	720	2.4
$k_H = 9\ k_V$	0.4	120	9.0	1.0	0.333	100	6.0	720	2.4
$k_H = 9\ k_V$	0.6	180	9.0	1.0	0.333	100	4.0	720	2.4
$k_H = 9\ k_V$	0.8	240	9.0	1.0	0.333	100	3	720	2.4
$k_H = 9\ k_V$	1.0	300	9.0	1.0	0.333	100	2.4	720	2.4
$k_H = 9\ k_V$	1.2	360	9.0	1.0	0.333	100	2	720	2.4
$k_H = 9\ k_V$	1.4	420	9.0	1.0	0.333	100	1.714	720	2.4
$k_H = 9\ k_V$	1.6	480	9.0	1.0	0.333	100	1.5	720	2.4

Table 5.4 Input parameters for analyses with $k_H = 16\ k_V$.

Soil Properties	*bR/T*	*b*	k_h	k_v	*R*	*T*	α	*L*	*LR/T*
$k_H = 16\ k_V$	0	0	16.0	1.0	0.250	100	N/A	960	2.4
$k_H = 16\ k_V$	0.2	80	16.0	1.0	0.250	100	12.0	960	2.4
$k_H = 16\ k_V$	0.4	160	16.0	1.0	0.250	100	6.0	960	2.4
$k_H = 16\ k_V$	0.6	240	16.0	1.0	0.250	100	4.0	960	2.4
$k_H = 16\ k_V$	0.8	320	16.0	1.0	0.250	100	3.0	960	2.4
$k_H = 16\ k_V$	1.0	400	16.0	1.0	0.250	100	2.4	960	2.4
$k_H = 16\ k_V$	1.2	480	16.0	1.0	0.250	100	2	960	2.4
$k_H = 16\ k_V$	1.4	560	16.0	1.0	0.250	100	1.714	960	2.4
$k_H = 16\ k_V$	1.6	640	16.0	1.0	0.250	100	1.5	960	2.4

5.5.2 Results

The results for form factor were compiled and compared for all analyses run. In general the form factor results show good agreement. The form factor results are shown for $bR/T = 0.2$, 1.0, 1.4 and 1.6 in Figures 5.8 to 5.11. For values of bR/T less than 1.6, the form factor agreement is excellent; however, for $bR/T = 1.6$ the difference in exit gradients is easily seen in Figure 5.12. This difference is not significant but can be noted and it was not deemed necessary to develop separate form factor charts for cases of anisotropic soil properties.

Although the form factor results for analyses conducted with isotropic and anisotropic soil parameters, the exit gradient plots did not compare quite as favorably. The resulting exit gradient plot for the same four bR/T ratios are shown in Figures 5.9 to 5.12.

It is interesting to note that although the resulting exit gradient curves for soils with varying degrees of anisotropy for the "Type A" fragment do not match one another as well as those for the "Type C" fragment did, they still behave similarly. For bR/T ratios equal to or less than 1.0, the curves for soils with anisotropic soil properties are all very near one another and are difficutl to distingish in the chart. This set of curves yields higher values of the non-dimensionalized form of the exit gradient, $i_e(s + bR)/h$, than those resulting for isotropic soil. Examining plots of exit gradient alone vs s/T, the same trend is observed. It is unknown what is causing this difference and the study of this phenomenon is unfortunately beyond the scope of this project.

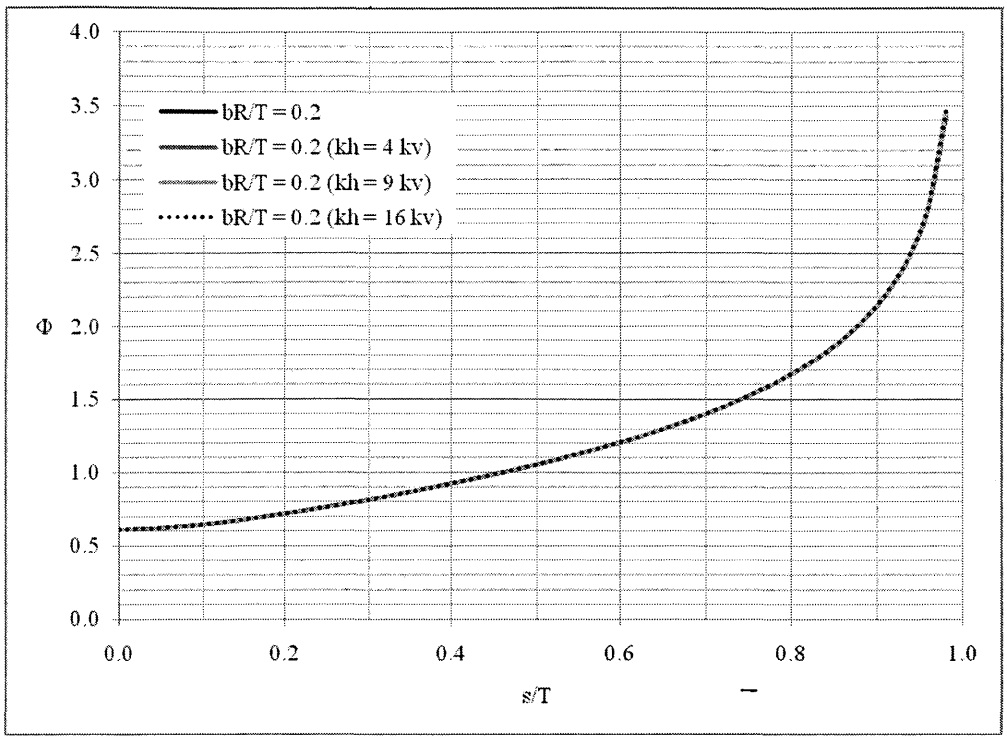


Figure 5.5 Comparison of form factor plots for $bR/T = 0.2$.

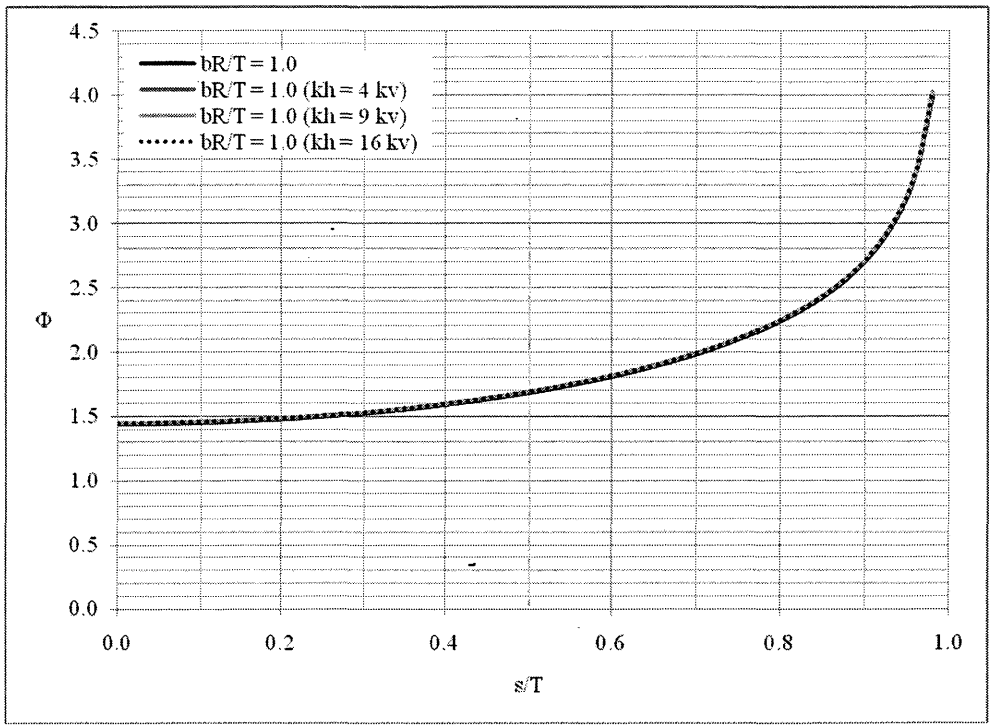


Figure 5.6 Comparison of form factor plots for $bR/T = 1.0$.

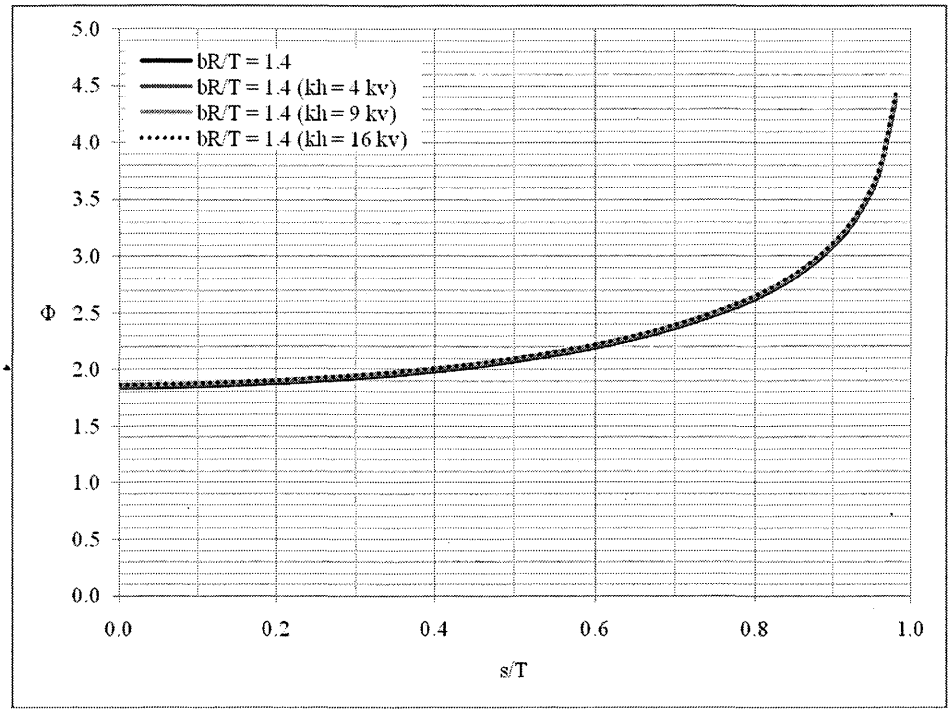


Figure 5.7 Comparison of form factor plots for $bR/T = 1.4$.

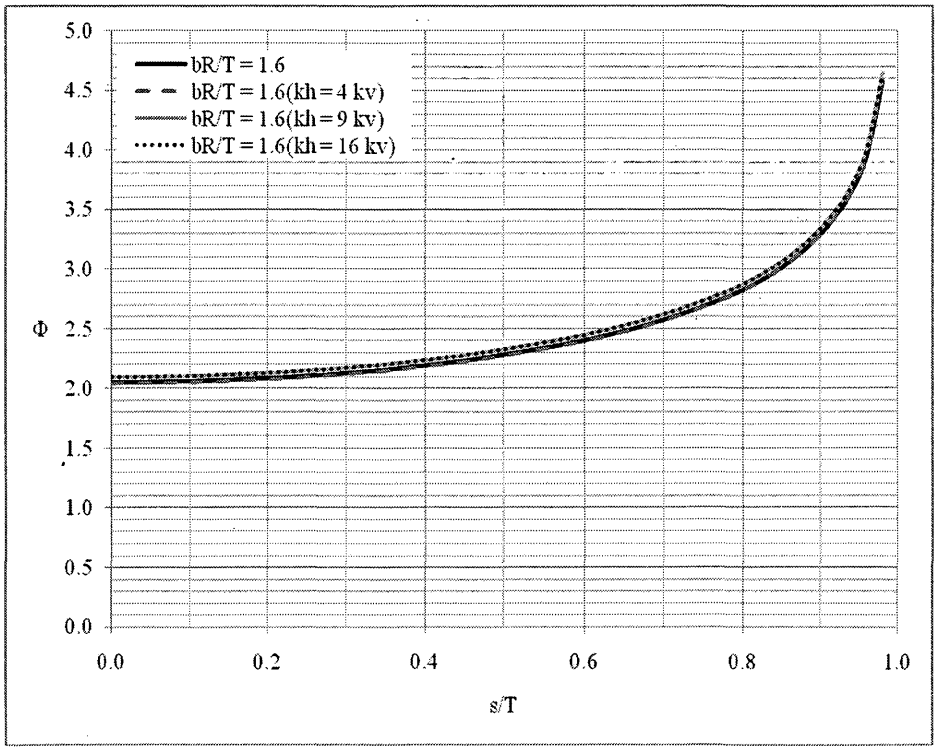


Figure 5.8 Comparison of form factor plots for $bR/T = 1.6$.

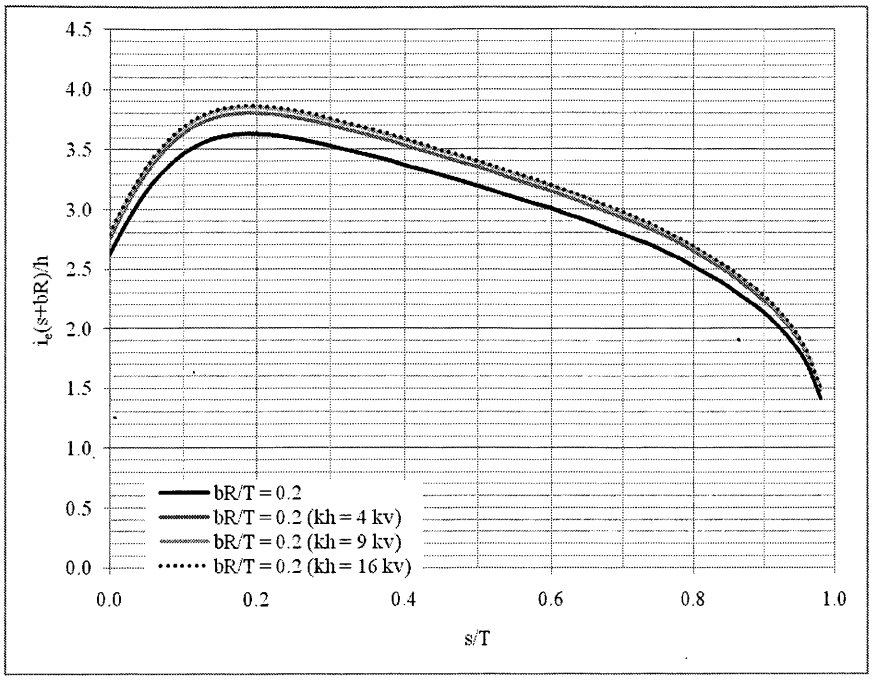


Figure 5.9 Comparison of exit gradient plots for $bR/T = 0.2$.

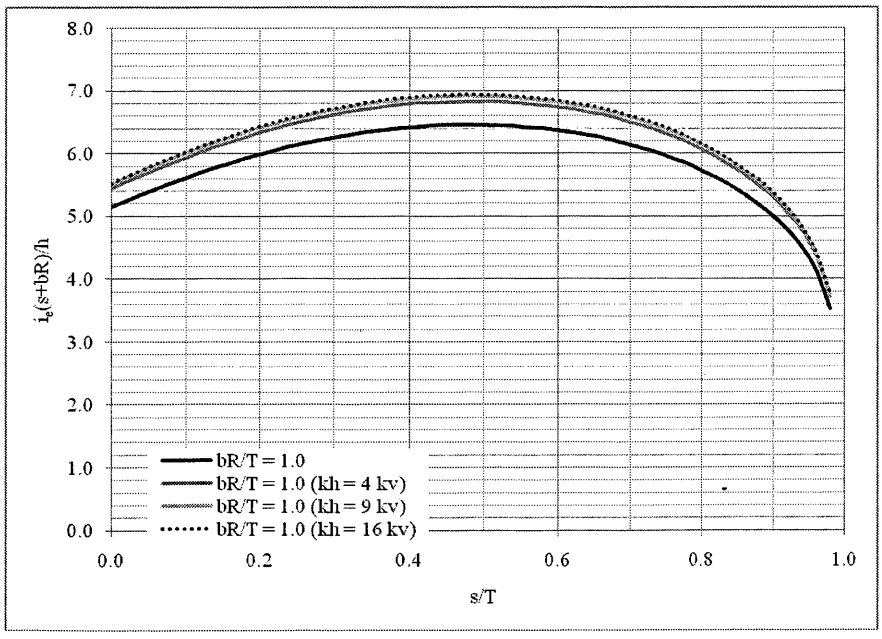


Figure 5.10 Comparison of exit gradient plots for $bR/T = 1.0$.

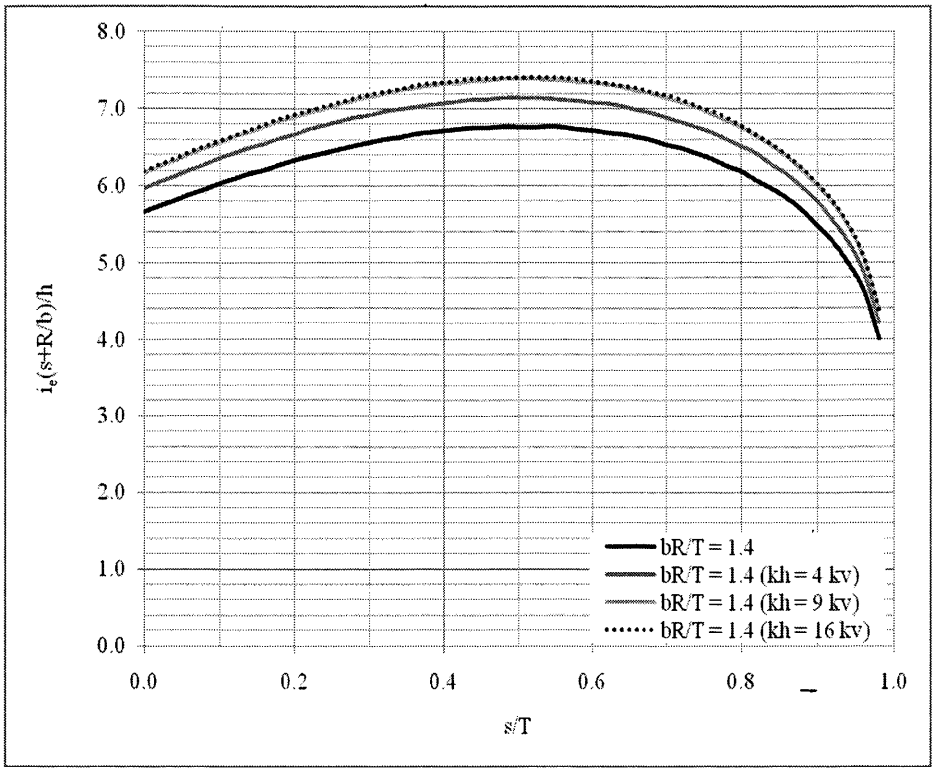


Figure 5.11 Comparison of exit gradient plots for $bR/T = 1.4$.

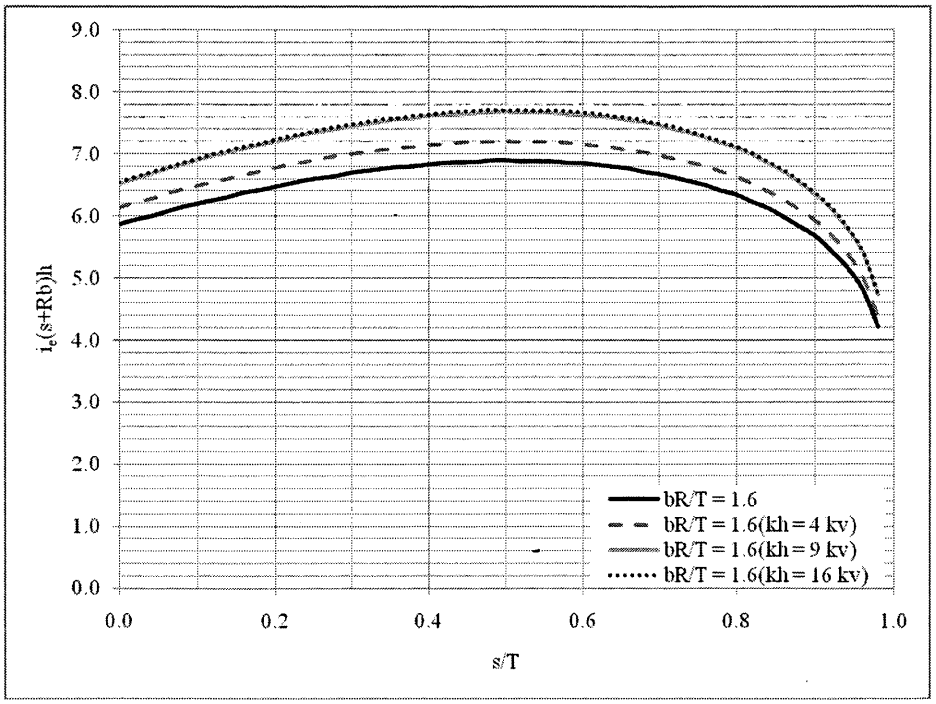


Figure 5.12 Comparison of exit gradient plots for $bR/T = 1.6$.

5.5.3 Exit Gradient Charts for Anisotropic Soils

Since there is such an apparent difference between exit gradient curves for soils with anisotropic properties, additional charts were developed to cope with three cases of anisotropy namely, $k_H = 4k_V$, $k_H = 9k_V$ and $k_H = 16k_V$. The charts are shown in Figures 5.12, 5.13 and 5.14.

5.6 Conclusions Regarding the "Type A" Fragment

The code modified for analyses of the "Type A" fragment performed well and gave excellent results for the form factor charts. In light of this, the author moved on to the next challenge, that of plotting exit gradients for the "Type A" fragment.

The exit gradient charts were plotted using the dimensionless form $i_e(s + bR)/h$. this method is employed based upon Bligh's emperically derived statement that the length of percolation can be taken as the length of the cutoff wall, s, plus the length of the downstream blanket, b. The factor R is included as a multiplier of the horizontal dimensionin order to account for anisotropi soils. The exit gradient charts were developed for a fragment depth of $T = 100$ units length and can be utilized to find the exit gradient, and subsequently the factor of safety against instability, for problems involving dams with downstream clay blankets with a depth to bedrock of 100 units.

In studies of inputs for anisotropic soil properties, the resutls for the form factor charts compared very well. Unfortunately, agreement for exit gradient plots was not as good. It is possible that reaching some sort of way of normalizing exit gradient results would resolve this contradiction.

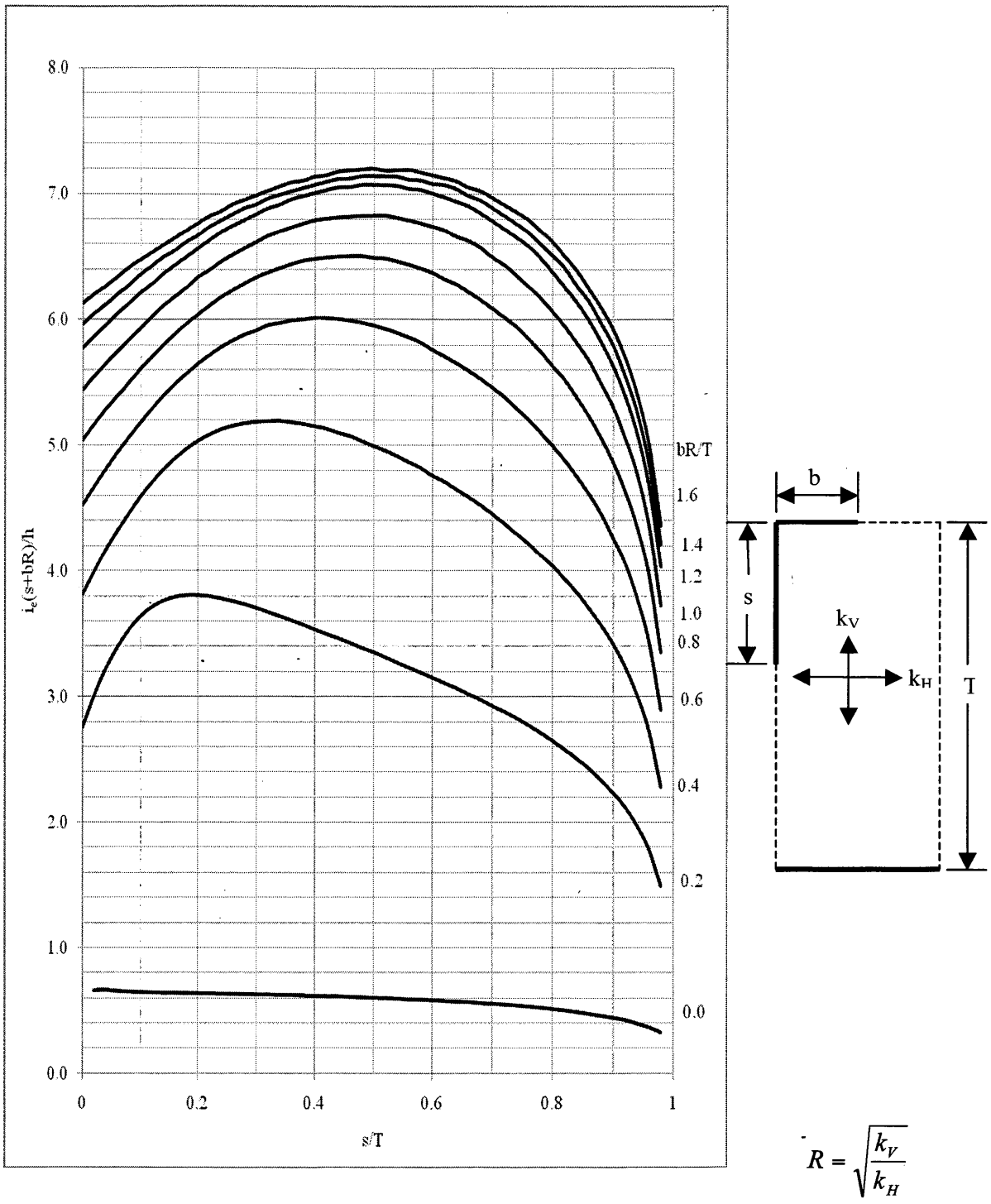
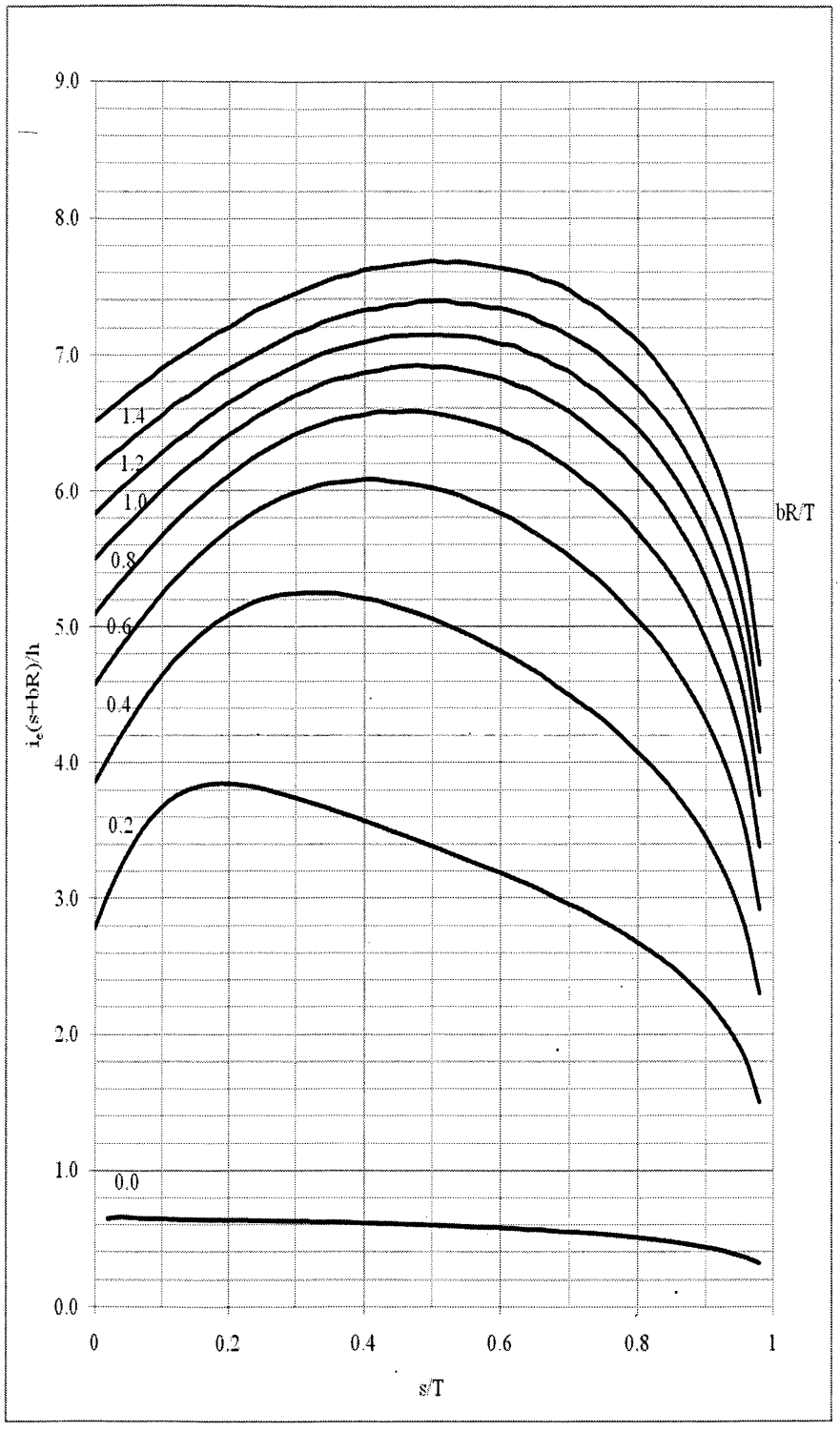
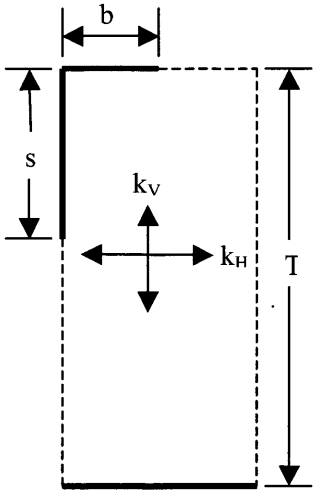


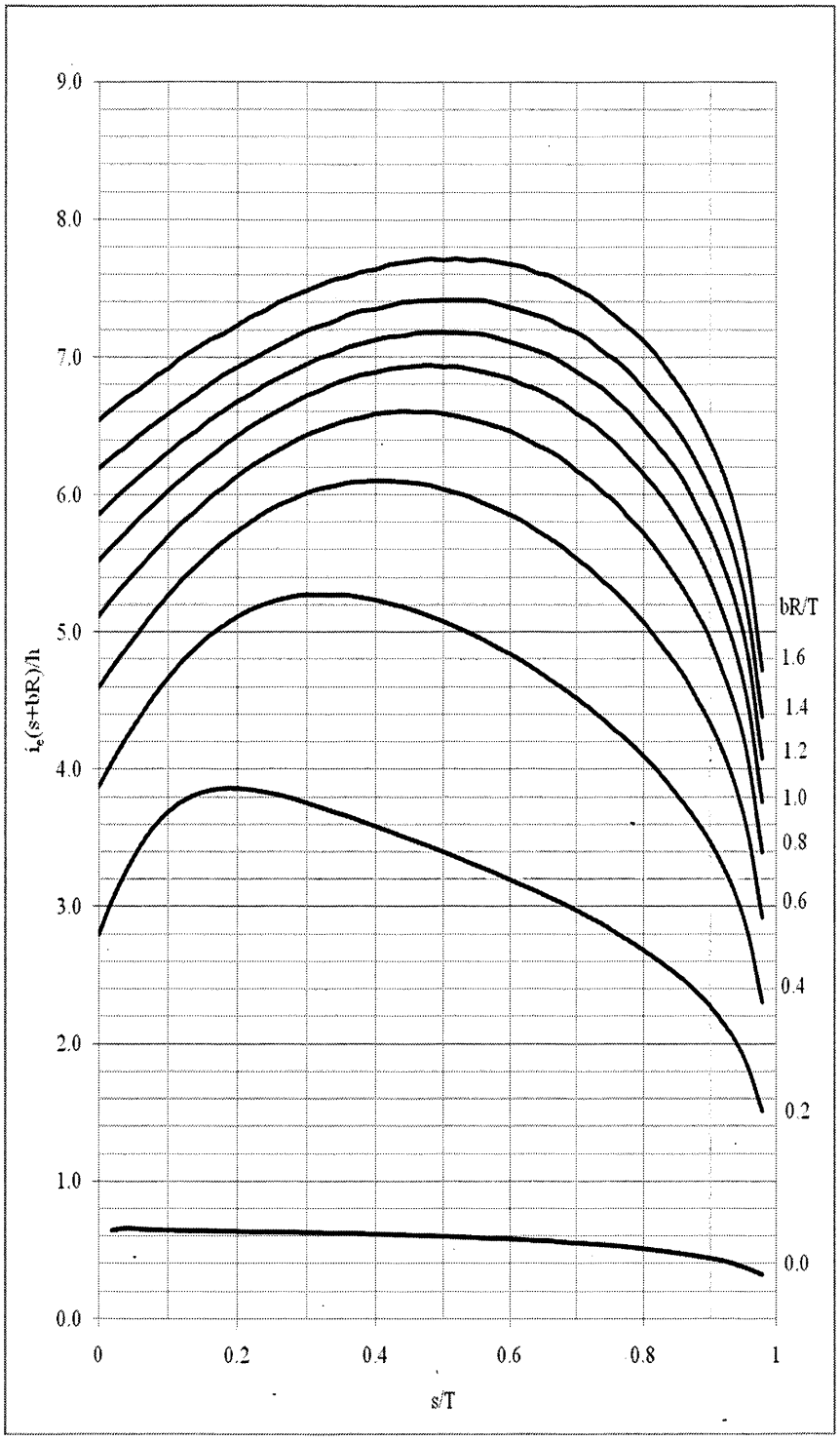
Figure 5.13 Exit gradient charts for the "Type A" fragment with $k_H = 4k_V$.

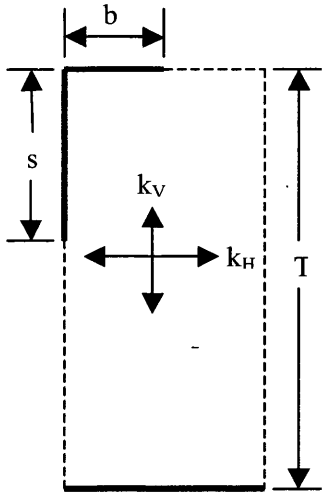




$$R = \sqrt{\frac{k_V}{k_H}}$$

Figure 5.14 Exit gradient charts for the "Type A" fragment with $k_H = 9k_V$.





$$R = \sqrt{\frac{k_V}{k_H}}$$

Figure 5.15 Exit gradient charts for the "Type A" fragment with $k_H = 16k_V$.

CHAPTER 6

CONCLUSIONS

With the computing tools available today, a finite element approach is a good way to model problems of steady-state seepage. A few modifications made to Program 7.2 (Smith and Griffiths 2004) enabled the analyses of two fragment types, "A" and "C", for many different configurations of each of the fragments with relative ease. Analyses yielded results for fragment form factors and exit gradients for each of the two fragment types.

6.1 The "Type C" Fragment Conclusions

Studies were conducted using the modified finite element code with two different mesh densities so that the second suite of studies done had a mesh twice as fine as the first suite of studies. Results from the first and second studies did not show very much mesh sensitivity and form factor and exit gradient charts matched very well to those previously generated by others (Griffiths 1984).

The code was also exercised for soils with anisotropic permeabilities and was found to have performed well. The results for form factors matched very well and those for exit gradient were very accurate for values of s/T greater than 0.1.

One of the main purposes of the analyses was to examine what effect changing s/T has on the exit gradient. It was determined that increasing s/T always results in decreasing exit gradient, as the length of percolation is increased. However, this effect is most pronounced for values of s/T less than 0.1. That is as s/T increases from 0.02 to 0.1, the rate of decrease of exit gradient is highest. As s/T continues to increase the exit gradient continues to fall but at a reduced rate. As seen in Figure 4.7, the exit gradient decreases the least rapidly when s/T is greater than 0.4 and the rate of change of exit gradient with increased cutoff length appears to be approximately linear from that point on.

6.2 The "Type A" Fragment Conclusions

The code modified for analyses of the "Type A" fragment performed well and gave excellent results for the form factor charts. In light of this, the author moved on to the next challenge, that of plotting exit gradients for the "Type A" fragment.

As far as this author has been able to determine exit gradient charts do not exist for the "Type A" fragment and a dimensionless form for plotting the charts has yet to be determined. The exit gradient charts were plotted using the dimensionless form $i_e(s + bR)/h$. This method is employed based upon Bligh's emperically derived statement that the length of percolation can be taken as the length of the cutoff wall, s, plus the length of the downstream blanket, b. The factor R is included as a multiplier of the horizontal dimensionin order to account for anisotropi soils. The exit gradient charts generated in this thesis can be utilized to find the exit gradient for cases when the depth of the soil layer in a "Type A" fragment is 100 units.

In studies of inputs for anisotropic soil properties, the resutls for the form factor charts compared very well. Unfortunately, agreement for exit gradient plots was not as good. It is possible that exit gradient outputs would be improved by taking output from nodes further from the end of the horizontal blanket but these values would likely be lower and therefore less conservative for use in design practice.

6.3 Future Work

The exit gradient charts for the "Type A" fragment developed in this thesis are of limited value since they only apply if the depth of the fragment is 100 units. Determining a way to normalize the charts for any dimensions of the "Type A" fragment would make them of more broad practical use to engineers.

REFERNCES CITED

Azizi, F. (1999). *Applied Analyses in Geotechnics*. Chapter 3. London: Spon Press.

Bligh, W.G. (1910). Dams, Barrages and Weirs on Porous Foundations. *Engineering News,* Vol. 64, No. 26.

Griffiths, D.V. (1984). Rationalized chars for the method of fragments. *Geotechnique* 34, No. 2, pp. 229-238.

Harr, M.E. (1962). *Groundwater and Seepage,* Chapter 6. New York, NY. McGraw-Hill.

Holtz, R.D. and W.D. Kovacs. (1981). *An Introduction to Geotechnical Engineering,* pp. 258-270. Englewood Cliff, NJ. Prentice-Hall.

Kiousis, P.D. (2002). Least-Squares Finite-Element Evaluation of Flow Nets. *Journal of Geotechnical and Geoenvironmental Engineering*, pp. 699-701.

Pavlovsky, N.N. (1933). Motion of water under dams. *1st Congress on Large Dams, Stockholm,* pp. 179-192.

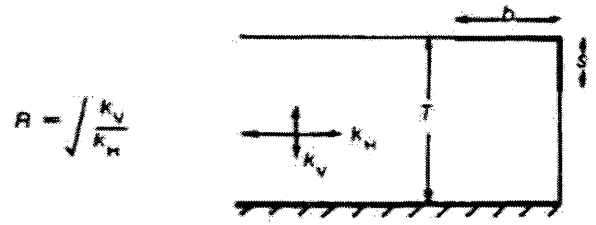
Polubarinova-Kochina, P. Ya. (1962). *Theory of the motion of ground water,* Chapter 3. Princeton University Press.

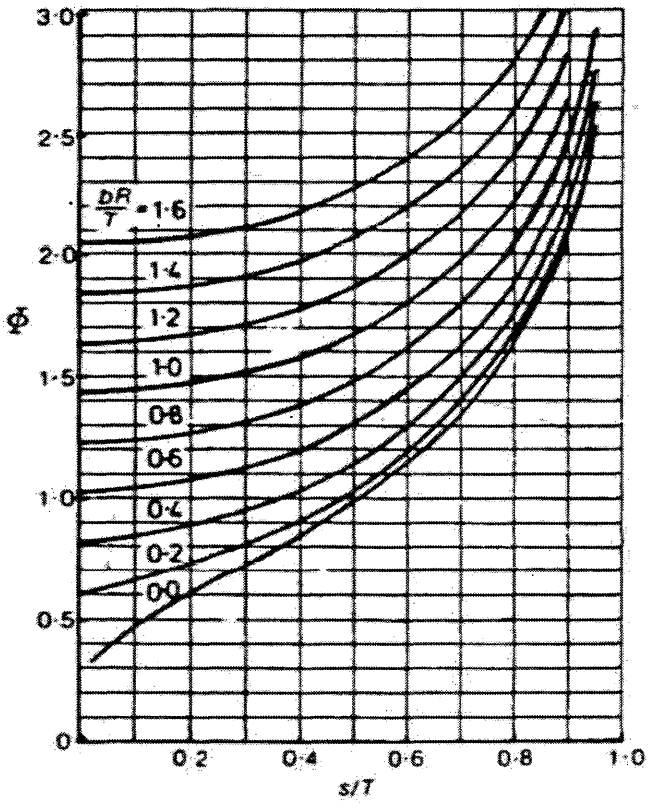
Smith, I.M. and D.V. Griffiths. (2004). *Programming the finite element method,* Chapter 7. John Wiley and Sons, Ltd.

APPENDIX A

GRIFFITHS, D. V. (1984). *Géotechnique* 34. No. 2. 229–238

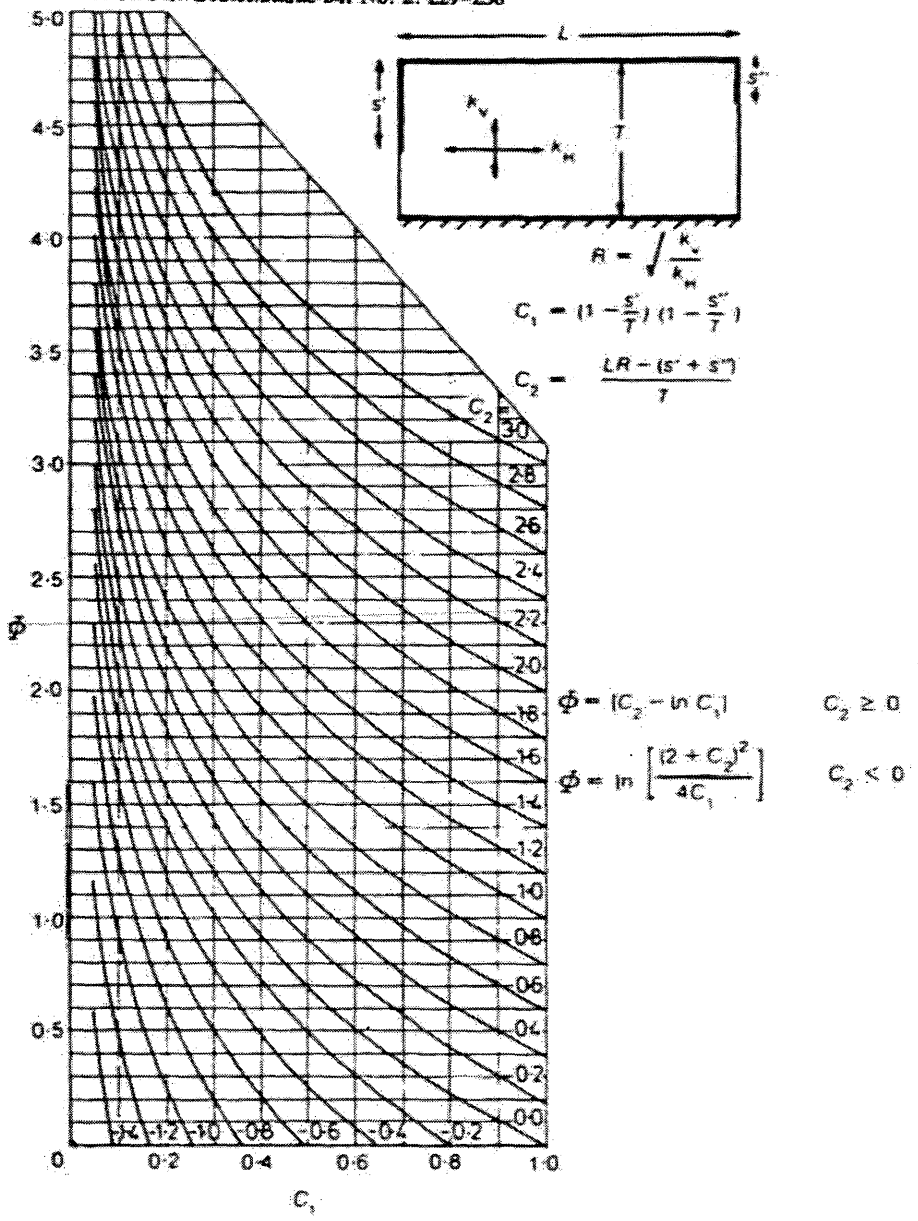
CHARTS FOR SEEPAGE IN CONFINED FLOW





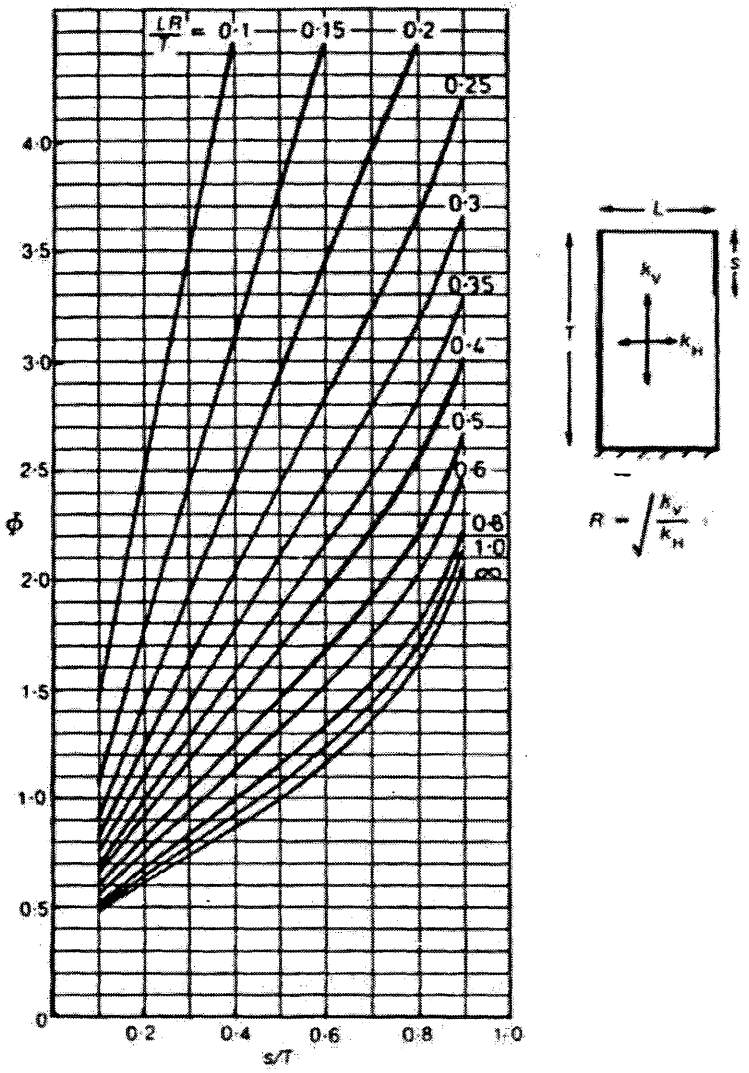
Form factor for fragment A (rearranged from Polubarinova-Kochina, 1962)

GRIFFITHS, D. V. (1984). *Géotechnique* **34**, No. 2. 229–238



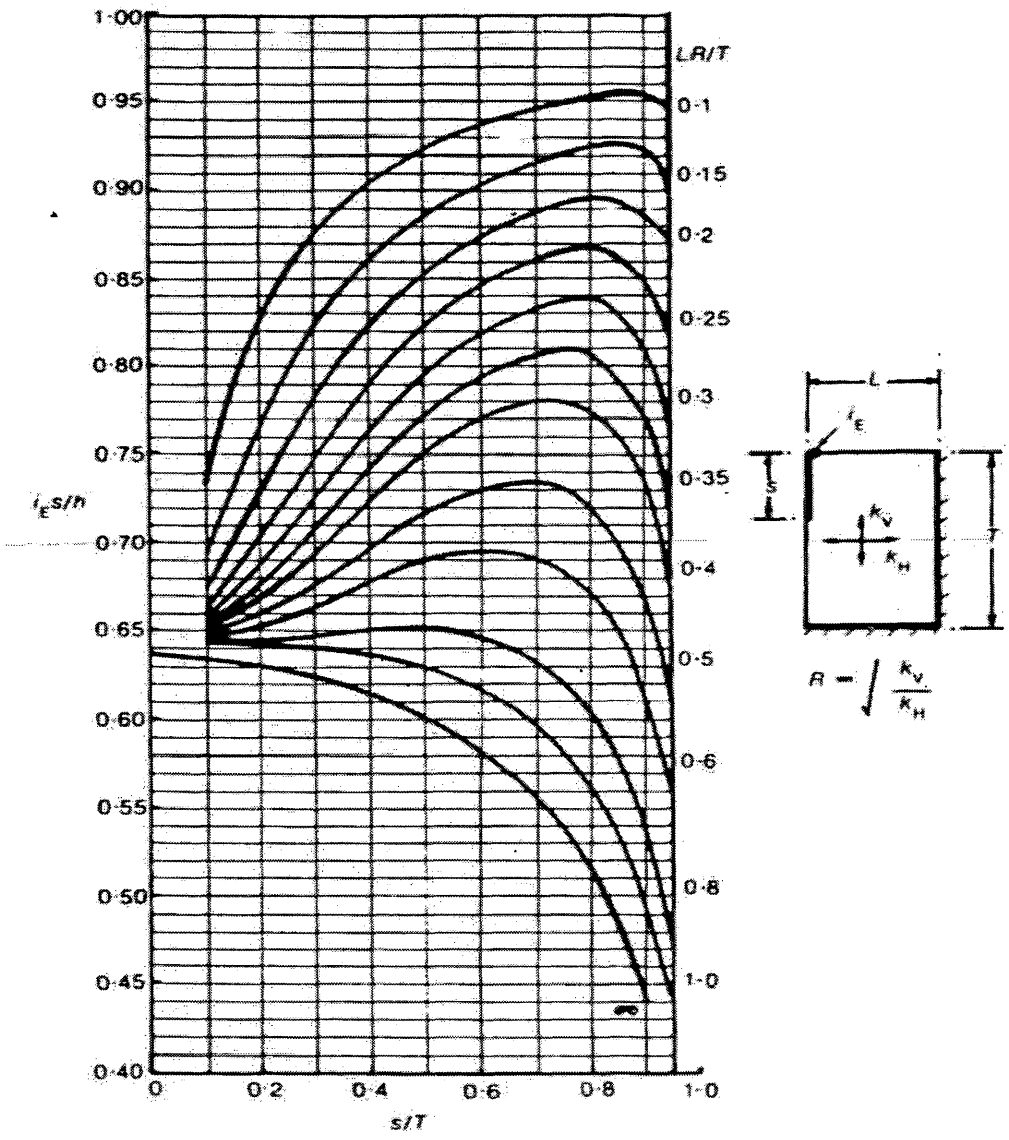
Form factor for fragment B (condensed from Harr, 1962)

GRIFFITHS, D. V. (1984). *Géotechnique* 34, No. 2, 229–238



Form factor for fragment C (from finite element solution)

GRIFFITHS, D. V. (1984). *Géotechnique* 34, No. 2, 229–238



Exit gradients for fragment C (from finite element solution)

APPENDIX B

Program Listing for type_c.f95

```
!     Last change:  JEH   07 Dec 2009    4:26 pm
PROGRAM type_c
!------------------------------------------------------------------------
! Program 7.2 Plane or axisymmetric analysis of steady seepage using
!             4-node rectangular quadrilaterals. Mesh numbered
!             in x(r)- or y(z)- direction.
!------------------------------------------------------------------------
 USE main
 USE geom
 IMPLICIT NONE
 INTEGER,PARAMETER::iwp=SELECTED_REAL_KIND(15)
 INTEGER::fixed_freedoms,i,iel,k,loaded_nodes,nci,ndim=2,nels,neq,nip=4,  &
   nod=4,nn,np_types,nxe,nye,nlen,z,zztop,s,test,yytop,nx_nod,ny_nod,     &
   fix_free,f
 REAL(iwp)::x,y,det,one=1.0_iwp,penalty=1.0e20_iwp,zero=0.0_iwp,          &
   h_one,h_two,flow_rate,i_e,kbar,hdiff
 CHARACTER(LEN=15)::dir,element='quadrilateral',type_2d,argv
!-----------------------dynamic arrays-----------------------------------
 INTEGER,ALLOCATABLE::etype(:),g_num(:,:),kdiag(:),node(:),num(:),        &
   setype(:),sg_num(:,:),skdiag(:),snode(:),snum(:)
 REAL(iwp),ALLOCATABLE::coord(:,:),der(:,:),deriv(:,:),disps(:),fun(:),   &
   gc(:),g_coord(:,:),jac(:,:),kay(:,:),kc(:,:),kv(:),kvh(:),loads(:),    &
   points(:,:),prop(:,:),value(:),weights(:),x_coords(:),y_coords(:),     &
   scoord(:,:),sder(:,:),sderiv(:,:),sdisps(:),sfun(:),                   &
   sgc(:),sg_coord(:,:),sjac(:,:),skay(:,:),skc(:,:),skv(:),skvh(:),      &
   sloads(:),spoints(:,:),sprop(:,:),svalue(:),sweights(:),sx_coords(:),  &
   sy_coords(:)
!----------------------input and initialisation--------------------------
 CALL getname(argv,nlen)
 OPEN(10,FILE=argv(1:nlen)//'.dat')
 OPEN(11,FILE=argv(1:nlen)//'.res')
 OPEN(13,FILE=argv(1:nlen)//'.con')
 READ(10,*)type_2d,dir,nxe,nye,np_types
 CALL mesh_size(element,nod,nels,nn,nxe,nye)
 neq=nn
 ALLOCATE(points(nip,ndim),g_coord(ndim,nn),coord(nod,ndim),              &
   jac(ndim,ndim),weights(nip),der(ndim,nod),deriv(ndim,nod),             &
   kc(nod,nod),num(nod),g_num(nod,nels),kay(ndim,ndim),etype(nels),       &
   x_coords(nxe+1),y_coords(nye+1),prop(ndim,np_types),gc(ndim),fun(nod), &
   kdiag(neq),loads(0:neq),disps(0:neq))
 READ(10,*)prop
 kbar=SQRT(prop(1,1)*prop(2,1))
 etype=1
 IF(np_types>1)READ(10,*)etype
!-------------------- assigning x and y coordinates ----------------------
 x=zero
 nx_nod=nxe+1
 DO i=1,nx_nod
   x_coords(i)=x
   x=x+1.0_iwp
 END DO
 y=zero
 ny_nod=nye+1
 DO z=1,ny_nod
   y_coords(z)=y
   y=y-1.0_iwp
 END DO
```

```
!-----------------------loop the elements to find global arrays sizes-----
 kdiag=0
 elements_1: DO iel=1,nels
   CALL geom_rect(element,iel,x_coords,y_coords,coord,num,dir)
   g_num(:,iel)=num
   g_coord(:,num)=TRANSPOSE(coord)
   CALL fkdiag(kdiag,num)
 END DO elements_1
 CALL mesh(g_coord,g_num,argv,nlen,12)
 DO i=2,neq
   kdiag(i)=kdiag(i)+kdiag(i-1)
 END DO
 ALLOCATE(kv(kdiag(neq)),kvh(kdiag(neq)))
 CALL sample(element,points,weights)
!-----------------------global conductivity matrix assembly---------------
 kv=zero
 gc=one
 elements_2: DO iel=1,nels
   kay=zero
   DO i=1,ndim
     kay(i,i)=prop(i,etype(iel))
   END DO
   num=g_num(:,iel)
   coord=TRANSPOSE(g_coord(:,num))
   kc=zero
   gauss_pts_1: DO i=1,nip
     CALL shape_der(der,points,i)
     CALL shape_fun(fun,points,i)
     jac=MATMUL(der,coord)
     det=determinant(jac)
     CALL invert(jac)
     deriv=MATMUL(jac,der)
     IF(type_2d=='axisymmetric')gc=MATMUL(fun,coord)
     kc=kc+MATMUL(MATMUL(TRANSPOSE(deriv),kay),deriv)*det*weights(i)*gc(1)
   END DO gauss_pts_1
   CALL fsparv(kv,kc,num,kdiag)
 END DO elements_2
 kvh=kv
!-----------------------specify boundary values--------------------------
 loads=zero
 READ(10,*)loaded_nodes
 IF(loaded_nodes/=0)READ(10,*)(k,loads(k),i=1,loaded_nodes)
    READ(10,*)s
    READ(10,*)h_one,h_two
    hdiff=ABS(h_one-h_two)
 fixed_freedoms=nx_nod+nye-s+1
 IF(fixed_freedoms/=0)THEN
    ALLOCATE(node(fixed_freedoms),value(fixed_freedoms))
    DO i=1,nx_nod
       node(i)=i
       value(i)=h_two
    END DO
    test=s*(nxe+1)+1
    DO i=nxe+2,fixed_freedoms
       node(i)=test
       value(i)=h_one
       test=test+nx_nod
   END DO
   kv(kdiag(node))=kv(kdiag(node))+penalty
   loads(node)=kv(kdiag(node))*value
 END IF
!-----------------------equation solution--------------------------------
 CALL sparin(kv,kdiag)
```

```
 CALL spabac(kv,loads,kdiag)
!------------------------retrieve nodal net flow rates--------------------
 i_e=loads(nxe+2)
 CALL linmul_sky(kvh,loads,disps,kdiag)
 disps(0)=zero
 flow_rate=SUM(disps,MASK=disps>zero)
 READ(10,*)nci
 f=1
 IF(nod==4) CALL contour1(loads,g_coord,g_num,nci,argv,nlen,13,s,nxe,f)
!------------------------global conductivity matrix assembly--------------
 kv=zero
 gc=one
 elements_4: DO iel=1,nels
   kay=zero
     kay(1,1)=one/prop(2,etype(iel))
     kay(2,2)=one/prop(1,etype(iel))
   num=g_num(:,iel)
   coord=TRANSPOSE(g_coord(:,num))
   kc=zero
   gauss_pts_2: DO i=1,nip
     CALL shape_der(der,points,i)
     CALL shape_fun(fun,points,i)
     jac=MATMUL(der,coord)
     det=determinant(jac)
     CALL invert(jac)
     deriv=MATMUL(jac,der)
     IF(type_2d=='axisymmetric')gc=MATMUL(fun,coord)
     kc=kc+MATMUL(MATMUL(TRANSPOSE(deriv),kay),deriv)*det*weights(i)*gc(1)
   END DO gauss_pts_2
  CALL fsparv(kv,kc,num,kdiag)
 END DO elements_4
 kvh=kv
!-----------------------specify boundary values--------------------------
 DEALLOCATE(node,value)
 loads=zero
 fixed_freedoms=ny_nod+s+nx_nod
! WRITE(11,'(I5)')fixed_freedoms
 IF(fixed_freedoms/=0)THEN
    ALLOCATE(node(fixed_freedoms),value(fixed_freedoms))
    test=nx_nod
    DO i=1,ny_nod
       node(i)=test
       value(i)=h_two
       test=test+nx_nod
    END DO
    zztop=1
    DO i=ny_nod+1,ny_nod+1+s
        node(i)=zztop
        value(i)=h_one
        zztop=zztop+nx_nod
    END DO
    yytop=(nx_nod)*nye+1
    DO i=ny_nod+1+s+1,fixed_freedoms
        node(i)=yytop
        value(i)=h_two
        yytop=yytop+1
    END DO
    kv(kdiag(node))=kv(kdiag(node))+penalty
    loads(node)=kv(kdiag(node))*value
 END IF
!-----------------------equation solution--------------------------------
 CALL sparin(kv,kdiag)
 CALL spabac(kv,loads,kdiag)
```

```
!-----------------------retrieve nodal net flow rates---------------------
 CALL linmul_sky(kvh,loads,disps,kdiag)
 disps(0)=zero
 nci=NINT(flow_rate*nci/kbar/hdiff)
 f=f+1
 IF(nod==4) CALL contour1(loads,g_coord,g_num,nci,argv,nlen,13,s,nxe,f)
 WRITE(13,'(a)')'showpage'
 WRITE(11,'(/A)')                                                        &
   "           L/t       s/T    Form Factor    i_e*s/h"
WRITE(11,'(5X,3F10.4,F13.4)')nxe*one/nye,s*one/nye,kbar*hdiff/flow_rate,i_e*s/h
diff
 CLOSE(13)
!
STOP
CONTAINS
SUBROUTINE contour1(loads,g_coord,g_num,ned,argv,nlen,ips,s,nxe,f)
!
! This subroutine produces a PostScript output file "*.con" displaying
! a contour map of loads over the finite element mesh.
!
 IMPLICIT NONE
 INTEGER,PARAMETER::iwp=SELECTED_REAL_KIND(15)
 REAL(iwp),INTENT(IN):: loads(0:),g_coord(:,:)
 INTEGER,INTENT(IN)::g_num(:,:),ned,ips,nlen,s,nxe,f
 CHARACTER(*),INTENT(IN)::argv
 REAL(iwp)::xmin,xmax,ymin,ymax,width,height,scale=72,sxy,xo,yo
 REAL(iwp)::pmin,pmax,ratio,x12,y12,x23,y23,x34,y34,x41,y41,x1,y1
 REAL(iwp)::pt5=0.5_iwp,elvn=11.0_iwp,ept5=8.5_iwp,eight=8.0_iwp,        &
   onept5=1.5_iwp,fpt5=5.5_iwp
 LOGICAL::s12,s23,s34,s41,draw
 INTEGER,ALLOCATABLE::corner(:,:)
 REAL(iwp),ALLOCATABLE::cont(:)
 INTEGER::i,j,k,l,nn,nels,nci,nod,ns,i1,i2,j1,j2
!
!                       compute size of mesh
!
 nn=UBOUND(g_coord,2)
 nod= UBOUND(g_num,1)
 nels=UBOUND(g_num,2)
 xmin=g_coord(1,1)
 xmax=g_coord(1,1)
 ymin=g_coord(2,1)
 ymax=g_coord(2,1)
 DO i=2,nn
   IF(g_coord(1,i)<xmin)xmin=g_coord(1,i)
   IF(g_coord(1,i)>xmax)xmax=g_coord(1,i)
   IF(g_coord(2,i)<ymin)ymin=g_coord(2,i)
   IF(g_coord(2,i)>ymax)ymax=g_coord(2,i)
 END DO
 width =xmax-xmin
 height=ymax-ymin
!
!                       allow 1.5" margin minimum on each side of figure
!
!                       portrait mode
!
 IF(height.GE.elvn/ept5*width)THEN
!
!                       height governs the scale
!
   sxy=scale*eight/height
   xo=scale*pt5*(ept5-eight*width/height)
   yo=scale*onept5
```

```
 ELSE
!
!                        width governs the scale
!
   sxy=scale*fpt5/width
   xo=scale*onept5
   yo=scale*pt5*(elvn-fpt5*height/width)
 END IF
!
!                        find range of potentials and contour values
!
 nci=ned+1
 pmin=MINVAL(loads(1:))
 pmax=MAXVAL(loads(1:))
 ALLOCATE(cont(nci))
 cont(1)=pmin
 DO i=2,nci
   cont(i)=cont(i-1)+(pmax-pmin)/ned
 END DO
!
!                        start PostScript output
!
 WRITE(ips,'(a)')'%!PS-Adobe-1.0'
 WRITE(ips,'(a)')'%%DocumentFonts: none'
 WRITE(ips,'(a)')'%%Pages: 1'
 WRITE(ips,'(a)')'%%EndComments'
 WRITE(ips,'(a)')'/m {moveto} def'
 WRITE(ips,'(a)')'/l {lineto} def'
 WRITE(ips,'(a)')'/s {stroke} def'
 WRITE(ips,'(a)')'%%EndProlog'
 WRITE(ips,'(a)')'%%Page: 0 1'
 WRITE(ips,'(a)')'gsave'
!
 WRITE(ips,'(2f9.2,a)') xo, yo, ' translate'
 WRITE(ips,'(f9.2,a)') 1.0, ' setlinewidth'
!
!                        draw the mesh outline
!
 IF(nod==3.OR.nod==6.OR.nod==10.OR.nod==15)ns=3
 IF(nod==4.OR.nod==8.OR.nod==9)ns=4
 ALLOCATE(corner(ns,2))
 IF(nod== 3)corner=RESHAPE((/1,2,3,2,3,1/),(/3,2/))
 IF(nod== 6)corner=RESHAPE((/1,3,5,3,5,1/),(/3,2/))
 IF(nod==10)corner=RESHAPE((/1,4,7,4,7,1/),(/3,2/))
 IF(nod==15)corner=RESHAPE((/1,5,9,5,9,1/),(/3,2/))
 IF(nod== 4)corner=RESHAPE((/1,2,3,4,2,3,4,1/),(/4,2/))
 IF(nod== 8)corner=RESHAPE((/1,3,5,7,3,5,7,1/),(/4,2/))
 IF(nod== 9)corner=RESHAPE((/1,3,5,7,3,5,7,1/),(/4,2/))
 DO i=1,nels
   DO j=1,ns
     draw=.TRUE.
     i1=g_num(corner(j,1),i)
     i2=g_num(corner(j,2),i)
     DO k=1,nels
       DO l=1,ns
         j1=g_num(corner(l,1),k)
         j2=g_num(corner(l,2),k)
         IF((i1==j2).AND.(i2==j1))THEN
           draw=.FALSE.
           EXIT
         END IF
       END DO
       IF(.NOT.draw)EXIT
```

```
      END DO
      IF(draw)THEN
        x1=sxy*(g_coord(1,i1)-xmin)
        y1=sxy*(g_coord(2,i1)-ymin)
        WRITE(ips,'(2f9.2,a)') x1, y1,' m'
        x1=sxy*(g_coord(1,i2)-xmin)
        y1=sxy*(g_coord(2,i2)-ymin)
        WRITE(ips,'(2f9.2,a)') x1, y1,' l'
        WRITE(ips,'(a)')' s'
      END IF
    END DO
  END DO
  WRITE(ips,'(f9.2,a)') 3.0, ' setlinewidth'
        x1=sxy*(g_coord(1,1)-xmin)
        y1=sxy*(g_coord(2,1)-ymin)
        WRITE(ips,'(2f9.2,a)') x1, y1,' m'
        i2=s*(nxe+1)+1
        x1=sxy*(g_coord(1,i2)-xmin)
        y1=sxy*(g_coord(2,i2)-ymin)
        WRITE(ips,'(2f9.2,a)') x1, y1,' l'
        WRITE(ips,'(a)')' s'
        x1=sxy*(g_coord(1,nxe+1)-xmin)
        y1=sxy*(g_coord(2,nxe+1)-ymin)
        WRITE(ips,'(2f9.2,a)') x1, y1,' m'
        x1=sxy*(g_coord(1,nn)-xmin)
        y1=sxy*(g_coord(2,nn)-ymin)
        WRITE(ips,'(2f9.2,a)') x1, y1,' l'
        WRITE(ips,'(a)')' s'
        x1=sxy*(g_coord(1,nn-nxe)-xmin)
        y1=sxy*(g_coord(2,nn-nxe)-ymin)
        WRITE(ips,'(2f9.2,a)') x1, y1,' m'
        x1=sxy*(g_coord(1,nn)-xmin)
        y1=sxy*(g_coord(2,nn)-ymin)
        WRITE(ips,'(2f9.2,a)') x1, y1,' l'
        WRITE(ips,'(a)')' s'
  WRITE(ips,'(f9.2,a)') 1.0, ' setlinewidth'
!
!                        check intersection of contours with each element
!
  WRITE(ips,'(f9.2,a)') 0.5, ' setlinewidth'
  IF(f==1)WRITE(ips,'(f9.2,a)') 1.0, ' 0.7 0.3 0 setrgbcolor'
  IF(f==2)WRITE(ips,'(f9.2,a)') 1.0, ' 0 0 1 setrgbcolor'
  DO i=2,nci-1
    DO j=1,nels
      s12=.FALSE.
      s23=.FALSE.
      s34=.FALSE.
      s41=.FALSE.
      IF((loads(g_num(1,j))<=cont(i).AND.loads(g_num(2,j))>cont(i)).OR.    &
        (loads(g_num(2,j))<=cont(i).AND.loads(g_num(1,j))>cont(i)))        &
        s12=.TRUE.
      IF((loads(g_num(2,j))<=cont(i).AND.loads(g_num(3,j))>cont(i)).OR.    &
        (loads(g_num(3,j))<=cont(i).AND.loads(g_num(2,j))>cont(i)))        &
        s23=.TRUE.
      IF((loads(g_num(3,j))<=cont(i).AND.loads(g_num(4,j))>cont(i)).OR.    &
        (loads(g_num(4,j))<=cont(i).AND.loads(g_num(3,j))>cont(i)))        &
        s34=.TRUE.
      IF((loads(g_num(4,j))<=cont(i).AND.loads(g_num(1,j))>cont(i)).OR.    &
        (loads(g_num(1,j))<=cont(i).AND.loads(g_num(4,j))>cont(i)))        &
        s41=.TRUE.
      IF(s12)THEN
        ratio=(cont(i)-loads(g_num(1,j)))/                                 &
        (loads(g_num(2,j))-loads(g_num(1,j)))
```

```
        x12=sxy*(g_coord(1,g_num(1,j))+                                    &
          ratio*(g_coord(1,g_num(2,j))-g_coord(1,g_num(1,j)))-xmin)
        y12=sxy*(g_coord(2,g_num(1,j))+                                    &
          ratio*(g_coord(2,g_num(2,j))-g_coord(2,g_num(1,j)))-ymin)
      END IF
      IF(s23)THEN
        ratio=(cont(i)-loads(g_num(2,j)))/                                 &
          (loads(g_num(3,j))-loads(g_num(2,j)))
        x23=sxy*(g_coord(1,g_num(2,j))+                                    &
          ratio*(g_coord(1,g_num(3,j))-g_coord(1,g_num(2,j)))-xmin)
        y23=sxy*(g_coord(2,g_num(2,j))+                                    &
          ratio*(g_coord(2,g_num(3,j))-g_coord(2,g_num(2,j)))-ymin)
      END IF
      IF(s34)THEN
        ratio=(cont(i)-loads(g_num(3,j)))/                                 &
          (loads(g_num(4,j))-loads(g_num(3,j)))
        x34=sxy*(g_coord(1,g_num(3,j))+                                    &
          ratio*(g_coord(1,g_num(4,j))-g_coord(1,g_num(3,j)))-xmin)
        y34=sxy*(g_coord(2,g_num(3,j))+                                    &
          ratio*(g_coord(2,g_num(4,j))-g_coord(2,g_num(3,j)))-ymin)
      END IF
      IF(s41)THEN
        ratio=(cont(i)-loads(g_num(4,j)))/                                 &
          (loads(g_num(1,j))-loads(g_num(4,j)))
        x41=sxy*(g_coord(1,g_num(4,j))+                                    &
          ratio*(g_coord(1,g_num(1,j))-g_coord(1,g_num(4,j)))-xmin)
        y41=sxy*(g_coord(2,g_num(4,j))+                                    &
          ratio*(g_coord(2,g_num(1,j))-g_coord(2,g_num(4,j)))-ymin)
       END IF
!
!                       draw contours
!
      IF(s12)THEN
        WRITE(ips,'(2f9.2,a)')x12,y12,' m'
        IF(s23)WRITE(ips,'(2f9.2,a)')x23,y23,' l s'
        IF(s34)WRITE(ips,'(2f9.2,a)')x34,y34,' l s'
        IF(s41)WRITE(ips,'(2f9.2,a)')x41,y41,' l s'
      END IF
      IF(s23)THEN
        WRITE(ips,'(2f9.2,a)')x23,y23,' m'
        IF(s34)WRITE(ips,'(2f9.2,a)')x34,y34,' l s'
        IF(s41)WRITE(ips,'(2f9.2,a)')x41,y41,' l s'
      END IF
      IF(s34.AND.s41)THEN
        WRITE(ips,'(2f9.2,a)')x34,y34,' m'
        WRITE(ips,'(2f9.2,a)')x41,y41,' l s'
      END IF
  END DO
 END DO
!                       close output file
 WRITE(ips,'(a)')'grestore'
! CLOSE(ips)
!
RETURN
END SUBROUTINE contour1
END PROGRAM type_c
```

Program Listing for type_a.f95

```
!     Last change:  JEH   06 Apr 2010    6:53 am
PROGRAM type_a
!------------------------------------------------------------------------
! Program 7.2 Plane or axisymmetric analysis of steady seepage using
!             4-node rectangular quadrilaterals. Mesh numbered
!             in x(r)- or y(z)- direction.
!------------------------------------------------------------------------
 USE main
 USE geom
 IMPLICIT NONE
 INTEGER,PARAMETER::iwp=SELECTED_REAL_KIND(15)
 INTEGER::fixed_freedoms,i,iel,k,loaded_nodes,nci,ndim=2,nels,neq,nip=4,   &
   nod=4,nn,np_types,nxe,nye,nlen,z,zztop,s,test,yytop,nx_nod,ny_nod,      &
   fix_free,f,b,T,bode
 REAL(iwp)::x,y,det,one=1.0_iwp,penalty=1.0e20_iwp,zero=0.0_iwp,           &
   h_one,h_two,flow_rate,i_e,kbar,hdiff,l_coeff,elem_size
 CHARACTER(LEN=15)::dir,element='quadrilateral',type_2d,argv
!----------------------dynamic arrays-----------------------------------
 INTEGER,ALLOCATABLE::etype(:),g_num(:,:),kdiag(:),node(:),num(:),         &
   setype(:),sg_num(:,:),skdiag(:),snode(:),snum(:)
 REAL(iwp),ALLOCATABLE::coord(:,:),der(:,:),deriv(:,:),disps(:),fun(:),    &
   gc(:),g_coord(:,:),jac(:,:),kay(:,:),kc(:,:),kv(:),kvh(:),loads(:),     &
   points(:,:),prop(:,:),value(:),weights(:),x_coords(:),y_coords(:),      &
   scoord(:,:),sder(:,:),sderiv(:,:),sdisps(:),sfun(:),                    &
   sgc(:),sg_coord(:,:),sjac(:,:),skay(:,:),skc(:,:),skv(:),skvh(:),       &
   sloads(:),spoints(:,:),sprop(:,:),svalue(:),sweights(:),sx_coords(:),   &
   sy_coords(:)
!----------------------input and initialisation-------------------------
 CALL getname(argv,nlen)
 OPEN(10,FILE=argv(1:nlen)//'.dat')
 OPEN(11,FILE=argv(1:nlen)//'.res')
 OPEN(13,FILE=argv(1:nlen)//'.con')
 READ(10,*)type_2d,dir,b,l_coeff,T,np_types
 nxe=b*l_coeff
 nye=T
 CALL mesh_size(element,nod,nels,nn,nxe,nye)
 neq=nn
 ALLOCATE(points(nip,ndim),g_coord(ndim,nn),coord(nod,ndim),               &
   jac(ndim,ndim),weights(nip),der(ndim,nod),deriv(ndim,nod),              &
   kc(nod,nod),num(nod),g_num(nod,nels),kay(ndim,ndim),etype(nels),        &
   x_coords(nxe+1),y_coords(nye+1),prop(ndim,np_types),gc(ndim),fun(nod),  &
   kdiag(neq),loads(0:neq),disps(0:neq))
 READ(10,*)prop
 kbar=SQRT(prop(1,1)*prop(2,1))
 etype=1
 IF(np_types>1)READ(10,*)etype
!-------------------- assigning x and y coordinates ----------------------
 x=zero
 elem_size=1.0_iwp
 nx_nod=nxe+1
 DO i=1,nx_nod
   x_coords(i)=x
   x=x+elem_size
 END DO
 y=zero
 ny_nod=nye+1
 DO z=1,ny_nod
   y_coords(z)=y
   y=y-elem_size
 END DO
```

```
!----------------------loop the elements to find global arrays sizes-----
 kdiag=0
 elements_1: DO iel=1,nels
   CALL geom_rect(element,iel,x_coords,y_coords,coord,num,dir)
   g_num(:,iel)=num
   g_coord(:,num)=TRANSPOSE(coord)
   CALL fkdiag(kdiag,num)
 END DO elements_1
!CALL mesh(g_coord,g_num,argv,nlen,12)
 DO i=2,neq
   kdiag(i)=kdiag(i)+kdiag(i-1)
 END DO
 ALLOCATE(kv(kdiag(neq)),kvh(kdiag(neq)))
! WRITE(11,'(2(A,I8))')                                                  &
!   "There are",neq," equations and the skyline storage is",kdiag(neq)
 CALL sample(element,points,weights)
!----------------------global conductivity matrix assembly---------------
 kv=zero
 gc=one
 elements_2: DO iel=1,nels
   kay=zero
   DO i=1,ndim
     kay(i,i)=prop(i,etype(iel))
   END DO
   num=g_num(:,iel)
   coord=TRANSPOSE(g_coord(:,num))
   kc=zero
   gauss_pts_1: DO i=1,nip
     CALL shape_der(der,points,i)
     CALL shape_fun(fun,points,i)
     jac=MATMUL(der,coord)
     det=determinant(jac)
     CALL invert(jac)
     deriv=MATMUL(jac,der)
     IF(type_2d=='axisymmetric')gc=MATMUL(fun,coord)
     kc=kc+MATMUL(MATMUL(TRANSPOSE(deriv),kay),deriv)*det*weights(i)*gc(1)
   END DO gauss_pts_1
   CALL fsparv(kv,kc,num,kdiag)
 END DO elements_2
 kvh=kv
!----------------------specify boundary values---------------------------
 loads=zero
 READ(10,*)loaded_nodes
 IF(loaded_nodes/=0)READ(10,*)(k,loads(k),i=1,loaded_nodes)
    READ(10,*)s
    READ(10,*)h_one,h_two
    hdiff=ABS(h_one-h_two)
 fixed_freedoms=nx_nod-(b+1)+T-s
! WRITE(11,'(I5)')fixed_freedoms
 IF(fixed_freedoms/=0)THEN
    ALLOCATE(node(fixed_freedoms),value(fixed_freedoms))
    DO i=1,nx_nod-(b+1)
       node(i)=i+b
       value(i)=h_two
    END DO
    test=(s+1)*nx_nod+1
    DO i=nx_nod-b,fixed_freedoms
       node(i)=test
       value(i)=h_one
       test=test+nx_nod
    END DO
    kv(kdiag(node))=kv(kdiag(node))+penalty
    loads(node)=kv(kdiag(node))*value
```

```
 END IF
!-----------------------equation solution--------------------------------
 CALL sparin(kv,kdiag)
 CALL spabac(kv,loads,kdiag)
!-----------------------retrieve nodal net flow rates--------------------
 i_e=loads(nxe+1+b+1)/elem_size
 CALL linmul_sky(kvh,loads,disps,kdiag)
! WRITE(11,'(/A)')"  Node Total Head  Flow rate"
! DO k=1,nn
!   WRITE(11,'(I5,2E12.4)')k,loads(k),disps(k)
! END DO
 disps(0)=zero
! WRITE(11,'(/A)')"       Inflow      Outflow"
! WRITE(11,'(5X,2E12.4)')                                                &
!   SUM(disps,MASK=disps>zero),SUM(disps,MASK=disps<zero)
 flow_rate=SUM(disps,MASK=disps>zero)
 READ(10,*)nci
 f=1
 IF(nod==4) CALL contour1(loads,g_coord,g_num,nci,argv,nlen,13,s,nxe,f)
!-----------------------global conductivity matrix assembly--------------
 kv=zero
 gc=one
 elements_4: DO iel=1,nels
   kay=zero
     kay(1,1)=one/prop(2,etype(iel))
     kay(2,2)=one/prop(1,etype(iel))
   num=g_num(:,iel)
   coord=TRANSPOSE(g_coord(:,num))
   kc=zero
   gauss_pts_2: DO i=1,nip
     CALL shape_der(der,points,i)
     CALL shape_fun(fun,points,i)
     jac=MATMUL(der,coord)
     det=determinant(jac)
     CALL invert(jac)
     deriv=MATMUL(jac,der)
     IF(type_2d=='axisymmetric')gc=MATMUL(fun,coord)
     kc=kc+MATMUL(MATMUL(TRANSPOSE(deriv),kay),deriv)*det*weights(i)*gc(1)
   END DO gauss_pts_2
  CALL fsparv(kv,kc,num,kdiag)
 END DO elements_4
 kvh=kv
!-----------------------specify boundary values--------------------------
 DEALLOCATE(node,value)
 loads=zero
 fixed_freedoms=b+1+s+nx_nod+ny_nod-1
! WRITE(11,'(I5)')fixed_freedoms
 IF(fixed_freedoms/=0)THEN
    ALLOCATE(node(fixed_freedoms),value(fixed_freedoms))
    DO i=1,b+1
       node(i)=i
       value(i)=h_one
    END DO
    zztop=nx_nod+1
    DO i=b+2,b+2+(s-1)
        node(i)=zztop
        value(i)=h_one
        zztop=zztop+nx_nod
    END DO
    yytop=(nx_nod)*nye+1
    DO i=b+2+s,b+1+s+nx_nod
        node(i)=yytop
        value(i)=h_two
```

```
        yytop=yytop+1
    END DO
    bode=nx_nod
    DO i=b+2+s+nx_nod,fixed_freedoms
        node(i)=bode
        value(i)=h_two
        bode=bode+nx_nod
    END DO
    kv(kdiag(node))=kv(kdiag(node))+penalty
    loads(node)=kv(kdiag(node))*value
 END IF
!
!-----------------------   call mesh1    ---------------------------------
!
CALL mesh1(g_coord,g_num,argv,nlen,12,s,b,nxe)
!
!-----------------------equation solution---------------------------------
 CALL sparin(kv,kdiag)
 CALL spabac(kv,loads,kdiag)
!-----------------------retrieve nodal net flow rates--------------------
 CALL linmul_sky(kvh,loads,disps,kdiag)
! WRITE(11,'(/A)')"  Nude Total Head  Flow rate"
! DO k=1,nn
!   WRITE(11,'(I5,2E12.4)')k,loads(k),disps(k)
! END DO
 disps(0)=zero
! WRITE(11,'(/A)')"       Inflow      Outflow"
! WRITE(11,'(5X,2E12.4)')                                              &
!   SUM(disps,MASK=disps>zero),SUM(disps,MASK=disps<zero)
 nci=NINT(flow_rate*nci/kbar/hdiff)
 f=f+1
 IF(nod==4) CALL contour1(loads,g_coord,g_num,nci,argv,nlen,13,s,nxe,f)
 WRITE(13,'(a)')'showpage'

 WRITE(11,'(/A)')                                                    &
   "         bR/T       s/T    Form Factor      i_e       h        b    s    T
nxe"
 WRITE(11,'(5X,3F10.4,F13.4,F10.4,I6,3I5)')                          &
         b*kbar/T,s*one/nye,kbar*hdiff/flow_rate,i_e,hdiff,b,s,T,nxe
 CLOSE(13)
!
STOP
!
!
!
CONTAINS
SUBROUTINE mesh1(g_coord,g_num,argv,nlen,ips,s,b,nxe)
!
! This subroutine produces a PostScript output file "*.msh" displaying
! the undeformed finite element mesh.
!
 IMPLICIT NONE
 INTEGER,PARAMETER::iwp=SELECTED_REAL_KIND(15)
 REAL(iwp),INTENT(IN)::g_coord(:,:)
 INTEGER,INTENT(IN)::g_num(:,:),ips,nlen,s,b,nxe
 CHARACTER(*),INTENT(IN)::argv
 REAL(iwp)::xmin,xmax,ymin,ymax,width,height,scale=72,sxy,xo,yo,x,y,  &
   pt5=0.5_iwp,opt5=1.5_iwp,fpt5=5.5_iwp,d8=8.0_iwp,ept5=8.5_iwp,     &
   d11=11.0_iwp,x1,y1
 INTEGER::i,ii,j,jj,nn,nod,nel,i2
 OPEN(ips,FILE=argv(1:nlen)//'.msh')
!
!                       compute size of mesh
```

```
!
 nn=UBOUND(g_coord,2)
 xmin=g_coord(1,1)
 xmax=g_coord(1,1)
 ymin=g_coord(2,1)
 ymax=g_coord(2,1)
 DO i=2,nn
   IF(g_coord(1,i)<xmin)xmin=g_coord(1,i)
   IF(g_coord(1,i)>xmax)xmax=g_coord(1,i)
   IF(g_coord(2,i)<ymin)ymin=g_coord(2,i)
   IF(g_coord(2,i)>ymax)ymax=g_coord(2,i)
 END DO
 width =xmax-xmin
 height=ymax-ymin
!
!                        allow 1.5" margin minimum on each side of figure
!
 IF(height.GE.d11/ept5*width)THEN
!
!                        height governs the scale
!
   sxy=scale*d8/height
   xo=scale*pt5*(ept5-d8*width/height)
   yo=scale*opt5
 ELSE
!
!                        width governs the scale
!
   sxy=scale*fpt5/width
   xo=scale*opt5
   yo=scale*pt5*(d11-fpt5*height/width)
 END IF
!
!                        start PostScript output
!
 WRITE(ips,'(a)')'%!PS-Adobe-1.0'
 WRITE(ips,'(a)')'%%DocumentFonts: none'
 WRITE(ips,'(a)')'%%Pages: 1'
 WRITE(ips,'(a)')'%%EndComments'
 WRITE(ips,'(a)')'/m {moveto} def'
 WRITE(ips,'(a)')'/l {lineto} def'
 WRITE(ips,'(a)')'/s {stroke} def'
 WRITE(ips,'(a)')'/c {closepath} def'
 WRITE(ips,'(a)')'%%EndProlog'
 WRITE(ips,'(a)')'%%Page: 0 1'
 WRITE(ips,'(a)')'gsave'
 WRITE(ips,'(2f9.2,a)') xo, yo, ' translate'
 WRITE(ips,'(f9.2,a)') 0.5, ' setlinewidth'
!
!                        draw the mesh
!
 nod=UBOUND(g_num,1)
 nel=UBOUND(g_num,2)
 IF(nod==5)nod=4
 IF(nod==9)nod=8
 IF(nod==10)nod=9
 IF(nod==15)nod=12
 DO i=1,nel
   ii=g_num(1,i)
   IF(ii==0)CYCLE
   x=sxy*(g_coord(1,ii)-xmin)
   y=sxy*(g_coord(2,ii)-ymin)
   WRITE(ips,'(2f9.2,a)')x,y,' m'
```

```
    DO j=2,nod
      jj=g_num(j,i)
      x=sxy*(g_coord(1,jj)-xmin)
      y=sxy*(g_coord(2,jj)-ymin)
      WRITE(ips,'(2f9.2,a)') x, y,' l'
    END DO
    WRITE(ips,'(a)')'c s'
  END DO
!
!                          draw the heavy lines
!
  WRITE(ips,'(f9.2,a)') 3.0, ' setlinewidth'
        x1=sxy*(g_coord(1,1)-xmin)
        y1=sxy*(g_coord(2,1)-ymin)
        WRITE(ips,'(2f9.2,a)') x1, y1,' m'
        i2=s*(nxe+1)+1
        x1=sxy*(g_coord(1,i2)-xmin)
        y1=sxy*(g_coord(2,i2)-ymin)
        WRITE(ips,'(2f9.2,a)') x1, y1,' l'
        WRITE(ips,'(a)')' s'

        x1=sxy*(g_coord(1,nxe+1)-xmin)
        y1=sxy*(g_coord(2,nxe+1)-ymin)
        WRITE(ips,'(2f9.2,a)') x1, y1,' m'
        x1=sxy*(g_coord(1,nn)-xmin)
        y1=sxy*(g_coord(2,nn)-ymin)
        WRITE(ips,'(2f9.2,a)') x1, y1,' l'
        WRITE(ips,'(a)')' s'

        x1=sxy*(g_coord(1,nn-nxe)-xmin)
        y1=sxy*(g_coord(2,nn-nxe)-ymin)
        WRITE(ips,'(2f9.2,a)') x1, y1,' m'
        x1=sxy*(g_coord(1,nn)-xmin)
        y1=sxy*(g_coord(2,nn)-ymin)
        WRITE(ips,'(2f9.2,a)') x1, y1,' l'
        WRITE(ips,'(a)')' s'

        x1=sxy*(g_coord(1,1)-xmin)
        y1=sxy*(g_coord(2,1)-ymin)
        WRITE(ips,'(2f9.2,a)') x1, y1,' m'
        x1=sxy*(g_coord(1,b+1)-xmin)
        y1=sxy*(g_coord(2,b+1)-ymin)
        WRITE(ips,'(2f9.2,a)') x1, y1,' l'
        WRITE(ips,'(a)')' s'

!
!                         close output file
!
  WRITE(ips,'(a)')'grestore'
  WRITE(ips,'(a)')'showpage'
  CLOSE(ips)
!
RETURN
END SUBROUTINE mesh1
!
!
!
!
SUBROUTINE contour1(loads,g_coord,g_num,ned,argv,nlen,ips,s,nxe,f)
!
! This subroutine produces a PostScript output file "*.con" displaying
! a contour map of loads over the finite element mesh.
!
```

```
 IMPLICIT NONE
 INTEGER,PARAMETER::iwp=SELECTED_REAL_KIND(15)
 REAL(iwp),INTENT(IN):: loads(0:),g_coord(:,:)
 INTEGER,INTENT(IN)::g_num(:,:),ned,ips,nlen,s,nxe,f
 CHARACTER(*),INTENT(IN)::argv
 REAL(iwp)::xmin,xmax,ymin,ymax,width,height,scale=72,sxy,xo,yo
 REAL(iwp)::pmin,pmax,ratio,x12,y12,x23,y23,x34,y34,x41,y41,x1,y1
 REAL(iwp)::pt5=0.5_iwp,elvn=11.0_iwp,ept5=8.5_iwp,eight=8.0_iwp,          &
   onept5=1.5_iwp,fpt5=5.5_iwp
 LOGICAL::s12,s23,s34,s41,draw
 INTEGER,ALLOCATABLE::corner(:,:)
 REAL(iwp),ALLOCATABLE::cont(:)
 INTEGER::i,j,k,l,nn,nels,nci,nod,ns,i1,i2,j1,j2
!
! OPEN(ips,FILE=argv(1:nlen)//'.con')
!
!                        compute size of mesh
!
 nn=UBOUND(g_coord,2)
 nod= UBOUND(g_num,1)
 nels=UBOUND(g_num,2)
 xmin=g_coord(1,1)
 xmax=g_coord(1,1)
 ymin=g_coord(2,1)
 ymax=g_coord(2,1)
 DO i=2,nn
   IF(g_coord(1,i)<xmin)xmin=g_coord(1,i)
   IF(g_coord(1,i)>xmax)xmax=g_coord(1,i)
   IF(g_coord(2,i)<ymin)ymin=g_coord(2,i)
   IF(g_coord(2,i)>ymax)ymax=g_coord(2,i)
 END DO
 width =xmax-xmin
 height=ymax-ymin
!
!                        allow 1.5" margin minimum on each side of figure
!
!                        portrait mode
!
 IF(height.GE.elvn/ept5*width)THEN
!
!                        height governs the scale
!
   sxy=scale*eight/height
   xo=scale*pt5*(ept5-eight*width/height)
   yo=scale*onept5
 ELSE
!
!                        width governs the scale
!
   sxy=scale*fpt5/width
   xo=scale*onept5
   yo=scale*pt5*(elvn-fpt5*height/width)
 END IF
!
!                        find range of potentials and contour values
!
 nci=ned+1
 pmin=MINVAL(loads(1:))
 pmax=MAXVAL(loads(1:))
 ALLOCATE(cont(nci))
 cont(1)=pmin
 DO i=2,nci
   cont(i)=cont(i-1)+(pmax-pmin)/ned
```

```
 END DO
!
!                        start PostScript output
!
 WRITE(ips,'(a)')'%!PS-Adobe-1.0'
 WRITE(ips,'(a)')'%%DocumentFonts: none'
 WRITE(ips,'(a)')'%%Pages: 1'
 WRITE(ips,'(a)')'%%EndComments'
 WRITE(ips,'(a)')'/m {moveto} def'
 WRITE(ips,'(a)')'/l {lineto} def'
 WRITE(ips,'(a)')'/s {stroke} def'
 WRITE(ips,'(a)')'%%EndProlog'
 WRITE(ips,'(a)')'%%Page: 0 1'
 WRITE(ips,'(a)')'gsave'
!
 WRITE(ips,'(2f9.2,a)') xo, yo, ' translate'
 WRITE(ips,'(f9.2,a)') 1.0, ' setlinewidth'

!
!                        draw the mesh outline
!
 IF(nod==3.OR.nod==6.OR.nod==10.OR.nod==15)ns=3
 IF(nod==4.OR.nod==8.OR.nod==9)ns=4
 ALLOCATE(corner(ns,2))
 IF(nod== 3)corner=RESHAPE((/1,2,3,2,3,1/),(/3,2/))
 IF(nod== 6)corner=RESHAPE((/1,3,5,3,5,1/),(/3,2/))
 IF(nod==10)corner=RESHAPE((/1,4,7,4,7,1/),(/3,2/))
 IF(nod==15)corner=RESHAPE((/1,5,9,5,9,1/),(/3,2/))
 IF(nod== 4)corner=RESHAPE((/1,2,3,4,2,3,4,1/),(/4,2/))
 IF(nod== 8)corner=RESHAPE((/1,3,5,7,3,5,7,1/),(/4,2/))
 IF(nod== 9)corner=RESHAPE((/1,3,5,7,3,5,7,1/),(/4,2/))
 DO i=1,nels
   DO j=1,ns
     draw=.TRUE.
     i1=g_num(corner(j,1),i)
     i2=g_num(corner(j,2),i)
     DO k=1,nels
       DO l=1,ns
         j1=g_num(corner(l,1),k)
         j2=g_num(corner(l,2),k)
         IF((i1==j2).AND.(i2==j1))THEN
           draw=.FALSE.
           EXIT
         END IF
       END DO
       IF(.NOT.draw)EXIT
     END DO
     IF(draw)THEN
       x1=sxy*(g_coord(1,i1)-xmin)
       y1=sxy*(g_coord(2,i1)-ymin)
       WRITE(ips,'(2f9.2,a)') x1, y1,' m'
       x1=sxy*(g_coord(1,i2)-xmin)
       y1=sxy*(g_coord(2,i2)-ymin)
       WRITE(ips,'(2f9.2,a)') x1, y1,' l'
       WRITE(ips,'(a)')' s'
     END IF
   END DO
 END DO

 WRITE(ips,'(f9.2,a)') 3.0, ' setlinewidth'
       x1=sxy*(g_coord(1,1)-xmin)
       y1=sxy*(g_coord(2,1)-ymin)
       WRITE(ips,'(2f9.2,a)') x1, y1,' m'
```

```
        i2=s*(nxe+1)+1
        x1=sxy*(g_coord(1,i2)-xmin)
        y1=sxy*(g_coord(2,i2)-ymin)
        WRITE(ips,'(2f9.2,a)') x1, y1,' l'
        WRITE(ips,'(a)')' s'

        x1=sxy*(g_coord(1,nxe+1)-xmin)
        y1=sxy*(g_coord(2,nxe+1)-ymin)
        WRITE(ips,'(2f9.2,a)') x1, y1,' m'
        x1=sxy*(g_coord(1,nn)-xmin)
        y1=sxy*(g_coord(2,nn)-ymin)
        WRITE(ips,'(2f9.2,a)') x1, y1,' l'
        WRITE(ips,'(a)')' s'

        x1=sxy*(g_coord(1,nn-nxe)-xmin)
        y1=sxy*(g_coord(2,nn-nxe)-ymin)
        WRITE(ips,'(2f9.2,a)') x1, y1,' m'
        x1=sxy*(g_coord(1,nn)-xmin)
        y1=sxy*(g_coord(2,nn)-ymin)
        WRITE(ips,'(2f9.2,a)') x1, y1,' l'
        WRITE(ips,'(a)')' s'

        x1=sxy*(g_coord(1,1)-xmin)
        y1=sxy*(g_coord(2,1)-ymin)
        WRITE(ips,'(2f9.2,a)') x1, y1,' m'
        x1=sxy*(g_coord(1,b+1)-xmin)
        y1=sxy*(g_coord(2,b+1)-ymin)
        WRITE(ips,'(2f9.2,a)') x1, y1,' l'
        WRITE(ips,'(a)')' s'

 WRITE(ips,'(f9.2,a)') 1.0, ' setlinewidth'
!
!                       check intersection of contours with each element
!
 WRITE(ips,'(f9.2,a)') 0.5, ' setlinewidth'
 IF(f==1)WRITE(ips,'(f9.2,a)') 1.0, ' 0.7 0.3 0 setrgbcolor'
 IF(f==2)WRITE(ips,'(f9.2,a)') 1.0, ' 0 0 1 setrgbcolor'
 DO i=2,nci-1
   DO j=1,nels
     s12=.FALSE.
     s23=.FALSE.
     s34=.FALSE.
     s41=.FALSE.
     IF((loads(g_num(1,j))<=cont(i).AND.loads(g_num(2,j))>cont(i)).OR.     &
       (loads(g_num(2,j))<=cont(i).AND.loads(g_num(1,j))>cont(i)))         &
       s12=.TRUE.
     IF((loads(g_num(2,j))<=cont(i).AND.loads(g_num(3,j))>cont(i)).OR.     &
       (loads(g_num(3,j))<=cont(i).AND.loads(g_num(2,j))>cont(i)))         &
       s23=.TRUE.
     IF((loads(g_num(3,j))<=cont(i).AND.loads(g_num(4,j))>cont(i)).OR.     &
       (loads(g_num(4,j))<=cont(i).AND.loads(g_num(3,j))>cont(i)))         &
       s34=.TRUE.
     IF((loads(g_num(4,j))<=cont(i).AND.loads(g_num(1,j))>cont(i)).OR.     &
       (loads(g_num(1,j))<=cont(i).AND.loads(g_num(4,j))>cont(i)))         &
       s41=.TRUE.
     IF(s12)THEN
       ratio=(cont(i)-loads(g_num(1,j)))/                                  &
       (loads(g_num(2,j))-loads(g_num(1,j)))
       x12=sxy*(g_coord(1,g_num(1,j))+                                     &
         ratio*(g_coord(1,g_num(2,j))-g_coord(1,g_num(1,j)))-xmin)
       y12=sxy*(g_coord(2,g_num(1,j))+                                     &
         ratio*(g_coord(2,g_num(2,j))-g_coord(2,g_num(1,j)))-ymin)
```

```
      END IF
      IF(s23)THEN
        ratio=(cont(i)-loads(g_num(2,j)))/                                    &
          (loads(g_num(3,j))-loads(g_num(2,j)))
        x23=sxy*(g_coord(1,g_num(2,j))+                                       &
          ratio*(g_coord(1,g_num(3,j))-g_coord(1,g_num(2,j)))-xmin)
        y23=sxy*(g_coord(2,g_num(2,j))+                                       &
          ratio*(g_coord(2,g_num(3,j))-g_coord(2,g_num(2,j)))-ymin)
      END IF
      IF(s34)THEN
        ratio=(cont(i)-loads(g_num(3,j)))/                                    &
          (loads(g_num(4,j))-loads(g_num(3,j)))
        x34=sxy*(g_coord(1,g_num(3,j))+                                       &
          ratio*(g_coord(1,g_num(4,j))-g_coord(1,g_num(3,j)))-xmin)
        y34=sxy*(g_coord(2,g_num(3,j))+                                       &
          ratio*(g_coord(2,g_num(4,j))-g_coord(2,g_num(3,j)))-ymin)
      END IF
      IF(s41)THEN
        ratio=(cont(i)-loads(g_num(4,j)))/                                    &
          (loads(g_num(1,j))-loads(g_num(4,j)))
        x41=sxy*(g_coord(1,g_num(4,j))+                                       &
          ratio*(g_coord(1,g_num(1,j))-g_coord(1,g_num(4,j)))-xmin)
        y41=sxy*(g_coord(2,g_num(4,j))+                                       &
          ratio*(g_coord(2,g_num(1,j))-g_coord(2,g_num(4,j)))-ymin)
       END IF
!
!                       draw contours
!
      IF(s12)THEN
        WRITE(ips,'(2f9.2,a)')x12,y12,' m'
        IF(s23)WRITE(ips,'(2f9.2,a)')x23,y23,' l s'
        IF(s34)WRITE(ips,'(2f9.2,a)')x34,y34,' l s'
        IF(s41)WRITE(ips,'(2f9.2,a)')x41,y41,' l s'
      END IF
      IF(s23)THEN
        WRITE(ips,'(2f9.2,a)')x23,y23,' m'
        IF(s34)WRITE(ips,'(2f9.2,a)')x34,y34,' l s'
        IF(s41)WRITE(ips,'(2f9.2,a)')x41,y41,' l s'
      END IF
      IF(s34.AND.s41)THEN
        WRITE(ips,'(2f9.2,a)')x34,y34,' m'
        WRITE(ips,'(2f9.2,a)')x41,y41,' l s'
      END IF
    END DO
   END DO
!                       close output file
   WRITE(ips,'(a)')'grestore'
! CLOSE(ips)
!
RETURN
END SUBROUTINE contour1

END PROGRAM type_a
```