DEVELOPMENT AND IMPLEMENTATION OF A FINITE ELEMENT PROGRAM FOR STATIC ELECTROMAGNETIC FIELD PROBLEMS

G. Gopal

A thesis submitted to the Faculty of Engineering at the University of Cape Town in fulfilment of the requirements for the degree Masters of Science in Engineering.

DECLARATION

I declare that this thesis is my own unaided work. It is being submitted for the degree of Masters of Science in Engineering at the University of Cape Town.

It has not previously been submitted in this or any other form for any degree or examination at any other university.

signature removed

27-04-90

G. GOPAL

DATE

ABSTRACT

A program is presented for the solution of static electromagnetic fields in bounded and unbounded 2-dimensional domains. The program comprises of a mesh generator which descretises the domain into triangular finite elements or if applicable into special elements, called infinite elements which model the bounded and unbounded domains respectively. The potential function is solved for, using a special solution technique which enhances the speed of the program. The program outputs data in the form of potential or flux component distributions along lines of interest.

ACKNOWLEDGEMENTS

I would like to express my thanks to the folowing people in particular :

Dr. R.E. Neubauer for supervising the research and for his much valued advice and helpful discussions.

The User Support Service of the Computer Science Department for their assistance in using the computer for the implementation of the program.

The CSIR for the financial support that made this research possible.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

CHAPTER 1

INTRODUCTION

The finite element method is a relatively new technique for the solution of differential equations in the field of Electrical Engineering, considering that the finite difference method was used till the late 1960's. However, since the finite element technique gained popularity, its use superceded the finite difference method, and has applications in a wide range of Electrical Engineering problems.

For the solution of bounded field problems the finite element method is used in various disciplines of engineering, examples are : O.C. Zienkiewicz in structural mechanics and C.A. Brebbia in fluid flow. The Finite Element method for the solution of electromagnetic field problems in bounded domains has been employed extensively by P.P. Silvester.

The solution of unbounded field problems can be done using various techniques, each having its own problems and limitations. The simplest technique is truncation, which cuts off the domain at some distant point. This method is easily implemented, because no modification of the finite element technique is required and is therefore widely used, but it yields inaccurate results, and is expensive as far as computer time is concerned.

Another technique involves the coupling of boundary integrals and standard finite elements. This was first suggested by O.C. Zienkiewicz, thereafter various formulations were developed (see [34]), but problems still arose due to the very different nature of the equations produced by the two techniques.

A more recent technique is the coupling of finite elements and infinite elements, to model the unbounded region. Various formulations exist, but are in fact variations of those produced by P. Bettes [3] and G. Beer and J.L. Meek [1]. This technique yielded reasonably good results, when the infinite elements were used "correctly", but this was not always a simple matter, because some infinite elements needed tuning of their geometrical specifications, and others needed tuning of their numerical integration. Both these drawbacks have been overcome by the mapped infinite elements used by L. Resende [20]. The elements of the mesh are mapped onto master infinite elements for which the geometrical specifications are fixed and for which decay shape-functions are known.

The methods of truncation and boundary integrals have been used in electromagnetic fields, but infinite elements, as far as known, have not been applied to this field. In this thesis, therefore finite elements are coupled with mapped infinite elements for the solution of electromagnetic field problems. The research is restricted to two-dimensional

static electromagnetic fields. Skin effect problems are not considered.

The program which is developed and implemented, inputs domain contour information from a data file, because no sophisticated graphics equipment was available. The progaram generates a triangular finite element mesh for the bounded region and an infinite element mesh for the unbounded region. The stiffness matrix is then calculated which is then used to determine the potential function at the nodes. This thesis is divided into four chapters.

Chapter 2 begins by giving the derivation of the energy functional which is then applied to second order triangular finite elements and then to two types of infinite elements. For the infinite elements, formulation is first done in one-dimension and this is then extended to two-dimensions. Finally the implementation of boundary conditions, and the handling of anisotropic media are dealt with.

In Chapter 3, the generation of triangular finite elements and infinite elements are discussed. Then, the generation of symmetrical meshes for axis-symmetric problems are dealt with.

Chapter 4 deals with two topics: Firstly, the equation solver which is a special technique used to speed up the process. Secondly, the post-processor which enables the user to determine potential distributions and components of flux

along any horizontal or vertical line. This is a primitive form of post-processing, because its enhancement was limited due to the lack of graphics facilities.

Chapter 5 begins by giving a detailed layout of the data input file. It then presents results for three examples with which the program was tested. The results are compared to analytical results, for two of the examples, from which a conclusion is drawn about the accuracy of the program.

CHAPTER 2

FINITE ELEMENTS AND ELECTROMAGNETIC FIELDS

The Poisson equation describes electrostatic and magnetostatic fields in Electrical Engineering. The Poisson equation is given by :

$$K \frac{\delta^2 u}{\delta x^2} + K \frac{\delta^2 u}{\delta y^2} = P \qquad (2.1)$$

Where K is the material constant, P is the source of the region and u is the potential function.

The electrostatic field is described by the scalar Poisson equation, whereas the magnetostatic field is described by the curl equation, but the introduction of the vector potential function and the Coulomb guage allows this curl equation to be expressed as a vector Poisson equation.

Analytical solutions to the Poisson equation only exist for a few problems of simple geometries . For problems of non-orthogonal contours, numerical methods must be used, of these, the finite element method is one of the most versatile.

This chapter begins by giving a brief review of the work done by previous researchers. It then goes on to formulating the energy functional for appropriate boundary conditions. The energy functional is then used to derive element matrices of finite elements for the bounded region, and matrices of special elements are derived for the unbounded region.

For the solution of bounded problems various people did research in various fields : O.C. Zienkiewicz and Y.K. Cheung [30], C.A. Brebbia and A.J. Ferrante [4] did work in structural mechanics; C.A. Brebbia and J.J. Connor [6] in fluid flow; the most prominent worker in the field of electromagnetics, is P.P. Silvester [22,23,24]. In the book "Finite Elements for Electrical Engineers" Silvester uses linear triangular elements, in [23] he uses higher order triangular elements, which resulted in solving problems with a much greater accuracy. For this reason, second order triangular elements were used.

For the solution of unbounded problems there are a variety of methods employed. They are briefly as follows :

(i) Truncation - the infinite region is cut off at a distant point.

(ii) The exterior region is modelled by an annular "superelement". This method is a recursion technique, see [25,15]. This was found to be time consuming, and was modified by P. Silvester and M. Hsieh [26], to where no recursion was required.

(iii) Damping of the infinite domain was proposed by Zienkiewicz and Newton [31].

(iv) The boundary integral technique was employed by many workers : Zienkiewicz et al [32,33], B. Mc Donald and A. Wexler [18].

(v) Infinite elements were used by P. Bettes [2,3], and P.P. Lynn and Hassoun A. Hadid [16].

(vi) Recently mapped infinite elements are employed by F. Damjanic and D.R.J. Owen [8], L. Resende [20].

In Zienkiewicz et al [34], a detailed review is done of the above methods, in which they expect mapped infinite elements to give an improvement in the results. L.Resende [20] also does a review of some of the above methods, he concluded that the mapped infinite elements, which he implemented, performed extremely well. For this reason, mapped infinite elements were used.

2.1 FORMULATION OF THE ENERGY FUNCTIONAL

By the calculus of variations it is easy to show that the solution of the Poisson equation is identical to finding the potential function u, such that u minimises the energy functional F, given by :

$$F = \frac{1}{2}\int\int K \left[\frac{\delta^2 u}{\delta x^2} + \frac{\delta^2 u}{\delta y^2} \right] dx\, dy - \int\int Pu\, dx\, dy \qquad (2.1.1)$$

The boundary conditions considered are basically of two types : (see Fig. 2.1)

(a) The Dirichlet boundary condition on S_A.

$$u(S) = \bar{u} \qquad (2.1.2)$$

Where $\bar{u}$ is a prescribed value of u.

(b) The Neuman boundary condition on S_B.

$$K \frac{\delta u}{\delta n} = \bar{v}_n = 0 \qquad (2.1.3)$$

Where n is the direction normal to the boundary.

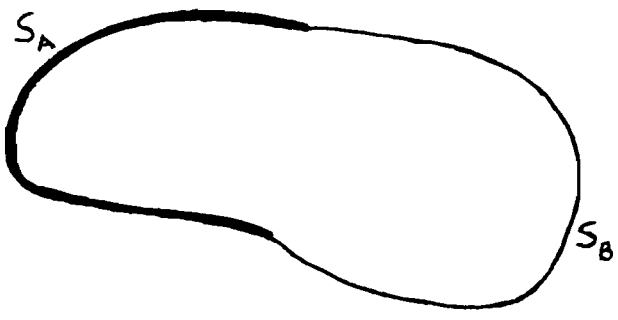


Figure 2.1 2-D domain with different boundary conditions

Using Galerkin's method of weighted residuals, equation (2.1.1) can be written as :

$$\int\int K\left[\frac{\delta^2 u}{\delta x^2} + \frac{\delta^2 u}{\delta y^2}\right]\eth u\, dx\, dy - \int\int Pu\, \eth u\, dx\, dy = 0 \qquad (2.1.4)$$

Integrating by parts using Gauss' theorem the Neuman boundary conditions of the problem can be found.

$$\int\int K\left[\frac{\delta u}{\delta x}\frac{\delta \eth u}{\delta x} + \frac{\delta u}{\delta y}\frac{\delta \eth u}{\delta y}\right]dxdy - \int\int Pu\eth u dxdy = K\int\frac{\delta u}{\delta x}\eth u dy + K\int\frac{\delta u}{\delta y}\delta u dx \qquad (2.1.5)$$

The right-hand side of (2.1.5) is given by :

$$\int_S K\left[\frac{\delta u}{\delta x}\frac{\delta y}{\delta S} - \frac{\delta u}{\delta y}\frac{\delta x}{\delta S}\right]\eth u dS = \int_S K\frac{\delta u}{\delta n}\eth u dS \qquad (2.1.6)$$

Using the natural boundary condition of the form given in (2.1.3) equation (2.1.6) becomes :

$$\int\int K\left[\frac{\delta u}{\delta x}\frac{\delta \eth u}{\delta x} + \frac{\delta u}{\delta y}\frac{\delta \eth u}{\delta y}\right]dxdy - \int\int Pu\eth u dxdy = \int_{S_B}\left[K\frac{\delta u}{\delta n} - \bar{v}_n\right]\eth u dS \qquad (2.1.7)$$

Integrating by parts as before we get :

$$\iint K\left[\frac{\delta u}{\delta x}\frac{\delta \eth u}{\delta x} + \frac{\delta u}{\delta y}\frac{\delta \eth u}{\delta y}\right]dxdy - \iint P u \eth u dxdy = \int_{S_B} \bar{v}_n \eth u dS \qquad (2.1.8)$$

The energy functional in this form will be applied to the different types of elements considered in this thesis. For each type of element trial functions will be chosen which model the potential inside the element. These trial functions together with the energy functional will be used to derive characteristic or element matrices for each type of element.

2.2 SECOND ORDER TRIANGULAR ELEMENTS FOR BOUNDED DOMAINS

The calculation of element matrices can be done in two ways:

(i) A master element is chosen for which trial functions for the potential are known. The master element is mapped onto each element from the mesh, see Fig. 2.2, the energy functional, trial functions, and the mapping can be used to determine the element matrices. This method is described by R.K. Livesley [14].

(ii) The Cartesian Coordinates of a triangle are changed to triangular coordinates, which are then used to derive the element matrices directly. This is also the method employed here.

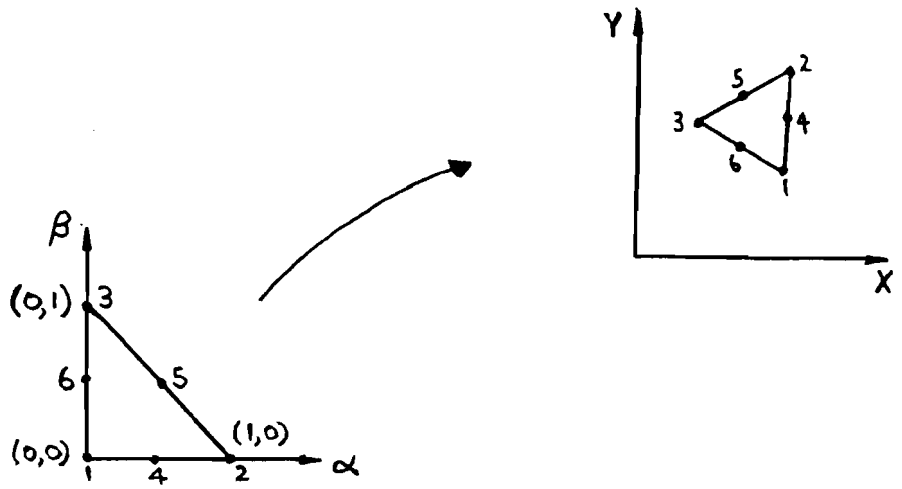


Figure 2.2 Master element mapped onto a triangle of mesh

2.2.1 Relationship between Cartesian and Triangular coordinates

The nodes for the triangles are numbered in an anti-clockwise order. The triangular coordinates L_1,L_2 and L_3 can be made dimensionless with respect to side length, and vary between 0 and 1. They are shown in Fig. 2.3, where coordinates are in the form (L_1,L_2,L_3).

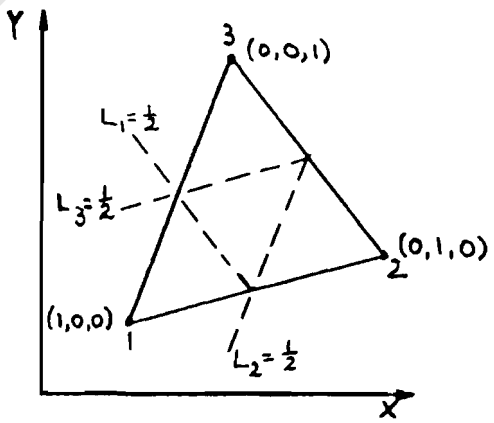


Figure 2.3 Triangular coordinates

The following relations may be written in order to establish the relationship between triangular and Cartesian coordinates.

$$x = x_3 + (x_1 - x_3)L_1 + (x_2 - x_3)L_2 \qquad (2.2.1)$$

$$y = y_3 + (y_1 - y_3)L_1 + (y_2 - y_3)L_2 \qquad (2.2.2)$$

The sum of L_1 and L_2 are constant along side 3 i.e.

$$L_1 + L_2 = 1 \qquad (2.2.3)$$

Where (x_i, y_i) are the coordinates of the node i.

Equations (2.2.1) and (2.2.2) can be rearranged to give :

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} x_1-x_3 & x_2-x_3 \\ y_1-y_3 & y_2-y_3 \end{bmatrix} * \begin{bmatrix} L_1 \\ L_2 \end{bmatrix} + \begin{bmatrix} x_3 \\ y_3 \end{bmatrix} \qquad (2.2.4)$$

This can be inverted to give :

$$L_1 = \frac{1}{2A}\left[A_1 + b_1 x + a_1 y \right] \qquad (2.2.5)$$

$$L_2 = \frac{1}{2A}\left[A_2 + b_2 x + a_2 y \right] \qquad (2.2.6)$$

Where,

$$a_1 = x_3 - x_2 , \quad b_1 = y_2 - y_3 , \quad A_1 = x_2y_3 - x_3y_2$$

$$a_2 = x_1 - x_3 , \quad b_2 = y_3 - y_1 , \quad A_2 = x_3y_1 - x_1y_3$$

A = total area of triangle = $\frac{1}{2}(b_1a_2 - b_2a_1)$

There are three coordinates altogether, but only two of them are independent, the third can be determined from the relation :

$$L_1 + L_2 + L_3 = 1 \qquad (2.2.7)$$

L_i is equal to 1 at node i but is zero on side i. Although these coordinates are not independent, they are convenient for generating trial functions.

2.2.2 Trial Functions

Trial functions are functions which describe the potential function, u at any point inside the triangle, in terms of the potentials at the nodes of the triangle. The trial functions, or shape-functions as they are often called, must fulfil certain conditions :

(i) They must be continuous inside the element as well as across inter-element boundaries, up to the order (n-1), where n is the highest order derivative in the functional.

(ii) As the number of elements increases, the partial derivatives within the energy functional tend to a constant value.

The trial functions can be expressed as :

$$u = \mathbf{A}^T(x,y)\ \mathbf{a} \qquad (2.2.8)$$

Where **a** is a set of parameters. For a six noded triangle, (2.2.8) is :

$$u = \alpha_1 + \alpha_2 x + \alpha_3 y + \alpha_4 x^2 + \alpha_5 xy + \alpha_6 y^2 \qquad (2.2.9)$$

The number of trial function parameters is equal to the number of nodes. Specialising equation (2.2.8) for element nodes, we get :

$$\mathbf{u}_n = \mathbf{C}(x_i,y_i)\ \mathbf{a} \qquad (2.2.10)$$

where $\mathbf{u}_n$ is the vector of nodal potentials, and (x_i,y_i) are the coordinates of the nodes of the triangle.

$$\mathbf{a} = \mathbf{C}^{-1}\ \mathbf{u}_n \qquad (2.2.11)$$

Using (2.2.11) and (2.2.8) we get :

$$u = \left[\ \mathbf{A}^T\mathbf{C}^{-1}\ \right]\ \mathbf{u}_n = \mathbf{\phi}^T\ \mathbf{u}_n \qquad (2.2.12)$$

The terms in ϕ are the trial functions.

Writing (2.2.9) in terms of triangular coordinates.

$$u = \alpha_1 + \alpha_2 L_1 + \alpha_3 L_2 + \alpha_4 L_1^2 + \alpha_5 L_1 L_2 + \alpha_6 L_2^2 \qquad (2.2.13)$$

Applying the above method, the trial functions at the vertices are found to be :

$$\phi_i = L_i(2L_i - 1) \qquad \text{for } i = 1,2,3 \qquad (2.2.14)$$

and at the midside nodes to be :

$$\phi_j = 4L_kL_i \text{ for } j = 4,5,6 \quad k = 1,2,3 \text{ and } l = 2,3,1 \qquad (2.2.15)$$

These trial functions can now be used together with the energy funtional to obtain the element matrices.

2.2.3 Evaluation of Stiffness Matrices

Consider an element without any boundary conditions imposed. The boundary conditions will be inserted afterwards. In equation (2.2.12) ϕ is given by :

$$\phi = \begin{bmatrix} L_1(2L_1 - 1) \\ L_2(2L_2 - 1) \\ L_3(2L_3 - 1) \\ 4L_1L_2 \\ 4L_2L_3 \\ 4L_3L_1 \end{bmatrix} \qquad (2.2.16)$$

and we know that

$$u = \phi^T u_n = \begin{bmatrix} \phi_1, \phi_2, \phi_3, \phi_4, \phi_5, \phi_6 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \end{bmatrix} \qquad (2.2.17)$$

To evaluate the energy functional in (2.1.8) we need to obtain the derivatives of (2.2.17) with respect to x and y .

As a result of the conditions which the trial functions satisfy, these derivatives will be continuous inside the element, and at its inter-element boundaries.

$$\frac{\delta u}{\delta x} = \frac{\delta u}{\delta L_1}\frac{\delta L_1}{\delta x} + \frac{\delta u}{\delta L_2}\frac{\delta L_2}{\delta x} + \frac{\delta u}{\delta L_3}\frac{\delta L_3}{\delta x} \qquad (2.2.18)$$

$$\frac{\delta u}{\delta y} = \frac{\delta u}{\delta L_1}\frac{\delta L_1}{\delta y} + \frac{\delta u}{\delta L_2}\frac{\delta L_2}{\delta y} + \frac{\delta u}{\delta L_3}\frac{\delta L_3}{\delta y} \qquad (2.2.19)$$

Using (2.2.13), (2.2.16) and (2.2.17), it can be shown that the derivatives are given by :

$$\frac{\delta u}{\delta x} = \frac{1}{2A}\sum_i b_i \frac{\delta u}{\delta L_i} = \frac{1}{2A}\mathbf{b}^T \Phi\, \mathbf{u}_n \qquad (2.2.20)$$

$$\frac{\delta u}{\delta y} = \frac{1}{2A}\sum_i a_i \frac{\delta u}{\delta L_i} = \frac{1}{2A}\mathbf{a}^T \Phi\, \mathbf{u}_n \qquad (2.2.21)$$

where

$$\mathbf{b}^T = \begin{bmatrix} b_1 , b_2 , b_3 \end{bmatrix} \qquad (2.2.22)$$

$$\mathbf{a}^T = \begin{bmatrix} a_1 , a_2 , a_3 \end{bmatrix} \qquad (2.2.23)$$

$$\Phi = \begin{bmatrix} 4L_1-1 & 0 & 0 & 4L_2 & 0 & 4L_3 \\ 0 & 4L_2-1 & 0 & 4L_1 & 4L_3 & 0 \\ 0 & 0 & 4L_3-1 & 0 & 4L_2 & 4L_1 \end{bmatrix} \qquad (2.2.24)$$

These equations can now be substituted into the variational statement (2.1.8), omitting boundary conditions the right-hand side becomes zero and only the left-hand side needs to be considered :

$$\delta u_n^T \frac{1}{4A^2}\iint K\left[\Phi^T bb^T \Phi + \Phi^T aa^T \Phi\right] dxdy - \delta u_n^T \iint P\phi \, dxdy \qquad (2.2.25)$$

Write the following equations as three matrices :

$$I_b = \iint \Phi^T \, b \, b^T \, \Phi \, dA \qquad (2.2.26)$$

$$I_a = \iint \Phi^T \, a \, a^T \, \Phi \, dA \qquad (2.2.27)$$

$$I_p = \iint P \, \phi \, dA \qquad (2.2.28)$$

Integrals I_a and I_b are similar thus only one will be considered for evaluation. In order to evaluate the integral, consider the following integral formula given in [6] (page 104).

$$\iint L_1^i \, L_2^j \, L_3^k \, dA = \frac{i!\, j!\, k!}{(i + j + k + 2)!} \, 2A \qquad (2.2.29)$$

Applying this formula to each of the terms of I_b, I_b can be evaluated and is given below.

$$I_b = A \begin{bmatrix} b_1^2 & -\frac{1}{3}b_1b_2 & -\frac{1}{3}b_1b_3 & \frac{4}{3}b_1b_2 & 0 & \frac{4}{3}b_1b_3 \\ & b_2^2 & -\frac{1}{3}b_2b_3 & \frac{4}{3}b_1b_2 & \frac{4}{3}b_2b_3 & 0 \\ & & b_3^2 & 0 & \frac{4}{3}b_2b_3 & \frac{4}{3}b_1b_3 \\ & & & \frac{8}{3}(b_1^2+b_1b_2+b_2^2) & \frac{4}{3}(b_1b_2+b_2^2+b_2b_3)+\frac{8}{3}b_1b_3 & \frac{4}{3}(b_1b_3+b_1^2+b_1b_2)+\frac{8}{3}b_2b_3 \\ & \text{SYMMETRY} & & & \frac{8}{3}(b_2^2+b_2b_3+b_3^2) & \frac{4}{3}(b_2b_3+b_3^2+b_1b_3)+\frac{8}{3}b_1b_2 \\ & & & & & \frac{8}{3}(b_3^2+b_1b_3+b_1^2) \end{bmatrix} \quad (2.2.30)$$

The integral I_p can be evaluated as follows :

$$I_p = P \iint \begin{bmatrix} 2L_1^2 - L_1 \\ 2L_2^2 - L_2 \\ 2L_3^2 - L_3 \\ 4L_1L_2 \\ 4L_2L_3 \\ 4L_1L_3 \end{bmatrix} dA \qquad (2.2.31)$$

Using the integral formula (2.2.29) each term can be evaluated giving :

$$I_p = P \begin{bmatrix} 0 \\ 0 \\ 0 \\ A/3 \\ A/3 \\ A/3 \end{bmatrix} \qquad (2.2.32)$$

The problem can then be written as :

$$\frac{1}{4A^2} \left[K \left(I_b + I_a \right) \right] = I_p \qquad (2.2.33)$$

To insert the Neuman boundary condition look at the right-hand side of equation (2.1.8). Assume that a Neuman boundary condition occurs along side 3 as shown in Fig. 2.4.

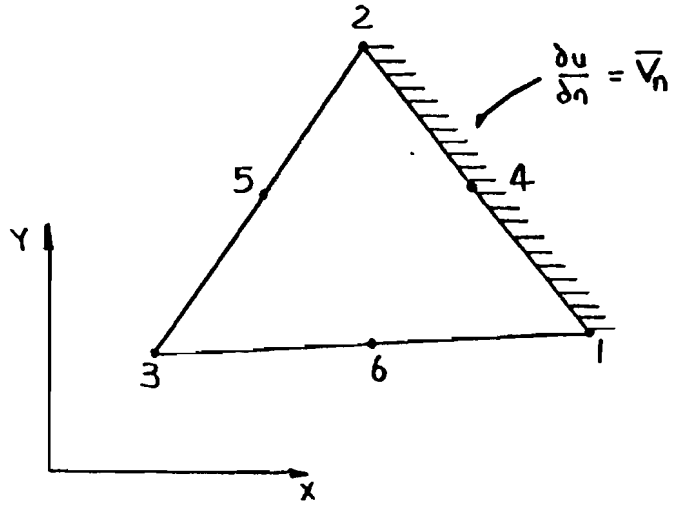


Figure 2.4 Implementing boundary conditions

Along side 3, $L_3 = 0$ and therefore ϕ is given by :

$$\phi = \begin{bmatrix} L_1(2L_1 - 1) \\ L_2(2L_2 - 1) \\ 0 \\ 4L_1L_2 \\ 0 \\ 0 \end{bmatrix} \qquad (2.2.34)$$

The integral for the Neuman boundary condition is :

$$I_n = \int_S \bar{v}_n \, \partial u \, dS = \int_S \bar{v}_n \, \phi \, dS \qquad (2.2.35)$$

For the example considered, S is the side 3 and I_n becomes :

$$I_n = \int_2^1 \bar{v}_n \, \phi \, dS = \int_2 \bar{v}_n \begin{bmatrix} L_1(2L_1 - 1) \\ L_2(2L_2 - 1) \\ 0 \\ 4L_1L_2 \\ 0 \\ 0 \end{bmatrix} dS \qquad (2.2.36)$$

To evaluate this integral, another formula given in [6] (page 104) will be used, and is given by :

$$\int L_1^{\,i} \, L_2^{\,j} \, dS = \frac{i!\ j!}{(i + j + 1)!} * l \qquad (2.2.37)$$

where l is the length of S.

Using this formula, equation (2.2.36) becomes :

$$I_n = \begin{bmatrix} 1/6 \ l \\ 1/6 \ l \\ 0 \\ 2/3 \ l \\ 0 \\ 0 \end{bmatrix} \qquad (2.2.38)$$

where l = length of side 3

The problem may now be written as :

$$\frac{1}{4A^2}\left[K\ (I_b + I_a) \right] = I_p - I_n \qquad (2.2.39)$$

or

$$\mathbf{S}\ u_n = \mathbf{Q} \qquad (2.2.40)$$

S is referred to as the Stiffness matrix.

2.3 MAPPED INFINITE ELEMENTS FOR UNBOUNDED DOMAINS

Although all the infinite element formulations, so far published [2,3,16], yield reasonably good results, however, applying these elements correctly is not always a straight forward matter. Some elements have to be tuned as far as their geometrical specification is concerned, this is due to the arbitrary definition of the decay shape-functions. Other infinite elements needed tuning of the numerical integration approximation. The mapped infinite elements that will be employed here, overcome these drawbacks [20].

The infinite elements are basically of two types :

(i) Those, that are treated as conventional finite elements, except that the integration is carried out to infinity although, the actual geometry does not extend to infinity. These were developed by P. Bettes [3].

(ii) Those, where an element is mapped onto an undistorted parent element which actually extends to infinity.

However, the integrations are carried out over a finite domain which is very advantageous. These were developed by G. Beer and J.L. Meek [1].

All other existing infinite elements are variations of the above two formulations.

2.3.1 Geometrical Mapping

A one-dimensional case may be used for the derivation of the geometrical mapping functions (see [20]).

The example used, is shown in Fig. 2.5.

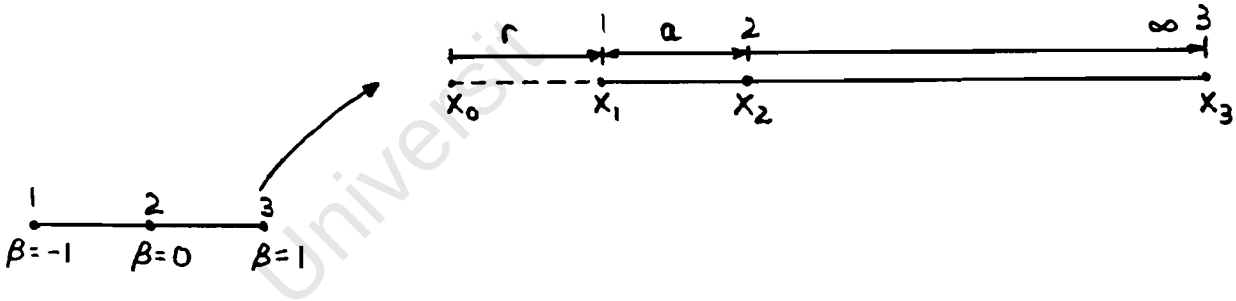


Figure 2.5 One-dimensional infinite element

The parent element extending from nodes 1 to 3 ($-1 \leq \beta \leq 1$) in the local coordinate system is mapped onto the undistorted element which extends from nodes 1 through 2, to node 3 at infinity. The position of the pole x_o is

arbitrary, but must be outside the element. In this case it is chosen so that x_2 is given by :

$$x_2 = 2x_1 - x_o \qquad (2.3.1)$$

The geometrical interpolation is given by :

$$x(\beta) = \Sigma\, N_i(\beta)\, x_i \qquad (2.3.2)$$

Where the summation extends over the finite nodes only and the geometrical mapping functions are given by :

$$N_1(\beta) = \frac{-2\,\beta}{1 - \beta} \qquad (2.3.3a)$$

$$N_2(\beta) = 1 - N_1(\beta) \qquad (2.3.3b)$$

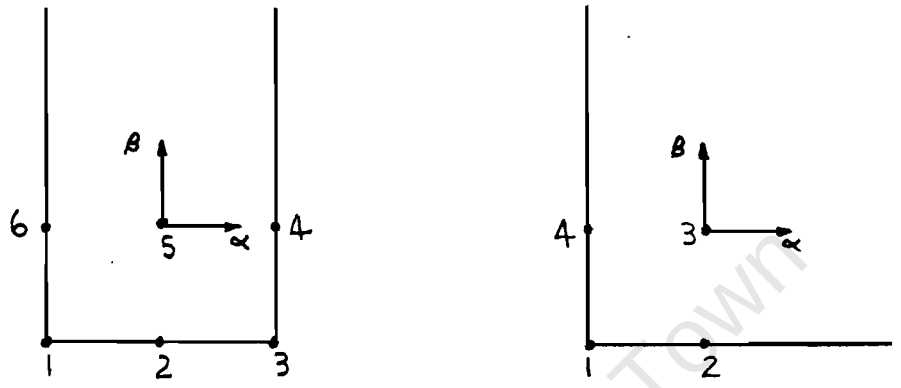
Examining the above mapping it can be seen that $\beta = -1,0,1$ correspond to $x = x_1, x_2, \infty$. It should be noted that the condition :

$$\sum_{i=1}^{2} N_i(\beta) = 1 \qquad (2.3.4)$$

which is satisfied, is necessary for the mapping to be independent of the coordinate system.

The process of constructing 2-dimensional geometrical shape-functions involves combining the 1-dimensional shape-functions in two directions, by taking the product of the two.

The geometrical shape-functions are constructed for two elements which are employed, namely the one-way infinite quadratic Lagrangian element, and the two-way infinite quadratic Lagrangian element, they are shown in Fig. 2.6 (a) and (b) respectively.



(a) (b)

Figure 2.6 Infinite elements

For the one-way element, the α direction is finite, and standard Lagrangian quadratic shape-functions are used as are in [20] (page 8) and for the α direction they are :

$$M_1 = \tfrac{1}{2}\alpha\left[\alpha - 1\right] \qquad (2.3.5a)$$

$$M_2 = \left[1 - \alpha^2\right] \qquad (2.3.5b)$$

$$M_3 = \tfrac{1}{2}\alpha\left[\alpha + 1\right] \qquad (2.3.5c)$$

The shape-functions for the one-way infinite element can now be constructed using (2.3.3) and (2.3.5) and are given by :

$$G_1 = M_1(\alpha)\ N_1(\beta) = -\alpha\beta\ \frac{\alpha - 1}{1 - \beta}$$

$$G_2 = M_2(\alpha)\ N_1(\beta) = -2\beta\ \frac{1 - \alpha^2}{1 - \beta}$$

$$G_3 = M_3(\alpha)\ N_1(\beta) = -\alpha\beta\ \frac{\alpha + 1}{1 - \beta}$$

$$G_4 = M_3(\alpha)\ N_2(\beta) = \tfrac{1}{2}\alpha \left[\alpha + 1\right] \left[1 + \frac{2\beta}{1 - \beta}\right]$$

$$G_5 = M_2(\alpha)\ N_2(\beta) = \left[1 - \alpha^2\right] \left[1 + \frac{2\beta}{1 - \beta}\right]$$

$$G_6 = M_1(\alpha)\ N_2(\beta) = \tfrac{1}{2}\alpha \left[\alpha - 1\right] \left[1 + \frac{2\beta}{1 - \beta}\right]$$

(2.3.6)

For the two-way element both directions extend to infinity, and only (2.3.3) need to be used, and are given by :

$$G_1 = N_1(\alpha)\ N_1(\beta) = \frac{4\alpha\beta}{(1 - \alpha)(1 - \beta)}$$

$$G_2 = N_2(\alpha)\ N_1(\beta) = \frac{-2\beta}{1 - \beta} \left[1 + \frac{2\alpha}{1 - \alpha}\right]$$

$$G_3 = N_2(\alpha)\ N_2(\beta) = \left[1 + \frac{2\alpha}{1 - \alpha}\right] \left[1 + \frac{2\beta}{1 - \beta}\right]$$

$$G_4 = N_1(\alpha)\, N_2(\beta) = \frac{-2\alpha}{1-\alpha}\left[1 + \frac{2\beta}{1-\beta}\right] \qquad (2.3.7)$$

Using these shape-functions , x and y can be expressed as :

$$\begin{bmatrix} x \\ y \end{bmatrix} = \sum_{i=1}^{n} G_i \begin{bmatrix} x_i \\ y_i \end{bmatrix} \qquad (2.3.8)$$

Where n = 4 for the two-way element

and n = 6 for the one-way element

The mapping of the parent element onto the undistorted element can be achieved through the Jacobian Operator which involves partial derivatives of α and β, and is given by :

$$J = \begin{bmatrix} \frac{\delta x}{\delta \alpha} & \frac{\delta y}{\delta \alpha} \\ \frac{\delta x}{\delta \beta} & \frac{\delta y}{\delta \beta} \end{bmatrix} \qquad (2.3.9)$$

Applying this equation to (2.3.8), we get :

$$J = \begin{bmatrix} \sum_{i=1}^{n} \frac{\delta G_i}{\delta \alpha} x_i & \sum_{i=1}^{n} \frac{\delta G_i}{\delta \alpha} y_i \\ \sum_{i=1}^{n} \frac{\delta G_i}{\delta \beta} x_i & \sum_{i=1}^{n} \frac{\delta G_i}{\delta \beta} y_i \end{bmatrix} \qquad (2.3.10)$$

2.3.2 Interpolation Functions

In choosing a shape-function for an element which extends to infinity, there are two requirements : firstly the shape-function should be realistic and secondly it should lead to finite integrations over the element domain.

Considering the one-dimensional case, the Lagrangian quadratic shape-functions given by equation (2.3.5) are used. The potential u can be written generally as :

$$u = \sum_{i=1}^{3} M_i(\beta)\, \phi_i \qquad (2.3.11)$$

Using equations (2.3.5) and (2.3.11) and simplifying the equations we get :

$$\phi = \phi_2 + \tfrac{1}{2}\left[\phi_3 - \phi_1\right]\beta + \tfrac{1}{2}\left[\phi_1 + 2\phi_2 + \phi_3\right]\beta^2 \qquad (2.3.12)$$

Using Fig. 2.5, it can be shown that :

$$\beta = 1 - \frac{2a}{r} \qquad (2.3.13)$$

where $a = x_2 - x_1$ and $r = x - x_o$

Substituting (2.3.13) into (2.3.12) we get :

$$\phi = \phi_3 + \left[-\phi_1 + 4\phi_2 - 3\phi_3\right]\frac{a}{r} + \left[2\phi_1 - 4\phi_2 + 2\phi_3\right]\left[\frac{a}{r}\right]^2 \qquad (2.3.14)$$

Investigating the various nodes of the infinite element, the following can be deduced :

$$\begin{array}{lll} x_o & \frac{a}{r} = \infty & \phi = \infty \\ x_1 & \frac{a}{r} = 1 & \phi = \phi_1 \\ x_2 & \frac{a}{r} = \frac{1}{2} & \phi = \phi_2 \\ x_3 & \frac{a}{r} = 0 & \phi = \phi_3 \end{array} \qquad (2.3.15)$$

It can be seen that x_o is the pole of the decay expansion and it is therefore important that it is outside the element, to avoid any singularities. From (2.3.14) it is evident that we have a reciprocal decay expansion.

A condition of $\phi_3 = 0$ is imposed in the formulation, satisfying the zero field boundary condition at infinity. This also means that the nodes at infinity do not have to be specified.

Constructing interpolation functions for the two 2-D elements can be done by using the shape-functions given in (2.3.5) as follows.

(i) One-way infinite quadratic Lagrangian element

$$H_1 = M_1(\alpha)\, M_1(\beta) = \tfrac{1}{4} \left[\alpha^2 - \alpha \right] \left[\beta^2 - \beta \right]$$

$$H_2 = M_2(\alpha)\, M_1(\beta) = \tfrac{1}{2} \left[1 - \alpha^2 \right] \left[\beta^2 - \beta \right]$$

$$H_3 = M_3(\alpha)\, M_1(\beta) = \tfrac{1}{4} \left[\alpha^2 + \alpha \right] \left[\beta^2 - \beta \right] \qquad (2.3.16)$$

$$H_4 = M_3(\alpha)\, M_2(\beta) = \tfrac{1}{2} \left[\alpha^2 + \alpha \right] \left[1 - \beta^2 \right]$$

$$H_5 = M_2(\alpha)\, M_2(\beta) = \left[1 - \alpha^2 \right] \left[1 - \beta^2 \right]$$

$$H_6 = M_1(\alpha)\, M_2(\beta) = \tfrac{1}{2} \left[\alpha^2 - \alpha \right] \left[1 - \beta^2 \right]$$

(ii) Two-way infinite quadratic Lagrangian element

$$H_1 = M_1(\alpha)\, M_1(\beta) = \tfrac{1}{4} \left[\alpha^2 - \alpha \right] \left[\beta^2 - \beta \right]$$

$$H_2 = M_2(\alpha)\, M_1(\beta) = \tfrac{1}{2} \left[1 - \alpha^2 \right] \left[\beta^2 - \beta \right] \qquad (2.3.17)$$

$$H_3 = M_2(\alpha)\, M_2(\beta) = \left[1 - \alpha^2 \right] \left[1 - \beta^2 \right]$$

$$H_4 = M_1(\alpha)\, M_2(\beta) = \tfrac{1}{2} \left[\alpha^2 - \alpha \right] \left[1 - \beta^2 \right]$$

These interpolation functions together with the Jacobian matrix can be used to determine element stiffness matrices.

2.3.3 Evaluation of Stiffness Matrix Elements

A general description is given of the procedure to evaluate stiffness matrices, because there are two types of elements, and also the shape-functions are complicated. The derivatives of the shape-functions are evaluated in Appendix A. Let n = 4 for two-way elements and n = 6 for one-way elements from now on.

The potential is given in terms of the shape-functions as :

$$u = \left[H_1, \ \ldots \ , H_n \right] \begin{bmatrix} u_1 \\ \cdot \\ \cdot \\ \cdot \\ u_n \end{bmatrix} = \phi^T u_n \qquad (2.3.18)$$

To obtain the Cartesian derivatives apply the chain rule.

$$\frac{\delta u}{\delta x} = \frac{\delta \alpha}{\delta x}\frac{\delta u}{\delta \alpha} + \frac{\delta \beta}{\delta x}\frac{\delta u}{\delta \beta} \qquad (2.3.19a)$$

$$\frac{\delta u}{\delta y} = \frac{\delta \alpha}{\delta y}\frac{\delta u}{\delta \alpha} + \frac{\delta \beta}{\delta y}\frac{\delta u}{\delta \beta} \qquad (2.3.19b)$$

This can be written in matrix form as :

$$\begin{bmatrix} \frac{\delta u}{\delta x} \\ \frac{\delta u}{\delta y} \end{bmatrix} = \begin{bmatrix} \frac{\delta \alpha}{\delta x} & \frac{\delta \beta}{\delta x} \\ \frac{\delta \alpha}{\delta y} & \frac{\delta \beta}{\delta y} \end{bmatrix} \begin{bmatrix} \frac{\delta u}{\delta \alpha} \\ \frac{\delta u}{\delta \beta} \end{bmatrix} \qquad (2.3.20)$$

$$= J^{-1} \begin{bmatrix} \frac{\delta u}{\delta \alpha} \\ \frac{\delta u}{\delta \beta} \end{bmatrix} \qquad (2.3.21)$$

where J^{-1} is the inverse of the Jacobian matrix given by (2.3.10).

Let $\mathbf{b}^T$ and $\mathbf{a}^T$ be the first and second rows of the inverse Jacobian matrix respectively, and are given below.

$$\mathbf{b}^T = \begin{bmatrix} \frac{\delta \alpha}{\delta x} & \frac{\delta \beta}{\delta x} \end{bmatrix} \qquad (2.3.22a)$$

$$\mathbf{a}^T = \begin{bmatrix} \frac{\delta \alpha}{\delta y} & \frac{\delta \beta}{\delta y} \end{bmatrix} \qquad (2.3.22b)$$

The derivatives of the potential with respect to α and β are given by :

$$\begin{bmatrix} \frac{\delta u}{\delta \alpha} \\ \frac{\delta u}{\delta \beta} \end{bmatrix} = \begin{bmatrix} \frac{\delta H_1}{\delta \alpha} & \frac{\delta H_2}{\delta \alpha} & \cdots & \frac{\delta H_n}{\delta \alpha} \\ \frac{\delta H_1}{\delta \beta} & \frac{\delta H_2}{\delta \beta} & \cdots & \frac{\delta H_n}{\delta \beta} \end{bmatrix} \begin{bmatrix} u_1 \\ . \\ . \\ . \\ u_n \end{bmatrix} \qquad (2.3.23)$$

$$= \Phi\, u_n \qquad (2.3.24)$$

Substituting equations (2.3.20) - (2.3.24) into the energy functional, omitting boundary conditions, we get :

$$\delta u_n^T \iint \left[\Phi^T b b^T \Phi + \Phi^T a a^T \Phi \right] |J| \, d\alpha \, d\beta - \delta u_n^T \iint P\phi |J| \, d\alpha \, d\beta \qquad (2.3.25)$$

The Φ matrix can easily be evaluated because H_i varies quadratically over the element's domain. J however, stays constant over the entire element. To check the limits of the above integrals, let's consider each element separately.

(i) In the one-way infinite quadratic Lagrangian element :

In the α direction : $-1 \leq \alpha \leq 1$

In the β direction : $-1 \leq \beta < \infty$

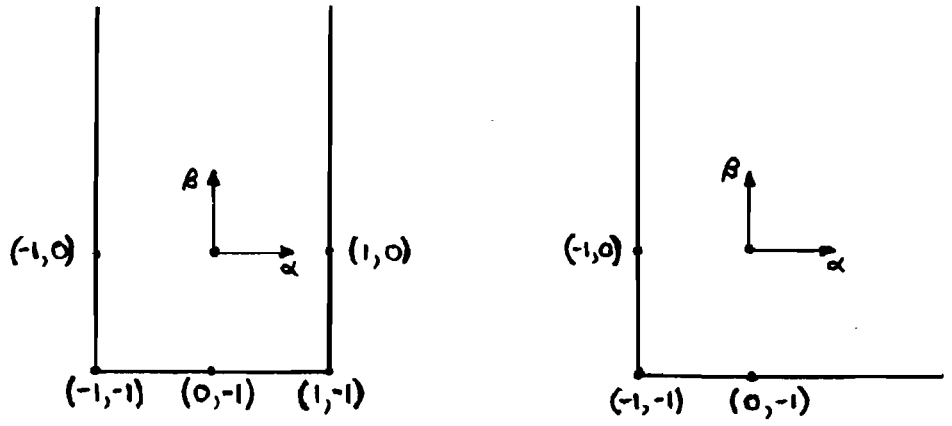
See Fig. 2.7 (a).

(ii) In the two-way infinite quadratic Lagrangian element :

In the α direction : $-1 \leq \alpha < \infty$

In the β direction : $-1 \leq \beta < \infty$

See Fig. 2.7 (b).



(a) (b)

Figure 2.7 Quadratic Lagrangian infinite elements

The limits for the above integrals therefore extend to infinity and numerical integration is required to evaluate the integrals.

2.3.3 (a) Numerical Integration

Two types of integrals need to be considered :

$$\text{(i)} \quad \int_{-1}^{\infty} f(x)\, dx \qquad (2.3.26a)$$

$$\text{(ii)} \quad \int_{-1}^{1} f(x)\, dx \qquad (2.3.26b)$$

To do the first type of integration there are two possibilities :

(i) Multiplying the integrand by an exponential term, and applying the Gauss Laguerre integration formulae.

(ii) To modify the Gauss Legendre integration integration scheme.

The second method was used, and is outlined in Davis and Rabinowitz [9]. Only a brief outline is given below.

$$\int_a^b f(x)\,dx \qquad (2.3.27)$$

in the case of Gauss Legendre integration a = -1 and b = 1. To extend the integral to infinity, do the following substitution :

$$x = \frac{a + bt}{1 + t} \qquad (2.3.28)$$

which changes the integral to :

$$\int_a^b f(x)\,dx = \left[b - a \right] \int_0^\infty f\left[\frac{a + bt}{1 + t} \right] \frac{dt}{1 + t^2} \qquad (2.3.29)$$

After manipulation this reduces to :

$$\int_0^\infty f(t)\,dt = 2 \int_{-1}^{+1} f\left[\frac{1 + x}{1 - x} \right] \frac{dx}{(1 + x)^2} \qquad (2.3.30)$$

Where $x = \dfrac{t - 1}{t + 1}$ and $t = \dfrac{1 + x}{1 - x}$

In case (i) above we need to integrate between -1 and ∞ therefore we define a new variable $\beta = t - 1$, the integral then changes to :

$$\int_{-1}^{\infty} f(\beta)\, d\beta = 2 \int_{-1}^{+1} f\left[\frac{2x}{1-x}\right] \frac{dx}{(1-x)^2} \qquad (2.3.31)$$

Now given the standard Gauss Legendre abscissae and weights, x and w_{old}, new abscissae and weights, β and w_{new} can be evaluated as follows :

$$\beta = \frac{2x}{1-x} \qquad (2.3.32a)$$

$$w_{new} = \frac{2}{(1-x)^2} * w_{old} \qquad (2.3.32b)$$

To do the second integral we can use the standard Gauss Legendre integration formula, as well as the abscissae and the weights. The abscissae and weights are given in Appendix B for second and third order integrations.

The entire process of generating stiffness matrices can best be illustrated by means of a flowchart. (See Fig. 2.8)

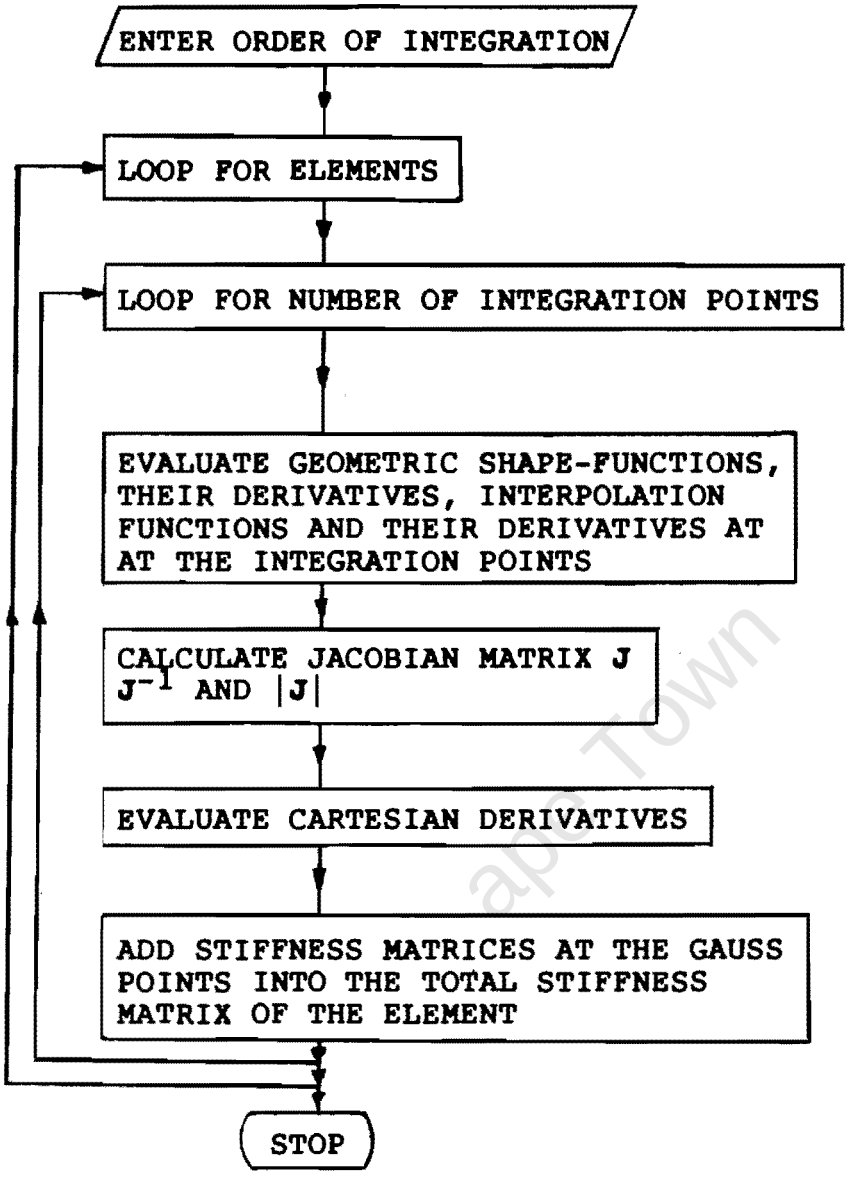


Figure 2.8 Flowchart to evaluate stiffness matrices

2.3.4 Coupling of Infinite and Finite Elements

Although it is possible to obtain solutions using infinite elements alone, better results can be obtained by the combination of infinite and finite elements,as done by L. Resende [20]. Conventional finite elements are used to model

the finite region of interest, which are then surrounded by infinite elements which are used to model the far field region. In order to ensure spatial continuity the number and the coordinates of the nodes at the finite edge of an infinite element must match with those at the outer edge of the finite element, to which it is connected.

The computational changes involved in including infinite elements are minimal as they are treated in a similar manner as the conventional finite elements. The infinite elements produce matrices which have the same banded characteristics as the finite elements.

Examples of problems solved are included in Chapter 5, and the success of the method will be commented on in Chapter 5.

2.4 ANISOTROPIC MEDIA AND IMPLEMENTATION OF BOUNDARY CONDITIONS

2.4.1 Anisotropic Media

The introduction of different media poses no problem to the finite element method for finite regions, as well as unbounded regions.

Consider two subregions Ω_1 and Ω_2 bounded by Γ_1 and Γ_2 respectively. Γ_3 seperates the two regions, and each contains a material that is homogeneous, linear, lossless and isotropic, within the region, having constants K_1 and K_2, as shown in Fig. 2.9. The elements at the boundary, will have a difference in the stiffness matrices, in that the one will be multiplied by K_1 and the other by K_2 which will enter the equation system and in that way the boundary condition at the boundary is automatically implemented. For this reason, no special treatment of the boundary conditions are required at Γ_3 as is required by other methods, such as analytical solutions and the finite difference techniques.

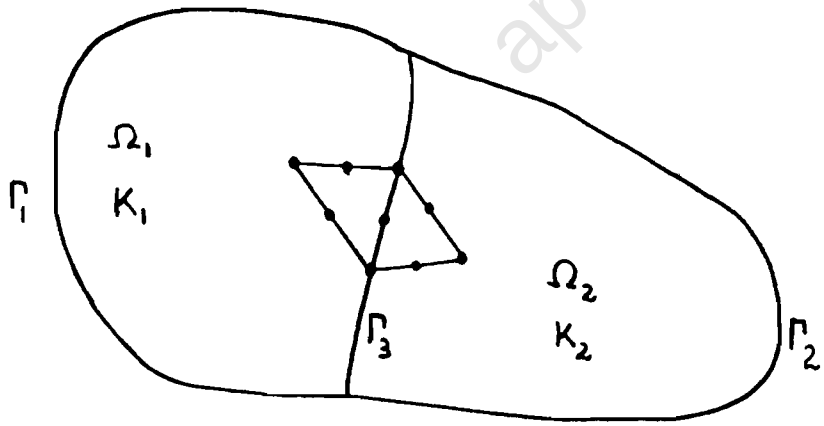


Figure 2.9 A domain with different media

2.4.2 Implementation of Boundary Conditions

The two types of boundary conditions dealt with, are given by equations (2.1.2) and (2.1.3). Both the Dirichlet and the

Neuman boundary conditions are made two vary in one of two ways :

(i) Constant along a segment.

(ii) Vary sinusoidally along a segment.

2.4.2 (a) The Dirichlet boundary condition

The Dirichlet boundary condition is realised in the following way :

If the row of the stiffness matrix corresponds to a node of prescribed potential (Dirichlet condition) all the entries in that row are made zero except the diagonal entry which is made unity. The right-hand side of the row is then set equal to the prescribed potential. For every other row the column corresponding to the prescribed potential is put to zero, and the entry is subtracted from the right-hand side vector. An example is given below.

Consider the 4 x 4 matrix given below, and assume node 3 has a prescribed potential C.

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix} \begin{bmatrix} U_1 \\ U_2 \\ U_3 \\ U_4 \end{bmatrix} = \begin{bmatrix} V_1 \\ V_2 \\ V_3 \\ V_4 \end{bmatrix} \qquad (2.4.1)$$

This matrix will then become :

$$\begin{bmatrix} a_{11} & a_{12} & 0 & a_{14} \\ a_{21} & a_{22} & 0 & a_{24} \\ 0 & 0 & 1 & 0 \\ a_{41} & a_{42} & 0 & a_{44} \end{bmatrix} \begin{bmatrix} U_1 \\ U_2 \\ U_3 \\ U_4 \end{bmatrix} = \begin{bmatrix} V_1 - a_{13}*C \\ V_2 - a_{23}*C \\ C \\ V_4 - a_{43}*C \end{bmatrix} \qquad (2.4.2)$$

2.4.2 (b) Neuman boundary condition

The Neuman boundary condition is implemented as shown by equations (2.2.34 - 2.2.39). For each triangle the midside nodes are checked (nodes 4,5 and 6 in Fig. 2.10)

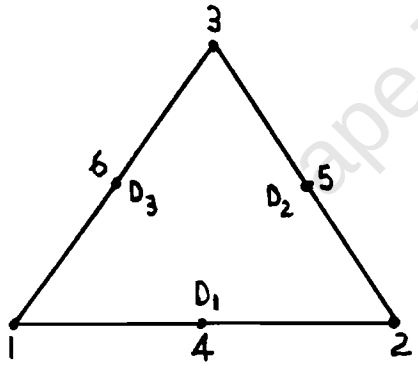


Figure 2.10 Triangular Finite Element

If they correspond to a Neuman boundary condition W_1, W_2, W_3 are set to the value $\bar{v}_n$. Note that the Neuman condition ($\bar{v}_n = 0$) is satisfied naturally i.e. $W_i = 0$, if the Neuman condition occurs at node i. The values of W_1, W_2, W_3 together with D_1, D_2 and D_3 (the lengths of the sides of the triangle), are used to evaluate a vector which is subtracted from the right-hand side of the total equation system, see equation (2.2.39) This vector is given as :

$$\begin{bmatrix} (W_1*D_1 + W_3*D_3)/6 \\ (W_1*D_1 + W_2*D_2)/6 \\ (W_2*D_2 + W_3*D_3)/6 \\ 2/3*W_1*D_1 \\ 2/3*W_2*D_2 \\ 2/3*W_3*D_3 \end{bmatrix} \qquad (2.4.3)$$

If only side D_1 has the Neuman boundary condition attached to it, then $W_2 = W_3 = 0$ and $W_1 = \bar{v}_n$ then (2.4.3) becomes :

$$\bar{v}_n \begin{bmatrix} D_1/6 \\ D_1/6 \\ 0 \\ 2/3*D_1 \\ 0 \\ 0 \end{bmatrix} \qquad (2.4.4)$$

Compare this to equation (2.2.38).

The process of solving the equation system, to get the nodal potentials, will now be dealt with in the next chapter.

CHAPTER 3

MESH GENERATION

The mesh generator implemented will be dealt with in 3 parts, namely:

(i) Generating finite elements for the bounded region.

(ii) Generating infinite elements for the unbounded region.

(iii) Generating of symmetrical meshes for axis symmetric domains.

The versatility of finite elements nowadays permits us to tackle much more complex physical problems, but at the same time we are dismayed at the enormous amount of input required for such problems. The success of the method depends much on the sound descretisation of the domain. As a result of this, numerous computer algorithms of various degrees of automation have been proposed. At one extreme are the fully automatic methods which require minimum user input, for which element densities are determined by the algorithm itself.

At the other extreme are simple methods which require the user to completely define the element mesh and the algorithm only does minor operations such as error detection and consistency checking. The task of the fully automatic methods is complicated and tends to be somewhat inflexible and time consuming to implement, while the simplest of the

mesh generation algorithms fails to relieve the user from tedious decomposition of the domain. With these considerations in mind, it can be said that the semi-automatic mesh generation algorithm used is a good compromise between the two extremes in mesh generation. The semi-automatic mesh generation algorithm does not automatically generate a graded triangulisation of the domain, but does offer the user a large amount of control over the mesh grading which is sometimes necessary if a particular region is of interest. Also the user does not need to provide large amounts of input data.

This mesh generator can handle bounded and unbounded domains. The bounded regions are divided into a triangular element mesh, whereas the unbounded regions can be descretised by using elements extending to infinity in one direction, called one-way infinite elements and elements extending to infinity in two directions, called two-way infinite elements.

3.1 BOUNDED DOMAINS

The mesh generator can create a triangular element mesh within any planar domain which is either simply- or multiply-connected. The existence of openings in the domain do not give rise to any difficulty. Although the algorithm can tackle domains composed of any number of sub-regions,

if a triangular element mesh of consistent size is required for a homogeneous domain, no sub-division of the domain is necessary. The number of elements is not fixed and the relative positions of the nodes are not predetermined by mathematical formulae.

The boundary of the domain is represented by a disjoint union of simple closed loops of straight line segments. For simply-connected regions there is only one closed loop, whereas for multiply-connected regions there may be as many internal loops as there are openings inside the domain. The nodes for the exterior boundary are entered in a counter-clockwise order, while the nodes on the interior boundaries are entered in a clockwise order. Nodes on any particular boundary need not follow a sequence. This flexibility makes it possible to generate a triangular element mesh from one sub-region to the next without having to bother about common boundaries between sub-regions.

The algorithm first generates interior node points according to the mesh size entered for the region to be triangulated. It then connects all the nodes to form triangular elements, in such a way that there is no element overlapping, and the entire region is covered. The triangulisation algorithm is designed to produce elements as near equilateral as possible.

3.1.1 Entering contours

Entering of the contour, nodes and segments are done in the program module DATA. The contours for each sub-region are entered in the program module DIVIDE.

3.1.1 (a) Data module

In this module the coordinates of all the input nodes and the connections between the input nodes called segments, are read in. As the nodes are read in, they are automatically assigned node numbers (starting from 1 and following sequentially). For the segments it is similar. For each segment the start and end node numbers, the segment type, and the mesh size need to be entered.

There are three types of segments:

(i) Straight line segment which is sub-divided into sub-segments. The number of sub-segments is determined from the mesh size for the segment.

(ii) Circular arc for which the start and end node numbers must be entered in a clockwise sequence along the arc. The circular arc is then approximated by straight line sub-segments.

(iii) A segment other than a straight line or a circular arc can be entered by sub-dividing the segment and entering the coordinates of the intermediate nodes.

If the mesh size does not fit the segment exactly, the mesh size is automatically adjusted to give an even distribution of points along the segment. This entire process is illustrated in Fig. 3.1.

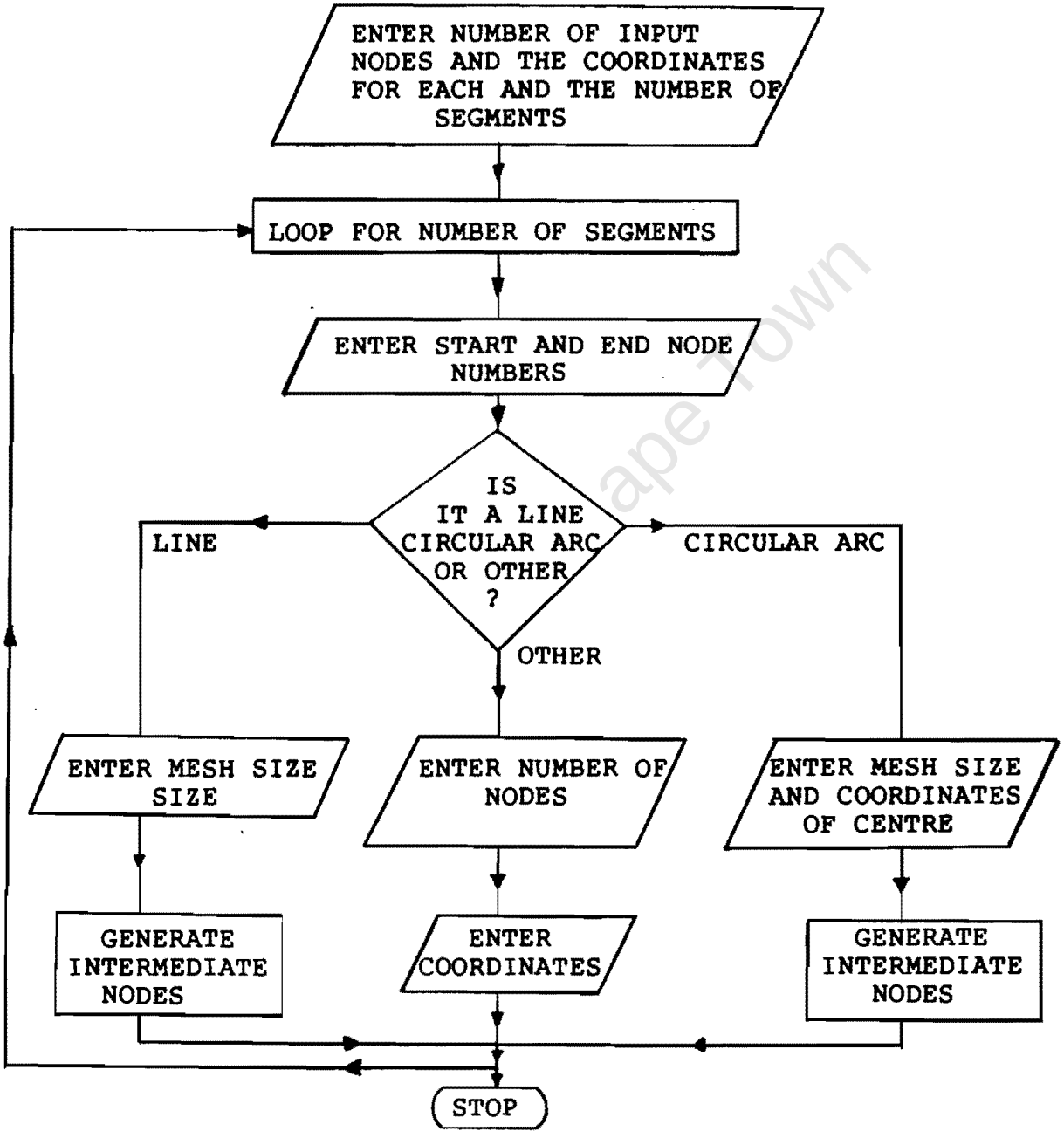


Figure 3.1 Flowchart of Data module

3.1.1 (b) Divide Module

For each sub-region, this module does the following:

(i) Reads in the segment numbers that form the sub-region; in a counter-clockwise order if it is an outer contour or in a clockwise order if it is an inner contour. See Fig. 3.2 .

(ii) Reads in the mesh size for the sub-region.

(iii) It will form a front (a list of all the sub-segments that form the contours of the sub-region), which is used to generate interior nodes, as well as triangles.

(iv) It will call routines that generate interior nodes, that triangulate the sub-region and that adjust interior nodes to make the triangles as near equilateral as possible.

Fig. 3.3 illustrates the procedure of this module.

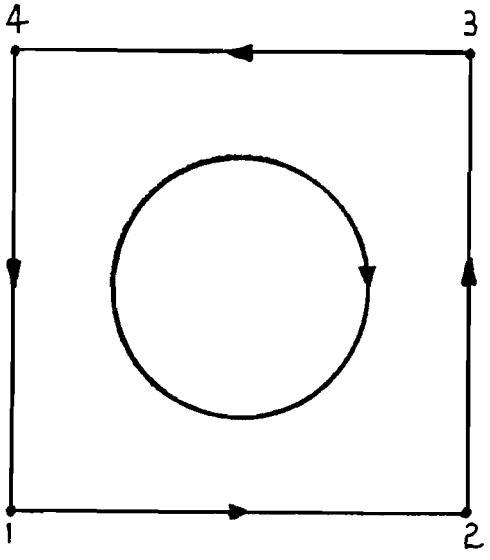


Figure 3.2 Domain illustrating how to enter contours

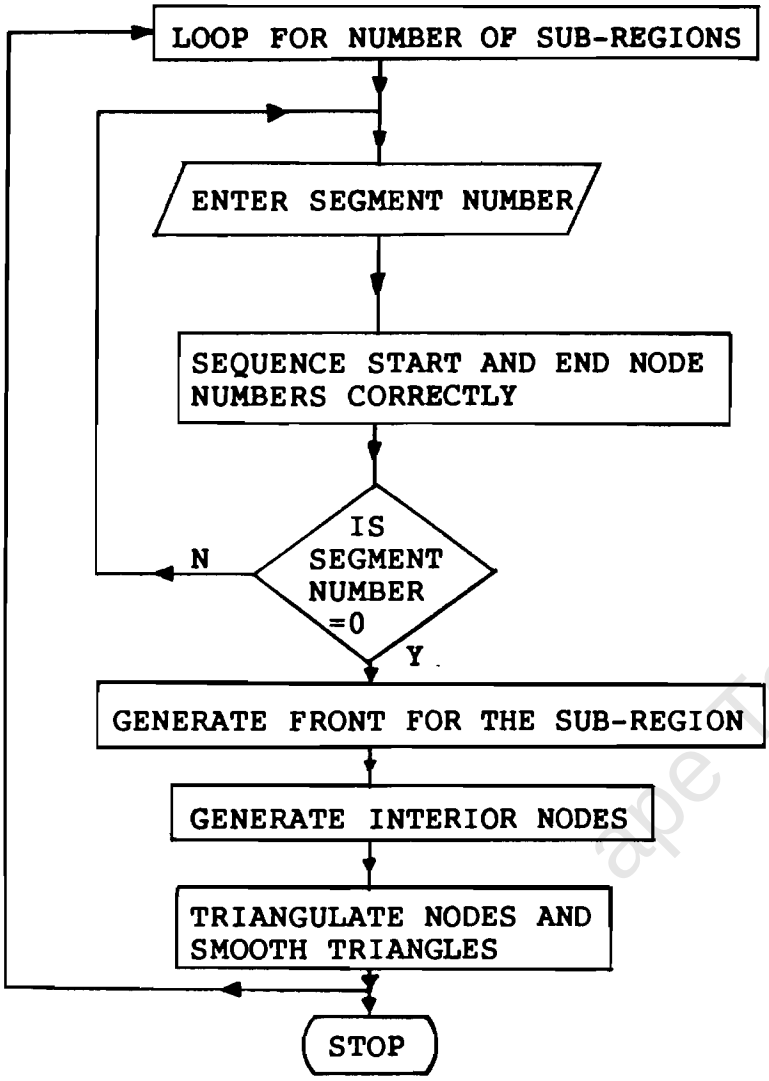


Figure 3.3 flowchart of Divide module

3.1.2 Interior Node Generation

The generation of interior nodes can be achieved by a simple and direct method, nevertheless previous workers in this area have accepted it as a complicated and indirect iterative process. Fukuda and Suhara [10] generated interior nodes by superimposing a square grid, large enough to cover

the whole domain. Nodes were then generated randomly until all the squares of the grid were used. This method has many drawbacks; it is unreliable, gives rise to ill-conditioned elements, and is time consuming due to the squares that occur outside the domain.

Later, Shaw and Pitchen [21] made modifications to the above method, but the process was still time consuming, and could not handle irregular boundaries and openings in the domain.

The algorithm used is as follows:

(i) The minimum and maximum y values are determined.

(ii) Imaginary horizontal lines are drawn at different levels between Y_{min} and Y_{max}. The spacing between these lines is equal to the mesh size for the region.

(iii) Determination of the intersection of a horizontal line with a boundary segment is as follows:

Let $S = \{P_iQ_i, i = 1,N\}$ be a set of line segments representing the domain boundaries and $Y = H$ be the equation of a horizontal line.

The intersecting points or cuts are obtained by considering the line segments of S one by one. When checking a segment, say P_kQ_k, then the next segment $P_{k+1}Q_{k+1}$ is also used to determine whether a boundary has a local maximum or minimum or to determine whether the imaginary horizontal line coincides with a segment of the domain boundary. For the former, two cuts, both

equal will be recorded, to aid in the generation of the interior nodes, see Fig. 3.4 (a), while for the latter a flag will be set for the intersecting point.

Consider a segment P_iQ_i and $X_1 = x(P_i)$, $Y_1 = y(P_i)$, $X_2 = X(Q_i)$ and $Y_2 = Y(Q_i)$. A cut occurs if :

$$(a) \quad \left[Y_1 - H \right] * \left[Y_2 - H \right] < 0 \qquad (3.1.1)$$

$$(b) \quad \left[Y_1 - H \right] * \left[Y_2 - H \right] = 0 \qquad (3.1.2)$$

When an intersection occurs, the coordinates of intersection are given by :

$$\left[\left[X_1 + (H - Y_1)*(X_2 - X_1)/(Y_2 - Y_1) \right], H \right] \qquad (3.1.3)$$

which are recorded together with the sub-segment of the boundary that was intersected. If a cut falls at the beginning or the end of a horizontal boundary segment of the domain then a flag is set to indicate that the cut occurs either at the start or the end of that horizontal boundary segment.

(iv) The intersecting points are then sorted in an increasing order of x.

(v) The cuts are considered in pairs, starting at the first and second cuts. Consider checking the N^{th} and $(N + 1)^{th}$ cuts, skip the pair if :

(a) the N^{th} and $(N + 1)^{th}$ cuts fall on the same horizontal boundary segment. This can be checked by the flag set in (iii). See Fig. 3.4 (b) - intersecting points 1 and 2 are skipped then 2 and 3 are considered.

(b) A dummy node is generated between two cuts, it is then tested whether the node is outside the domain. This is done by checking if the node is to the right of the intersecting sub-segments that are stored for each of the intersecting points. See Fig.3.4 (c) - for intersecting points 1 and 2, node A is generated and is to the left of the sub-segments, and is therefore inside the domain. For intersecting points 3 and 4, node B is to the right of both sub-segments, and is therefore outside the domain.

If a pair is skipped, the next pair considered will be the $(N + 1)^{th}$ and $(N + 2)^{th}$ intersecting points. If a pair is not skipped, the number of nodes to be generated is determined from the mesh size, and generated. The next pair considered will be the $(N + 2)^{th}$ and $(N + 3)^{th}$ intersecting points.

(vi) After all the intersecting points are used, the next horizontal line is considered and the process is

repeated, as is shown in Fig. 3.5, until the entire region is covered.

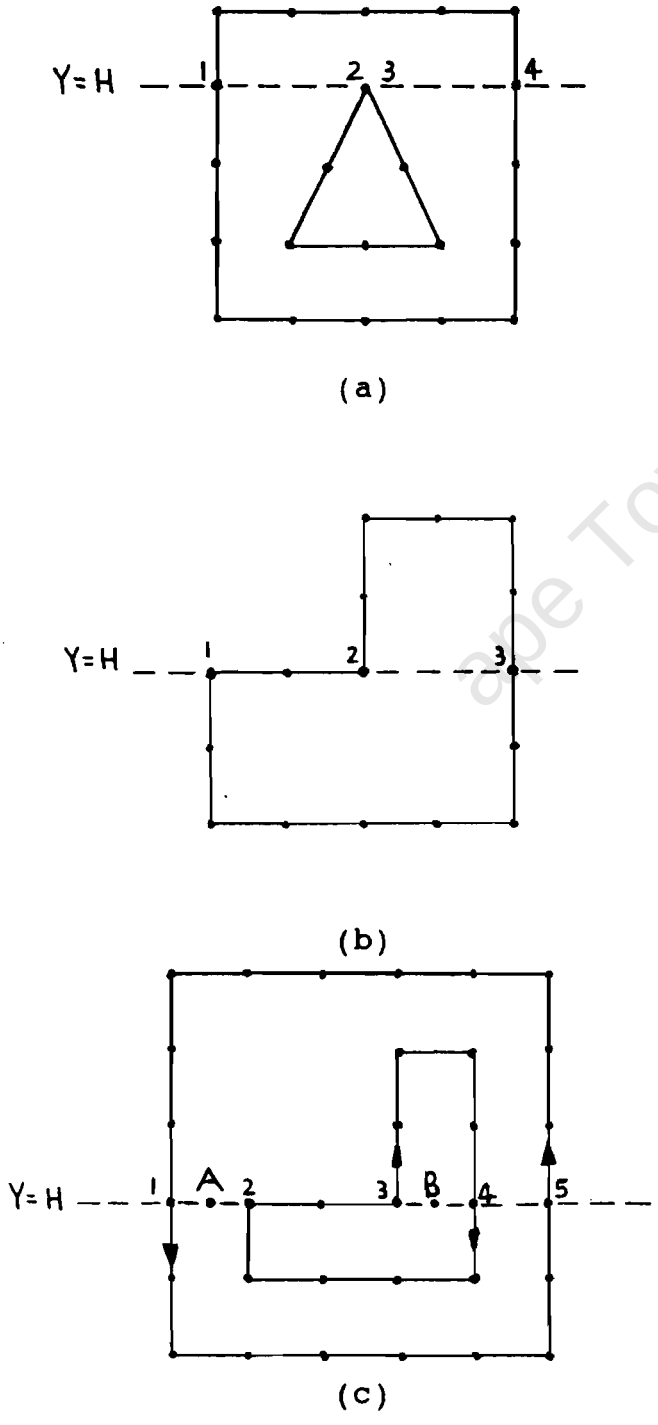
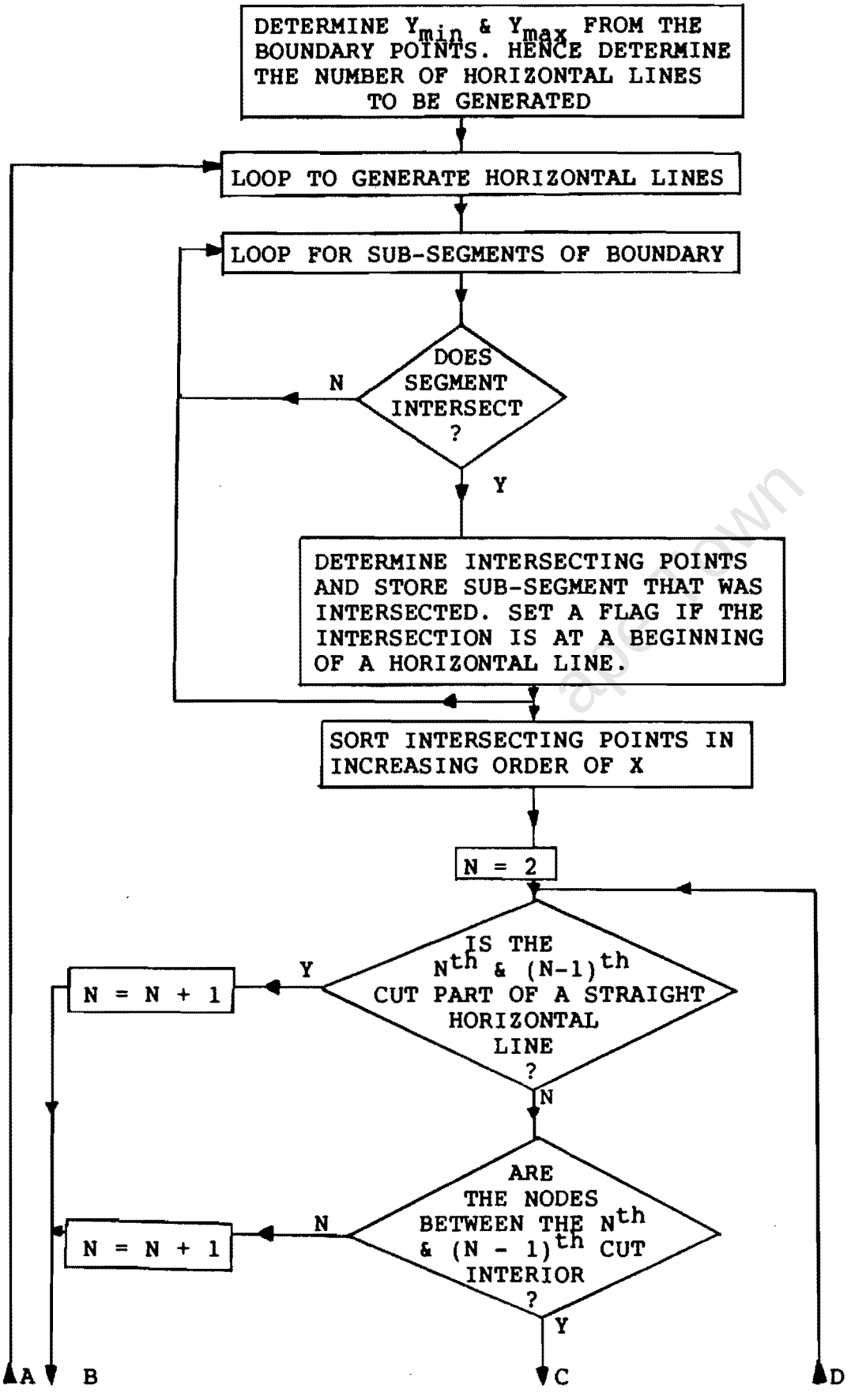


Figure 3.4 Interior node generation examples


DETERMINE Y_{min} & Y_{max} FROM THE BOUNDARY POINTS. HENCE DETERMINE THE NUMBER OF HORIZONTAL LINES TO BE GENERATED
LOOP TO GENERATE HORIZONTAL LINES
LOOP FOR SUB-SEGMENTS OF BOUNDARY
DOES SEGMENT INTERSECT ?
N
Y
DETERMINE INTERSECTING POINTS AND STORE SUB-SEGMENT THAT WAS INTERSECTED. SET A FLAG IF THE INTERSECTION IS AT A BEGINNING OF A HORIZONTAL LINE.
SORT INTERSECTING POINTS IN INCREASING ORDER OF X
N = 2
IS THE N^{th} & $(N-1)^{th}$ CUT PART OF A STRAIGHT HORIZONTAL LINE ?
Y
N = N + 1
N
ARE THE NODES BETWEEN THE N^{th} & $(N-1)^{th}$ CUT INTERIOR ?
N
N = N + 1
Y
A
B
C
D

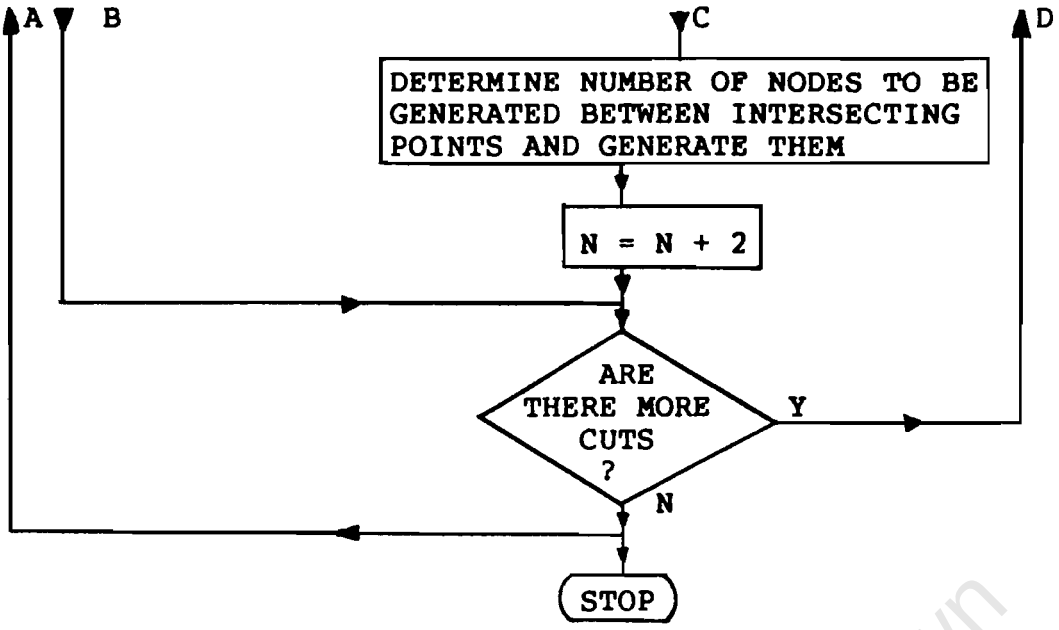


Figure 3.5 Flowchart of Intnodes module

3.1.3 Triangulisation

There are various techniques for inter-connecting nodes to form a triangulisation : Watson's algorithm computes Delaunay triangulisations [28], and a method described by A. Recuero and J.P. Gutierrez [19] uses simple and compound edges which requires a lot of input data, to form a triangulisation.

Assuming a complete nodal system is generated by the method described, a algorithm that interconnects these nodes to form a triangulisation can now be described.

Define W to be the generation front, all sub-segments can still form triangles, and V to be a set of interior nodes.

A very important point is that, at the beginning of the triangulisation the generation front is exactly equal to the collection of the domain boundaries, while the given domain boundaries remain the same, the generation front changes continuously throughout the process and has to be updated from time to time, whenever a new element is formed. By the fact that the outer contour is entered in a counter-clockwise sequence, and the inner contours in a clockwise sequence, the domain to be triangulated will always be to the left of the boundary segments.

The triangulisation is initiated by selecting the first segment of an element of W. A segment selected is called AB. The goal is to determine a node C from W union V such that C lies to the left of AB and triangle ABC is in a sense optimal - near equilateral. The algorithm chooses a node C such that it is the closest to AB i.e. minimum($AC^2 + BC^2$), this is sufficient to determine the point C, insuring the best triangulisation obtainable from the system of interior nodes and boundary nodes. The choice of node C by the above criterion is simple enough, but it may not be sufficient to ensure the best triangulisations for regions having irregular boundaries. For such cases (see Fig. 3.6), it is best to choose two nodes C_1 and C_2 closest to AB - minimum($AC^2 + BC^2$), and the vector products $\overline{ABC}_1$ and $\overline{ABC}_2$ are positive, where $\overline{ABC}$ is given by :

$$\overline{ABC} = \tfrac{1}{2}\ \overline{AB} \times \overline{AC} \qquad (3.1.4a)$$

and lets define :

$$\Gamma_{ABC} = \tfrac{1}{2}\ |\overline{AB} \times \overline{AC}| \qquad (3.1.4b)$$

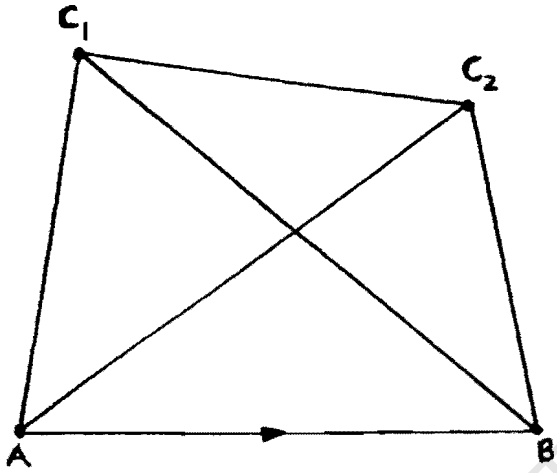


Figure 3.6 Selection between nodes C_1 and C_2

For each triangle there exists a quality factor, the bigger the quality factor the closer the triangle is to an equilateral triangle, therefore to choose between C_1 and C_2, consider Fig. 3.6, and define quality factors for ΔABC_1, ΔC_1BC_2 and ΔAC_1C_2 as α_1, β_1 and δ_1 respectively, and for ΔABC_2, ΔC_2BC_1 and ΔAC_1C_2 as α_2,β_2 and δ_2 respectively. The quality factors α_1 and α_2 are those for the triangle to be formed, the other quality factors are for the adjacent triangles. τ_1 and τ_2 are factors for the adjacent triangles.

$$\alpha_1 = \Gamma_{ABC1}/(AB^2 + BC_1{}^2 + C_1A^2) \qquad (3.1.5)$$

$$\beta_1 = \Gamma_{C1BC2}/(C_1B^2 + BC_2{}^2 + C_1C_2{}^2) \qquad (3.1.6)$$

$$\delta_1 = \Gamma_{AC1C2}/(AC_1{}^2 + C_1C_2{}^2 + C_2A^2) \qquad (3.1.7)$$

$$\tau_1 = \text{Maximum}(\beta_1,\delta_1) \qquad (3.1.8)$$

$$\alpha_2 = \Gamma_{ABC2}/(AB^2 + BC_2{}^2 + C_2A^2) \quad (3.1.9)$$

$$\beta_2 = \Gamma_{C2BC1}/(C_2B^2 + BC_1{}^2 + C_1C_2{}^2) \quad (3.1.10)$$

$$\delta_2 = \Gamma_{AC2C1}/(AC_2{}^2 + C_2C_1{}^2 + C_1A^2) \quad (3.1.11)$$

$$\tau_2 = \text{Maximum}(\beta_2, \delta_2) \quad (3.1.12)$$

The reason for considering all the triangles is that after forming a triangle, the triangles to be formed adjacent to it must not be ones of low quality. For this reason, to get optimal triangulisation, node C_1 is selected if $\alpha_1 * \tau_1 > \alpha_2 * \tau_2$ and vice versa. After a node is selected, the triangle is formed and the front W is updated in the following way :

(i) If the node selected for triangulisation is an interior node, two new segments will be entered in the generation front W, and one will be deleted from V.

(ii) If the node selected, is from the generation front, there are two possibilities - one segment can be part of W, and the other not. Then only one side is added to W. - If both segments are part of W then no new segments are added to W.

(iii) When new segments are added they must follow the same sequence as AB, eg. for case (i) new segments will be AC and CB.

(iv) The segment AB used will be deleted from the front.

Now a new AB is selected, and the process is continued until the entire domain is triangulated. See Fig. 3.7 for the flowchart of this procedure.

3.1.4 Smoothing of triangulisation

It is found that the triangulisation produced can be improved by the application of a smoothing technique. This technique is designed to perturb the triangulisation so that the elements are closer to equilateral triangles. This method was first suggested by J.C. Cavendish [5] and he found it to be simple and very effective.

This is a process where each interior node is shifted to the centre of the surrounding polygon. Let P be a set of all interior nodes and let Q be a set of all the triangles formed for a particular region. For each node P_i, say S, select triangles from Q, that contain S. In this way all the nodes connected to S can be found. These nodes, say a set T_i for i = 1,k form a polygon about S. Now the coordinates of S can be replaced by the centroid of the polygon. The centre of the polygon has coordinates X_p and Y_p given by :

$$X_p = \left[\sum_{i=1}^{k} x(T_i) \right] / k \qquad (3.1.13)$$

$$Y_p = \left[\sum_{i=1}^{k} y(T_i) \right] / k \qquad (3.1.14)$$

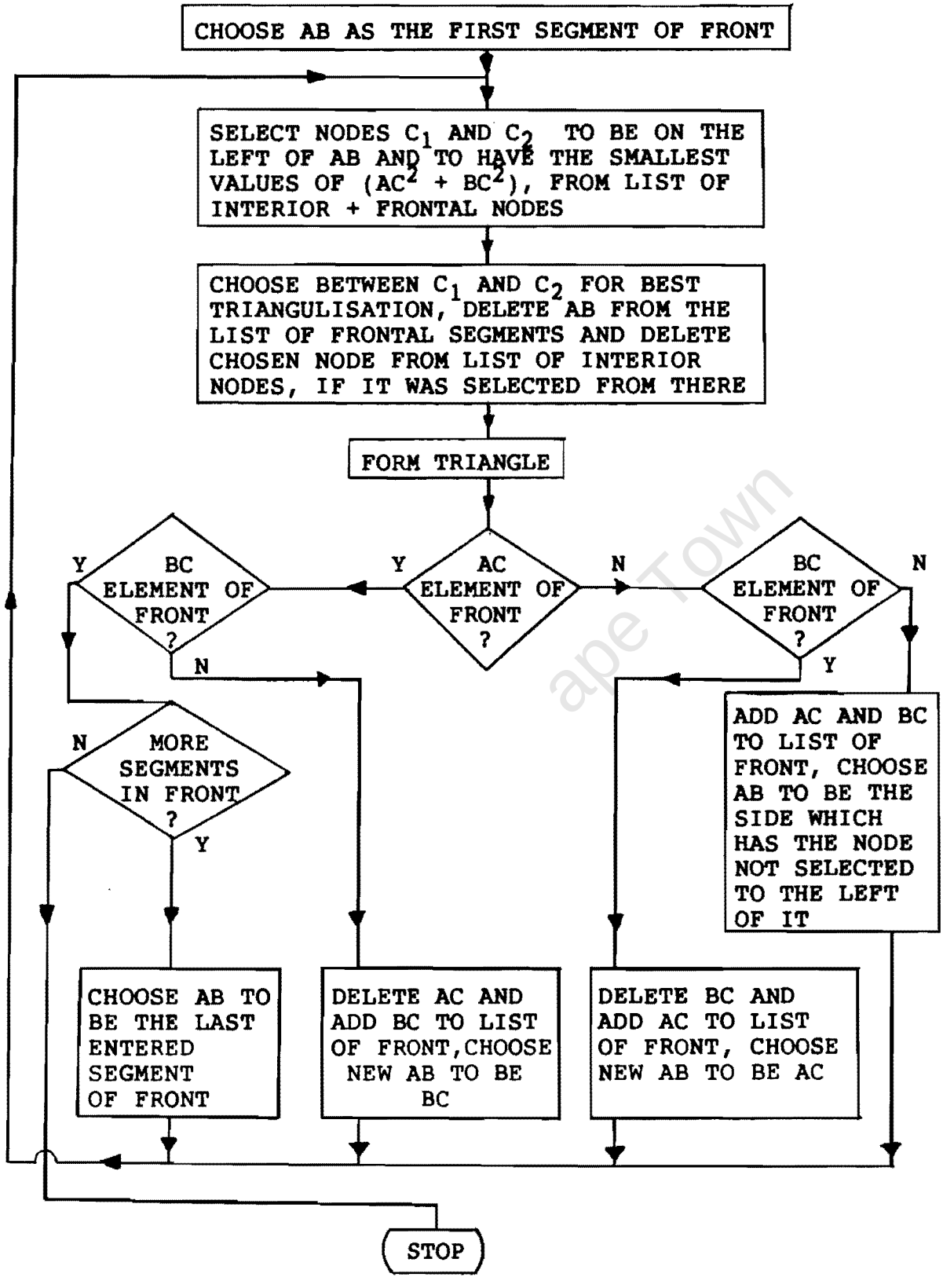


Figure 3.7 Flowchart of Triangles module

Fig. 3.8 shows a flowchart of this procedure

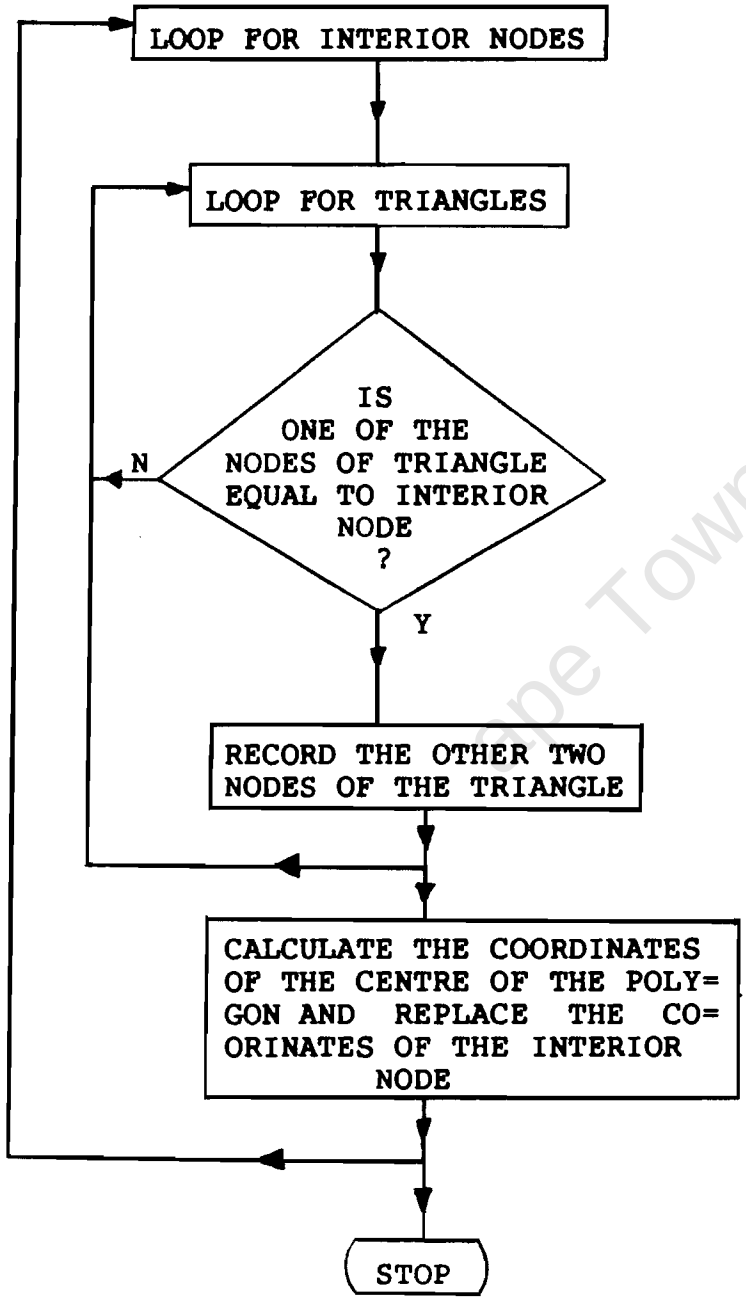
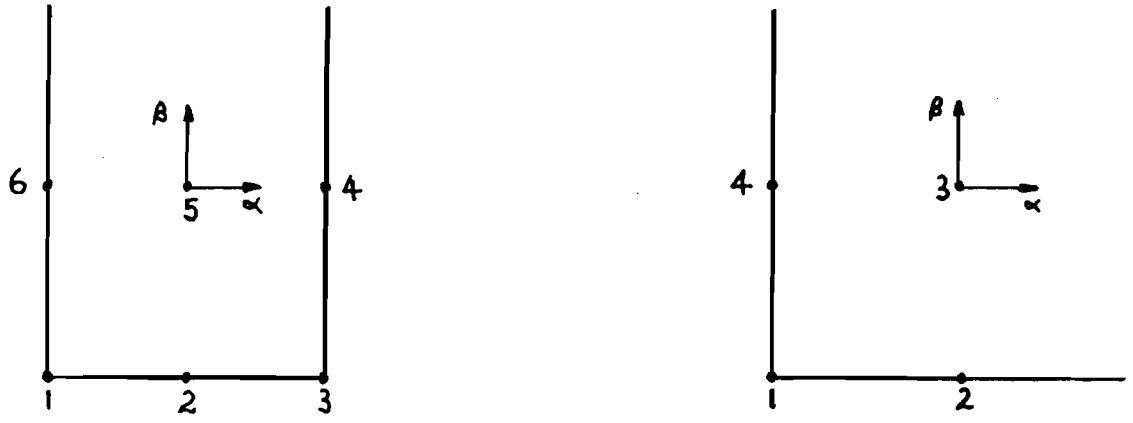


Figure 3.8 Flowchart of Smooth module

3.2 UNBOUNDED DOMAINS

Mapped infinite elements are used to solve unbounded field problems, hence a need for the mesh generator to produce infinite elements which are mapped onto master elements, arises. Two types of master elements are used : the one-way infinite Lagrangian element which is infinite in one direction; and the two-way infinite Lagrangian element which is infinite in two directions, see Fig. 3.9 (a). The elements produced by the mesh generator look like those shown in Fig. 3.9 (b). Data required by the mesh generator are :the number of infinite sub-regions, the segments that form each sub-region in a clockwise order and two segments, the first and last, of the infinite region. The algorithm used to form the infinite elements, automatically decides where two-way elements should be inserted.

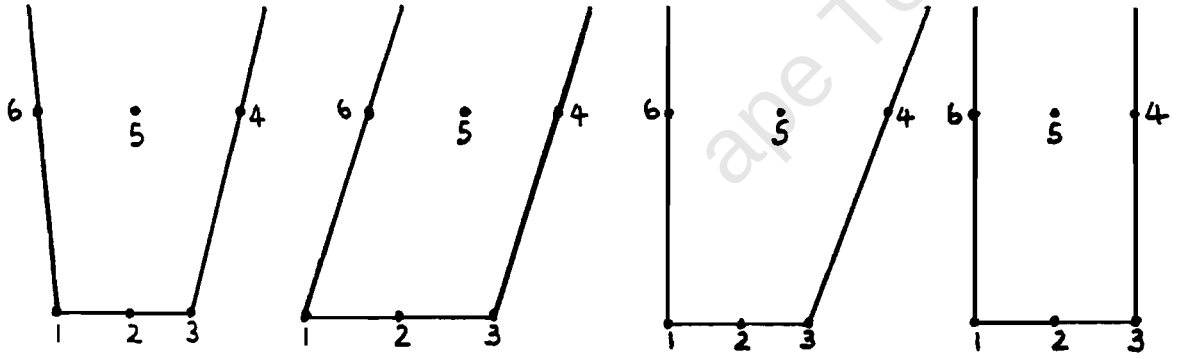
One problem encountered, was not to produce elements where the two segments that extend to infinity cross each other, see Fig. 3.9 (c). For this reason, the mesh generator operates in two parts for straight line segments :in the reverse direction, starting with the first segment, and in the forward direction. Each of these parts generate element sides either parallel to the guides, or perpendicular to the segments. Infinite regions about sectors are dealt with separately.



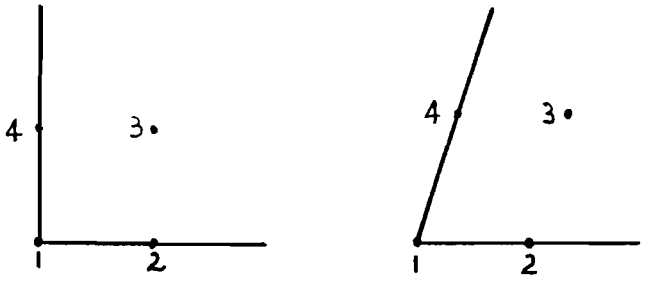
One-way element

Two-way element

Figure 3.9 (a) Master elements



One-way elements



Two-way elements

Figure 3.9 (b) Elements generated by mesh generator

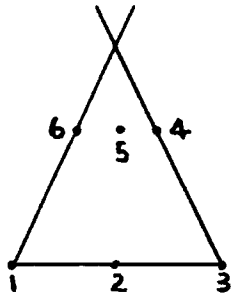


Figure 3.9 (c) Crossed over element

3.2.1 Parallel Generation of Element Sides

The element sides are generated parallel to the guides if the angle between the guide and the segment is less than 90^{o}.The coordinates of a new node can be calculated considering Fig. 3.10 (a).

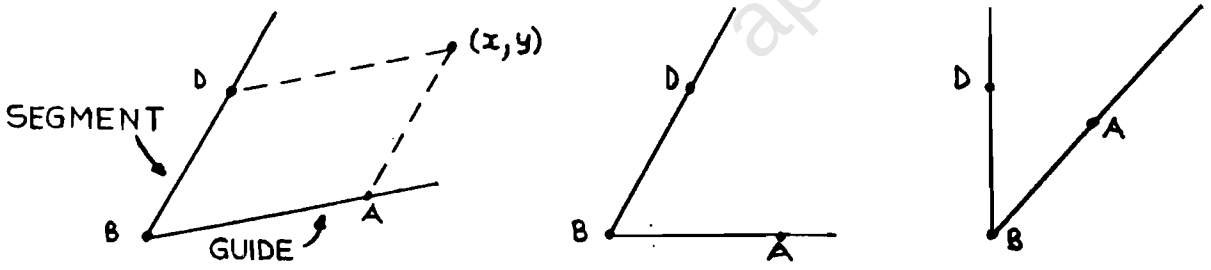


Figure 3.10 Parallel generation of element sides

The coordinates of the point (x,y) are given by the following equations :

$$x = \left[C_1 - C_2 \right] / \left[K_1 - K_2 \right] \qquad (3.2.1)$$

$$y = K_1 * \left[x \right] - C_1 \qquad (3.2.2)$$

Where

$$K_1 = \left[Y_A - Y_B \right] / \left[X_A - X_B \right] \qquad (3.2.3)$$

$$K_2 = \left[Y_B - Y_D \right] / \left[X_B - X_D \right] \qquad (3.2.4)$$

$$C_1 = K_1 * X_D - Y_D \qquad (3.2.5)$$

$$C_2 = K_2 * X_A - Y_D \qquad (3.2.6)$$

Two special cases must be considered :

(i) When $X_A = X_B$ (see Fig. 3.10 (b)), then the coordinates of x and y are given by :

$$x = X_D \qquad (3.2.7)$$

$$y = Y_A + Y_D - Y_B \qquad (3.2.8)$$

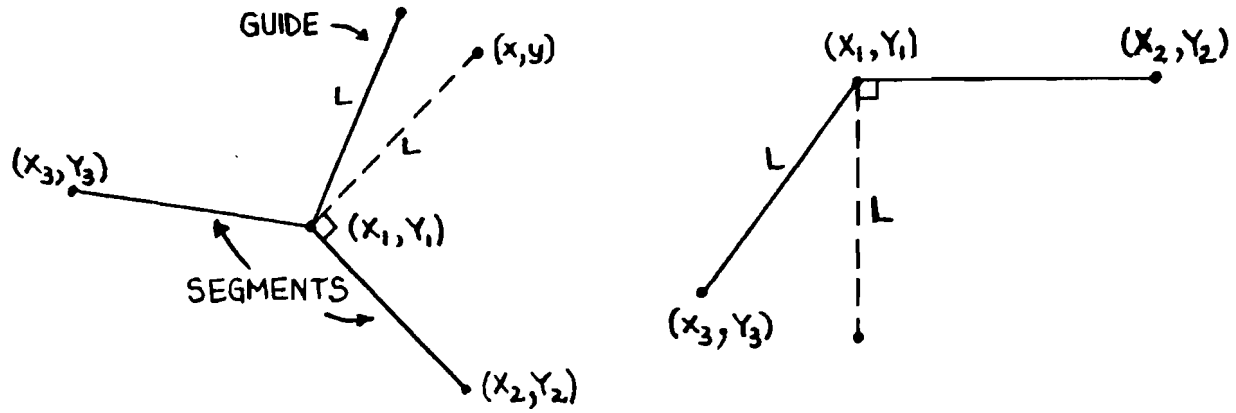
(ii) When $X_B = X_D$ (see Fig. 3.10 (c)), the coordinates of x and y are given by :

$$x = X_A \qquad (3.2.9)$$

$$y = Y_A + Y_D - Y_B \qquad (3.2.10)$$

3.2.2 Perpendicular Generation of Element Sides

The element sides are generated perpendicular to the segment if the angle between the guide and the segment is greater than or equal to 90° . The coordinates of the new node can be determined considering Fig. 3.11 (a).



(a) (b)

Figure 3.11 Perpendicular generation of element sides

The coordinates of the new node (x,y) are then given by the following equations:

$$x = \pm L / \sqrt{1 + k^2} + X_1 \qquad (3.2.11)$$

$$y = K*\left[x - X_1 \right] + Y_1 \qquad (3.2.12)$$

Where

$$L = \sqrt{(X_3 - X_1)^2 + (Y_3 - Y_1)^2} \qquad (3.2.13)$$

$$K = - \left[X_2 - X_1 \right] / \left[Y_2 - Y_1 \right] \qquad (3.2.14)$$

Note that the x coordinate has two possibilities which means that it could be the one above the segment, as shown in Fig. 3.11 (a), or the one below the segment. There is a need to check for which one of the two cases the node is outside the

bounded domain. This is done by checking that the node is to the right of the segment of generation.

A special case that must be considered is when $Y_2 = Y_1$ (see Fig. 3.11 (b), the coordinates of x and y are then given by:

$$x = X_1 \qquad (3.2.15)$$

$$y = Y_1 + L \qquad (3.2.16)$$

3.2.3 Reverse Direction for Generating infinite elements

The angle between the last segment (or the guide) and the second last segment is calculated - segments 4 and 3 in Fig. 3.12 (a). If it is less than 90° the nodes for that particular segment is obtained, and element sides are generated parallel to the guide, as shown in Section 3.2.1, forming one-way infinite elements. The last element side generated forms the guide for the next segment. The process is repeated until :

(i) The angle between the guide and the segment is ≥ 90°, the segment number and the angle is recorded, and the process jumps to the forward direction of generating infinite elements. The segment number and the angle which was recorded, will be used to prevent the forward and reverse processes from overlapping.

(ii) The first segment or guide of the unbounded region is reached, in which case the unbounded domain is complete and the process is terminated.

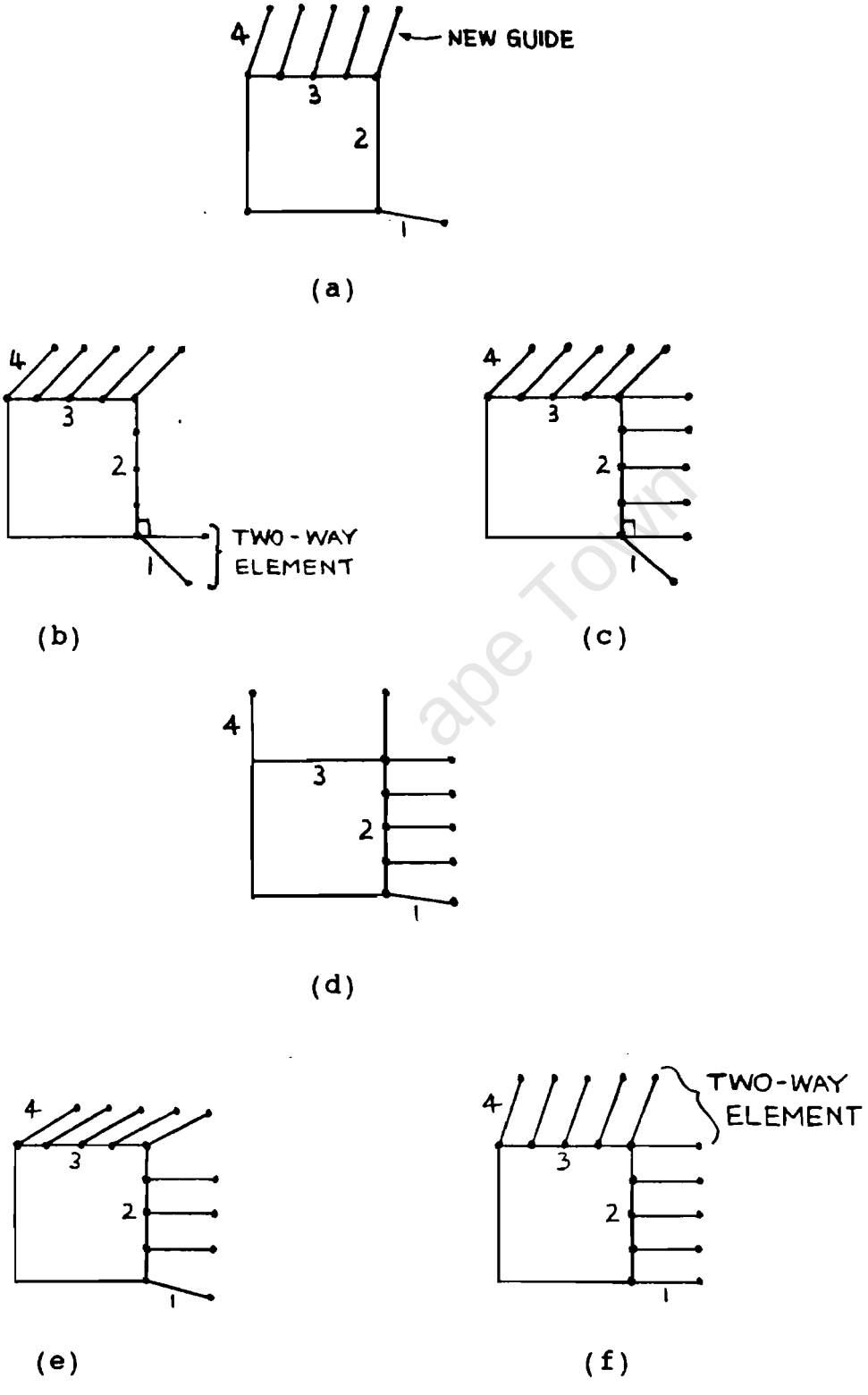


Figure 3.12 Infinite element generation examples

3.2.4 Forward direction for generating infinite elements

The angle between the first segment (or the guide) and the second segment is calculated(see segments 1 and 2 in Fig.3.12 (b)). The nodes of the segment are then obtained in the correct sequence, and the element sides are generated in the following manner :

(i) If the angle is < 90° then element sides are generated parallel to the guide forming one-way infinite elements. The last element side is used as the guide for the next segment.

(ii) If the angle is ≥ 90° but > 150° then an infinite two-way element is generated by generating an element side perpendicular to the segment (see segments 1 and 2 in Fig. 3.12 (b). This element side is now used as the guide for the segment, (for segment 2 in Fig. 3.12 (b)), nodes for the segment are now obtained in sequence and element sides are generated perpendicular to the segment (see Fig. 3.12 (c)).

(iii) If the angle is ≥ 90° but < 150°, the element sides are generated perpendicular to the segment (see segments 1 and 2 in Fig. 3.12 (d)) and the last element side forms the guide for the next segment.

The entire process is repeated for the next segment and guide. Each time a new segment is tackled, a test is done to see if a two-way element is needed. A test is inserted to see where the forward and reverse processes meet, using the segment number stored in the reverse process. A test is also done to determine whether a two-way element is needed where the two processes meet, using the same criterion as in (ii) of this section. See Fig. 3.12 (e) no two-way element is needed, whereas in Fig. 3.12 (f) a two-way element is needed.

Fig. 3.13 and Fig. 3.14 show flowcharts of the reverse and forward processes of infinite element generation respectively.

3.2.5 Infinite Element Generation along sectors

The generation of infinite elements along sectors cannot be done by parallel or perpendicular generation and is therefore done separately. The generation for any sector is treated as a single sub-region. The guides for such a sub-region has to be radial, see Fig. 3.15. The generation of infinite elements is also done radially, using only one-way elements. The sweep angle of the sector is determined as well as the number of nodes along the segment, in order to determine the number of elements that are to be generated. The sweep angle is then sub-divided and the one-way elements are then formed. An example is shown in Fig. 3.15.

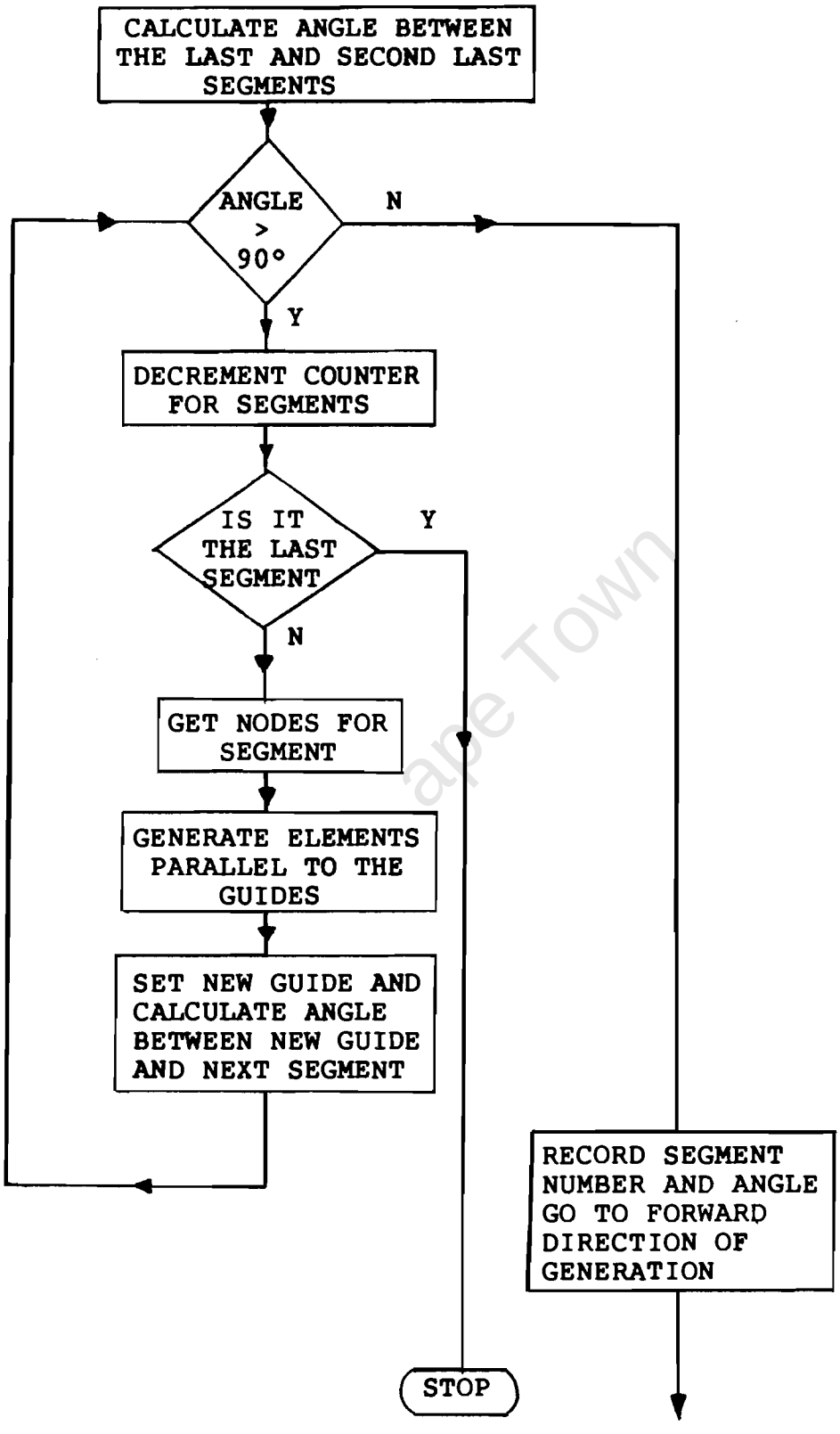


Figure 3.13 Reverse direction of infinite element generation

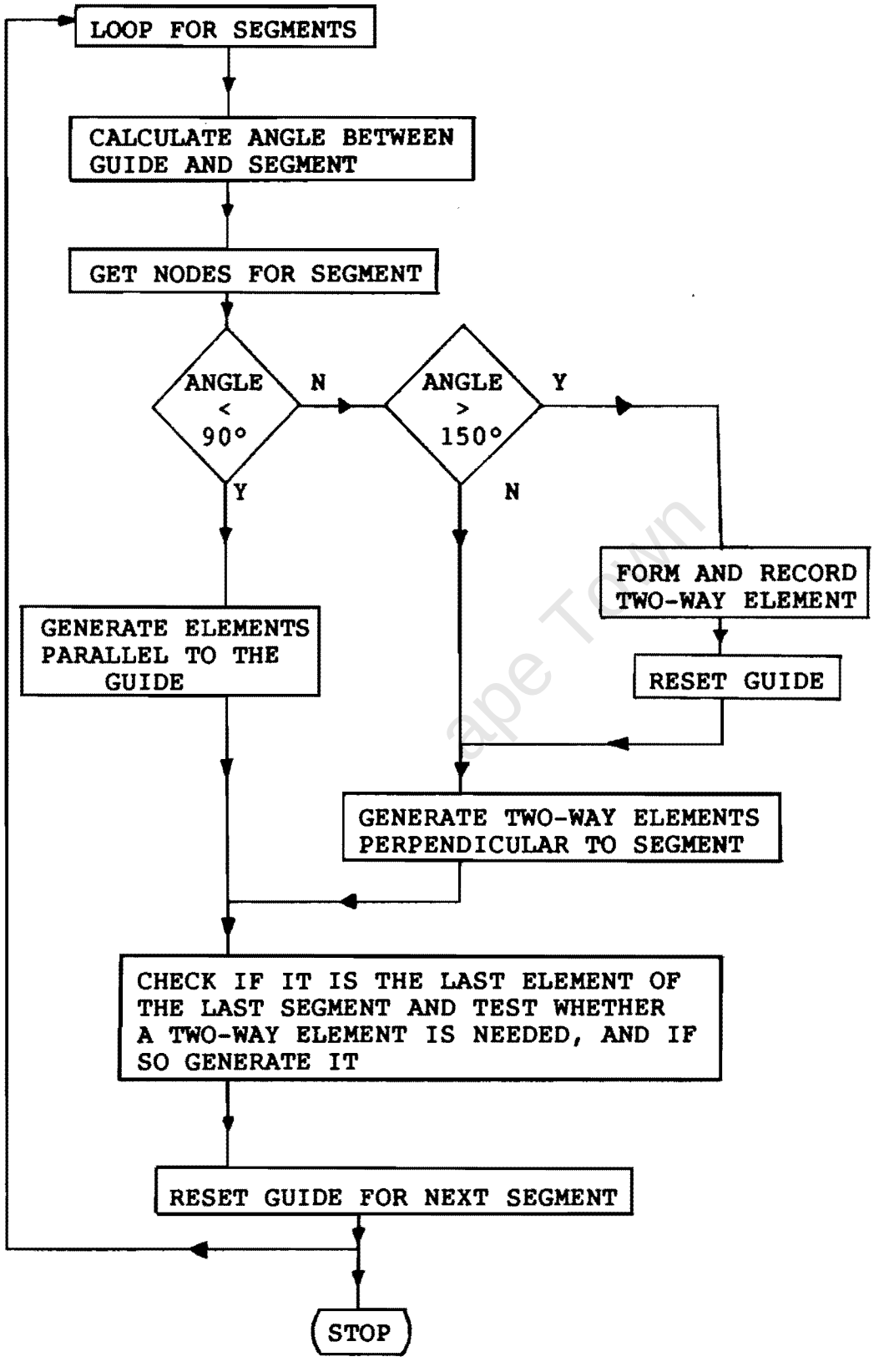


Figure 3.14 Forward direction of infinite element generation

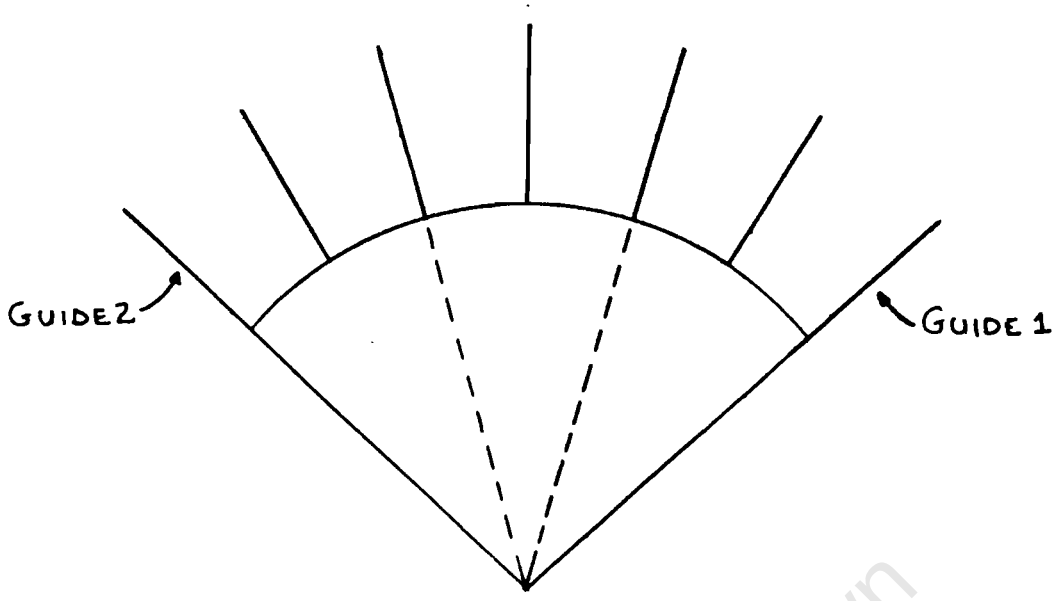


Figure 3.15 Generating infinite elements along sectors

3.3 SYMMETRICAL PROBLEMS

When symmetrical domains were entered, problems arose in that the triangulisation of the mesh was not symmetrical which resulted in errors creeping into the results, forming asymmetrical solutions. For this reason it was necessary to generate a symmetrical mesh.

This is achieved by entering the one half of the domain and the axis of symmetry, either x = C or y = C. The mesh generator then produces a symmetrical mesh in the following way :

(i) Forming the symmetrical outer boundary segments.

(ii) Forming a symmetrical node for every node generated.

(iii) Forming a symmetrical triangle whenever a triangle is formed.

(iv) In this way an exact image of the domain entered, is produced on the other side of the axis of symmetry.

CHAPTER 4

THE EQUATION SOLVER AND POST-PROCESSOR

In this Chapter the following will be dealt with :

(i) Solution techniques for the system of linear equations.

(ii) A process whereby data is put into a meaningful form, called post-processing.

4.1 THE EQUATION SOLVER

The implementation of the finite element technique for solving differential equations is done by considering an equivalent problem for which a finite set of simultaneous linear equations can be solved. There are basically two types of solution techniques:

(i) Direct methods

(ii) Iterative methods

Direct methods introduce errors due to rounding off, whereas iterative methods have errors only dependent on the number of iterations done. However, due to the limitation in time, only direct methods are considered in this chapter.

The solution technique used initially is the Gauss Jordan algorithm. This algorithm is outlined in detail by J. Stoer

and R. Bulirsch [29]. This technique worked very well but was time consuming. There was therefore a need to speed up the solution process. Considering the properties of the matrices produced, large sparse and symmetric, a method had to be selected to economise on the number of operations in the solution process thereby speeding up the process. This could be done in two ways:

(i) Carrying out only half the operations because the matrix is symmetric.

(ii) Taking advantage of the large quantity of zero elements in the matrix.

Various techniques exist which use one or both of the above strategies. A review of such techniques is done by C. Meyer [17]. J.A. George [11] discusses different techniques which economise on the sparsity of the matrices. Most techniques economising on sparsity of matrices involve some form of node renumbering to give the matrix banded properties. Banded matrices may be solved by one of two methods :

(i) Choleski factorisation, see H.R. Schwarz et al [27], which can only be used if the stiffness matrix is symmetric and positive definite.

(ii) Gaussian elimination for which the stiffness matrix only has to be symmetric.

Numerous renumbering techniques exist of which the Cuthill and Mc Kee [7] is perhaps the most commonly used. A new method developed by N.E. Gibbs, W.G. Poole and P.K. Stockmeyer [12], however, improves on the Cuthill and Mc Kee method. This method was implemented in the equation solver.

In order to understand the renumbering algorithm a graph in matrix methods has to be defined. A graph is a plot to indicate all the non-zero elements in a matrix.

4.1.1 Basic Concepts of Graph Theory

A graph G = (V,E) consists of a finite set of vertices or nodes, V(G), and edges E(G). Given a matrix A with entries a_{ij}, a graph G = (V,E) can be defined where V has vertices $\{v_1,.....,v_n\}$ and edges $\{v_i,v_j\} \in E$ if a_{ij} is non-zero and i not equal to j. If $\{v_1,v_2\} \in E$ then v_1 and v_2 are said to be adjacent. The degree of a vertex is the number of vertices adjacent to it.

If G has n vertices, then a one-to-one map, f of V(G) onto a set {1,2,...,n} is called a numbering of G. For each numbering f, a bandwidth of G, $\beta_f(G)$ can be defined:

$$\beta(G) = \max\{\ |f(v_1) - f(v_2)| : \{v_1,v_2\} \in E(G)\ \} \qquad (4.1.1)$$

The minimum of $\beta_f(G)$ over all the numberings of G is called the bandwidth of G and denoted by $\beta(G)$.

A partitioning of V(G) into levels $L_1,....,L_k$ is called a level structure, L(G). Formation of a level structure can be done as follows:

(i) $L_1 = \{v\}$, which is the start vertex.

(ii) L_i for $i > 1$, L_i is the set to all the vertices adjacent to vertices of L_{i-1}, which are not yet assigned to a level.

For each vertex $v \in V(G)$ there exists a level structure $L_v(G)$ called a level structure rooted at v. For every level of a level structure, a level width, $w(L_i)$ can be defined, which is the number of vertices in the level i. The width of the level structure is defined as:

$$w(L) = \max\{w(L_i)\} \qquad (4.1.2)$$

The depth of a level structure is equal to the number of levels in the level structure.

4.1.2 The Renumbering Algorithm

The algorithm can be divided into three parts.

4.1.2 (a) Choosing the start vertex

A path through vertices, from the one vertex to another is called a diameter - eg. $\{v_o,v_1\},\{v_2,v_3\}, \ldots ,\{v_{t-1},v_t\}$ is the path from v_o to v_t. Ideally start vertices got to be at

the end points of a diameter which is the biggest - i.e. nodes that are a maximal distance apart. The algorithm to obtain this is as follows :

(i) Choose an arbitrary vertex of minimal degree and call it v.

(ii) Generate a level structure L_v rooted at v and let S be a set of vertices in the last level of L_v.

(iii) Generate level structures rooted at $s \in S$, selected in order of increasing degree. If for some s the depth of L_s is greater than the depth of L_v, then set v to s and return to step (ii).

(iv) Let u be a vertex of S whose level width is the smallest.

Now there are two vertices u and v which are a maximal distance apart. The level structures associated with each is L_u and L_v.

4.1.2 (b) Minimising level width

This algorithm combines level structures L_u and L_v into a new level structure whose width is usually less than that of L_u and L_v.

(i) Let levels in L_u an L_v be as follows :

$$L_u = \{L_1, L_2, \ldots L_k\} \qquad (4.1.3)$$

$$L_v = \{M_1, M_2, \ldots M_k\} \qquad (4.1.4)$$

For each vertex w of G the ordered pair (i,j) called associated level pair is defined, where i is the level of w in L_v and k+1-j is the level of w in L_u.

(ii) Assign the vertices of G to levels in a new level structure $L = \{N_1, N_2, \ldots, N_k\}$ as follows :

1. If for some w a vertex of G , the associated level pair, i = j then assign w to level i.
2. The graph G now consists of one or more disjoint connected components $C_1, C_2, \ldots C_t$ ordered so that $|V(C_1)| \geq |V(C_2)| \geq \ldots \geq |V(C_t)|$.
3. For each of the connected components do the following :

   (a) Compute a vector $\{n_1, \ldots, n_k\}$ where n_i is the number of vertices in level N_i.

   (b) Compute vectors $\{h_1, \ldots, h_k\}$ and $\{l_1, \ldots, l_k\}$ where h_i is the number of vertices that would be in N_i if the first element of the associated level pairs were used. l_i is the number of vertices that would be in N_i if the second element of the associated level pairs is used.

   (c) Find $h_o = \max_i \{h_i : h_i - n_i > 0\}$ and $l_o = \max_i \{l_i : h_i - n_i > 0\}$. Now to find out which of the elements of the associated level pairs to use :

(i) If $h_o < l_o$ choose the first element of the associated level pair.

(ii) If $l_o < h_o$ choose the second element of the associated level pair.

(iii) If $h_o = l_o$ use elements of level pairs which arise from a level structure of smaller width. If the widths are equal use the first elements.

Now a single level structure is formed which must be mapped onto a numbering.

4.1.2 (c) Numbering

The numbering scheme assigns consecutive positive integers to the vertices of G, level by level - starting at level 1. The numbering algorithm can be divided into four parts :

(i) If the degree of u is less than the degree of v then interchange u and v and reverse the level structure by setting N_i to N_{k+1-i}.

(ii) Assign consecutive positive integers to the vertices of level N_1 in the following order :

(a) Assign number one to v.

(b) Let w be the lowest numbered vertex in L_1 which has vertices of level N_1 adjacent to it, then number these vertices in order of increasing degree. Repeat this step until all the vertices of N_1 adjacent to the numbered vertices are numbered.

(c) If any unnumbered vertices remain in this level, number the one of minimal degree, then go to step (b).

(iii) Number the vertices of level N_i = 2,...,k as follows :

(a) Let w be the lowest numbered vertex of level N_{i-1}, that has unnumbered vertices of level N_i adjacent to it. Number the vertices of N_i adjacent to w in order of increasing degree. Repeat this step until all the vertices of N_i adjacent to vertices of N_{i-1} are numbered.

(b) Repeat steps (ii) (b) and (c) replacing 1 with i.

(iv) The numbering is reversed by setting i to n-i+1 for i = 1,....,n, if either of the following conditions hold.

(a) If step (i) interchanged u and v and if in 4.1.2 (b) the second elements of the associated level pairs were chosen for C_1.

(b) If step (i) did not interchange u and v and in 4.1.2 (b) the first elements of the associated level pairs were chosen for C_1.

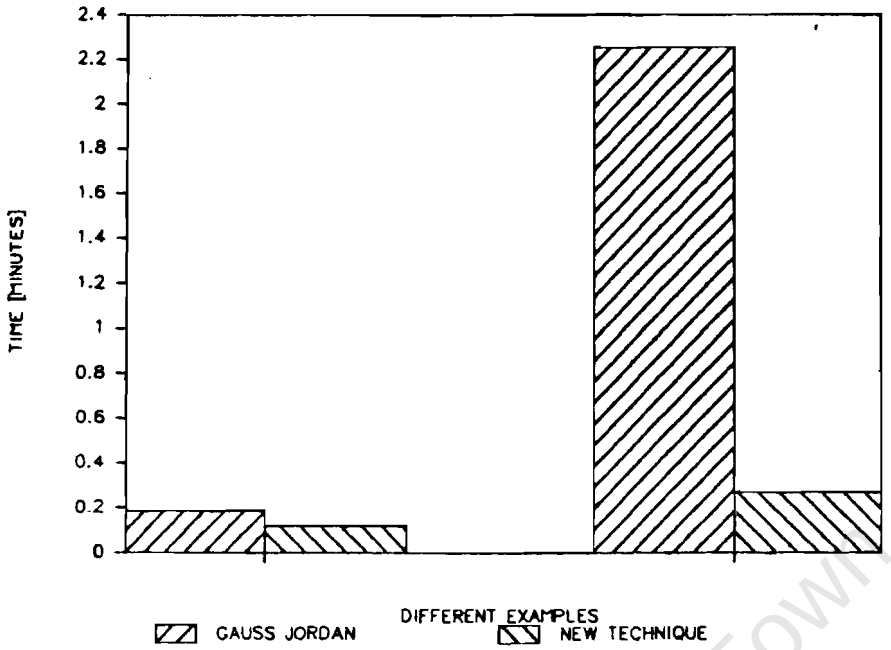
4.2 THE SOLUTION TECHNIQUE

The stiffness matrices that were produced for finite elements were symmetric positive definite. However, when infinite elements were introduced, the matrices lost their properties of positive definiteness due to the rounding off

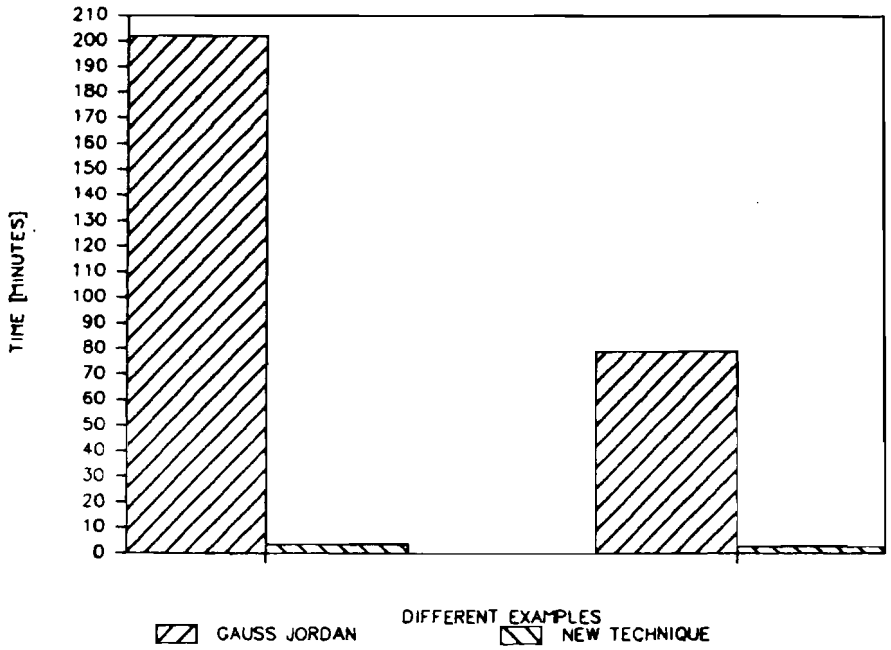
errors on the computer. For this reason the Gaussian elimination solution technique was used. This technique is outlined in detail by J. Stoer and Bulirsch [29]. As a result of the banded characteristics of the matrices, a few modifications were done to the technique to economise on the number of arithmetic operations.

Pointers were set for each of the rows of the matrix to indicate the beginning and end of the non-zero elements. These were then used to set limits for the "do loops" in the elimination process, to economise on the number of arithmetic operations, consequently reducing the execution time. A few test runs were done using both the old technique, Gauss Jordan and the new technique. Fig. 4.1 shows the comparison of execution times for four examples. Fig. 4.1 (a) shows examples of 93 (left) and 226 (right) nodes. Fig. 4.1 (b) shows examples of 749 (left) and 774 (right) nodes. For a number of nodes less than 100 there is not much improvement, however for a number nodes greater than 200 the improvement is greater than 85 % . The new equation solver therefore works efficiently.

The equation solver outputs data in the form of nodal potentials. As the nodes are placed at arbitrary positions, there is a need for a process which enables the user to determine potentials at regions of interest. This process is called post-processing.



(a)



(b)

Figure 4.1 Execution time comparisons of old and new techniques

4.3 THE POST-PROCESSOR

Ideally, a post-processor should do a field plot or an equipotential plot. This is however not done in this program and can be considered as a possible improvement.

The post-processor implemented is a building block for forming equipotential plots. The post-processor enables the user to obtain a potential distribution along any horizontal or vertical line, i.e. y = c or x = c, where c is a constant specified by the user. The process by which this is done is best illustrated by means of the flowchart in Fig. 4.2.

The post-processor can be used to obtain a grid of potentials over the entire domain. The post-processor also gives the x and y components of the field. This was achieved by using the following equations :

(i) For the vector potential, the flux components are derived by using:

$$B_x = \frac{\delta \phi}{\delta y} \qquad (4.2.1)$$

$$B_y = -\frac{\delta \phi}{\delta x} \qquad (4.2.2)$$

(ii) For the scalar potential, the flux components are derived by using :

$$D_x = -\frac{\delta \phi}{\delta x} \qquad (4.2.3)$$

$$D_y = -\frac{\delta \phi}{\delta y} \qquad (4.2.4)$$

Where ø is given by equation (2.2.16) in Chapter 2.

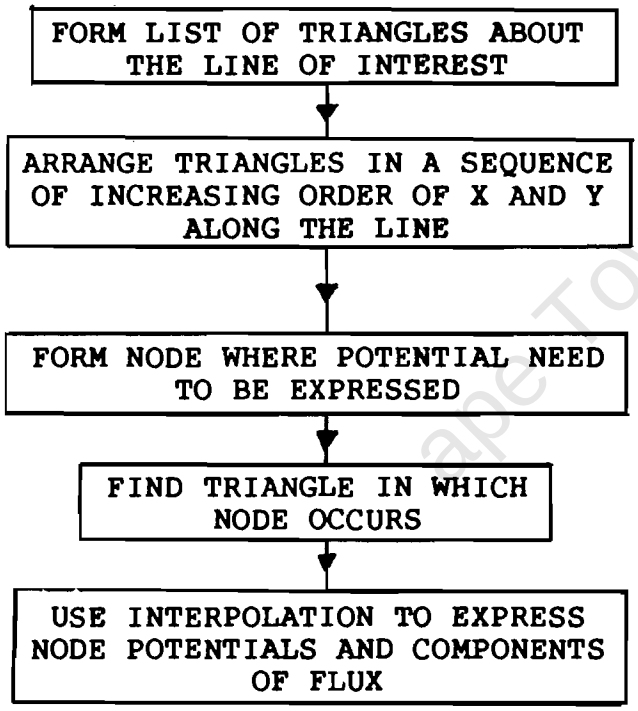


Figure 4.2 Flowchart of the post-processor

The post-processor was used on many examples, of which 3 is included in Chapter 5, and proved to be a satisfying and efficient way for the evaluation of the overall program.

CHAPTER 5

APPLICATION OF THE PROGRAM

Ideally, the domain contours should be entered by means of a light pen or a mouse, but due to the limitation of graphics facilities the contours are entered into a data file from which the computer reads the information, and forms the domain in memory.

The application of the program is straight forward and is dealt with in the following way :

(i) The creation of a data file is discussed.

(ii) Numerous examples were used for testing the program, of which only 3 will be included in this chapter, namely :

  (a) Straight machine slot
  (b) Slanted machine slot
  (c) Linear machine

5.1 DATA FILE LAYOUT

The basic parts of the data file is given in Section 5.1.1 and for details on each part see the corresponding number in Section 5.1.2.

5.1.1 Basic Parts of the Data File

The data file can be divided into the following sections :

(1) Domain type - bounded or unbounded and symmetry conditions.

(2) Number of input nodes and their coordinates.

(3) Number of input segments and details of segments.

(4) Number of outer segments and the boundary conditions for each.

(5) Number of sub-regions.

(6) Different media.

(7) Material properties for each region, mesh size and segments that form each region.

(8) If region unbounded - number of infinite regions, material properties for each and order of numerical integration.

(9) If domain is symmetrical - material properties and boundary conditions.

(10) Output of data.

5.1.2 Detailed Layout of Data File

(1) Enter - 1 if domain is unbounded
- 2 if domain is bounded.

Enter - 0 for no symmetry
- 1 for symmetry about y = C
- 2 for symmetry about x = C.

Enter C if symmetry exists.

(2) Enter number of input nodes.

Enter coordinates of nodes - x,y below each other.

(3) Enter number of segments.

Enter for each segment :

(i) Start and end node numbers.

(ii) Type of segment 0 - Straight line

1 - Circle or sector

2 - Manually.

(iii) Enter mesh size if type = 0

Enter mesh size, Xcenter,Ycenter if type = 1

Enter number of nodes and their coordinates below each other if type = 2.

(4) Enter number of outer segments.

Enter for each outer segment :

(i) Segment number,type of condition,distribution.

(a) type of condition 0 - Dirichlet

1 - Neuman

3 - segment truncates infinite region.

(b) Distribution 0 - constant

1 - V_o SIN(Aπx + Bπy + C)

2 - V_o COS(Aπx + Bπy + C).

(ii) Enter Constant if distribution = 0

Enter A,B,C,V_o if distribution = 1 or 2.

(5) Enter number of sub-regions.

(6) Enter 1 - if problem has different media

2 - if problem has one medium.

(7) For each region enter below each other :

(i) source for each region.

(ii) (if problem has different media) material constant i.e. μ and ϵ.

For each region enter below each other :

(i) mesh size.

(ii) segments that form each region - If it is an outer contour enter segments in counter-clockwise order, else enter segments in a clockwise order. Segments should be entered below each other and after all segments for a region is entered place 0 below last segment number.

(8) Enter only if domain is unbounded.

Enter number of unbounded regions.

Enter 1 - If unbounded regions has different media

2 - If unbounded regions has one medium.

For each region enter:

(i) Number of segments that form infinite region.

(ii) Source for infinite region.

If there are different media enter material properties.

(iii) Enter segments that form infinite region below each other.

Enter order of numerical integration.

(9) Enter only if problem is symmetrical:

Enter 0 - for even boundary conditions

1 - for odd boundary conditions.

Enter 1 - if sources are different to that of symmetrical bounded region

2 - if sources are the same.

Enter Sources for symmetrical bounded regions in same sequence as the other half.

Enter 1 - if media of symmetrical bounded half is different

2 - if media of symmetrical bounded half is the same.

Enter media of symmetrical regions in the same sequence as the other half if Media are different.

Enter 1 - if sources are different to that of symmetrical unbounded region

2 - if sources are the same.

Enter Sources for symmetrical unbounded regions in same sequence as the other half.

Enter 1 - if media of symmetrical unbounded half is different

2 - if media of symmetrical unbounded half is the same.

Enter media of symmetrical regions in the same sequence as the other half if media are different.

(10) Enter 1 - for solution of vector potential

2 - for solution of scalar potential.

Enter 1 - for potentials along y = C

2 - for potentials along x = C.

Enter C.

Enter steps at which potentials are desired.

Enter normalisation constants as follows for :

x or y, Potential, Flux.

Enter 1 - if another line is desired and start from second step of (10)

2 - if no more lines are desired.

Listing of data files, for examples 1 and 2 done below, is given in Appendix C.

5.2 SELECTED EXAMPLES

5.2.1 Straight Machine Slot

The dimensions and boundary conditions of the problem tackled is shown in Fig. 5.1 (a) and the mesh generated is shown in Fig. 5.1 (b).

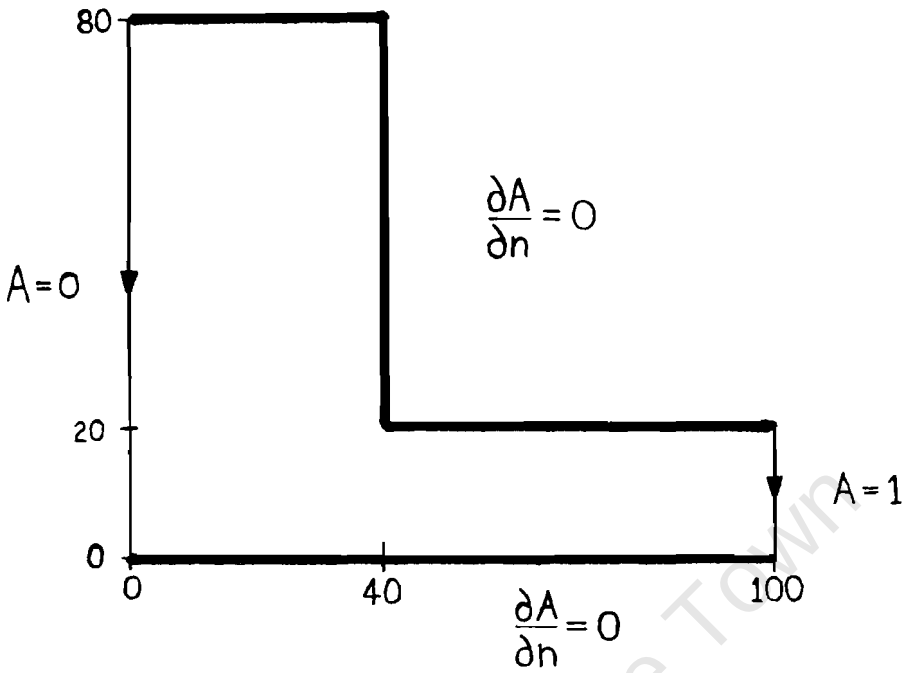
For this problem the potential distribution is compared to the analytical solution in Fig. 5.2. and the components of flux are compared to the analytical solution in Fig. 5.3.

The potentials compare very well with those of the analytical solution, however the components of flux has a significant error at the corner of the slot, this is because no mesh refinement is used at this corner.

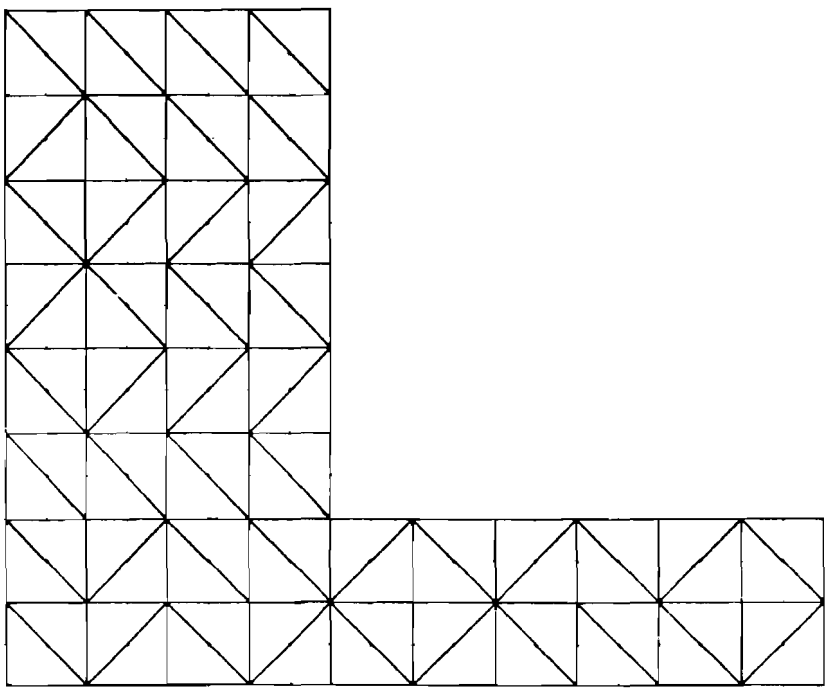
5.2.2 Slanted Machine Slot

This is an unbounded field problem and the configuration is shown in Fig. 5.4 (a). and the mesh generated is shown in Fig. 5.4 (b).

For this problem only the components of flux are compared to the analytical solutions, and are shown in Fig.5.5. The flux components calculated by finite element method compare very well with those calculated by the analytical method.



(a)



(b)

Figure 5.1 Straight machine slot configuration, boundary conditions and mesh generated.

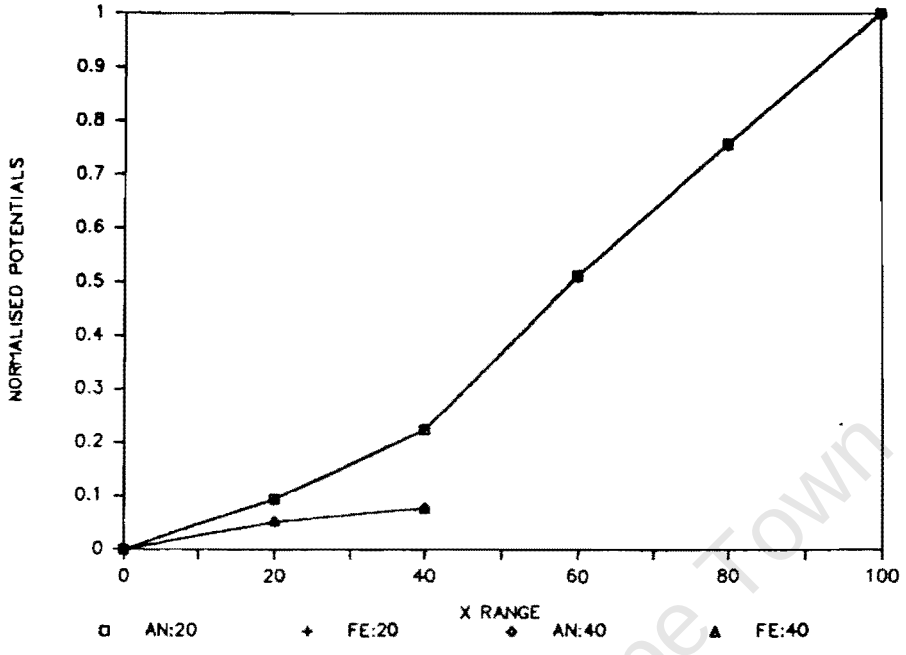


Figure 5.2 Straight machine slot potential distribution at two levels.

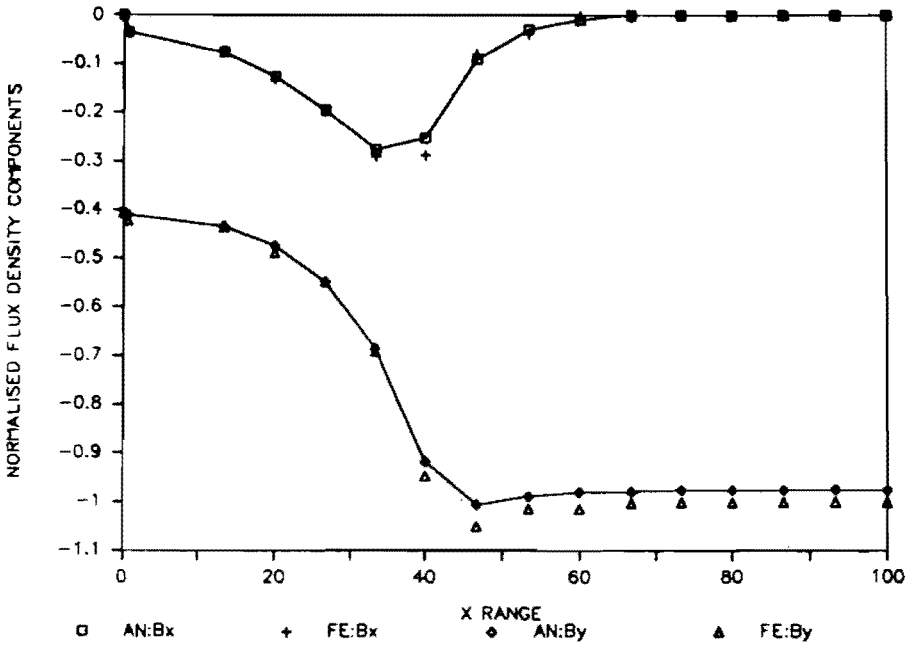
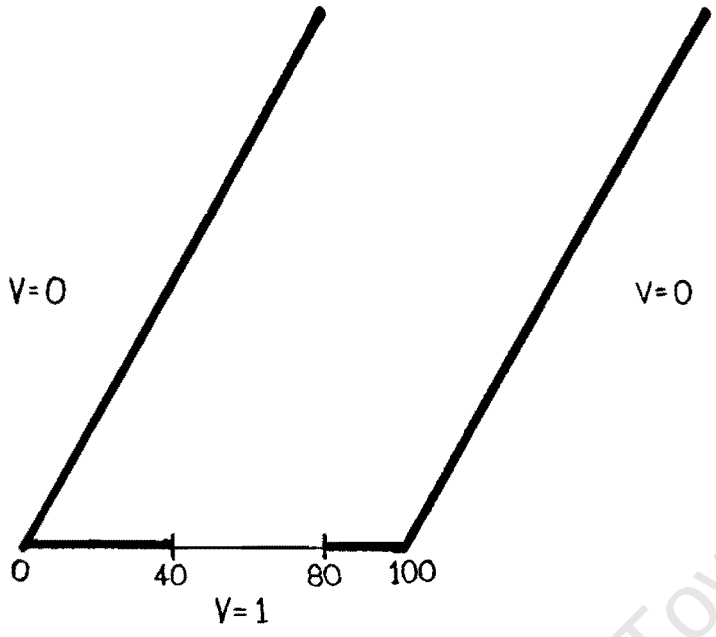
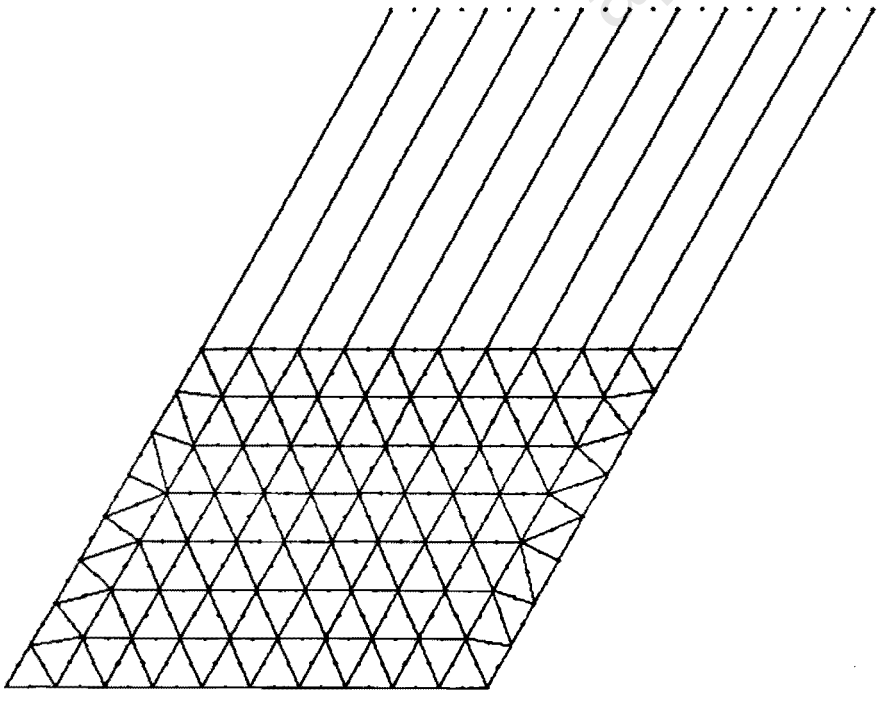


Figure 5.3 Straight machine slot flux density distribution.



(a)



(b)

Figure 5.4 Slanted machine slot configuration, boundary conditions and mesh generated.

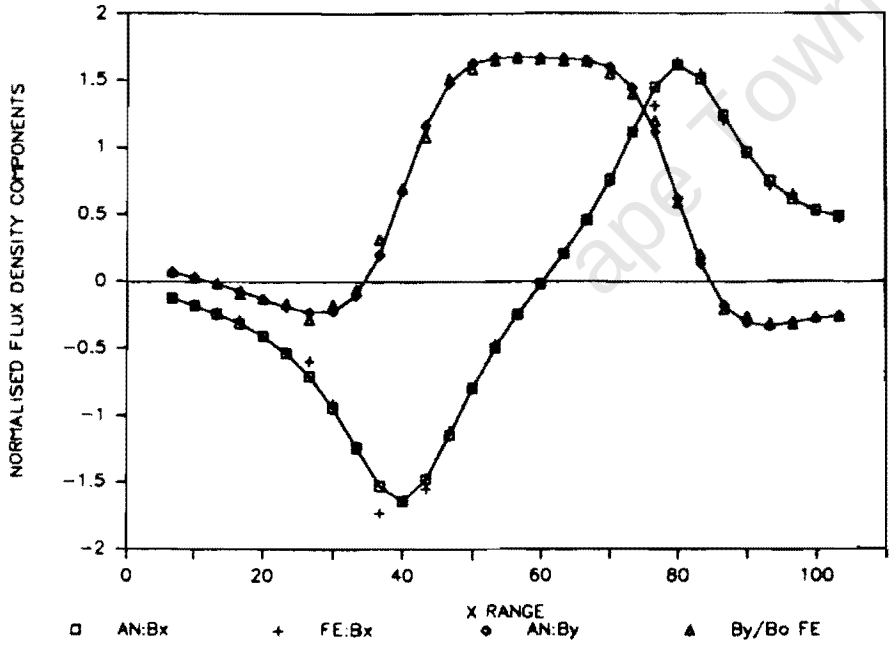
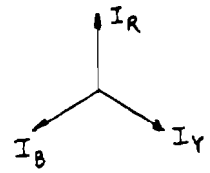


Figure 5.5 Slanted machine slot flux density distribution.

5.2.3 Linear Machine

Fig. 5.6 (a) shows the configuration of the problem and Fig.5.6 (b) shows the mesh generated. For this problem flux components are plotted for 3 different current density distributions. They are as follows :

(A)

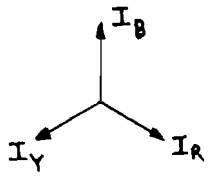


$I_R = J$

$I_Y = -\frac{1}{2} J$

$I_B = -\frac{1}{2} J$

(B)

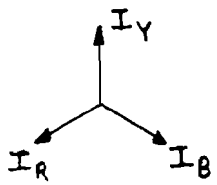


$I_R = -\frac{1}{2} J$

$I_Y = -\frac{1}{2} J$

$I_B = J$

(C)

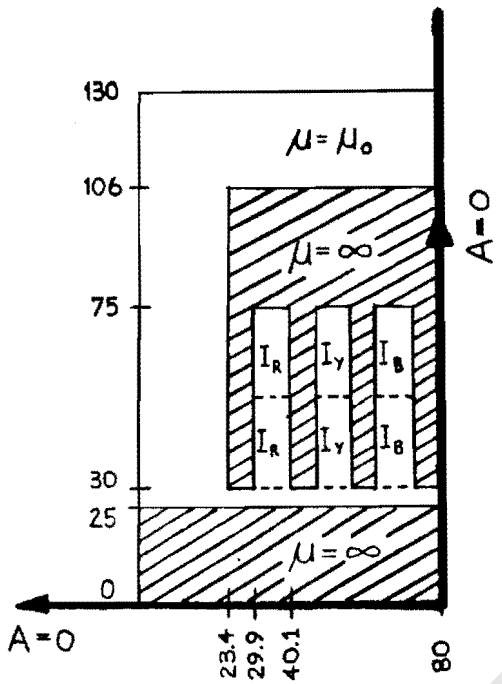


$I_R = -\frac{1}{2} J$

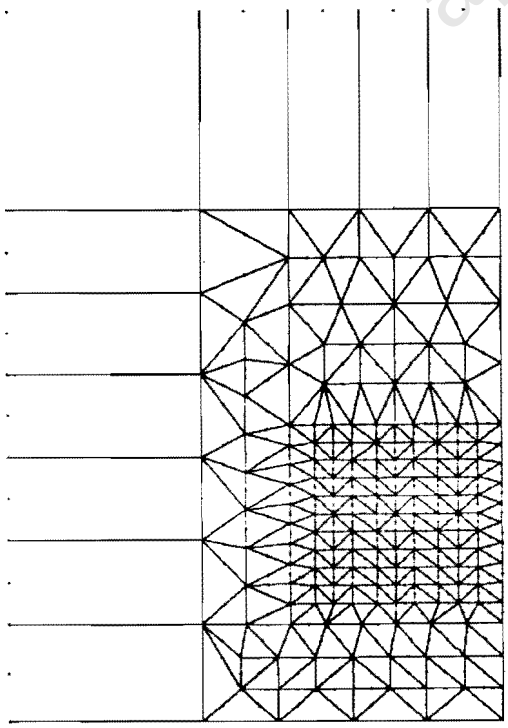
$I_Y = J$

$I_B = -\frac{1}{2} J$

The graphs for current density distributions (A), (B) and (C) are plotted in Fig. 5.7 (a), (b) and (c) respectively and could be qualitively compared to results known from a book written by J.F. Gieras [13].

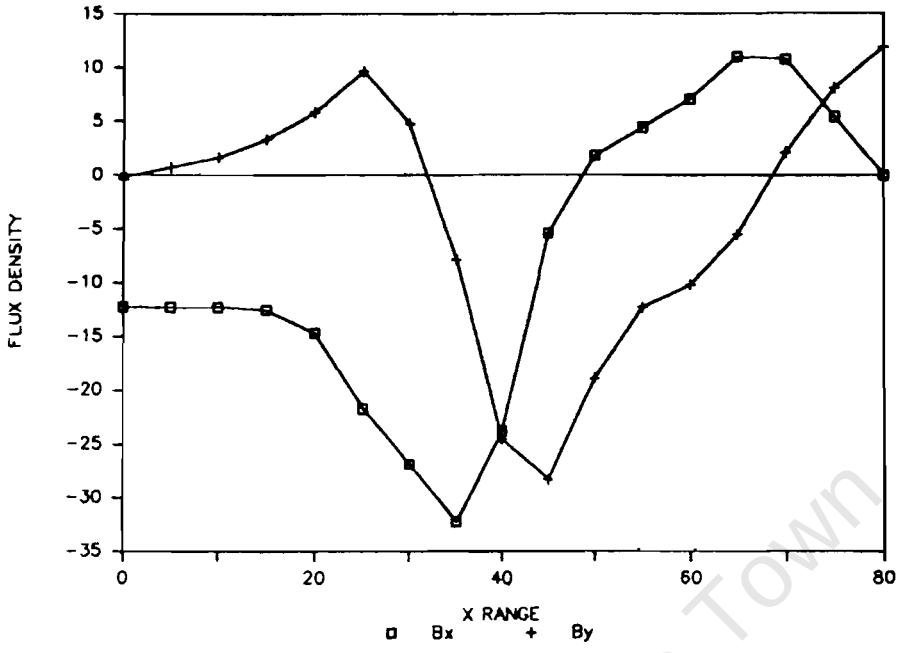


(a)

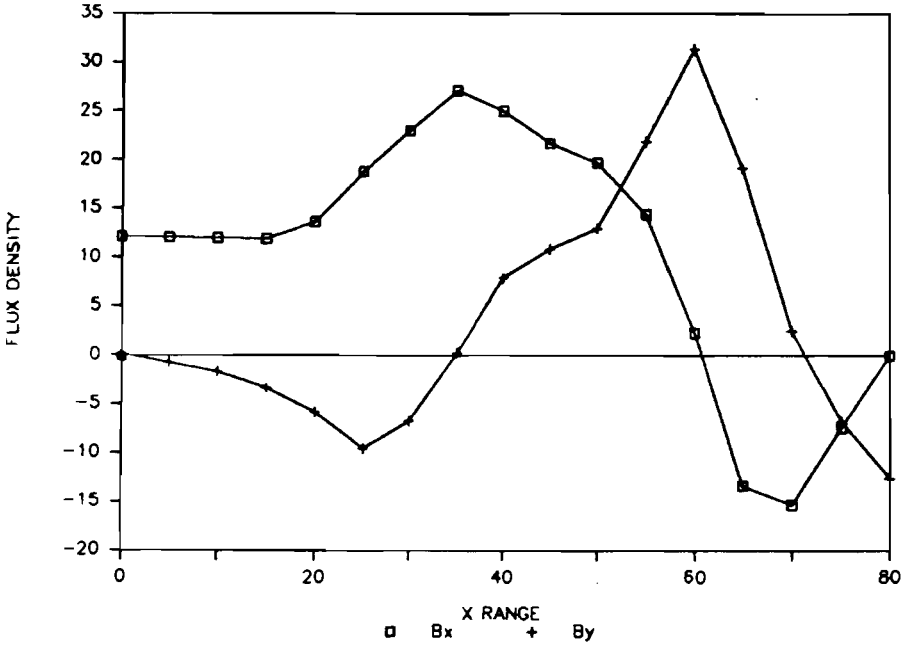


(b)

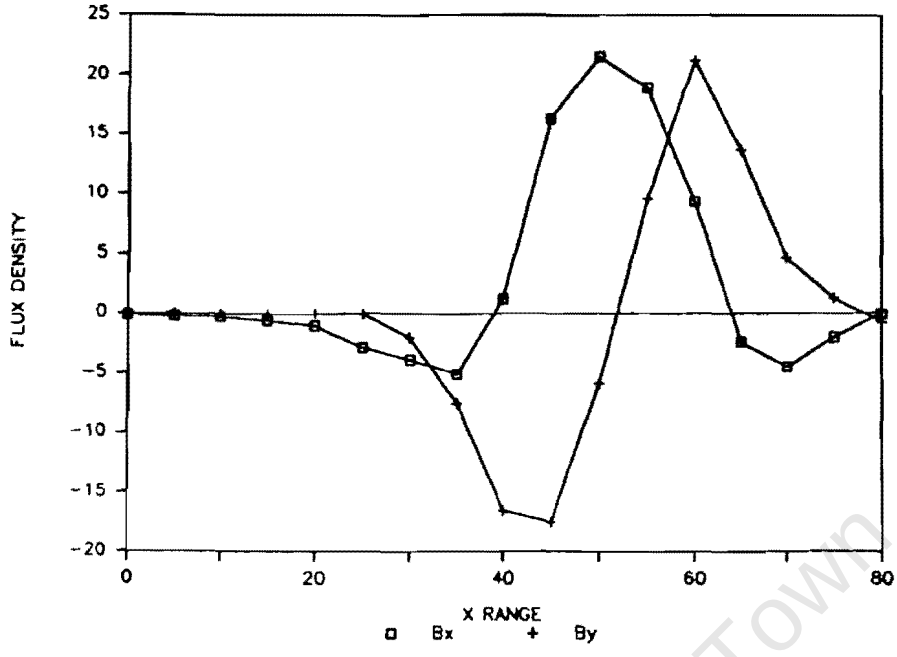
Figure 5.6 Linear machine configuration, boundary conditions and mesh generated.


15
10
5
0
-5
-10
-15
-20
-25
-30
-35
FLUX DENSITY
0
20
40
60
80
X RANGE
Bx
By

(a)


35
30
25
20
15
10
5
0
-5
-10
-15
-20
FLUX DENSITY
0
20
40
60
80
X RANGE
Bx
By

(b)



(c)

Figure 5.7 Linear machine flux density distributions for different current density distributions.

BIBLIOGRAPHY

1. G. Beer and J.L. Meek, " Infinite Domain Elements.", International Journal of Numerical Methods in Engineering, Vol. 17, pages 43-52, 1981.

2. P. Bettes, " More on Infinite Elements.", International Journal of Numerical Methods in Engineering, Vol. 15, pages 1613-1626, 1980.

3. P. Bettes, "Infinite Elements", International Journal of Numerical Methods in Engineering, Vol. 11, pages 53-64, 1977.

4. C.A. Brebbia and A.J. Ferrante, "Solution of Poisson's Equation using finite elements.", Microsoftware for Engineers, Vol. 1(2), 1985.

5. J.C. Cavendish, " Automatic Triangulation of Arbitrary Planar Domains for the Finite Element Method.", International Journal of Numerical Methods in Engineering, Vol. 8, pages 679 - 696, 1974.

6. J.J. Connor and C.A. Brebbia, Finite Element Techniques for Fluid Flow. London: Newness Butterworths, 1976.

7. E. Cuthill and J. Mc Kee, " Reducing the bandwidth of Sparse Symmetric Matrices.", Proceedings ACM National Conference Association for Computing Machinery, pages 157-172, New York,1969.

8. F. Damjanic and D.R.J. Owen, "Mapped Infinite Elements in Transient Thermal Analysis .", Computers and Structures, Vol. 19 (4), pages 673-687, 1984.

9. P.J. Davis and P. Rabinowitz, Methods of Numerical Integration, New York : Academic Press, 1975.

10. J. Fukuda and J. Suhara, "Automatic mesh generation for finite element analysis. ", Advances in Computational Methods in Structural Mechanics & Design, Vol. 1972.

11. J.A. George, " Sparse Matrix aspects of the Finite Element Method.", Proceedings of the 2nd International Symposium on Computing Methods in Applied Science and Engineering, Springer-Verlag, 1976.

12. N.E. Gibbs, W.G. Poole and P.K. Stockmeyer, " An Algorithm for Reducing the Bandwidth and Profile of a Sparse Matrix.", SIAM Journal of Numerical Analysis, Vol. 13 (2), pages 236-250, 1976.

13. J.F. Gieras, "Special Purpose Electrical Machines." (in Polish), ATR, BYDGOSZCZ, 1983, pages 233-241.

14. R.K. Livesley, Finite Elements : An Introduction for Engineers, London: Cambridge University Press, 1983.

15. D.A. Lowther, C.B. Rajanathan and P.P. Silvester, "A Finite Element Technique for solving 2-Dimensional Open boundary problems.", I.E.E.E. Transactions on Magnetics, Vol. MAG-14 (5), pages 467-469, 1978.

16. P.P. Lynn and H.A. Hadid, " Infinite Elements with $1/r^n$ Type Decay.", International Journal of Numerical Methods in Engineering, Vol. 17, pages 347-355, 1981.

17. C. Meyer, " Solution of Linear Equations -State-of-the-art.", Journal of the Structural Division, ASCE Proceedings, Paper 9861, pages 1507-1527, July 1973.

18. B.H. Mc Donald and A. Wexler, " Finite-Element Solution of Unbounded Field Problems.", I.E.E.E. Transactions on Microwave Theory and Techniques, Vol. MTT-20 (12), pages 841-847, 1972.

19. A. Recuero and J.P. Gutierrez, "Automatic Generation of Triangular Mesh of Finite Elements.", Engineering Software for Microcomputers, Proceedings of the First International Conference, Venice, Italy, April 1984, Swansea Wales: Pineridge Press, pages 393-401.

20. L. Resende, Review of Infinite Elements., University of Cape Town : 1982.

21. R.D. Shaw and R.G. Pitchen, " Modification to the Suhara-Fukuda method of network generation.", International Journal of Numerical Methods in Engineering, Vol. 12, pages 93-99, 1978.

22. P.P. Silvester and R.L. Ferrari, Finite Elements for Electrical Engineers. U.K. : Cambridge University Press, 1983.

23. P. Silvester, "High Order Polynomial Triangular Finite Elements for Potential Problems.", International of Engineering Science, Vol. 7 pages 849 - 861.

24. P. Silvester, H.S. Cabayan and B.T. Browne, "Efficient Techniques for Finite Element Analysis of Electric Machines." I.E.E.E. Transactions on Power Apparatus and Systems, Vol. PAS-92 (4), pages 1274-1281, 1973.

25. P.P. Silvester, D.A. Lowther, C.J. Carpenter and B.A. Wyatt, "Exterior Finite Elements for 2-Dimensional Field Problems with Open Boundaries.", Proceedings of I.E.E., Vol. 124, Pages 1267-1270, 1977.

26. P. Silvester and M.S. Hsieh, "Finite-Element Solution of 2-Dimensional Exterior field problems.", Proceedings of I.E.E., Vol. 118 (12), pages 1743-1747, 1971.

27. H.R. Schwarz, H. Rutishauser and E. Stiefel, Numerical Analysis of Symmetric Matrices, New Jersey: Prentice-Hall, 1973.

28. S.W. Sloan, "An implementation of Watson's algorithm for computing 2-Dimensional Delaunay triangulations.", Advances in Engineering Software, Vol. 6(4), pages 192-197, 1984.

29. J. Stoer and R. Bulirsch, " Introduction to numerical analysis.", 1989, New York, Springer-Verlag.

30. O.C. Zienkiewicz and Y.K. Cheung, "Finite Elements in the Solution of Field problems", The Engineer, September 24, 1965.

31. O.C. Zienkiewicz and R.E. Newton, "Coupled Vibrations of a Structure Immersed in a Compressible Fluid.", Proceedings of the Symposium on Finite Element Techniques, Stuttgart, 1969.

32. O.C. Zienkiewicz , D.W. Kelly and P. Bettes, " The coupling of the Finite Element Method and Boundary Solution Procedures.", International of Numerical Methods in Engineering, Vol. 11, pages 355-375, 1977.

33. O.C. Zienkiewicz, " The Finite Element Method, 3rd Edition, London : Mc Graw - Hill, 1977.

34. O.C. Zienkiewicz, P. Bettes, T.C. Chiam and C. Emson, " Numerical Methods for Unbounded Field Problems and a New Infinite Element Formulation.", Computational Methods for Infinite Domain Media - Structure Interaction, AMD Vol. 46, pages 115-148, 1981.

APPENDICES

APPENDIX A

The derivatives of the geometric shape-functions and the interpolation functions for the infinite elements, dealt with in Section 2.3.3 in Chapter 2, are given below.

1.ONE-WAY INFINITE QUADRATIC LAGRANGIAN ELEMENT

(a) Geometric Shape-Functions

$$\frac{\delta G_1}{\delta \alpha} = -2*\left[\alpha - \tfrac{1}{2}\right]*\left[\frac{\beta}{1-\beta}\right]$$

$$\frac{\delta G_2}{\delta \alpha} = 4*\alpha*\left[\frac{\beta}{1-\beta}\right]$$

$$\frac{\delta G_3}{\delta \alpha} = -2*\left[\alpha + \tfrac{1}{2}\right]*\left[\frac{\beta}{1-\beta}\right]$$

$$\frac{\delta G_4}{\delta \alpha} = \left[\alpha + \tfrac{1}{2}\right]*\left[1 + \frac{2\beta}{1-\beta}\right]$$

$$\frac{\delta G_5}{\delta \alpha} = -2*\alpha*\left[1 + \frac{2\beta}{1-\beta}\right]$$

$$\frac{\delta G_6}{\delta \alpha} = \left[\alpha - \tfrac{1}{2}\right]*\left[1 + \frac{2\beta}{1-\beta}\right]$$

$$\frac{\delta G_1}{\delta \beta} = -\alpha * \left[\alpha - 1 \right] * \left[\frac{1}{(1 - \beta)^2} \right]$$

$$\frac{\delta G_2}{\delta \beta} = -2 * \left[1 - \alpha^2 \right] * \left[\frac{1}{(1 - \beta)^2} \right]$$

$$\frac{\delta G_3}{\delta \beta} = -\alpha * \left[\alpha + 1 \right] * \left[\frac{1}{(1 - \beta)^2} \right]$$

$$\frac{\delta G_4}{\delta \beta} = \alpha * \left[\alpha + 1 \right] * \left[\frac{1}{(1 - \beta)^2} \right]$$

$$\frac{\delta G_5}{\delta \beta} = 2 * \left[1 - \alpha^2 \right] * \left[\frac{1}{(1 - \beta)^2} \right]$$

$$\frac{\delta G_6}{\delta \beta} = \alpha * \left[\alpha - 1 \right] * \left[\frac{1}{(1 - \beta)^2} \right]$$

(b) Interpolation Functions

$$\frac{\delta H_1}{\delta \alpha} = \tfrac{1}{4} * \beta * \left[-1 + 2\alpha \right] * \left[\beta - 1 \right]$$

$$\frac{\delta H_2}{\delta \alpha} = -\alpha * \beta * \left[\beta - 1 \right]$$

$$\frac{\delta H_3}{\delta \alpha} = \tfrac{1}{4} * \beta * \left[1 + 2\alpha \right] * \left[\beta - 1 \right]$$

$$\frac{\delta H_4}{\delta \alpha} = \tfrac{1}{2} * \left[1 + 2\alpha \right] * \left[1 - \beta^2 \right]$$

$$\frac{\delta H_5}{\delta \alpha} = -2 * \alpha * \left[1 - \beta^2 \right]$$

$$\frac{\delta H_6}{\delta \alpha} = \tfrac{1}{2} * \left[-1 + 2\alpha \right] * \left[1 - \beta^2 \right]$$

$$\frac{\delta H_1}{\delta \beta} = \tfrac{1}{4} * \alpha * \left[\alpha - 1 \right] * \left[-1 + 2\beta \right]$$

$$\frac{\delta H_2}{\delta \beta} = \tfrac{1}{2} * \left[1 - \alpha^2 \right] * \left[-1 + 2\beta \right]$$

$$\frac{\delta H_3}{\delta \beta} = \tfrac{1}{4} * \alpha * \left[\alpha + 1 \right] * \left[-1 + 2\beta \right]$$

$$\frac{\delta H_4}{\delta \beta} = -\alpha * \beta * \left[\alpha + 1 \right]$$

$$\frac{\delta H_5}{\delta \beta} = -2 * \beta * \left[1 - \alpha^2 \right]$$

$$\frac{\delta H_6}{\delta \beta} = -\alpha * \beta * \left[\alpha - 1 \right]$$

2. TWO-WAY INFINITE QUADRATIC LAGRANGIAN ELEMENT

(a) Geometric Shape-Functions

$$\frac{\delta G_1}{\delta \alpha} = \left[\frac{2}{(1-\alpha)^2} \right] * \left[\frac{2\beta}{1-\beta} \right]$$

$$\frac{\delta G_2}{\delta \alpha} = -\left[\frac{2}{(1-\alpha)^2} \right] * \left[\frac{2\beta}{1-\beta} \right]$$

$$\frac{\delta G_3}{\delta \alpha} = \left[\frac{2}{(1-\alpha)^2} \right] * \left[1 + \frac{2\beta}{1-\beta} \right]$$

$$\frac{\delta G_4}{\delta \alpha} = -\left[\frac{2}{(1-\alpha)^2} \right] * \left[1 + \frac{2\beta}{1-\beta} \right]$$

$$\frac{\delta G_1}{\delta \beta} = \left[\frac{2}{(1-\beta)^2} \right] * \left[\frac{2\alpha}{1-\alpha} \right]$$

$$\frac{\delta G_2}{\delta \beta} = -\left[\frac{2}{(1-\beta)^2} \right] * \left[1 + \frac{2\alpha}{1-\alpha} \right]$$

$$\frac{\delta G_3}{\delta \beta} = \left[\frac{2}{(1-\beta)^2} \right] * \left[1 + \frac{2\alpha}{1-\alpha} \right]$$

$$\frac{\delta G_4}{\delta \beta} = -\left[\frac{2}{(1-\beta)^2} \right] * \left[\frac{2\alpha}{1-\alpha} \right]$$

(b) Interpolation Functions

$$\frac{\delta H_1}{\delta \alpha} = \tfrac{1}{4} * \beta * \left[-1 + 2\alpha \right] * \left[\beta - 1 \right]$$

$$\frac{\delta H_2}{\delta \alpha} = -\alpha * \beta * \left[\beta - 1 \right]$$

$$\frac{\delta H_3}{\delta \alpha} = -2 * \alpha * \left[1 - \beta^2 \right]$$

$$\frac{\delta H_4}{\delta \alpha} = \tfrac{1}{2} * \left[-1 + 2\alpha \right] * \left[1 - \beta^2 \right]$$

$$\frac{\delta H_1}{\delta \beta} = \tfrac{1}{4} * \alpha * \left[\alpha - 1 \right] * \left[-1 + 2\beta \right]$$

$$\frac{\delta H_2}{\delta \beta} = \tfrac{1}{2} * \left[1 - \alpha^2 \right] * \left[-1 + 2\beta \right]$$

$$\frac{\delta H_3}{\delta \beta} = -2 * \beta * \left[1 - \alpha^2 \right]$$

$$\frac{\delta H_4}{\delta \beta} = -\alpha * \beta * \left[\alpha - 1 \right]$$

APPENDIX B

Abscissae and weights are given for second and third order Gauss Legendre numerical integration, as required in Chapter 2, Section 2.3.3(a).

1. ABSCISSAE

The abscissae for both the One-way and Two-way Quadratic Lagrangian Infinite elements are the same and are given in coordinate form.

(a) Second Order Integration

(i) (-0.57735027,-0.57735027)

(ii) (-0.57735027, 0.57735027)

(iii) (0.57735027,-0.57735027)

(iv) (0.57735027,0.57735027)

(b) Third Order Integration

(i) (-0.77459667, -0.77459667)

(ii) (-0.77459667, 0.0)

(iii) (-0.77459667, 0.77459667)

(iv) (0.0, -0.77459667)

(v) (0.0,0.0)

(vi) (0.0, 0.77459667)

(vii) (0.77459667, -0.77459667)

(viii) (0.77459667, 0.0)

(ix) (0.77459667, 0.77459667)

2. WEIGHTS

The weights for both the One-way and Two-way Quadratic Lagrangian Infinite elements are the same.

(a) Second Order Integration

(i) 1.0 , 1.0

(ii) 1.0 , 1.0

(iii) 1.0 , 1.0

(iv) 1.0 , 1.0

(b) Third Order Integration

(i) 0.55555556 , 0.55555556

(ii) 0.55555556 , 0.88888889

(iii) 0.55555556 , 0.55555556

(iv) 0.88888889 , 0.55555556

(v) 0.88888889 , 0.88888889

(vi) 0.88888889 , 0.55555556

(vii) 0.55555556 , 0.55555556

(viii) 0.55555556 , 0.88888889

(ix) 0.55555556 , 0.55555556

APPENDIX C

The data files for the selected examples 1 and 2 in Chapter 5 are given below.

1. STRAIGHT MACHINE SLOT

Page 1	Page 2	Page 3	Page 4
2	3,4	3,0,0	10.0
0	0	1.0	8
7	10.0	4,1,0	5
0.0,0.0	4,5	0.0	6
100.0,0.0	0	5,1,0	7
100.0,20.0	10.0	0.0	1
40.0,20.0	5,6	6,1,0	0
40.0,80.0	0	0.0	1
0.0,80.0	10.0	7,0,0	1
40.0,0.0	6,1	0.0	13.333
8	0	2	10.0
1,7	10.0	2	1.0,1.0,0.122
0	4,7	0.0	2
10.0	0	0.0	
7,2	10.0	10.0	
0	7	2	
10.0	1,1,0	3	
2,3	0.0	4	
0	2,1,0	8	
10.0	0.0	0	

2. SLANTED MACHINE SLOT

Page 1	Page 2	Page 3
1	0	2
0	20.0	0.0
8	6,1	20.0
0.0,0.0	0	1
40.0,0.0	20.0	2
80.0,0.0	6,7	3
100.0,0.0	0	4
200.0,173.205	0	5
100.0,173.205	5,8	6
200.0,346.410	0	0
300.0,346.410	0	1
8	6	2
1,2	1,0,0	3
0	0.0	0.0
20.0	2,0,0	8
2,3	1.0	5
0	3,0,0	7
20.0	0.0	2
3,4	4,0,0	1
0	0.0	1
20.0	5,3,0	11.57
4,5	0.0	10
0	6,0,0	1.0,1.0,1.0
20.0	0.0	2
5,6	1	