THE UNIVERSITY OF MANITOBA

AN ISOPARAMETRIC BOUNDARY ELEMENT FORMULATION FOR ELASTOSTATICS

BY

W. NEIL AITKEN

A THESIS SUBMITTED TO THE FACULTY OF GRADUATE STUDIES
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE

DEPARTMENT OF MECHANICAL ENGINEERING
WINNIPEG, MANITOBA

DECEMBER, 1985

AN ISOPARAMETRIC BOUNDARY ELEMENT FORMULATION FOR ELASTOSTATICS

BY

W. NEIL AITKEN

A thesis submitted to the Faculty of Graduate Studies of the University of Manitoba in partial fulfillment of the requirements of the degree of

MASTER OF SCIENCE

ABSTRACT

An advanced boundary element method was developed to solve two-dimensional elastostatic boundary value problems. The method is characterized by replacing the physical problem with an infinite plane problem which has the same solution, and for which there exists a simple analytical solution. This is accomplished by distributing a fictitious layer of stress over a tracing of the boundary in an infinite plane in such a way as to duplicate applied boundary conditions.

The current fictitious stress method was advanced by incorporating the accurate modelling of curved boundaries, a higher order distribution of the unknown fictitious stress layer, and the formulation of Galerkin's method as a means of optimizing the solution. These developments were achieved by using shape function representations of the space co-ordinates and of the unknown fictitious stress distribution in the integral equations of the boundary element method.

The isoparametric boundary element algorithm was numerically implemented in a program called BEAST. The results from several theoretical elasticity problems showed a close correlation with the analytical solutions for both displacement and stress traction problems. BEAST was also found to yield far more accurate results than the current line element method.

Current research has focused on the accurate solution of mixed boundary value problems and on the addition of useful input/output features.

ACKNOWLEDGEMENTS

I would like to express my gratitude to Dr. A. B. Thornton-Trump for his guidance and encouragement during the course of this research.

I would also like to thank the other members of the advisory committee: Dr. W. L. Cleghorn and Dr. R. B. Pinkney for their helpful discussions on the technical and literary aspects of this work.

A special thanks is extended to my friend and colleague Allan Dolovich for his interest and enthusiasm, and for his extensive contribution to this research.

Finally, I would like to share the joy of completing this work with my intimate friend Barbara Woelcke, and to thank her for her love, support and understanding during the trying months of composing this thesis.

TABLE OF CONTENTS

LIST OF FIGURES

NOMENCLATURE

a	=	field point/element
A_i	=	unknown coefficients
b	=	source point/element
B_i	=	displacement or stress traction vector
C	=	boundary of finite solid
C'	=	tracing of boundary in the infinite plane
C_{ij}	=	influence coefficients for displacement or stress traction
E	=	Young's modulus
$f(\xi)$	=	arbitrary function
F_i	=	point force
G	=	shear modulus
I_{ij}	=	integral of influence functions (general)
I^*_{ij}	=	integral of influence functions times shape functions
$J(\xi)$	=	Jacobian of transformation
J_n	=	nodal Jacobian
L	=	linear operator
n_i	=	unit outward normal
R	=	distance between field point and source point
S_{ijk}	=	influence functions for stress
t_i	=	stress traction vector
T_{ij}	=	influence functions for stress traction
u_i	=	displacement vector
U_{ij}	=	influence functions for displacement
Wt	=	weighting factor for Gaussian quadrature

X_i	=	x co-ordinates of nodes
Y_i	=	y co-ordinates of nodes
$\alpha_i(\xi)$	=	shape functions
$\beta_i(\xi)$	=	derivatives of shape functions
γ	=	shear strain
ε	=	normal strain
μ_i	=	expansion functions
ν	=	Poisson's ratio
ξ	=	parameter, Gauss point
σ_{ij}	=	stress tensor
ϕ_i	=	fictitious stress vector
Φ	=	Airy's stress function
Φ_i	=	nodal fictitious stress vector
ω	=	unknown functions

CHAPTER I

Introduction

1.1 Boundary Element Methods

Most practical engineering problems are impossible to solve by analytical methods, in which exact mathematical expressions are obtained for the required variables. In the past, a broad range of assumptions was used to simplify problems to a point where an analytical solution could be obtained. It was hoped that these solutions bore a reasonable resemblance to the solutions of the real problem. However, with the advent of digital computers, the emphasis in engineering analysis has moved towards more versatile and accurate numerical methods.

The most established numerical technique for solving engineering problems is the finite element method. In this technique, variational methods are used to obtain approximate solutions to the partial differential equations governing the physical process. The distribution of the unknown variables are obtained as approximate values at a finite number of discrete points over the entire domain of interest. In this sense, the finite element technique may be described as a domain method.

In boundary element methods, the primary result is the distribution of the unknown variables on the boundary of the solution domain only. Thus, any boundary element technique may be described as a boundary method.

Boundary element methods are formulated by deriving a boundary integral equation which is equivalent to the partial differential equation that governs the physical process. Therefore, calculations are performed over only the boundary of the domain. This effectively reduces the dimensionality of the problem by one, so that a three-dimensional volume problem becomes a two-dimensional surface problem, and a two-dimensional planar problem is reduced to a one-dimensional line problem.

In boundary element methods, the interior of the solution domain is not discretized. Therefore, there is much less approximation involved in representing the solution variables, and rapid variations of these variables can be resolved very accurately.

1.2 A New Concept in Boundary Elements

Many vastly different numerical techniques have been proposed which could be classified as boundary element methods. Most of these were derived in order to solve specific types of problems, while others apply to general classes of problems. This thesis is concerned with the boundary element solution of elastostatic problems. This class of problem requires that the solid body remain elastic under the applied loading, and that no acceleration of the body result.

The objective of this thesis is to develop a new boundary element method which is generally applicable to two-dimensional elastostatic problems of the complexity found in practice. This method must be accurate, simple, and must lend itself well to numerical implementation. The resulting program should run quickly and efficiently on a micro-computer and, most importantly, should be simple to use.

Given these requirements, the "fictitious stress" boundary element method was selected to form the basis on which a new advanced formulation would be developed. This method is characterized by replacing the physical problem with an infinite plane problem which has the same solution, and for which there exists a simple analytical solution.

Extensions to the fictitious stress method that are required by the new method include the accurate modelling of curved boundaries, reduction in the number of elements required to solve a problem, and improved accuracy and versatility, all without a loss of the computational efficiency which distinguishes this method.

CHAPTER II

Literature Review

2.1 Introduction

Boundary element research in elasticity has had a short though varied history. To date, there is no single accepted boundary element method, and this has inhibited it's widespread use in industry. Boundary element formulations have been developed for many specialized applications but the development of generalized programs has been slow.

The mathematical foundation for every boundary element method comes from the theory of integral equations, first investigated by Fredholm [01] in 1906. This theory was later applied to solve integral equations in elasticity,using complex variable theory, by the Soviet researcher Muskhelishvili [02] in 1953. In 1959, Mikhlin [03] presented similar work but avoided using complex variables, and thereby opened the avenue for numerical solutions.

In the 1960's the research into useful applications of integral equation theory accelerated because of the advent of the computer. In the middle of the decade two separate schools of research evolved, distinct in their approach to solving the same problems. One school of research developed an integral equation method from the direct application of potential theory to elastostatics. The other school solved the integral equations indirectly by replacing the original problem with an equivalent fictitious problem for which the fundamental solutions of elasticity are applicable.

2.2 Direct School of Research

In 1967 Rizzo [04] introduced an integral equation method for solving two-dimensional elastostatic problems. He derived the Somigliana identity for the displacements inside a body, by using Betti's identity. The integral equation was obtained by taking the limit of Somigliana's identity for points located on the boundary. Solution of these equations yielded the unknown stress tractions and displacements on the boundary which could be used to obtain interior values. Kelvin's solution for a point force in an infinite plane was used as the basis of the formulation.

The following year, Cruse [05] presented an extended version of Rizzo's theory which could solve elastodynamic problems. The integral equations were altered by applying Laplace transforms. In the same year, Rizzo and Shippy [06] solved inclusion problems using the previous integral equation method.

In 1969 Cruse [07] described the first three-dimensional boundary element method which was based on Rizzo's original two-dimensional research. Surfaces were discretized using flat triangular elements and several problems were solved. Cruse and Swedlow [08] introduced the first boundary element method for elastoplastic problems. The elastoplastic methods have since been advanced by many researchers including Riccardella [09], who applied the Von Mises criterion, and Mendelson [10], who presented a three-dimensional formulation.

In 1973 Cruse [11] presented the first comprehensive comparison between the direct boundary element method and finite element solutions of three-dimensional problems. In addition, several new crack propagation problems were solved. Cruse concluded that the boundary element method was superior for problems requiring good resolution. This was due to the reduced problem size and run time.

Cruse [12] altered his formulation the following year to improve the accuracy of solution. Instead of using constant boundary data over each element, he used a linear variation between the end-points of elements. This was accomplished by formulating the unknown boundary data in terms of a double Taylor series expansion and retaining the linear terms.

A significant advancement of the direct method was accomplished by Lachat and Watson [13] in 1977. These researchers used quadratic shape functions to describe the curve of three-dimensional elements and to describe the quadratic variation of the unknown displacements and stress tractions on the surface. This improvement was found to increase the accuracy of solutions and required far fewer elements to model a surface accurately.

Since this improvement, the direct boundary element theory applied to elastostatics has remained essentially unchanged. In 1977 Brebbia [14] described a procedure for coupling the direct method to finite elements so that a new hybrid algorithm was possible. Brebbia [15] also consolidated the previous research in a comprehensive publication on two and three-dimensional, linear and quadratic elements. Recently, the research into direct methods has concentrated on elastoplastic methods.

2.3 Indirect School of Research

While direct approaches have remained at the fore-front of past research, indirect methods have evolved much more slowly. This is primarily due to a lack of any rigorous justification for the fictitious quantities that arise in the indirect approaches. However, Brebbia and Butterfield [16] have proven that the indirect method and direct method are actually equivalent by deriving one from the other. The indirect method has proven to be a short-cut to the same solution, which avoids the excessive mathematical treatment. Thus, the resulting integral equations are very much simpler than those solved in direct methods.

The first true boundary element method was an indirect approach proposed by Massonet [17] in 1965. This was the first integral equation solution suitable for numerical implementation. Massonet used Flamant's half plane problem as a fundamental solution. Fictitious loads of unknown magnitudes were distributed around the solid imbedded in a half plane and their strengths were determined from the given boundary conditions. An iterative method was used to solve the integral equations. Once the magnitude of the fictitious loads was determined, Flamant's solution provided stresses at any interior points.

In 1968 Oliveira [18] advanced this research by using Kelvin's solution for a point force in an infinite plane in place of Flamant's solution. The fictitious forces were distributed on an auxiliary boundary, removed some distance from the actual boundary. This improved results near the boundary and near corners. However, this technique applied only to plane stress problems and, in some instances, the resulting system of equations could become unstable.

Important research which contributed to the advancement of indirect methods was done by Kupradze [19] in 1964. He described an integral formuation based on an elastic body subjected to periodic body forces and boundary conditions, where the static problem was a special case. Kupradze introduced the concept of elastic potentials which arise from the simple and double layers of fictitious force. In 1972 Watson [20] described both two and three-dimensional elastostatic boundary element methods based on Kupradze ideas of elastic potentials.

In 1970 and 1971, Banerjee and Butterfield [21] applied a fictitious stress method to the analysis of compressible piles. This formulation used Mindlin's solution for a point force in the interior of a semi-infinite solid as a fundamental solution. Their approach was very similar to Massonet's original formulation. In 1972 Butterfield and Tomlin [22] extended this formulation to nonhomogeneous and anisotropic problems using a fundamental solution for a point force in orthotropic lamina.

In the same year Benjumea and Sikarskie [23] presented two approaches. The first was similar to Oliviera's method but the auxiliary boundary was made tangent to the real boundary. The second approach was a refined version of Massonet's method and, although very general, included all of the essential integral equations of the current level of advancement.

In 1976 Banerjee [24] outlined an indirect method in detail using Kelvin's solution. This was an integral equation method which extended Benjumea's work to piece-wise, nonhomogeneous, three-dimensional elastic bodies using elements composed of multiple flat surfaces. This formulation is indicative of the current state of research since all later developments are variations on this technique.

Crouch [25], also in 1976, introduced the displacement discontinuity method which used fictitious displacements instead of fictitious stresses. Though different, this method is no more sophisticated than any previous techniques.

In 1978, Altiero and Gavazza [26] proposed the dislocation dipole method. This method used a double layer theory, applying one layer of fictitious body force and one layer of fictitious displacements. This ensured that all the resulting integral equations were singular, as is convenient for numerical integration. However, the algorithm used only straight line elements and any refinements have yet to be reported.

A new concept in elements was described by Mahajerin [27] in 1983. These elements were circular and defined by a radius of curvature. However, problems with straight boundaries or corners could not be solved (because the radius approaches infinity or zero) and special trigonemetric equations were required to describe the boundary shape and the boundary conditions. No real physical problems could be solved.

Though many variations on the indirect approach have been proposed, no researcher has advanced his formulation to a level where it is useful for solving complex engineering problems. In all cases, the boundary elements are straight lines (or flat surfaces in three-dimensions) and the fictitious layer is distributed in constant blocks over the boundary. Obviously, advancements in these areas are necessary in order to make the boundary element method useful to industry.

CHAPTER III

Solutions to Problems in Elasticity

3.1 Introduction

The foundations for the boundary element method presented in this thesis are found in the theory of elasticity. This chapter reviews the topics in elasticity which are necessary for a complete understanding of the boundary element solution.

The boundary element technique is applicable to the solution of elastostatic boundary value problems. The physical problem must be modelled accurately by the biharmonic equation, subject to a set of continuous boundary conditions.

The boundary element method is capable of solving complex engineering problems by superimposing fundamental solutions to the biharmonic equation. Kelvin's solution for a point load in an infinite solid forms the basis on which to develop the essential integral equations of this technique.

3.2 Boundary Value Problems

In order to obtain a boundary element solution, an engineering problem must be posed in the form of a "boundary value problem" [28]. From the physical problem, a mathematical model is created which simplifies though closely approximates reality. The problem is then formulated mathematically, usually in the form of a partial differential equation. This equation is solved, subject to certain constraints, to yield an approximate solution to the original physical problem.

Consider a region of material R which is defined by a boundary C as shown in figure 3.1. External factors are applied to the boundary such as electropotential, temperature or forces and the resulting conditions at some point P(x,y,z) in the region are desired.

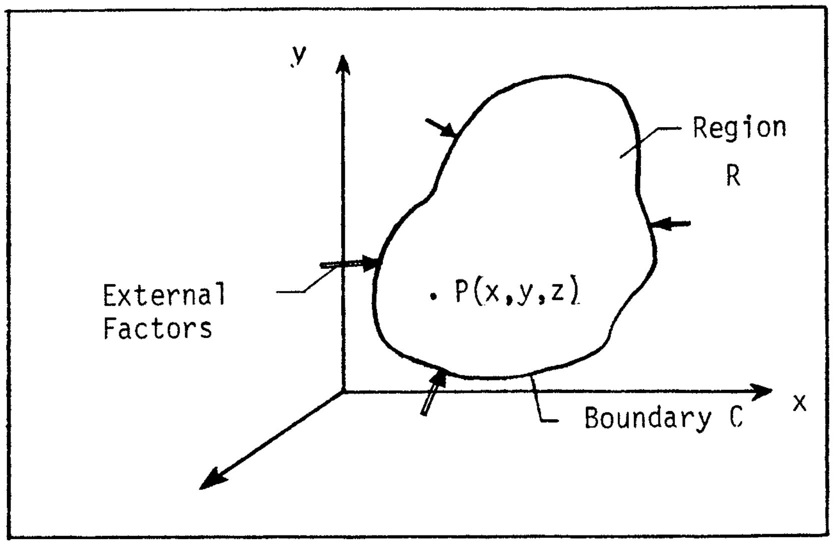


Figure 3.1: A Boundary Value Problem

A partial differential equation, such as Laplace's equation, Poisson's equation or the biharmonic equation, describes the conditions in R as a function of position. By specifying the conditions at all points on C to the partial differential equation, a unique condition at each point in the region may be determined by solving the equation.

Therefore, a boundary value problem is characterized by a partial differential equation and a continuous set of boundary conditions. If the problem is well posed, an approximate solution may be obtained for a physical problem that is well described by this equation.

3.3 Biharmonic Equation

A partial differential equation that is used to solve two-dimensional problems in elasticity is called the biharmonic equation [29].

Consider the equilibrium of a small block with a rectangular section of width w, height h and unit depth as shown in figure 3.2. The rectangular section consists of faces 1, 2, 3 and 4 which are each acted upon by a shear and a normal stress that result from some applied forces.

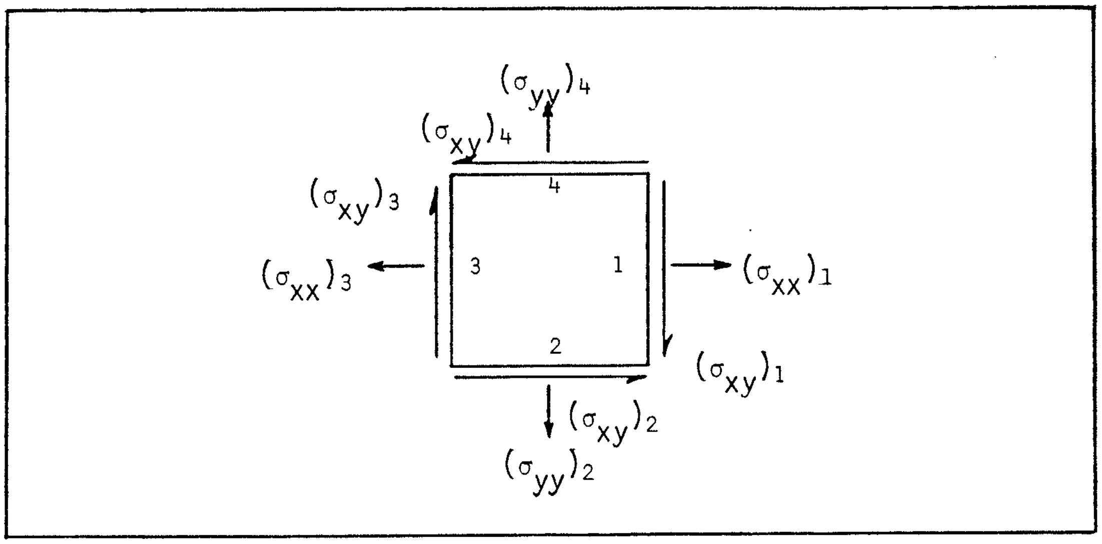


Figure 3.2: Equilibrium of Small Block

The magnitude of the stress components at the midpoint of each face i are represented by $(\sigma_{xy})_i$ and $(\sigma_{xx})_i$ or $(\sigma_{yy})_i$. The force acting on any face may be approximated by multiplying the midpoint stress by the area of the face. If X, Y denote components of the body force per unit volume, the equation of equilibrium of forces in the x-direction is

$$(\sigma_{xx})_1 h - (\sigma_{xx})_3 h + (\sigma_{xy})_2 w - (\sigma_{xy})_4 w + X hw = 0 \qquad (3.1)$$

Dividing this expression by the rectangular area hw gives

$$\frac{(\sigma_{xx})_1 - (\sigma_{xx})_3}{w} + \frac{(\sigma_{xy})_2 - (\sigma_{xy})_4}{h} + X = 0 \quad . \tag{3.2}$$

In the limit as the area of the rectangular section approaches zero, the term$[(\sigma_{xx})_1 - (\sigma_{xx})_3]/w$ becomes $\partial\sigma_{xx}/\partial x$ and the term $[(\sigma_{xy})_2 - (\sigma_{xy})_4]/h$ becomes $\partial\sigma_{xy}/\partial y$. The equation of equilibrium in the x-direction is then

$$\frac{\partial\sigma_{xx}}{\partial x} + \frac{\partial\sigma_{xy}}{\partial y} + X = 0 \quad . \tag{3.3}$$

By a similar derivation the equation of equilibrium in the y-direction is

$$\frac{\partial\sigma_{yy}}{\partial y} + \frac{\partial\sigma_{xy}}{\partial x} + Y = 0 \quad . \tag{3.4}$$

The normal and shear strains at a point are defined in terms of the components of elastic displacement u, v as

$$\varepsilon_{xx} = \frac{\partial u}{\partial x} \qquad \varepsilon_{yy} = \frac{\partial v}{\partial y}$$

$$\gamma_{xy} = \frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} \quad . \tag{3.5}$$

This definition implies that the components of strain are not independent but are related by relationships called "compatibility equations". Differentiating ε_{xx} twice with respect to x and ε_{yy} twice with respect to y yields

$$\frac{\partial^2\varepsilon_x}{\partial y^2} = \frac{\partial^3 u}{\partial y^2\partial x} \quad , \quad \frac{\partial^2\varepsilon_y}{\partial x^2} = \frac{\partial^3 v}{\partial x^2\partial y} \tag{3.6}$$

and differentiating γ_{xy} once with respect to each of x and y gives

$$\frac{\partial^2 \gamma_{xy}}{\partial x \, \partial y} = \frac{\partial^3 u}{\partial x \, \partial y^2} + \frac{\partial^3 v}{\partial x^2 \partial y} \tag{3.7}$$

It follows that the two-dimensional equation for compatibility requires

$$\frac{\partial^2 \varepsilon_x}{\partial y^2} + \frac{\partial^2 \varepsilon_y}{\partial x^2} = \frac{\partial^2 \gamma_{xy}}{\partial x \, \partial y} \tag{3.8}$$

Generalized Hooke's law for plane stress [29] relates strain and stress components as follows

$$\varepsilon_{xx} = \frac{1}{E}(\sigma_{xx} - \nu\sigma_{yy}) , \qquad \varepsilon_{yy} = \frac{1}{E}(\sigma_{yy} - \nu\sigma_{xx})$$
$$\gamma_{xy} = \frac{1}{G}\sigma_{xy} = \frac{2(1+\nu)}{E}\sigma_{xy} \tag{3.9}$$

Substituting these into the compatibility equations (3.8) yields

$$\frac{\partial^2}{\partial y^2}(\sigma_{xx} - \nu\sigma_{yy}) + \frac{\partial^2}{\partial x^2}(\sigma_{yy} - \nu\sigma_{xx}) = 2(1+\nu)\frac{\partial^2 \sigma_{xy}}{\partial x \, \partial y} \tag{3.10}$$

This expression may be simplified by using the equations of equilibrium. Differentiating equation (3.3) with respect to x and equation (3.4) with respect to y and summing the two yields (assuming a constant body force)

$$2\frac{\partial^2 \sigma_{xy}}{\partial x \, \partial y} = -\frac{\partial^2 \sigma_{xx}}{\partial x^2} - \frac{\partial^2 \sigma_{yy}}{\partial y^2} \tag{3.11}$$

which may be substituted into equation (3.10) to yield the compatibilty equation for stress

$$(\partial^2/\partial x^2 + \partial^2/\partial y^2) (\sigma_{xx} + \sigma_{yy}) = 0 \quad . \tag{3.12}$$

In the absence of body forces, the equations of equilibrium and compatibility are satisfied if σ_{xx} , σ_{yy} and σ_{xy} are defined in terms of the so-called Airy stress function Φ . This function is related to the stress tensor by

$$\sigma_{xx} = \frac{\partial^2\Phi}{\partial y^2} \qquad \sigma_{yy} = \frac{\partial^2\Phi}{\partial x^2} \qquad \sigma_{xy} = -\frac{\partial^2\Phi}{\partial x\, \partial y} \quad . \tag{3.13}$$

Substituting these relations into the compatibility equation for stress implies that the function must satisfy

$$\frac{\partial^4\Phi}{\partial x^4} + 2\frac{\partial^4\Phi}{\partial x^2\partial y^2} + \frac{\partial^4\Phi}{\partial y^4} = 0 \quad . \tag{3.14}$$

Equation (3.14) is called the biharmonic equation of elasticity. A two-dimensional boundary value problem may be solved by enforcing a set of boundary conditions associated with the physical problem on the biharmonic equation. The stress tensor components at any points may be found by integrating the solution Φ in accordance with the definition of the stress function. Note that the Airy stress function is a fictitious quantity, without physical meaning, which is used indirectly to obtain a solution to the equilibrium and compatability equations.

3.4 Boundary Conditions

In addition to specifying the partial differential equation which governs the problem, boundary conditions must be specified to indicate the applied loading. In elastostatics, boundary conditions are specified in the form of stress tractions and displacements.

The stress tensor components in a solid at a point very near the boundary must be in equilibrium with the external forces. In this sense, the external forces may be regarded as a continuation of the internal stress distribution. The external force per unit area is a vector representation of the stress acting on a boundary and is call a <u>stress traction</u>. It may be defined in terms of the stress tensor as

$$\begin{aligned} t_x &= \sigma_{xx} n_x + \sigma_{xy} n_y \\ t_y &= \sigma_{yx} n_x + \sigma_{yy} n_y \end{aligned} \tag{3.15}$$

where n_x and n_y are components of the unit outward normal to the boundary. Thus, the boundary condition at a point is reduced to a vector from a second rank tensor representation because the plane on which the stress acts is defined by the tangent plane to the boundary.

Displacements may also be specified as boundary conditions of an elastostatic problem. In particular, if the external forces are not self-equilibriating then the displacement of at least one point must be specified for the body to remain static.

To properly pose an elastostatic boundary value problem, either a stress traction or a displacement must be specified at every point on the boundary of the solid. For a mixed problem, stress tractions may be specified on some parts of the boundary and displacements on others, or different components of each may be specified over the same portion of the boundary.

Sections of the boundary which are not subjected to external forces, displacements or constraints are called "traction free". They are defined by applying a zero stress traction so that the boundary conditions will be continuous over the entire boundary.

Note that an applied force is not a valid boundary condition in elastostatics. A boundary condition must be compatible with the variables in the partial differential equation. Therefore, forces must be divided by the area of application and prescribed as stress tractions.

3.5 Singular Solutions

Analytical solutions have been derived in many disciplines for the case where is some sort of disturbance in an infinite homogeneous region. These solutions are useful because they give the effect of the disturbance upon any point in the region. Such cause and effect relationships are referred to as singular solutions because they are well behaved everywhere in the region except at the point of the disturbance. At this point the solution usually tends to infinity as a result of a mathematical anomaly.

One such singular solution of the biharmonic equation is Kelvin's solution for a point force in an infinite solid. In this example, the disturbance is a concentrated force that induces stress in the surrounding field. At the application point of the force the stress is theoretically infinite because a finite force is acting over an infinitesimal area. However, since the region very close to the force is in fact plastically strained, Kelvin's solution does not apply here (see figure 3.3).

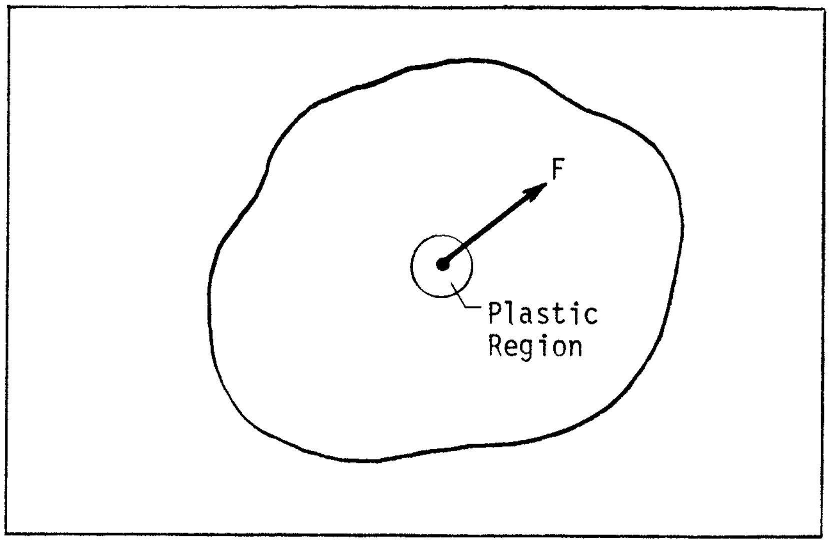


Figure 3.3: A Singular Solution

3.6 Kelvin's Solution

Kelvin's solution for plane strain [30] yields stress and displacement components at a point in an infinite plane when a concentrated force is applied to another point. Figure 3.4 shows a plane within an infinite solid subjected to a point force F at the source point B. Assuming a plane strain condition, then any cross-section of the solid will be representative of the whole body. Referring to figure 3.4, the displacement components at a field point A are described by

$$u_x = \frac{F_x}{2G}\left[(3-4\nu)g - R_x\, g_{,x}\right] + \frac{F_y}{2G}\left[-R_y\, g_{,x}\right]$$

$$u_y = \frac{F_x}{2G}\left[-R_x\, g_{,y}\right] + \frac{F_y}{2G}\left[(3-4\nu)g - R_y\, g_{,y}\right] \tag{3.16}$$

$$\text{where} \quad g(x,y) = \frac{-1}{4\pi(1-\nu)} \ln\sqrt{R_x^2 + R_y^2} \qquad (3.17)$$

$g,_x$ and $g,_y$ are the partial derivatives of $g(x,y)$, G is the shear modulus and ν is Poisson's ratio. These equations may be written in tensor notation as

$$u_i = U_{ij} * F_j \qquad (3.18)$$

where U_{ij} are called the influence functions for displacement and represent the coefficients of F_x and F_y in equation (3.16). These functions measure the contribution of the point force F to the displacement of an arbitrary point A.

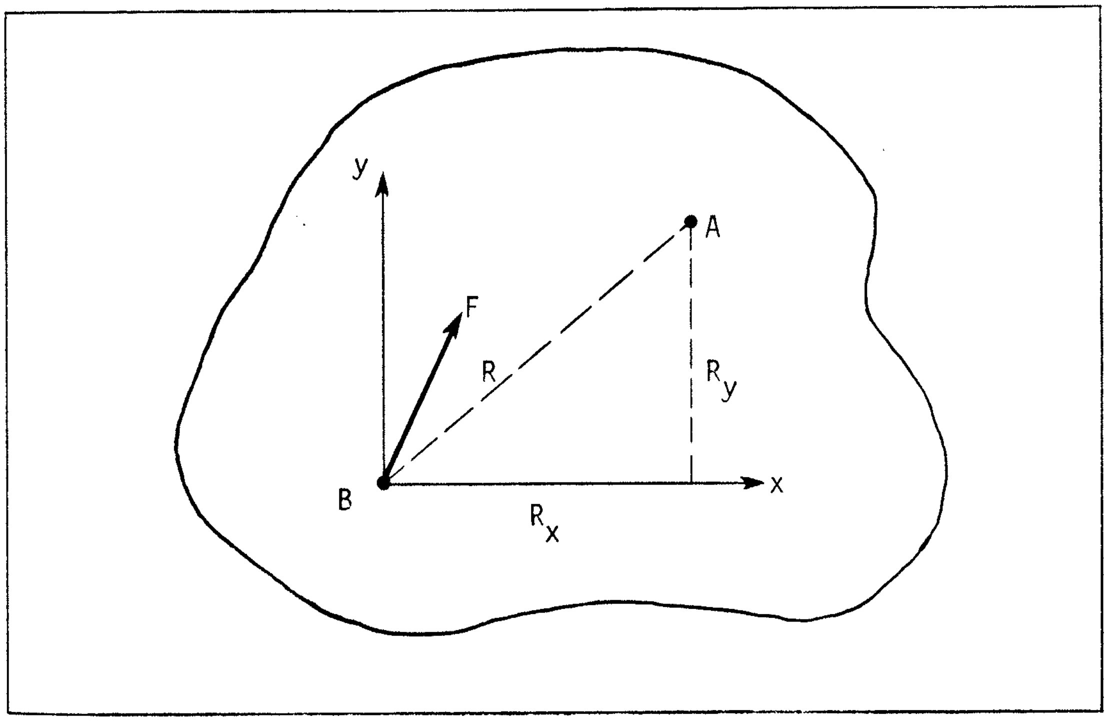


Figure 3.4: Kelvin's Solution

If the expressions for displacement are substituted into the Lamé equation that relates displacement to stress, the following three components of the stress tensor will result:

$$\sigma_{xx} = F_x\,[2(1-\nu)g,_x - R_x\,g,_{xx}] + F_y\,[2\nu\,g,_y - R_y\,g,_{xx}]$$

$$\sigma_{yy} = F_x\,[2\nu\,g,_x - R_x\,g,_{yy}] + F_y\,[2(1-\nu)g,_y - R_y\,g,_{yy}] \qquad (3.19)$$

$$\sigma_{xy} = F_x\,[(1-2\nu)g,_y - R_x\,g,_{xy}] + F_y\,[(1-2\nu)g,_x - R_y\,g,_{xy}]$$

or more concisely

$$\sigma_{ij} = S_{ijk} * F_k \qquad (3.20)$$

where S_{ijk} are the influence functions for stress.

In elasticity, boundary stresses are usually represented by a stress traction vector t_i . These components were defined in terms of the stress tensor in equation (3.15). By this definition, the Kelvin solution for stress traction is

$$t_i = \sigma_{ij} * n_j = T_{ij} * F_j \qquad (3.21)$$

where T_{ij} are the influence functions for stress traction.

Equations (3.18), (3.20) and (3.21) are expressions for the components of displacement, the stress tensor and stress traction (on some plane) at a point on an infinite plane, due to a concentrated point force. For simplicity, the equations for displacement and stress traction may be written collectively as

$$B_i = C_{ij} * F_j \qquad (3.22)$$

where

B_i = Displacement or stress traction component u_i, t_i

C_{ij} = Influence functions U_{ij}, or T_{ij}

F_j = Applied point force

Kelvin's solution for plane strain provides the basic analytical solution on which to develop this boundary element method for elastostatics. In fact, any fundamental singular solution of elasticity, such as Flamant's or Mindlin's solution, may be used in place of Kelvin's solution. However, Kelvin's solution was selected because it yields the most generally applicable boundary element formulation.

CHAPTER IV

Fictitious Stress Method

4.1 Introduction

The "Fictitious Stress" boundary element technique was first proposed by Massonet [17] in 1965 and has since been refined and generalized by researchers such as Banerjee [24] and Crouch and Starfield [30]. In this chapter, the fictitious stress method is developed with an emphasis on the physical interpretation of all mathematical derivations. Presently, this technique is only suitable for solving simple engineering problems because of the excessive computation required and the difficulty of formulating problems. However, in chapters 5 and 6 this fictitious stress method is used as a basis for deriving a new technique that can be used to solve complex problems easily and accurately.

4.2 Infinite Plane Model

Stress and displacement components obtained from Kelvin's solution are valid only when the region of interest is infinite in all directions and without cavities. However, Kelvin's solution may be applied to finite solids or infinite solids containing cavities by modelling these bodies as a portion of an infinite region.

Figure 4.1 illustrates the cross-section of a finite solid that is defined by a boundary C. Stress tractions and displacements are applied to C so that the solid is in a state of plane strain. Figure 4.1 also shows an infinite plane on which a tracing of the finite solid boundary C' is drawn. If the

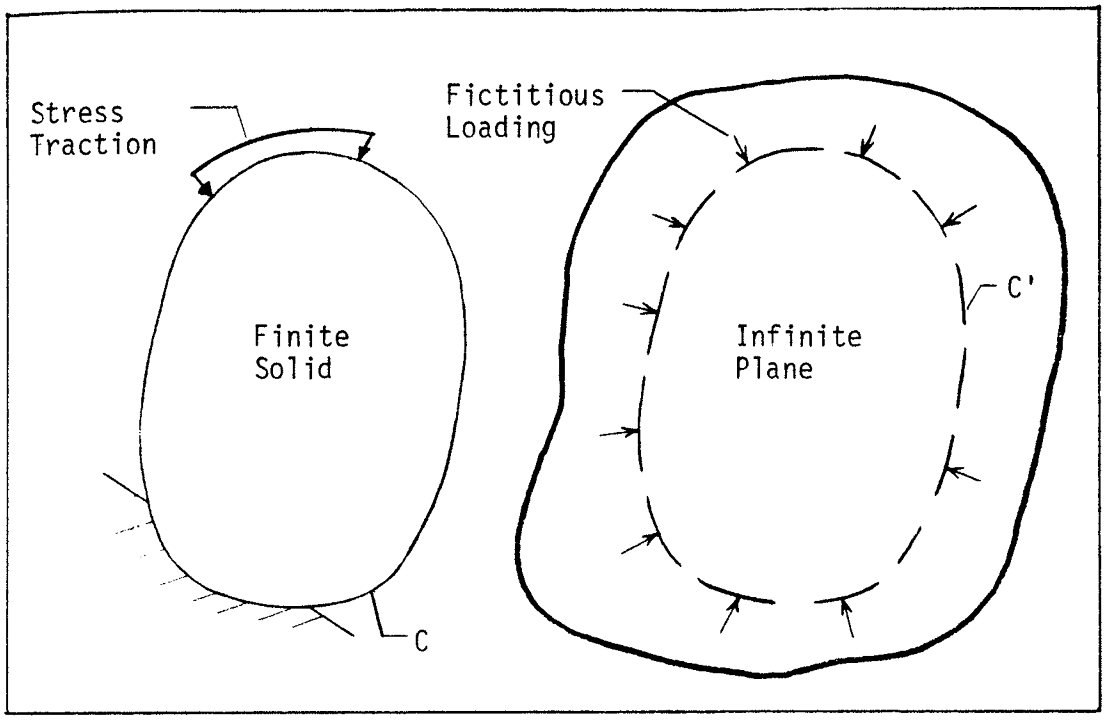


Figure 4.1: Infinite Plane Model

boundary conditions on C can be identically matched at all points on the tracing C', then the conditions in the interior of C' will duplicate those within the finite solid. The matching of interior conditions is guaranteed by the uniqueness of solutions to the biharmonic equation for a particular set of boundary conditions. Since the boundary conditions are reproduced, so must all other conditions, regardless of the extent of the region.

In order to develop an infinite plane model, a fictitious loading is applied to the infinite region along C'. This loading is selected to duplicate the effect of boundary conditions on C that create stress and displacement fields within the interior. The distribution of fictitous loading takes into account the material exterior to C' and therefore it will not be the same as to the real forces applied to the finite solid.

Once the stress field in the finite solid is duplicated within the tracing C', Kelvin's solution may be applied to the model to obtain displacement and stress tensor components at any point within the tracing. The loading applied to C' is called fictitious because it is not part of the real physical problem. A fictitious loading is used in the model to indirectly obtain a solution to the finite solid problem by replacing it with an equivalent infinite plane problem for which Kelvin's solution is applicable.

The infinite plane model may also be used to represent a finite or infinite solid containing cavities. Infinite solid problems arise in the analysis of mine shafts, wells and other rock mechanics applications. In this case, C' traces the boundary of the cavity and a fictitious loading is applied to duplicate the conditions on the cavity walls. Kelvin's solution is then applicable to the infinite plane model to determine the stress tensor and displacement components at any points outside the cavity.

4.3 <u>Fictitious Stress</u>

To create an infinite plane model it is necessary to determine the distribution of fictitious loading which will accurately reproduce a given set of boundary conditions. The attainable accuracy of Kelvin's solution is largely determined by how well the boundary conditions can be duplicated.

To select a fictitious loading distribution it is useful to divide the boundary tracing into a series of line segments called boundary elements. The order of loading distribution, which may be constant, linear, quadratic or higher, is then assumed over each individual element. By dividing the boundary into elements, the fictitious loading distribution around the entire tracing C' is represented by a piece-wise continuous function composed of a series of polynomial distributions, as shown in figure 4.2.

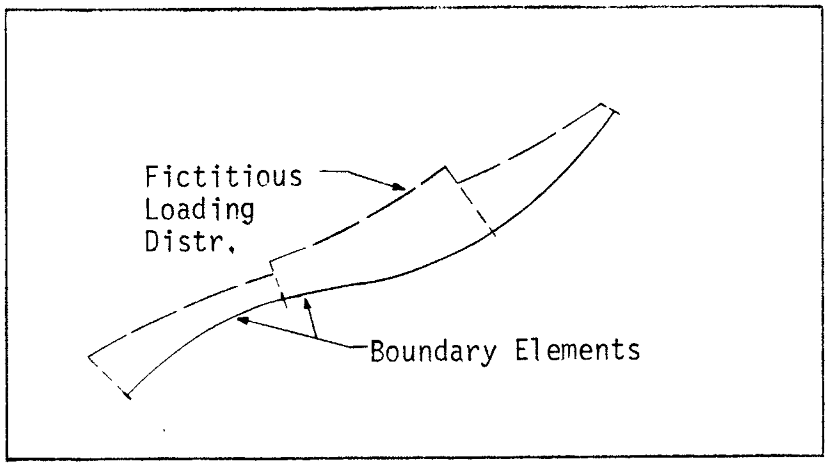


Figure 4.2: Piece-wise Distribution of Fictitious Loading

This discretization allows for a highly variable distribution of loading around the boundary tracing which is governed by the number of boundary elements and the order of distribution on each.

A suitable order of fictitious loading distribution depends mainly on the complexity of the boundary conditions (ie. the variability and discrete nature). Acceptable accuracy may be expected by choosing either a large number of elements and a low order loading distribution (such as constant) or a smaller number of elements with a higher order distribution over each. Both of these approaches will be considered.

Figure 4.3 illustrates an infinite region model that results from the first approach. A portion of an arbitrary solid is modelled by a series of straight line elements and a constant fictitious loading is applied over each. Assuming that a point force is equivalent to an infinitesimal portion of a stress distribution, then a point force F may be represented by

$$dF = \phi(s)\ ds \tag{4.1}$$

where $\phi(s)$ is a <u>fictitious stress</u> applied over source element b. Substituting this point force representation into Kelvin's solution and integrating the expression over element b yields

$$B_i = \int_b C_{ij}\ \phi_j(s)\ ds\ . \tag{4.2}$$

This integral form of Kelvin's solution yields stress and displacement components at a field point 'a' caused by a stress applied to a curvilinear element in the infinite plane. In effect, the stresses caused by a series of point forces along the element are superimposed. A problem for which the integral form of Kelvin's solution is applicable is illustrated in figure 4.4.

For a constant fictitious stress distribution over a straight source element, the term ϕ may be moved outside the integral so that equation (4.2) becomes

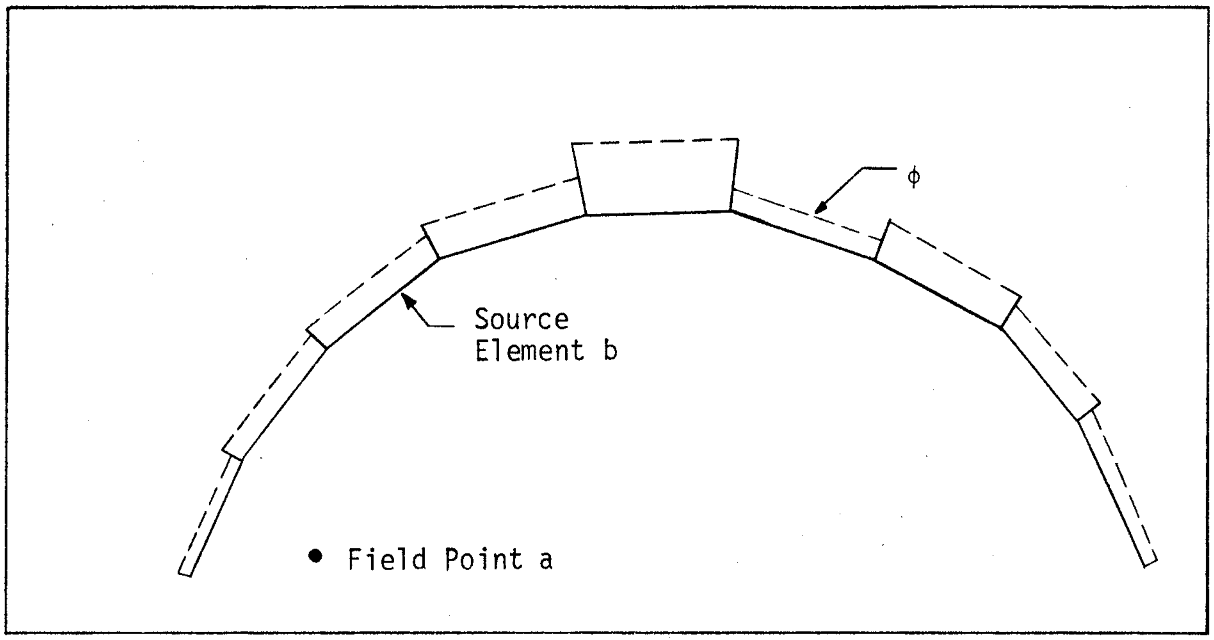


Figure 4.3: Constant Distribution of Fictitious Stress

$$B_i = \phi_j \int_b C_{ij}\, ds. \qquad (4.3)$$

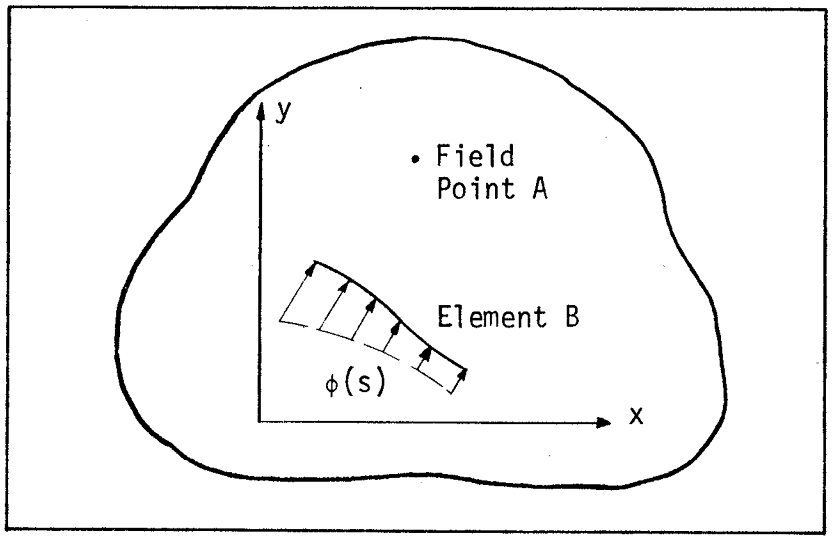


Figure 4.4: Integral Form of Kelvin's Solution

Letting $I_{ij} = \int_b C_{ij}\, ds$, the integral of the influence function, equation (4.3) may be written

$$B_i = \phi_j I_{ij} . \qquad (4.4)$$

Superimposing the contribution of every source element b to the stress on a point 'a' in the field gives

$$B_i(a) = \sum_{b=1}^{N} \phi_j I_{ij} . \qquad (4.5)$$

where N is the total number of boundary elements.

Equation (4.5) is an expression for the displacement and stress components at a point, due to a known fictitious stress distribution. However, to develop an infinite plane model a fictitious stress distribution must be determined from a known set of boundary conditions. If field point 'a' is moved to a point on a boundary element, then B becomes the known boundary condition and ϕ_i represents the unknown fictitious stress on that element. If field point 'a' is then moved to each and every element, a system of linear equations is assembled where the unknowns are the fictitious stress components on each element. The resulting system of 2N equations in 2N unknowns ϕ_i is of the form

$$\begin{aligned}
\overset{1}{B}_x &= \overset{11}{I}_{xx}\overset{1}{\phi}_x + \dots + \overset{1N}{I}_{xx}\overset{N}{\phi}_x + \overset{11}{I}_{xy}\overset{1}{\phi}_y + \dots + \overset{1N}{I}_{xy}\overset{N}{\phi}_y \\
&\vdots \\
\overset{N}{B}_x &= \overset{N1}{I}_{xx}\overset{1}{\phi}_x + \dots + \overset{NN}{I}_{xx}\overset{N}{\phi}_x + \overset{N1}{I}_{xy}\overset{1}{\phi}_y + \dots + \overset{NN}{I}_{xy}\overset{N}{\phi}_y \\
\overset{1}{B}_y &= \overset{11}{I}_{yx}\overset{1}{\phi}_x + \dots + \overset{1N}{I}_{yx}\overset{N}{\phi}_x + \overset{11}{I}_{yy}\overset{1}{\phi}_y + \dots + \overset{1N}{I}_{yy}\overset{N}{\phi}_y \\
&\vdots \\
\overset{N}{B}_y &= \overset{N1}{I}_{yx}\overset{1}{\phi}_x + \dots + \overset{NN}{I}_{yx}\overset{N}{\phi}_x + \overset{N1}{I}_{yy}\overset{1}{\phi}_y + \dots + \overset{NN}{I}_{yy}\overset{N}{\phi}_y
\end{aligned} \qquad (4.6)$$

where B_i are the displacement and stress traction boundary conditions and I_{ij} are the influence functions integrated over each source element.

When the field point is moved to an element where a displacement boundary condition is prescribed, then

$$B_i = u_i \quad \text{and} \quad I_{ij} = \int_b U_{ij}\, ds \,. \qquad (4.7)$$

However, when the field point is moved to a stress traction element the equation must be modified. Application of the fictitious stress layer causes a discontinuity in the variation of stress across the boundary. This generates a singularity in the stress traction equation, the limit of which is one half the fictitious stress on element 'a' (see page 110 of appendix for details). Thus for elements where a stress traction is prescribed

$$B_i = t_i \quad \text{and} \quad I_{ij} = \int_b T_{ij}\, ds \ . \tag{4.8}$$

except that $I_{ij} = 1/2$ when the field point corresponds to the source element b.

The equations of system (4.6) may be solved numerically using techniques such as Gaussian elimination or Cholesky's method to yield two components of fictitious stress for each element. Once these components are obtained, the finite solid boundary value problem is replaced by an equivalent infinite plane model. Displacement and stress tensor components can then be calculated by substituting the fictitious stress components into Kelvin's solution. The effect of all fictitious stress values on a point 'a' in the field is given by

$$u_i(a) = \sum_{b=1}^{N} \phi_j \int_b U_{ij}\, ds \quad , \quad \sigma_{ij}(a) = \sum_{b=1}^{N} \phi_k \int_b S_{ijk}\, ds \ . \tag{4.9}$$

4.4 Fictitious Stress Boundary Element Algorithm

The preceding boundary element method for elastostatic boundary value problems may be summarized in the following 5 steps;

1) Divide the boundary of the solid into straight line segments connected end to end.

2) Assume a constant fictitious stress is applied over each element. The mid-point co-ordinate may be used to represent the entire element.

3) Apply the integral form of Kelvin's solution to each element of the boundary and thereby assemble a system of linear equations in unknowns ϕ_j .

4) Solve numerically for ϕ_j .

5) Substitute ϕ_j into the integral form of Kelvin's soution to determine the stress tensor and displacement components at any desired points.

The fictitious stress method, or singularity method as it is called by some researchers, is part of a general group of "indirect" approaches. Other indirect methods include the "Displacement Discontinuity Method" of Crouch [25],which utilizes fictitious displacements, and the "Dislocation Dipole Method" of Altiero and Gavanzza [26]. Each of these indirect approaches is characterized by replacing the actual problem with an equivalent fictitious problem for which a singular solution is applicable. These boundary element researchers, however, have yet to develop advanced formulations using curved elements and high-order distributions of the fictitious quantity.

CHAPTER V

Isoparametric Elements

5.1 Introduction

The boundary element method developed thus far is based on several simplifying assumptions which limit the accuracy and usefulness of the method. By reviewing these assumptions the algorithm can be modified to improve the attainable accuracy and to greatly simplify the formulation of problems.

In the first step, the boundary of the finite region is modelled by a series of straight line elements. If the boundary of the region contains curves, a very large number of elements may be required to accurately model the shape. This would require a tedious problem formulation and excessive computer time and memory. An obvious improvement to the method would be to use curved elements that could more closely model the shape of a complex body. A quadratic element shape is especially convenient since one or two elements may be chosen to model each curve in the boundary, as shown in figure 5.1. Each element is defined by two end-point nodes and one mid-point node. Element equations would then take the form

$$y = ax^2 + bx + c \tag{5.1}$$

In the second step, a constant distribution of fictitious stress is assumed over each element. Assuming a higher order of distribution over each element could vastly improve the accuracy with which boundary conditions could be duplicated in the

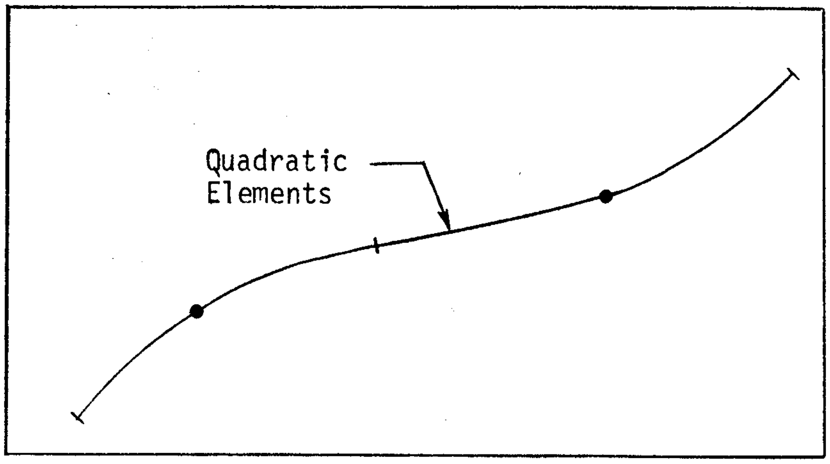


Figure 5.1: Quadratic Elements

infinite plane. This implies that the fictitious stress becomes a function of position, or

$$\phi = \phi(x,y) \; . \tag{5.2}$$

Step 3 of the algorithm assembles a set of linear equations in unknowns ϕ_i . An alternative procedure that can optimize the solution is called Galerkin's method and is introduced into the algorithm in chapter 6.

The implementation of curved elements and polynomial fictitious stress distributions can be unified by using the concept of shape functions.

5.2 <u>Shape Functions</u>

5.2.1 <u>Linear Interpolation</u>

It is now necessary to digress from the theory of boundary elements to develop a useful mathematical tool. It will then be demonstrated how shape functions can be incorporated into the existing theory to improve the current boundary element method.

Shape functions or, more descriptively, interpolation functions, are derived from interpolation methods, where some function value $f(\xi)$ is determined by assuming the distribution of the function between known values $f(\xi_i)$. An example of linear interpolation is illustrated in figure 5.2

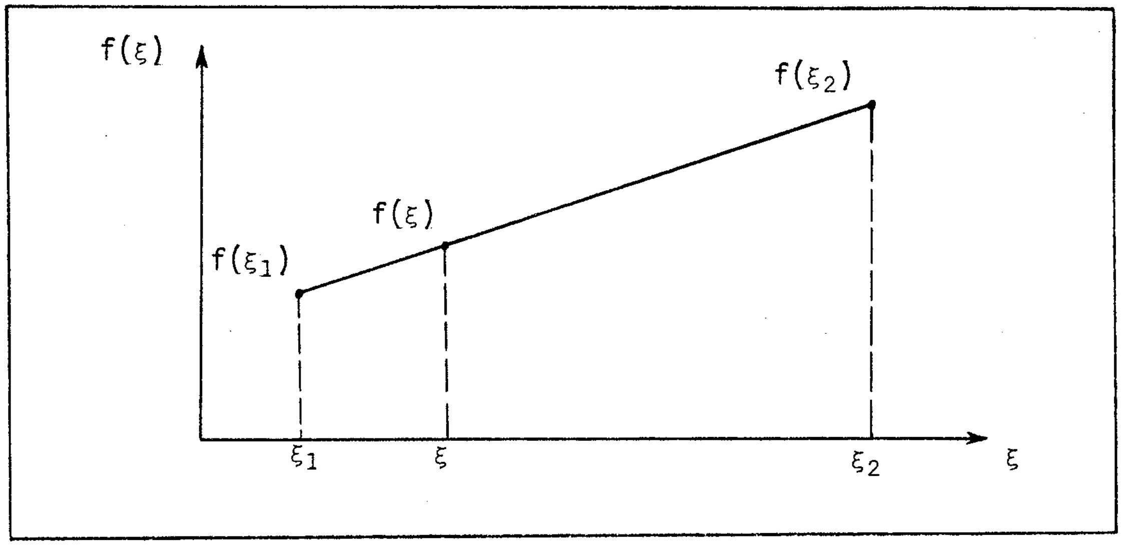


Figure 5.2: Linear Interpolation

The two end points, or nodal function values, $f(\xi_1)$ and $f(\xi_2)$ are known and the value corresponding to some point ξ in between is desired. Assuming a linear distribution of the function, two equivalent ratios of the co-ordinate lengths are found from similar triangles to be

$$\frac{\xi - \xi_1}{\xi_2 - \xi_1} = \frac{f - f_1}{f_2 - f_1} \qquad (5.3)$$

Solving equation (5.3) for $f(\xi)$, then

$$f(\xi) = \frac{\xi - \xi_1}{\xi_2 - \xi_1}(f_2 - f_1) + f_1 \qquad (5.4)$$

Rearranging equation (5.4) in order to isolate the nodal function values gives

$$f(\xi) = [1 - \frac{\xi - \xi_1}{\xi_2 - \xi_1}] f_1 + [\frac{\xi - \xi_1}{\xi_2 - \xi_1}] f_2 \qquad (5.5)$$

Equation (5.5) may be written in the form

$$f(\xi) = L_1 f_1 + L_2 f_2 \qquad (5.6)$$

where L_1 and L_2 are linear interpolation or shape functions and f_1 and f_2 are nodal values. A general form for any order of interpolation n is

$$f(\xi) = \sum_{i=1}^{n+1} \alpha_i(\xi)\, f_i \qquad (5.7)$$

where n+1 is the number of nodal values and $\alpha_i = L_i$ for linear interpolation. Note that the shape function values are equal to 1 at the home node (ie. node 1 for α_1) and zero at all other nodes. This is a property of all shape functions.

Now, assume that the fictitious stress values are known at the end-points, or nodes, of a series of line elements. Letting $f_i = \phi_i$ (the nodal fictitious stress values) in equation (5.7), then $f(\xi) = \phi(\xi)$ is a description of a linear distribution of fictitious stress over an element, as illustrated in figure 5.3. Notice that the variation over the boundary more closely resembles a continous distribution than the constant fictitious stress model.

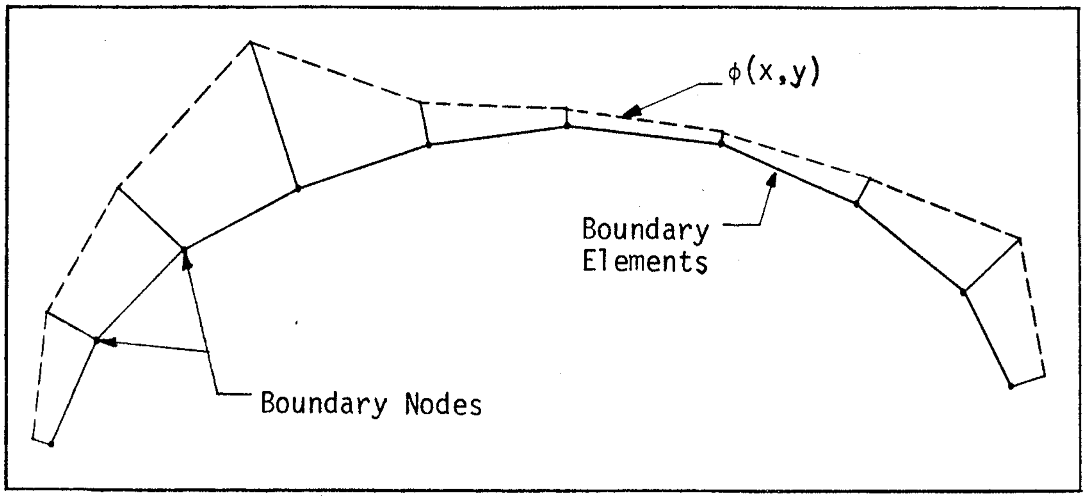


Figure 5.3: Linear Fictitious Stress Distribution

5.2.2 Lagrange Interpolation Formula

The Lagrange interpolation formula is a general expression for interpolation of any order, of which the linear interpolation functions of equation (5.5) are a special case. With this formula, fictitious stress distributions and curved elements of any order may be defined.

Assume that $\xi_1, \xi_2, \ldots, \xi_{n+1}$ are the co-ordinates at which the function values $f(\xi_i)$ are known. The interpolation functions $\alpha_i(\xi)$ of order n must have the property that

$$\begin{aligned} \alpha_i(\xi_j) &= 0 \qquad j \neq i \ ; \ j = 1, 2, 3, \ldots, n. \\ \alpha_i(\xi_i) &= 1. \end{aligned} \tag{5.8}$$

This requirement states that the functions α_i must be equal to 1 at the home point j and zero at all other points i. Therefore, a set of polynomials $\alpha_i(\xi)$ are required with roots

$$\xi_1, \xi_2, \ldots, \xi_{i-1}\ , \xi_{i+1}\ , \ldots, \xi_{n+1}\ .$$

Such a polynomial must be of the form

$$\alpha_i(\xi) = k(\xi - \xi_1)(\xi - \xi_2) \ldots (\xi - \xi_{i-1})(\xi - \xi_{i+1}) \ldots (\xi - \xi_{n+1}) \quad (5.9)$$

where k is selected such that $\alpha_i(\xi) = 1$. Evaluating this polynomial at a specific point ξ_i yields

$$\alpha_i(\xi_i) = k(\xi_i-\xi_1)(\xi_i-\xi_2) \ldots (\xi_i-\xi_{i-1})(\xi_i-\xi_{i+1}) \ldots (\xi_i-\xi_{n+1}) \quad (5.10)$$

Setting $\alpha_i(\xi_i) = 1$ in equation (5.10), as required by condition (5.8), and solving for k gives

$$k = \frac{1}{(\xi_i-\xi_1)(\xi_i-\xi_2) \ldots (\xi_i-\xi_{i-1})(\xi_i-\xi_{i+1}) \ldots (\xi_i-\xi_{n+1})} \quad (5.11)$$

Substituting this expression for k into the polynomial equation (5.9) gives the result

$$\alpha_i(\xi) = \frac{(\xi-\xi_1)(\xi-\xi_2)\ldots(\xi-\xi_{i-1})(\xi-\xi_{i+1})\ldots(\xi-\xi_{n+1})}{(\xi_i-\xi_1)(\xi_i-\xi_2)\ldots(\xi_i-\xi_{i-1})(\xi_i-\xi_{i+1})\ldots(\xi_i-\xi_{n+1})} \cdot \quad (5.12)$$

Equation (5.12) is the Lagrange interpolation polynomial of order n for any points i = 1 ... n. Note that any polynomial of order n may be formed by omitting the term involving ξ_i from both the numerator and denominator.

Consider a linear element defined by the normalized coordinate ξ as shown in figure 5.4. Linear shape functions may be formed by setting n = 1 in equation (5.12). The shape functions associated with node 1 and node 2 of figure 5.4 are found by setting i = 1 and i = 2, yielding

$$\alpha_1(\xi) = \frac{(\xi - \xi_2)}{(\xi_1 - \xi_2)} \tag{5.13}$$

$$\alpha_2(\xi) = \frac{(\xi - \xi_1)}{(\xi_2 - \xi_1)} . \tag{5.14}$$

For a normalized element $\xi_1 = 0$ and $\xi_2 = 1$ which reduces the functions to

$$\alpha_1(\xi) = \frac{(\xi - 1)}{(0 - 1)} = 1 - \xi \tag{5.15}$$

$$\alpha_2(\xi) = \frac{(\xi - 0)}{(1 - 0)} = \xi . \tag{5.16}$$

These correspond to the linear shape functions of equation (5.5) when $\xi_1 = 0$ and $\xi_2 = 1$.

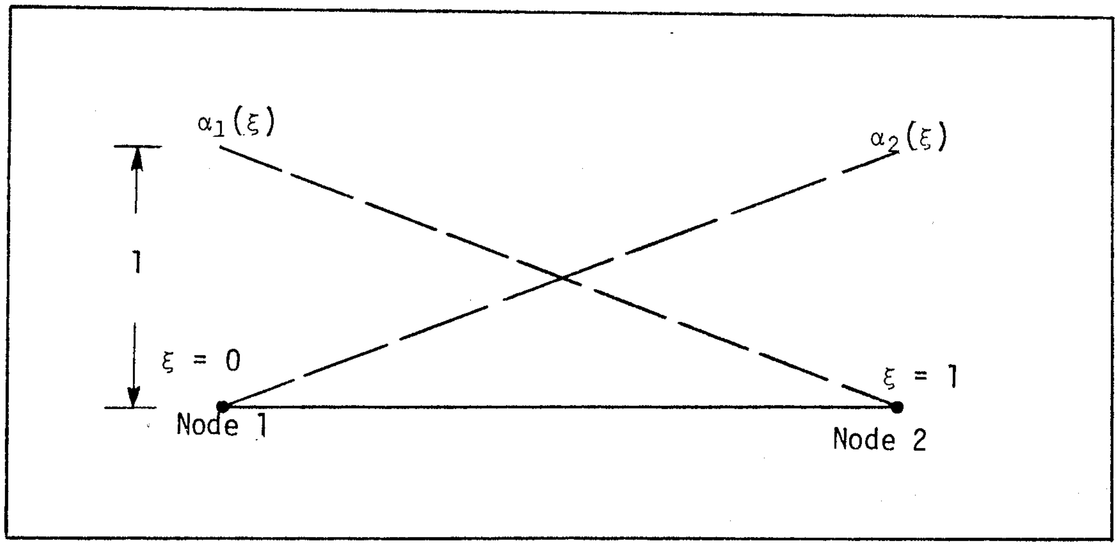


Figure 5.4: Linear Interpolation Functions

These functions are plotted along the element in figure 5.4. The linear variation of any function $f(\xi)$ may be defined by

$$f(\xi) = (1 - \xi)f(\xi_1) + \xi\, f(\xi_2) \tag{5.17}$$

where ξ is a normalized co-ordinate over the interval ξ_1-ξ_2.

Quadratic interpolation may be performed over the linear normalized element of figure 5.5. In this case, three nodes are required to define the quadratic functions. Quadratic interpolation functions are formed by setting n = 2 in equation (5.12) and setting i = 1, i =2 and i = 3, yielding

$$\alpha_1(\xi) = \frac{(\xi - \xi_2)(\xi - \xi_3)}{(\xi_1 - \xi_2)(\xi_1 - \xi_3)} \tag{5.18}$$

$$\alpha_2(\xi) = \frac{(\xi - \xi_1)(\xi - \xi_3)}{(\xi_2 - \xi_1)(\xi_2 - \xi_3)} \tag{5.19}$$

$$\alpha_3(\xi) = \frac{(\xi - \xi_1)(\xi - \xi_2)}{(\xi_3 - \xi_1)(\xi_3 - \xi_2)} \quad . \tag{5.20}$$

For the normalized element $\xi_1 = 0$, $\xi_2 = 1/2$ and $\xi_3 = 1$, which reduces these interpolation functions to

$$\alpha_1(\xi) = 2\xi^2 - 3\xi + 1 \tag{5.21}$$

$$\alpha_2(\xi) = 4(\xi - \xi^2) \tag{5.22}$$

$$\alpha_3(\xi) = 2\xi^2 - \xi \; . \tag{5.23}$$

The quadratic variation of any function $f(\xi)$ may be defined by

$$f(\xi) = \sum_{i=1}^{3} \alpha_i(\xi)\, f_i \quad . \tag{5.24}$$

where $\alpha_i(\xi)$ are given by equations (5.21) - (5.23).

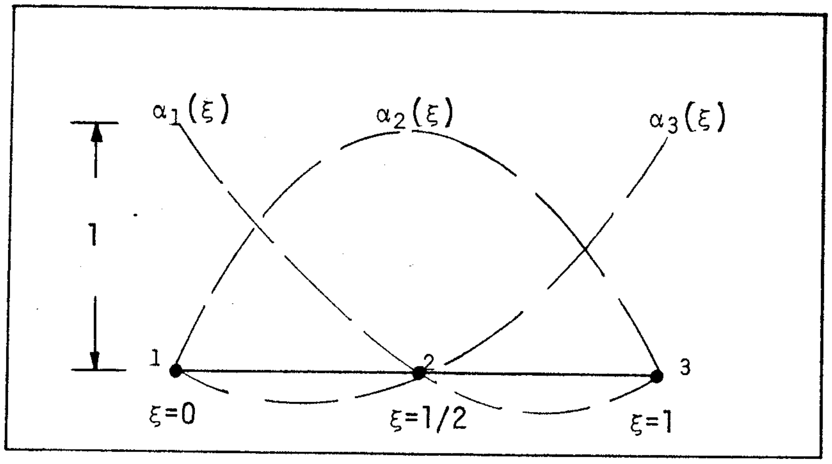


Figure 5.5: Quadratic Interpolation Functions

The shape functions for cubic or quartic interpolation may be obtained by setting n = 3 and n = 4 in equation (5.12) and reducing the expressions in a similar fashion. Note that an interpolation of order n will always require n+1 nodes distributed over the element.

5.2.3 Curved Elements

The shape functions (5.21) - (5.23) may be used to describe the curve of a boundary element in the x - y plane. Letting $f(\xi) = x$ and then $f(\xi) = y$ in equation (5.24) then the quadratic variation of the space co-ordinates over a normalized co-ordinate ξ are

$$x(\xi) = \sum_{i=1}^{3} \alpha_i(\xi)\, X_i \tag{5.25}$$

$$y(\xi) = \sum_{i=1}^{3} \alpha_i(\xi)\, Y_i \tag{5.26}$$

where X_i and Y_i are the nodal co-ordinates specified in some global co-ordinate system. Figure 5.6 illustrates the quadratic variation of the global co-ordinates x and y over the local co-ordinate ξ. This implies that a curved element in x - y space can be defined parametrically in terms of a normalized linear element in ξ space.

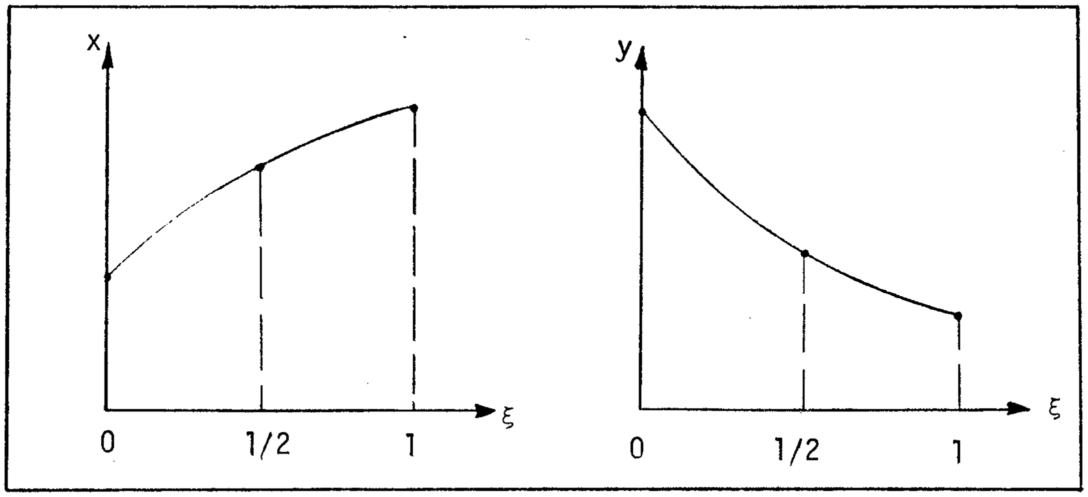


Figure 5.6: Quadratic Variation of x - y

The global co-ordinates of any point lying on a parabolic curve formed by the three nodes may be determined from equations (5.25) and (5.26). This is calculated by summing the product of the global nodal co-ordinates and the shape functions evaluated at the corresponding local nodal co-ordinate.

5.2.4 Fictitious Stress Distributions

The same shape functions (5.21) - (5.23) may be used to describe a quadratic variation of fictitious stress over an element. Letting the function $f(\xi)$ in equation (5.24) be the fictitious stress $\phi(\xi)$, the relation becomes

$$\phi(\xi) = \sum_{i=1}^{3} \alpha_i(\xi)\,\Phi_i \tag{5.27}$$

where Φ_i are the three nodal values and α_i are the three quadratic shape functions. Figure 5.7 illustrates the quadratic distribution of fictitious stress as a function of the normalized co-ordinate ξ . A specified value of ξ between 0 and 1 will correspond to a value of fictitious stress that lies on a quadratic variation between the end nodes. Using this equation form, a piece-wise quadratic distribution of fictitious stress may be specified around any boundary,which is composed of a series of quadratic distributions over each element.

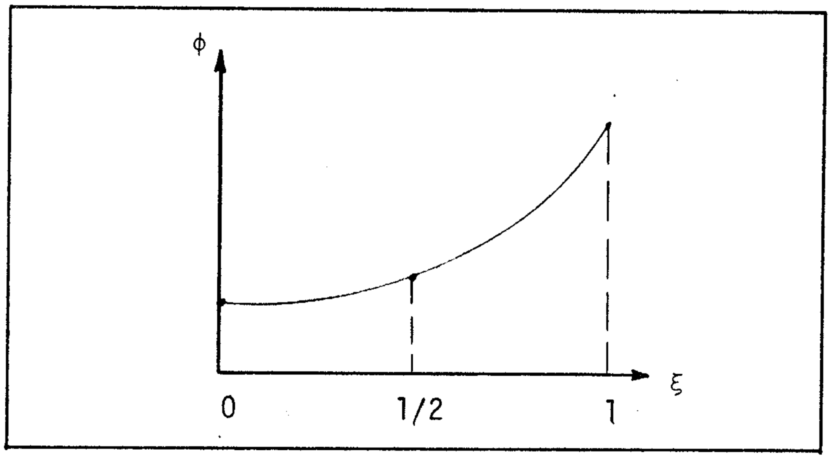


Figure 5.7: Quadratic ϕ Distribution

Recognize that defining both space co-ordinates and fictitious stresses in terms of shape functions implies a relationship between them. Figure 5.8 shows a curved element in the x - y plane and perpendicular to the plane is the quadratic distribution of fictitious stress over the element. Figure 5.8 illustrates that for any ξ there exists a unique value for x, y, and ϕ . Thus each value of the parameter ξ specifies a single point on an element and the applied fictitious stress at that point.

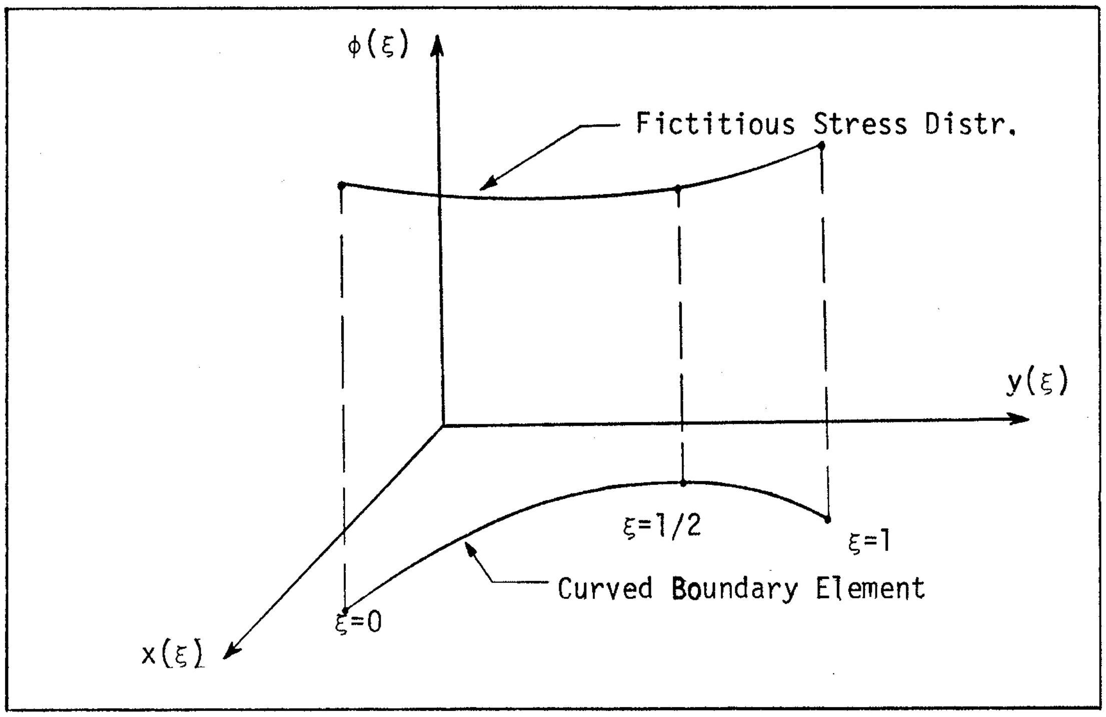


Figure 5.8: Relation Between x - y and ϕ

5.2.5 Shape Functions as a Solution Basis

The group of shape functions associated with a specific order of interpolation n form an orthogonal set. That is, no two or more shape functions of the same order are linearly dependent. Thus a linear combination of the shape functions α_i may be formed to approximate any function $f(\xi)$ by determining the necessary coefficients f_i . In this sense, equation (5.24)

$$f(\xi) = \sum_{i=1}^{3} \alpha_i(\xi)\, f_i \tag{5.24}$$

may be viewed as a quadratic functional expansion of $f(\xi)$ to three terms. Because $f(\xi)$ is formed from a linear combination of

α_i, all values of the function $f(\xi)$ must lie within the domain of the three shape functions. Thus the shape functions form a mathematical basis that defines the space in which $f(\xi)$ must be.

The essential task required to solve any boundary value problem using the boundary element method is to determine the distribution of fictitious stress. The ability of this distribution to accurately reproduce boundary conditions applied to the solid, governs the accuracy of the solutions. The quadratic variation of fictitious stress over an element takes the same form as equation (5.24), where

$$\phi(\xi) = \sum_{i=1}^{3} \alpha_i(\xi)\ \Phi_i \qquad (5.27)$$

This implies that the variation of fictitious stress is restricted to the domain of the shape functions. Thus, the shape functions define the basis of the boundary element solution. If the exact value of fictitious stress required lies outside this domain, then equation (5.27) can only provide an approximate solution.

Increasing the number of shape functions (and therefore the order of interpolation) will greatly increase the size of the solution domain. As a result, when solving a complex engineering problem, quadratic fictitious stress elements can provide a more accurate solution that constant and linear elements, and cubic elements can give a better solution than all three. This, however, is at the expense of much added computation. An acceptable compromise between the attainable accuracy and the computational expense is found using quadratic fictitious stress elements.

CHAPTER VI

Isoparametric Boundary Element Method

6.1 Introduction

In chapter 5, the mathematical foundations were established for a new type of boundary element formulation. It was shown that the variation of any function of order n may be represented by a set of n+1 shape functions and specific values of the function. Shape function representations are especially well suited to numerical applications because they are defined parametrically in terms of a local coordinate which always varies between 0 and 1. Therefore, a point along the local co-ordinate between 0 and 1 will correspond to a point along the function.

In this chapter it is shown that a shape function representation of the fictitious stress can be used to solve the integral equations of the boundary element method. In addition, integrations over the boundary can be performed more precisely by representing the space co-ordinates in terms of shape functions. Within this solution, the fictitious stress can be given any order of distribution over an element.

Galerkin's method of solving the integral equations is also presented in this chapter. This technique uses a different application of the shape function representation which results in an optimized solution.

These extensions to the boundary element method lead to a new method of accurately solving elastostatic problems which has not been attempted before.

6.2 Integral Equations

Recall from chapter 4 that stress and displacement components in an infinite plane are defined by

$$B_i = \int_b C_{ij} \, \phi_j \, ds \tag{6.1}$$

where ϕ is the fictitious stress distributed in some manner over a curvalinear element b. If the fictitious stress is allowed to vary over element b, then ϕ becomes a function of position. Since ϕ is no longer a constant over b, it can not be taken outside of the integral.

If field point 'a' is moved to any node on the boundary at which a displacement is applied, then equation (6.1) represents

$$u_i = \int_b U_{ij} \, \phi_j(x,y) \, ds \tag{6.2}$$

where u_i are components of the applied displacement and $\phi_j(x,y)$ is the unknown fictitious stress distribution over b. Since the unknown quantity ϕ appears within the integral, equation (6.2) takes the form of a Fredholm integral equation of the first kind [31]. If a stress traction is applied to field node 'a', then equation (6.1) represents

$$t_i = \int_b T_{ij} \, \phi_j(x,y) \, ds + \frac{1}{2} \Phi_i(a) \tag{6.3}$$

where $\Phi_i(a)$ is the fictitious stress at field node 'a'. In this expression, the unknown quantity appears within and outside of the integral so that equation (6.3) takes the form of a Fredholm integral equation of the second kind. Notice that these equations have the same form as those solved in chapter 4 except that ϕ_i was removed from the integral for mathematical convenience. This implies that the algorithm of chapter 4 is equivalent to a numerical solution of these integral equations. This was accomplished by reducing these integral equations to a system of linear equations with constant coefficients I_{ij} and unknowns ϕ_i .

The solution of integral equations (6.2) and (6.3) by forming a system of linear equations is no longer straight -forward because the unknown ϕ must remain within the integral. However, this difficulty may be overcome by incorporating shape functions into these equations.

6.3 Solution of Integral Equations

Consider the quadratic variation of fictitious stress over a curvalinear element b, given by

$$\phi_i(\xi) = \sum_{n=1}^{3} \overset{n}{\Phi}_i \alpha_n \qquad (6.4)$$

where α_n are three quadratic shape functions and $\overset{n}{\Phi}_i$ are the three nodal fictitious stress values on an element. Substituting this expression into integral equation (6.1) and expanding the summation yields

$$B_i = \int_b [\overset{1}{\Phi}_j \alpha_1 + \overset{2}{\Phi}_j \alpha_2 + \overset{3}{\Phi}_j \alpha_3] C_{ij} ds \qquad (6.5)$$

which may be separated into three integrals over b:

$$B_i = \int_b \overset{1}{\Phi}_j \alpha_1 C_{ij} ds + \int_b \overset{2}{\Phi}_j \alpha_2 C_{ij} ds + \int_b \overset{3}{\Phi}_j \alpha_3 C_{ij} ds . \qquad (6.6)$$

Recall from chapter 5 that Φ_1 , Φ_2 and Φ_3 are constant nodal values of fictitious stress. Therefore, these terms may be taken outside of the integrals, giving

$$B_i = \overset{1}{\Phi}_j \int_b \alpha_1 C_{ij}\, ds + \overset{2}{\Phi}_j \int_b \alpha_2 C_{ij}\, ds + \overset{3}{\Phi}_j \int_b \alpha_3 C_{ij}\, ds \quad (6.7)$$

or using the summation convention

$$B_i = \sum_{n=1}^{3} \overset{n}{\Phi}_j \int_b \alpha_n C_{ij}\, ds \; , \quad (6.8)$$

Superimposing the effects of all source elements b on field point 'a' leads to

$$B_i(a) = \sum_{b=1}^{N} \sum_{n=1}^{3} \overset{n}{\Phi}_j \int_b \alpha_n C_{ij}\, ds \; . \quad (6.9)$$

where the quadratic variation of fictitious stress over each element is guaranteed by incorporating the shape functions into the integral of the influence functions.

If the boundary condition at field node 'a' is a displacement then equation (6.9) represents

$$u_i(a) = \sum_{b=1}^{N} \sum_{n=1}^{3} \overset{n}{\Phi}_j \int_b \alpha_n U_{ij}\, ds \; . \quad (6.10)$$

If the boundary condition is a stress traction then equation (6.9) represents

$$t_i(a) = \sum_{b=1}^{N} \sum_{n=1}^{3} \overset{n}{\Phi}_j \int_b \alpha_n T_{ij}\, ds + \frac{1}{2}\, \Phi_i(a) \quad (6.11)$$

where the second term is the limit of the singularity in Kelvin's solution for stress traction. Letting

$$I^*_{ij} = \int_b \alpha_n C_{ij}\, ds$$

in equation (6.9), then

$$B_i(a) = \sum_{b=1}^{N} \sum_{n=1}^{3} \overset{n}{\Phi}_j \, I^*_{ij} \; . \quad (6.12)$$

Thus, integral equation (6.1) has been reduced to a linear equation in which I^*_{ij} is the constant coefficient and Φ_j is the unknown nodal fictitious stress.

If field point 'a' is moved to each and every node on the boundary, then equation (6.12) assembles a system of 2N linear equations in 2N unknowns Φ_j, similar in form to system (4.6). The resulting solution of this system is two components of fictitious stress Φ_x, Φ_y associated with each node on the boundary. The displacement or stress tensor components may now be determined at any points in the solid by substituting the nodal values of Φ into these integral Kelvin solutions:

$$u_i(a) = \sum_{b=1}^{N} \sum_{n=1}^{3} \Phi_j^n \int_b \alpha_n U_{ij} \, ds \tag{6.13}$$

$$\sigma_{ij}(a) = \sum_{b=1}^{N} \sum_{n=1}^{3} \Phi_k^n \int_b \alpha_n S_{ijk} \, ds \quad . \tag{6.14}$$

6.4 Isoparametric Form of the Integral

The integral in equation (6.12)

$$I^*_{ij} = \int_b \alpha_n C_{ij} \, ds \tag{6.15}$$

is a line integral which must be evaluated over each source element b. These elements may be straight or curved so that a general integration technique is required.

The differential ds of equation (6.15) can be approximated by the tangential straight line length

$$ds = \sqrt{(dx)^2 + (dy)^2} \quad . \tag{6.16}$$

The global co-ordinates x and y were defined in chapter 5 as varying quadratically over an element in terms of the parameter ξ by

$$x(\xi) = \sum_{i=1}^{3} X_i \, \alpha_i(\xi)$$
$$y(\xi) = \sum_{i=1}^{3} Y_i \, \alpha_i(\xi) \tag{6.17}$$

where X_i and Y_i are nodal co-ordinates. This definition implies that an equivalent expression for ds in terms of ξ would be

$$ds = \sqrt{(dx/d\xi)^2 + (dy/d\xi)^2} \quad d\xi \tag{6.18}$$

where $\sqrt{(dx/d\xi)^2 + (dy/d\xi)^2}$ is herein called the Jacobian of transformation $J(\xi)$. Substituting the shape function expressions into equation (6.18) gives

$$ds = J(\xi) \, d\xi = \sqrt{(ds_x)^2 + (ds_y)^2} \quad d\xi \tag{6.19}$$

where

$$ds_x = \sum_{i=1}^{3} (\partial\alpha_i/\partial\xi) \, X_i \quad , \quad ds_y = \sum_{i=1}^{3} (\partial\alpha_i/\partial\xi) \, Y_i \quad . \tag{6.20}$$

Recognize that x and y and the Jacobian J are all functions of ξ. The influence functions C_{ij} are expressions in terms of x and y and are therefore functions of ξ also. By definition, $\alpha_n = \alpha_n(\xi)$, so the integral of equation (6.15) may be written as

$$I^*_{ij} = \int_b \alpha_n(\xi) \, C_{ij}(\xi) \, J(\xi) \, d\xi \tag{6.21}$$

where all quantities are integrated with respect to parameter ξ . Because equation (6.21) is a function of only one parameter, this expression is said to be in <u>isoparametric form</u>.

This isoparametric form lends itself well to numerical integration techniques such as Simpson's rule and Gaussian quadrature because all integration can be performed over a normalized linear element and transformed into integration over a curvilinear element by the Jacobian J. A Gaussian quadrature formulation for this integral is shown in chapter 7 and a presentation of suitable quadrature formula is provided on page 107 of the appendix.

6.5 Isoparametric Boundary Element Algorithm

The preceding isoparametric boundary element method for elastostatic boundary value problems may be summarized in the following 5 steps:

1) Divide the region boundary into quadratic line segments connected end to end with each element defined by three adjacent nodes.

2) Assume a quadratic variation of fictitious stress over each element using shape functions.

3) Apply the integral form of Kelvin's solution to the boundary in order to assemble a system of linear equations where the unknowns are the nodal fictitious stress components Φ_i.

4) Solve the system of linear equations for Φ_i.

5) Substitute Φ_i into the integral Kelvin's solution to calculate the displacement and stress tensor components at any desired points in the solid.

These 5 steps describe a modified fictitious stress method which affords many advantages over the linear element algorithm presented in chapter 4. The equations were derived for any order of shape functions so that a linear, quadratic, cubic or quartic model of any physical problem may be constructed by selecting the appropriate shape functions.

6.6 Galerkin's Method of Solution

The isoparametric boundary element method can be optimized by incorporating an alternative numerical solution to integral equation (6.1). Galerkin's method [32] is a numerical technique for finding an approximate solution to a problem posed in the form

$$L\,\omega = g \tag{6.22}$$

where L is a linear operator (such as integration or differentiation), g are known quantities and ω are unknown functions. Assuming the function ω can be expanded as a linear combination of functions μ_i , then

$$\omega = \sum_{i=1}^{\infty} A_i \; \mu_i \quad . \tag{6.23}$$

where A_i are coefficients to be determined. An approximation to ω is obtained from

$$\omega_a = \sum_{i=1}^{n} A_i \; \mu_i \tag{6.24}$$

where n is a finite number.

Equation (6.22) may be written in the form

$$L\,\omega - g = 0 \quad . \tag{6.25}$$

Substituting the approximation ω_a into equation (6.25) yields

$$R = L\left(\sum_{i=1}^{n} A_i\,\mu_i \right) - g \tag{6.26}$$

where R is defined as the residual of the approximation and represents the error in ω_a. Since summation is a linear operator and A_i are constants, an alternative form of this expression is

$$R = \sum_{i=1}^{n} A_i\,(L\,\mu_i) - g \quad . \tag{6.27}$$

To optimize the solution the residual R must be forced to zero. Notice that R is composed of a linear combination of μ_i (minus a constant g) and must therefore be linearly dependent on μ_i .

Galerkin's method is applied by setting the inner product of R with each μ to zero:

$$< R, \mu_i > = \int_d R\,\mu_i\,ds = 0 \quad . \tag{6.28}$$

This inner product forces R to be orthogonal to all μ_i . Two functions can be both orthogonal and linearly dependent only if one function or both are zero. Clearly R is linearly dependent on μ_i and therefore, to be orthogonal, R must be forced to zero. However, since μ_i is not a complete set of expansion functions, R will not actually approach zero. Rather, R will be minimized so that the best solution possible will be obtained.

Recall that the quadratic distribution of fictitious stress over an element was given in terms of shape functions as

$$\phi_i = \sum_{n=1}^{3} \Phi_i^{\,n}\,\alpha_n \quad . \tag{6.29}$$

This expression is equivalent in form to the functional expansion (6.24) where $\phi_i = \omega_i, \alpha_n = \mu_i$ and $\Phi_i = A_i$. Incorporating this distribution into Kelvin's solution resulted in the linear equation

$$B_i(a) = \sum_{b=1}^{N} \sum_{n=1}^{3} \Phi_j^n \int_b \alpha_n C_{ij} \, ds \qquad (6.9)$$

which may be rewritten as

$$R = \sum_{b=1}^{N} \sum_{n=1}^{3} \Phi_j^n \int_b \alpha_n C_{ij} \, ds - B_i(a) \quad . \qquad (6.30)$$

Applying Galerkin's method as in equation (6.28) requires

$$< R, \alpha_n > = 0 \quad . \qquad (6.31)$$

Substituting the residual into the inner product and rearranging gives

$$\int_a \alpha_m B_i(m) \, ds_a = \int_a \alpha_m \sum_{b=1}^{N} \sum_{n=1}^{3} \Phi_j^n \int_b \alpha_n C_{ij} \, ds_b \, ds_a \qquad 5.32)$$

$$m = 1, 2, 3$$

where 'a' now represents a field element. Because Φ_i are constant nodal values, they may be taken outside of the integral so the relation becomes

$$\int_a \alpha_m B_i(m) \, ds_a = \sum_{b=1}^{N} \sum_{n=1}^{3} \Phi_j^n \int_a \alpha_m \int_b \alpha_n C_{ij} \, ds_b \, ds_a \qquad (6.33)$$

$$m = 1, 2, 3$$

If field element 'a' is moved to each and every element on the boundary, then equation (6.33) assembles a system of linear equations in unknowns Φ_i. Upon solution of the system, the values of Φ may be substituted into equations (6.13) and (6.14) to calculate displacement and stress tensor components at any prescribed field points.

Recall that the previous isoparametric algorithm matches the conditions on the boundary of the real problem with those on the tracing in the infinite plane at the nodes only. Conditions on the boundary between the nodes are allowed to vary freely, though it is assumed that there are no large oscillations. Galerkin's method offers an improvement in results by giving the boundary conditions and the fictitious stress the same order of distribution. Instead of matching discrete points exactly, Galerkin's method approaches the exact conditions on the boundary of the solid at every point. Figure 6.1 illustrates the difference between point matching the two sets of boundary conditions and Galerkin's method of evenly distributing the error.

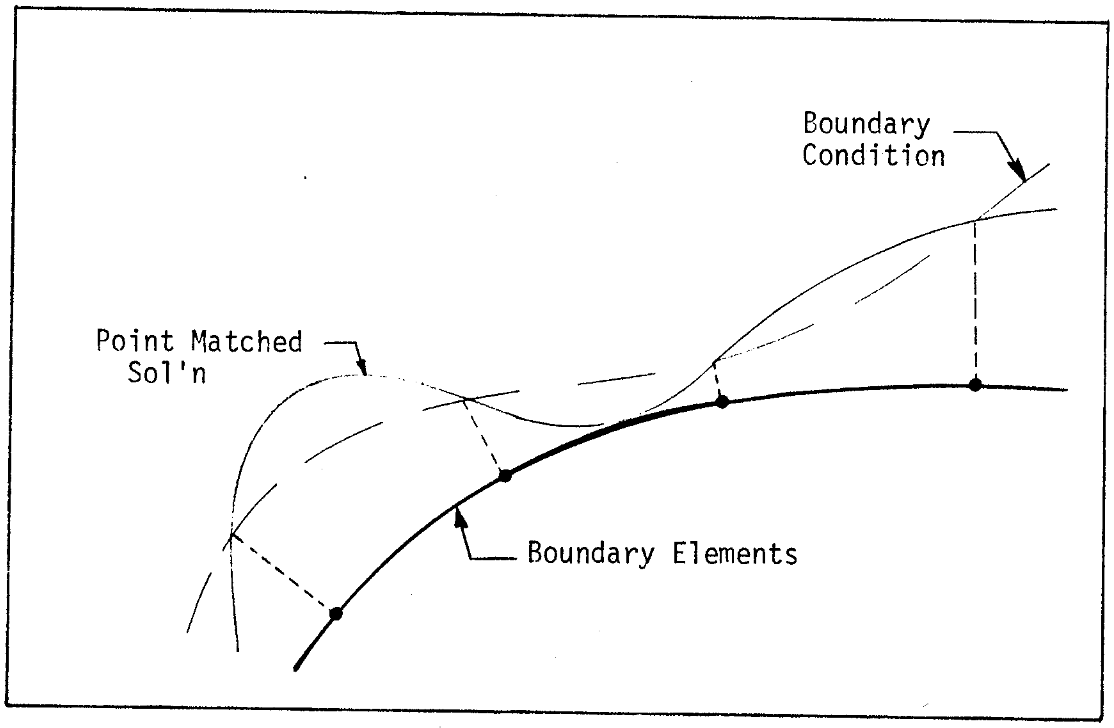


Figure 6.1a : Point Matching

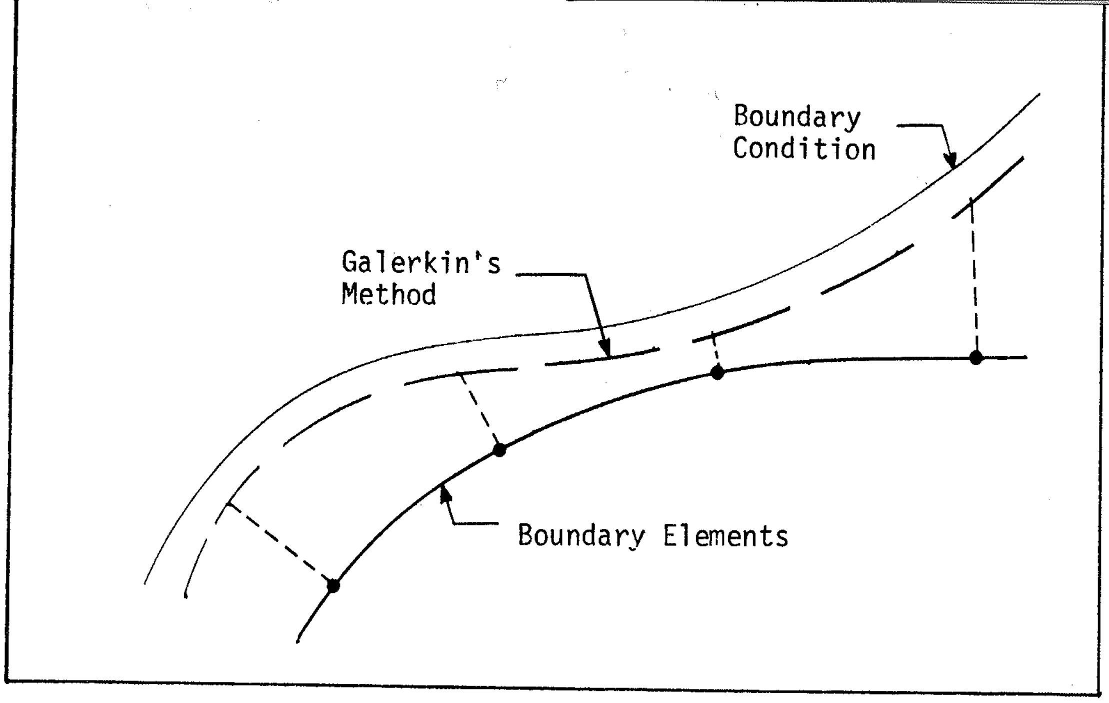


Figure 6.1b : Galerkin's Method

Galerkin's method, in general, will not give an exact solution because the set of expansion functions μ_i is incomplete. It was stated in section 5.2.5 that the shape functions define the domain in which the approximate solution must lie. If the exact solution lies outside of this domain, Galerkin's method will determine the projection of the exact solution on the shape function domain. As an illustration, consider the case when there are two shape function α_1 and α_2 which define a plane in which all boundary element solutions must be found (see figure 6.2). Point A is the location of the exact solution which does not lie in the α_1-α_2 plane because it contains a component in the z-direction. Galerkin's method will select point A' as a best approximation to A within the domain of the shape functions. The length AA' is the smallest distance between point A and the α_1-α_2 plane and represents the magnitude of the residual R.

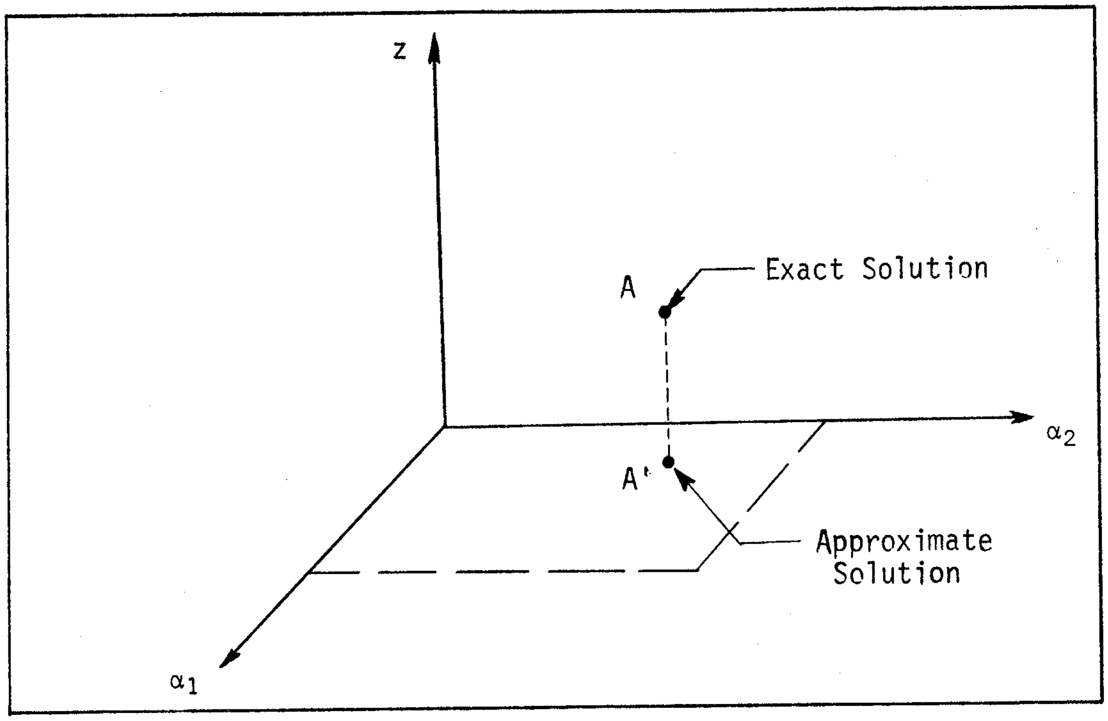


Figure 6.2: Optimization Using Galerkin's Method

CHAPTER VII

Numerical Implementation

7.1 Introduction

The isoparametric boundary element theory is designed to be readily programmable. This theory may be programmed in any number of ways, depending on the flexibility, accuracy and speed required.

In this chapter, the isoparametric boundary element program BEAST (for Boundary Element Analysis of STress) is described. This program uses curved elements and quadratic fictitious stress distributions, and the point matching technique to assemble the linear equations. The theory has been successfully programmed in FORTRAN, BASIC and Pascal and used to solve several theoretical elasticity problems. The results from these tests are presented in order to demonstrate the validity of the theory and to evaluate the attainable accuracy and speed of the program.

7.2 "BEAST" - An Isoparametric Boundary Element Program

7.2.1 Program Structure

The boundary element program BEAST consists of 7 subroutines plus a control program. These subroutines perform the following functions:

i) Read in the input file

ii) Initialize the integration data and shape functions

iii) Generate Jacobians and outward normals

iv) Generate influence coefficient matrix

v) Solve linear equations

vi) Calculate the stress tensor and displacements

vii) Print-out results

In addition there are two graphics routines which are used to draw the boundary element mesh and to graph the solution on the screen.

The TURBO Pascal version of BEAST, without the graphics routines, occupies about 23k of memory on a diskette. The amount of available RAM required to run the program depends on the number of nodes N, since a 2N x 2N matrix is generated by the program. A listing of BEAST is contained in the appendix on page 113 .

The function of the first subroutine is to read in the parameters which describe the boundary element model, the boundary conditions and the required solution points. Subroutines 2, 3 and 4 are used to evaluate the isoparametric integral

$$I^{*}_{ij} = \int_{b} \alpha_n(\xi)\, C_{ij}(\xi)\, J(\xi)\, d\xi \ . \qquad (6.21)$$

Within this task, subroutine 2 initializes the Gaussian quadrature integration data, subroutine 3 determines $\alpha_n(\xi)$ and $J(\xi)$, and subroutine 4 calculates $C_{ij}(\xi)$ and integrates the product over the boundary elements.

Subroutine 5 is a linear equation solver which solves the system formed by the coefficient matrix and the boundary condition vector. Finally, the last two subroutines calculate the stress tensor and displacement components at the prescribed points and write the results to a file.

7.2.2 Input File

The input file contains the information necessary to accurately describe a boundary value problem. The program BEAST is accompanied by an interactive data preparation program called BEDAP (for Boundary Element DAta Preparation) which prompts the user for the information and creates a proper input file. A listing of BEDAP is contained in the appendix on page 137.

The data required to solve a problem is divided into four parts. The first section consists of the control data and material properties including the following:

i) Title of the problem
ii) Name of the output file
iii) Number of boundary nodes
iv) Young's modulus E
v) Poisson's ratio ν

The second section contains the x and y co-ordinates of the nodes. For a contour which describes the exterior boundary of a problem the nodes must be ordered in a clockwise manner. Conversely, the nodes on the boundary of a cavity must be ordered counter-clockwise. The order of the nodes is used by the program to determine the direction of the unit outward normal, which is contained in Kelvin's solution for stress traction.

The third section contains the components of the boundary conditions applied to each node. These must be either a stress traction or a displacement. Finally, the last section consists of the co-ordinates of points in the field at which solutions are required.

When the program BEAST is run, all input information is echoed to the output file. To ensure the location of the nodes will accurately describe the shape of the boundary, BEAST will draw the model on the screen. The drawing routine uses quadratic shape functions to interpolate points between the nodes as is done by the calculations within the program. Therefore, this picture is an accurate description of how the program interprets the shape of the boundary from the prescribed nodal co-ordinates.

A more detailed presentation of how to model a physical problem using boundary elements is presented in section 7.3.

7.2.3 Integration Data

The isoparametric integral of equation (6.21) is evaluated by BEAST using two types of Gaussian quadrature. Integration over each element is performed over the interval 0 to 1 and scaled to account for the actual size and shape of the element by the Jacobian J.

When integrating Kelvin's solution over the element which contains the field node 'a' as a mid-point node, care must be taken to avoid this point as a sampling point of the Gaussian quadrature. At this point Kelvin's solution contains a singularity which can not be integrated well using Gaussian quadrature. Therefore, 5 point quadrature is used over two intervals, one on either side of the field node. The two solutions are then summed to yield the integral over the entire element. To complete the integral, the limit of the singularity must be added, since this portion of the integral was avoided. The limit of the displacement singularity is 0 and the limit of the stress traction singularity is 1/2. All other integrals are evaluated using 10 point quadrature over the entire element.

In order to use Gaussian quadrature for evaluating the integral (6.21), the shape functions and their derivatives are evaluated at all Gauss sampling points for both 5 and 10 point quadrature.

7.2.4 Jacobians

In the third subroutine, the global co-ordinates of all Gauss sampling points (or Gauss points) are calculated for use in evaluating the isoparametric integral. These are interpolated from the nodes assuming a quadratic curve of the element. Co-ordinates are determined from

$$x(\xi) = \sum_{i=1}^{3} \alpha_i(\xi)\, X_i \tag{5.25}$$

$$y(\xi) = \sum_{i=1}^{3} \alpha_i(\xi)\, Y_i \tag{5.26}$$

where X_i and Y_i are the global co-ordinates of the three nodes which compose an element and $\alpha_i(\xi)$ are the three quadratic shape functions evaluated at the Gauss points ξ.

In addition, the Jacobians are determined for every element and evaluated at each Gauss point for use in the isoparametric integral. These are calculated from

$$J(\xi) = \sqrt{(\Sigma\beta_i X_i)^2 + (\Sigma\beta_i Y_i)^2} \qquad (7.1)$$

where β_i are the derivatives of the shape functions α_i with respect to ξ. Note that one set of co-ordinates and Jacobians is required for each type of Gaussian quadrature.

The unit outward normals are also initialized in this routine for use in the stress traction equations. The normals are calculated from the Jacobians using

$$n_x = -\frac{\Delta y}{J_n} \quad , \quad n_y = \frac{\Delta x}{J_n} \qquad (7.2)$$

where J_n are the Jacobians evaluated at the nodes and Δy , Δx are the rise and run between adjacent nodes.

7.2.5 Influence Coefficient Matrix

After reading in the problem and initializing the integration data, the program begins to assemble the system of linear equations. Two equations are formed for each node on the boundary, one for the x-component and one for the y-component.

Firstly, node 1 becomes the field node and the isoparametric integral is evaluated over each and every element around the boundary. If a boundary condition at the field node is a displacement, then the following Gaussian Quadrature formulation is used:

$$I^*_{ij} = \sum_{\xi=1}^{10} Wt(\xi)\, \alpha_n(\xi)\, U_{ij}(r)\, J(\xi) \qquad (7.3)$$

where Wt are the weights for integration, ξ are the Gauss sampling points , n is the source node (1,2,3) and r is the distance from the field point to the Gauss point. Equation (7.3) represents the influence of one source element on the field node. The integral is evaluated over each element using this formulation except the element which contains the field node as a mid-point node. For this case, the integration formula is

$$I^*_{ij} = \left(\sum_{\xi'=1}^{5} Wt(\xi') \, \alpha_n(\xi') \, U_{ij}(r') \, J(\xi') \right) + \left(\sum_{\xi'=6}^{10} Wt(\xi') \, \alpha_n(\xi') \, U_{ij}(r') \, J(\xi') \right) \tag{7.4}$$

where ξ' denotes two sets of 5 point Gaussian quadrature sampling points, one from 0 to 0.5, the other from 0.5 to 1.

Next, node 2 becomes the field node and if a component of the boundary condition is a stress traction, then the Gaussian quadrature formulation is

$$I^*_{ij} = \sum_{\xi'=1}^{10} Wt(\xi) \, \alpha_n(\xi) \, T_{ij}(r) \, J(\xi) \quad . \tag{7.5}$$

For the case when 'a' is a mid-point node on a stress traction element, the integration is performed over the two intervals using

$$I^*_{ij} = \left(\sum_{\xi'=1}^{5} Wt(\xi') \, \alpha_n(\xi') \, T_{ij}(r') \, J(\xi') \right) + \left(\sum_{\xi'=6}^{10} Wt(\xi') \, \alpha_n(\xi') \, T_{ij}(r') \, J(\xi') \right) + 1/2 \tag{7.6}$$

where the 1/2 accounts for the integral over the singular point.

The field node is moved to each and every node on the boundary and the solution of the integrals I^*_{ij} are stored in a matrix. The final system of equations is stored in the form

$$\left|\begin{array}{c|c} I_{xx} & I_{xy} \\ \hline I_{yx} & I_{yy} \end{array}\right| \begin{bmatrix} \Phi_x \\ \hline \Phi_y \end{bmatrix} = \begin{bmatrix} B_x \\ \hline B_y \end{bmatrix} \qquad (7.7)$$

where Φ_x , Φ_y represents all of the x and y components of the unknown fictitious stress, and B_x and B_y are the corresponding boundary condition components at the nodes.

7.2.6 Solving Linear Equations

The program BEAST uses a method attributed to Cholesky [33] to solve the system of linear equations (7.7) for the fictitious stress components. This method has proven to be the most economical of all elimination methods in terms of computer time and memory.

Cholesky's method is useful for solving a problem of the form

$$A\ x = b \qquad (7.8)$$

where A is a square matrix of coefficients, b are given numbers and x is a vector of unknowns. The matrix A can be decomposed into the product of two matrices such that

$$A = L\ U \qquad (7.9)$$

where L and U are lower and upper triangular matrices. Thus, equation (7.8) may be represented by

$$L\ U\ x = b. \qquad (7.10)$$

Premultiplying both sides of (7.10) by L^{-1} gives

$$U\,x = z \qquad \text{where} \quad z = L^{-1}\,b\,. \tag{7.11}$$

The system of equation (7.8) is solved by first determining z from

$$L\,z = b \tag{7.12}$$

and then determining x from

$$U\,x = z\,. \tag{7.13}$$

Since L and U are triangular matrices these equations may be solved directly through back substitution. Therefore, no row operations are required.

Cholesky's method is especially economical of computer memory because the U and L matrices may overlay the A matrix storage locations. This may be done since there is no need to store the zeros and ones of the U and L matrices. Therefore, no extra arrays in addition to A, x and b are required to solve a system of linear equations.

7.2.7 Obtaining Field Solutions

Once the fictitious stress components at each node have been determined, the problem is essentially solved. The stress tensor and displacement components can be calculated at any point in the field by integrating Kelvin's solution evaluated at the point. These are determined from

$$u_i(a) = \sum_{b=1}^{NE} \sum_{\xi=1}^{10} Wt(\xi)\ \alpha_n(\xi)\ U_{ij}(a)\ \Phi_j(a)\ J(\xi)$$

$$\sigma_{ij}(a) = \sum_{b=1}^{NE} \sum_{\xi=1}^{10} Wt(\xi)\ \alpha_n(\xi)\ S_{ijk}(a)\ \Phi_k(a)\ J(\xi) \qquad (7.14)$$

where NE represents the number of elements. These equations sum the effect of the fictitious stress distributions on each element upon a point 'a' in the field.

7.3 Boundary Element Models

Boundary element models of physical problems are much simpler and more versatile than models used in finite element techniques. Thus, the important advantage of using boundary elements, and in particular isoparametric elements, is that a minimum amount of information is required to solve a physical problem accurately.

Once a problem is posed in the form of a boundary value problem, a suitable model must be created. The information required to formulate and solve a problem using the isoparametric boundary element technique is as follows:

i) The location of points, within an arbitrary co-ordinate system, which are indicative of the general shape of the boundary(s).

ii) The components of the boundary conditions at each of these points (displacements and stress tractions).

iii) Properties of the material (E, ν).

iv) The location of points at which a solution is required.

The quality of a boundary element solution may be increased by increasing the number of boundary nodes (until affected by computer round-off error). However, for most problems the number of points required to describe the shape and the variation of boundary conditions is a sufficient number of nodes to obtain an acceptable solution. The optimum number of nodes may then be found by increasing the density of nodes until the solutions converge to a single set of values.

As an illustration of a boundary element model, consider the case of a bracket used to connect two members of a linkage, as shown in figure 7.1. The outer halves of each hole are subjected to a specified distribution of force. The bracket is made of steel with material properties E = 30,000 ksi, ν = 0.3 and a

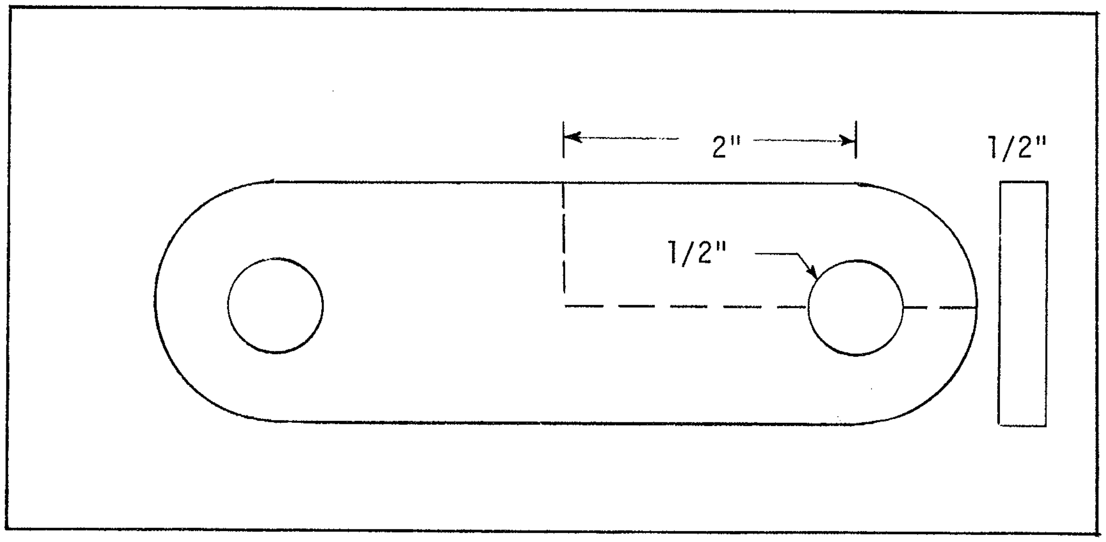


Figure 7.1: Connecting Bracket

force of 100 lbs. is applied.

Before a boundary element model can be created, the problem must first be posed as a boundary value problem. The problem contains two axes of symmetry so that only one quarter need be considered. Next a suitable co-ordinate system must be selected, which is chosen to originate at the centre of the hole.

Boundary conditions must then be determined from the applied loading. Assuming the force is applied uniformly to a contact area of 1/4 the hole circumference then

$$t = P/\pi rd\,(1/2) = 100/\pi(1/2)(1/2)(1/2) = 254.6 \text{ psi}$$

is the stress traction applied radially over 1/4 of the hole. The link is allowed to move freely along the axes of symmetry but is constrained across these lines, and the remainder of the boundary is traction free. Thus, the boundary value problem becomes as shown in figure 7.2.

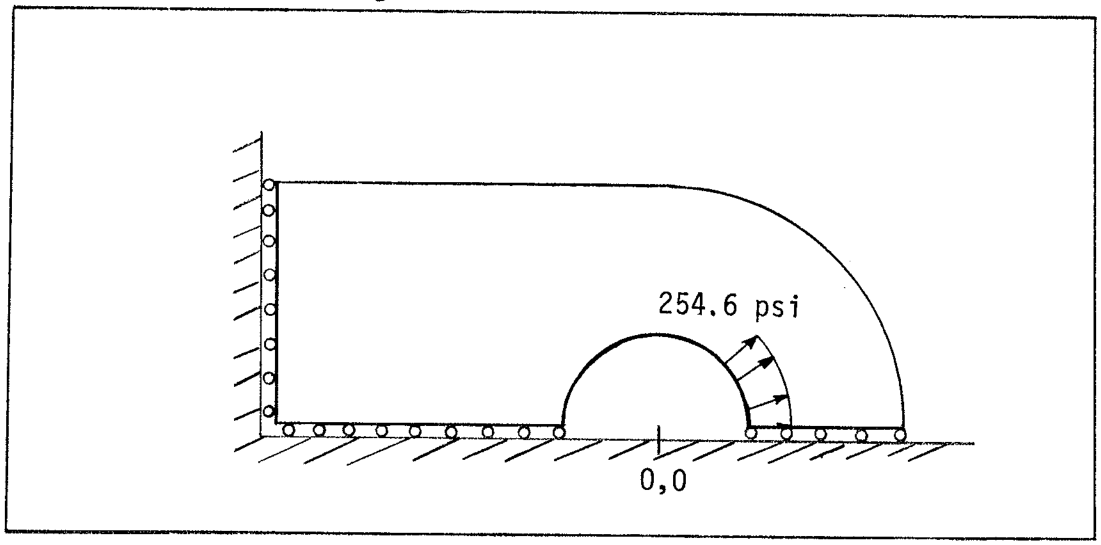


Figure 7.2: Boundary Value Problem Representation

To create a boundary element model, points on the boundary must be selected which are indicative of the shape of the bracket. Each isoparametric element consists of three nodes which represent a parabola. Therefore, an element should not be bent around a right angle since this is not well modelled by a parabola. Given this requirement, the minimum number of nodes needed to describe this problem is 14 (ie. 7 elements), as shown in figure 7.3.

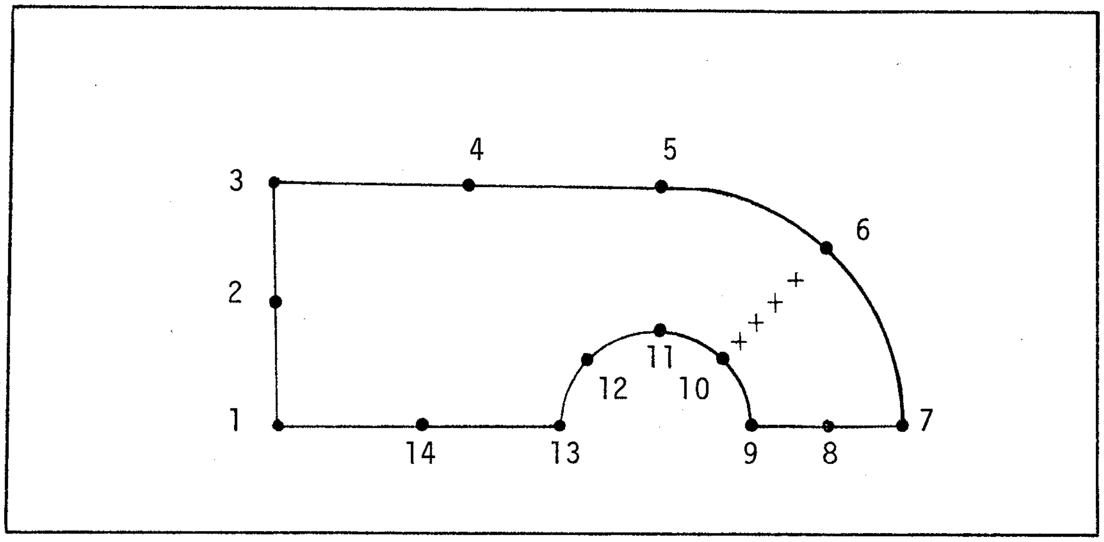


Figure 7.3: Boundary Element Model

The nodes are numbered in a clockwise order to specify an external boundary. The co-ordinates of these points constitute the nodal information. Notice that it is imperative to place a node at the junction of two different boundary conditions.

Next, the boundary conditions must be prescribed at every node on the boundary. As a result of symmetry nodes 1, 2 and 3 have zero displacement in the x-direction and nodes 1, 7, 8, 9, 13 and 14 have zero displacement in the y-direction. Nodes 2 and 3 have zero stress traction in the y-direction and nodes

7, 8, 13 and 14 have zero stress traction in the x-direction. Nodes 4, 5, 6, 11 and 12 are unaffect by loading and are therefore traction free. The loading is applied to the hole such that node 9 has a 254.6 psi stress traction in the x-direction and node 10 has a 180.0 psi stress traction in both the x and y-directions.

In some instances when a stress traction is applied to a corner, it is useful to place an extra node very near the corner and to apply the load here instead of the corner. This is because the outward normal, and therefore the direction of the stress traction, is not well defined at corners.

The material properties required for a solution are those used in Kelvin's solution. In general, Young's modulus E and Poisson's ratio ν are specified from which the shear modulus G may be calculated. If the problem is best modelled as plane stress then ν' should be specified where

$$\nu' = \frac{\nu}{1+\nu}$$

The location of points where the solution is required must then be determined. For this problem, the location of the maximum stress should lie somewhere to the right of a vertical line through the hole center. Thus, field points should be selected in this area such as the points marked by '+' in figure 7.3.

All of the information required to solve this problem is listed in figure 7.4. Alternative designs of this bracket may easily be tested by changing the locations of the nodes. For example, to test the effect of different hole sizes on the stress around the hole, nodes 9-13 may be moved outward and inward from the origin. By moving nodes 3-7 the effect of different widths may also be tested.

Node	x	y	BC-x	BC-y
1	-2.0	0.0	0.0 d	0.0 d
2	-2.0	0.5	0.0 d	0.0 t
3	-2.0	1.0	0.0 d	0.0 t
4	-1.0	1.0	0.0 t	0.0 t
5	0.0	1.0	0.0 t	0.0 t
6	0.625	0.625	0.0 t	0.0 t
7	1.25	0.0	0.0 t	0.0 d
8	0.875	0.0	0.0 t	0.0 d
9	0.5	0.0	254.6 t	0.0 d
10	0.25	0.25	180.0 t	0.0 t
11	0.0	0.5	0.0 t	0.0 t
12	-0.25	0.25	0.0 t	0.0 t
13	-0.5	0.0	0.0 t	0.0 d
14	-1.25	0.0	0.0 t	0.0 d
E = 30 E06 psi , PR = 0.3				

Figure 7.4: Problem Information

7.4 Program Applications and Results

7.4.1 Kelvin Solution Test

The boundary element program BEAST was applied to the problem of a point load in an infinite plane to determine whether the results could match those predicted by Kelvin's solution. The results from this test would not confirm that BEAST could be used to solve any elastostatic boundary value problem, but would confirm that the entire algorithm could duplicate the results of the fundamental solution on which it was based.

Since a point load in an infinite plane is not a boundary value problem, a new problem with approximately the same solution was required. A method of converting a point load problem into a displacement boundary value problem was developed by Allan Dolovich at the University of Manitoba. Figure 7.5 illustrates a point force of 100 lbs in an infinite plane. Surrounding the load is a ring at a radius of 0.1 inches on which 4 points are marked. Kelvin's solution for displacement was used to determine

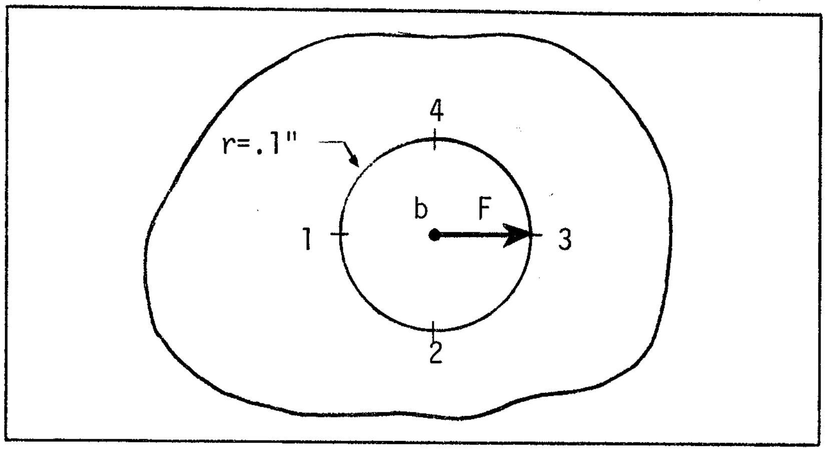


Figure 7.5: Kelvin Solution Test

the displacements of these 4 points, which were then used as boundary conditions on a small hole in an infinite plane. This boundary value problem was used as an approximation to Kelvin's problem.

The boundary element model of this problem consisted of two approximate semi-circles as illustrated in figure 7.6a. The problem was solved by the program BEAST and the stress tensor and displacement components were determined at 4 points (A, B, C, D) at a radius of 2 inches. These results are tabulated in figure 7.6b along with the results predicted by Kelvin's solution.

The maximum error in the solution found by BEAST was about 2.5 %. Most of this error was attributed to the inexact model used to approximate Kelvin's problem. Both the displacements and

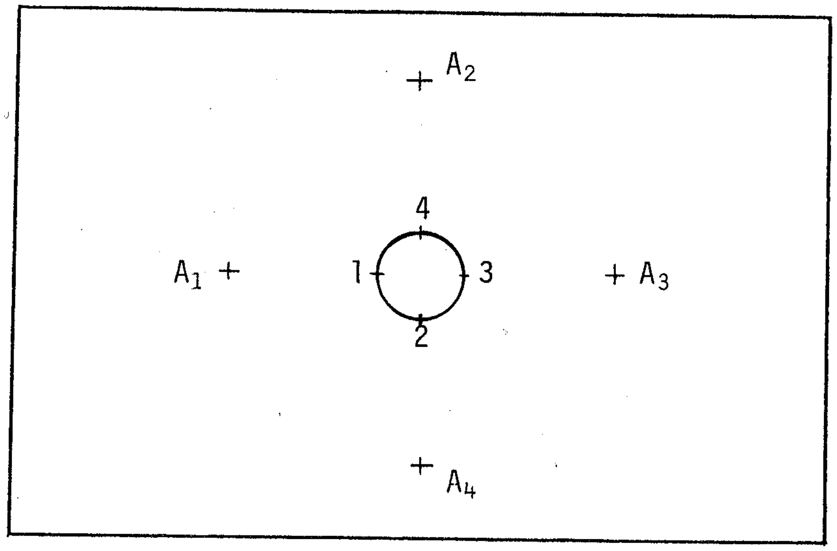


Figure 7.6a: BE Model of Kelvin Solution Test

Pt	Kelvin's Sol'n	BEAST	% Error
	$u_x \times 10^4$ inches		
A_1	-1.4077	-1.3722	2.5
A_2	-7.0919	-6.9214	2.4
A_3	-1.4077	-1.3722	2.5
A_4	-7.0919	-6.9214	2.4
	σ_{xx} (psi)		
A_1	-13.642	-13.329	2.3
A_2	0.0	0.0	-
A_3	13.642	13.329	2.3
A_4	0.0	0.0	-

Figure 7.6b: Kelvin Solution Test Results

stresses were found to be in close agreement with the fundamental solution.

The results of this test could not prove the generality of the isoparametric boundary element solution technique. However, they did prove that this method could solve at least one elastostatic problem accurately.

7.4.2 Rigid Die Displacement Problem

The successful results of the Kelvin solution test were used as a premise to begin solving theoretical elastostatic boundary value problems. Though of no practical importance, these problems were chosen because analytical solutions exist which could be used to evaluate the results of BEAST.

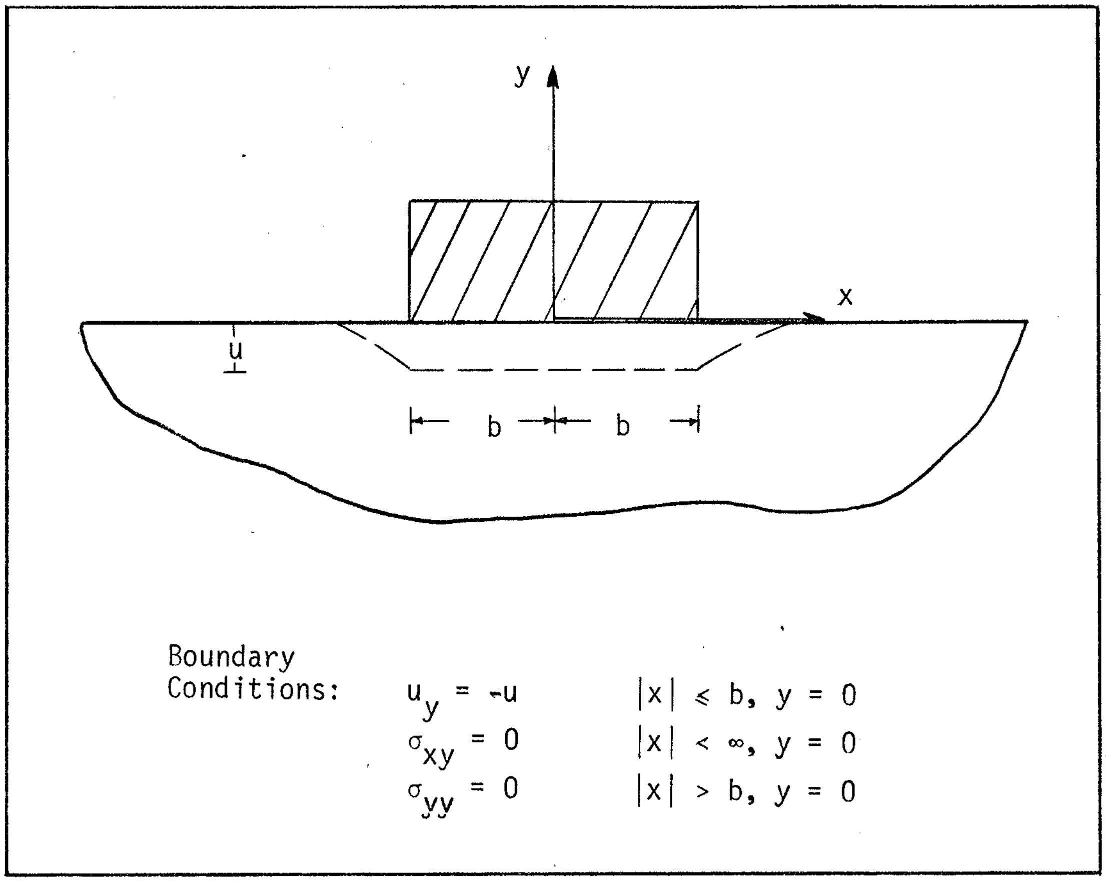
The first theoretical problem solved by BEAST was the indentation of a half plane by a lubricated rigid die. This boundary value problem and the accompanying boundary conditions are shown in figure 7.7.

The problem is characterized by a rigid block, lying on the surface of a half plane, which is given a vertical displacement of -u. The die is assumed to impart no shear stress on the half plane and the surface of the half plane, not under the die, is assumed to be traction free. The analytical solution for the y-component of displacement along the surface is [30]

$$u_y = -u \qquad |x| \leq b\ , \quad y = 0$$

$$u_y = -u \left[\frac{1 - \ln(x/b + \sqrt{x^2/b^2 - 1})}{\ln 2} \right] \qquad |x| \geq b\ , \quad y = 0 \tag{7.15}$$

where displacements beneath the die are measured relative to the displacement of the surface at a point x = 1.25b.



Boundary Conditions:

Boundary Conditions:	$u_y = -u$	$\lvert x \rvert \leqslant b,\ y = 0$
	$\sigma_{xy} = 0$	$\lvert x \rvert < \infty,\ y = 0$
	$\sigma_{yy} = 0$	$\lvert x \rvert > b,\ y = 0$

Figure 7.7: Rigid Die Problem

To approximate this problem, a boundary element model of a thin crack in an infinite plane was used, as shown in figure 7.8b. For this problem, the two stress traction boundary conditions could be eliminated since these were implied in the model. Each crack surface was modelled by 1 isoparametric element for a total of 4 nodes. These nodes were given a total displacement of -0.1 inches and the displacements were measured at points corresponding to the surface of the half plane.

Figure 7.8a shows a graph of the displacements of the half plane surface as a fraction of the applied displacement for both the 2 element BEAST solution and the analytical solution. The results from BEAST show a close correlation between displacements which gradually diverge from the exact solution at distances away from the die. At a distance of 2 crack lengths from the origin, the error was found to be about 29 % . The model accurately predicted the location of the point of zero displacement at x = 1.25b.

This test was repeated using a 4 element model of the crack surfaces, as shown in figure 7.9b. Figure 7.9a shows that the addition of these elements produced results much closer to the exact solution at points away from the die. At a distance of 2 crack lengths from the origin the error was reduced to about 8 %.

Finally, an 8 element model of the crack was tested, as shown in figure 7.10b. The results in figure 7.10a show an excellent correlation between results. In fact, between x = 1.4b and x = 1.7b the exact solution was calculated by BEAST. At 2 crack lengths from the origin the error was reduced to about 4 %.

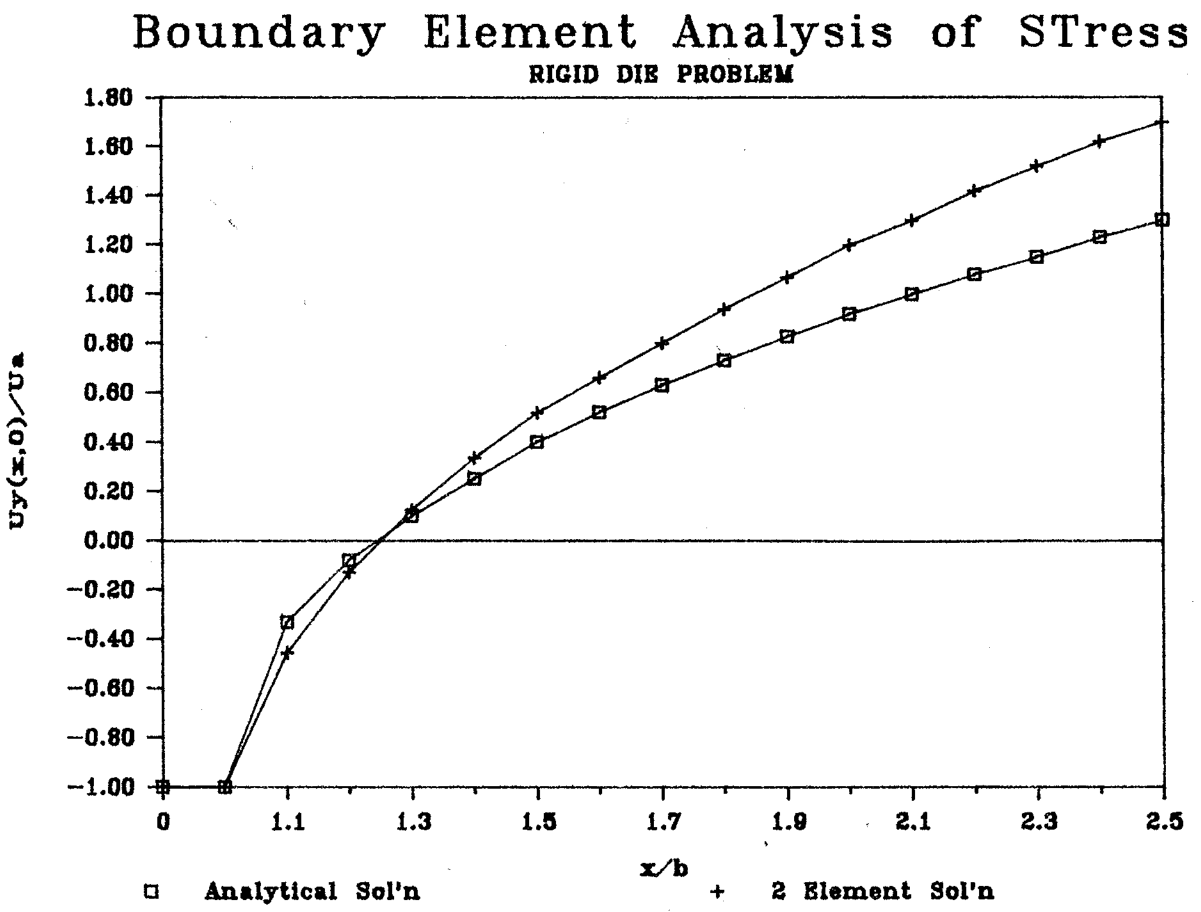


Figure 7.8a: 2 Element Die Results

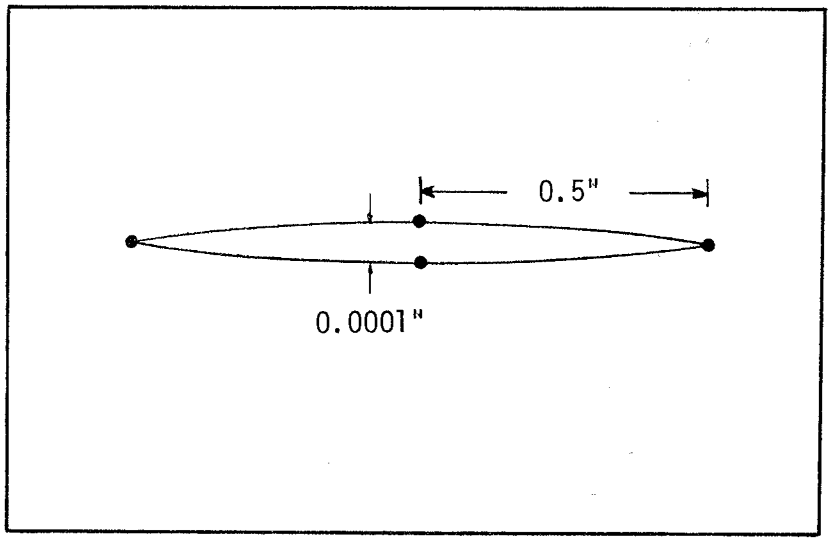


Figure 7.8b: 2 Element Die Model

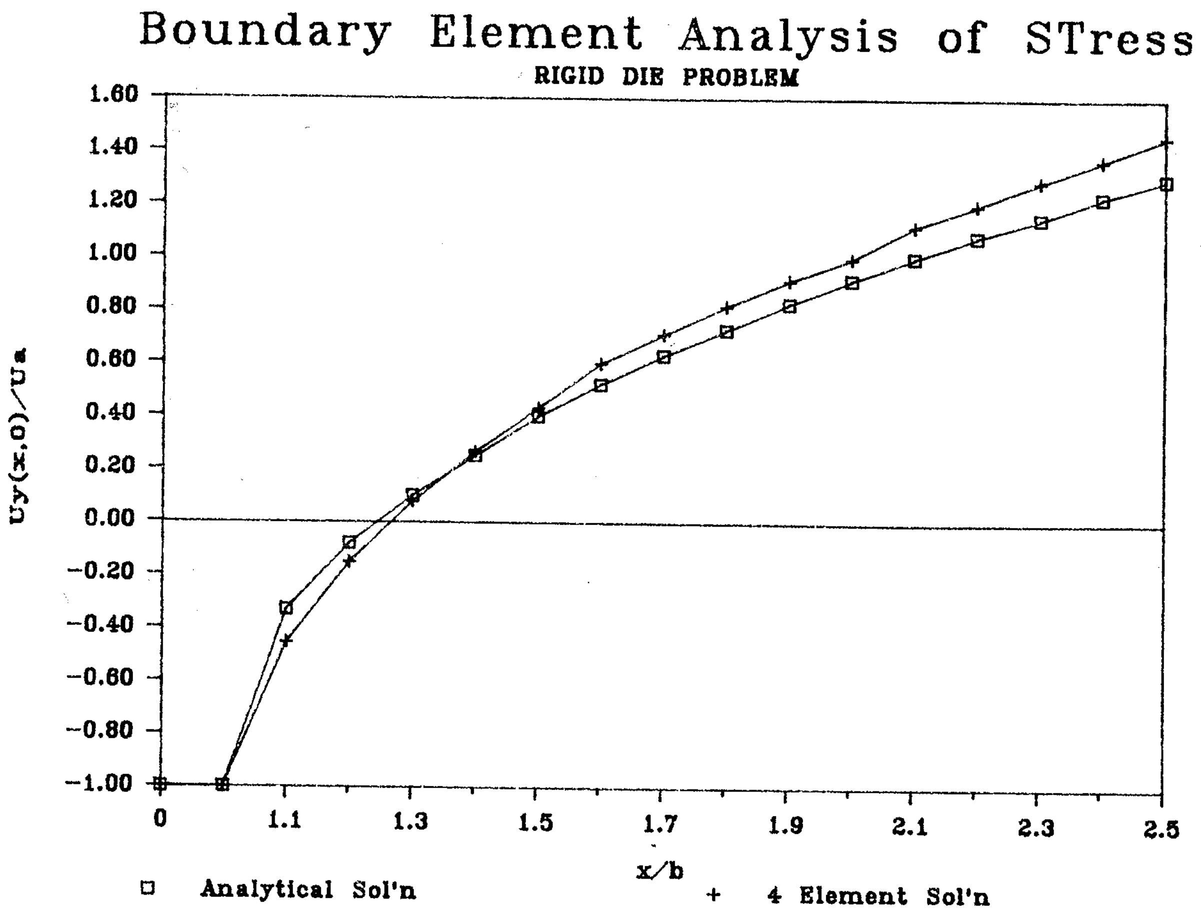


Figure 7.9a: 4 Element Die Results

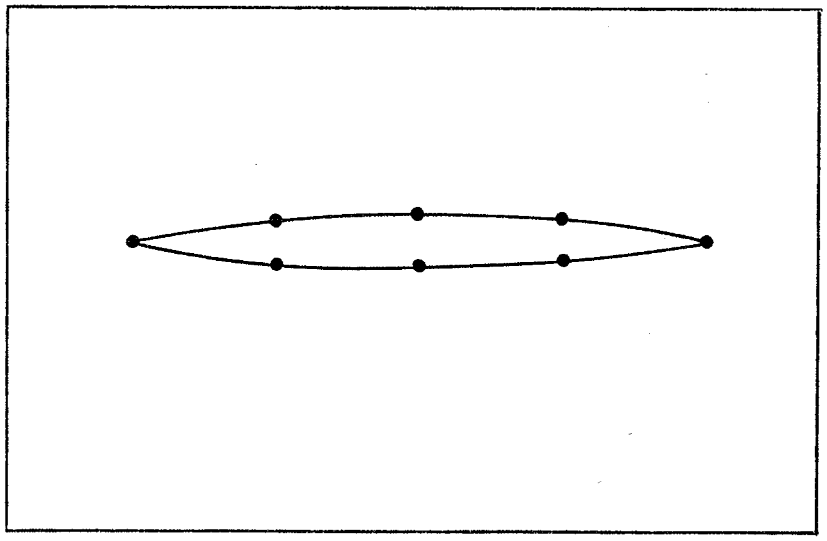


Figure 7.9b: 4 Element Die Model

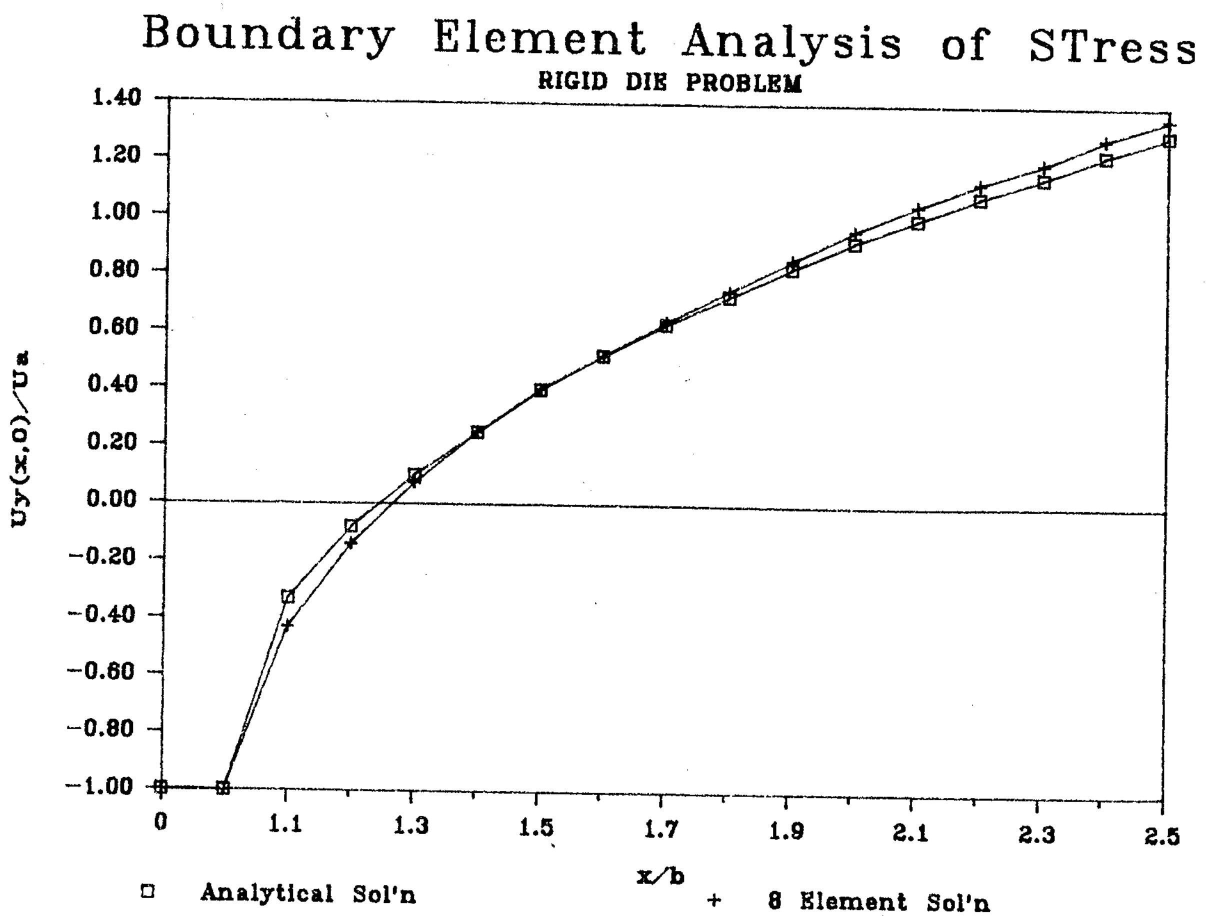


Figure 7.10a: 8 Element Die Results

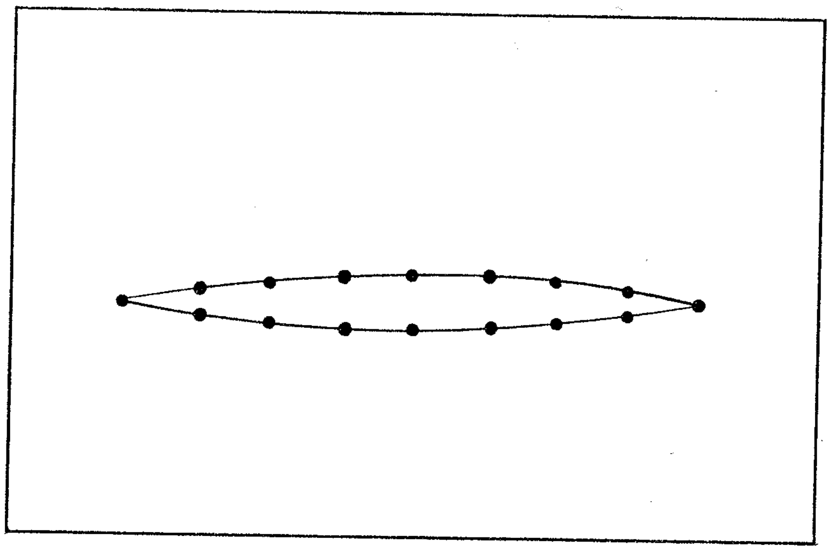


Figure 7.10b: 8 Element Die Model

Notice how the curves of figures 7.8a - 7.10a converge towards the exact solution with the addition of more elements. Ofcourse the exact solution could never be produced at all points because the models used are only approximations to the real problem.

The quality of results at points near the edge of the die did not significantly improve with the addition of elements. This effect is a by-product of the indirect boundary element technique. The formulation of the fictitious stress method required that the fictitious stress layer be distributed in a piece-wise continuous manner over the boundary. As well, the numerical implementation assumed that the fictitious stress could be applied at discrete points along each element. The assumption that this is equivalent to the effect of a continuous distribution of fictitious stress over the boundary could only be made by invoking Saint Venant's principle [34]. This principle states that for points sufficiently far from the point of application of the load, the stresses and displacements at these points are independent of the manner in which the load is distributed. Since Saint Venant's principle does not apply to points close to the application points of the fictitious stress layer, neither does the isoparametric boundary element method.

The extent of this limitation of the method is explored in the next section and the available means of overcoming this problem are discussed in chapter 9.

7.4.3 <u>Circular Hole in an Infinite Plate Under Internal Pressure</u>

The isoparametric algorithm was tested again by solving the problem of a circular hole in an infinite plate subjected to an internal pressure. The analytical solution to this problem [35] for the radial stress distribution is given by

$$\sigma_r = - \frac{R^2}{r^2} P \tag{7.16}$$

where R is the radius of the hole, P is the internal pressure and r is the radius of an arbitrary point.

The first boundary element model created to solve this problem contained 4 elements, as shown in figure 7.11b. The radius of the hole was chosen to be 3 inches and a 20 ksi internal pressure was applied to the boundary. The radial stress was measured at 0.1 inch intervals between a radius of 3.1 inches and 4.5 inches.

The computer time on an IBM PC-XT required by BEAST to obtain a solution at 15 points was about 21 seconds. The results from this test as well as the analytical solution are plotted in figure 7.11a. Close to the boundary (r = 3.1 inches), the solution from BEAST was found to be in error by 17.6 %. This error decreased steadily to 1.0 % at a radius of 4.5 inches. Thus, it was found that results within 1 inch of the boundary were affected by the manner in which the fictitious stress was distributed, and results outside this range were in excellent agreement with the analytical solution.

A second model was created which consisted of 6 elements, as shown in figure 7.12b. In addition to solving this problem using

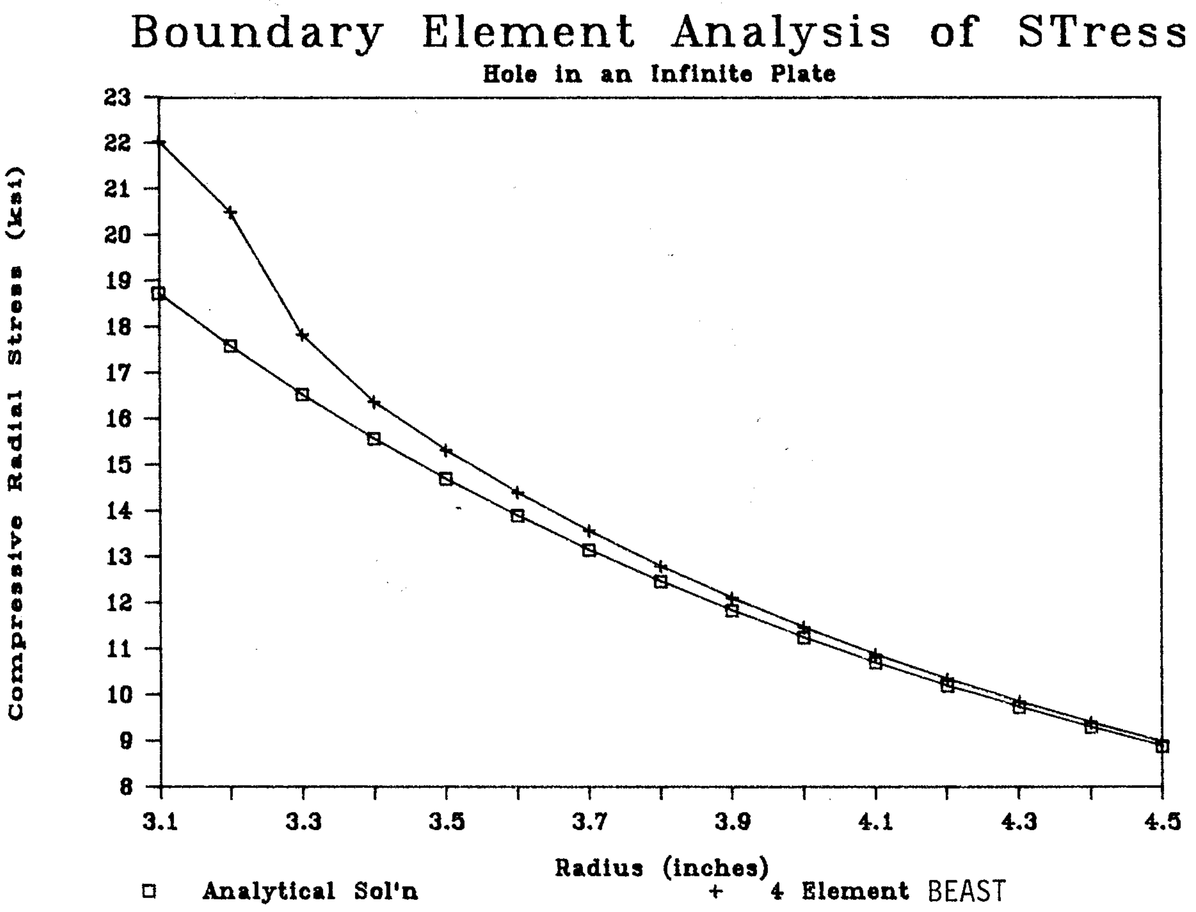


Figure 7.11a: 4 Element Hole Results

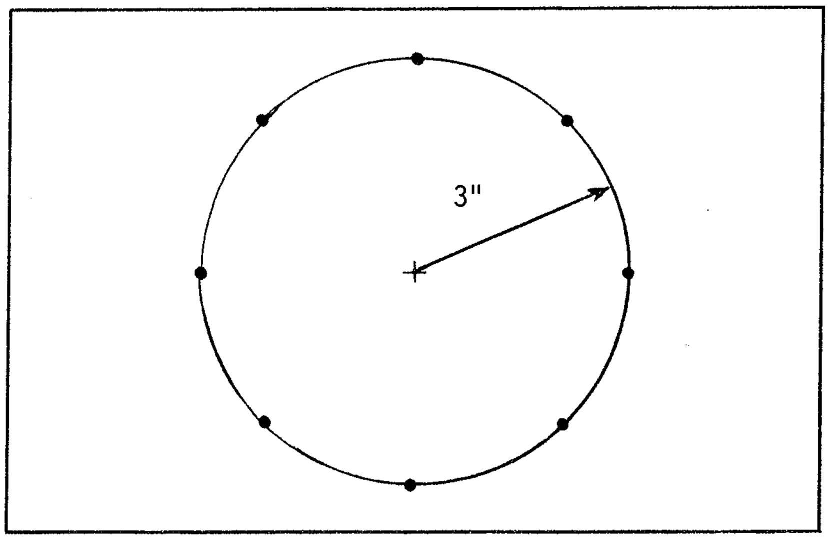


Figure 7.11b: 4 Element Hole Model

BEAST, a two-dimensional fictitious stress program was used. This program, called TWOFS, was published by Crouch and Starfield [30] in 1983 and uses straight line elements with constant fictitious stress distributions. Since this program requires only one node per element and BEAST requires two, the 6 element BEAST solution was compared to the 12 line element TWOFS solution. Therefore, the amount of input information was equal.

The results from these tests are shown in figure 7.12a. For 6 elements, BEAST required a computer time of about 43 seconds. The quality of results outside a radius of 3.2 inches increased dramatically. The error in this range decreased from 1.4 % at 3.2 inches to 0.2 % at 4 inches, and remained less than 1.0 % outside this range. At 0.1 inches from the boundary, the error decreased to 15 % with the addition of 2 elements. Thus, the range affected by the discrete distribution of fictitious stress was reduced to less than 0.2 inches.

The results from TWOFS were in error by 25 % at a 3.1 inch radius. At a 3.2 inch radius the error decreased to 16.3 %, as compared to the 1.4 % error in BEAST. Even at large distances from the boundary the error in TWOFS remained above 7.0 %, compared with less than 1.0 % error in BEAST.

Finally, a third model was tested which consisted of 8 elements, as shown in figure 7.13b. BEAST required 70 seconds for a solution at 15 points. A listing of the output file for this test is contained in the appendix on page 131 . The results in figure 7.13a show that the error at 0.1 inches from the boundary decreased to 5.8 %. Further testing showed the area affected by the discrete distribution of fictitious stress was

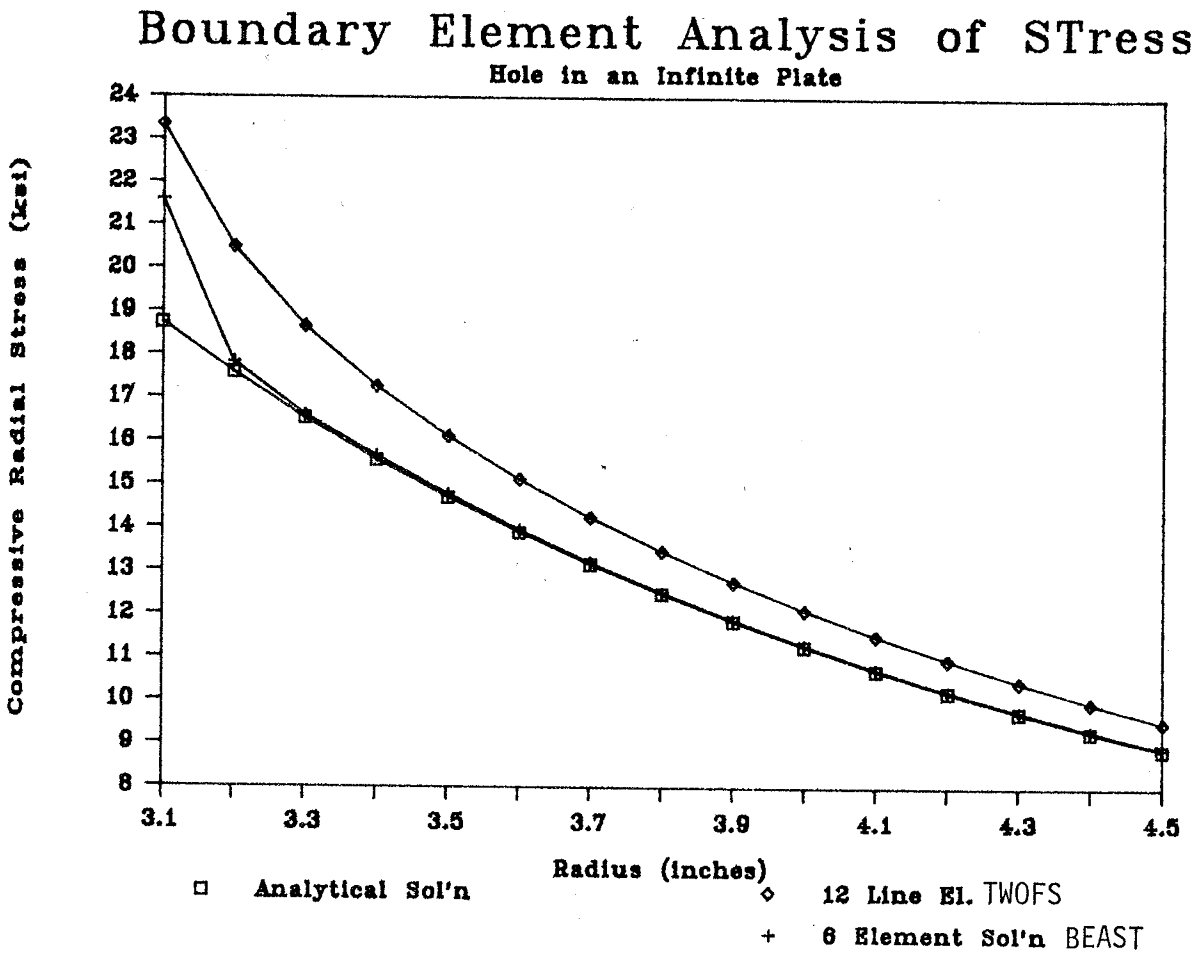


Figure 7.12a: 6 Element Hole Results

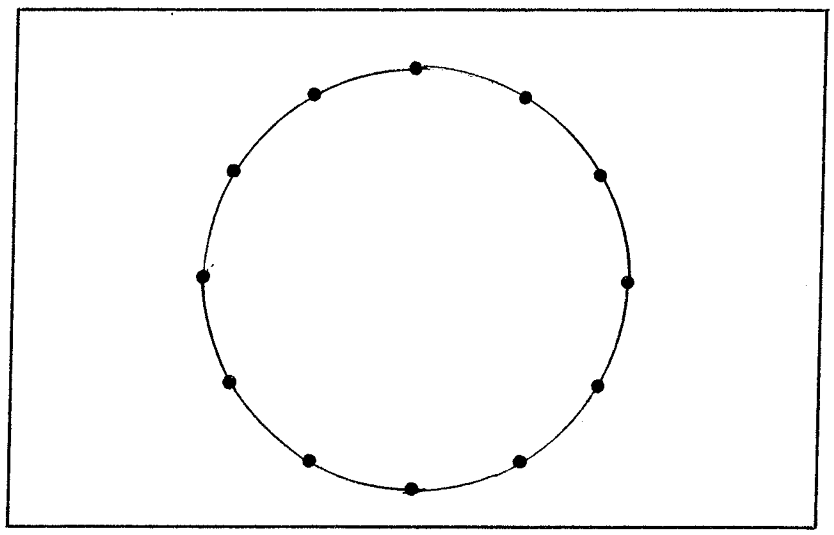


Figure 7.12b: 6 Element Hole Model

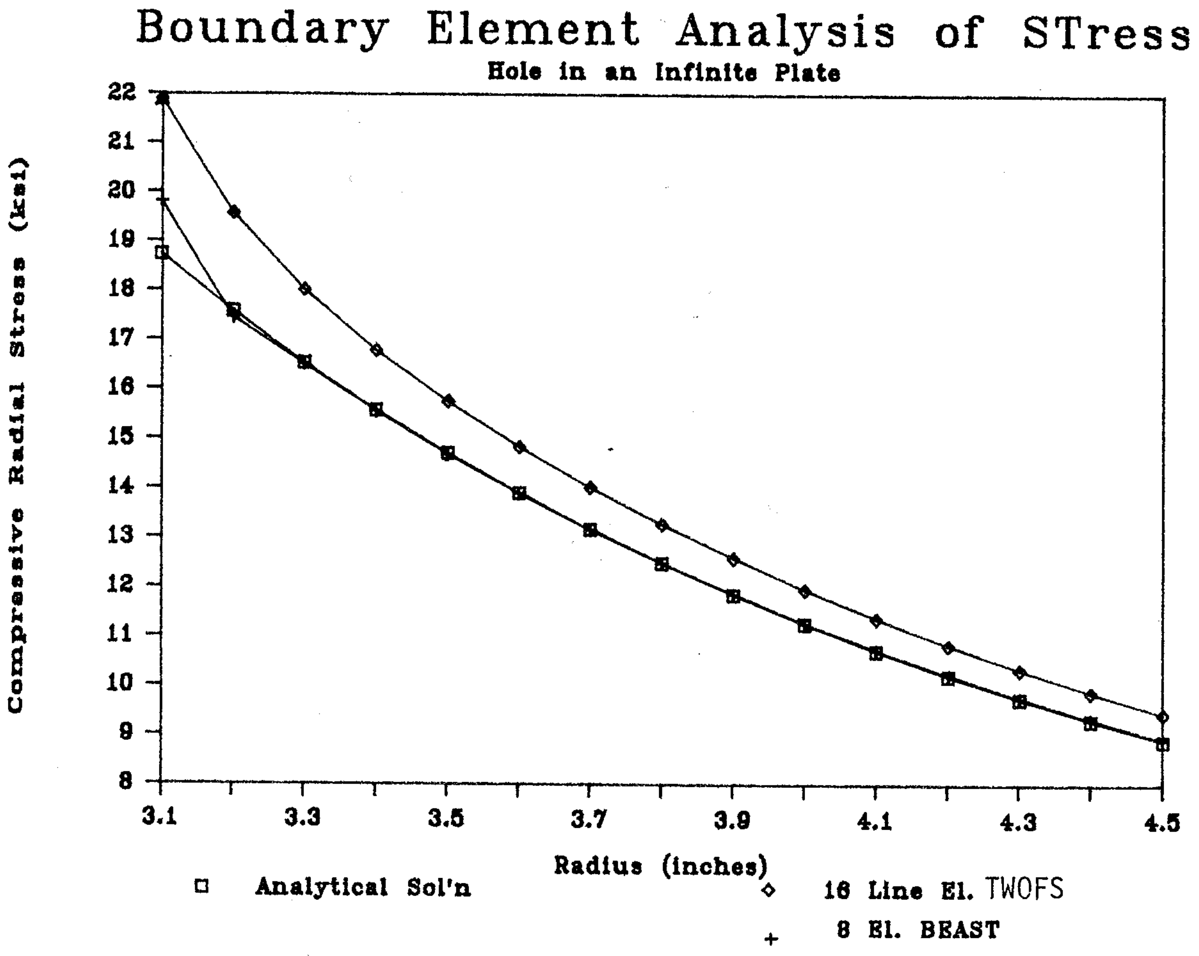


Figure 7.13a: 8 Element Hole Results

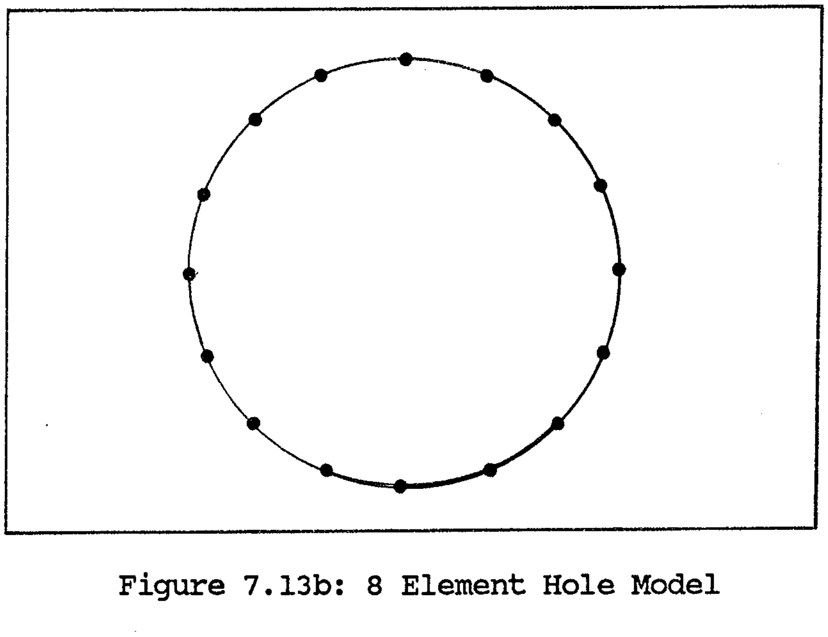


Figure 7.13b: 8 Element Hole Model

restricted to within 0.12 inches of the boundary. The error outside of a 3.3 inch radius was less than 0.2 %.

With 16 line elements, TWOFS was in error by 16.8 % at 0.1 inches from the boundary and the results did not improve beyond 6.1 % error at points away from the boundary.

The problem of a circular hole in an infinite plate confirmed the validity of the isoparametric algorithm for solving stress traction problems. Results were found to be excellent with a small number of elements, often less than 1.0 % error. The area near the boundary affected by the discrete distribution of fictitious stress was found to be small and to decrease in size rapidly with the addition of elements. If L is the length of an element and 1.0 % error is the maximum allowable, then the range affected decreased from 0.33L with 4 elements to 0.13L with 6 elements and 0.05L with 8 elements.

The isoparametric algorithm was found to yield far more accurate results than the current line element method for the same amount of input information. Increasing the number of isoparametric elements dramatically increased the quality of results but this was not found to occur with line elements. Thus, the isoparametric boundary element method has proven to be a more efficient use of the input data, and therefore to require far fewer elements to obtain a solution of a given accuracy.

7.4.4 Current Research

Thus far, the isoparametric algorithm has been applied to the solution of theoretical elasticity problems only. The solution of these problems has confirmed the validity of the theory and has illustrated the accuracy and simplicity of the method.

Although BEAST has been used successfully for solving displacement problems and for solving stress traction problems, the program has not yet successfully solved problems involving both boundary conditions. In fact, the pressurized cavity problem of section 7.4.3 has been solved as both a displacement problem and a stress traction problem, yielding excellent results in each case. However, mixing the boundary conditions has resulted in very poor quality results. This difficulty was unexpected since the solution of each coefficient matrix yielded very similar nodal fictitious stresses, yet mixing equations resulted in an entirely different solution.

No conclusions have yet been drawn as to the cause of the problem. However, it is believed that the dilemma arises in the programming of the algorithm since the theory does not indicate any abnormalities should arise from mixing equations in the coefficient matrix.

Galerkin's method has been successfully programmed into the isoparametric algorithm in a BASIC program called BEASTG. Preliminary results from this program have shown a faster convergence to the exact solution with the addition of elements than the point matching technique. However, a comprehensive evaluation of the accuracy of Galerkin's method could not be derived from the theoretical elasticity problems presented here. For a complete assessment, BEASTG must first be programmed in a high level language (such as Pascal) to compare with the floating point accuracy of BEAST.

In the numerical implementation of Galerkin's method, each Gauss point becomes a field point instead of every node, as in the point matching technique. Thus, BEASTG has been found to run much slower than the equivalent point matching method (programmed in BASIC) and also requires a larger code size. It is believed that the efficiency of BEASTG would benefit from reducing the Gaussian quadrature from 10 to 5 points, which should allow the program to run much faster with no significant loss in accuracy.

CHAPTER VIII

Conclusions

Over the past 20 years, research into indirect boundary element methods has yielded no advanced formulations suitable for industry. It has been the purpose of this thesis to elevate the formulation of an indirect method to a level comparable to similar numerical solution techniques in order to demonstrate the power of this method.

The boundary element formulation developed here was designed to solve two-dimensional elastostatic boundary value problems. A foundation for the method was provided by Kelvin's solution, a singular solution which satisfies the biharmonic equation. From this research the following conclusions were drawn:

i) It was demonstrated that a complex physical problem could be replaced by an equivalent infinite plane problem, for which Kelvin's solution was applicable. This was accomplished by distributing a fictitious stress layer over a tracing of the boundary in an infinite plane, which duplicates the effect of boundary conditions on the physical problem.

ii) The existing elementary applications of the concepts in i) were advanced using isoparametric elements, characterized by polynomial fictitious stress distributions and curved boundary elements. In addition, an alternative formulation was presented which optimizes the isoparametric solution.

iii) The isoparametric boundary element algorithm was implemented in a program called BEAST. This program was determined to be quick, expandable and flexible enough to incorporate any order of solution.

iv) The program BEAST was used to solve several theoretical elasticity problems. The results showed a close correlation between the isoparametric solution and the analytical solution for both displacement and stress traction problems. Dramatic improvements in the quality of results were found with the addition of a small number of elements. BEAST proved to be a more efficient use of input data than the line element method, and therefore to require far fewer elements to obtain a solution of a given accuracy.

v) The quality of results was found to deteriorate at points close to the boundary. This was attributed to the discrete distribution of fictitious stress that was used to approximate an actual continuous distribution. However, this effect was found to decrease rapidly with the addition of elements.

vi) Although BEAST has been used successfully to solve displacement and stress traction problems, the solution of mixed boundary condition problems has yet to be accomplished. This capability is necessary if the program is to be useful to industry. It is believed that this shortcoming can be overcome with additional research.

CHAPTER IX

Recommendations

9.1 Introduction

The emphasis in this work has been on deriving and demonstrating the validity of the isoparametric boundary element method. The development of a generalized stress analysis package based on this theory has been left to future research.

The potential of this technique to replace finite element and other boundary element methods has been demonstrated throughout this work. For this to be realized, research must be conducted on generalizing the method to solve more elasticity problems and to make the program BEAST more user-friendly.

This chapter outlines some concepts which could be implemented into the theory and the program. With these features, the isoparametric boundary element technique could become a useful industrial tool.

9.2 Extensions of the Boundary Element Formulation

9.2.1 Multi-Media Solutions

The boundary element theory developed in chapter 6 is only applicable to the solution of problems with homogeneous material properties. However, many real engineering problems consist of several materials joined by bolts, rivets or welds. The domain of such a problem is sometimes called a multiply connected region because it consists of several homogeneous media connected to form a single body.

As an example, consider the stress problem illustrated in figure 9.1. Here, a copper ring anchored by a shaft is welded to a brass rod under a vertical load P. In this case, material 1 is the brass bar, material 2 is the copper ring and material 3 is the weld itself. A boundary element solution to problems of this type can be obtained with minor changes to the existing theory.

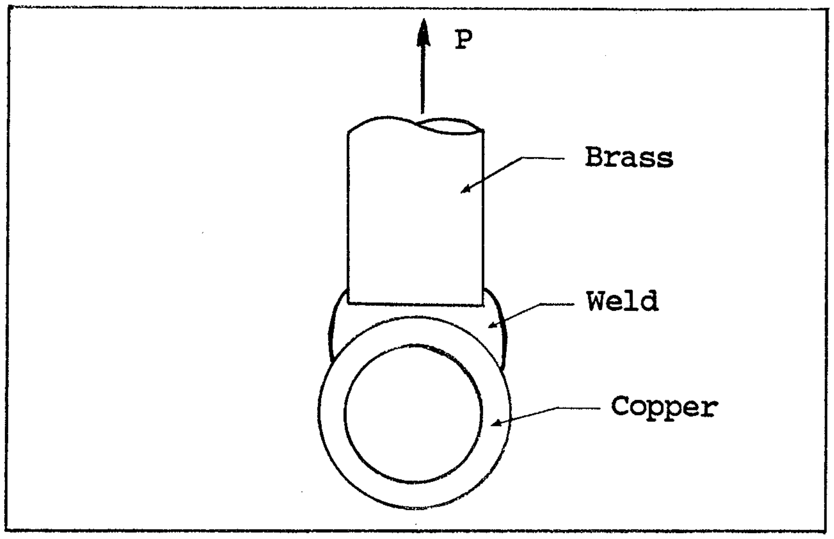


Figure 9.1: Multi-Media Problem

To solve this problem, the boundaries and the interfaces between materials must be divided into elements. The problem may then be viewed as three dependent homogeneous region problems. The boundary conditions would include the stress traction on the bar created by P, a zero displacement line within the ring, and zero stress tractions over the remainder of the free boundary. Boundary conditions can not be applied to the interfaces because these are unknown.

In place of boundary conditions, equilibrium and continuity conditions can be enforced to eliminate the interface conditions as unknowns. These conditions require at the interface between materials 1 and 2

$$\overset{1}{u}_i = \overset{2}{u}_i \tag{9.1}$$

$$\overset{1}{t}_i + \overset{2}{t}_i = 0. \tag{9.2}$$

where u_i and t_i represent the displacement and stress traction vectors respectively. However, condition (9.1) is implicit in the method (ie. a point can only have one displacement) so that only (9.2) need be enforced.

Interface condition (9.2) implies that the Kelvin solution for stress traction becomes

$$\overset{1}{t}_i + \overset{2}{t}_i = \int_b [\overset{1}{T}_{ij}\ \phi_j(x,y) - \overset{2}{T}_{ij}\ \phi_j(x,y)]\ ds + (\frac{1}{2} + \frac{1}{2})\overset{a}{\Phi}_i = 0 \tag{9.3}$$

where the negative sign arises because the outward normal of two adjacent materials are in opposite directions, or

$$\frac{\partial}{\partial n_1} = \frac{-\partial}{\partial n_2}\ . \tag{9.4}$$

Since only one distribution of ϕ can exist on an element, equation (9.3) may be written as

$$\int_b (\overset{1}{T}_{ij} - \overset{2}{T}_{ij})\ \phi_j(x,y)\ ds + \overset{a}{\Phi}_i = 0. \tag{9.5}$$

This equation may be transformed into isoparametric form for numerical integration.

The incorporation of equation (9.5) into the algorithm would allow for the solution of problems composed of any number of connected homogeneous regions.

9.2.2 Improved Accuracy Near Boundaries

Another important feature which could be added to augment the present method is to improve the solution near boundaries and corners. Section 7.4.2 discussed the decrease in quality of solutions near the boundary which was attributed to the manner in which the fictitious stress layer was distributed.

An excellent method of circumventing this problem, inherent in all indirect methods, is currently under investigation by the Electrical Engineering Dept. at the University of Manitoba. When a field point is sufficiently close to the boundary, elements in the vicinity of the point are divided into sub-elements as shown in figure 9.2. This occurrence is determined when the distance from the field point to the source node is less than a minimum value, say 10 % of an element length.

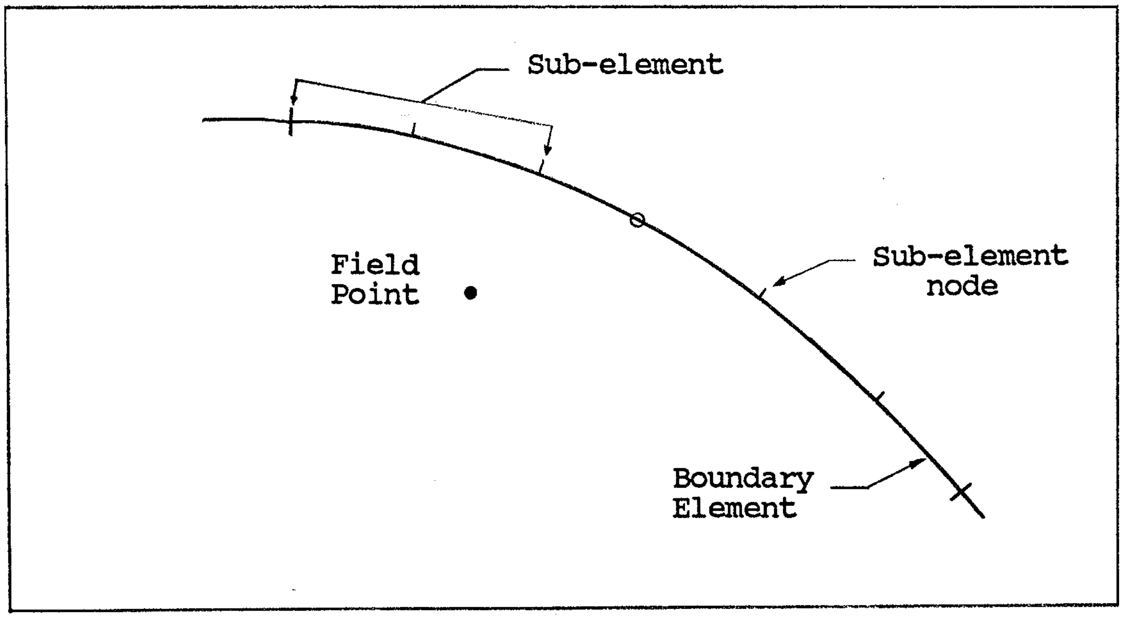


Figure 9.2: Sub-elements

The fictitious stress at the sub-element nodes may be found by interpolating between the actual nodes using the shape function equations. The Kelvin solutions can then be integrated over each individual sub-element and summed to form the integral over the entire element. In this way, the effective distance between the field point and the source nodes has been increased in relation to the element size. This is an automated method of clustering a large number of small elements in the vicinity of a point near the boundary at which a solution is desired.

An improvement in results near the boundary requires no fundamental changes to the boundary element theory. Therefore, it is recommended that such a special use subroutine be added to the program.

9.2.3 Three-Dimensional Solutions

A natural extension of the isoparametric boundary element method is to solve three-dimensional problems. An equivalent three-dimensional formulation is entirely analogous to that developed in chapters 5 and 6 with some alterations to the theory.

When solving a three-dimensional problem using boundary elements, the surface of the body must be divided into curved triangular or square elements as illustrated in figure 9.3. Three-dimensional isoparametric elements can be defined in terms of two-dimensional shape function equations [36] similar to those used in two-dimensional finite element solutions. A unit triangular element, consisting of 6 nodes, is mapped into curved space as shown in figure 9.4. In this case, two orthogonal

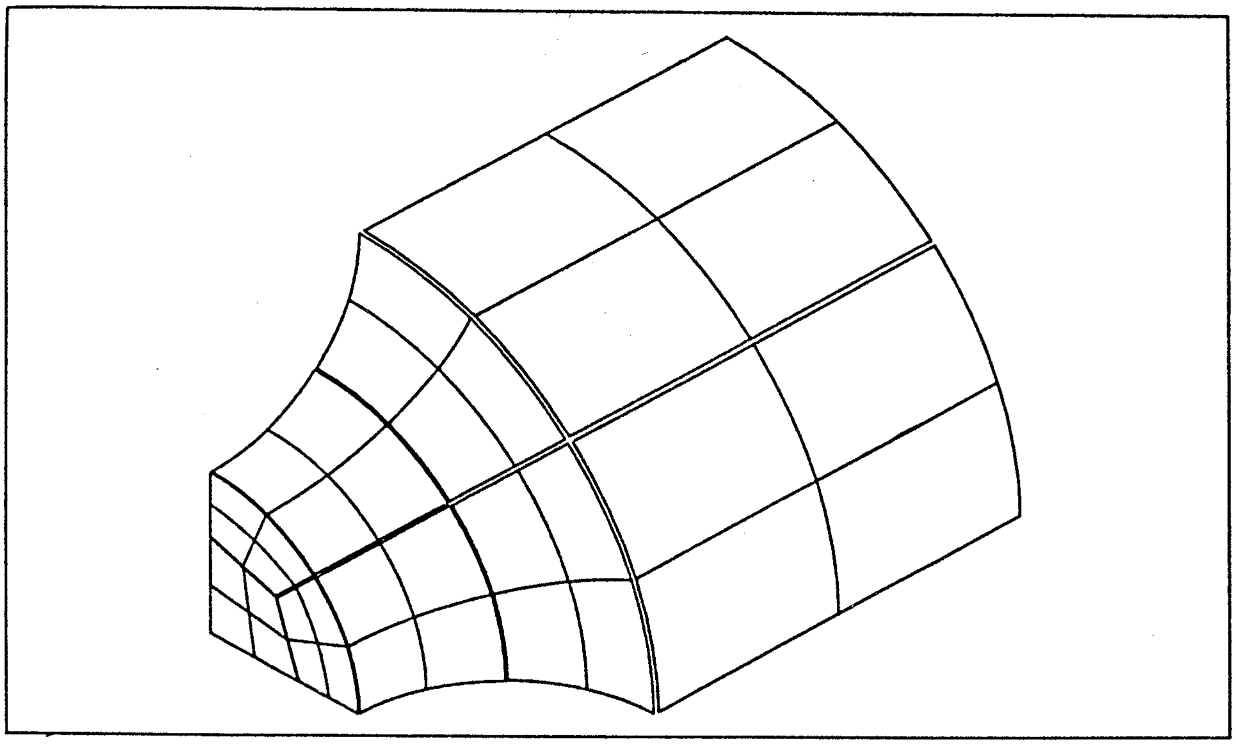


Figure 9.3: 3-D Problem

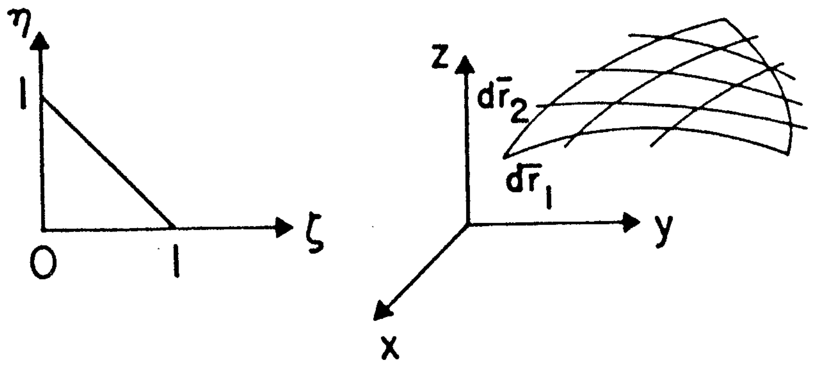


Figure 9.4: 3-D Isoparametric Elements

normalized co-ordinates are used to form elements in the x-y-z co-ordinate system.

A three-dimensional form of Kelvin's solution [24], similar to Kelvin's solution for plane strain, may be used as the fundamental solution. This solution yields stress and displacement components caused by a point load in an infinite solid. A two-dimensional isoparametric integration scheme is required to assemble the coefficient matrix and to determine field point solutions.

A three-dimensional isoparametric boundary element formulation most vividly illustrates the advantages of this method over three-dimensional finite element solutions. All calculations are performed using two-dimensional elements and only the surface need be discretized. Therefore, the formulation of problems is greatly simplified and the number of unknowns is very much reduced.

The development of a three-dimensional formulation would require much research and testing and would not be suitable for programming on a micro-computer. However, the result would be an extremely powerful stress analysis tool which could eventually rival three-dimensional finite element programs.

9.3 A Tool For Industry

The isoparametric boundary element program BEAST is capable of solving many stress and displacement problems quickly and accurately. The program is also much easier to use than any available finite element program. With the addition of a multi-zoned capability and a routine for finding accurate solutions close to the boundary, BEAST could become a generally applicable stress analysis package.

The unique solution procedures within this method allow for the addition of some convenient input/output features. The current version of the program is capable of drawing the boundary element model and graphing the field point results on the screen but more sophisticated features may also be added.

Rather than specify the co-ordinates of each node, the boundary could be traced using a digitizing tablet and the program would divide the boundary into a specified number of elements. Sections of the boundary could be marked, using the tablet, for a given boundary condition, which the program would divide into components and assign to individual nodes. The program would then solve for the distribution of fictitious stress and pause to display several output options.

Output options would include the plotting of stress contours on the screen or plotter and the colouring of zones of stress within a given range. The digitizing tablet could be used to select points at which the stress components are desired. This could be done by moving cross-hairs on the screen to the point and entering. The program could also allow for the calculation of stress and displacement components along straight or curved lines which may be entered by selecting points along the line. The program would calculate the solution components at several points along the line and graph the results as a smooth curve on the screen or plotter. These solutions would be obtained very quickly because the problem would have already been solved by determining the fictitious stress distribution.

With the addition of these input/output features, BEAST could develop into a useful stress analysis tool for industry. The ease of formulating problems and the potential output features are unique to this type of solution. BEAST is a small, efficient program which can run quickly on a personal computer. With the advent of the personal computer in almost every engineering office, the isoparametric boundary element solution has the potential to become an attractive software package.

REFERENCES

[01] Fredholm, I., Solution d'un probleme de la theorie de l'elasticite, Ark. Mat. Astr. Fys., Vol. 2, 1906, pp. 3.

[02] Muskhelishvili, N. I., Some Basic Problems of the Mathematical Theory of Elasticity, Nordhoff, Groningen, 1953.

[03] Mikhlin, S. G., Integral Equations, Pergamon Press, London, 1959.

[04] Rizzo, F. J., An integral equation approach to boundary value problems of classical elastostatics, Q. Appl. Math Vol.25, 1967, pp. 83-95.

[05] Cruse, T. A, A direct formulation and numerical solution of the general transient elastodynamic problem, J. Math. Anal. Appl., Vol. 22, 1968, pp. 341-355.

[06] Rizzo, F. J. and Shippy, D. J., A formulation and solution procedure for the general non-homogeneous elastic inclusion problem, Int. J. Solids Str., Vol. 4, 1968, p. 1161.

[07] Cruse, T. A., Numerical solutions in three-dimensional elastostatics, Int. J. Solids Str., Vol. 5, pp. 1259-1274.

[08] Cruse, T. A. and Swedlow, J. L., Formulation of boundary integral equations for three-dimensional elastoplastic flow, Int. J. Solids Str., Vol. 7, 1971, pp. 1673-1683.

[09] Riccardella, P. C., An implementation of the boundary integral technique for planar problems in elasticity and elastoplasticity, Report No. SM-73-10, Dept. Mech. Engng., Carnegie Mellon University, Pittsburg, 1973.

[10] Mendelson, A., Boundary integral methods in elasticity and plasticity, Report No. NASA TN D-7418, NASA, 1973.

[11] Cruse, T. A., Application of the boundary integral equation method to three-dimensional stress analysis, Comp. Str., Vol. 3, 1973, p. 509.

[12] Cruse, T. A., An improved boundary integral equation method for three-dimensional elastic stress analysis, Comp. Str., Vol. 4, 1974, p. 741.

[13] Lachat, J. C. and Watson, J. O., Progress in the use of boundary integral equations illustrated by examples, Comp. Meth. Appl. Mech. Engng., Vol. 10, 1977, pp. 273-289.

[14] Brebbia, C. A., In "Finite Elements in Geomechanics" , G. Gudehus (ed.), John Wiley, New York, 1977.

[15] Brebbia, C. A. (ed.) New Developments in Boundary Element Methods, CML Publications, Southampton, 1980.

[16] Brebbia, C. A. and Butterfield, R., Formal equivalence of direct and indirect boundary element methods, Appl. Math. Mod., Vol. 2, 1978, pp. 132-134.

[17] Massonet, C. E., Numerical use of integral procedures, In "Stress Analysis - Recent Developments in Numerical and Experimental Methods", O. C. Zienkiewicz and G. S. Holister (ed.), John Wiley, New York, 1965, pp. 198-235.

[18] Oliveira, E. R., Plane stress analysis by a general integral method, J. ASCE Engng. Mech. Div., Vol. 97, 1968, p. 79.

[19] Kupradze, Y. D., Dynamical problems in elasticity, In "Progress in Solid Mechanics", Vol. 3, I. N. Sneddon and R. Hill (ed.), North Holland, Amsterdam, 1964.

[20] Watson, J., The analysis of three-dimensional problems of elasticity by integral representation of displacement, In "Variational Methods in Engineering", Vol. II, International Conference, Southampton, 1972.

[21] Banerjee, R. and Butterfield, P. K., The elastic analysis of compressible piles and pile groups, Geotechnique, Vol. 21, No. 1, 1971, p. 43.

[22] Butterfield, P. K. and Tomlin, G. R., Integral techniques for solving zoned anisotropic continuum problems, In "Variational Methods in Engineering", Vol. II, International Conference, Southampton, 1972.

[23] Benjumea, R. and Sikarskie, D. L., On the solution of plane orthotropic elasticity problems by an integral method, J. Appl. Mech., Vol. 9, 1972.

[24] Banerjee, P. K., Integral equation methods for analysis of piece-wise non-homogeneous three-dimensional elastic solids of arbitrary shape, Int. J. Mech. Sc., Vol. 18, 1976, pp. 294-303.

[25] Crouch, S. L., Solution of plane elasticity problems by the displacement discontinuity method, Int. J. Num. Meth. Engng., Vol. 10, 1976, pp. 301-343.

[26] Altiero, N. J. and Gavazza, S. D., An effective boundary integral approach for the mixed boundary value problems of linear elastostatics, Appl. Math. Mod., Vol. 3, 1979.

[27] Mahajerin, E., A boundary element method for elastostatics in a plane region bounded by a smooth curve, Comp. Str., Vol. 18, No. 3, 1984, pp. 441-445.

[28] Hanna, J. R., Fourier Series and Integral of Boundary Value Problems, John Wiley, New York, 1982.

[29] Timoshenko, S. P. and Goodier, J. N., Theory of Elasticity, McGraw Hill, New York, 1970.

[30] Crouch, S. L. and Starfield, A. M., Boundary Element Methods in Solid Mechanics, George Allan and Unwin, London, 1983.

[31] Jaswon, M. A. and Symm, G. T., Integral Equation Methods in Potential Theory and Elastostatics, Academic Press, New York, 1977.

[32] Marchuk, G. I., Methods of Numerical Analysis, Springer-Verlag, New York, 1975.

[33] James, M. L., Smith, G. M. and Wolford, J. C., Applied Numerical Methods for Digital Computation with FORTRAN and CSMP, Harper Row, New York, 1977, pp. 194-199.

[34] Saada, A. S., Elasticity: Theory and Applications, Pergamon Press, New York, 1974.

[35] Volterra, E. and Gaines, J. H., Advanced Strength of Materials, Prentice-Hall, Englewood Cliffs, N. J., 1971.

[36] Wexler, A., Finite Elements for Technologists, University of Manitoba, Dept. of Elec. Engng., Report. No. TR80-4, 1980.

APPENDIX A

A.1 Gaussian Quadrature

Gaussian quadrature is a numerical technique for approximating the definite integral

$$\int_a^b f(x)\, dx. \tag{A.1}$$

In this technique, the integral is replaced by a summation over a finite number of points N, and a series of weighting factors A_i so that

$$\int_a^b f(x)\, dx \simeq \sum_{i=1}^{N} A_i\, f(x_i)\ . \tag{A.2}$$

These weighting factors measure the contribution of each $f(x_i)$ to the integral.

If a function R(x) could be found to satisfy

$$\int_a^b f(x)\, dx = \int_a^b R(x)\, dx \tag{A.3}$$

where R(x) is of lower degree than f(x), then the integration of f(x) could be found more efficiently by instead determining the integral R(x). However, it is unnecessary to obtain R(x), but rather to determine the sampling points at which R(x) = f(x), called Gauss points. This implies that

$$\sum_{i=1}^{N} A_i\, f(x_i) = \sum_{i=1}^{N} A_i\, R(x_i) \tag{A.4}$$

where x_i are the Gauss points.

To determine the location of the Gauss points, consider the expression

$$\frac{f_{2N-1}(x)}{P_N(x)} = L_{N-1}(x) + \frac{R(x)}{P_N(x)} \tag{A.5}$$

which states that the division of a polynomial f(x) of degree 2N-1 by another ploynomial P(x) of degree N yields a polynomial L(x) of degree N-1 plus a remaider R(x), divided by P(x). Multiplying both sides of equation (A.5) by P(x) yields

$$f_{2N-1}(x) = P_N(x)\ L_{N-1}(x) + R(x) \quad . \tag{A.6}$$

Recall that the Gauss points occur when f(x) = R(x). If P(x) is specified to be the Lagrange polynomial of degree N, then at the roots x_i of P(x)

$$P_N(x_i) = 0 \tag{A.7}$$

which reduces equation (A.6) to

$$f(x_i) = R(x_i) \quad . \tag{A.8}$$

Thus, the Gauss points are located at the roots of the Lagrange polynomial. Note that any Lagrange polynomial is defined for a particular interval [a,b], which is usually chosen to be either [-1,1] or [0,1].

The weighting factors A_i of equation (A.2) may be determined by enforcing that

$$\sum_{i=1}^{N} A_i\ f(x_i) \tag{A.9}$$

will give exactly the integral of any polynomial R(x) of degree N-1. To enforce this, a system of linear equations is formed in unknowns A_i from

$$\int_0^1 R(x)\,dx = \sum_{i=1}^{N} A_i\,R(x_i) \tag{A.10}$$

by setting $R(x) = 1, x, \ldots, x^{N-1}$. Once the values of A_i are determined, these weighting factors can be used to integrate exactly any polynomial R(x), and therefore any f(x).

A.2 Stress Traction Singularity

Kelvin's solution for stress traction contains an infinite discontinuity when the source point b coincides with the field point 'a'. For example, the xx-component of the influence function is given by

$$T_{xx} = \frac{-C_1}{2R^2}\left[(1-2\nu) + \frac{2R_x^2}{R^2}\right]\left[R_x n_x + R_y n_y\right] \qquad \text{(A.11)}$$

where $C_1 = \dfrac{1}{4\pi(1-\nu)}$, $R^2 = R_x^{\,2} + R_y^{\,2}$

n_x , n_y = components of the unit outward normal to the boundary

As the source point approaches the field point $R \to 0$. However, the denominator of (A.11) approaches zero faster than the numerator so that near R=0, the influence function tends to infinity as illustrated in figure A.1.

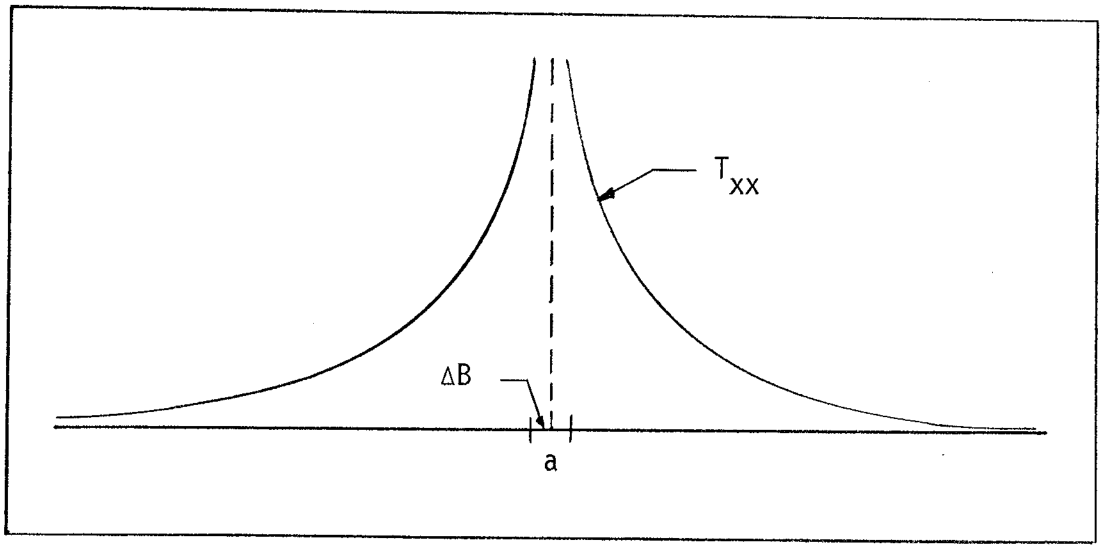


Figure A.1: Stress Traction Singularity

To integrate Kelvin's solution for stress traction around a boundary, it must also be integrated over this "singular" point, if the field point is on the boundary. To accomplish this, recall the integral is

$$t_i(a) = \int_B \phi_i(x,y)\, T_{ij}(x,y)\, ds. \tag{A.12}$$

where B represents the boundary of some region. Equation (A.12) may be separated into the two integrals

$$t_i(a) = \int_{B-\Delta B} \phi_i(x,y)\, T_{ij}(x,y)\, ds + \int_{\Delta B} \phi_i(x,y)\, T_{ij}(x,y)\, ds \tag{A.13}$$

where ΔB represents a small portion of the boundary around the singular point. For the second integral it can be shown that

$$\lim_{\Delta B \to 0} \int_{\Delta B} \phi_i(x,y)\, T_{ij}(x,y)\, ds = \frac{\phi(a)}{2} \tag{A.14}$$

where the 1/2 assumes the load is applied internally. Thus, the stress traction at a point 'a' on the boundary is given by

$$t_i(a) = \int_B \phi_i(x,y)\, T_{ij}(x,y)\, ds + \frac{\phi(a)}{2} \tag{A.15}$$

where the second integral represents the contribution of the fictitious stress at point 'a' to the stress traction at this point, and the first integral must be interpreted in the Cauchy principal value sense (ie. does not include point 'a').

Equation (A.14) is often stated in the literature without a formal proof or the source of the derivation. Though it was attempted, this proof could not be duplicated here. Therefore, it was assumed that this result was obtained through a set of simplifying assumptions which also apply to the isoparametric formulation.

APPENDIX B

B.1 BEAST in Pascal

```
(*******************************************************)
(*                                                     *)
(*                                                     *)
(*                       B E A S T                     *)
(*                                                     *)
(*                                                     *)
(*           Boundary Element Analysis of STress       *)
(*                                                     *)
(*                                                     *)
(*                 Point Matching Solution             *)
(*                                                     *)
(*                      Pascal Version                 *)
(*                                                     *)
(*                                                     *)
(*                 By    W. Neil Aitken                *)
(*                                                     *)
(*                           and                       *)
(*                                                     *)
(*                     Allan T. Dolovich               *)
(*                                                     *)
(*                                                     *)
(*******************************************************)

Program BEAST;

{$I typedef.sys}

CONST

   MaxElem=15;
   MaxNodes=30;
   NumGaussPts=10;
   MaxPhi=60;
   MaxFldPts=25;
   ExtPhi=61;

TYPE

   OneToTen      =ARRAY[1..10] OF REAL;
   ThreeByTen    =ARRAY[1..3,1..10] OF REAL;
   NumElemByTen  =ARRAY[1..MaxElem,1..10] OF REAL;
   OneToNumNodes =ARRAY[1..MaxNodes] OF REAL;
   NumElemByThree=ARRAY[1..MaxElem,1..3] OF INTEGER;
   Code          =STRING[1];
   CodeType      =ARRAY[1..MaxNodes] OF Code;
   OneToMaxPhi   =ARRAY[1..MaxPhi] OF REAL;
   MaxPhiByExtPhi=ARRAY[1..MaxPhi,1..ExtPhi] OF REAL;
   OneToMaxFldPts=ARRAY[1..MaxFldPts] OF REAL;


VAR
```

```
  FileNameIn,FileNameOut     :String[80];
  title                      :String[80];
  InFile,OutFile             :Text;
  Wt,Wt2                     :OneToTen;
  alpha,beta,alpha2,beta2    :ThreeByTen;
  xs,ys,xs2,ys2,ds,ds2       :NumElemByTen;
  x,y,Nx,Ny                  :OneToNumNodes;
  NodeNum                    :NumElemByThree;
  xCode,yCode                :CodeType;
  Nu,E,c1,c2                 :Real;
  Coeff                      :MaxPhiByExtPhi;
  phi,B                      :OneToMaxPhi;
  Xpt,Ypt,Ux,Uy,Sx,Sy,Ss     :OneToMaxFldPts;
  NumNodes,NumElem,
  NumFldPts                  :INTEGER;

{$I graphix.sys}
{$I kernel.sys}
{$I windows.sys}
{$I Polygon.hgh}
{$I Axis.hgh}
{$I spline.hgh}
{$I findwrld.hgh}
{$I Intro.bst}


Procedure blankln(n:integer); (*** Prints n blank lines in output file ***)

Var

  j     :INTEGER;

Begin

   for j:=1 to n do
     writeln(OutFile);

End;


Procedure ErrorOut; (*** Error Message for Linear Equation Solver ***)

Begin

  ClrScr; gotoxy(18,12);
  TextColor(15);
  TextBackground(4);
  writeln('--- Matrix Solution Impossible ---');
  Halt;
End;

Procedure Heading;
```

```
Begin

  TextBackground(4);
  gotoxy(22,5);
  writeln('Boundary Element Analysis of Stress');
  TextBackground(1);

End;

{$I graph.bst}
{$I plot.bst}

{---------------------------
|  Read and Echo Input File |
|                           |
 ---------------------------}

Procedure ReadInFile;


Var

    i                   :INTEGER;
    blank, choice       :CHAR;


Begin


  Intro;
  TextBackground(1);
  TextColor(15);
  ClrScr;
  Heading;
  gotoxy(15,12); write('Enter the input datafile name ---> ');
  readln(FileNameIn);
  ClrScr;
  Heading;
  gotoxy(15,12); write('Enter the output file name ---> ');
  readln(FileNameOut);
  Assign(InFile,FileNameIn);
  Reset(InFile);
  Assign(OutFile,FileNameOut);
  Rewrite(OutFile);
  writeln(OutFile,' ':15,'*************************************************
*');
  blankln(1);
  writeln(OutFile,' ':15,'                     B E A S T ');
  blankln(2);
  writeln(OutFile,' ':15,'         Boundary Element Analysis of STress');
  blankln(2);
  writeln(OutFile,' ':15,'*************************************************
*');
  blankln(3);
```

```
(**** Read Problem Parameters and Write them to a File ***)

  ClrScr;
  Heading;
  gotoxy(27,12);
  writeln('--- Reading Input File ---');
  readln(InFile,title);
  writeln(OutFile,' ':25,title);
  readln(InFile,NumNodes,NumElem);
  blankln(1);
  writeln(OutFile,' ':24,'Number of Boundary Elements = ',NumElem:2);
  blankln(1);
  writeln(OutFile,' ':26,'Number of Boundary Nodes = ',NumNodes:2);
  readln(InFile,E,Nu); blankln(1);
  writeln(OutFile,' ':27,'Youngs Modulus = ',E:11:2); blankln(1);
  writeln(OutFile,' ':29,'Poissons Ratio = ',Nu:4:2);
  blankln(2);
  writeln(OutFile,' ':31,'NODAL CO-ORDINATES');
  writeln(OutFile,' ':31,'==================');
  blankln(2);
  writeln(OutFile,' ':21,'Node No.',' ':8,'X Co-ord',' ':6,'Y Co-ord');
  blankln(1);
  for i:=1 to NumNodes do
    begin
    readln(InFile,x[i],y[i]);
    writeln(OutFile,' ':24,i:3,' ':9,x[i]:9:4,' ':5,y[i]:9:4);
    blankln(1);
    end {i};
  blankln(3);
  writeln(OutFile,' ':30,'BOUNDARY CONDITIONS');
  writeln(OutFile,' ':30,'===================');
  blankln(2);
  writeln(OutFile,' ':15,'Node No.',' ':5,'X-Boundary Value',' ':5,
           'Y-Boundary Value'); blankln(1);
  for i:=1 to NumNodes do
    begin
    readln(InFile,B[i],blank,xCode[i],B[i+NumNodes],blank,yCode[i]);
    writeln(OutFile,' ':17,i:3,' ':5,B[i]:15:8,' ',xCode[i]:1,' ':4,
             B[i+NumNodes]:15:8,' ',yCode[i]:1);
    if xCode[i] = 't' then xCode[i]:='T';
    if yCode[i] = 't' then yCode[i]:='T';
    blankln(1);
    end {i};
  readln(InFile,NumFldPts);
  blankln(2);
  writeln(OutFile,' ':28,'FIELD POINT CO-ORDINATES');
  writeln(OutFile,' ':28,'========================');
  blankln(2);
  writeln(OutFile,' ':21,'Point',' ':8,'X Co-ord',' ':6,'Y Co-ord');
  blankln(1);
  for i:=1 to NumFldPts do
    begin
    readln(InFile,Xpt[i],Ypt[i]);
    writeln(OutFile,' ':24,i:3,' ':6,Xpt[i]:9:4,' ':5,Ypt[i]:9:4);
    blankln(1);

(**** Zero the Field Solution Vectors ***)

    Ux[i]:=0.0; Uy[i]:=0.0;
    Sx[i]:=0.0; Sy[i]:=0.0; Ss[i]:=0.0;
    end {i};
   ClrScr;
   Heading;
   gotoxy(27,12); writeln('Would you like to view the');
   gotoxy(24,13); write('Boundary Element model (y/n) --> ');
   readln(choice);
   if (choice = 'y') or (choice = 'Y') then PlotMesh;
   TextColor(15);
   TextBackground(1);


End;



{-------------------------------------------------------------
|                                                             |
| Initialize Gauss Points and Weights For Gaussian Quadrature.|
| Calculate Quadratic Interpolation Values at Gauss Points    |
|                                                             |
 -------------------------------------------------------------}

Procedure Initialize;

Const

  pi=3.1415926535;

Var

  Gs,eta,eta2               :REAL;
  i,j,k,m                   :INTEGER;
  GaussPt,GaussPt2          :ARRAY[1..10] OF REAL;

Begin  {initialize}


  ClrScr;
  Heading;
  gotoxy(12,12);
  writeln('--- Initializing Shape Functions and Quadrature Data ---');


  (**** Gauss Points and Weights for
        Regular Gaussian Quadrature ****)

  GaussPt[1]:= 0.0130467358;
  GaussPt[2]:= 0.0674683167;
  GaussPt[3]:= 0.1602952159;
  GaussPt[4]:= 0.283302303;
  GaussPt[5]:= 0.4255628305;
  GaussPt[6]:= 0.5744371695;
  GaussPt[7]:= 0.7166976971;
  GaussPt[8]:= 0.8397047842;
  GaussPt[9]:= 0.9325316834;
  GaussPt[10]:= 0.9869532643;

  Wt[1]:= 0.0333356722;
  Wt[2]:= 0.0747256746;
  Wt[3]:= 0.1095431813;
  Wt[4]:= 0.1346333597;
  Wt[5]:= 0.1477621124;
  Wt[6]:= 0.1477621124;
  Wt[7]:= 0.1346333597;
  Wt[8]:= 0.1095431813;
  Wt[9]:= 0.0747256746;
  Wt[10]:=0.0333356722;


  (**** Gauss Points and Weights for
        2-Interval Gaussian Quadrature ****)

  GaussPt2[1]:= 0.023455039;
  GaussPt2[2]:= 0.115382673;
  GaussPt2[3]:= 0.25;
  GaussPt2[4]:= 0.384617328;
  GaussPt2[5]:= 0.476544961;

  Wt2[1]:= 0.059231721;
  Wt2[2]:= 0.119657168;
  Wt2[3]:= 0.142222222;
  Wt2[4]:= 0.119657168;
  Wt2[5]:= 0.059231721;

  GaussPt2[6]:= 0.523455039;
  GaussPt2[7]:= 0.615382673;
  GaussPt2[8]:= 0.75;
  GaussPt2[9]:= 0.884617328;
  GaussPt2[10]:=0.976544961;

  Wt2[6]:= 0.059231721;
  Wt2[7]:= 0.119657168;
  Wt2[8]:= 0.142222222;
  Wt2[9]:= 0.119657168;
  Wt2[10]:=0.059231721;


  (**** Shape Functions ****)

  for i:=1 to NumGaussPts do
    begin
    eta:=GaussPt[i];
    alpha[1,i]:=2.0*eta*eta-3.0*eta+1.0;
    alpha[2,i]:=4.0*(eta-eta*eta);
    alpha[3,i]:=2.0*eta*eta-eta;
    beta[1,i]:=4.0*eta-3.0;
    beta[2,i]:=4.0*(1.0-2.0*eta);
    beta[3,i]:=4.0*eta-1.0;
    eta2:=GaussPt2[i];
    alpha2[1,i]:=2.0*eta2*eta2-3.0*eta2+1.0;
    alpha2[2,i]:=4.0*(eta2-eta2*eta2);
    alpha2[3,i]:=2.0*eta2*eta2-eta2;
    beta2[1,i]:=4.0*eta2-3.0;
    beta2[2,i]:=4.0*(1.0-2.0*eta2);
    beta2[3,i]:=4.0*eta2-1.0;
    end;{i}


(**** Zero Coefficient Matrix ****);

   for i:=1 to 2*NumNodes do
     begin
     for j:=1 to 2*NumNodes+1 do
       Coeff[i,j]:=0.0;
     end;


 (**** Create a Global Node No. Matrix from Element No. ****)
 {                    and Local Node No.                   }

   k:=1;
   for i:=1 to NumElem do
      begin
      for j:=1 to 3 do
        begin
        m:=k+j-1;
        if m = NumNodes+1 then m:=1;
        NodeNum[i,j]:=m;
        end {j};
      k:=k+2;
      end {i};


(**** Material Constant Coefficients of Kelvin Solutions ***)

  Gs:=0.5*E/(1.0+Nu);
  c1:=1.0/(8.0*pi*Gs*(1.0-Nu));
  c2:=1.0/(4.0*pi*(1.0-Nu));


End; {initialize}


{----------------------------------------------------------
| Calculate the Global Co-ordinates of Gauss Points.       |
| Generate the Jacobians and Components of the Unit Normal. |
|                                                           |
 ----------------------------------------------------------------}

Procedure Jacobians;

Var

  dsx,dsy,dsx2,dsy2,dx,dy,dsn    :REAL;
  j,n,m,k,node                   :INTEGER;

Begin {jacobians}

  ClrScr;
  Heading;
  gotoxy(26,12);
  writeln('--- Generating Jacobians ---');


(**** Global Gauss Points ****)

   for j:=1 to NumElem do
    begin
    for n:=1 to NumGaussPts do
      begin
      xs[j,n]:=0.0; ys[j,n]:=0.0;
      xs2[j,n]:=0.0; ys2[j,n]:=0.0;
      for m:=1 to 3 do
        begin
        node:=NodeNum[j,m];
        xs[j,n]:=xs[j,n]+alpha[m,n]*x[node];
        ys[j,n]:=ys[j,n]+alpha[m,n]*y[node];
        xs2[j,n]:=xs2[j,n]+alpha2[m,n]*x[node];
        ys2[j,n]:=ys2[j,n]+alpha2[m,n]*y[node];
        end;{m}
      end;{n}
    end;{j}



  (**** Jacobians ****)

  for j:=1 to NumElem do
    begin
    for n:=1 to NumGaussPts do
      begin
      dsx:=0.0;  dsy:=0.0;
      dsx2:=0.0; dsy2:=0.0;
      for m:=1 to 3 do
        begin
        node:=NodeNum[j,m];
        dsx:=dsx+beta[m,n]*x[node];
        dsy:=dsy+beta[m,n]*y[node];
        dsx2:=dsx2+beta2[m,n]*x[node];
        dsy2:=dsy2+beta2[m,n]*y[node];
        end;{m}
      ds[j,n]:=sqrt(dsx*dsx+dsy*dsy);
      ds2[j,n]:=sqrt(dsx2*dsx2+dsy2*dsy2);
      end;{n}
    end;{j}


  (**** Unit Outward Normals ****)

  for j:=1 to NumNodes do
    begin
    if j=1 then k:=NumNodes else k:=j-1;
    if j=NumNodes then m:=1 else m:=j+1;
    dx:=x[m]-x[k];
    dy:=y[m]-y[k];
    dsn:=sqrt(dx*dx+dy*dy);
    Nx[j]:= -dy/dsn;
    Ny[j]:=  dx/dsn;
    end; {j}

End; {jacobians}



{----------------------------------------
| Generate Influence Coefficient Matrix  |
|                                        |
 ----------------------------------------}

Procedure CoeffMatrix;


Var

  FldNode,SrcElem,SrcNode,GaussPt,
  l,m,n                                   :INTEGER;

  sumXX,sumXY,sumYX,sumYY,Rx,Ry,Rs,
  GaussQuad,G,Gx,Gy,Gxy,Gxx,Gyy,a,b,
  xx,xy,yx,yy                             :REAL;

  self                                    :BOOLEAN;


Begin   {coeffmatrix}

  ClrScr;
  Heading;
  gotoxy(19,12);
  writeln('--- Generating Influence Coefficient Matrix ---');
  TextBackground(2);
  for FldNode:=1 to NumNodes do
    begin
    gotoxy(33,14);
    writeln('Field Node ',FldNode);
    self:=false;
    for SrcElem:=1 to NumElem do
      begin
      if SrcElem=FldNode div 2 then self:=true;
      for SrcNode:=1 to 3 do
        begin
        sumXX:=0.0; sumXY:=0.0; sumYX:=0.0; sumYY:=0.0;


    (**** Integrate Kelvin's Solution Evaluated
             at the Boundary Nodes ****)

        for GaussPt:=1 to NumGaussPts do
          begin
          if self = true then
            begin
            Rx:=x[FldNode]-xs2[SrcElem,GaussPt];
            Ry:=y[FldNode]-ys2[SrcElem,GaussPt];
            GaussQuad:=Wt2[GaussPt]*alpha2[SrcNode,GaussPt]
                       *ds2[SrcElem,GaussPt];
            end
          else

            begin
            Rx:=x[FldNode]-xs[SrcElem,GaussPt];
            Ry:=y[FldNode]-ys[SrcElem,GaussPt];
            GaussQuad:=Wt[GaussPt]*alpha[SrcNode,GaussPt]
                       *ds[SrcElem,GaussPt];
            end; {if}
          Rs:=Rx*Rx+Ry*Ry;

          G:= -Ln(Rs)/2.0;
          Gx:= -Rx/Rs;
          Gy:= -Ry/Rs;
          Gxy:=2.0*Rx*Ry/(Rs*Rs);
          Gxx:=(Rx*Rx-Ry*Ry)/(Rs*Rs);
          Gyy:= -Gxx;
          a:= Nx[FldNode]; b:=Ny[FldNode];

          if xCode[FldNode] = 'T' then begin

            {x traction solution}
            xx:=c2*((2.0*(1.0-Nu)*Gx-Rx*Gxx)*a
                  +((1.0-2.0*Nu)*Gy-Rx*Gxy)*b);
            xy:=c2*(((2.0*Nu*Gy)-Ry*Gxx)*a
                  +((1.0-2.0*Nu)*Gx-Ry*Gxy)*b);
            end

          else  begin

            {x displacement solution}
            xx:=c1*((3.0-4.0*Nu)*G-Rx*Gx);
            xy:= -c1*Ry*Gx;
            end; {if}

          if yCode[FldNode] = 'T' then begin

            {y traction solution}
            yx:=c2*(((1.0-2.0*Nu)*Gy-Rx*Gxy)*a
                  +(2.0*Nu*Gx-Rx*Gyy)*b);
            yy:=c2*(((1.0-2.0*Nu)*Gx-Ry*Gxy)*a
                  +(2.0*(1.0-Nu)*Gy-Ry*Gyy)*b);
            end
          else begin

            {y displacement solution}
            yx:= -c1*Rx*Gy;
            yy:= c1*((3.0-4.0*Nu)*G-Ry*Gy);
            end; {if}

          sumXX:=sumXX+xx*GaussQuad;
          sumXY:=sumXY+xy*GaussQuad;
          sumYX:=sumYX+yx*GaussQuad;
          sumYY:=sumYY+yy*GaussQuad;
          end; {GaussPt}

          l:=FldNode;
          m:=NodeNum[SrcElem,SrcNode];
          n:=NumNodes;


      (**** Store in Influence Coefficient Matrix ****)

          Coeff[l,m]:=Coeff[l,m]+sumXX;
          Coeff[l,m+n]:=Coeff[l,m+n]+sumXY;
          Coeff[l+n,m]:=Coeff[l+n,m]+sumYX;
          Coeff[l+n,m+n]:=Coeff[l+n,m+n]+sumYY;
        end; {SrcNode}
      end; {SrcElem}
    end; {FldNode}


(**** Add Traction Discontinuity ****)

  for n:=1 to NumNodes do
    begin
    m:=n+NumNodes;
    if xCode[n] = 'T' then Coeff[n,n]:=Coeff[n,n]+0.50;
    if yCode[n] = 'T' then Coeff[m,m]:=Coeff[m,m]+0.50;
    end;{n}

End;{coeffmatrix}


{-----------------------------------------------------------
|  Cholesky Linear Equation Solution for Fictitious Stresses |
|                                                            |
 ------------------------------------------------------------}

Procedure Cholesky;

Var

  NumVar,m,n,l,j,i,ii,jj,k,nm           :INTEGER;
  sum                                   :REAL;

Begin   {cholesky}

  TextBackground(1);
  ClrScr;
  Heading;
  gotoxy(24,12);
  writeln('--- Solving Linear Equations ---');
  NumVar:=2*NumNodes;
  m:=NumVar+1;
  n:=NumVar;

  for l:=1 to n do
     Coeff[l,m]:=B[l];

   if abs(Coeff[1,1]) < 1.0e-20 then ErrorOut;

  for j:=2 to m do Coeff[1,j]:=Coeff[1,j]/Coeff[1,1];

  for i:=2 to n do
    begin
    j:=i;
    for ii:=j to n do
      begin
      sum:=0.0;
      for k:=1 to j-1 do
        sum:=sum+Coeff[ii,k]*Coeff[k,j];
      Coeff[ii,j]:=Coeff[ii,j]-sum;
      end; {ii}
    for jj:=i+1 to m do
      begin
      sum:=0.0;
      for k:=1 to i-1 do
        sum:=sum+Coeff[i,k]*Coeff[k,jj];
      if abs(Coeff[i,i]) < 1.0e-20 then ErrorOut;
      Coeff[i,jj]:=(Coeff[i,jj]-sum)/Coeff[i,i];
      end; {jj}
    end; {i}


(**** Store Linear Equation Solution in Fictitious Stress Vector ****)

  phi[n]:=Coeff[n,n+1];
  for nm:=1 to n-1 do
```

```
    begin
    sum:=0.0;
    i:=n-nm;
    for j:=i+1 to n do
      sum:=sum+Coeff[i,j]*phi[j];
    phi[i]:=Coeff[i,m]-sum;
    end; {nm}

End; {cholesky}


{---------------------------------------------------------------
|  Multiply Nodal Fictitious Stress Values By Shape Functions.  |
|  Calculate Displacement and Stress Components at Field Points.|
|                                                               |
 ---------------------------------------------------------------}

Procedure FieldPts;


Var

  Uxx,Uyy,Sxx,Sxy,Syy,Rx,Ry,Rs,G,Gx,
  Gy,Gxy,Gxx,Gyy,xx,xy,yx,yy,xxx,xxy,
  xyx,xyy,yyx,yyy,phiX,phiY              :REAL;
  node,i,j,k,l,n,pt,elem,GaussPt         :INTEGER;
  phiAlpha                               :ARRAY[1..MaxNodes,1..10] OF REAL;
  choice                                 :CHAR;


Begin  {fieldpts}

  ClrScr;
  Heading;
  gotoxy(18,12);
  writeln('--- Calculating Displacement and Stress ---');
  gotoxy(18,13);
  writeln('       Components at Field Points');


(**** Multiply Nodal Fictitious Stresses By Shape Functions ****)

  i:=NumNodes;
  j:=NumElem;
  for k:=1 to NumElem do
    begin
    for l:=1 to NumGaussPts do
      begin
      phiAlpha[k,l]:=0.0;
      phiAlpha[k+j,l]:=0.0;
      for n:=1 to 3 do
        begin
        node:=NodeNum[k,n];
        phiAlpha[k,l]:=phiAlpha[k,l]+phi[node]*alpha[n,l];
        phiAlpha[k+j,l]:=phiAlpha[k+j,l]+phi[node+i]*alpha[n,l];
        end; {n}
      end; {l}
    end; {k}


(**** Integrate Kelvin Solutions over Boundary ****)

    for pt:=1 to NumFldPts do
      begin
      for elem:=1 to NumElem do
        begin
        Uxx:=0.0; Uyy:=0.0;
        Sxx:=0.0; Sxy:=0.0; Syy:=0.0;
        for GaussPt:=1 to NumGaussPts do
          begin

          Rx:=Xpt[pt]-xs[elem,GaussPt];
          Ry:=Ypt[pt]-ys[elem,GaussPt];
          Rs:=Rx*Rx+Ry*Ry;
          G:= -Ln(Rs)/2.0;
          Gx:= -Rx/Rs;
          Gy:= -Ry/Rs;
          Gxy:= 2.0*Rx*Ry/(Rs*Rs);
          Gxx:= (Rx*Rx-Ry*Ry)/(Rs*Rs);
          Gyy:= -Gxx;

          {Displacement Soln}
          xx:=c1*((3.0-4.0*Nu)*G-Rx*Gx);
          xy:= -c1*Ry*Gx;
          yx:= -c1*Rx*Gy;
          yy:=c1*((3.0-4.0*Nu)*G-Ry*Gy);

          {Stress Tensor Soln}
          xxx:=c2*(2.0*(1.0-Nu)*Gx-Rx*Gxx);
          xxy:=c2*(2.0*Nu*Gy-Ry*Gxx);
          xyx:=c2*((1.0-2.0*Nu)*Gy-Rx*Gxy);
          xyy:=c2*((1.0-2.0*Nu)*Gx-Ry*Gxy);
          yyx:=c2*(2.0*Nu*Gx-Rx*Gyy);
          yyy:=c2*(2.0*(1.0-Nu)*Gy-Ry*Gyy);

          phiX:=phiAlpha[elem,GaussPt]*ds[elem,GaussPt]
                *Wt[GaussPt];
          phiY:=phiAlpha[elem+NumElem,GaussPt]*ds[elem,GaussPt]
                *Wt[GaussPt];

          Uxx:=Uxx+xx*phiX+xy*phiY;
          Uyy:=Uyy+yx*phiX+yy*phiY;

          Sxx:=Sxx+xxx*phiX+xxy*phiY;
          Sxy:=Sxy+xyx*phiX+xyy*phiY;
          Syy:=Syy+yyx*phiX+yyy*phiY;

          end;{GaussPt}
```

```
   (**** Store Displacement and Stress components ****)

        Ux[pt]:=Ux[pt]+Uxx;
        Uy[pt]:=Uy[pt]+Uyy;
        Sx[pt]:=Sx[pt]+Sxx;
        Ss[pt]:=Ss[pt]+Sxy;
        Sy[pt]:=Sy[pt]+Syy;

        end; {elem}
      end; {pt}
      Clrscr;
      Heading;
      gotoxy(22,12); writeln('Would you like to plot the solution');
      gotoxy(22,13); write('on a graph? (y/n) --> ');
      readln(choice);
      if (choice = 'y') or (choice = 'Y') then graph;

End {fieldpts};


 {-----------------------------------------------------------
 | Print Fictitious Stress Components at the Nodes.          |
 | Print Displacement and Stress Components at Field Points. |
 |                                                           |
  -----------------------------------------------------------}


 Procedure PrintOutFile;


 Var

   i                       :INTEGER;

 Begin


     ClrScr;
     Heading;
     gotoxy(24,12);
     writeln('--- Printing Results to File ---');
     gotoxy(24,14);
     TextColor(14);
     writeln('       ',FileNameOut);
     blankln(3);
     writeln(OutFile,' ':30,'FICTITIOUS STRESSES');
     writeln(OutFile,' ':30,'===================');
     blankln(2);
     writeln(OutFile,' ':20,'Node No.          Phi-X            Phi-Y');
     blankln(2);
     for i:=1 to NumNodes do
       begin
```

```
        writeln(OutFile,' ':22,i:3,' ':10,phi[i]:11:4,' ':4,
                phi[i+NumNodes]:11:4);
        blankln(1)
        end {i};
      blankln(3);
      writeln(OutFile,' ':20,'DISPLACEMENT COMPONENTS AT FIELD POINTS');
      writeln(OutFile,' ':20,'======================================');
      blankln(1);
      writeln(OutFile,' ':15,'Point   X Co-ord   Y Co-ord',' ':12,'Ux',
               ' ':11,'Uy');
      blankln(1);
      for i:=1 to NumFldPts do
        begin
        writeln(OutFile,' ':17,i:3,' ':3,Xpt[i]:7:4,' ':4,Ypt[i]:7:4,
                ' ':5,Ux[i]:12:7,Uy[i]:12:7);
        blankln(1);
        end {i};
      blankln(3);
      writeln(OutFile,' ':24,'STRESS COMPONENTS AT FIELD POINTS');
      writeln(OutFile,' ':24,'=================================');
      blankln(1);
      writeln(OutFile,' ':5,'Point   X Co-ord   Y Co-ord',' ':9,
                'Sigma-XX     Sigma-YY     Sigma-XY');
      writeln(OutFile);
      for i:=1 to NumFldPts do
        begin
        writeln(OutFile,' ':7,i:3,' ':3,Xpt[i]:7:4,' ':5,Ypt[i]:7:4,
                ' ':10,Sx[i]:8:2,' ':5,Sy[i]:8:2,' ':5,Ss[i]:8:2);
        blankln(1);
        end {i};
        TextColor(15);
        TextBackground(4);
        gotoxy(1,24); writeln('Program BEAST has finished...');
        close(OutFile);
        close(InFile);

 End {printoutfile};



{-------------------
|  Control Program  |
|                   |
 -------------------}

BEGIN


   TextMode(3);
   ReadInFile;
   Initialize;
   Jacobians;
   CoeffMatrix;
   Cholesky;
   FieldPts;
   PrintOutFile;

END.
```

B.2 BEAST Output File

```
**************************************************

                    B E A S T

       Boundary Element Analysis of STress

**************************************************


             Hole in an Infinite Plate

             8 elements/ traction b/c

         Number of Boundary Elements =  8

          Number of Boundary Nodes = 16

          Youngs Modulus = 30000000.00

              Poissons Ratio = 0.30

               NODAL CO-ORDINATES
               ==================

      Node No.        X Co-ord      Y Co-ord

         1            3.0000        0.0000

         2            2.7720        1.1480

         3            2.1210        2.1210

         4            1.1480        2.7720

         5            0.0000        3.0000

         6           -1.1480        2.7720

         7           -2.1210        2.1210

         8           -2.7720        1.1480

         9           -3.0000        0.0000

        10           -2.7720       -1.1480

        11           -2.1210       -2.1210

        12           -1.1480       -2.7720
```

13	0.0000	-3.0000
14	1.1480	-2.7720
15	2.1210	-2.1210
16	2.7720	-1.1480

BOUNDARY CONDITIONS

Node No.	X-Boundary Value	Y-Boundary Value
1	20000.00000000 t	0.00000000 t
2	18477.60000000 t	7653.70000000 t
3	14142.10000000 t	14142.10000000 t
4	7653.70000000 t	18477.60000000 t
5	0.00000000 t	20000.00000000 t
6	-7653.70000000 t	18477.60000000 t
7	-14142.10000000 t	14142.10000000 t
8	-18477.60000000 t	7653.70000000 t
9	-20000.00000000 t	0.00000000 t
10	-18477.60000000 t	-7653.70000000 t
11	-14142.10000000 t	-14142.10000000 t
12	-7653.70000000 t	-18477.60000000 t
13	0.00000000 t	-20000.00000000 t
14	7653.70000000 t	-18477.60000000 t
15	14142.10000000 t	-14142.10000000 t
16	18477.60000000 t	-7653.70000000 t

FIELD POINT CO-ORDINATES

Point	X Co-ord	Y Co-ord
1	3.1000	0.0000
2	3.2000	0.0000
3	3.3000	0.0000
4	3.4000	0.0000
5	3.5000	0.0000
6	3.6000	0.0000
7	3.7000	0.0000
8	3.8000	0.0000
9	3.9000	0.0000
10	4.0000	0.0000
11	4.1000	0.0000
12	4.2000	0.0000
13	4.3000	0.0000
14	4.4000	0.0000
15	4.5000	0.0000

FICTITIOUS STRESSES

Node No.	Phi-X	Phi-Y
1	69215.8668	-0.0001
2	64454.4807	26727.5921
3	49037.5179	49035.9036
4	26717.4922	64455.3221
5	2.4554	69319.7585

6	-26712.4126	64454.4028
7	-49031.6350	49034.5834
8	-64451.0263	26715.2045
9	-69315.9701	-0.0001
10	-64451.0265	-26715.2040
11	-49031.6349	-49034.5835
12	-26712.4132	-64454.4026
13	2.4556	-69319.7585
14	26717.4915	-64455.3224
15	49037.5180	-49035.9034
16	64454.4810	-26727.5922

DISPLACEMENT COMPONENTS AT FIELD POINTS

Point	X Co-ord	Y Co-ord	Ux	Uy
1	3.1000	0.0000	0.0024960	-0.0000000
2	3.2000	0.0000	0.0024175	-0.0000000
3	3.3000	0.0000	0.0023444	-0.0000000
4	3.4000	0.0000	0.0022756	-0.0000000
5	3.5000	0.0000	0.0022107	-0.0000000
6	3.6000	0.0000	0.0021495	-0.0000000
7	3.7000	0.0000	0.0020916	-0.0000000
8	3.8000	0.0000	0.0020367	-0.0000000
9	3.9000	0.0000	0.0019846	-0.0000000
10	4.0000	0.0000	0.0019352	-0.0000000
11	4.1000	0.0000	0.0018881	-0.0000000
12	4.2000	0.0000	0.0018433	-0.0000000

13	4.3000	0.0000	0.0018005	-0.0000000
14	4.4000	0.0000	0.0017597	-0.0000000
15	4.5000	0.0000	0.0017207	-0.0000000

STRESS COMPONENTS AT FIELD POINTS

Point	X Co-ord	Y Co-ord	Sigma-XX	Sigma-YY	Sigma-XY
1	3.1000	0.0000	-19629.77	19507.75	0.00
2	3.2000	0.0000	-17274.32	17464.51	0.00
3	3.3000	0.0000	-16323.22	16472.81	0.00
4	3.4000	0.0000	-15374.43	15508.63	0.00
5	3.5000	0.0000	-14501.90	14618.76	0.00
6	3.6000	0.0000	-13707.09	13807.09	0.00
7	3.7000	0.0000	-12978.75	13063.98	0.00
8	3.8000	0.0000	-12308.23	12380.99	0.00
9	3.9000	0.0000	-11688.97	11751.33	0.00
10	4.0000	0.0000	-11115.60	11169.31	0.00
11	4.1000	0.0000	-10583.53	10630.04	0.00
12	4.2000	0.0000	-10088.78	10129.29	0.00
13	4.3000	0.0000	-9627.88	9663.39	0.00
14	4.4000	0.0000	-9197.79	9229.08	0.00
15	4.5000	0.0000	-8795.80	8823.54	0.00

B.3 BEDAP in Pascal

```
(************************************************************)
(*                                                          *)
(*                     B E D A P                            *)
(*                                                          *)
(*                                                          *)
(*          Boundary Element DAta Preparation               *)
(*                                                          *)
(*                                                          *)
(*                 By  W. Neil Aitken                       *)
(*                                                          *)
(*                        and                               *)
(*                                                          *)
(*                 Allan T. Dolovich                        *)
(*                                                          *)
(************************************************************)


Program BEDAP(output);


TYPE

   Code                          =STRING[1];
   CodeType                      =ARRAY[1..50] OF Code;


VAR

  NewFile                        :TEXT;
  FileName,title                 :STRING[80];
  choice,choice2                 :STRING[1];
  NumNodes,NumElem,i,j,k,
  NumFldPts                      :INTEGER;
  x,y,Xpt,Ypt                    :ARRAY[1..50] OF REAL;
  B                              :ARRAY[1..100] OF REAL;
  xCode,yCode                    :CodeType;
  E,Nu                           :REAL;


LABEL 1;


Procedure Header;

Begin

  ClrScr;
  gotoxy(35,8); writeln('B E D A P');
  gotoxy(23,12); writeln('Boundary Element DAta Preparation');
  Delay(3000);
  gotoxy(30,15); writeln('By W. Neil Aitken');
  gotoxy(38,17); writeln('and');
  gotoxy(32,19);writeln('Allan T. Dolovich');
  Delay(2000);
  ClrScr;

End;



Procedure ReadInFile;

Begin;

  ClrScr;
  gotoxy(15,12); write('Enter the New DataFile Name ---> ');
  readln(FileName);
  ClrScr;
  Assign(NewFile,FileName);
  Rewrite(Newfile);
  gotoxy(19,12);writeln('--- Enter the Title of the Problem ---');
  writeln;writeln;write('                              ');
  readln(title);
  writeln(NewFile,title);
  ClrScr;
  gotoxy(15,12); write('Enter the Number of Boundary Nodes ---> ');
  readln(NumNodes);
  NumElem:=NumNodes div 2;
  writeln(NewFile,NumNodes,'  ',NumElem);
  ClrScr;
  gotoxy(30,5); writeln('MATERIAL PROPERTIES');
  gotoxy(30,6); writeln('===================');
  gotoxy(26,13); write('Youngs Modulus ---> ');
  readln(E);
  gotoxy(26,15); write('Poissons Ratio ---> ');
  readln(Nu);
  writeln(NewFile,E,'    ',Nu);
  ClrScr;
  gotoxy(31,5); writeln('NODAL CO-ORDINATES');
  gotoxy(31,6); writeln('==================');
  for i:=1 to NumNodes do
    begin
    gotoxy(36,10); writeln('NODE ',i);
    gotoxy(36,11); writeln('-------');
    gotoxy(28,13); write('X Co-ordinate ---> ');
    readln( x[i] );
    gotoxy(28,15); write('Y Co-ordinate ---> ');
    readln( y[i] );
    gotoxy(1,13);ClrEol;
    gotoxy(1,15);ClrEol;
    writeln(NewFile,x[i],' ':10,y[i]);
    end {i};
  ClrScr;
  gotoxy(30,5); writeln('BOUNDARY CONDITIONS');
  gotoxy(30,6); writeln('===================');
  for i:=1 to NumNodes do
    begin
    gotoxy(36,10); writeln('NODE ',i);
    gotoxy(36,11); writeln('-------');
    gotoxy(25,13); write('X Boundary Value ---> ');
    readln( B[i] );
    gotoxy(24,15);write('T');LowVideo;write('raction or ');
                  NormVideo;write('D');LowVideo;
                  write('isplacement ');NormVideo;
                  write('---> ');
    readln( xCode[i] );
    gotoxy(25,17); write('Y Boundary Value ---> ');
    readln( B[i+NumNodes] );
    gotoxy(24,19);write('T');LowVideo;write('raction or ');
                  NormVideo;write('D');LowVideo;
                  write('isplacement ');NormVideo;
                  write('---> ');
    readln( yCode[i] );
    gotoxy(1,13);ClrEol;
    gotoxy(1,15);ClrEol;
    gotoxy(1,17);ClrEol;
    gotoxy(1,19);ClrEol;
    writeln(NewFile,B[i],' ',xCode[i],' ':10,B[i+NumNodes],' ',yCode[i]);
    end {i};
    ClrScr;
    gotoxy(25,12); writeln('Enter the Number of Field Points');
    gotoxy(25,13); writeln('at which Stress and Displacement');
    gotoxy(25,14); writeln('Components are to be Calculated');
    gotoxy(25,16); write  ('        ---> ');
    readln(NumFldPts);
    writeln(NewFile,NumFldPts);
    ClrScr;
    gotoxy(28,5); writeln('FIELD POINT CO-ORDINATES');
    gotoxy(28,6); writeln('========================');
    for i:=1 to NumFldPts do
      begin
      gotoxy(36,10); writeln('POINT ',i);
      gotoxy(36,11); writeln('--------');
      gotoxy(28,13); write('X Co-ordinate ---> ');
      readln( Xpt[i] );
      gotoxy(28,15); write('Y Co-ordinate ---> ');
      readln( Ypt[i] );
      gotoxy(1,13); ClrEol;
      gotoxy(1,15); ClrEol;
       writeln(NewFile,Xpt[i],' ':10,Ypt[i]);
      end {i};
    ClrScr;

End;



Procedure PrintOutFile;


Begin
```

```
    ClrScr;
    gotoxy(35,5); writeln('MATERIAL PROPERTIES');
    gotoxy(35,6); writeln('===================');
    gotoxy(29,13); writeln('Youngs Modulus = ',E:11:2);
    gotoxy(29,15); writeln('Poissons Ratio = ',Nu:4:2);
    gotoxy(1,25); writeln('Press any key to continue ...');
    j:=0;
    repeat
    gotoxy(31,3); writeln('NODAL CO-ORDINATES');
    gotoxy(31,4); writeln('==================');
    gotoxy(22,6); writeln('X Co-ord                   Y Co-ord');
    gotoxy(22,7); writeln('--------                   --------');
    gotoxy(1,9);
    for i:=1 to 8 do
      begin
      k:=i+j;
      writeln;
      if k <= NumNodes then
        writeln('   NODE ',k,' ':12,x[k]:8:4,' ':20,y[k]:8:4);
      end {i};
    gotoxy(1,25);writeln('Press any key to continue ...');
    repeat until Keypressed;
    ClrScr;
    j:=j+8;
  until j >= NumNodes ;
  j:=0;
  repeat
    gotoxy(30,3); writeln('BOUNDARY CONDITIONS');
    gotoxy(30,4); writeln('===================');
    gotoxy(22,6); writeln(' X B/V                         Y B/V');
    gotoxy(22,7); writeln(' -----                         -----');
    gotoxy(1,9);
    for i:=1 to 8 do
      begin
      k:=i+j;
      writeln;
      if k <= NumNodes then
        writeln('   NODE ',k,' ':9,B[k]:14:7,' ',xCode[k],
                ' ':13,B[k+NumNodes]:14:7,' ',yCode[k]);
      end {i};
    gotoxy(1,25); writeln('Press any key to continue ...');
    repeat until KeyPressed;
    ClrScr;
    j:=j+8;
  until j >= NumNodes ;
  j:=0;
  repeat
    gotoxy(28,3); writeln('FIELD POINT CO-ORDINATES');
    gotoxy(28,4); writeln('========================');
    gotoxy(22,6); writeln('X Co-ord                   Y Co-ord');
    gotoxy(22,7); writeln('--------                   --------');
    gotoxy(1,9);
    for i:=1 to 8 do
      begin
```

```
          k:=i+j;
          writeln;
          if k <= NumFldPts then
            writeln('  POINT ',k,' ':12,Xpt[k]:8:4,' ':20,Ypt[k]:8:4);
          end {i};
        gotoxy(1,25);writeln('Press any key to continue ...');
        repeat until Keypressed;
        ClrScr;
        j:=j+8;
      until j >= NumFldPts ;
      repeat until KeyPressed;

End;


BEGIN

  Header;
  1:
  ReadInFile;
  gotoxy(20,12); write('Would you like to review the datafile? ---> ');
  readln(choice);
  if (choice = 'y') or (choice = 'Y') then PrintOutFile;
  ClrScr;
  gotoxy(20,12); write('Would you like to create another file? ---> ');
  read(choice2);
  if (choice2 = 'y') or (choice2 = 'Y') then goto 1;
  ClrScr;
  gotoxy(20,10); writeln('The input file for the problem');
  gotoxy(22,13); writeln('" ',title,' "');
  gotoxy(20,16); writeln('has been created and is stored in');
  gotoxy(35,20); write(FileName);
  close(NewFile);

END.
```