APPLICATION OF THE FINITE ELEMENT METHOD TO PROBLEMS IN FRACTURE MECHANICS

by

PETER CHARLES WOOD

A

thesis submitted for

the degree of

DOCTOR OF PHILOSOPHY

in the

Department of Mechanical Engineering

at

THE UNIVERSITY OF ASTON IN BIRMINGHAM

APRIL 1979

APPLICATION OF THE FINITE ELEMENT METHOD TO PROBLEMS IN FRACTURE MECHANICS

by

P. C. WOOD

Ph.D. 1979

SUMMARY

Numerical techniques have been finding increasing use in all aspects of fracture mechanics, and often provide the only means for analyzing fracture problems. The work presented here, is concerned with the application of the finite element method to cracked structures.

The present work was directed towards the establishment of a comprehensive two-dimensional finite element, linear elastic, fracture analysis package. Significant progress has been made to this end, and features which can now be studied include multi-crack tip mixed-mode problems, involving partial crack closure. The crack tip core element was refined and special local crack tip elements were employed to reduce the element density in the neighbourhood of the core region.

The work builds upon experience gained by previous research workers and, as part of the general development, the program was modified to incorporate the eight-node isoparametric quadrilateral element. Also, a more flexible solving routine was developed, and provided a very compact method of solvi large sets of simultaneous equations, stored in a segmented form.

To complement the finite element analysis programs, an automatic mesh generation program has been developed, which enables complex problems, involving fine element detail, to be investigated with a minimum of input data. The scheme has proven to be versatile and reasonably easy to implement.

Numerous examples are given to demonstrate the accuracy and flexibility of the finite element technique.

FRACTURE FINITE ELEMENTS PLANE SOLIDS

ACKNOWLEDGEMENTS

The author wishes to express his sincere gratitude to his supervisor Mr. T. H. Richards, for his encouragement and advice throughout this work.

The author is grateful to Professor K. Foster, for providing the facilities for this work and would also like to thank Dr.P. D. Mallinson and the Computer Advisory Staff for their assistance in numerous programming obstacles.

Finally, I would like to thank Mrs. D. Scott for the typing of the manuscript.

CONTENTS

CHAPTER 1

INTRODUCTION

Fracture mechanics has gained in importance over recent years due to the requirements of high technology industries to attain greater life expectancy and reliability of their products. In striving for technical excellence, particularly in the aircraft and associated industries, where modern stress analysis techniques allow designers to develop optimum component configurations, the possibility of failure due to fracture becomes more prominent. Hence the need to determine the relationship between stress level and resistance to crack propagation, in order that a tolerable flaw size can be specified which forms part of the normal design processes. In other words, continuity between the design method for flawed or unflawed components is required.

The modes in which components fail have been categorized, and for the general three dimensional situation these are; the opening mode, sliding mode and the out of plane shearing mode. The presence of a crack in a load carrying structure intensifies the stress distribution around the crack tip, and in theory, the stress is infinite at the crack tip. In fact, a plastic zone forms around the crack tip and hence stress singularities cannot exist. However, the theory of linear fracture mechanics is still a valuable tool. The severity of this local stress field can be interpreted using Irwin's[12] concept of a stress

intensity factor. The three modes of fracture, given earlier, can be represented by three stress intensity factors namely, K_I the opening mode, K_{II} the tearing or inplane shearing mode and K_{III} the out of plane shearing mode. Knowledge of the stress intensity factors enables the onset of unstable fracture to be predicted in real structures, provided suitable fracture experiments are carried out on cracked specimens. It is of importance, therefore, to obtain accurate values of the stress intensity factors in order to provide design data which can be used in practical situations.

The analysis of fracture mechanics problems has been one of the most active branches of numerical methods in structural mechanics in the last decade. The application of the finite element method to determine crack tip stress fields has been finding increasing use in all aspects of fracture mechanics, providing a versatile and efficient engineering tool. Techniques used, based on this method, can be grouped under two broad headings: firstly methods for interpretation of the finite element results and secondly the construction of formulations which model the singularity at the crack tip. The Hilton and Hutchinson[51] crack tip element can be placed in the latter group of methods and this technique is the subject of detailed examination and development in the work presented in this thesis.

The finite element programs, on which this work is based, were established by Robertson[28], whos primary objective was to develop a general program capable of analysing any two-dimensional structure containing any number of cracks. This objective was far from being fully realised; a number of problems were unresolved and they form the basis of the work reported in this thesis.

The finite element method although providing a powerful tool for the engineer, has one major disadvantage and this is the large amount of input data required in any realistic problem. Hence a very important element in the present work was to provide an auto-mesh generation program, which was versatile and relatively easy to implement. The details of this program can be found in Chapter Five.

The fracture programs developed by Robertson were designed for special cases, for example, a program which caters for problems where there are two crack tips in a mixed-mode configuration. Under the heading of general program development it was proposed that a multi-tip fracture program should be compiled, which would allow greater flexibility in the types of problems which could be investigated. It was also suggested that the crack tip element should be refined, that is, extra terms in the series expansion for the near crack tip displacement field, should be retained. This would benefit the program in two ways: firstly the solution accuracy would be improved and secondly the crack tip element could be enlarged, and thus reduce the local element density required in the previous scheme; refer to Chapters 7 and 4. A further requirement, necessary in the study of crack propagation, was the implementation of Sih's[16] strain-energy density concept. The approach provides a method for predicting both the onset of cracking and the direction of subsequent crack growth, and relies on the determination of the relevant stress intensity factors. As these values are determined directly as output from the fracture program, the scheme can be readily accepted within the program; refer to section (7.6).

Robertson investigated the problem of partial crack closure and achieved some degree of success for mode I crack configurations only. However, the scheme is not applicable to mixed-mode cases, and this is an area of research which required further development. Section (7.4) describes the various techniques which have been employed in order to provide a physically acceptable finite element model for crack closure.

A variety of examples have been investigated in order to validate the various numerical procedures which were developed to resolve the aforementioned fracture mechanics topics. The computed results are given in Chapter 7 and the general program developments are described in Chapter 4. Where applicable, computer procedures are explained with the aid of a flowchart and a separate program listing is provided in the Appendix (9.3).

To summarize, the present work was directed towards a general fracture mechanics program capable of encompassing mode I and mixed-mode fracture problems of any complexity, with extra facilities to examine partial crack closure and crack propagation.

CHAPTER 2

THE FINITE ELEMENT METHOD

2.1 INTRODUCTION

The finite element method is essentially a process through which a continuum, with infinite degrees of freedom, can be approximated to by an assemblage of sub-regions each with a specified but now finite number of degrees of freedom. The behaviour of each sub-region or element is described by a set of assumed functions representing the stresses or displacements. The assumed functions are usually of a polynomial form and by using a sufficient number of elements, an acceptable representation of the overall real situation is obtained. The process is analogous to a piece-wise Rayleigh-Ritz method, where the integrations required to define the appropriate functional must be evaluated for each element in turn and the total contribution obtained by summation.

The advent of the digital computer in the early 1950's enabled the structural engineer, using the techniques of matrix algebra, to deal with problems previously too large to contemplate. This represented the beginning of the finite element method as a significant tool for engineering analysis. The variational approach to finite elements was not introduced until a decade later and this provided a rational basis from which the technique could be extended into non-structural fields.

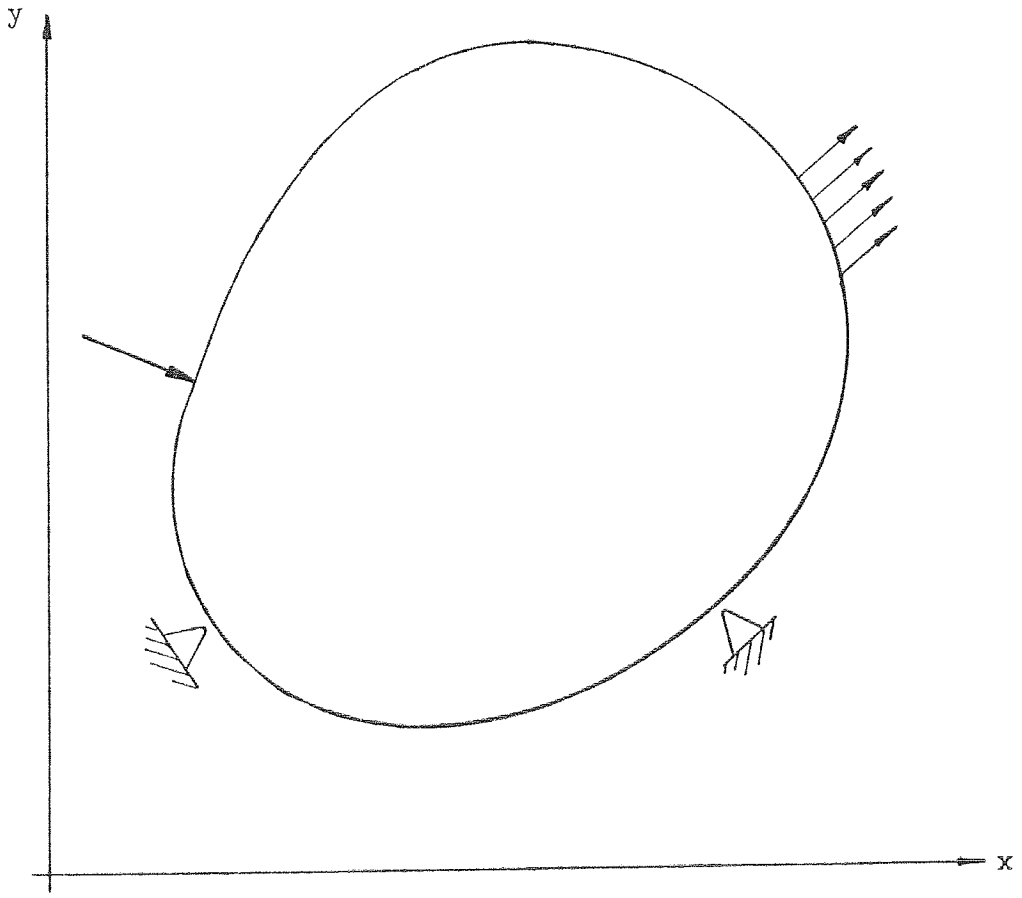
The finite element method gradually became well known during the 1960's when a vast number of papers were written. The publication of the text by Zienkiewicz and Cheung (1) consolidated the method, and similar publications are given in reference (2-4).

2.2 BRIEF GENERALIZED ACCOUNT OF THE FINITE ELEMENT METHOD

The sub-region or elements behave according to a prescribed function which describes an assumed displacement and/or stress field. More commonly the, so-called, displacement or stiffness formulation is employed. This approach has been adopted in the work described in this thesis. For convenience in subsequent reference, the process can be summarized as follows.

2.2.1 DEFINITION OF THE FINITE ELEMENT MESH

The continuum is imagined subdivided into sub-regions or elements. Each element is connected to the next, through node points on its boundary, and the nodes are numbered and referenced to a coordinate origin. The elements are defined by a series of node numbers, (element nodal connections), and from this information an element stiffness matrix relation may be determined between nodal forces and displacements. In general purpose programs an assortment of elements are available, ranging from bar-type elements to three-dimensional elements. In the case of two-dimensional problems the element usually takes the form of a triangle or quadrilateral. As a general guide, the element density is increased in areas of rapid stress changes and conversely, reduced in areas of nearly uniform stress. Figure (2.1) shows a simplified representation of a discretised two-dimensional problem.



Two Dimensional Continuum.

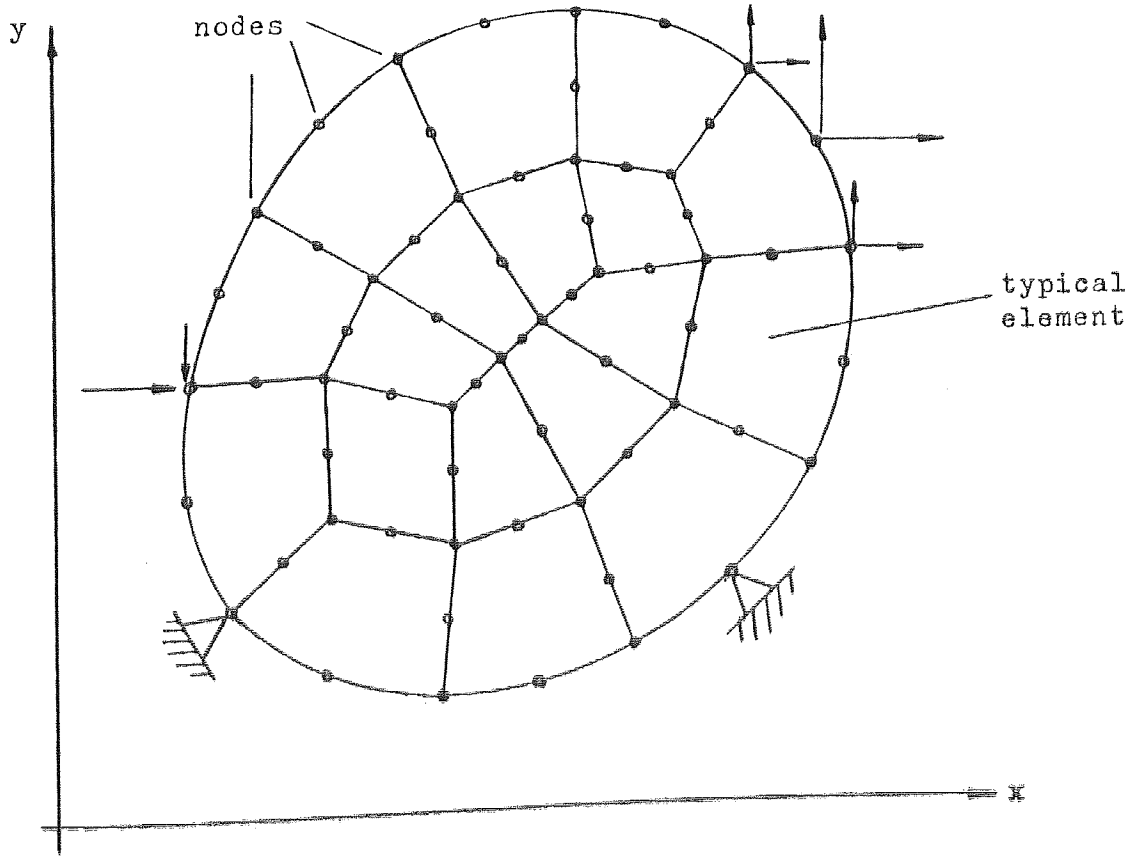


FIG 2.1 Idealization of Continuum using 8-node Isoparametric Quadrilateral elements.

2.2.2 DISPLACEMENT FUNCTION

As recounted earlier, the assumed element behaviour is governed by its displacement function, and this is chosen to define uniquely the displacement field within the element in terms of its nodal displacements. If a polynomial function is used, then it will be appreciated that the degree of the polynomial will govern the ability of the element to approximate the true displacement field. Generally, therefore, more 'linear' elements are required as compared to 'higher order elements', for a desired accuracy in any particular problem. Usually the same element formulation is used throughout the discretisation, but it is possible to mix element types in order to gain a better approximation to the real structure.

The assumed shape functions limit the infinite degrees of freedom of the system, and the true minimum of the energy may never be reached, irrespective of the fineness of subdivision. To ensure convergence to the correct result the displacement function must satisfy three criteria, namely,

i. The displacement must be continuous between adjacent elements, i.e. no opening or overlaps must be implied.

ii. The state of constant strain must be included in the function.

iii. Rigid body displacements must be represented.

2.2.3 DERIVATION OF THE STIFFNESS EQUATIONS

It is possible to define the strain distribution, and consequently the strain energy within each element, on the basis of the displacement expressions of the previous section. Thus the element strain energy is given by,

$$U_e = \int_{vol} \tfrac{1}{2}\{\varepsilon\}^t [C]\{\varepsilon\} dvol \qquad (2.1)$$

where $\{\varepsilon\}$ is the vector of strains and from Hooke's law the matrix $[C]$ is given by $\{\sigma\} = [C]\{\varepsilon\}$. Considering the total potential energy of the system, we have,

$$V = U + \Omega = \sum_{elements} (U_e + \Omega_e) \qquad (2.2)$$

Imposing the condition $\delta V = 0$ for equilibrium yields the stiffness equations,

$$[K]\{q\} = \{Q\} \qquad (2.3)$$

Here $[K]$ is a matrix of stiffness coefficients for the system, $\{Q\}$ is a vector of forces and $\{q\}$ represents the vector of unknown displacements. The element stiffness matrix is normally derived using numerical integration, although in simpler element formulations this may be found explicitly.

2.2.4 SOLUTION OF THE STIFFNESS EQUATIONS

Numerous routines are available for the solution of the stiffness equations (2.3). These are usually based on the Gaussian elimination or Cholesky decomposition processes. Efficient routines take account of the symmetric, banded nature of the stiffness matrix in order to reduce the storage requirements demanded of the computer. These techniques are further discussed in Chapter Four.

2.2.5 DETERMINING THE ELEMENT STRESSES AND STRAINS

Having computed the nodal displacements it is a simple matter to calculate the strains via the element displacement shape function used previously in the formation of the stiffness matrix; the stresses can then be obtained using Hooke's law.

Since equilibrium is not satisfied exactly but only in some average fashion, the stress field is discontinuous from element to element. As an aid to improved accuracy, the stresses at a particular node may be expressed as the average of the individual stresses at that node.

2.3 ISOPARAMETRIC QUADRILATERAL ELEMENT

With the firm establishment of the principles of finite element analysis, the possibilities of improvement of approximation were confined to devising alternative element configurations and developing new shape functions. The family of curved, isoparametric, quadrilateral elements was introduced by Ergatoudis, Irons and Zienkiewicz (1) and represented a considerable improvement over simpler 'constant strain' element formulations.

The quadrilateral element can be defined in terms of local coordinates ξ and η, as shown in figure (2.2). The relationship between the global cartesian coordinates and the local coordinates can be written in a general form as,

$$\begin{aligned} x &= N_1x_1 + N_2x_2 + N_3x_3 + \ldots\ldots = [N]^t\{x_n\} \\ y &= N_1y_1 + N_2y_2 + N_3y_3 + \ldots\ldots = [N]^t\{y_n\} \end{aligned} \qquad (2.4)$$

Where x_n and y_n lists the nodal coordinates x and y, and N_1, N_2, etc. are some function of ξ and η. For any values of ξ and η the x and y coordinates can be found once the functions N_i are known.

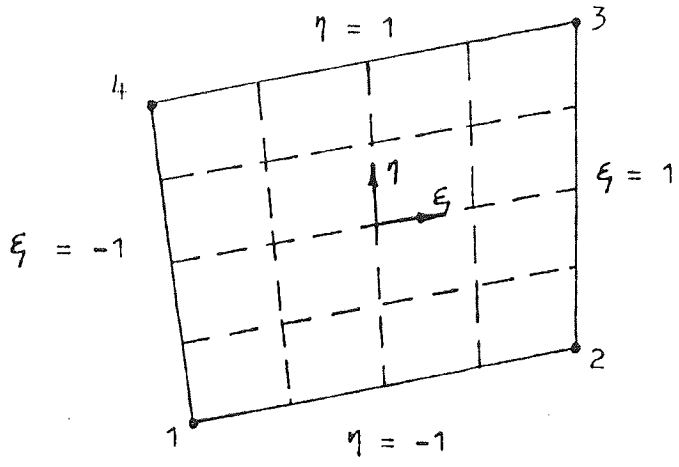


Fig a.

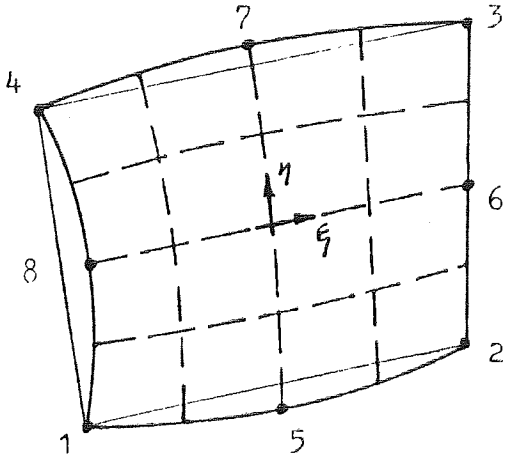


Fig b.

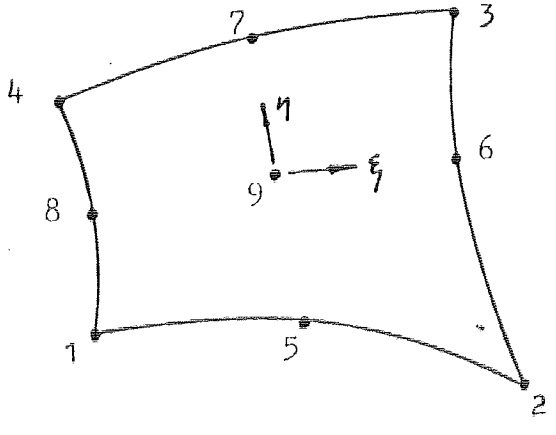


Fig c.

FIG 2.2. General quadrilateral elements.

In finite element analysis it will be necessary to define the variation of displacement components u and v in terms of the nodal values of these functions. In the, so-called, isoparametric formulation the same functions N_1, N_2, etc., previously used in equation (2.4) can be employed again. Thus we have,

$$\begin{aligned} u(\xi,\eta) &= N_1u_1 + N_2u_2 + \ldots\ldots = \{N\}^t\{u_n\} \\ v(\xi,\eta) &= N_2v_1 + N_2v_2 + \ldots\ldots = \{N\}^t\{v_n\} \end{aligned} \qquad (2.5)$$

in which N_1, etc., are termed shape functions and u_n, v_n represent the nodal values of displacement.

2.3.1 GENERATION OF POLYNOMIAL SHAPE FUNCTIONS

Suitable polynomials for the various elements which satisfy the necessary conditions of continuity can be written by introducing only terms which give the appropriate variation along the sides of the element. For the quadrilateral element, in figure (2.2.b), we can write,

$$x = \alpha_1 + \alpha_2\xi + \alpha_3\eta + \alpha_4\eta\xi + \alpha_5\xi^2 + \alpha_6\eta^2 + \alpha_7\xi^2\eta + \alpha_8\eta^2\xi \qquad (2.6)$$

and substituting the appropriate nodal values;

$$x = x_1,\ \eta = -1,\ \xi = -1,$$

$$x = x_2,\ \eta = -1,\ \xi = 1,\ \text{etc.}$$

yields eight equations of the type,

$$\{x_n\} = [C]\{\alpha_n\} \qquad (2.7)$$

from which $\{\alpha_n\} = [C]^{-1}\{x_n\}$ (2.8)

and the shape function follows as

$$[N_1,\ N_2,\ N_3 \ldots] = [1, \xi, \eta, \xi\eta, \xi^2, \eta^2, \xi^2\eta, \xi\eta^2][C]^{-1} \qquad (2.9)$$

It is possible to use a more direct approach and write by inspection.

For Corner nodes

$$N_i = \frac{1}{4}(1 + \xi_0)(1 + \eta_0)(\xi_0 + \eta_0 - 1) \tag{2.10}$$

Mid-side nodes

$$\begin{aligned} \xi_i = 0, \quad N_i &= \frac{1}{2}(1 - \xi^2)(1 + \eta_0) \\ \eta_i = 0, \quad N_i &= \frac{1}{2}(1 + \xi_0)(1 - \eta^2) \end{aligned} \tag{2.11}$$

where $\xi_0 = \xi\xi_i$

$\eta_0 = \eta\eta_i$

The quadratic element with the central node, figure (2.2.c), is a member of what is called the Lagrange family. Members of this family have equally spaced nodes forming a grid. An element of any order can be generated easily using the Lagrange polynomials, see ref.(1). The usefulness of this family is limited however, not only due to a large number of internal nodes present, but also due to the poor curve-fitting properties of the higher order polynomials. See figure (2.3).

2.4 EVALUATION OF THE ELEMENT STIFFNESS MATRIX

In the standard formulation of plane stress or strain the stiffness matrix is given by,

$$[K] = \iint [B]^t [C][B]\,dxdy \tag{2.12}$$

in which [B] is the matrix defining the strains in terms of the nodal displacements and [C] is the elasticity matrix which relates the stresses to the strains. The strain matrix is given by,

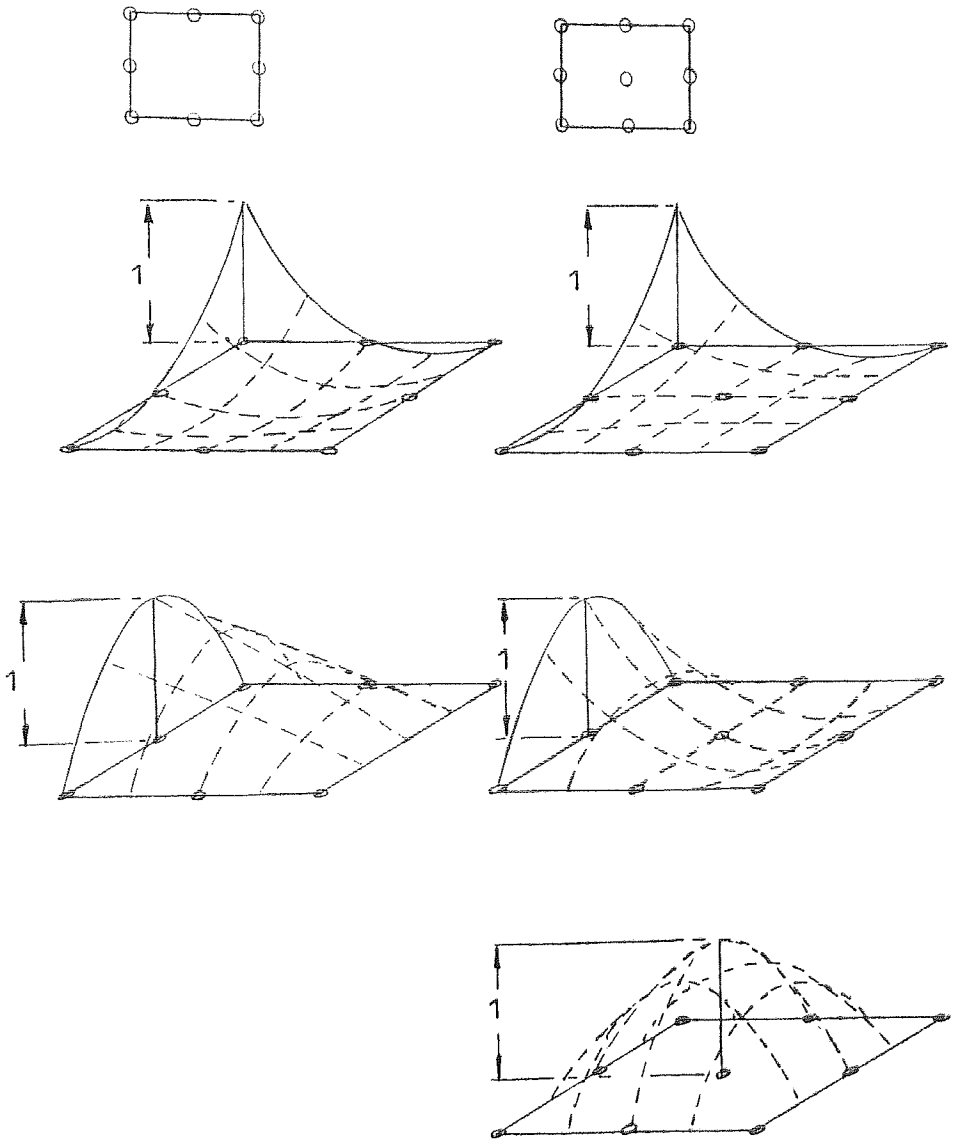


FIG 2.3. Shape functions for an eight-node Isoparametric and a nine-node Lagrange Quadrilateral Element. Ref (1).

$$\{\varepsilon\} = \begin{Bmatrix} \frac{\partial u}{\partial x} \\ \frac{\partial v}{\partial y} \\ \frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} \end{Bmatrix} = [B] \begin{Bmatrix} u_1 \\ v_1 \\ u_2 \\ v_2 \\ \cdot \\ \cdot \\ \cdot \\ \cdot \end{Bmatrix} \tag{2.13}$$

in which

$$[B] = [B_1, B_2 \ldots\ldots] \tag{2.14}$$

with

$$B_i = \begin{bmatrix} \frac{\partial N_i}{\partial x} , & 0 \\ 0 , & \frac{\partial N_i}{\partial y} \\ \frac{\partial N_i}{\partial y} , & \frac{\partial N_i}{\partial x} \end{bmatrix} \tag{2.15}$$

Equation (2.15) is established using equation (2.4) from Section (2.3) in which the displacements u, v are related to the nodal displacements through the shape function [N]. As N_i is defined in terms of ξ and η it is necessary to change the derivatives to $\partial/\partial x$ and $\partial/\partial y$ under the integration sign, equation (2.12).

Noting that

$$\begin{Bmatrix} \frac{\partial N_i}{\partial \xi} \\ \frac{\partial N_i}{\partial \eta} \end{Bmatrix} = \begin{bmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial y}{\partial \xi} \\ \frac{\partial x}{\partial \eta} & \frac{\partial y}{\partial \eta} \end{bmatrix} \begin{Bmatrix} \frac{\partial N_i}{\partial x} \\ \frac{\partial N_i}{\partial y} \end{Bmatrix} = [J] \begin{Bmatrix} \frac{\partial N_i}{\partial x} \\ \frac{\partial N_i}{\partial y} \end{Bmatrix} \tag{2.16}$$

in which [J] is the Jacobian matrix which can be easily evaluated noting that,

$$\frac{\partial x}{\partial \xi} = \frac{\partial x}{\partial N_i} \frac{\partial N_i}{\partial \xi} ,$$

and from equation (2.4)

$$\frac{\partial x}{\partial N_i} = x_i$$

then

$$[J] = \begin{bmatrix} \frac{\partial N_1}{\partial \xi}, & \frac{\partial N_2}{\partial \xi}, & \cdots \\ \frac{\partial N_1}{\partial \eta}, & \frac{\partial N_2}{\partial \eta}, & \cdots \end{bmatrix} \begin{bmatrix} x_1 & y_1 \\ x_2 & y_2 \\ . & . \\ . & . \\ . & . \end{bmatrix} \tag{2.17}$$

We can write

$$\frac{\partial N_i}{\partial x} = [1,\ 0][J]^{-1} \begin{Bmatrix} \frac{\partial N_i}{\partial \xi} \\ \frac{\partial N_i}{\partial \eta} \end{Bmatrix}$$

and

$$\frac{\partial N_i}{\partial y} = [0,\ 1][J]^{-1} \begin{Bmatrix} \frac{\partial N_i}{\partial \xi} \\ \frac{\partial N_i}{\partial \eta} \end{Bmatrix} \tag{2.18}$$

Hence the matrix $[B_i]$ can be formulated. The integral of equation (2.12) is evaluated numerically; thus, $dxdy = \det[J]d\xi d\eta$, and the limits are -1 and 1 in both integrals.

2.4.1 NUMERICAL INTEGRATION OF RECTANGULAR REGIONS

To obtain the integral,

$$I = \int_{-1}^{1}\int_{-1}^{1} f(\xi,\eta)d\xi d\eta \tag{2.19}$$

using the Gaussian quadrature formulae we first obtain the inner integral, keeping η constant, i.e.

$$\int_{-1}^{1} f(\xi,\eta)d\xi = \sum_{j=1}^{n} H_j\ f(\xi_j,\eta) = \psi(\eta) \tag{2.20}$$

Evaluating the outer integral in a similar way, we have

$$I = \int_{-1}^{1} \psi(\eta)d\eta = \sum_{i=1}^{n} H_i \psi(\eta_i)$$

$$= \sum_{i=1}^{n} H_i \sum_{j=1}^{n} H_j f(\xi_j,\eta_i)$$

$$= \sum_{i=1}^{n} \sum_{j=1}^{n} H_i H_j f(\xi_j,\eta_i) \qquad (2.21)$$

In the above the number of integrating points in each direction was assumed to be the same. Clearly this is not necessary and on occasion it may be of advantage to use different numbers in each direction of integration (e.g. the Lagrangian family).

It is of interest to note that the double summation can be interpreted as a single one over (n x n) points for a rectangle. Thus in figure -(2.4), we show the nine sampling points which produce the exact result for a polynomial of the fifth order in each direction.

2.5 LOSS OF ACCURACY IN CURVED ISOPARAMETRIC ELEMENTS

It has been found in practice that the accuracy of the isoparametric element degenerates if the element sides are curved or if the mid-side nodes are displaced. Several reports examining this phenomena have come to light. Thomas (5) investigates the problem from a practical point of view and his work forms part of a survey attempting to give some guidelines for the finite element idealization. This work looks at the effects of element shape distortion, with and without curvature of the sides, and also displacement of the mid-side nodes. The test cases are all based on problems which can be modelled 'exactly' by the normal use of the element.

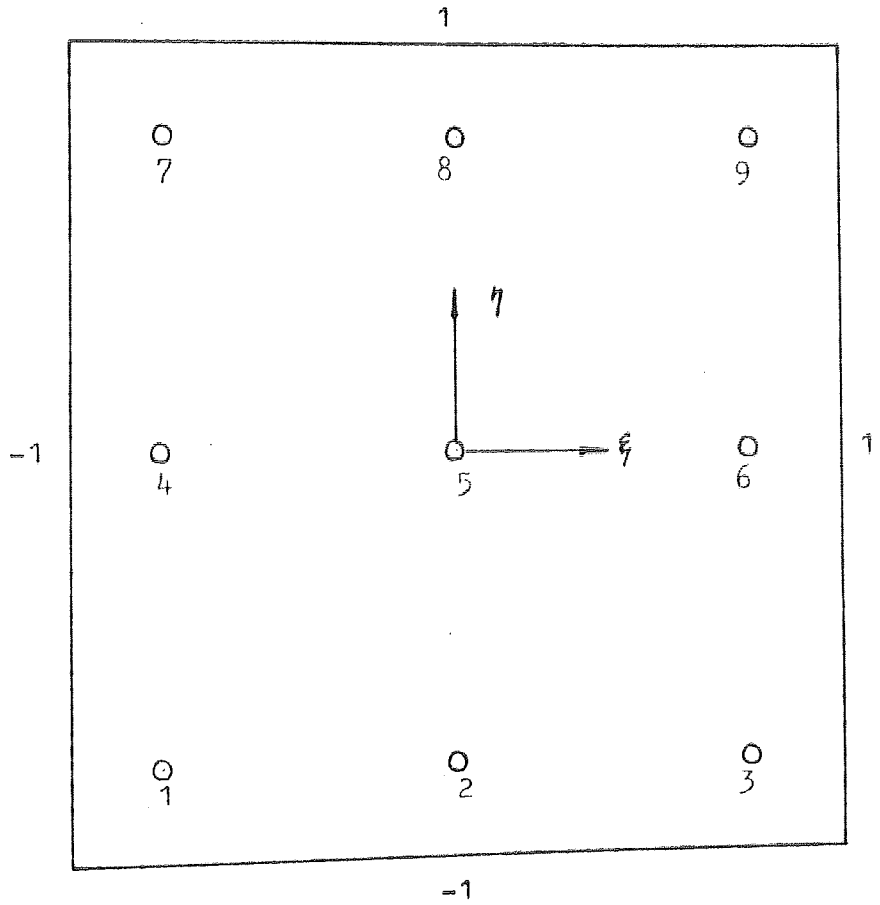


FIG 2.4. Integration points for n = 3. (Exact for polynomial of fifth order in each direction).

Fried (6) examined the loss of accuracy in the element from a theoretical stand point and shows that excessive distortion of a complex isoparametric element might reduce it to a first-order element. He suggests that the element should be curved only in the approximation of curved boundaries and if indiscriminately distorted it will unnecessarily lose accuracy.

Henshell et al.(7) approached the matter in a more basic manner and investigated the change in the shape function as the element becomes distorted. The following section presents the results of these investigations and the advantages of element distortion.

2.5.1 SOME PRACTICAL CONSIDERATIONS OF ELEMENT DISTORTION

The problem of loss of accuracy in distorted isoparametric elements became known to the author whilst working on a simple problem involving a load carrying component. It was discovered that large spurious shear stresses were being computed at stress free boundaries, and equilibrium was not being maintained. This result was first thought to be due to a programming error, possibly a boundary condition fault. In subsequent investigations into the program coding no fault could be detected. Parallel with these investigations, a series of simple tests were being run and it became apparent that the quadratic element performed perfectly when approximating a linearly varying stress field. Fried's paper reveals that distorting the element degenerates its accuracy and to demonstrate this effect two examples were chosen. The first being a plate in uniform tension and the second a bar in a linearly varying stress field.

For the simple plate in tension, excellent results were obtained for both the straight and distorted elements, see figure (2.5 a,b). This result would be expected as the stress field is constant throughout the region under consideration and hence any loss of accuracy would not be noticeable. The quadratic element can model a linearly varying stress field 'exactly' and the second case amplifies the degenerate effects of distortion. It can be seen from figure (2.6 a) that the axial stress varies from zero to some maximum value, the transverse stress and shear stress in this case should be zero. The first idealization and straight sided elements and the axial stress σ_y was found to be correct to five decimal places, similarly the shear and transverse stresses were of the order of 2×10^{-6}. The internal element faces were then slightly distorted having a radius of curvature of approximately 7.0 ins. Discrepancies of 10.0% were found in the axial stress, whilst the shear and transverse stresses attained values within 5% of $\hat{\sigma}_y$ maximum. By subdividing the basic two-element idealization into eight elements and maintaining the same curvature, the discrepancies were reduced by 50%. This indicates that the degeneration is a function of relative element size. See figure (2.6 b). The effects of distortion are examined in greater detail by Thomas (5).

2.5.2 THE SINGULARITY OF ISOPARAMETRIC ELEMENTS

The discovery that the shape function, of the distorted isoparametric elements, becomes modified and causes poor element performance was first noted by Henshell et al.(7). To illustrate this point a three-node isoparametric bar element was used where the mid-side node position was arbitrary. The subsequent dis-

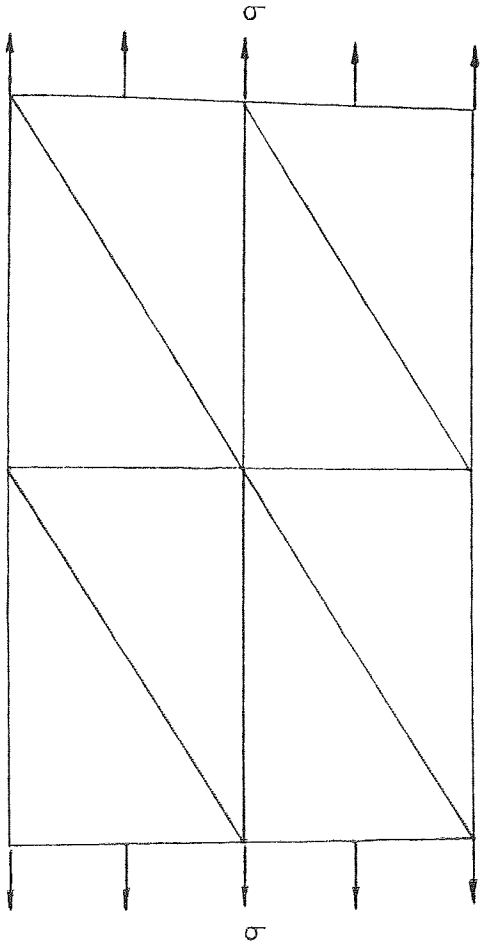


FIG a. Plate in uniform tension.

P = 12 000 lbf

σ = 3 000 lbf/in^2

h = 6.0 in

w = 4.0 in

t = 1.0 in

FIG b. Plate in uniform tension, note curved element faces.

P = 6 000 lbf

σ = 3 000 lbf/in^2

h = 4.0 in

w = 2.0 in

t = 1.0 in

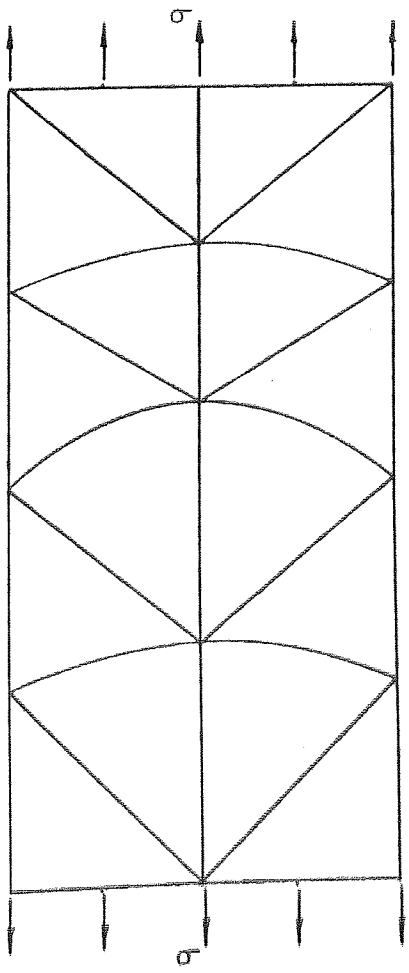


FIG 2.5. Example 1, uniform stress field.

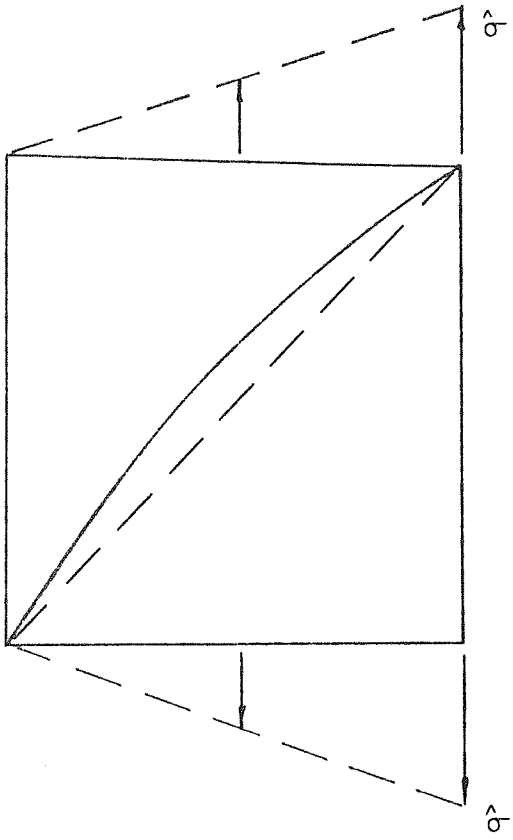


FIG a. Plate in linear stress field.

$\hat{\sigma} = 12*10^3$ lbf/in^2

h = 3.0 in

w = 3.0 in

t = 1.0 in

central displacement

0.3 in

FIG b. Plate in linear stress field.

$\hat{\sigma} = 12*10^3$ lbf/in^2

h = 3.0 in

w = 3.0 in

t = 1.0 in

central displacement

0.15 in

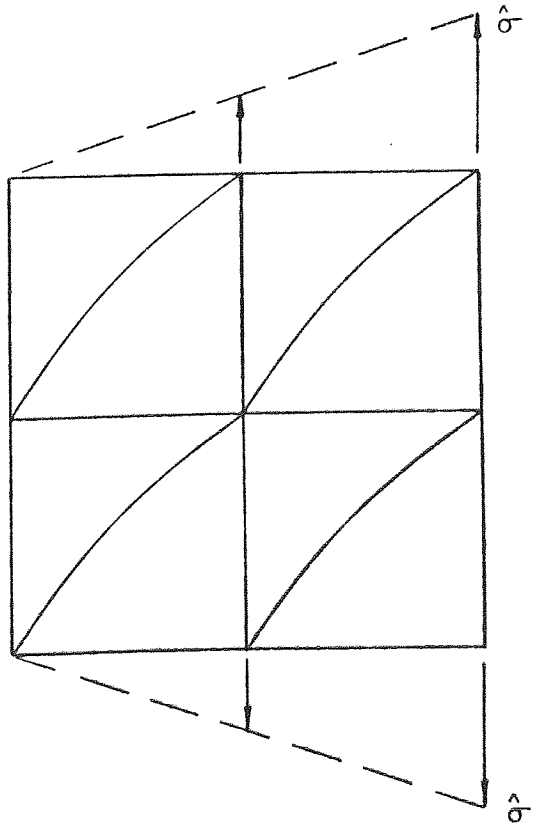


FIG 2.6.

placement formulation shows that a singularity point exists on the edge of the element causing the displacement field to change rapidly. It was later realized by Henshell (55) and Barsoum (56,57), that this element behaviour could be employed to represent the singularity sought in fracture mechanics analysis as discussed in the following chapter.

In order to clarify the above statements, consider a one-dimensional element, which may form the side of an eight-noded isoparametric element. Figure (2.7) shows that the nodes of this element have been mapped from the isoparametric coordinates, ξ = -1, 0 and 1, to the cartesian coordinates x = 0, p and 2. The transformation can be written as,

$$x = C_1 + C_2\xi + C_3\xi^2 \tag{2.22}$$

Substituting corresponding values of ξ in equation (2.22), we can derive the values of C_1, C_2, and C_3, which are $C_1 = p$, $C_2 = 1$, and $C_3 = 1-p$. Rearranging equation (2.22) and using the standard quadratic solution we find,

$$\xi = \frac{-1\pm\sqrt{(1 - 4(1-p)(p-x))}}{2(1-p)} \tag{2.23}$$

and

$$\frac{d\xi}{dx} = \frac{1}{\sqrt{(1-4(1-p)(p-x))}} \tag{2.24}$$

In the isoparametric element the displacement function can be written in terms of ξ analogously to equation (2.22) and it has been shown (8) that the vanishing of the quantity under the radical sign of equation (2.24) is responsible for the stress singularity. Therefore the desired singularity occurs when,

$$x = p - \frac{1}{4(1-p)} \tag{2.25}$$

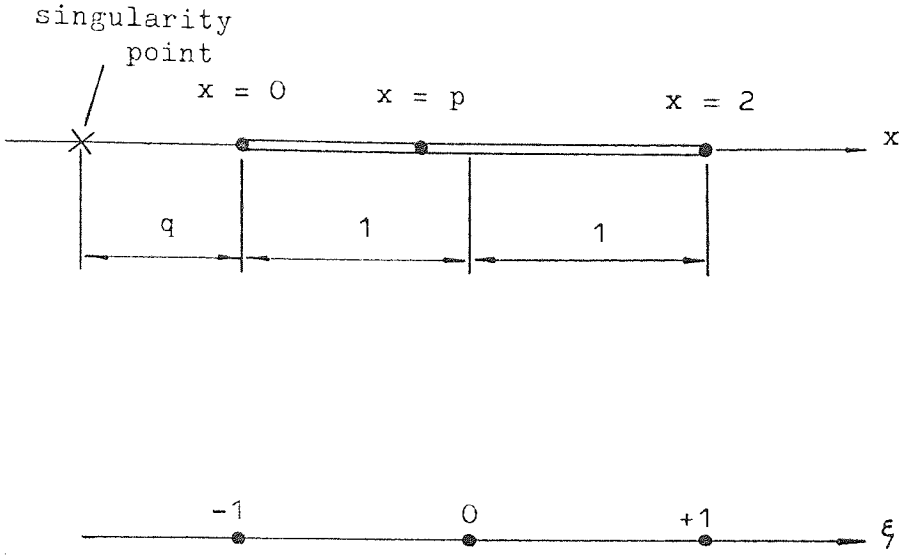


FIG 2.7. Element co-ordinate mapping.

p	q	x
$\frac{1}{2}$	0	$\frac{1}{2} + \xi + \xi^2/2$
$\frac{3}{4}$	$\frac{1}{4}$	$\frac{3}{4} + \xi + \xi^2/4$
1	-	$1 + \xi$

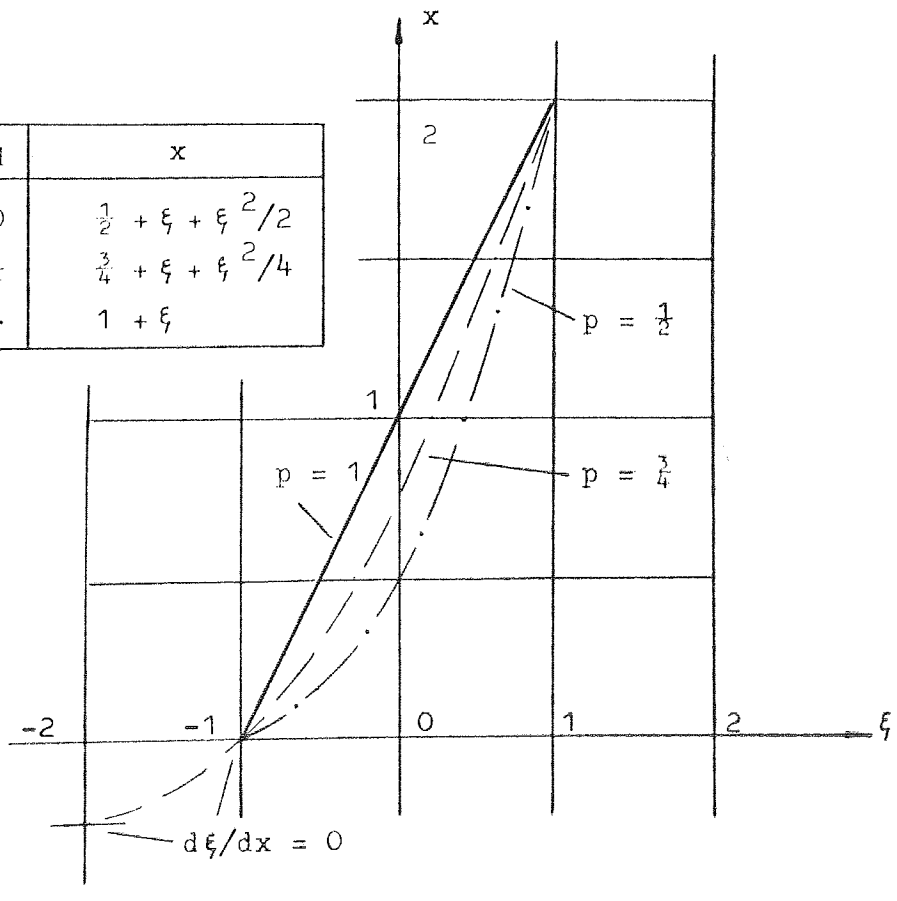


FIG 2.8. Behaviour of quadratic mapping for various mid-side node positions.

Now, it is possible to select where the singularity will occur relative to the element, i.e., this is dependant on the position of the mid-side node as defined by p. Let $x = -q$, then substitution into equation (2.25), gives,

$$p = \frac{(1-q) \pm \sqrt{(q^2 + 2q)}}{2} \tag{2.26}$$

This information can be represented graphically and figure (2.8) illustrates the response of the element for various mid-side node positions. Note the singularity positions are given by $d\xi/dx = 0$. The element can therefore be constructed so that it is sensitive to singularity points outside its domain. This ability has been used to great effect by Lynn and Ingraffea (58), in what are termed 'transition' elements. These elements are further discussed in Chapters Five and Seven.

CHAPTER 3

REVIEW OF SOME TOPICS IN FRACTURE MECHANICS

3.1 INTRODUCTION

The susceptibility of engineering structures to the initiation and propagation of a crack, resulting in catastrophic failure, has led to the development of Fracture Mechanics. During the early part of the industrial revolution it was known that pre-existing flaws could initiate cracks and fractures resulting in structural failure. Prevention of such flaws coupled with better production methods and an increased knowledge of material properties reduced the number of failures to a more acceptable level. An interesting collection of accident reports from the last two hundred years is given by Anderson (9).

With the introduction of all welded designs during the 1940's, a new spate of accidents occurred resulting from structural failure. Out of the 2500 Liberty ships built during this period, 145 broke in two and almost 700 experienced serious failures. The failures often occurred under conditions of low stresses which made them seemingly inexplicable. Extensive investigations were initiated and the work revealed that, here again, flaws and stress concentrations present in the welded structure were responsible for the failures.

The following section describes briefly the foundations of Fracture Mechanics and the subsequent sections review the techniques

employed in the analysis of fracture. The methods employed may be broadly divided into three groups, namely experimental, analytical and numerical. Because this thesis is directly concerned with the numerical analysis of fracture, the two former methods will be given a brief introduction only.

3.2 THE FOUNDATIONSOF FRACTURE MECHANICS

3.2.1 THE GRIFFITH CONCEPT

It was appreciated by the earliest workers in the field that most materials show a tendency to fracture when stressed beyond some critical level. Hence the idea of a 'critical applied stress' was generally accepted, and used in engineering design as the limiting stress level. However it was realised, with the growth of experimental data, that the fracture strength of a material was not consistent, and that the simple premise of a critical applied stress did not hold true.

The breakthrough came in 1920 with a classic paper by A.Griffith (10). He considered an isolated crack in a solid subjected to an applied load, and formulated a criterion for its extension in terms of the fundamental energy theorems of classical mechanics and thermodynamics. Griffith simply sought the configuration which minimized the total free energy of the system, the crack would then be in a state of equilibrium and thus on the verge of extension. Consider the figure (3.1) which defines an elastic body E, containing an internal crack S, of length 2c, and is subjected to applied loads on its boundary L. The first step in the treatment is to write down an expression for the total energy of the system.

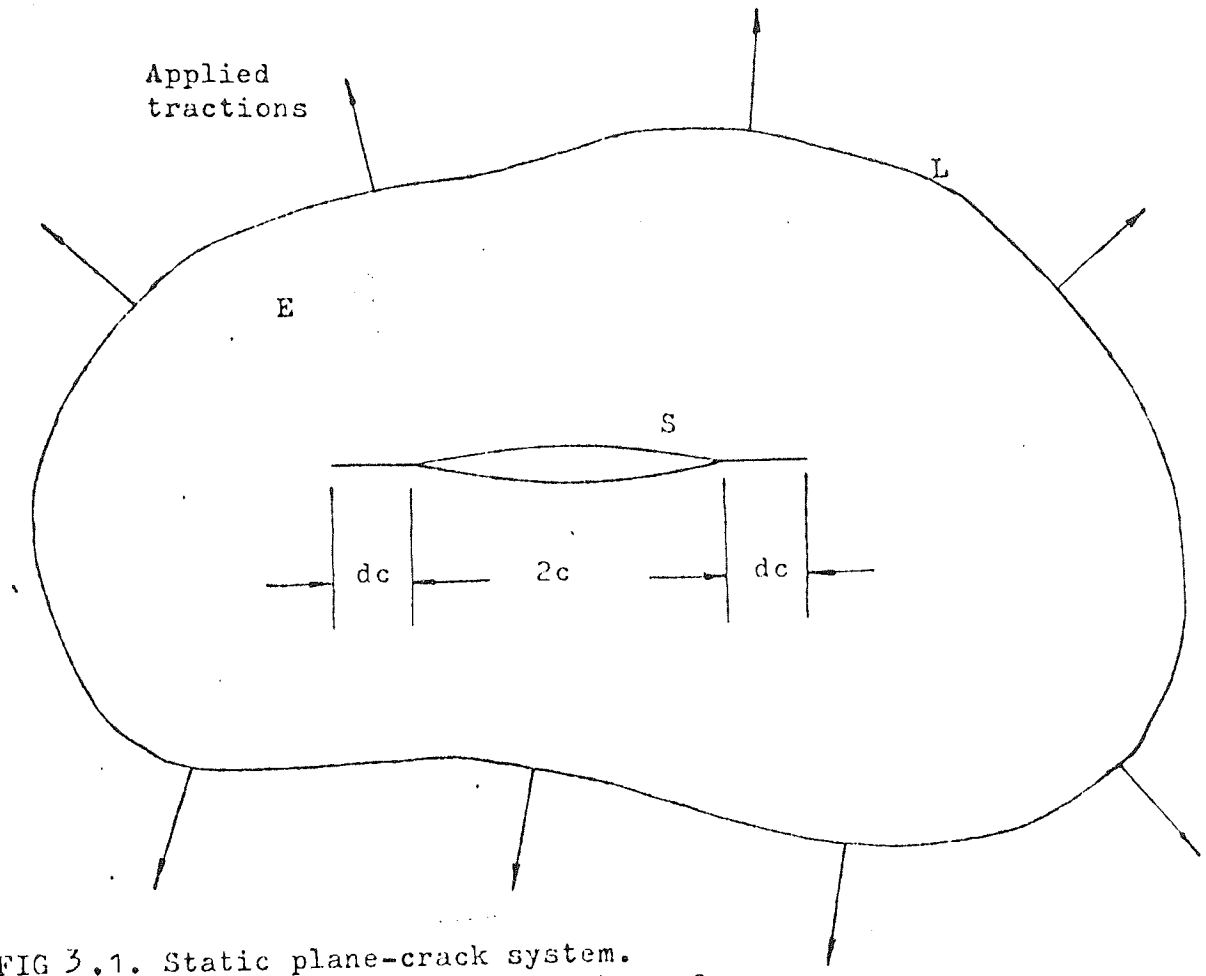


FIG 3.1. Static plane-crack system.
E, elastic medium; S, crack surface;
L, applied loading.

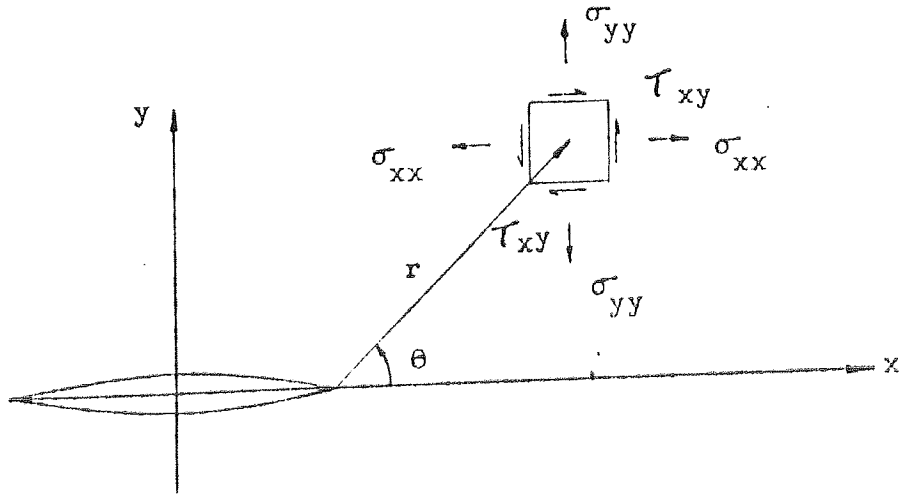


FIG 3.2. Cylindrical polar coordinates with respect to the crack tip.

For the static situation the total energy is the sum of three terms, namely:-

1. Work done by the applied loads as they undergo displacement, W_L.
2. Strain potential energy stored in the elastic medium, U_E.

and 3. The mere act of creating a new crack surface requires the expenditure of free surface energy, U_S.

Hence the total system energy is,

$$U = (-W_L + U_E) + U_S \tag{3.1}$$

For equilibrium therefore,

$$dU/dc = 0 \tag{3.2}$$

where dc is a small increment of crack length. Equation(3.2) represents a criterion for predicting the fracture behaviour of a body, in that, a crack would propagate if $dU/dc > 0$.

In this case, Griffith found that crack growth will be unstable if,

$$\sigma_{crt} \geq 4\sqrt{\mu\gamma/(\pi c(\kappa+1))}$$

where μ is the shear modulus, γ is the specific surface energy, and c is the half crack length. The critical stress σ_{crt} corresponds to the value of σ_y required to cause instability.

3.2.2 THE STRESS INTENSITY FACTOR APPROACH TO FRACTURE

The stress intensity factor concept focuses attention on the local stress field around the crack tip. Westergaard (11), using the method of complex functions, derived singular expressions for the stresses in the vicinity of a crack tip and it was noted by Irwin (13, 14), that they could be interpreted in the following form:

$$ {}_m\sigma_{ij} = K_m r^{-\frac{1}{2}} \, {}_m f_{ij}(\theta) \qquad (3.3) $$

$$ {}_m u_i = (K_m r^{\frac{1}{2}}/\mu) \, {}_m g_i(\theta) $$

where $i,j = 1,2$

$m = I, II, III$

K_m = stress intensity factor of the m^{th} mode

and

${}_m f_{ij}(\theta)$, ${}_m g_i(\theta)$ are functions given in Appendix (9.1)

The cylindrical polar coordinates, r and θ, refer to the crack tip, see figure (3.2). It could be shown that the mechanical energy released during incremental crack extension is independant of the loading configuration and that the strain energy release rate is given by,

$$ G = -\partial U/\partial A $$

where U = potential energy stored in the elastic medium.

A = crack area.

The factor K, represents the stress intensity and originated from Irwin's concept of strain energy release rate, denoted by G. It was demonstrated by Irwin that G and K are equivalent for describing the stress field intensity in the neighbourhood of the crack tip. In fact

$$ G = \frac{1 + \kappa}{8\mu} K^2 \qquad (3.4) $$

where μ is the shear modulus and κ is a function of Poisson's ratio ν.

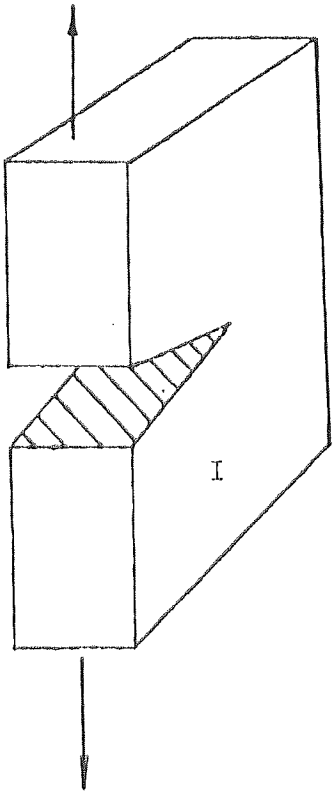
This practical approach proposed by Irwin, does enable the onset of unstable fracture to be predicted in real structures, provided suitable fracture experiments on cracked specimens have been carried out. Thus, the material property K_c, can be defined as the stress intensity operative at the point of fracture and is analogous to the limiting stress at the onset of yield. Therefore a characteristic spatial distribution of stresses was found, each specific case characterized by the stress intensity factor, K. The three basic stress environments experienced at the crack tips are, the opening mode K_I, the sliding mode K_{II}, and the tearing mode K_{III}, as depicted in figure (3.3).

It became apparent that, except in cases where fracture occurred at very low stress levels, some account of plastic behaviour was necessary. The value of K_I at the point of fracture was found to be strongly dependant on plate thickness, and only above a certain thickness could the critical value (K_{I_c}), be regarded as a material property.

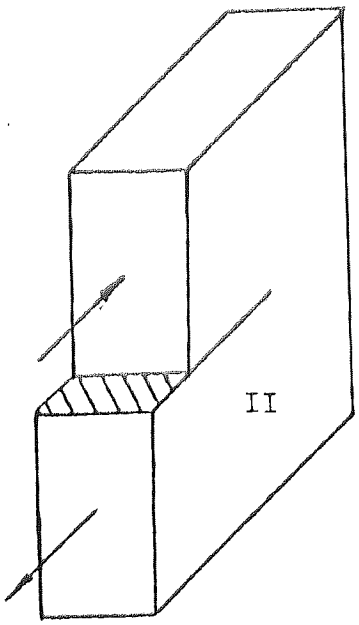
Because of the high stress concentration ahead of the crack tip the material becomes plastically deformed. Irwin (14) and Orowan (15), noted that the energy required for crack growth, was larger than the surface energy to create the new free surface, and suggested that this was due to the extra energy needed in forming the plastic zone at the tip of the advancing crack. To allow for this in the calculation of the stress intensity factor, Irwin made the crack length longer by adding a correction parameter. The effective crack length is given by

$$a_{eff} = a + \delta \qquad (3.5)$$

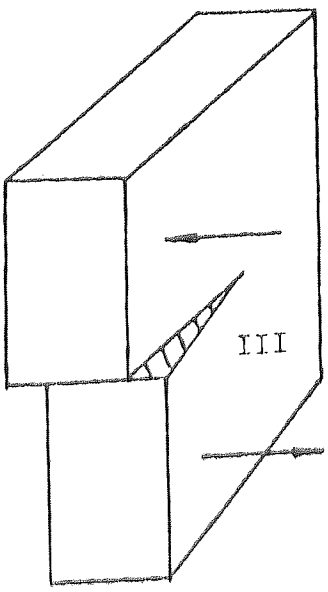
where δ is the correction due to plasticity.



Mode I Opening mode.



Mode II Sliding mode.



Mode III Tearing mode.

FIG 3.3. The three modes of cracking.

3.2.3 THE STRAIN-ENERGY-DENSITY FACTOR

The critical stress intensity approach in fracture mechanics is analogous to the maximum stress criterion applied to a simple tension specimen and cannot be used in combined loading situations. For this reason it has limited applications in structural design. An alternative interpretation of fracture phenomena, presented by Sih (16,17) is the strain energy density theory. Consider the three dimensional case of a crack in a combined stress field, figure (3.4). We can express the strain energy stored in the element dV = dxdydz thus,

$$dW = [\frac{1}{2E}(\sigma_x^2 + \sigma_y^2 + \sigma_z^2) - \frac{\nu}{E}(\sigma_x\sigma_y + \sigma_y\sigma_z + \sigma_z\sigma_x) + \frac{1}{2\mu}(\tau_{xy}^2 + \tau_{xz}^2 + \tau_{yz}^2)]dV \qquad (3.6)$$

where the symbols have their usual meaning.

Substituting the expressions for the local crack tip stresses, equation (3.3) of the previous section, yields the quadratic form for the strain energy density function.

$$\frac{dW}{dV} = \frac{1}{r}(a_{11}K_I^2 + 2a_{12}K_IK_{II} + a_{22}K_{II}^2 + a_{33}K_{III}^2) + \ldots\ldots \qquad (3.7)$$

The higher order terms in r have been neglected and we note that the strain energy density function near the crack tip possesses a 1/r energy singularity. The quadratic,

$$S = a_{11}K_I^2 + 2a_{12}K_IK_{II} + a_{22}K_{II}^2 + a_{33}K_{III}^2 \qquad (3.8)$$

represents the amplitude or the intensity of the strain energy density

field, and it varies with the polar angle θ, shown in figure (3.4). The coefficients a_{11}, a_{12},..... a_{33} are given by,

$$a_{11} = \frac{1}{16\mu}[(3 - 4\nu - \cos\theta)(1 + \cos\theta)]$$

$$a_{12} = \frac{1}{16\mu}[\cos\theta - (1 - 2\nu)]2\sin\theta$$

$$a_{22} = \frac{1}{16\mu}[4(1 - \nu)(1 - \cos\theta) + (1 + \cos\theta)(3\cos\theta - 1)] \qquad (3.9)$$

$$a_{33} = \frac{1}{4\mu}$$

where ν is the Poisson's ratio, μ is the shear modulus of elasticity K_I - K_{III} and the stress intensity factors.

The fundamental hypotheses on unstable crack growth in the Sih theory are as follows,

1. Crack initiation takes place in a direction determined by the stationary value of the strain energy density factor, i.e.

$$\frac{\partial S}{\partial \theta} = 0, \quad \text{as } \theta = \theta_0 \qquad (3.10)$$

2. Crack extension occurs when the strain energy density factor reaches a critical value, i.e.

$$S_c = S(K_I, K_{II}, K_{III}), \text{ for } \theta = \theta_0 \qquad (3.11)$$

The difference between S and S_c is analogous to the difference between K and K_c, and thus S_c is also a measure of the resistance of a material against fracture. The additional feature of the strain energy density approach, is that the direction of crack initiation can be found and also a single parameter, S_c, can be used as a material constant that serves as an indication of the fracture toughness of the material.

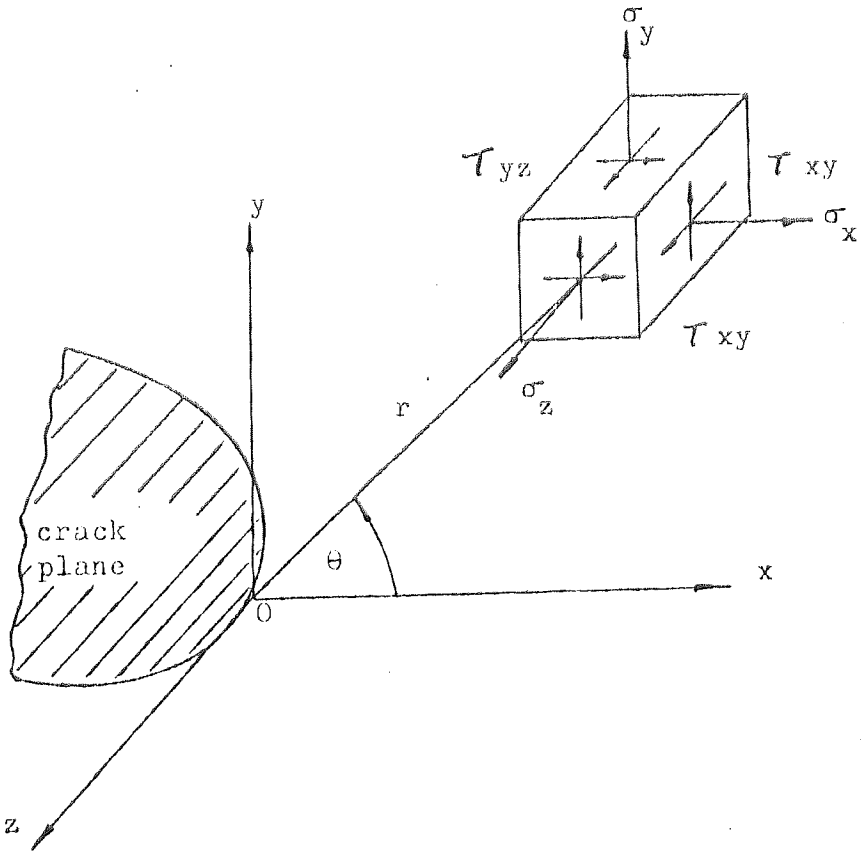
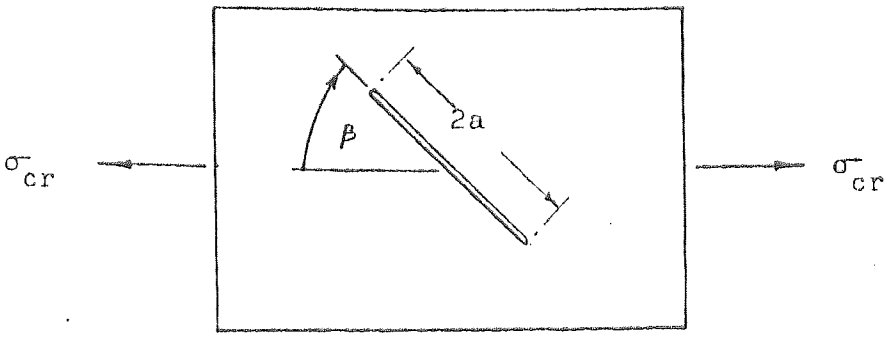


FIG 3.4. Stress components near crack border.



Test specimen configuration.

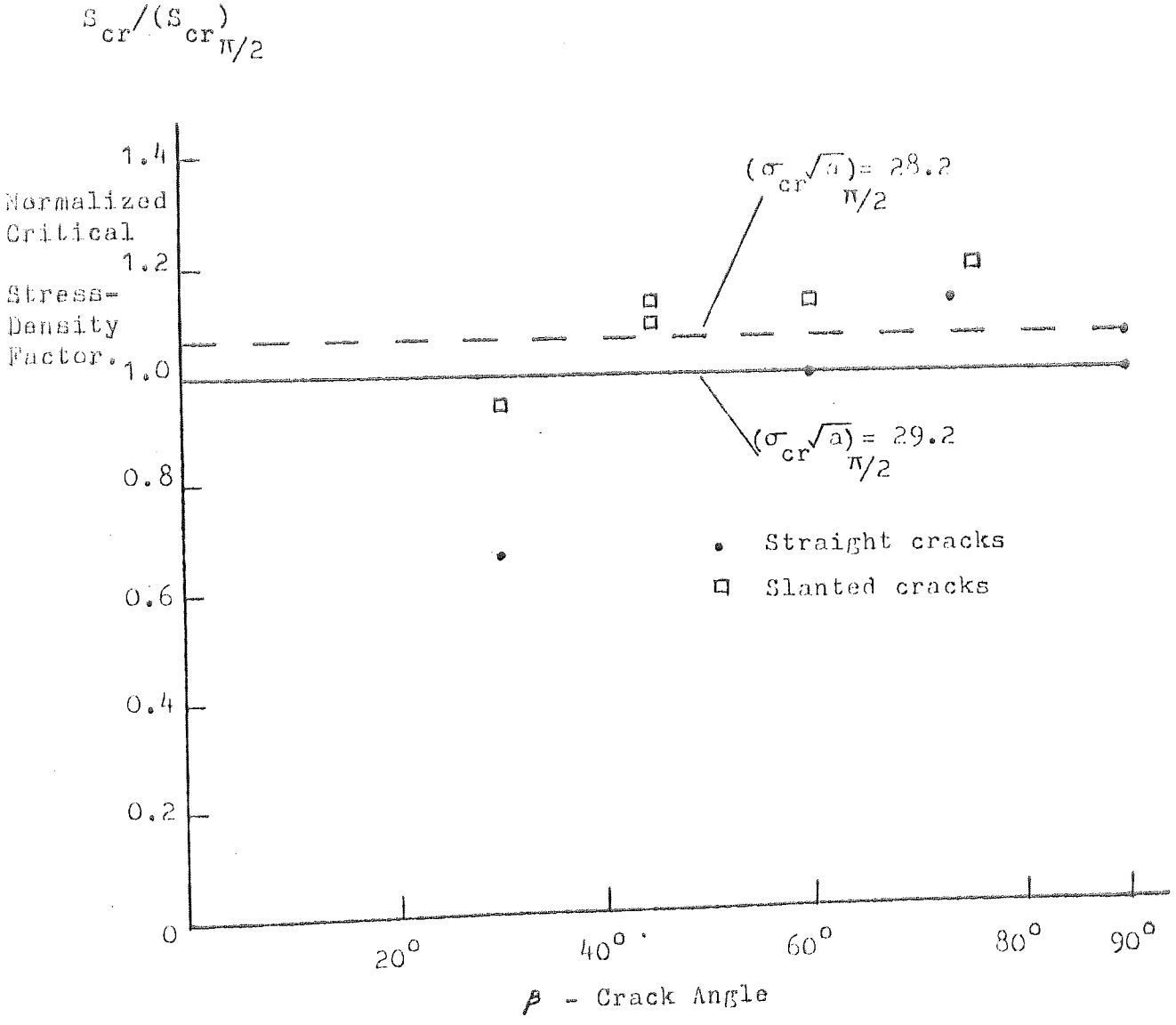


FIG 3.5. Critical Density Factor as a material constant.

The invariant property of the strain energy density factor can be seen from figure (3.5), where the normalized factor S_{cr} is plotted against crack angle β. It can be seen that the factor S_{cr} remains essentially constant (16).

3.3 EVALUATION OF STRESS INTENSITY FACTORS

3.3.1 EXPERIMENTAL METHODS

Experimental methods may either involve direct measurements on a model or use a known relationship between the measurable quantity and the stress intensity factor. In this section two methods are described as an example of the above techniques.

I. COMPLIANCE

It was shown by Irwin and Kies (18), that the strain energy release rate G, could be written in terms of the applied load Q and the change with respect to crack area A of the compliance C of the test specimen as,

$$G = \frac{Q^2}{2} \frac{dC}{dA} \tag{3.12}$$

Using the relationship,

$$G = \frac{1 + \kappa}{8\mu} K_I^2 \tag{3.13}$$

the stress intensity factor is given by

$$K_I = 2Q\left[\frac{\mu}{(\kappa+1)} \frac{dC}{dA} \right]^{\frac{1}{2}} \tag{3.14}$$

Measuring the compliance C, over a range of crack lengths, and determining the derivative of the plotted results K_I may be found, using expression (3.14). Considerable care is required if satisfactory results are to be obtained. See Strawley et al. (19).

II. PHOTOELASTICITY

Of the optical methods for determining stress intensity factors photoelasticity has been most used. The technique is well known and experimental equipment and birefringement materials are readily available. Two methods are presented here which involve measurements of the stress near to a simulated crack. These are based on the equation for maximum shear stress, τ_m, given by

$$\tau_m = \frac{1}{2\sqrt{2\pi r}} [(K_I \sin\theta + 2K_{II}\cos\theta)^2 + (K_{II}\sin\theta)^2]^{\frac{1}{2}} \qquad (3.15)$$

where r and θ are polar coordinates at the crack tip.

The first method involves measuring the shear stress τ_m on lines perpendicular to and through the crack tip, represented by,

$$\tau_m = \frac{(K_I^2 + K_{II}^2)^{\frac{1}{2}}}{2\sqrt{2\pi r}} \qquad (3.16)$$

and also on a line outside and collinear with the crack given by,

$$\tau_m = \frac{K_{II}}{\sqrt{2\pi r}} \qquad (3.17)$$

Having evaluated τ_m the above expressions can be used to determine K_I and K_{II}. See references (20,21).

The second method, used by Smith and Smith (22,23), utilizes the condition $\partial\tau_m/\partial\theta = 0$, which can be restated as,

$$\left(\frac{K_{II}}{K_I}\right)^2 - \frac{4}{3}\left(\frac{K_{II}}{K_I}\right)\cos 2\theta_m - \frac{1}{3} = 0 \qquad (3.18)$$

The angle θ_m, is that at which a tangent to the isochromatic fringes is perpendicular to the radius r. By obtaining θ_m the ratio K_{II}/K_I can be determined from equation (3.18) and hence, knowing τ_m, K_I and K_{II} can be found using equation (3.15).

III. CRACK OPENING DISPLACEMENT METHOD

The crack opening displacement is a measure of the resistance of material to fracture initiation under conditions where gross plastic deformation occurs and linear elastic fracture mechanics becomes invalid. The main objective of the crack opening displacement (COD) test is to determine the critical COD at the tip of a sharp crack at the onset of crack extension. This is done by measuring the displacement at the mouth of the notch using some form of clip gauge and by performing a suitable calculation. It is difficult to interpret the crack tip opening displacement δ, from the clip gauge displacement q. Several methods have been proposed (62,63), and all the methods assume that plastic deformation occurs by a hinge mechanism about a centre of rotation at a depth r(W-a) below the crack tip. Here, r is the notch root radius, W is the specimen width and (a) is the crack length.

Having calculated the critical crack tip opening displacements, the mode I stress intensity factor is given by the relationship,

$$\delta_{c_t} = \frac{K_I^2(1 - \nu^2)}{E\lambda\sigma_{yield}} \quad \text{(plane strain)}$$

or

$$\delta_{c_t} = \frac{K_I^2}{E\lambda\sigma_{yield}} \quad \text{(plane stress)}$$

where the symbols have their usual meaning, and values for λ have been reported in the literature. A review of these values is available in (64).

More recently moiré and speckle pattern techniques have been introduced for this work, but little, as yet, has been reported on this new development.

3.3.2 ANALYTICAL METHODS

The methods considered here are those which satisfy all the boundary conditions exactly. Such methods have the advantage of leading to explicit expressions for stress intensity factors; but only certain classes of problems can be solved. In deriving the stress intensity factor use is made of the formal definition,

$$K_N = \lim_{r \to o} \sigma_N \sqrt{2\pi r} \qquad (3.19)$$

where σ_N is appropriate to the mode of cracking. For simplicity, all the methods are described for a crack of length 2a along the x axis with the origin of the x,y coordinates at the crack centre.

I. WESTERGAARD STRESS FUNCTIONS

Westergaard (11) formulated an Airy stress function F, which for mode I, takes the form

$$F_I = \mathrm{Re}[\bar{\bar{Z}}_I] + y\ \mathrm{Im}[\bar{Z}_I]$$

where Z_I is the Westergaard stress function.

$\bar{Z}_I$ and $\bar{\bar{Z}}_I$ are defined by

$$\frac{d\bar{\bar{Z}}_I}{dz} = \bar{Z}_I \quad \text{and} \quad \frac{d\bar{Z}_I}{dz} = Z_I$$

where $z = x + jy$. The cartesian components of stress, in terms of F_I, are

$$\sigma_x = \frac{\partial^2 F_I}{\partial y^2}, \quad \sigma_y = \frac{\partial^2 F_I}{\partial x^2} \quad \text{and} \quad \tau_{xy} = -\frac{\partial^2 F_I}{\partial x \partial y} \tag{3.20}$$

The simplest crack configuration studied by Westergaard was that of a crack in an infinite sheet subjected to uniform biaxial tension σ_∞ at infinity; the stress function is,

$$Z_I = \sigma_\infty z/\sqrt{z^2 - a^2} \tag{3.21}$$

Westergaard also studied a crack opened by wedge forces and an infinite series of collinear cracks under various loading conditions. The method can be extended to modes II and III, and comparison of the stress field in terms of the Westergaard stress function Z_N with equation (3.19) shows that the stress intensity factor is given by,

$$K_N = \sqrt{2\pi} \lim_{z \to a} \{\sqrt{z - a}\bar{Z}_N\} \tag{3.22}$$

where N = I, II, III.

Several workers (41, 65, 66) have used Westergaard's method for solving crack problems.

II. COMPLEX STRESS FUNCTIONS

Mushkelishvili's complex stress function approach (25) enables the Airy stress function F to be written in terms of two complex functions $\phi(z)$ and $\psi(z)$, as,

$$F = Re[\bar{z}\phi(z) + \int\psi(z)dz] \tag{3.23}$$

which yields from equation (3.20) with F_I replaced by F,

$$\sigma_x + \sigma_y = 4Re[\phi'(z)] \tag{3.24}$$

and

$$\sigma_x - \sigma_y + j2\tau_{xy} = 2[\bar{z}\phi''(z) + \psi'(z)] \tag{3.25}$$

From equation (3.23) and the known properties of $\phi'(z)$ (25) it can be shown that,

$$K_I - j\,K_{II} = 2\sqrt{2\pi} \lim_{z \to z_1} \{\sqrt{z - z_1}\phi'(z)\} \tag{3.26}$$

This method of determining stress intensity factors is superior to the Westergaard method in as much as conformal mapping can be used to map cracks into holes. This is important from a practical view point as many cracks initiate from areas of rapid stress change. This method is discussed in greater detail by Sih (26).

For practical components, however, these techniques are limited to certain idealized situations and thus numerical methods are adopted as described in the following section.

3.3.3 NUMERICAL METHODS

The analysis of fracture mechanics problems has been one of the most active branches of numerical methods in structural mechanics in the 1970's. The development of finite element methods to fracture mechanics has been so extensive that it is proposed to examine this branch of numerical analysis only and reference to alternative techniques may be found in (27,28,29).

The manner in which numerical methods have been used can be grouped under two broad headings: firstly, methods for the interpretation of the finite element results and secondly the construction of formulations which model the singularity at the crack tip. The methods of both groups will be discussed in brief only and the reader is referred to the original papers for a more detailed treatment.

I. INTERPRETATION OF THE FINITE ELEMENT ANALYSIS RESULTS

Initial efforts with conventional elements, demonstrated clearly that hundreds, or perhaps thousands of simple elements are required to achieve a solution accuracy within 5% (30,31). The complexity of the problem and the methods employed in the analysis of the results, dictate the number of elements required for an accurate solution. Higher order elements may be used to obtain a more efficient analysis, but care must be exercised, as studies by Fried et al. (32) and Tong et al (33) disclose.

i. EVALUATION BASED ON CLASSICAL SOLUTIONS

The displacements in the vicinity of the crack tip are given in Appendix (9.1). This expression can be rearranged to yield,

$$K_I = 2\mu \sqrt{\frac{2}{r}} \frac{u_A}{f(\theta,\nu)} \tag{3.27}$$

where $f(\theta,\nu)$ represents the first term in the expression and u_A is the displacement at location (A). A similar expression holds for v_A. Equation (3.27) can be evaluated at nodal points along a radial line emanating from the crack tip, producing a plot, such as that given in figure (3.6).

As stated earlier the number of elements required to achieve a solution accuracy within 5% is immense, thus, in using a reasonable element mesh density, the finite element displacement solution near the crack tip is inaccurate. By disregarding the results close to the crack tip and extrapolating to $r = 0$, a solution for K_I can be obtained. This method was introduced by Kobayashi (31) and later an extrapolation procedure was devised by Chan (30). The method has recently been employed in investigations on distorted isoparametric elements (34), as described in subsection II(ii) and (iii), below.

ii. EVALUATION BASED ON STRESSES

A similar method to that described above utilizes the classical expression for the local stress at the crack tip. From Appendix (9.1), the stress intensity K_I can be interpreted in terms of the radial stress (σ_r) as,

$$K_I = \frac{2\sigma_r}{f(\theta)} \sqrt{2r} \tag{3.28}$$

Again $f(\theta)$ represents the first term in the expression.

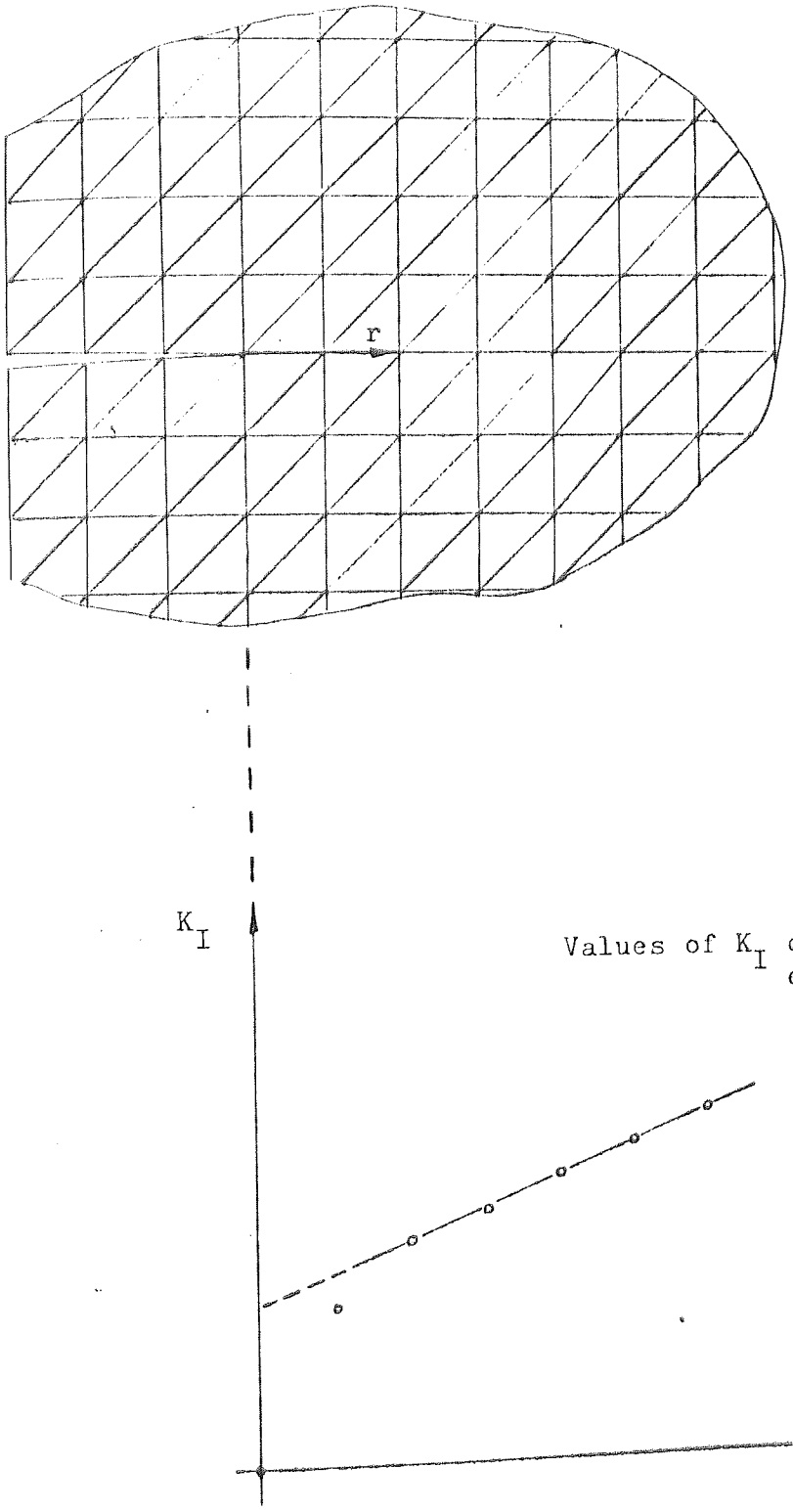


FIG 3.6. Extrapolation to obtain K_I at r=0.

Equation (3.28) is evaluated at points along a radial line as before, and the results can be extrapolated to determine K_I at r = 0. This method gives erratic results if simple constant strain elements are used and studies (35), indicate that the procedure is less accurate than the corresponding displacement method. This is not unexpected since the stresses in an assumed-displacement solution are themselves inferior in accuracy to the displacements.

iii. CORRELATION METHOD

A procedure for the determination of stress intensity factors has recently been proposed by Murakami (36). The method involves performing two finite element analyses, the first analysis is carried out on the component without a crack to determine the hypothetical crack tip stress (σ_g) and secondly the crack tip stress (σ_{TIP}) is computed for the component with the crack. Murakami argues that the ratio of the stress differences, $(\sigma_{TIP} - \sigma_g)_A/(\sigma_{TIP} - \sigma_g)_B$, for problems A and B, is approximately equal to the ratio of the stress intensity factors K_A/K_B. Hence, if K_A is known then K_B can easily be computed. Conversely the stress difference $(\sigma_{TIP} - \sigma_g)$ is directly related to the stress intensity factor K, via a conversion factor. For two dimensional problems the centrally cracked plate in uniform tension is used to determine the standard correlation between $(\sigma_{TIP} - \sigma_g)$ and K_I. The final expression for the stress intensity factor is given by,

$$K_I = \frac{(\sigma_{TIP_I} - \sigma_{g_I})}{\sigma_c} \sqrt{\pi c} \qquad (3.29)$$

where σ_c is the correction factor and

c is the crack length.

The method has been applied to semi-elliptical three dimensional crack problems and is currently being used in crack propagation work (39).

iv. STRAIN ENERGY RELEASE RATE

The relationship between the strain energy release rate G, and the stress intensity factor K, has been previously demonstrated by equation (3.4). By evaluating the strain energy U for particular cases as the crack is opened, the strain energy release rate can be obtained. Thus, on a finite difference basis,

$$G = \frac{\delta U}{\delta A} \tag{3.30}$$

where δA is the increase in surface area. Hence subsequent substitution in equation (3.4) leads to the stress intensity factor. The method is not as sensitive to grid refinement in the vicinity of the crack tip as were methods i, and ii, and consequently a relatively coarse mesh can be used.

It is possible to determine the change in strain energy, using only the original cracked configuration. The theoretical basis of this procedure is as follows. Examining the stiffness equation, we have,

$$[K]\{u\} = \{b\} \tag{3.31}$$

where $\{u\}$ is a vector of displacements, $\{b\}$ is a vector of corresponding nodal loads, and $[K]$ is the structural stiffness matrix. Now the total potential energy of the structure is given by,

$$U = \frac{1}{2}\{u\}^t[K]\{u\} - \{u\}^t\{b\}$$

Consider a small virtual increase δa in crack length, with no change in external loads, then the energy release rate G is obtained from the variation of u with respect to the constant load, i.e.

$$\delta U = \frac{1}{2} \{u\}^t [\delta K] \{u\} + \{\delta u\}^t [K] \{u\}$$

$$- \{\delta u\}^t \{b\} - \{u\}^t \{\delta b\}$$

Using equation (3.31), this reduces to,

$$\delta U = \frac{1}{2} \{u\}^t [\delta K] \{u\} - \{u\}^t \{\delta b\} \tag{3.32}$$

Providing the loading is due to forces outside the crack tip elements, the vector $\{\delta b\}$ is null, and the final term is dropped, thus,

$$\delta U = \frac{1}{2} \{u\}^t \{\delta K\} \{u\}$$

then

$$G = \frac{\delta U}{\delta a} = - \frac{1}{2} \{u\}^t \{\frac{\delta K}{\delta a}\} \{u\} \tag{3.33}$$

Therefore the stress intensity factor can be obtained by substitution into equations (3.4). Both Parks (37) and Hellen (38) have exploited this idea, which has been termed the 'differential stiffness procedure'.

v. CRACK CLOSURE INTEGRALS

Yet another procedure for avoiding two analyses is based on the crack closure integral. (Rybicki and Kanninen (40)). This method calculates the amount of work required to close the crack at the node points immediately ahead of the crack tip. Using the properties of

the so-called crack closure integral (41), an analytical solution (42) in the vicinity of the crack, and the customary approximation of the finite element analysis, Kanninen establishes expressions for the strain energy release rate in terms of the nodal point displacements and forces of the finite element solution.

vi. LINE INTEGRALS

A quantity of importance in fracture studies is the J-integral, first introduced by Rice (43) and subsequently used by several investigators for inelastic as well as elastic situations (67,68). The J-integral, taken over any boundary Γ of a region containing a crack, results in a constant, given by

$$J = \int_{\Gamma} (U\, dy - T.\frac{\partial u}{\partial x}\, ds) \tag{3.34}$$

where U is the strain energy density, T is the traction vector defined according to the outward normal along Γ, u is the displacement vector, and ds is the element of arc along Γ. (see fig. 3.7). The J-integral is linked directly with the strain energy release rate and hence can be used to determine the stress intensity factor. It can be shown that the stress intensity factor K_I, is related to J by

$$K_I = \left(\frac{JE}{(1-\nu^2)}\right)^{\frac{1}{2}} \tag{3.35}$$

for plane strain

This method has been applied to fracture problems by Chan et al.(30) and Anderson et al.(44), and the concept has been used extensively by other investigators.

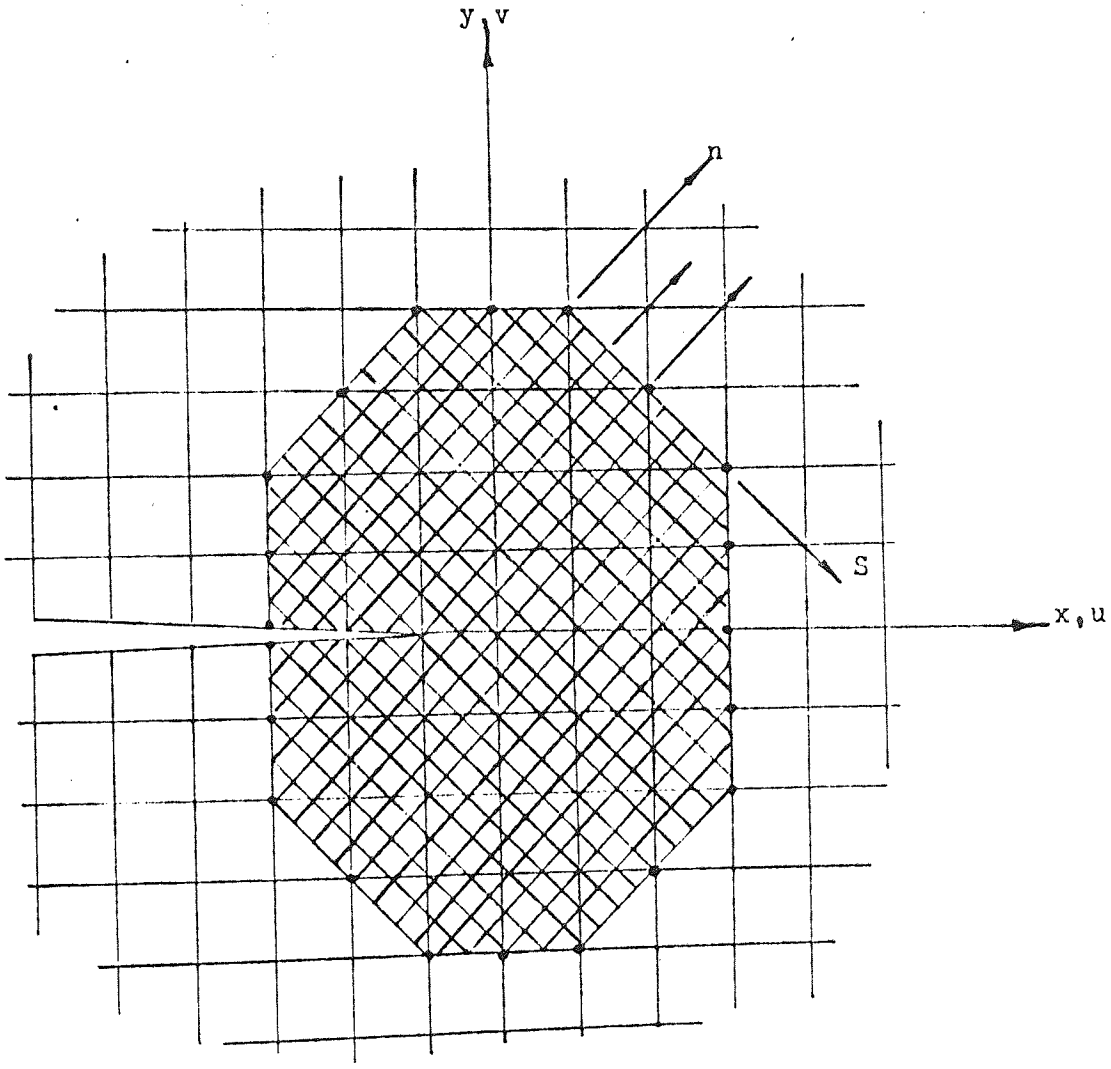


FIG 3.7. Boundary integral coordinate system.

The J-integral is not the only integral which can be employed, a Somogliana-type singular integral representation has been proposed by Stern et al.(45) and Soni et al.(46). This integral has the advantage of giving the stress intensity factors directly.

II. SINGULARITY FORMULATIONS

The most appealing approach to finite element fracture mechanics analysis is that which takes explicit account of the crack tip singularity. Because the majority of finite element programs are based on the assumed displacement finite element formulation, many of the embedded singularity procedures incorporate a displacement field which includes the desired singularity. Singularity formulations can be divided into four groups,

1. Methods which utilize the displacement functions already available in classical mechanics.
2. Polynomial displacement functions.
3. Distorted isoparametric element formulations.
4. Hybrid elements.

i. CLASSICAL SOLUTION BASED FUNCTIONS

The series solutions established by Westergaard, Muskhelishvili, and others, for the stress distribution around a crack tip in an infinite plate, take the form of equation (3.3). The expanded expressions are given in Appendix (9.1). The first term in the stress series is proportional to $r^{-\frac{1}{2}}$ and represents the singularity in the stress distribution at the crack tip. By expanding the expression, additional unknown parameters can be employed as the assumed displacements in a multi-node finite element. The incorporation of the

classical solution into a finite element was introduced by Byskov (47), who formulated a singularity element based on the equations of Muskhelishvili (25). Figure (3.8) shows the element configuration, which is sited at the crack tip and surrounded by conventional, constant strain triangles.

The procedures required to incorporate a singularity element, into a conventional finite element mesh can be conveniently described with the aid of Lagrange multipliers. See Richards (76,77). Referring to Chapter Six, the algebraic manipulations result in four basic equations accordingly,

$$[K_c]\{\alpha\} - [A]^t\{\lambda\} = 0 \tag{3.36}$$

$$[K_{11}]\{q_1\} + [K_{12}]\{q_2\} - \{Q_1\} + \{\lambda\} = 0 \tag{3.37}$$

$$[K_{21}]\{q_1\} + [K_{22}]\{q_2\} - \{Q_2\} = 0 \tag{3.38}$$

$$[A]\{\alpha\} = \{q_1\} \tag{3.39}$$

where $\{\lambda\}$ represents the Lagrange multiplier,

$[K_c]$ stiffness matrix for the singularity element.

$[K_{ij}]$, $\{q_i\}$ and $\{Q_i\}$ represent the partitioned stiffness matrices,

$\{\alpha\}$ parameters in the singularity expression which contain the stress intensity factors,

and $[A]$ connection matrix relating the displacements at the nodes on the special element and the parameters in the singularity expression.

We can use these equations to describe two fundamental approaches, the merits of which will be discussed more fully in Chapter Seven.

The first method can be adapted to a conventional finite element program with minor adjustments to the stiffness matrix [K]. The method solves for the vector $\{q_i\}$, being the nodal displacements and then evaluates the parameters $\{\alpha\}$ via expression (3.39). Clearly if the connecting matrix [A], in the above expression, is not square then some pseudo inverse method must be employed, in order to obtain the vector $\{\alpha\}$. It is possible to minimize computational effort by making the matrix [A] square, i.e. by taking enough terms in the singularity expression to balance the displacement degrees of freedom on the elements boundary.

This approach was used by Fawkes (48) and his element is shown in figure (3.9); it is rectangular and described by 8 nodes (16 dof). By using the stresses derived from the Williams stress function (49), the element stiffness $[K_c]$ can be found via the usual strain energy expression; the integration being carried out numerically. The element was divided into convenient triangles for the integration process and 72 gaussian points were used overall.

Utilizing equations (3.36) to (3.39), we can illustrate the Fawkes method. Substituting equation (3.39) into (3.36) and re-arranging we have,

$$\{\lambda\} = ([A]^t)^{-1}[K_c][A]^{-1}\{q_1\} \tag{3.40}$$

Subsequent substitution into equation (3.37) reveals that a simple additional term is imposed on the submatrix $[K_{11}]$, thus,

$$\begin{aligned} (([A]^t)^{-1}[K_c][A]^{-1}+[K_{11}])\{q_1\}+[K_{12}]\{q_2\}&=\{Q_1\} \\ [K_{21}]\{q_1\}+[K_{22}]\{q_2\}&=\{Q_2\} \end{aligned} \tag{3.41}$$

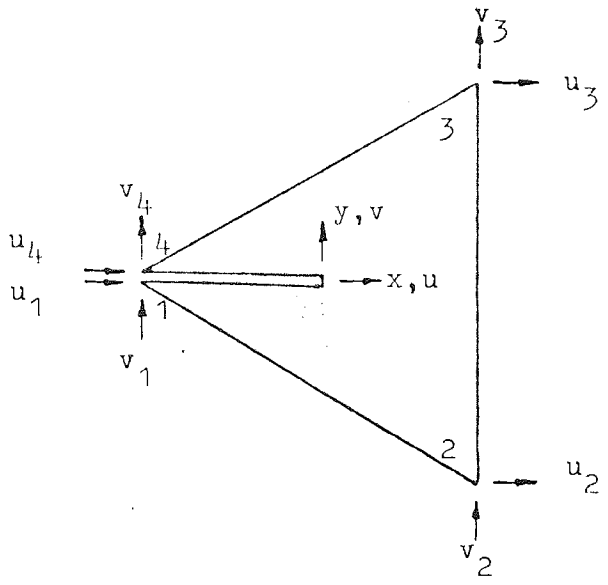


FIG 3.8. Byskov singularity element.

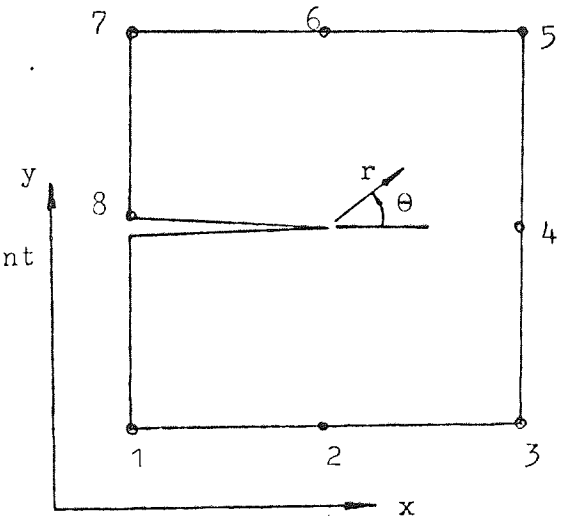


FIG 3.9. Fawkes 8-node crack tip element

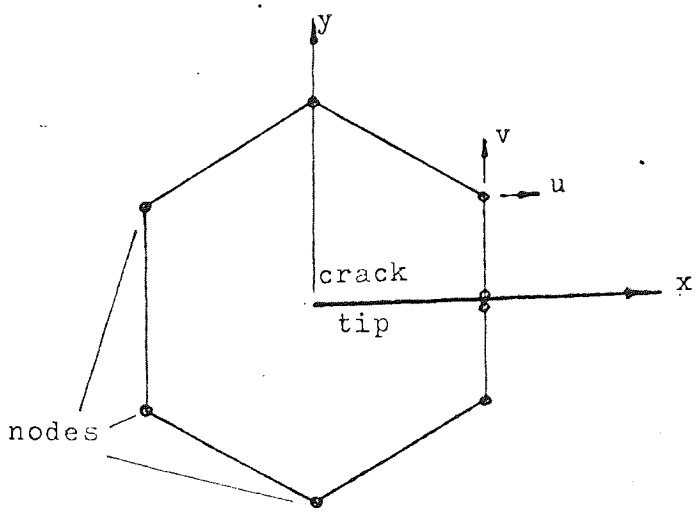


FIG 3.10. Jones and Callinan crack tip element. The number of boundary nodes can be adjusted.

Solving this set of equations yields the nodal displacements and on substituting the vector $\{q_1\}$ into equation (3.39) the stress intensity factors contained in the vector $\{\alpha\}$ can be computed.

A similar approach is used by Jones and Callinan (50), it differs in that the number of terms taken in the series expression and also the number of nodes on the circular element boundary, is optional. The element is given in figure (3.10). Because of the facility enabling any number of terms to be taken in the series expression, matrix [A] will now be non-square. Hence a pseudo inverse or least squares method is employed in the solution for vector $\{\alpha\}$. Thus, by premultiplying equation (3.36) by $[A]^t$, it can be seen that the resulting multiple $([A]^t[A])$ gives a square matrix.

$$[A]^t[A]\{\alpha\} = [A]^t\{q_1\}$$

or
$$\{\alpha\} = ([A]^t[A])^{-1}[A]^t\{q_1\} \tag{3.42}$$

Following the same algebraic manipulations used in the previous method, it will be found that the Lagrange multiplier becomes,

$$\{\hat{\lambda}\} = (([A]^t[A])^{-1}[A]^t)^t[K_c]([A]^t[A])^{-1}[A]^t\{q_1\} \tag{3.43}$$

and substituting into equation (3.37) leads to an expression similar to that of equation (3.41). The solution of the stiffness equation yields the vector $\{q_1\}$ and the stress intensity factors can be found on substituting this vector into equation (3.42).

The second scheme determines the stress intensity factors directly and this is achieved by replacing the displacement vector $\{q_1\}$, by the

singularity terms $\{\alpha\}$. The method, due to Hilton and Hutchinson (51), was originally applied by Wilson (52) and again the element formulation is based on the Williams stress function. As with the previous element, any number of boundary or interface nodes may be chosen, and the shape of the element is chosen to be circular. Utilizing the four basic equations (3.36) to (3.39), the scheme proceeds as follows; premultiplying (3.37) by $[A]^t$ and combining with equation (3.36) gives,

$$[A]^t[K_{11}]\{q_1\} + [A]^t[K_{12}]\{q_2\} + [K_c]\{\alpha\} - [A]^t\{Q_1\} = 0$$
$$[K_{21}]\{q_1\} + [K_{22}]\{q_2\} - \{Q_2\} = 0 \qquad (3.44)$$

replacing the vector $\{q_1\}$ from equation (3.39) results in the final expression,

$$([K_c] + [A]^t[K_{11}][A])\{\alpha\} + [A]^t[K_{12}]\{q_2\} - [A]^t\{Q_1\} = 0$$
$$[K_{21}][A]\{\alpha\} + [K_{22}]\{q_2\} - \{Q_2\} = 0 \qquad (3.45)$$

Solving this set of equations yields the stress intensity factors and also the nodal displacements.

Both of the methods described above have certain salient features, and the complexity of installing either scheme in a conventional finite element program is dependent on the type of stiffness matrix handling procedures employed. These techniques are examined in more detail in Chapters Six and Seven.

ii. POLYNOMIAL BASED DISPLACEMENT FUNCTIONS

Using elementary polynomial displacement fields, rather than the more complex classical function, allows simpler elements to be constructed. Tracey (53) introduced a triangular element, which

contained the $1/\sqrt{r}$ singularity. Several of these elements were used around the crack tip as shown in figure (3.11) and solving for the stress intensity factors required some secondary procedure to interpret the displacement results. The displacement field for this element can be described by,

$$\Delta = (1 - \xi^p)\Delta_1 + \xi^p(1 - \eta)\Delta_2 + \xi^p\eta\Delta_3 \qquad (3.46)$$

The coordinates ξ and η are defined by figure (3.11) and the displacement field represents a $1/r^p$ singularity, where r is the radial coordinate with origin at point 1, the crack tip.

A triangular element formulation similar to that above was proposed by Blackburn (54). The coordinate system is the same as that used by Tracey, the element has six nodes and the displacement field is given by,

$$\Delta = b_1 + b_2\xi + b_3\eta + \frac{b_4\xi + b_5\eta + b_6\xi\eta}{\sqrt{\xi+\eta}} \qquad (3.47)$$

where ξ and η are as defined in figure (3.11). Again, in all of these element representations, the stress intensity factors are not obtained directly and must be calculated using one of the schemes presented in Section I.

iii. ISOPARAMETRIC REPRESENTATIONS

The distorted isoparametric element technique is extremely interesting from the practical point of view, because of the ease with which it can be incorporated within a standard finite element

program. From the early investigations, involving loss of accuracy in distorted elements, it was recognised that the transformation from cartesian to isoparametric coordinates can be singular if the nodal points along the sides of the elements are positioned in a certain way. Henshell (55) and Barsoum (56) first realized that this element behaviour could be used in fracture mechanics work, and indeed seems to be a natural extension of the polynomial element formulation.

During the initial trials with the isoparametric element both the triangular and quadrilateral forms were used, and in the two dimensional case, the mid-side nodes were positioned one-quarter of the way along the edges, nearest the crack tip, see figure (3.12). It was subsequently found however that the singularity conditions prevail only along the edges of the element, and not on an arbitrary ray emanating from the corner singularity point. This condition was resolved by Barsoum (57), who found by collapsing one side of a quadrilateral element to form a triangle and adjusting the mid-side nodes appropriately the $1/\sqrt{r}$ singularity prevails on all rays emanating from the crack tip. See figure (3.13). It appears that the doubly distorted elements only give acceptable results if the three edges are straight. The solution accuracy can be improved by surrounding the distorted crack tip elements by a further ring of elements which also include the effect of the singularity at the crack tip. This was used by Lynn and Ingraffea (58), who show that the inclusion of such 'transition' elements, significantly improves the accuracy without additional degrees of freedom.

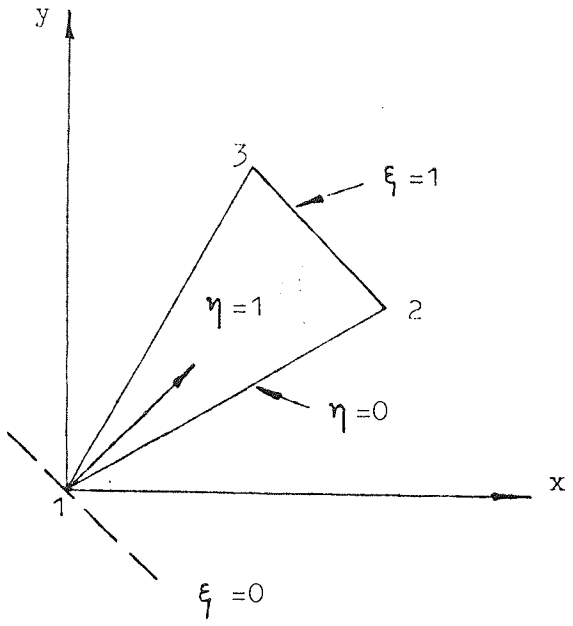


FIG 3.11. Tracey triangular element, singularity at point 1.

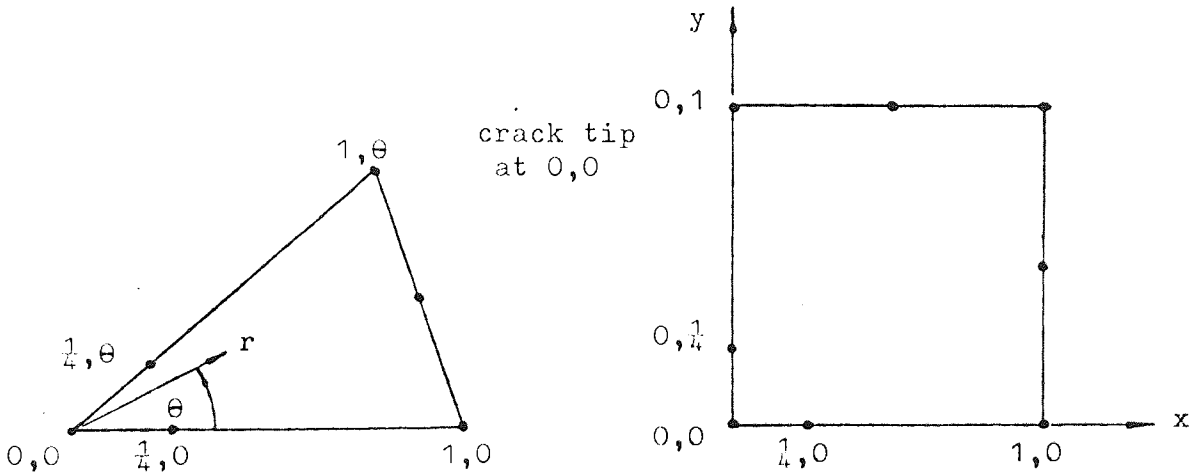


FIG 3.12. 'Mid -side' node distortion of Natural Isoparametric Triangular and Quadrilateral Elements.

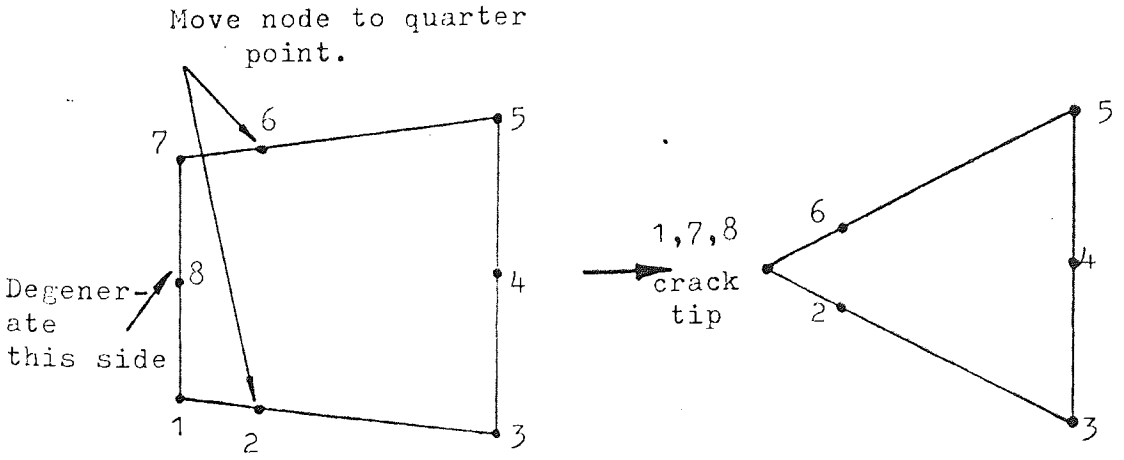


FIG 3.13. Double distorted Isoparametric Quadrilateral Element.

Again, as with the polynomial based elements, the stress intensity factors are not obtained directly and must be determined using one of the methods described in Section I. The element stiffness matrix can be calculated in the usual manner, by numerical integration; the sampling points are not located at the singularity point and therefore present no difficulty. There is only a slight difference between the displacement function corresponding to the degenerate isoparametric element and the special shape function given by Blackburn (54). Thus, the displacement function for the distorted element is given by,

$$\Delta = a_1 + a_2\xi + a_3\eta + \frac{a_4\xi + a_5\eta}{\sqrt{\xi+\eta}} + \frac{\xi\eta}{\xi+\eta}\, a_6 \qquad (3.48)$$

whereas in Blackburns formulation, the displacements are given by equation (3.47); it can be seen that the only difference is in the last term. The performance of these two elements has been examined by Hellen (59), and there is no significant difference in the accuracy of the two formulations.

iv. HYBRID FORMULATIONS

Hybrid formulations involve the choice of two or more assumed fields of behaviour in a single element. There are three basic hybrid models of which the stress formulation seems to be the simplest to comprehend. Pian, Tong and Luk (60) introduced this hybrid formulation and used the classical stress field functions to describe the stress state within the element, together with a simple conforming boundary displacement field employed on the edges of the element.

We can describe the combined stress field as

$$\{\sigma\} = [P_o]\{\beta_o\} + [P_s]\{\beta_s\} \tag{3.49}$$

where $[P_o]$ represents the coefficients of a simple expansion in the element, $[P_s]$ contains the coefficients from the singularity stress field, and $\{\beta_s\}$ contains the stress intensity factors. For the two-dimensional analysis,

$$\{\beta_s\} = \begin{bmatrix} K_I \\ \\ K_{II} \end{bmatrix} \tag{3.50}$$

Similarly the surface tractions can be described by,

$$T = [R_o]\{\beta_o\} + [R_s]\{\beta_s\} \tag{3.51}$$

and the edge displacements are,

$$u = [Y]\{\Delta\} \tag{3.52}$$

Using the above expressions, we can show that the complementary energy is given by,

$$\begin{aligned} \pi_c = & \frac{1}{2}\{\beta_o\}^t[H_o]\{\beta_o\} + \{\beta_o\}^t[H_{os}]\{\beta_s\} \\ & + \frac{1}{2}\{\beta_s\}^t[H_s]\{\beta_s\} - \{\beta_o\}^t[Q_o]\{\Delta\} \\ & - \{\beta_s\}^t[Q_s]\{\Delta\} \end{aligned} \tag{3.53}$$

where $[H_o] = \int [P_o]^t[E]^{-1}[P_o]dvol$, and similarly for the remaining terms $[H_s]$, $[Q_o]$ and $[Q_s]$. On taking the variation of π_c with respect to $\{\beta_s\}$ and $\{\Delta\}$, we obtain equations of the form

$$\begin{bmatrix} K & M^t \\ M & N \end{bmatrix} \begin{Bmatrix} \Delta \\ \beta_s \end{Bmatrix} = \begin{Bmatrix} F \\ 0 \end{Bmatrix} \tag{3.54}$$

Solving these equations yields the stress intensity factors directly.

Other hybrid methods are combinations of assumed displacement fields and the assumed stress field.

3.3.4 SUMMARY

The various numerical techniques used to determine the stress intensity factors will be summarized here so that the method adopted in this thesis can be put into perspective. As a means of comparing the efficiency of the various approaches, the complexity of the mesh, and the accuracy of the solutions, a standard problem will be used as a reference. This standard problem is that of a rectangular plate, containing a 90 degree edge crack, in uniaxial tension. From symmetry only half of the plate need be considered, see figure (3.14).

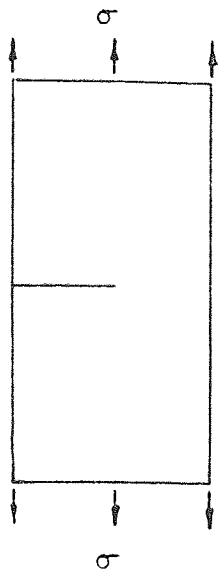
The interpretation of the finite element nodal displacements or stresses using the classical solutions, was found to be inefficient due to the large number of conventional elements required to give an accurate result, say within 5%. The development of the polynomial and degenerate isoparametric singularity elements has, however, revived the method and investigators are experimenting to determine

the most effective crack tip mesh configuration. Using the isoparametric degenerate element, Barsoum, having only 10 elements in the basic edge crack problem, obtained results within ± 10% of the generally accepted theoretical solution. Also Lynn and Ingraffea using a more refined approach, that is with transition elements, obtained results within ± 4% for the same mesh, that is, 10 elements and 90 d.o.f. The elements used are eight-node quadrilateral isoparametric elements and Barsoum recommends a 3 x 3 or 9 point Gaussian integration over the element.

Murakami's approach, which is another technique for interpreting the nodal stresses, investigates the single edge crack problem and shows graphically that good results are obtained in comparison with Bowies solution. A typical fine mesh idealization contains 295 elements and 175 nodes, whereas the coarse mesh has 165 elements and 101 nodes. The elements used are simple constant strain triangles and consequently some numerical advantage is gained in that the element stiffness matrix is explicitly defined.

The hybrid method, employed by Pian and others, has been used to solve the edge crack problem and values for K_I within 5%, using 35 nodes in the mesh, have been obtained. By increasing the element density resulting in 92 nodes, the error was reduced to 0.5%.

Finally we have the classically based formulations. The first interpretation discussed earlier was that used by Fawkes. In the application of this method very few elements are employed and approximately 29 nodes or 6-7 elements would be used in the comparison problem.



Problem used to compare the different techniques employed in solving for K_I.
Plate in tension with a single edge crack.

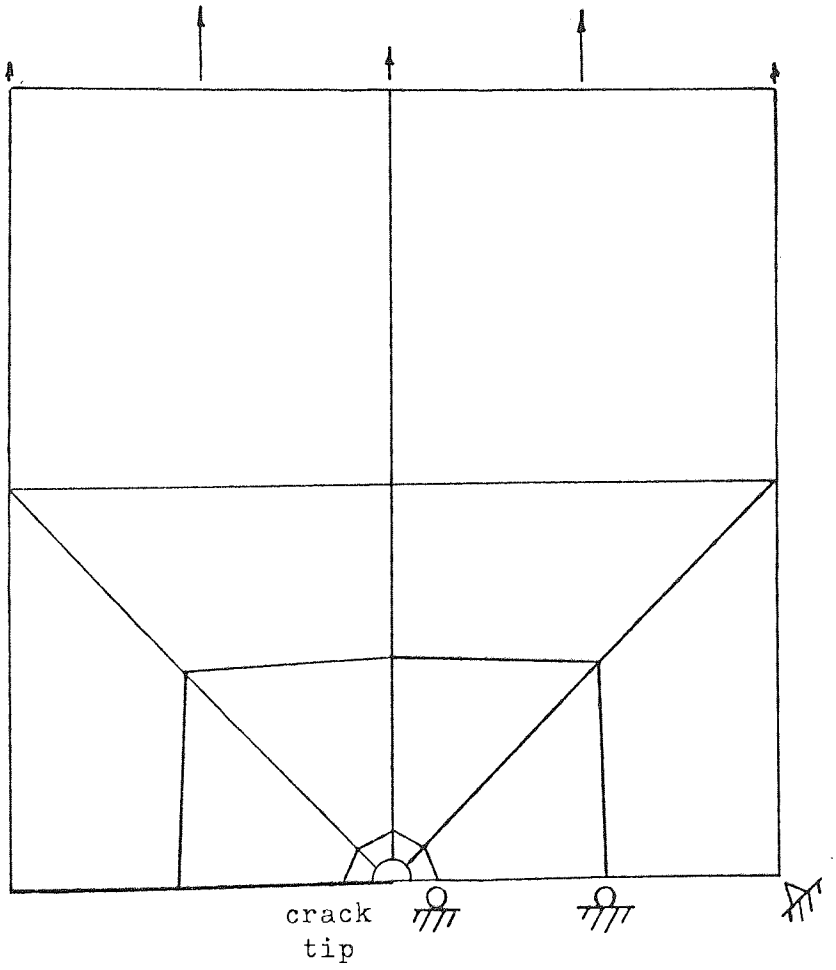


FIG 3.14. Typical mesh used in the finite element method.

Fawkes states that the technique yields good results and with a minimum of user errors.

The singularity element formulated by Jones and Callinan, is very versatile allowing the number of nodes on the element boundary and the number of terms in the singularity expression to be variable. From investigations carried out to determine the effect of local mesh density and size of the special element, it was found that a relatively large singularity element produced the optimum solution accuracy. Used in conjunction with 4-node isoparametric quadrilateral elements the basic edge crack problem contained 88 nodes and the solution was within 3.0% of the accepted value.

The last of the classical formulation methods to be described, was that used by Hilton and Hutchinson. This technique was employed by former research workers, Robertson and Alsharqi, and the programs generated form the basis for the work presented in this thesis. The method has been improved and for the case of the single edge crack problem a total of 14 elements and 59 nodes were used to obtain a result within 2.0% of the accepted solution by Bowie. The techniques used are given in Chapter 6.

Table (3.15) brings together the various methods and lists the mesh specifications and solution accuracy for the simple 90 degree edge crack problem. From these results it is worth noting that the Hybrid method and the Fawkes formulation, use only a small number of elements in their model. In both of these cases the singularity element is relatively large and hence the back-up mesh is sparse.

Method	No: of Elements	No: of Nodes	D.O.F	Accuracy	Element type in back-up mesh
Degenerate Isoparametric Quadrilateral	10	45	90	± 4%	8-node Quad'l
Murakami method	coarse 165 fine 295	101 175	202 350	* good	3-node Trig'l
Hybrid element	coarse fine	35 92	70 184	5% 0.5%	8-node Quad'l
Fawkes formulation	6-7	29	58	* good	8-node Quad'l
Jones & Callinan element	-	88	176	3%	4-node Quad'l
Present technique used herein	14	59	118	2%	8-node Quad'l

TABLE 3.15. Comparing the mesh specification and solution accuracy of six numerical techniques used in solving a simple edge crack problem.
Accuracy according to authors. * within 5%

In this simple symmetric test problem a sparse mesh seems to be sufficient to yield good results, i.e. within 5%, but in more complex practical problems the back-up mesh density should be the same in all methods used, as appropriate to the element formulation. In other words, in order to model the stress field accurately a similar mesh grid must be applied in each case. Therefore, in terms of mesh complexity, there is very little difference in the techniques. Also in most practical engineering situations a solution accuracy within 5% is acceptable, and this is met by all of the methods examined.

The factor which segregates the methods, is the degree of complexity involved in implanting the technique into an existing finite element program. Perhaps the simplest of these methods is that of Murakami, which interprets the stress at the crack tip, requiring no modification in the program structure. This approach does, however, have a disadvantage in that two runs are required for the complete solution. The degenerate isoparametric formulation is possibly the most readily acceptable technique in this respect and can easily be adapted to most general finite element programs, with the addition of a displacement interpretation routine. The remaining singularity formulations, all affect the stiffness matrix to some degree, as previously explained.

The Hilton and Hutchinson method as used in this thesis was formulated by Richards and Robertson (61).

The technique and the further developments carried out by the present author, are further discussed in Chapters Six and Seven.

CHAPTER 4

REFINEMENT OF EXISTING PROGRAMS

4.1 INTRODUCTION

Two former research workers namely Robertson[28] and Alsharqi[29], investigated two dimensional and axisymmetric crack behaviour using the finite element method. The structure of both basic programs are similar with several procedures being common. Both programs utilize the six-node isoparametric triangular element with appropriate modifications in each case. As would be expected the mesh configurations are identical and consequently the same mesh generator can be employed in either program. Due to time restrictions a general mesh generation program could not be developed and both investigators resorted to ways of reducing data input e.g. interpolation, etc. A mesh generation scheme has now been developed as part of the present investigation and the input procedures of both programs have subsequently been modified to accept the new data format. See Chapter Five. With the increased use of degenerate isoparametric quadrilateral elements in fracture work, it was felt that the integration of this element type within the two-dimensional scheme was desirable. It would also establish the method for introducing higher order elements in later modifications. An outline of this program's operation and subsequent alteration is given in the following section.

With the development of the fracture mechanics programs and the introduction of a labour saving mesh generation scheme, larger problems, involving greater element detail, could be undertaken. The freedom provided by these developments, however, prompted the need for a versatile solving routine. Consequently a new segmented solving routine capable of encompassing the problems which were envisaged has been implemented. The routine logic is based on a report by Jennings and Tuft (69), and has been successfully used in the two-dimensional finite element program.

A file-index will be found in the Appendix (9.3) referring to all the programs, with a brief description of their function.

4.2 2-D FINITE ELEMENT PROGRAM DESCRIPTION

The two-dimensional, plane stress, plane strain finite element program, written by a previous research worker, A.Robertson[28] utilised the isoparametric six-node triangular element. The program is conveniently structured into sub-procedures which will be briefly discussed here to set the scene for the present authors work.

The control data, nodal coordinates, element nodal connections and boundary conditions, are all read by procedure INPUT. The program uses the solving routine SYMVBSOL, as initially described by Jennings and Tuft[70], and as a result the overall stiffness matrix is economically stored in a one dimensional array. The coefficients of the one-dimensional array are referenced to the two-dimensional stiffness matrix via an address sequence. This address sequence is compiled from

the input data by, ADDARRAY, which is the next procedure to be accessed. The address sequence is a one-dimensional array of length equal to the structures degree of freedom. Each coefficient in the sequence is equivalent to the sum of all the previous coefficients in the array plus the number of terms between the first non-zero term and the leading diagonal of the stiffness matrix.

The elasticity matrix [C] is evaluated in procedure CMATRIX and relates the stresses to the strains. Plane stress or plane strain can be selected using the appropriate control variable.

The external loading is transferred to the load vector {Q} in the procedure LOADAPP.

The overall stiffness matrix is assembled within the procedure FEASSEMBLY. Each element is called in turn and using numerical integration the element stiffness matrix is obtained. Referring back to section (2.4.1), in order to perform the necessary integration the [B] and Jacobian matrices must be compiled, this is achieved in sub-procedure AUX. The element stiffness matrix coefficients are positioned in the one-dimensional stiffness matrix using the address sequence found earlier.

Where the structure has been given prescribed nodal displacements, such as anchor points, etc., the procedure GEOMBC zero's the rows and columns of the stiffness matrix, the leading diagonal is made unity and the appropriate displacement is allocated to the load vector. Again, as the stiffness matrix is held in a one-dimensional array, the appropriate stiffness coefficients are found via the address sequence.

Having compiled the stiffness matrix and load vector the solving routine is called; this was written by Jennings and Tuft, and uses Cholesky's decomposition. The load vector is overwritten by the newly determined displacements and the stiffness array is also overwritten by the upper triangular coefficients.

With the nodal displacementsknown at this stage, the nodal strains and stresses are determined by re-invoking the AUX routine which formulates the [B] matrix of equation (2.13), hence the strains can be found and consequently the stresses via the [C] matrix. The nodal stresses are compiled by considering the average contributions from the surrounding elements. The procedure FENOSTR determines the stresses and strains at the nodes and a separate routine, FEELSTR, determines the stresses and strains at the element centroids; both are optional and can be called using the appropriate control variable.

The sequence in which these procedures are called is summarized here for clarity.

Procedure name	Function
INPUT	Reads the control data, nodal coordinates and element nodal connections.
ADDARRAY	Forms address sequence for the 1-D stiffness array.
CMATRIX	Forms matrix [C] which relates the stresses to the strains.
LOADAPP	Transfers external loads to the vector {Q}.
FEASSEMBLY	Forms the element stiffness matrix and positions it within the global 1-D stiffness matrix.
AUX*	

GEOMBC	Adjusts the matrix [K] and vector {Q} in allocating the prescribed displacements.
SYMVBSOL	Solving routine.
FENOSTR	Calculates and prints the nodal stresses and strains.
AUX*	
FEELSTR	Calculates and prints the centroidal stresses and strains.
AUX*	
*AUX	Sub-routine used to determine detJ and matrix [B] which relates the strains to the nodal displacements.

4.3 MODIFICATIONS IN ADOPTING THE ISOPARAMETRIC QUADRILATERAL ELEMENT

Having looked briefly at the procedures used in formulating the displacements and subsequent stresses and strains for the six-node triangular element, we can examine each routine to determine the alterations necessary in introducing, as an option, a second element; the eight-node quadrilateral.

Firstly, to distinguish the two element types a control variable QORT has been adopted, its value being 0 - for the triangular element and 1 - for the quadrilateral element. Procedures which are entirely unaffected by the modification are:-

CMATRIX,

LOADAPP,

GEOMBC,

SYMVBSOL,

SKEWLOAD,

Procedures which can be adjusted fairly easily, by altering control loops and array sizes, etc. are:

ADDARRAY

INPUT

SKEWEDCON

These procedures can accept both element types by the simple use of the QORT parameter, i.e. (6 + 2*QORT).

It would have been possible to modify the procedure FEASSEMBLY to accept both element types, but it was felt that it would be more satisfactory to establish a new routine QFEASSEMBLY. The basic alterations were to increase the size of the element stiffness matrix, change the values used in the four point numerical integration, and introduce the new [B] matrix as in equation (2.15). A similar situation arose with procedures FENOSTR and FEELSTR, but by using the QORT variable the two elements could be accommodated within the same routine. The [B] matrix is again used in the forementioned routines and it was necessary to write a new procedure QAUX to mirror the existing AUX routine.

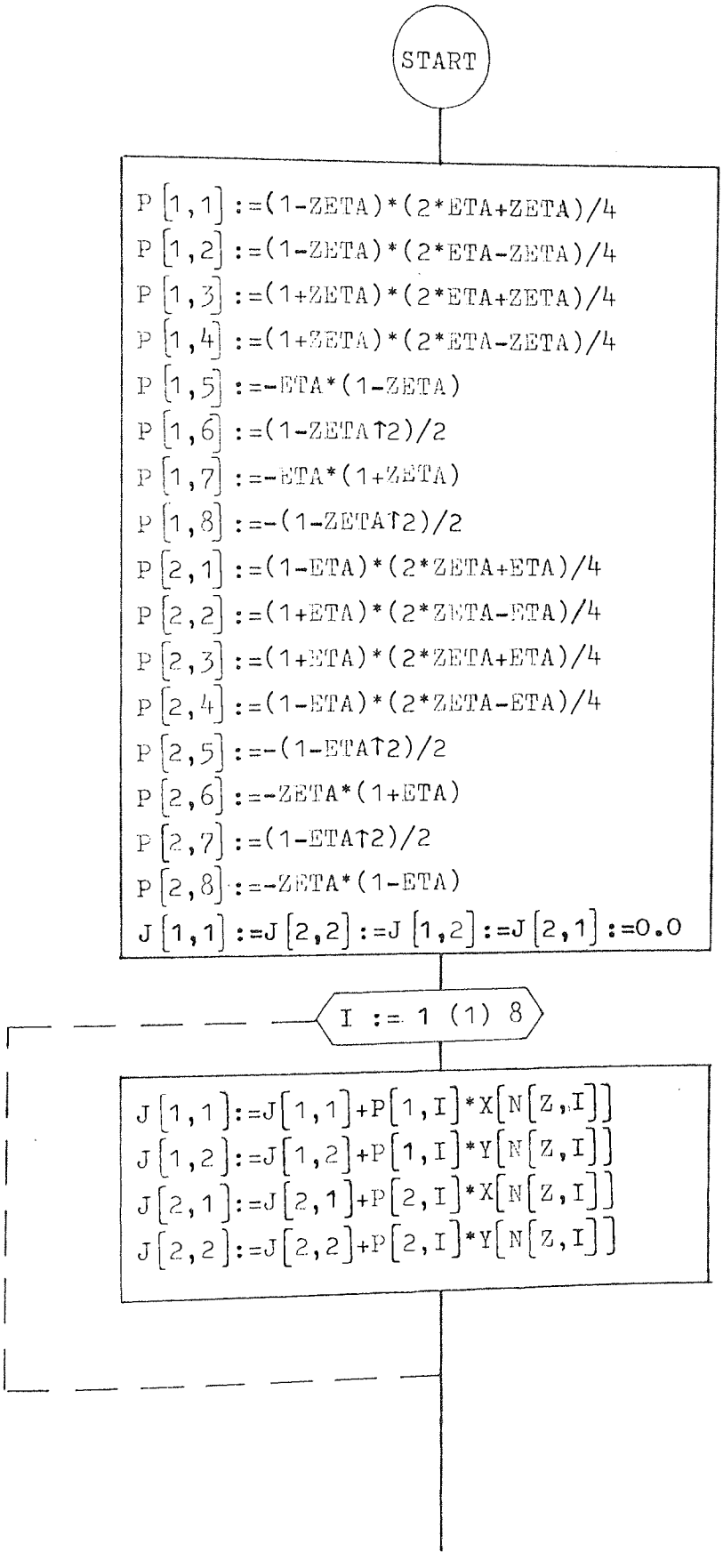
A flowchart for routines QFEASSEMBLY, FENOSTR and FEELSTR, was felt to be unnecessary for this thesis but a formal listing is given in the Appendix. The QAUX procedure, however, merits a detailed description.

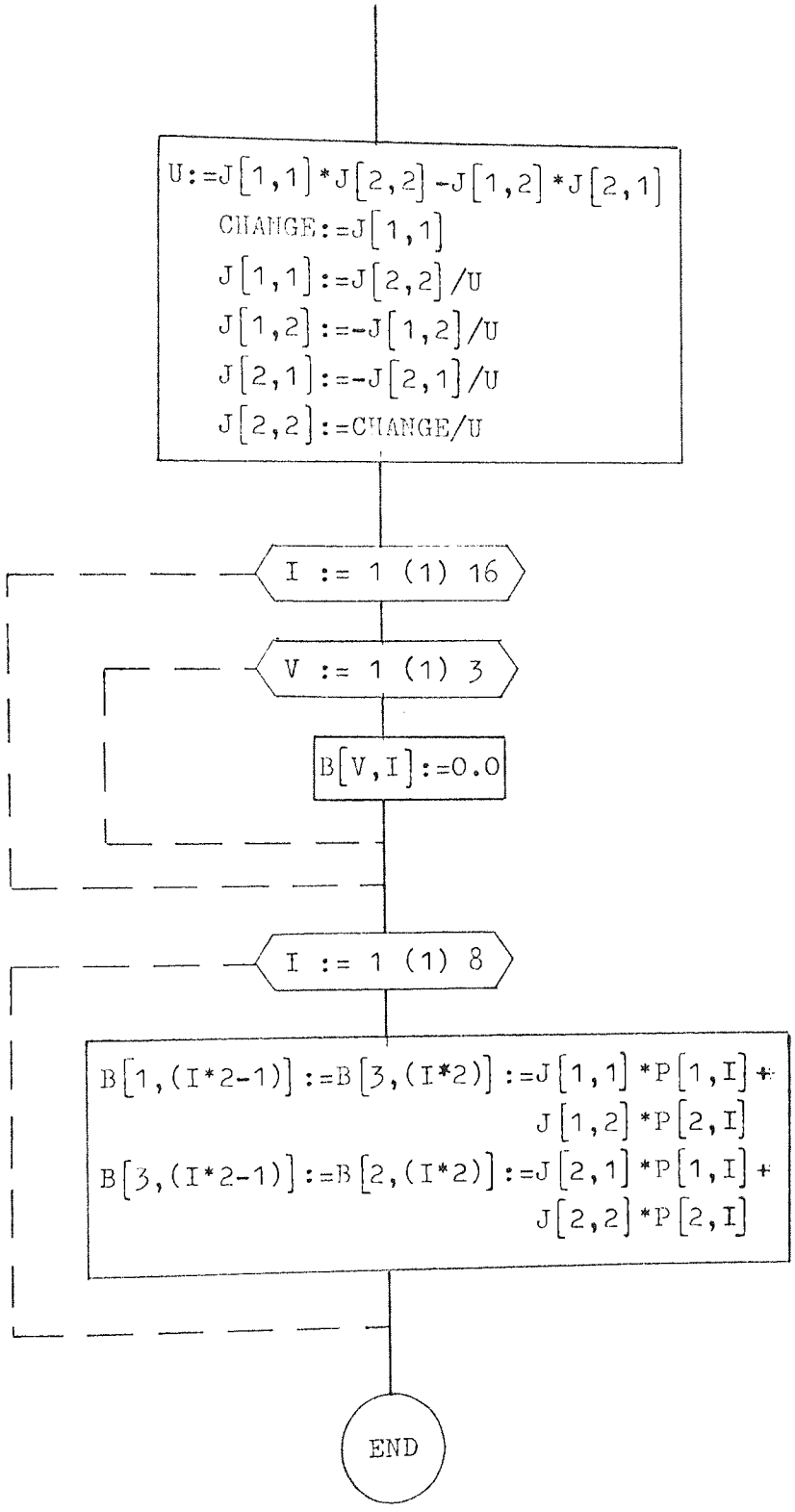
4.3.1 PROCEDURE QAUX

The evaluation of the [B] matrix and the Jacobian matrix is explained in section (2.4.1). The QAUX procedure is explained in the following steps which refer to the flowchart overleaf.

1. The matrix [P] is formulated and holds the differentiated shape function coefficients $[\partial N/\partial \xi]_i$ and $[\partial N/\partial \eta]_i$. The local coordinates are assigned a value on entering the routine, and these refer to the integration points or locations in evaluating the stresses.
2. The Jacobian matrix is initialised.
3. From equation (2.17) Chapter (2), the Jacobian matrix can be formed, using the differentiated coefficients of matrix [P] and the nodal coordinates.
4. The 4 x 4 Jacobian matrix can easily be inverted and its determinant is held in variable U.
5. Matrix [B] is initialised.
6. Equation (2.18) of Chapter (2) shows how the derivatives $\partial N/\partial x$ and $\partial N/\partial y$ can be found, hence the [B] matrix is formulated. This is achieved within the loop counter I, using the inverted Jacobian matrix and the [P] matrix.

Flowchart for Procedure Qaux:-




U:=J[1,1]*J[2,2]-J[1,2]*J[2,1]
CHANGE:=J[1,1]
J[1,1]:=J[2,2]/U
J[1,2]:=-J[1,2]/U
J[2,1]:=-J[2,1]/U
J[2,2]:=CHANGE/U
4.
I := 1 (1) 16
V := 1 (1) 3
B[V,I]:=0.0
5.
I := 1 (1) 8
B[1,(I*2-1)]:=B[3,(I*2)]:=J[1,1]*P[1,I]+
J[1,2]*P[2,I]
B[3,(I*2-1)]:=B[2,(I*2)]:=J[2,1]*P[1,I]+
J[2,2]*P[2,I]
6.
END

4.3.2 EXAMPLES

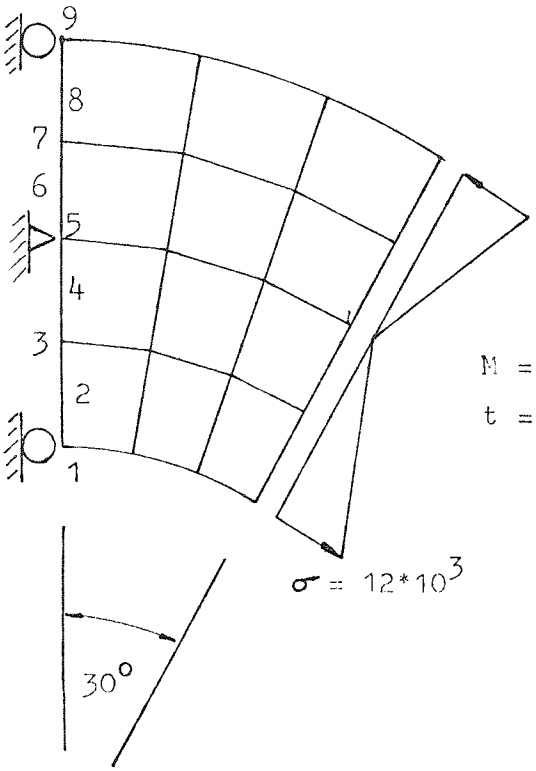
To test the modified program three examples were chosen. The first example is that of a curved bar in pure bending shown in figure (4.1). The results, showing the stress distribution along the vertical axis of the bar, are plotted in figure (4.3), where the theoretically 'exact' curves, taken from reference (71), are also shown for comparison. It can be seen that the hoop stress σ_θ fits the theoretical curve at all points, whereas the radial stress σ_r fits the general shape of the curve but the points are scattered. In theory the stress distribution over the end of the bar is not linear, as shown in figure (4.1), but parabolic in form. The size of the segment in relation to the depth of the bar, would suggest that the proximity of the linear loading may have attributed to the discrepancies found in the finite element results.

Example (2), figure (4.2), illustrates the mesh used for a ring in compression. The results are again plotted to compare with the theoretically 'exact' stress distribution and figure (4.4) shows the hoop stress distribution across the horizontal and vertical faces of the ring.

The finite element results fit the theoretical stress curve extremely well except close to the singularity at the point of application of the concentrated force.

The element has been used in the fracture mechanics work in progress and example (3), figure (4.5) shows the results for a

EXAMPLE .1. CURVED BAR IN PURE BENDING.

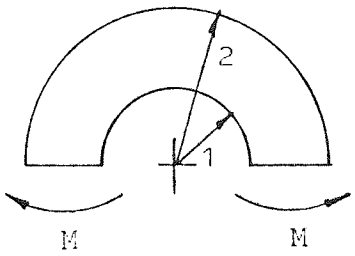


$M = 2*10^3$

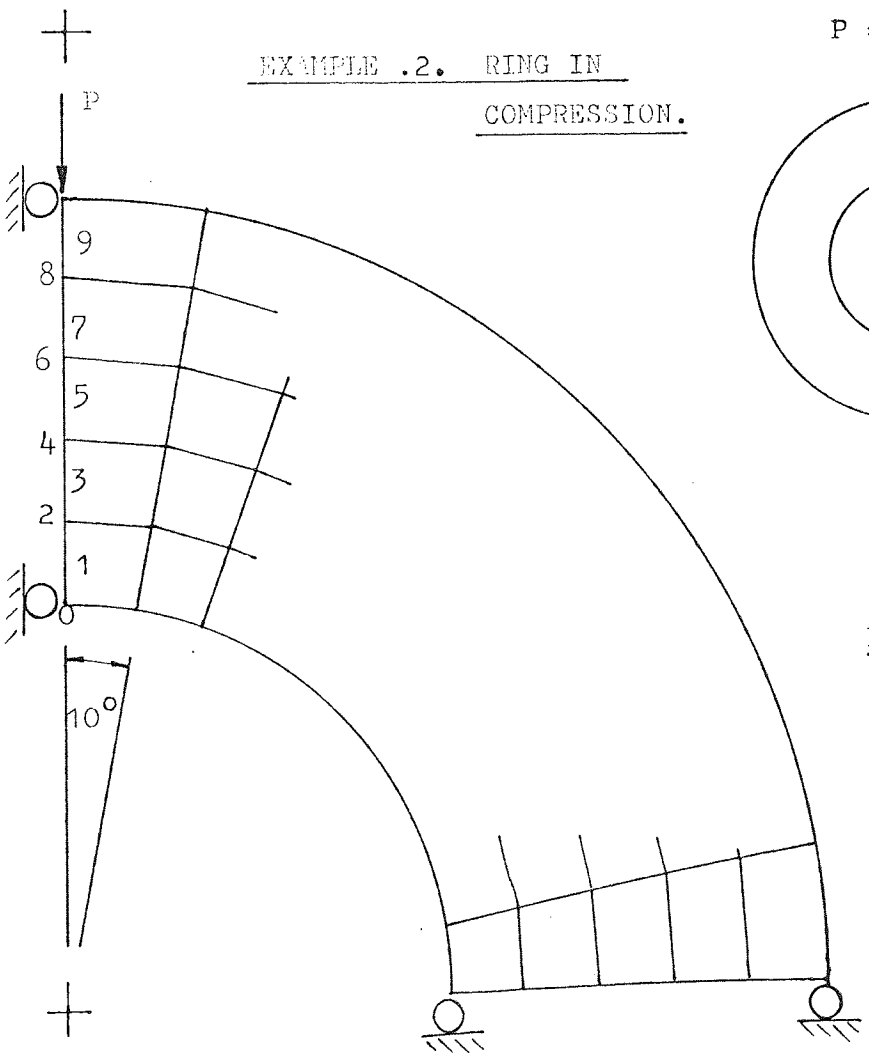
$t = 1.0$

$\sigma = 12*10^3$

FIG 4.1.



EXAMPLE .2. RING IN COMPRESSION.



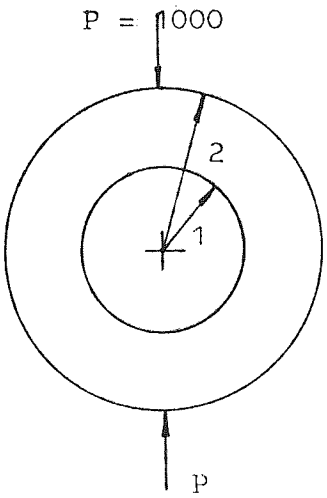


FIG 4.2.

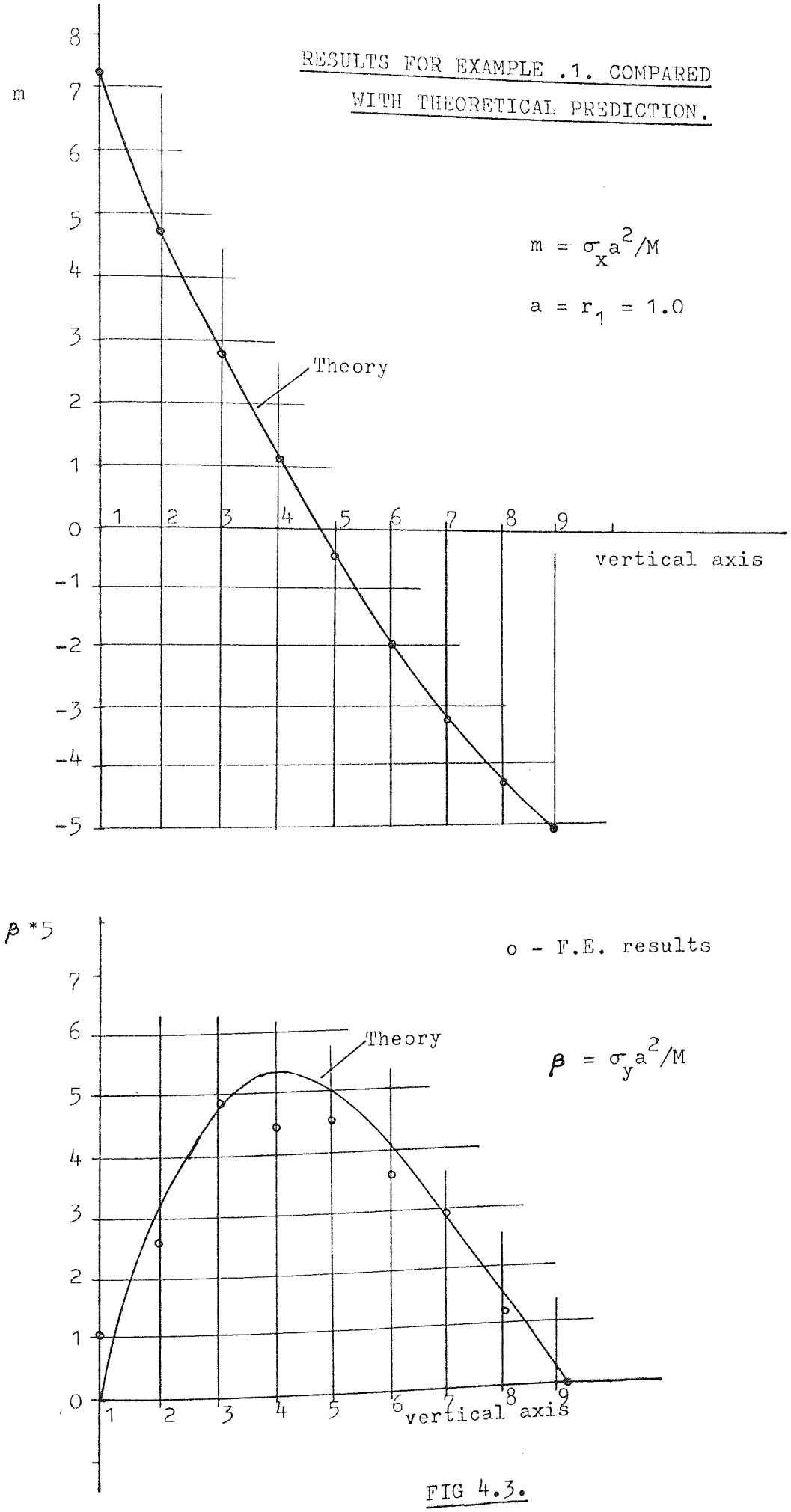
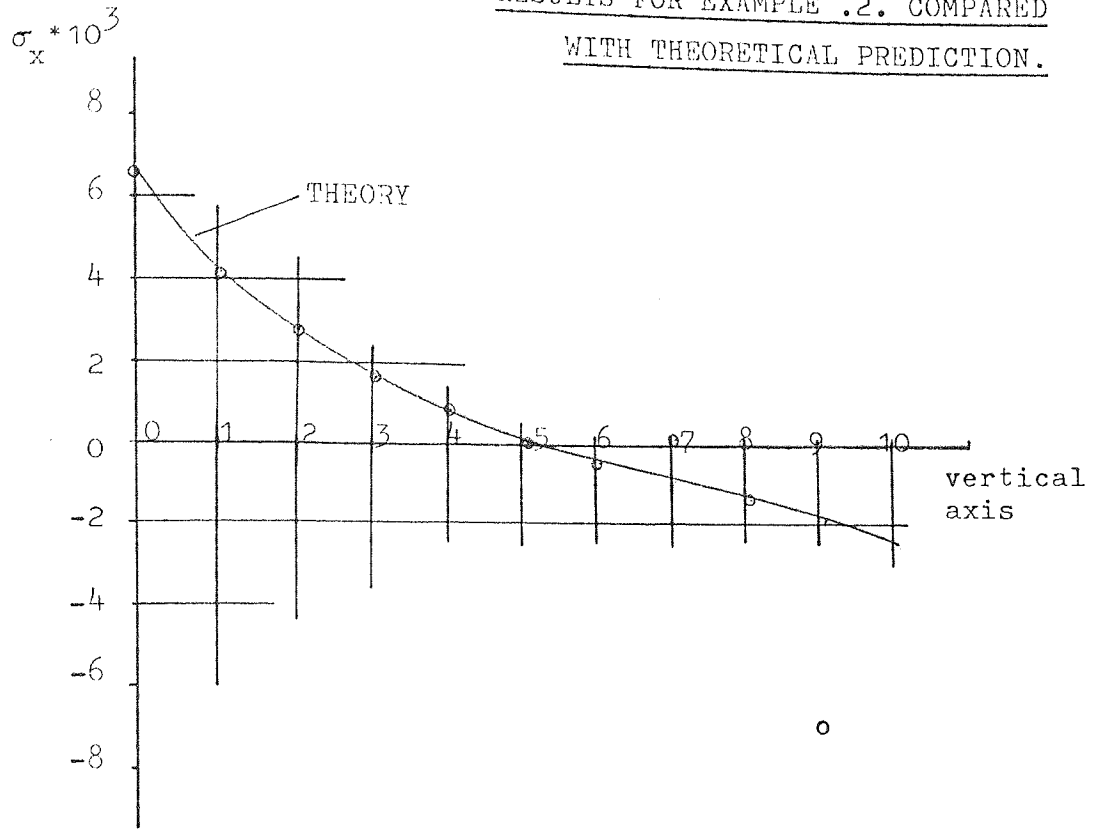

RESULTS FOR EXAMPLE .1. COMPARED WITH THEORETICAL PREDICTION.
m
$m = \sigma_x a^2/M$
$a = r_1 = 1.0$
Theory
vertical axis
$\beta * 5$
o - F.E. results
Theory
$\beta = \sigma_y a^2/M$
vertical axis

FIG 4.3.

RESULTS FOR EXAMPLE .2. COMPARED WITH THEORETICAL PREDICTION.



o

o ——— Point of singularity

o --- F.E. results.

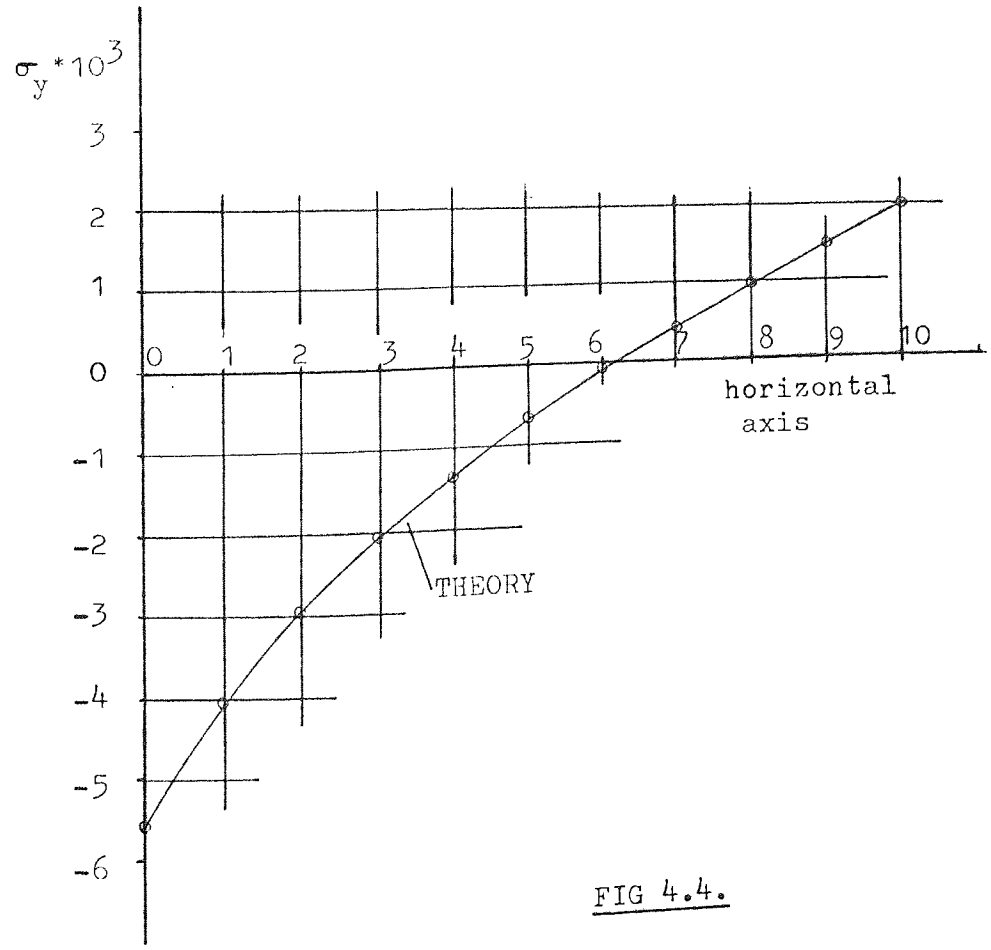
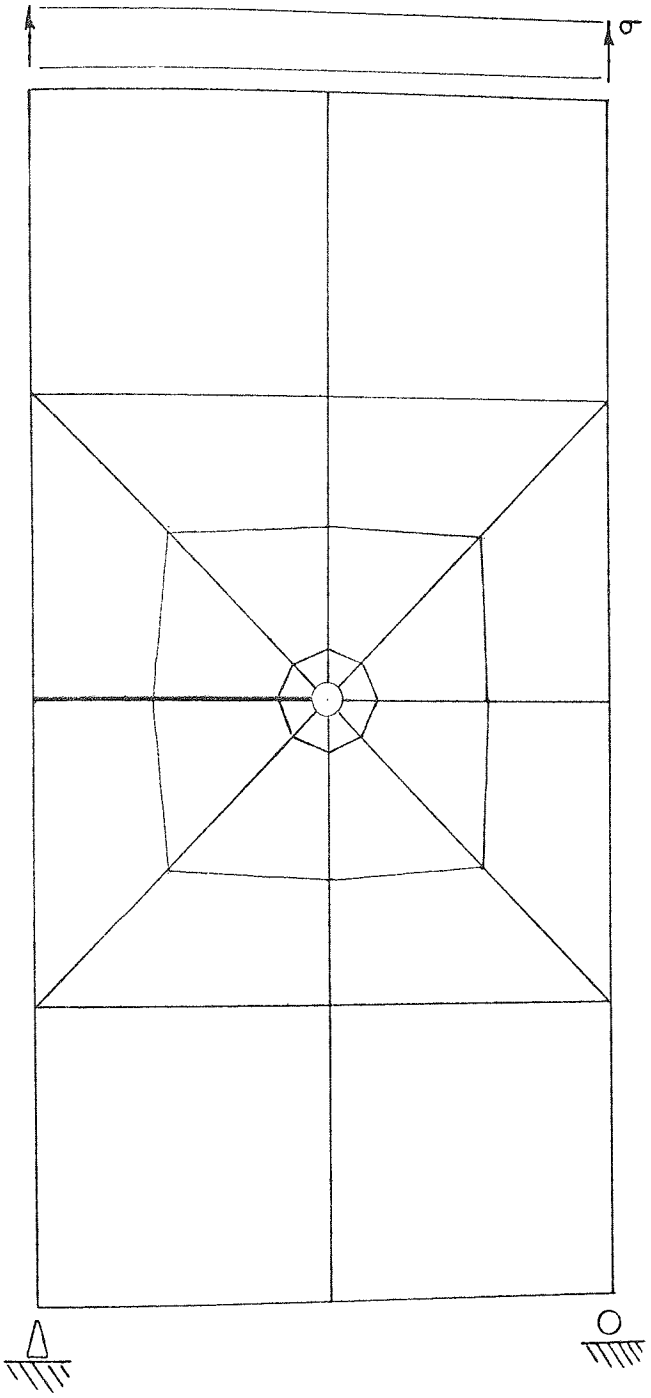


FIG 4.4.



W = 5.0 in

L = 10 in

Rc = 0.083 in - core radius

a = 2.5 in - crack length

Mesh details

No. of interface nodes - 17

No. of elements - 28

No. of nodes - 111

Result F.E.

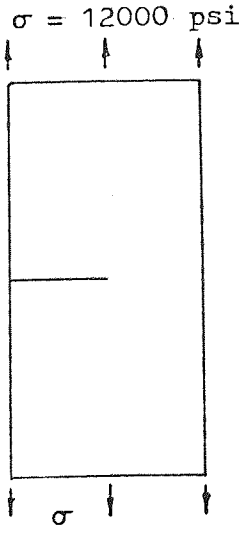
$K_I/\sigma\sqrt{a} = 2.8613$

$K_{II}/\sigma\sqrt{a} = 2.71 \times 10^{-9}$

Theory ref(16)

$K_I/\sigma\sqrt{a} = 2.86$

$K_{II}/\sigma\sqrt{a} = 0.0$



EXAMPLE .3. PLATE WITH A 90 DEGREE EDGE CRACK, IN TENSION.

FIG 4.5.

rectangular plate with a 90 degree edge crack in tension. The results are compared with other predictions from reference (16), and the discrepancy in the mode I stress intensity was only of the order of 2%.

It has thus been shown that higher order elements could easily be accommodated in the basic program framework by adding a simple shape function procedure, QAUX, together with slight modifications to the existing procedures.

4.4 A SEGMENTED SOLVING ROUTINE FOR LARGE SETS OF SPARSE SYMMETRIC SIMULTANEOUS EQUATIONS

4.4.1 INTRODUCTION

During the development of the fracture mechanics program, it became necessary, due to the increasing size of the problems encountered and also due to the limited facilities at Aston, to transfer our work to the much larger computing facilities at Manchester. The initial difficulties of adjusting to the new system were overcome, and consisted basically of print and conditional statement variations and coping with the 1900/7600 operations. Once the programs had been established, it was found that the size of the stiffness matrix, in some cases, could not be held on the computers' large core memory. In an attempt to overcome this difficulty, various 'dodges' and 'tricks' were resorted to, but it became obvious that a new method for storing the stiffness matrix was necessary.

One method of achieving the required result in a short space of time, was to use a binary sequential file and manipulate the stiffness

coefficient to and from the file, as if the matrix was stored in core memory. This was considered to be a very time consuming system and hence a very costly procedure to operate. Another approach, given in reference (69), and on which this work is based, was the method proposed by Jennings and Tuff.[69] The method ideally suited large sets of sparse simultaneous equations, in which the coefficient array was symmetric positive, definite. The local variable bandwidth storage scheme is particularly effective and was used by Robertson[28] in the earlier development of the finite element programs, together with the compact solving routine given in reference (70).

4.4.2 THE LOCAL VARIABLE BANDWIDTH STORAGE SCHEME

During the reduction of a set of sparse equations by any variant of the Gaussian elimination method, the zero elements before the first non-zero element in a row will always remain zero provided that there is no row or column interchange. The Local Variable Bandwidth storage scheme makes use of this fact by storing, for each row of the matrix, only the elements between the first non-zero element and the diagonal. The rows are stored consecutively in a single one-dimensional array and an address sequence is used to locate the position of the diagonal elements within the array. Thus the following set of left-hand side coefficients,

row

$$
\begin{array}{c}
1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6
\end{array}
\begin{bmatrix}
1,5 & & & & & \\
0,2 & 1,2 & & & & \\
-1,1 & 0 & 2,2 & \text{SYMMETRIC} & & \\
0 & 0 & 5.1 & 10,6 & & \\
0 & 0 & 0 & 0 & 2.6 & \\
0 & 0 & -1.2 & 0 & 0 & 6.1
\end{bmatrix}
$$

would be stored in the computer as a main sequence.

Location 1 2 3 4 5 6 7 8 9 10 11 12 13

[1.5, 0.2, 1.2, -1.1, 0, 2.2, 5.1, 10.6, 2.6, -1.2, 0, 0, 6.1]

with a corresponding address sequence

row 1 2 3 4 5 6

[1 3 6 8 9 13]

where the row numbers refer to the rows of the square matrix and the coefficients locate the position of the diagonal elements with the one-dimensional main sequence. Thus row 5 has a location number 9 which corresponds with the main sequence term 2.6, and also with the diagonal element of the square matrix.

If the address sequence is designated S[i], i = 1,2,...,n, then the position of element (i,j) in the main sequence is,

$$K[i,j] = K[S[i] - i + j],$$

for example

$$K[4,3] = K[S[4] - 4 + 3]$$
$$= K[8 - 4 + 3]$$
$$= K[7] = 5.1$$

This means that the formation of the left-hand side coefficients can be carried out without difficulty.

4.4.3 CHOLESKI REDUCTION SEQUENCE

The Choleski triangular factorisation method is well suited to large problems, requiring no storage facilities other than that available for the stiffness matrix or left hand coefficients [K].

Expressing the simultaneous equations in matrix form, we have,

$$[K]\{x\} = \{b\} \tag{4.1}$$

where [K] is a symmetric positive-definite matrix.

Using Choleski factorisation, [K] can be written, thus,

$$[L][L]^t = [K] \tag{4.2}$$

To obtain the displacement vector {x}, by substituting equation (4.2) into (4.1) we have,

$$[L][L]^t\{x\} = \{b\} \tag{4.3}$$

Now letting $[L]^t\{x\}$ = vector {y}, we have two sets of equations which can easily be solved by forward and backward substitution, eliminating the need for complicated inversion of the stiffness matrix.

Hence, $$[L]\{y\} = \{b\} \tag{4.4}$$

and $$[L]^t\{x\} = \{y\} \tag{4.5}$$

It is possible to evaluate vector {y} during the factorisation of array [K]. The equations (4.4) and (4.5) are solved with vector {y} overwriting vector {b} and vector {x} overwriting {y}. In the solving routine the matrix [L] will overwrite matrix [K] by using the recursive relationships,

$$\ell_{ij} = (k_{ij} - \sum_{z=1}^{j-1} \ell_{iz}\ell_{jz})/\ell_{jj}, \text{ for } j < i \tag{4.6}$$

and

$$\ell_{ii} = \sqrt{k_{ii} - \sum_{z=1}^{i-1} (\ell_{iz})^2} \tag{4.7}$$

4.4.4 PROGRAM LOGIC

From the previous section, equations (4.6) and (4.7), it is apparent that to form any coefficient ℓ_{ij}, only rows (i) and (j) of the main sequence need to be in the core store. Therefore, if the main sequence is segmented and held in a backing store facility, only two segments are required in the core at any one time. Whereas it is suggested that large segments should be adopted to give the least number of storage transfer operations, it is possible to work with segments containing only one row, so that the segmented structure is very versatile and can be used with only a small amount of available core store (e.g. a desk top machine).

There are time penalties in adopting this method of storage, and it would appear that large segments are more efficient, as less time would be used in transfer operations.

Returning to equations (4.6) and (4.7), as stated previously only row (i) and (j) of the main sequence need to be in the core store. The segment in which row (i) is located will be called the Active segment while the segment in which row (j) is located will be called the Passive segment. Segments which are termed passive have all their coefficients factorised. Active segments however are in the process of being factorised. Figure (4.6) shows the area in which coefficients ℓ_{ij} may be determined with the given active and passive segments, Q and P respectively, in store. The procedure therefore is to factorise the active segment in sections, as shown heavily shaded in figure (4.6). To perform the factorisation of the Q'th segment

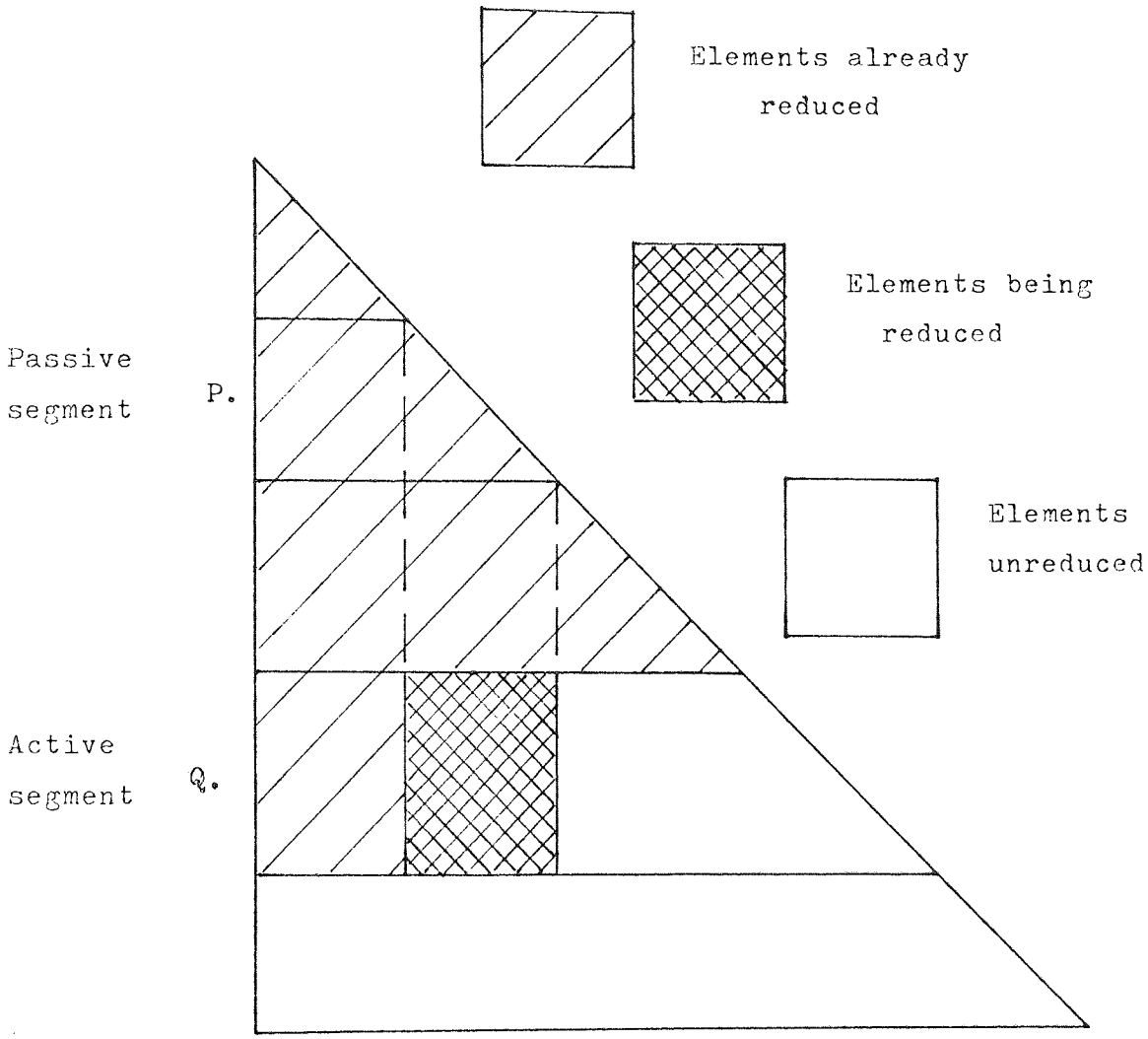


Figure 4.6 Showing the area in which coefficients l_{ij} may be determined with the given active and passive segments in the computer core.

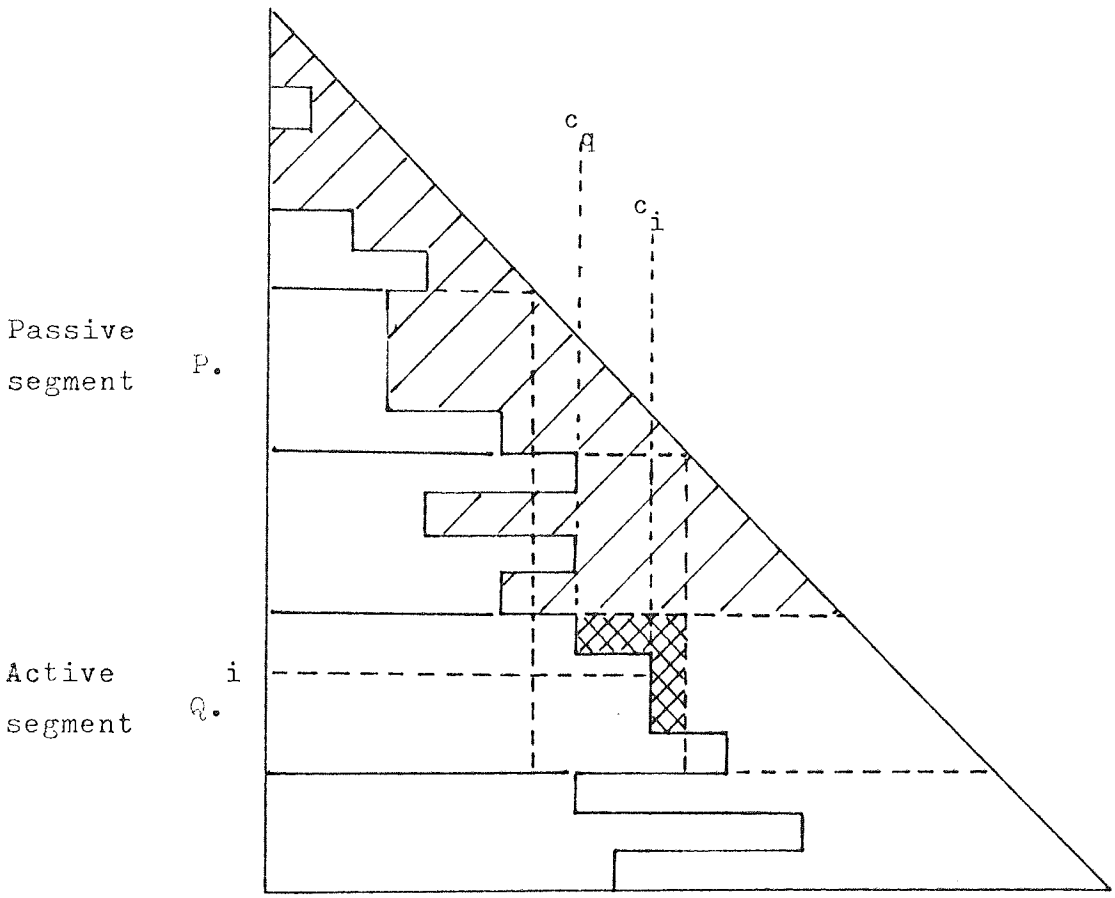


Figure 4.7 Showing the area in which elements l_{ij} will be determined for a given active segment Q and passive segment P associated with the local variable bandwidth type storage. Within this area the coefficients l_{ij} will be determined in sequence by rows.

it is necessary to call in the passive segments in turn from 1 to Q-1, with a final operation in which the active and passive segments can be considered to coincide.

Figure (4.7) illustrates a sparse array, in which the local variable bandwidth scheme has been employed. It can be seen that for a given active segment Q, it will not be necessary to call in the first passive segment. Therefore to determine the first passive segment to be called, it is necessary to inspect the column numbers c_i of the first elements in each of the rows of the active segment. By taking the least of those to be c_q it is possible to determine in which segment the corresponding row occurs, and this will be the first passive segment which needs to be called. Figure (4.7) shows the area in which elements ℓ_{ij} will be determined for a given active segment Q and passive segment P. Within this area the coefficients ℓ_{ij} will be determined in sequence by rows.

4.4.5 DESCRIPTION OF PROCEDURES

(i) PROCEDURE ADDSEQ

The purpose of this procedure is to determine the coefficients of an address array ADD, which relates the overall stiffness matrix coefficients in two dimensional form to their one dimensional form, thus

$$K[i,j] = K[ADD[i] - i + j] \qquad (4.8)$$

It was originally constructed by Robertson[28], and has now been extended to obtain control variables used in procedure SEGSOL. A brief account of the address array assembly will be given here, for further details see references (28,29) or refer back to section (4.4.2).

For a node (A), the corresponding ADD coefficient is found by scanning the nodal connections of the first element containing node (A), and if the largest node number is (B), the number of terms between the first non-zero and the leading diagonal of that particular row is,

$$\text{Number} = A - B + 1 \tag{4.9}$$

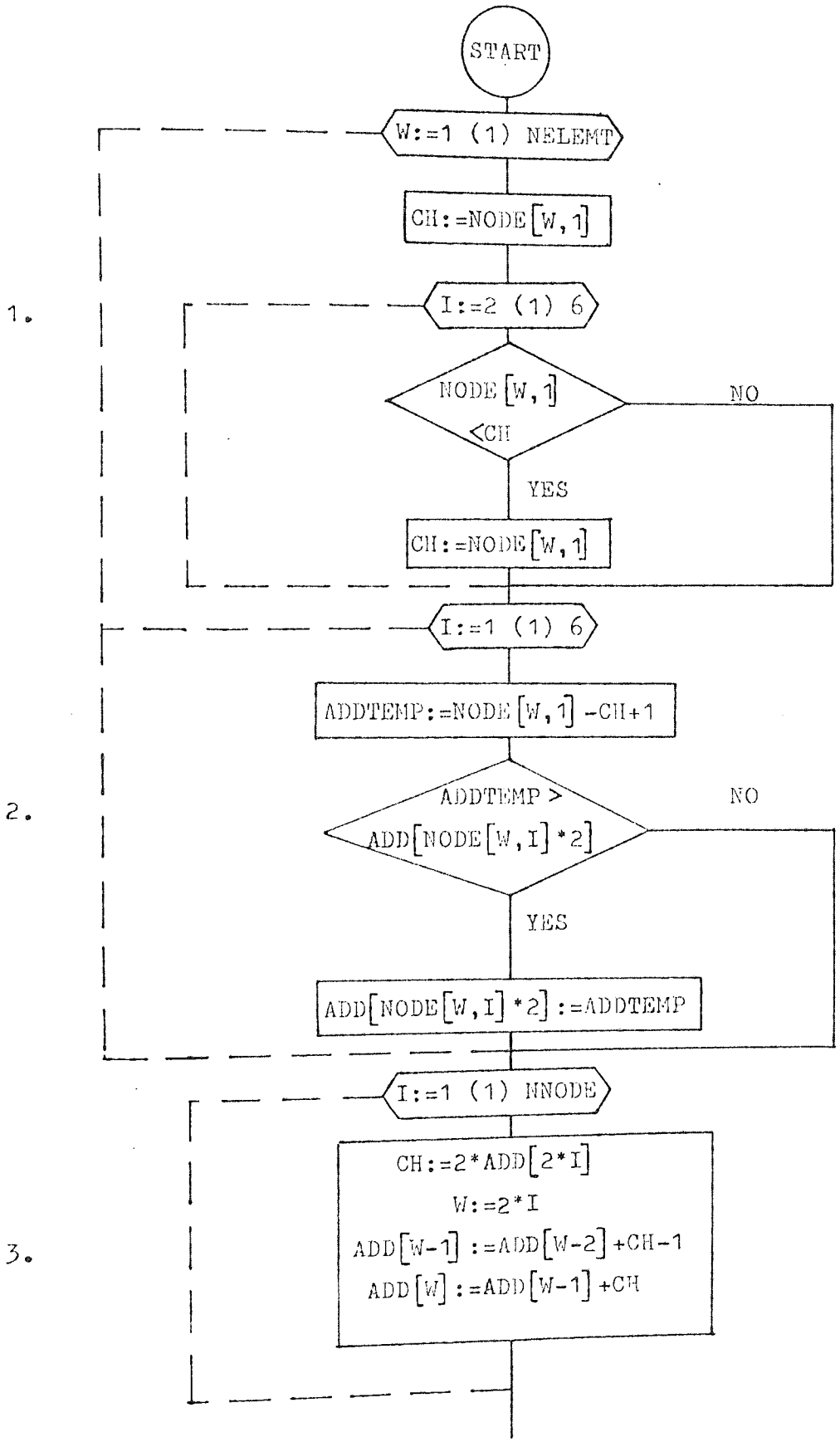
If, in a later element a larger coefficient is found for this row, it will replace the smaller one, so that the maximum number is obtained.

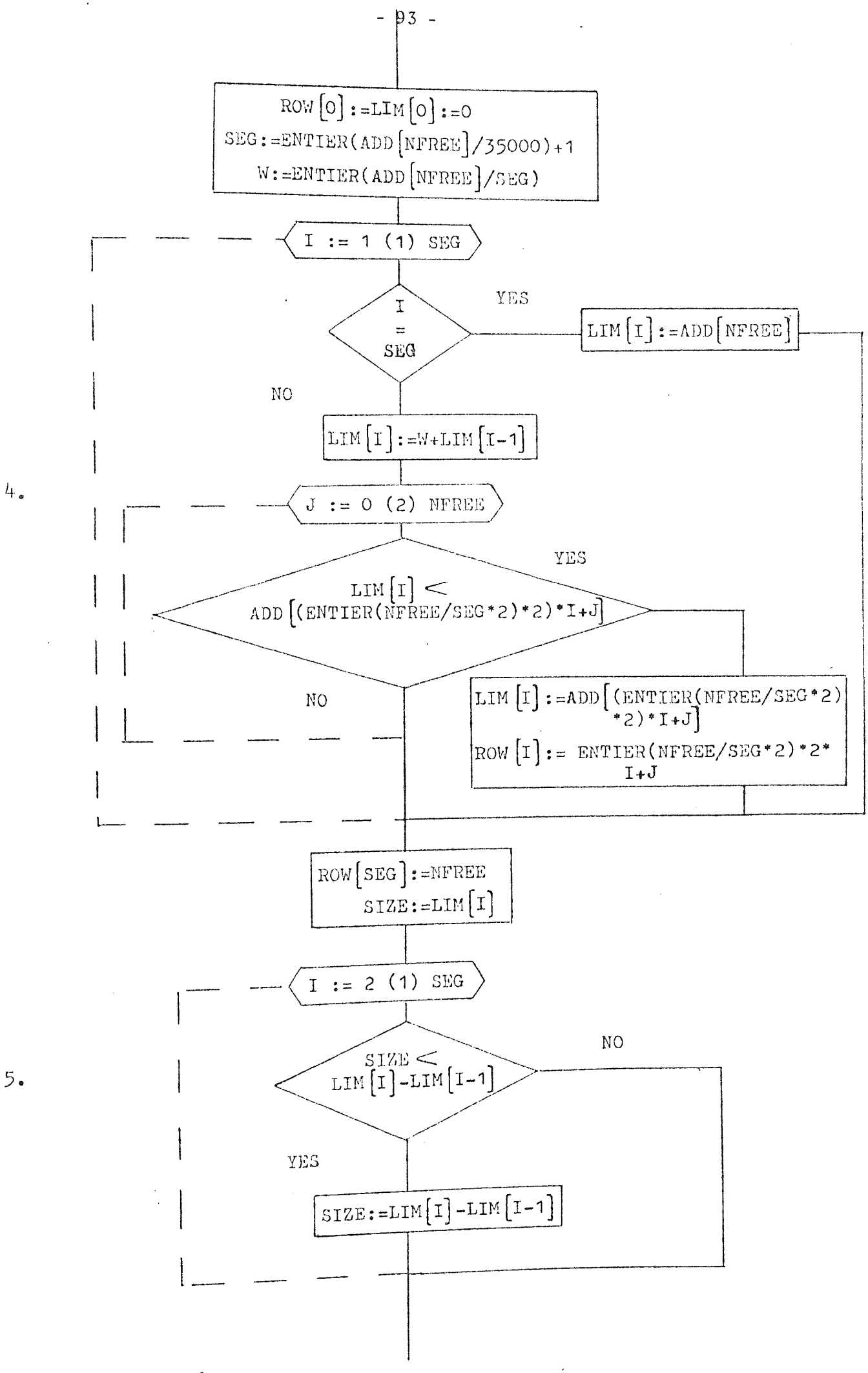
By extending this procedure, it was possible to use the address sequence in sub-dividing the main stiffness array. The array is divided into segments each containing approximately the same number of coefficients and also an integral number of rows. As the segments have been limited to 35000 words in length, the number of segments can easily be found. Various control values are also obtained from the subsequent division of the main array. Firstly, the limits of each segment must be computed, so that coefficients from the main array can be correctly located. Also the maximum segment size is found in order that sufficient computer core can be made available. Lastly, the solving routine requires the row limits of each segment and also the identity of the first passive segment which needs to be called, for each segment, when it becomes active.

The following steps describe how this procedure operates and correspond with the flow-chart.

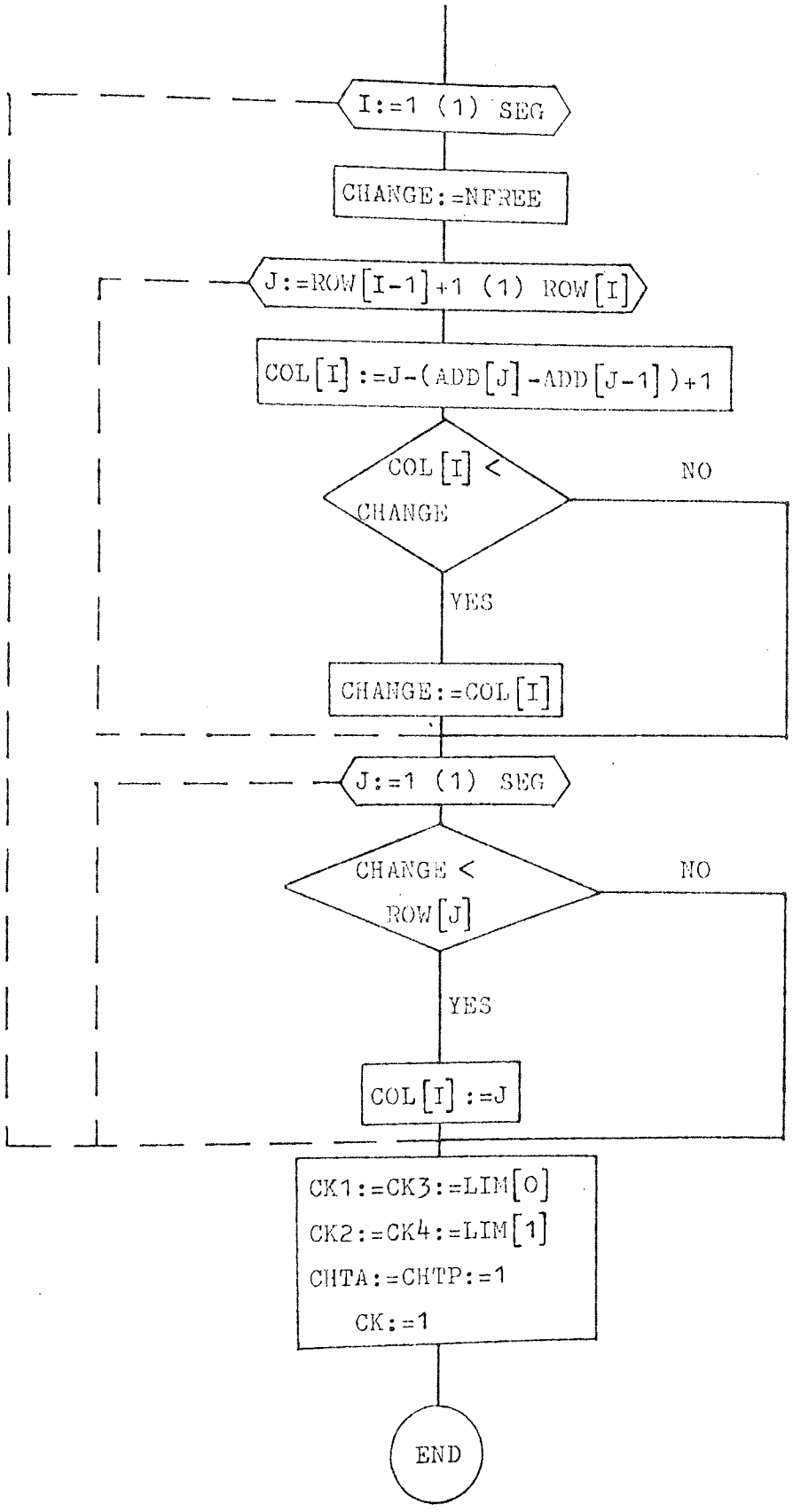
1. For the first element, the nodal connections are compared with each other to find the smallest node number, and the number of terms between the first non-zero coefficient and the major diagonal for each node in this element, that is in each row of the overall stiffness matrix, is calculated from equation (4.9).
2. Step (1) is repeated for the rest of the elements and if the number of terms for a node was found to be greater than the previously stored value, it is overwritten by the larger value.
3. The coefficients of ADD are determined from the assembled values found by step (2).
4. The first segment limit and row number are set to zero. The main array is then segmented, the limits of each segment and its corresponding row, being recorded in array LIM and ROW.
5. The largest segment is found, using the limits previously calculated in step 4.
6. The number of the first passive segment, which needs to be transferred into core, initially, when that segment becomes active, is recorded in array COL.
   This is achieved by computing the smallest column number in each row for a particular segment and then comparing this value with the row limits of each segment.
7. Various control variables are set so that procedure TEST is activated correctly.

Procedure ADDSEQ Flowchart




ROW[0]:=LIM[0]:=0
SEG:=ENTIER(ADD[NFREE]/35000)+1
W:=ENTIER(ADD[NFREE]/SEG)
I := 1 (1) SEG
I = SEG
YES
LIM[I]:=ADD[NFREE]
NO
LIM[I]:=W+LIM[I-1]
4.
J := 0 (2) NFREE
LIM[I] < ADD[(ENTIER(NFREE/SEG*2)*2)*I+J]
YES
NO
LIM[I]:=ADD[(ENTIER(NFREE/SEG*2)*2)*I+J]
ROW[I]:= ENTIER(NFREE/SEG*2)*2*I+J
ROW[SEG]:=NFREE
SIZE:=LIM[I]
I := 2 (1) SEG
5.
SIZE < LIM[I]-LIM[I-1]
NO
YES
SIZE:=LIM[I]-LIM[I-1]

6.



7.

```
CK1:=CK3:=LIM[0]
CK2:=CK4:=LIM[1]
CHTA:=CHTP:=1
   CK:=1
```

END

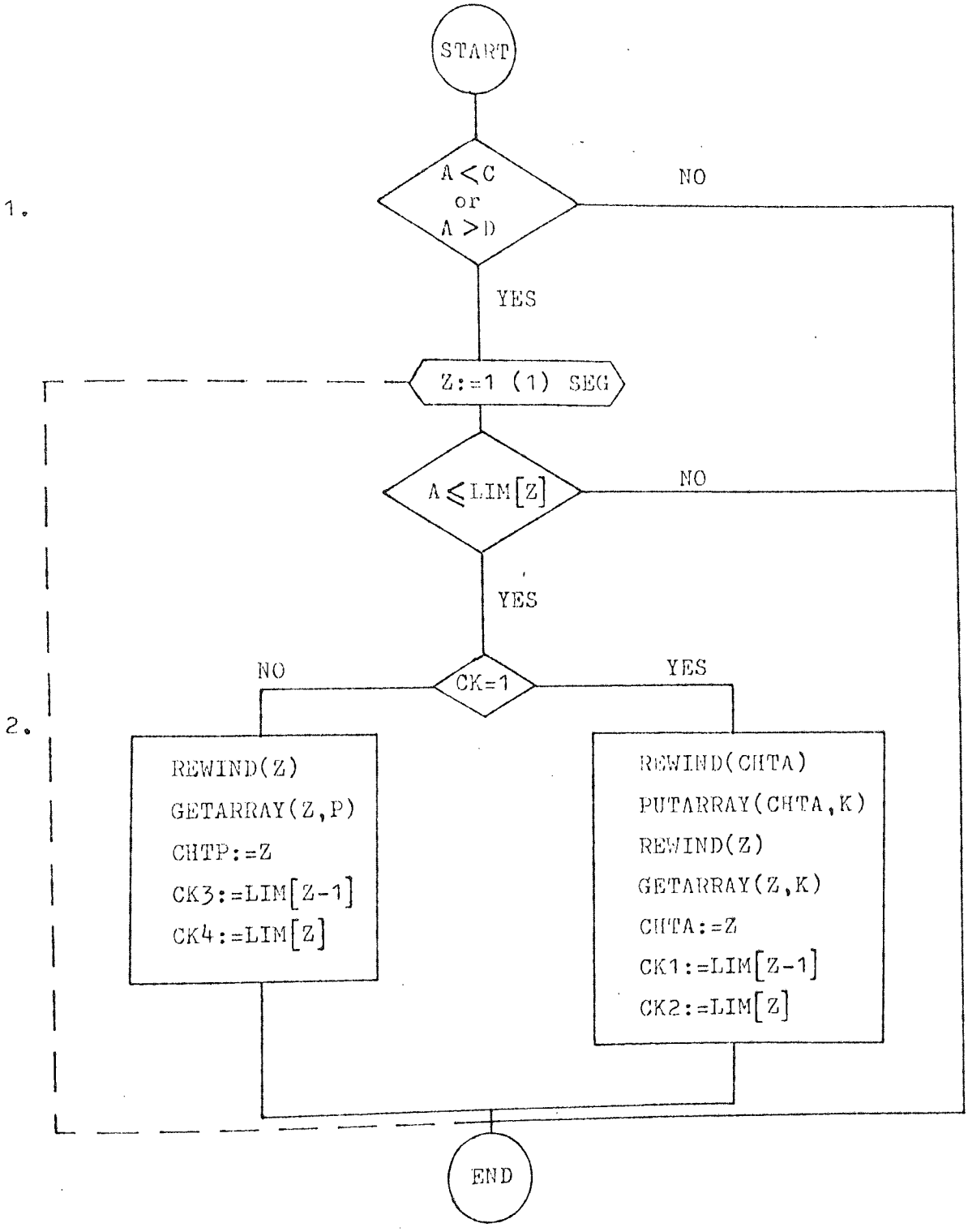
(ii) PROCEDURE TEST

In the ASSEMBLY procedure and various other ancillary procedures, it is necessary to locate or recall coefficients in the main array. Therefore as only one segment is held in core during this operation, a check is made to determine whether the coefficient lies within the bounds of the segment in store. The TEST procedure does precisely this and also exchanges the segment in core with the required segment in the backing store if the coefficient lies outside the core segment limits.

The following steps describe how the procedure operates and corresponds with the flow chart.

1. The main array element A is compared with the upper and lower limits of the segment in core, D and C respectively.
2. If the coefficient A lies outside the bounds of the segment in store, then a loop is set up to find in which segment A lies. When this is achieved the old segment is returned and the new segment recalled. The parameter CK refers to the ACTIVE or PASSIVE segments.

Procedure TEST Flowchart



(iii) PROCEDURE SEGSOL

This procedure operates as indicated earlier in section (4.4.4); the program logic. The construction of the procedure can be split into two distinct parts; (a) the first section factorises the array [K] and also determines the first or intermediate vector {y} equation (4.4), by forward substitution; (b) the second section determines the displacement vector {x} equation (4.5), by backward substitution.

The first part of the procedure operates in three nested loops: loop I which calls the active segments into core, loop W which calls in the passive segments corresponding to the variable COL[I], and finally the Z loop, which specifies the row numbers within each active segment.

The recursive relationships, (4.6) and (4.7), are formed within these loops. The coefficients ℓ_{ij} are constructed by the loop J for each subsection of the active segment; the limits being dependent on the passive segment that is in core. The coefficients ℓ_{ii} of equation (4.7), are only constructed when the W and I variables coincide, that is at the diagonal of the active segment.

The second section of the segmented solving routine is formed around the nested loops J and I. The J loop, in this section, is used to recall the factorised segments. Note that the last segment is carried across to the second part of the routine and hence is lost, as it is never transferred to the backing store. The loop I acts as a row counter within which the backward substitution is performed.

The following steps correspond with the flowchart and describe how the procedure operates.

1. The variable H is initialised and the segment in core is transferred to the backing store and replaced by the first active segment, held in array [A].

2. The loops I and W are set up and the first segment is called and stored in array [L].

3. The row counter z is constructed and variables, necessary for the control of the following steps, are computed. The variable H represents the element number of coefficients held in the one-dimensional array and is computed to correspond with the active sub-section shown heavily shaded in figures (4.6) and (4.7). The variables Q and P1 set the limits in the loop J, of the following section.

4. This step of the procedure corresponds to the recursive equation (4.6). The J counter, in effect, moves along each row of the active sub-section, enabling the coefficient ℓ_{ij} to be evaluated. At the start of this section control variables are computed which determine the limits of loop U and also the locations of the various coefficients within the active and passive segments in core. When the active and passive segments coincide, coefficients which have already been factorised are taken from the active segments, as indicated by the conditional statement. All factorised values are stored in the active segments.

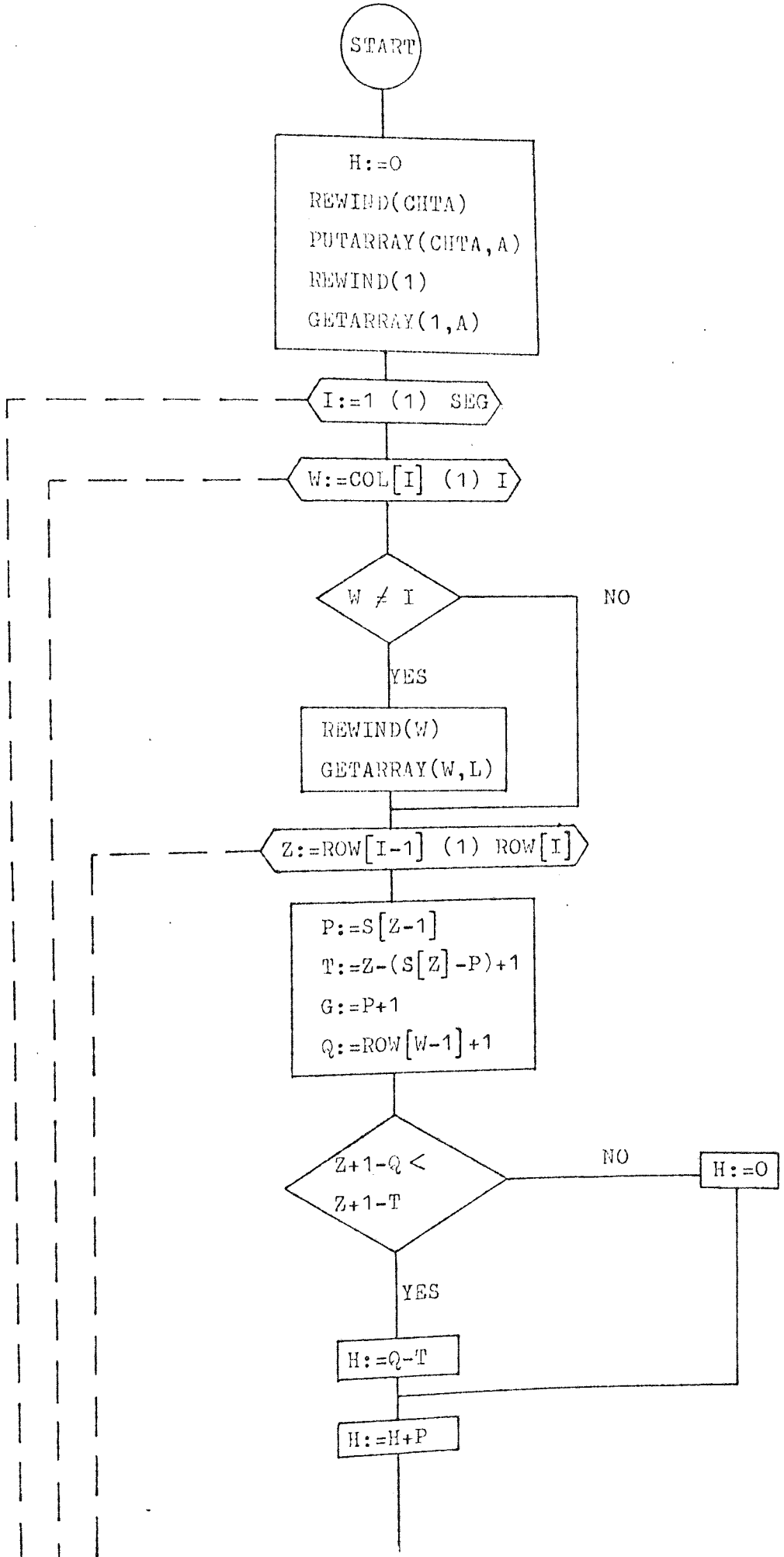
   As the factorised values are computed, so the forward substitution is performed in the counter loop M.

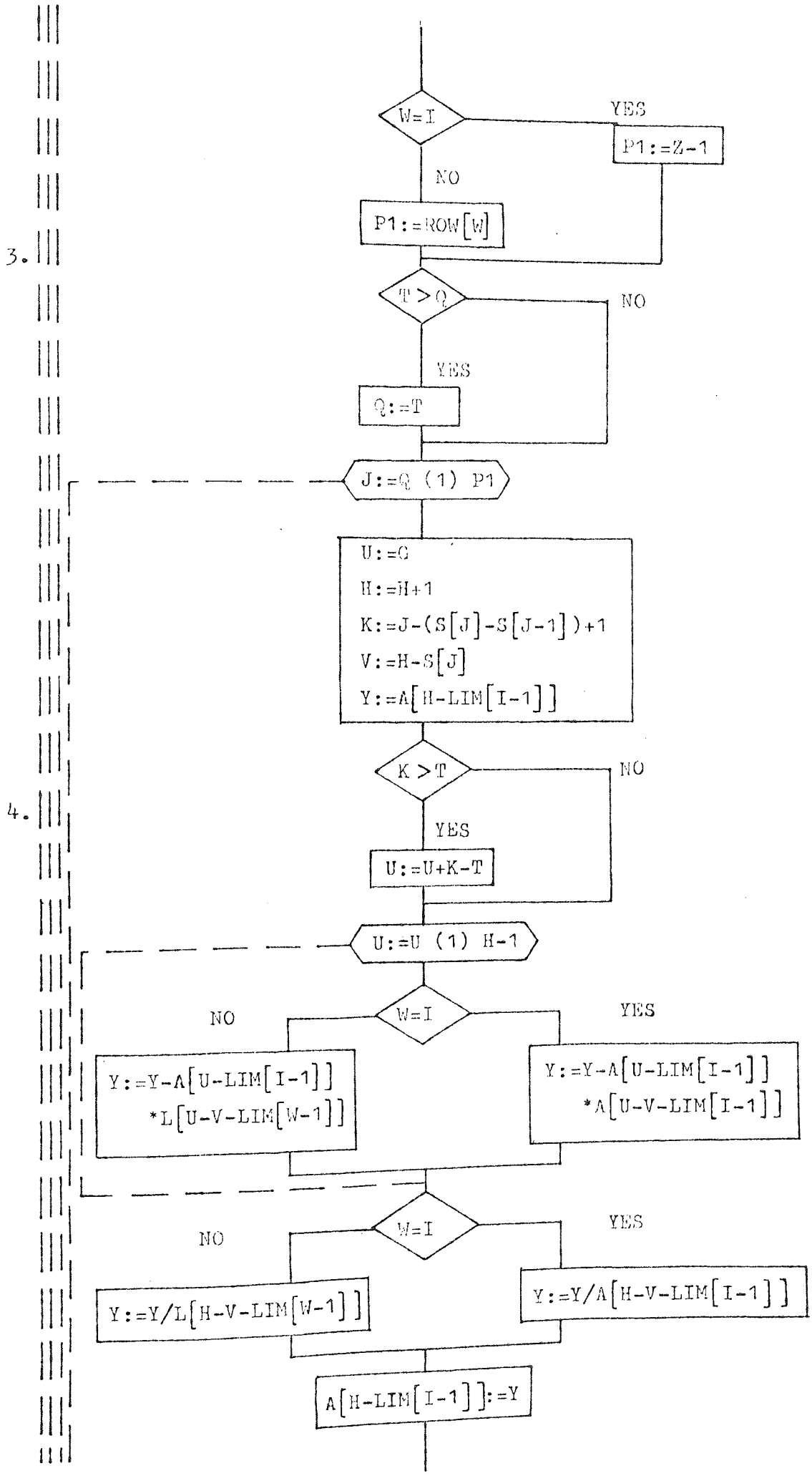
5. The fifth step is only accessed when W equals I, that is when the last sub-section of the active segment is being reduced. The algorithm represents the second recursive equation (4.7) and evaluates coefficients ℓ_{ii}. Again H represents the one-dimensional location of the stiffness elements. The forward substitution is performed as the reduced elements are obtained as in the previous step.

6. When the active segment has been reduced it is put into backing store and the next segment replaces it in array [A]. Steps 1 to 6 are then repeated until all the segments have been factorised. The last segment to be reduced is not transferred to the backing store, instead it is carried forward into the next section where the backward substitution is performed.

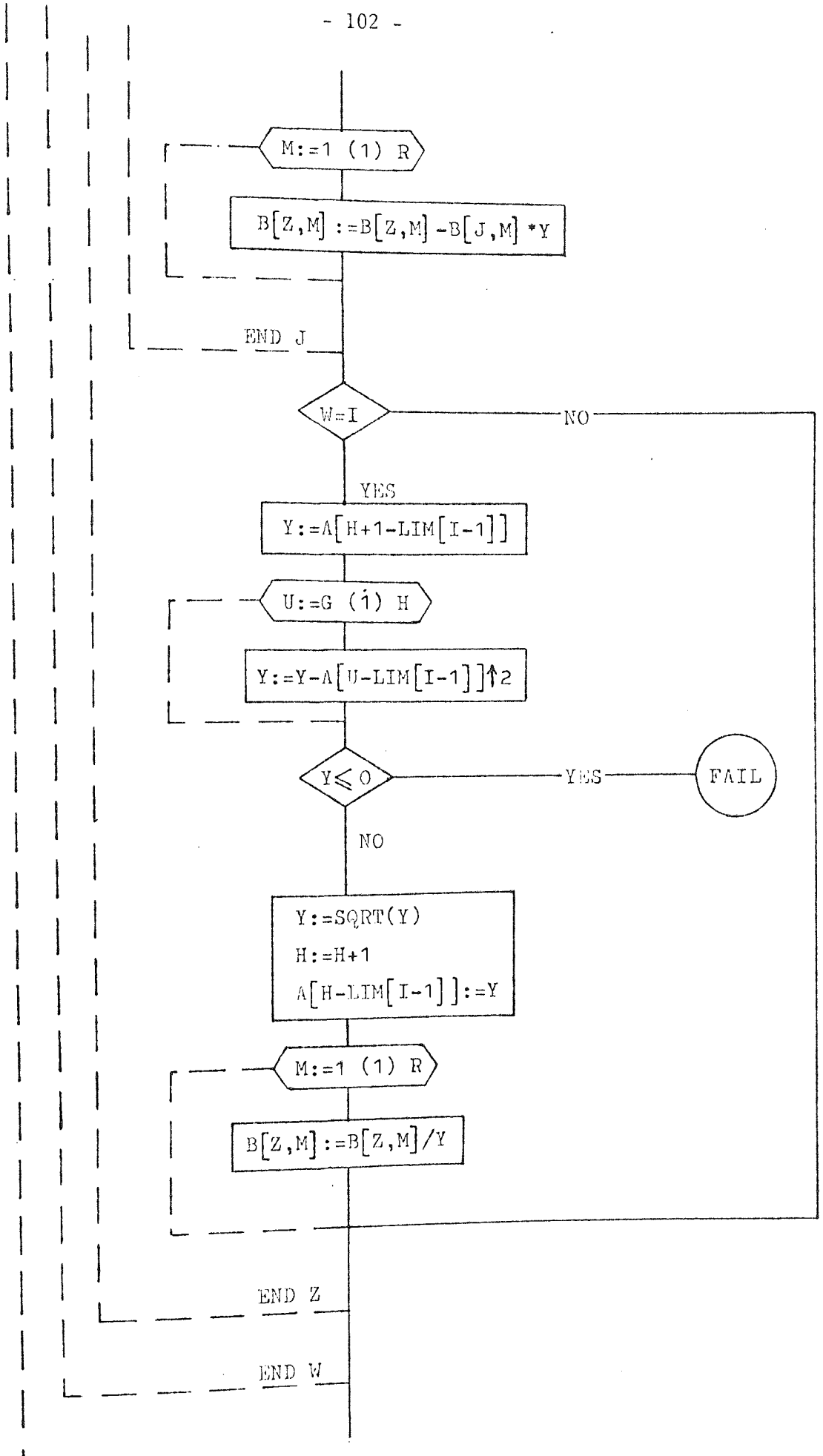
7. This last step performs the backward substitution and finally the displacement vector {x} is derived. The factorised segments are brought into core in reverse order via the loop counter J and the substitution process is similar to that of the above steps 4 and 5.

Procedure SEGSOL Flowchart

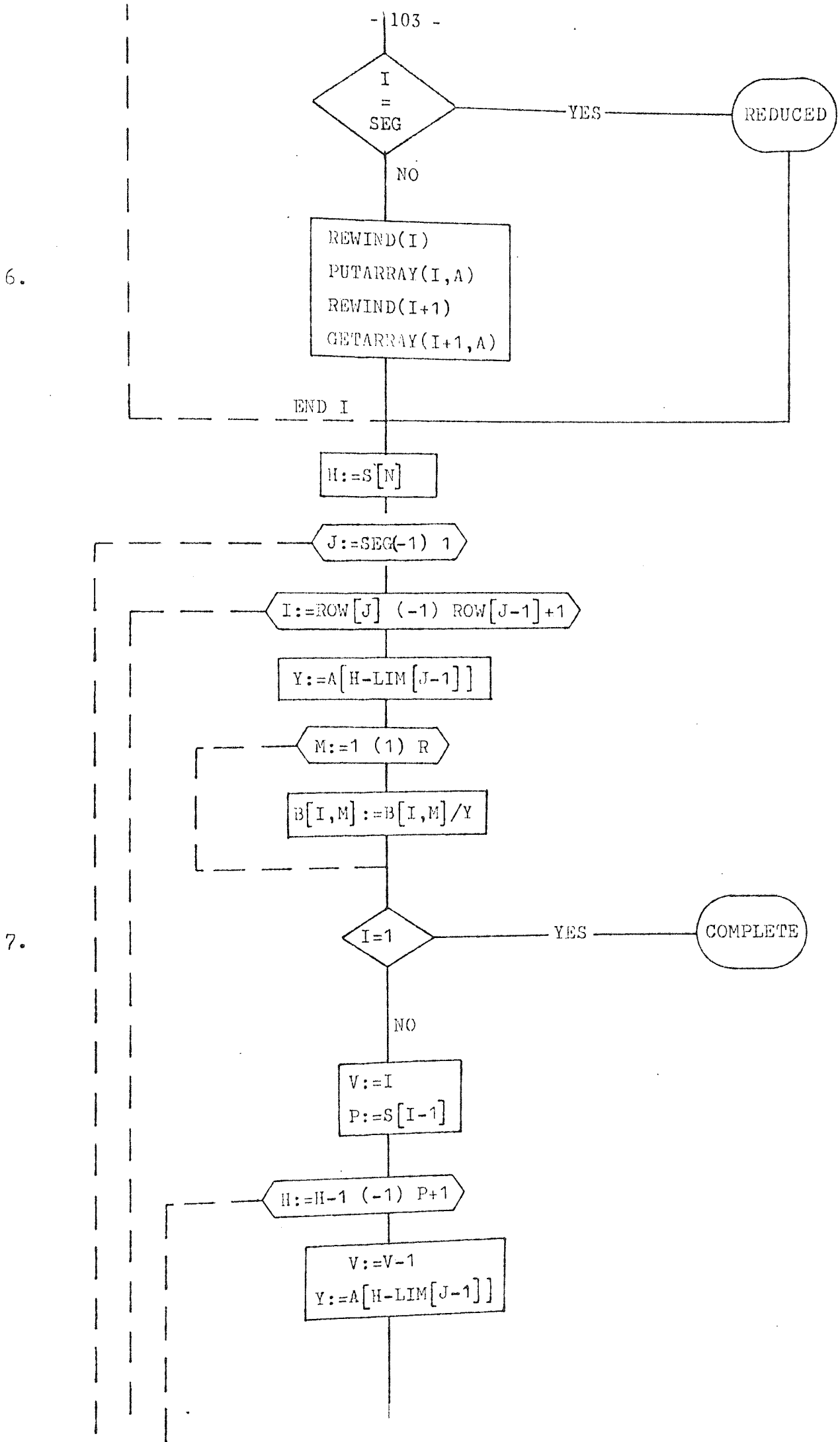



3.
W=I
YES
P1:=Z-1
NO
P1:=ROW[W]
T>Q
NO
YES
Q:=T
J:=Q (1) P1
U:=G
H:=H+1
K:=J-(S[J]-S[J-1])+1
V:=H-S[J]
Y:=A[H-LIM[I-1]]
4.
K>T
NO
YES
U:=U+K-T
U:=U (1) H-1
W=I
NO
YES
Y:=Y-A[U-LIM[I-1]]
*L[U-V-LIM[W-1]]
Y:=Y-A[U-LIM[I-1]]
*A[U-V-LIM[I-1]]
W=I
NO
YES
Y:=Y/L[H-V-LIM[W-1]]
Y:=Y/A[H-V-LIM[I-1]]
A[H-LIM[I-1]]:=Y

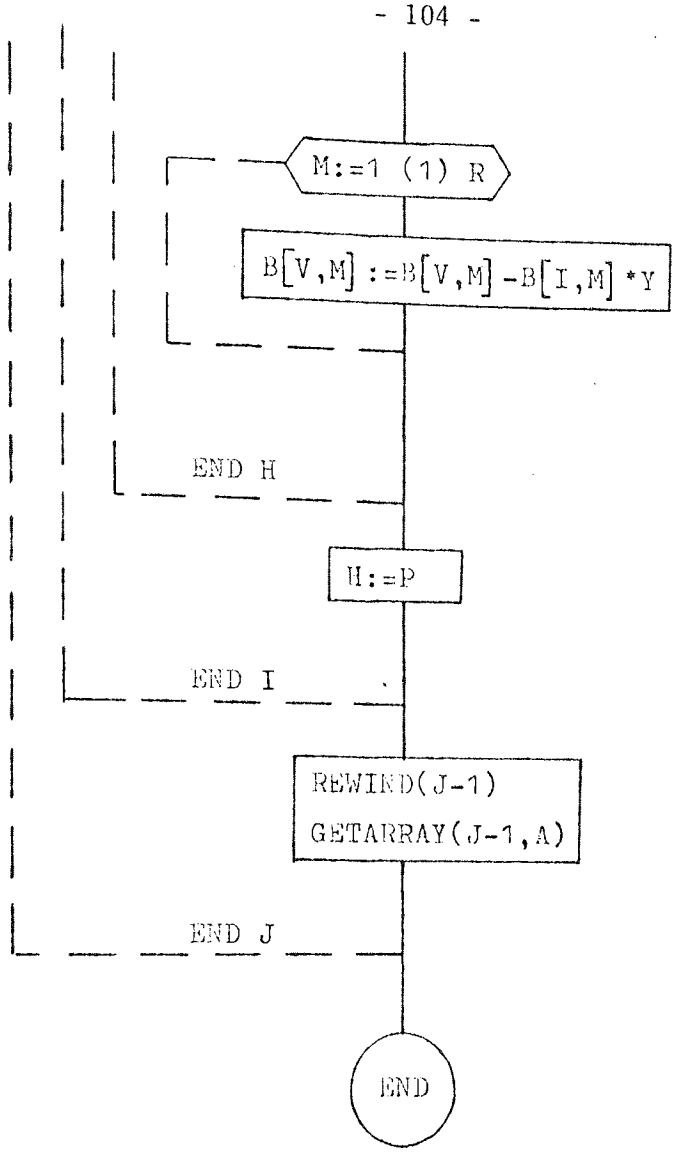
4.


M:=1 (1) R
B[Z,M]:=B[Z,M]-B[J,M]*Y
END J
W=I
NO
YES
Y:=A[H+1-LIM[I-1]]
U:=G (1) H
Y:=Y-A[U-LIM[I-1]]↑2
Y≤0
YES
FAIL
NO
Y:=SQRT(Y)
H:=H+1
A[H-LIM[I-1]]:=Y
M:=1 (1) R
B[Z,M]:=B[Z,M]/Y
END Z
END W

5.


I = SEG
YES
REDUCED
NO
REWIND(I)
PUTARRAY(I,A)
REWIND(I+1)
GETARRAY(I+1,A)
6.
END I
H:=S[N]
J:=SEG(-1) 1
I:=ROW[J] (-1) ROW[J-1]+1
Y:=A[H-LIM[J-1]]
M:=1 (1) R
B[I,M]:=B[I,M]/Y
I=1
YES
COMPLETE
7.
NO
V:=I
P:=S[I-1]
H:=H-1 (-1) P+1
V:=V-1
Y:=A[H-LIM[J-1]]

7.


M:=1 (1) R
B[V,M]:=B[V,M]-B[I,M]*Y
END H
H:=P
END I
REWIND(J-1)
GETARRAY(J-1,A)
END J
END

(iv) ALTERATIONS IN ANCILLARY PROCEDURES

As the stiffness matrix is assembled, one element at a time, each element must be tested using the previous procedure and positioned in the main array. This simply means that the procedure TEST is inserted in the appropriate place in the ASSEMBLY procedure. Also when the boundary conditions are applied in procedure GEOMBC, where rows and columns in the stiffness matrix are made zero, the TEST procedure is again used. In fact wherever the stiffness coefficients are manipulated the TEST procedure must be incorporated.

In other programs, such as the fracture solving routines, the incorporation of the singularity element involves the restructuring of the stiffness array and hence uses the TEST procedure extensively.

4.4.6 DATA TRANSFER

All data transfers are made via a channel number, these are declared in the job control cards, see Appendix (9.2). It would seem logical to have one channel to transfer the segmented stiffness matrix to and from the backing store, this could then be divided into several blocks using the limits obtained from the procedure ADDSEQ. Unfortunately there is no facility to achieve this efficiently, hence 'SEG' number of separate channels are required, one channel per segment. To transfer data, each channel is allocated 5K octal buckets from the Large Core Memory. Extra channels are required to transfer the finite element data and also record clean copies of the stiffness matrix and force vector. Therefore a possible 35 K octal bucket for data transfer alone is taken from the existing 345 K octal

Large Core Memory. The computer also requires a large amount of core to manipulate the program, and hence the problem becomes one of conflict between computer operational space and data storage. It is possible to limit the core space required for each channel by inserting a REQUEST card in the job deck, this will reduce the storage allocation to 1 K octal per channel. However the time taken to transfer data increases because the data can only be moved in 1 K octal buckets.

The channels can be used in a variety of ways, for example the finite element data is fed into the program via channel 11, so that all read and copy statements select data from this channel. The segmented stiffness array is held on 'SEG' number of binary sequential files and the files are accessed by the PUTARRAY and GETARRAY statements. See program listings. Before the above access statements can be used the file must be re-wound so that the file marker is positioned at the start of the data sequence. Because of the system used at Manchester Computer Centre each backing store file is given an input and output channel number differing by twenty. This avoids confusion and from the job deck examples, in the Appendix (9.2), each input channel number is equated to its equivalent output channel number.

4.5 MODIFICATION OF THE AXISYMMETRIC PROGRAM TO ACCEPT THE GENERATED DATA FORMAT

The axisymmetric finite element program , written by Alsharqi[29] has been modified to accept the data format output by the general mesh generation scheme. The program already contained a limited mesh generating system incorporated within the procedure FEINPUT. Input

variables controlling this mesh generation scheme, were used to identify elements which coincided with the vertical axis, i.e. r = 0. This information was then employed in procedure STRDIS, the equivalent of procedure AUX in the 2-D program, in order to avoid an indefinite solution, that is, division by zero. The simple mesh generation scheme always provided consecutive element numbers on the vertical axis, however with the more general mesh generation program this situation would rarely be encountered.

The essential steps needed to modify the program are:-

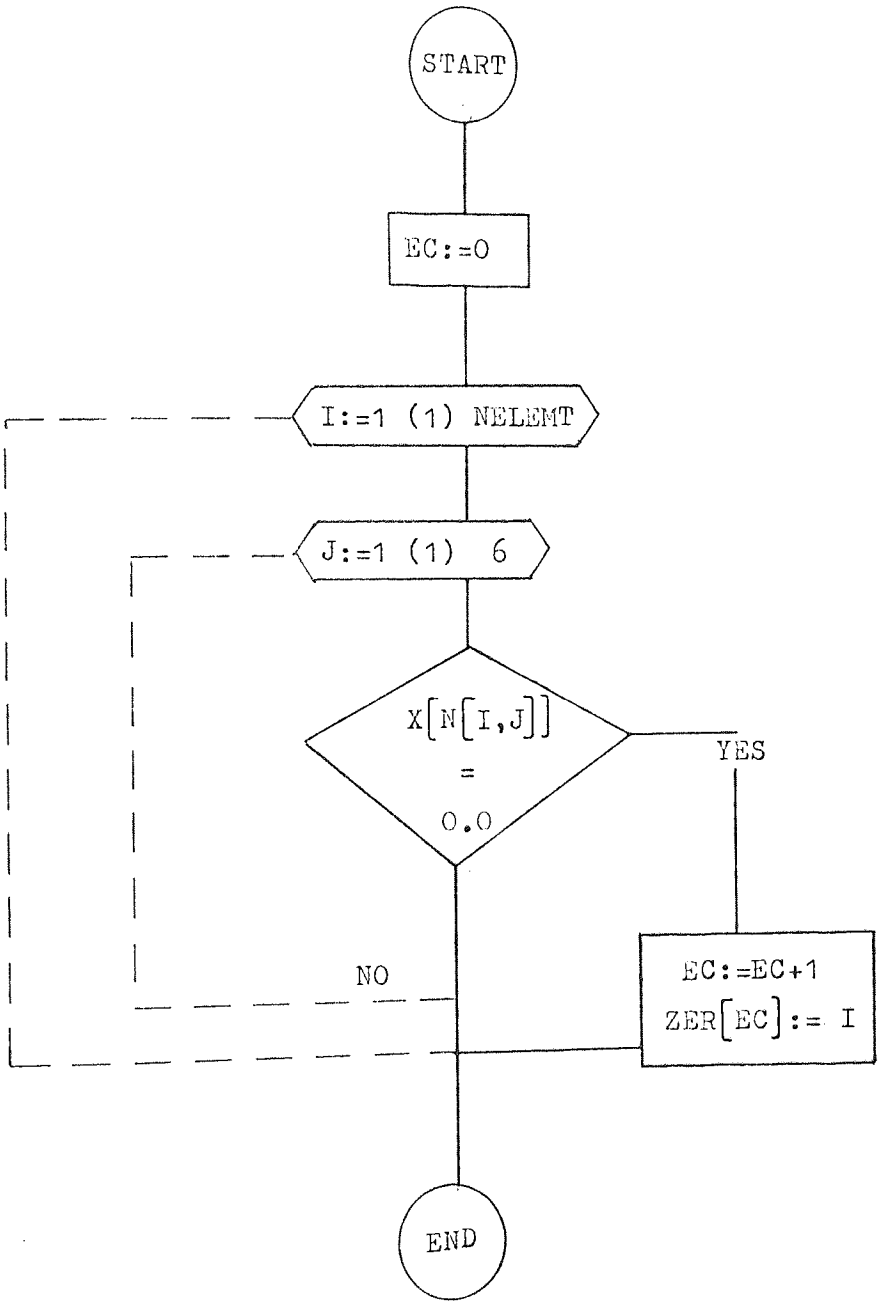
1. Remove the simple mesh generation scheme attached to the procedure FEINPUT.
2. Alter the read statements so that the new mesh scheme data format is accepted.
3. Insert a simple routine for determining all the elements lying on the vertical axis.

4.5.1 PROCEDURE RZERO

In the limited generation scheme, the elements lying on the vertical axis could be identified through a simple expression, with the new mesh generation program this was not possible and the elements on the vertical axis had to be computed. The procedure RZERO determines these elements and stores the information in array {ZER}. The following steps refer to the flowchart overleaf.

1. The summing integer EC is equated to zero.
2. The loops I and J are formed so that each nodal coordinate can be scanned. If the node is found to lie on the vertical axis then its element number is recorded in vector {ZER} and the next element's nodes are examined.

Procedure RZERO Flowchart



4.5.2 ANCILLARY MODIFICATIONS

The procedure STRDIS had to be altered slightly to accept the new element information of vector {ZER}, the essential logic of the routine was not altered. Several small adjustments were made to accommodate the above modifications. A listing of procedure RZERO and STRDIS is given in the Appendix (9.3) The new axisymmetric program is held on file under the name IAAXMG: see the file index in the Appendix (9.3.1).

The modified program was proven using a simple test problem and the results compared with those obtained using the original program. The outputs were found to be identical and the program is therefore considered to function correctly.

CHAPTER 5

AUTOMATIC GENERATION OF PLANE FINITE ELEMENT MESHES

5.1 INTRODUCTION

The finite element method, although providing a powerful tool for the Stress Engineer, has potentially one great disadvantage: the amount of input data required for any realistic problem is vast. The preparation of this data is exacting, time consuming and in most cases a large portion of the work is attributed to checking the element nodal connections and mesh co-ordinates. Therefore a general automatic mesh generation program would greatly relieve this work load. Having reduced the amount of input data required, the occurrence of human error will correspondingly be greatly reduced.

Recognizing the need for some form of rationalized data input scheme, a number of approaches have been suggested. The scheme presented by Akyuz[72] considers a region as an assembly of sub-domains, each sub-domain being defined mathematically. The approach is somewhat complex and stems from the mathematics of topology. A similar method is given by Reid and Turner[73], however it cannot be applied to general problems and is only referred to here for completeness. A technique outlined by Zienkiewicz and Phillips[74], indicating how a mesh generation scheme of maximum flexibility could be devised, has the appealing quality of simplicity. The scheme provides a method for

generating meshes in planes and more general surfaces in three dimensions. This method has been adopted here and appropriate computer coding developed. The mode of operation is given in the following sections.

5.2 THEORETICAL BACKGROUND

The basis of the scheme is the use of isoparametric curvi-linear mapping of quadrilaterals. The mapping technique was originally devised for generating element shape functions, providing an element which could approximate a curved boundary. The name 'isoparametric' is derived from using the same interpolation functions to define the element shape as are used to define displacements within the element. Isoparametric co-ordinates are a type of natural co-ordinate system.

Now consider an eight-noded quadrilateral element having sixteen degrees of freedom u_i, v_i (i = 1-8) corresponding to two at each corner and midside node. Refer to section (2.3). The mapping technique relates a unit square in isoparametric co-ordinates ξ and η to the quadrilateral in x,y co-ordinates, whose size and shape are determined by eight nodal co-ordinates x_1, y_1, x_2.....y_8. The mapping functions also provide an interpolation scheme that yields the x and y co-ordinates of any point within the element when the corresponding iso-parametric co-ordinates are given. Hence mapping functions of the form,

$$x = [N_1, N_2, N_3, N_4, N_5, N_6, N_7, N_8] \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \end{Bmatrix} \qquad (5.1)$$

or $$x = \sum_{i=1}^{8} N_i x_i \tag{5.2}$$

and $$y = \sum_{i=1}^{8} N_i y_i \tag{5.3}$$

are required.

The above functions $N_i(\xi,\eta)$, have been derived by inspection and are from the so-called "Serendipity" family of shape functions described in References (1, 3 & 4). Hence for

Corner nodes

$$N_i = \frac{1}{4}(1 + \xi_0)(1 + \eta_0)(\xi_0 + \eta_0 - 1) \tag{5.4}$$

and Mid-side nodes

for $$\xi_i = 0, \quad N_i = \frac{1}{2}(1 - \xi^2)(1 + \eta_0) \tag{5.5}$$

$$\eta_i = 0, \quad N_i = \frac{1}{2}(1 + \xi_0)(1 - \eta^2) \tag{5.6}$$

$$\xi_0 = \xi\xi_i, \quad \eta_0 = \eta\eta_i$$

If the region in which the mesh is to be generated could be described adequately by a quadrilateral, as given in figure (5.1a), then a mesh of any refinement could be automatically generated inside it by specifying the co-ordinates of the eight nodal points and the number of required subdivisions in the ξ and η directions.

In the present scheme the nodal points of the mesh are created and numbered from the lower left-hand corner, vertically and from column to column, in equal ξ, η increments. From this information the element nodal connections can be established using an appropriate rule. It can be seen that various element types can be generated using this inter-

polation scheme, and figure (5.1a) illustrates how a quadrilateral or triangular element can be formed. The program presented in Section (5.4), has been designed to generate data for both the eight-node quadrilateral element and the six-node triangular element. Returning to figure (5.1a), it can be seen how the elementary quadrilaterals are formed, and the shaded area shows how this region is divided, using the shorter diagonal, to give two triangular elements. Selecting the shorter diagonal of the elementary quadrilateral ensures that a better or reasonable triangular shape is obtained.

To avoid confusion, the quadrilateral element used as the basis of the mapping scheme, will be termed a 'zone'. As recounted earlier, the input information consists of the zone's eight nodal co-ordinates and it's required subdivision in the ξ and η directions. If the zone boundaries are straight, then the co-ordinates of the mid-side nodes can be omitted from the input data, and this information can be determined by interpolation. However, the mid-side node will not only specify the parabolic shape of the zone's boundary, but will also grade the internally generated elements. This can be seen from figure (5.1b), where the square symbol represents the mid-side node, positioned so as to form a graded internal mesh. Figure (5.1b) also represents the next stage of development, where the zones are grouped in a 'chequerboard' fashion, described as a zone array, allowing more complicated component shapes to be meshed. By placing the zones in this uniform grid it is possible to construct a logical program which examines each zone in turn and links the zones through the generated nodal points.

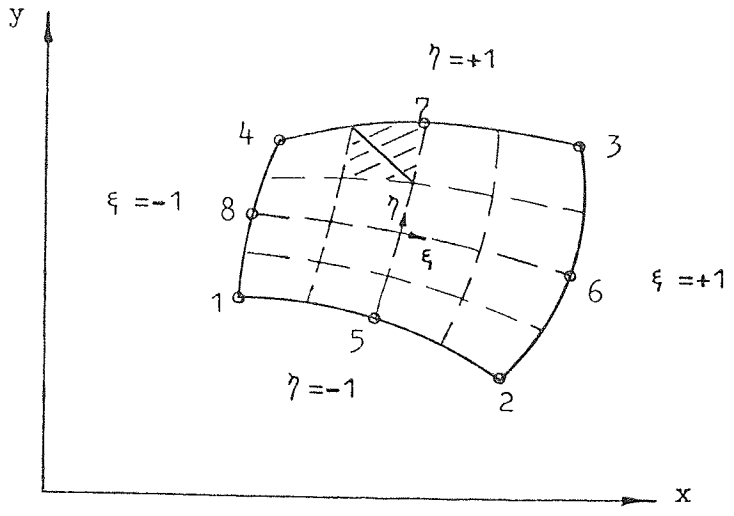
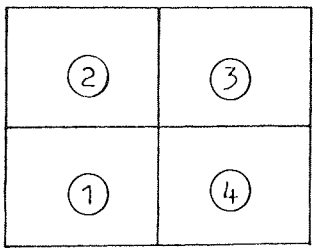
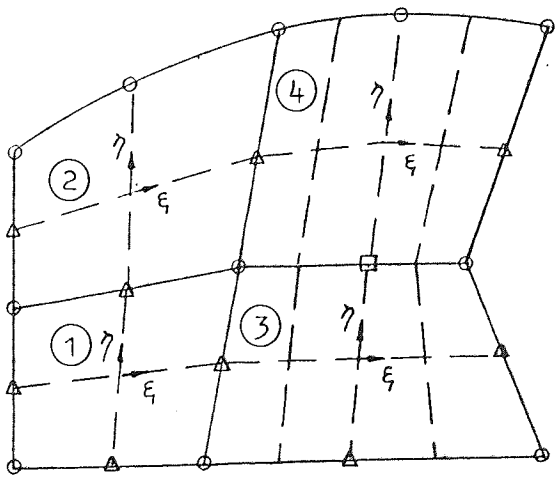


FIG 5.1a Isoparametric Quadrilateral element, internal points can be interpolated using the function $N(\eta,\xi)$.



Grid of Quadrilateral elements - Zone Array.

Distorted Zone Array.



o Specified Points.
Δ Not specified interpolated.
□ Specified for mesh grading purposes.

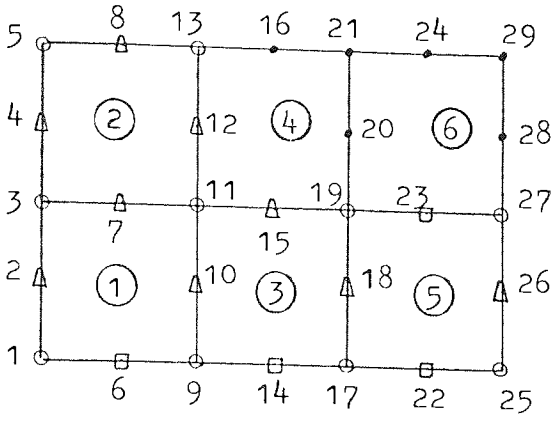
FIG 5.1b Grouping Quadrilateral elements in a grid or zone array allows more complicated components to be meshed.

To demonstrate the sequence of operation in the construction of a finite element mesh, figure (5.2) illustrates the basic steps, from the regular zone array to the final meshed component.

The construction of the zone array is dependent on several basic criteria. Firstly, the zone array must be regular in form, that is rectangular. It's dimensions are arbitrary, allowing the operator to select the size of array which is appropriate to his particular requirements. Secondly the orientation of the zone array may affect the bandwidth of the resulting stiffness matrix. The matrix bandwidth is proportional to the difference in adjacent element node numbers. Hence, for example, the zone array given in figure (5.2), if rotated through 90 degrees would still provide the same result, although the zones would be numbered differently, however, the difference in the element node numbering would be greater and as a result one would expect the storage space for the [K] matrix to increase. As stated previously the element nodal points are created and numbered from the left-hand corner of the zone array, moving vertically and from column to column in equal ξ and η increments. Finally zones can be declared as 'void' that is they are not used in the construction of the finite element mesh. Figure (5.2) illustrates two such void zones and it is clear, from the accompanying figures that they play no roll in the mesh construction. The material number for a void is zero, hence the program has been designed accordingly to account for such redundant zones. The zones can be drastically distorted, even to the point of making two sides lie on the same line, forming a triangle. Care must be taken, however, not to make any corner angles greater than 180 degrees,

as non-uniqueness of mapping may result. Having provided the basic framework for the construction of the zone array, it is left to the user to experiment; some examples are given in Section (5.5) which may help in this respect.

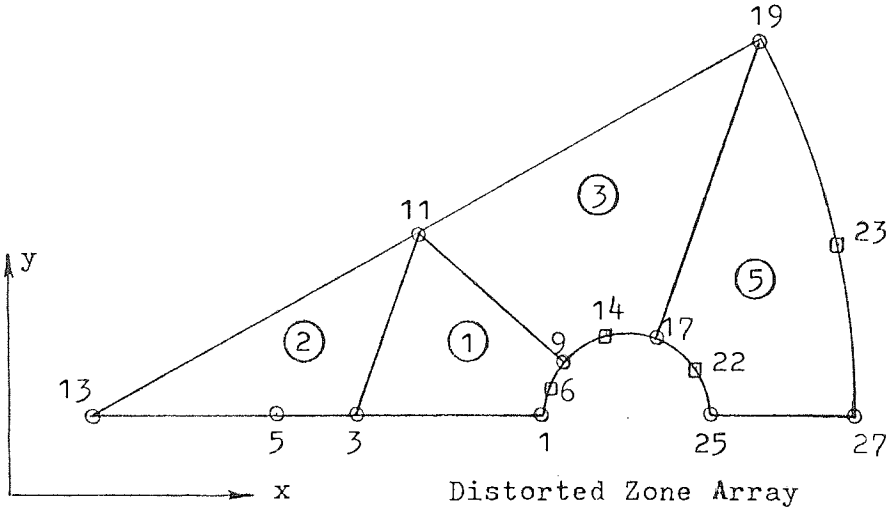
The zone array is accompanied in figure (5.2) by the distorted zone array, which illustrates how the zones are used in the construction of the component shape. Here it is convenient to 'wrap' the zones around the semi-circular cut out and discarding zones 4 and 6, allows zone 2 to be used to complete the component shape. Each zone must be defined by eight nodal coordinates obtained by direct input, as data, or interpolated during the program operation, and also by a set of zone nodal connections. It was decided to number the zone array in a regular manner, from the dimensions of the zone grid, as shown in figure (5.2), and thus avoid the need to insert the zone nodal co-ordinates. In other words, because of the uniform zone numbering pattern adopted, it was a relatively simple matter to predict the zone nodal connections from an appropriate rule. This represents a great saving in the amount of data input, especially in large problems, but it does require the user to number the zone array in the manner shown in figure (5.2). It would over-complicate the zone numbering scheme, if redundant or void zones were to be taken into account and consequently all of the zones are allocated node numbers whether they are used or not. The distorted zone array shows clearly the zone node numbers which need to be specified as input data, and the remaining node co-ordinates can be interpolated where necessary. Comparing the zone array and the distorted zone array, it can be seen how the mapping process functions.



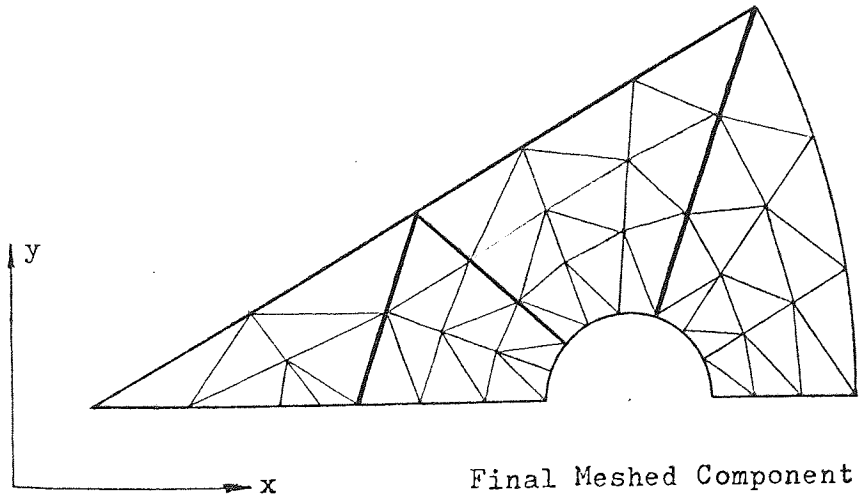
Zone Array

- Specified points.
- Specified to fit curved boundary.
- Not specified interpolated.
- Redundant points.

◯ - Zones



Distorted Zone Array



Final Meshed Component

FIG 5.2 Constructing a finite element mesh.

The final stage of the mesh generation process involves determining the element nodal co-ordinates. This is achieved as indicated by the previous example, in which a single zone was used, and the element node co-ordinates were found using the specified zone subdivision. The same process is used in the zone array and each zone is treated in turn, in the order given by the zone numbers shown in figure (5.2). The element nodal connections are obtained using an appropriate rule and the program structure is given in section (5.4) where further information is given on the nodal numbering scheme.

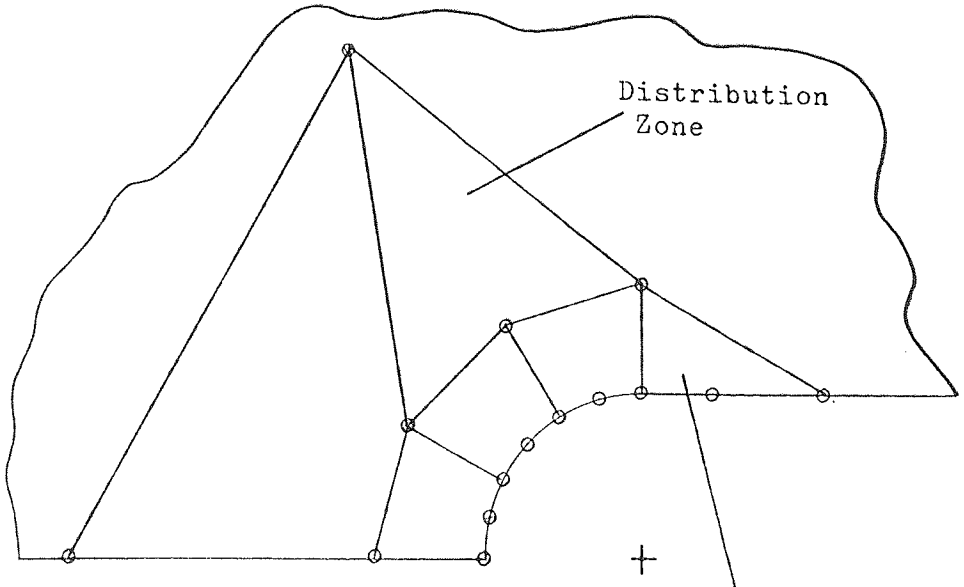
5.3 GENERAL COMMENTS ON MESH CONSTRUCTION

When setting up the finite element mesh, consideration must be given to where large stress changes are likely to occur, and to accommodate them by grading the mesh accordingly. Once an idea of the type of mesh required has been formed, the component can be drawn out to scale and the distorted zones sketched in. After this rough assembly has been completed it is necessary to determine whether the zone array can be constructed. To achieve this, the distorted zone array may have to be altered. This commonly occurs and some manipulation may be required before the zone array can be formed.

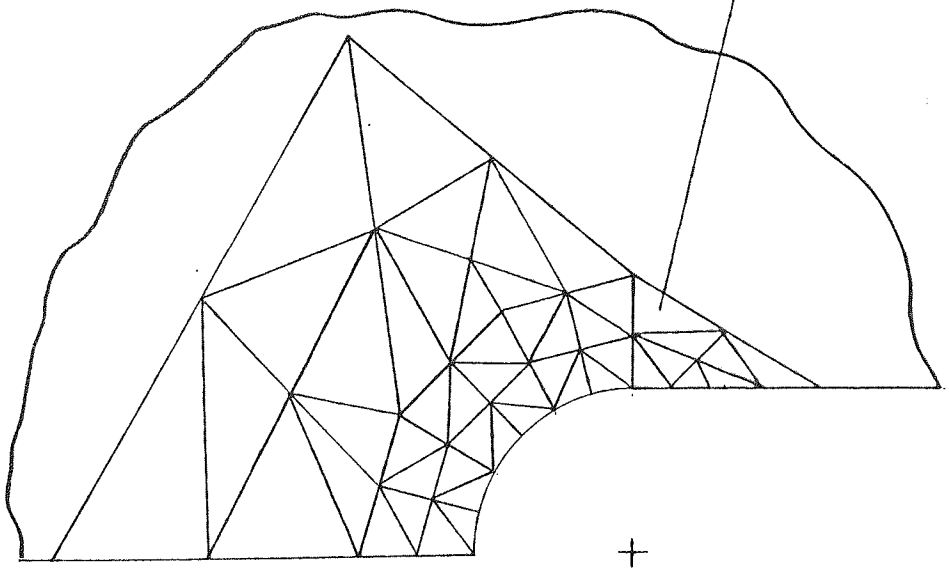
At this stage the elements should be sketched in lightly and again the zone co-ordinates might require alteration to acquire a reasonable element size and proportion. (Note that when drawing in the elements, for the triangular element, the quadrilateral or zone is sub-divided and each sub-quadrilateral is split into two triangles across the shortest diagonal.) Having assembled these two diagrams it is simply a matter of compiling the data as described in section (5.5.1).

In practical applications, it is valuable to construct two zone arrays, one containing the 'super' node numbering and zone numbers, and the second containing the element node numbers. The term 'super node' is used to distinguish zone node numbers from element node numbers. The first diagram is useful when compiling the mesh generation data, and the second diagram acts as a check on the total number of finite elements and nodes specified, and also the boundary conditions can be applied using the element node numbers in conjunction with the distorted zone array.

In applying the auto-mesh generation program to a large number of problems, various devices have emerged which can be used directly in the majority of cases. In areas of high element concentration a 'terminating zone' is usually employed to confine the dense mesh and prevent it from propagating into low stress areas where larger elements could be used. A 'terminating zone' takes the form of a triangle, as shown in figure (5.3), where two sides of the quadrilateral zone form one side of the triangle. Therefore, if the zones placed around a high stress concentration, have been subdivided to give a fine element mesh, then rather than allowing the fine zone subdivision to propagate from one zone to the next, it can be terminated on the component's boundary using a 'termination zone'. Where a rapid element size change is required, it has been found that an arrow shaped zone, termed a 'distribution zone', can be used. Figure (5.3) illustrates the way in which the distribution zone provides a gradation in element size, in moving from a dense to a sparse mesh area, avoiding abrupt changes in element size. If a simple extension of the zone array were to be used,



a, Distorted zone configuration.



b, Meshed zones.

FIG 5.3. Diagrams a, and b, show two basic zone types, and are suitably named the Distribution Zone and the Terminating Zone.

either a gradual element size increase is obtained or an undesirable spiked mesh is generated (see figures 5.4a,b). In the first case larger elements could have been used with no loss in accuracy, hence computer storage space has been wasted in handling the extra number of elements; in the second case by reducing the number of elements a spiky mesh has resulted. Care must be taken when using a 'distribution zone' which has a corner angle exceeding 180 degrees, as the elements may be mapped outside the zone area. For example care must be taken not to 'over stretch' the zone in an attempt to reduce the element density rapidly. An example of this is given in figure (5.5), where the element co-ordinates are mapped outside the zone boundaries. Note that if this does occur it can easily be identified in the graphical output. The advantages of the distribution zone, become apparent, when comparing figures (5.3) and (5.4).

It has been shown in section (2.5), that the isoparametric element is sensitive to distortion and this can give rise to loss of accuracy in the finite element solution. The mesh generation program has been designed so that all zone internal element faces are straight and yet allows the zone faces to be curved when modelling curved component boundaries. This safeguards against curved boundary propagation into the zone and therefore indiscriminate loss of element performance. The element accuracy is also susceptible to mid-side nodal displacement. It has been previously stated that mid-side super-nodes can be used to fit curved boundaries and also effect an element gradation. Using the mid-side supernode to grade the mesh in this fashion, displaces the mid-side element nodes on the zone's boundary and can modify the displacement function causing loss of accuracy. This problem comes under the heading of program development and will be discussed in Chapter Eight.

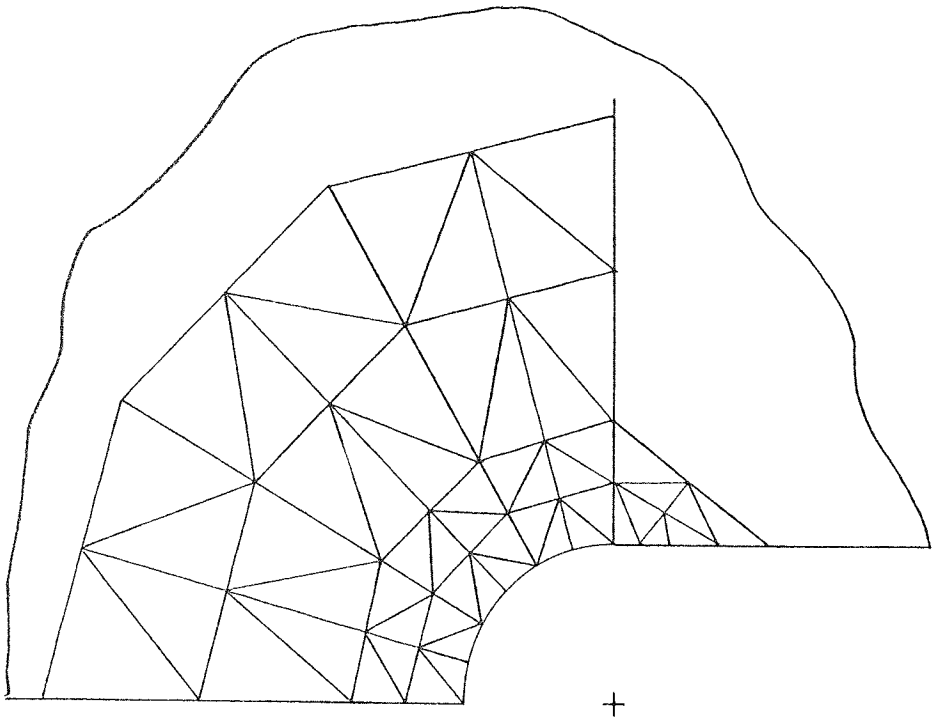


FIG a

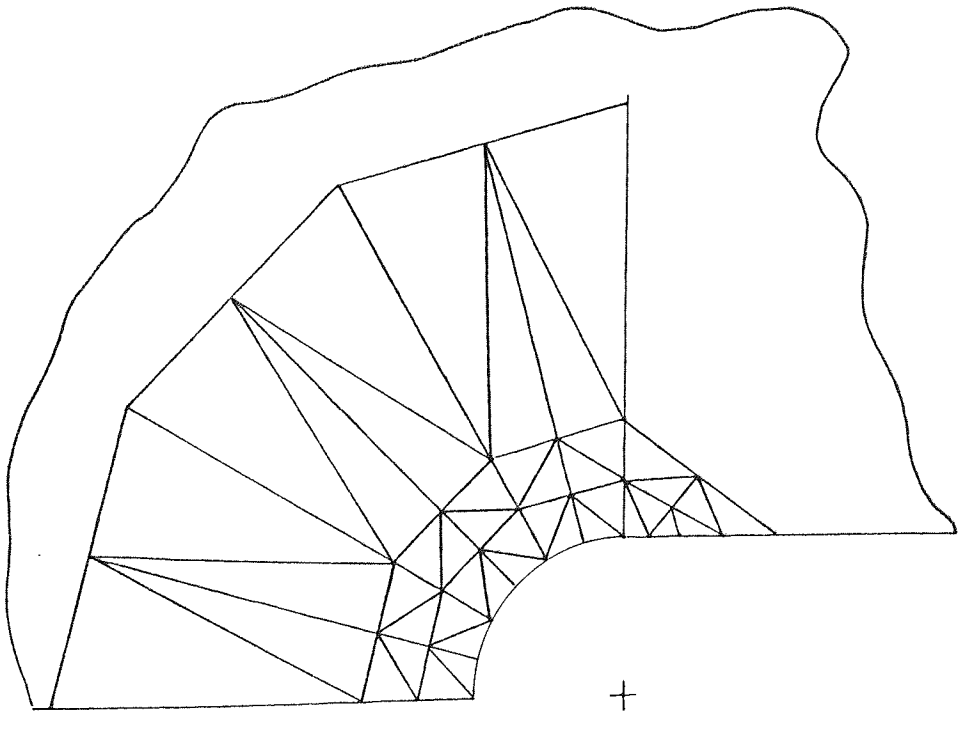


FIG b

FIG 5.4. Alternative mesh types to that shown in figure 5.3.
a, Gradual element size increase, resulting in a high number of nodes and wasted computer storage space.
b, A reduced element density at the cost of poor element shapes.

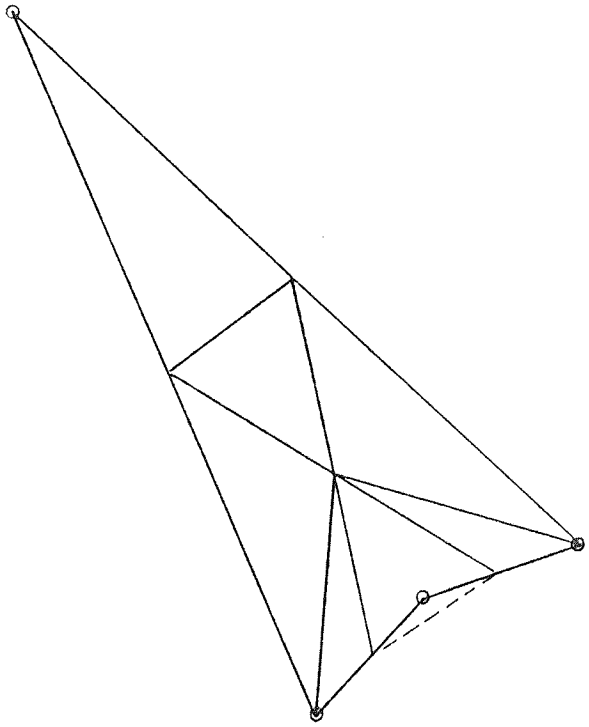


FIG 5.5. A 'stretched' Distribution Zone showing how the element is mapped outside the zone's boundary.

In cases where the mesh is regularly spaced around a circular hole or re-entrant corner, the zone array can be considered to be of a standard form. As these standard geometries occur regularly in practice, it was found helpful to recognise them as such and develop a routine which would generate the input data from a few relevant parameters. Thus the data input is further reduced. Two routines were developed, COREGEN which generates the input data for a local crack tip mesh, and STDGEN which can be used in general situations. Both procedures operate in the same manner and the relevant operation notes can be found in the Appendix (9.3). Some examples, notably the local crack tip mesh, are given which indicate the areas where these procedures can be applied. Section (2.5.2) has dealt with the relatively new 'transition' elements, which are constructed so as to respond to the crack tip singularity. The procedure COREGEN has been designed automatically to construct this element type in a standard geometry. Refer to Chapter Six and the Appendix (9.3) for further details.

5.4 THE PROGRAM STRUCTURE

The objectives in designing a flexible mesh generation computer program, apart from minimizing the data input are:-

1. A facility for grading the mesh so as to achieve required accuracy of idealization.
2. A nodal numbering system which results in the best computational efficiency.
3. A facility for describing areas of different materials whenever these occur.
4. An achievement of element shapes which are of reasonable form and do not lead to numerical ill-conditioning.

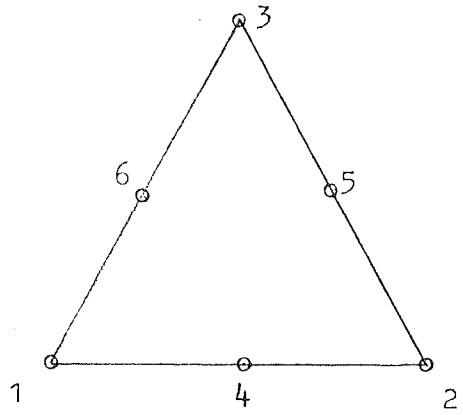
These points will be enlarged upon, as the program structure evolves.

The program has been developed to cater for a six-node triangular element and an eight-node quadrilateral element. Because of the two different element shapes, some of the program procedures had to be split. Essesntially the procedures perform the same operations and are prefixed by TR or Q, signifying a triangular or quadrilateral routine respectively. Figure (5.6) illustrates the two element types.

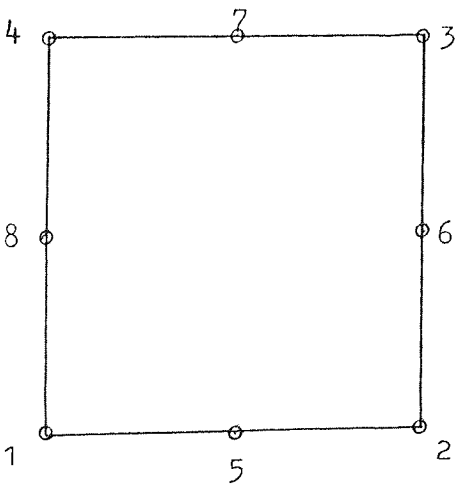
The program operates by considering each zone in turn, starting from the left-hand corner of the zone array, and moving vertically through a column and then from column to column. The following steps indicate how the program is organised:-

1. The zone array data is read.
2. The eight super node co-ordinates for the first zone, are determined, either from the input data or by direct interpolation.
3. From the zone's specified subdivision, the element data is generated, determining the node co-ordinates.
4. The element nodal connections are found using the same information as in step 3.
5. Steps 2 - 4 are repeated for the remaining zones, until the whole array has been scanned.
6. The finite element data is stored in a specified file and the element mesh is plotted.

The generated data can be used in several finite element programs and each program has a 'code' ensuring that the output format is correct. Following the flowchart, for the program, it can be seen that the first procedure encountered is OUTPUT1. This procedure, as its name suggests,



Isoparametric
Triangular Element.



Isoparametric
Quadrilateral
Element.

FIG 5.6. The Mesh Generation Program outputs data for either of the above element types.

simply prints out the control variables required by the coded program, such as the number of elements and nodes in the finite element mesh.

The data needed to generate the nodal co-ordinates and element nodal connections is read in by procedure MGINPUT. The zone array can be thought of as a chequerboard, each rectangle being a zone. The zones are dealt with in turn, starting in the left-hand column and working vertically, then progressing from column to column. Two loops are established which control this operation, namely ZN and ZONE.

From the previous section it will be realised that each zone has eight nodal points which need to be completely identified. It is, however unnecessary to input the co-ordinates for the mid-side nodes for a zone which has straight sides, as these can be obtained by interpolation. Therefore the next procedure to be accessed is ZONEXY, which performs this particular function. The eight super-node co-ordinates are established by using an identifying code. For further information on this procedure refer to the Appendix (9.3).

The next procedure to be accessed is TR/QELETXY, which determines the finite element nodal co-ordinates. Each set of nodal co-ordinates generated is allocated a node number which is predetermined by the overall numbering scheme. Nodal numbering begins from the lower left-hand corner of the zone array and hence ξ and η are assigned to be -1 on entering TR/QELETXY. With the input data each zone is given a sub-division parameter, DIVX and DIVY. The local co-ordinates can be specified using this information and the corresponding global cartesian co-ordinates generated, through the mapping function $N(\xi,\eta)$.

As previously mentioned, zones within the array may be declared redundant or void, that is the zones are not used in the construction of the finite element mesh. This facility complicates the element nodal numbering scheme, which has to accommodate any configuration of zones which may arise. The numbering organisation is similar in both element types and the scheme, used in constructing the triangular elements, will be examined here. The zone array is inspected within the nested loops ZN and ZONE, as previously stated, and the variables which combine to make up the numbering system, monitor each zone as it is met.

Consider a single column of three zones, as in figure (5.7). As previously stated, the program operates on each zone in turn, generating the element nodal coordinates and element nodal connections. In the present example zone 1 would be operated on first, generating nodal co-ordinates 1-3, 8-10 and 15-17. In order to achieve this an expression must be available which can define all the node numbers within the zone under consideration. This expression is developed through a set of controlling variables, which react with the input data as the program is operating. The main control variables are,

P, A, ADD, DIFF, ZNAD, VD, VD1 and IN,

their significance will be shown through examples. Knowing the zone subdivision, it is possible to construct a nested loop system which can be used to determine the element node numbers. The inner loop, denoted by J, is used to increment the vertical element node numbering, hence its limits are dependent on the zone's element subdivision. The outer loop, denoted by I, is used to traverse the zone horizontally,

again using the specified zone subdivision. It will be noticed, from figure (5.7), that the element node numbering of the first zone increases from 3 to 8 after the first cycle of the inner loop, thus, some means of allowing for the influence of the zones above the zone under consideration is necessary. This can be achieved using variable A, which represents the number of element nodes on the edge of the column of zones under consideration. Using these variables it is possible to define the element nodes within the first zone, thus

```
FOR I = 1 TO DIVX(ZONE)*2+1
FOR J = 1 TO DIVY(ZONE)*2+1
    P = A*(I-1)+J
```

where P represents the element node number, DIVX and DIVY represent the horizontal and vertical zone subdivision, and ZONE is the zone number. As the zone element subdivision is unity in both directions, the limit of index integers I and J is three. From figure (5.7), integer A = 7 and therefore indexing J gives element node numbers 1-3. Incrementing I by 1 and repeating the process yields node numbers 8-10. Similarly nodes 15-17 are obtained on completing loops I and J.

The above expression, however, does not function for zones higher in the column and an extra parameter ADD is required to enable the expression to be generally applicable to the column of zones. The integer ADD acts as a spacing device and is dependent on the lower zone's subdivision. The expression becomes,

$$P = A*(I-1)+J+ADD \qquad (5.7)$$

where the loops I and J have been omitted.

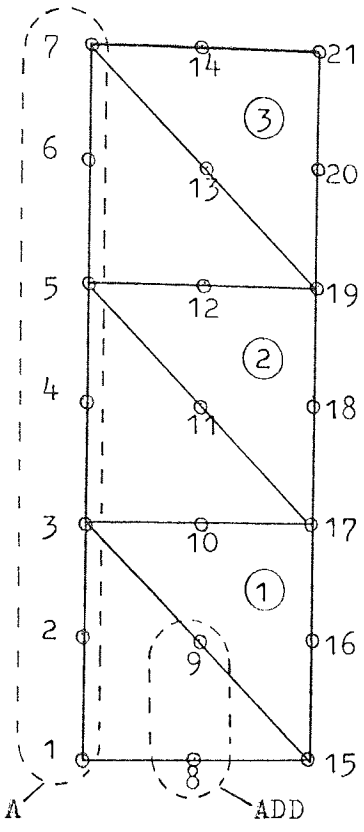


FIG 5.7 A column of zones sub-divided into triangular elements. Note the zone numbering system.

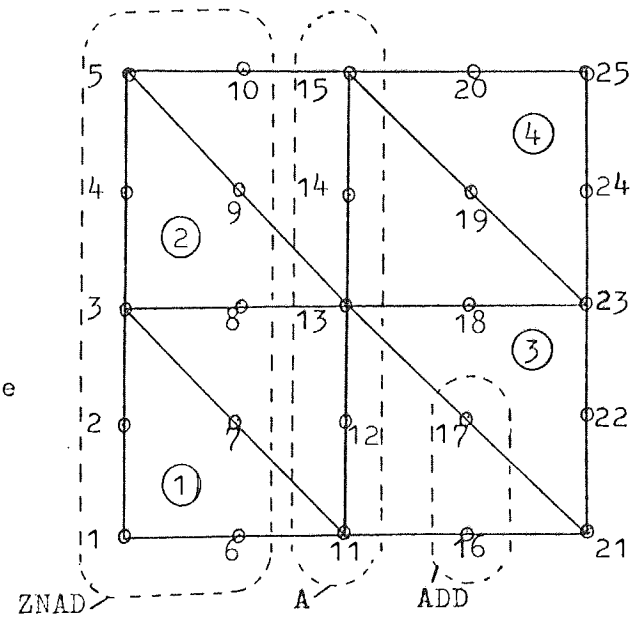


FIG 5.8 Zone array. The number of nodes within the dashed areas indicates the magnitude of the variables.

The same process is repeated, as previously explained, however, in the second zone of figure (5.7), ADD = 2, thus, instead of generating element node numbers 1-3 the process yields node numbers 3-5, etc. In zone three, ADD = 4, hence node numbers 5-7 are generated during the first cycle of counter J. It is possible to infer the value of integer ADD from the figure (5.7).

Extending expression (5.7) to generate the element node numbers within a zone other than in the first column of the zone array, requires a record to be made of the total number of nodes generated in the previous column of zones, and this is given by integer ZNAD. The expression now becomes,

$$P = A*(I-1)+J+ADD+ZNAD \qquad (5.8)$$

where the nested loops I and J have been omitted. Using this expression in example two, figure (5.8), for zone 3, we have,

$$ZNAD = 10,\ A = 5,\ ADD = 0$$

and repeating the same steps as before, we obtain element node numbers 11-13, 16-18 and 21-23.

To account for the arbitrary use of voids in the zone array, the expression (5.8) becomes more complicated and dependent on a set of conditional statements. The extra variables now encompassed by the numbering scheme are shown in figure (5.9), and their values can be interpreted from the number of nodes within the dashed areas. Zones which are not required in the construction of the distorted zone arary, need not be declared, and are considered to be void. The material

number zero is automatically attached to these void zones and can thus be dealt with appropriately. Consider the second column of zones, where zone 5 is void; in this case variable A = 6 and a new parameter DIFF is used to represent the number of nodes on the left-hand edge of the void, not including the end nodes. The final expression is given below with the corresponding conditional statements,

```
FOR I = 1 TO DIVX(ZONE)*2+1
FOR J = 1 TO DIVY(ZONE)*2+1
IF I = 1 THEN C = VD; ELSE
IF I = DIVX(ZONE)*2+1 THEN
   C = DIFF + VD1; ELSE C = DIFF + IN;
P: = A*(I-1)+J+ADD+ZNAD+C                                   (5.9)
```

The variables VD, VD1 and IN are defined as

```
VD = DIVY(ZONE-VZONE)*2
VD1 = DIVY(ZONE+VZONE)*2
IN = 1,
```

and are only evaluated after a void has been detected. The integer VZONE represents the vertical zone subdivision of the zone array. Consider zone 6 of figure (5.9), evaluating the variables, we have,

```
ADD = VD = VD1 = 2,
IN = 1, ZNAD = 14, A = 6,
and DIFF = 1.
```

Now utilizing expression (5.9) the element node numbers are generated, thus,

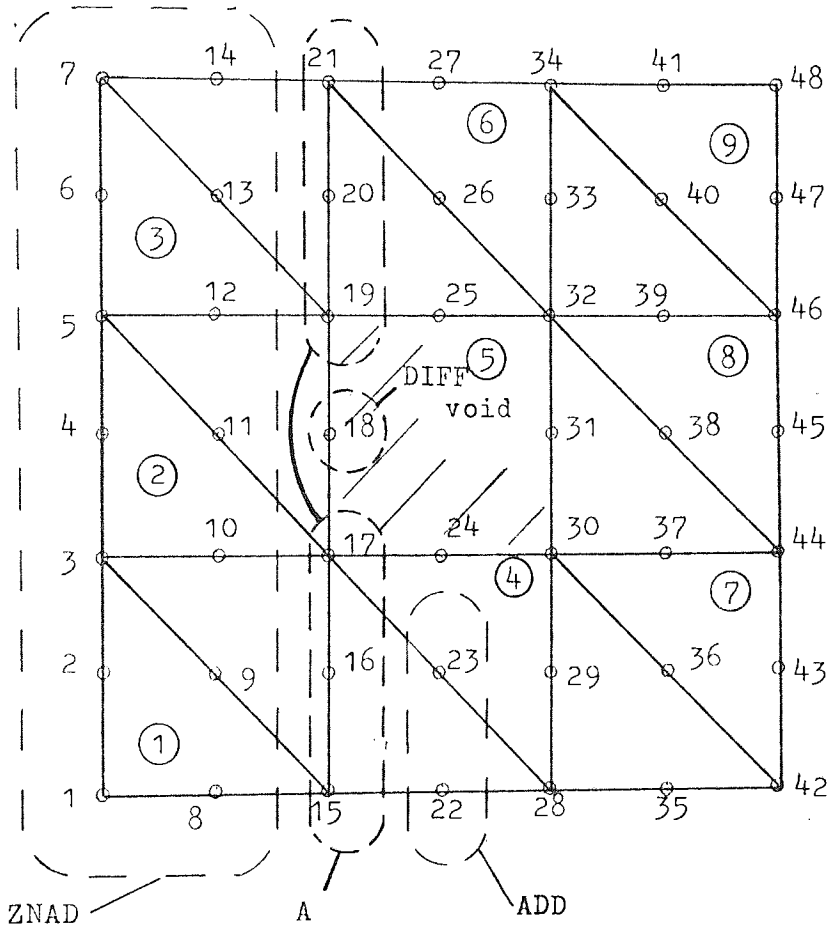


FIG 5.9. Zone array showing an enclosed void. The number of nodes within the balloons represents the magnitude of the variables.

for I = 1, P = 6*0+J+2+14+2

P = 19 - 21.

for I = 2, P = 6*1+J+2+14+2

P = 25 - 27

for I = 3 P = 6*2+J+2+14+2

P = 32 - 34.

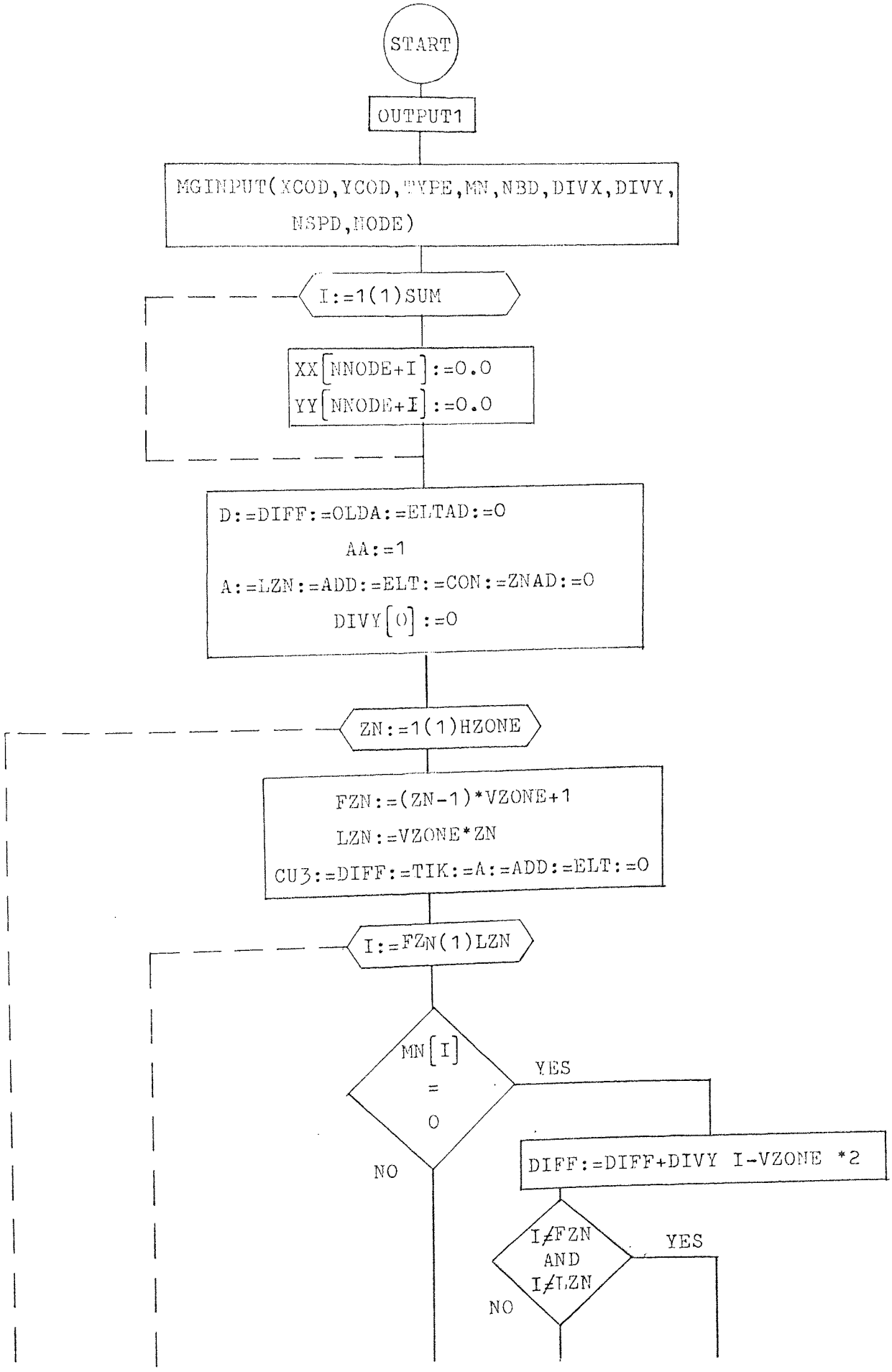
Having clarified the operation of the element numbering scheme, we can return to the program flowchart where the next procedure to be accessed is TR/QELENCONS. This derives the element nodal connections, using the numbering scheme and operating on each element, within the zone, in turn.

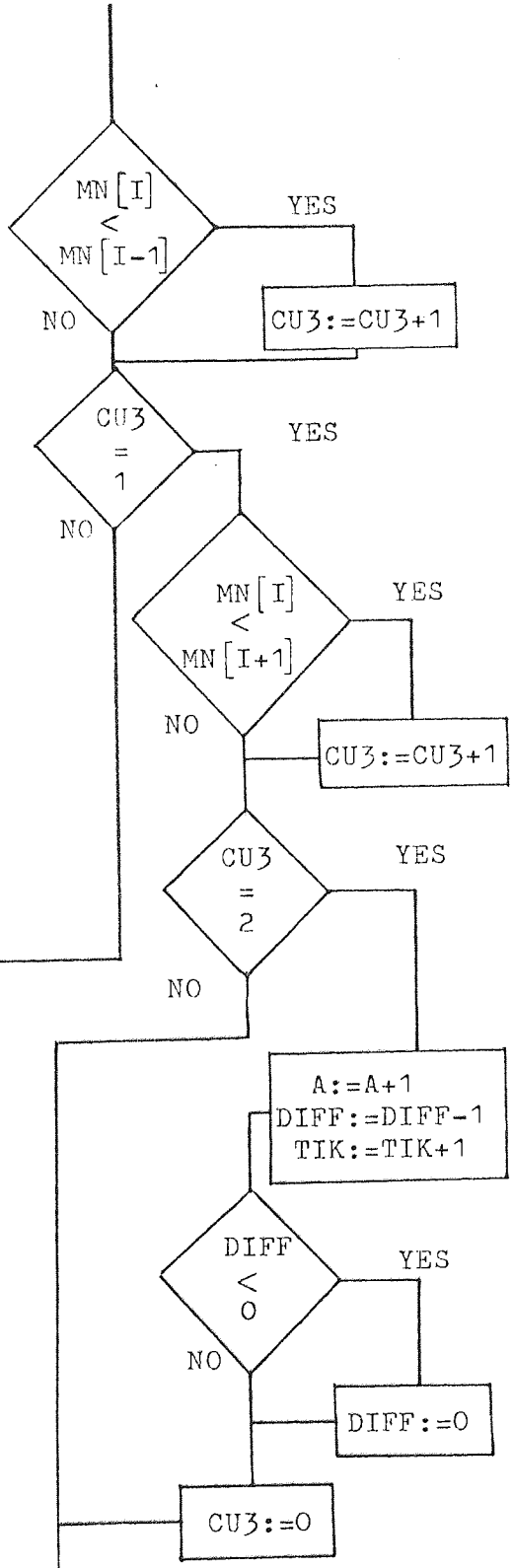
Zones can be joined, usually to effect a better element distribution. This is achieved by recording the nodes to be retained, on the joining zone faces, via procedure TR/QIDN, and subsequently refining and condensing the nodal co-ordinates and element nodal connections using procedures COINCID and HATCHET.

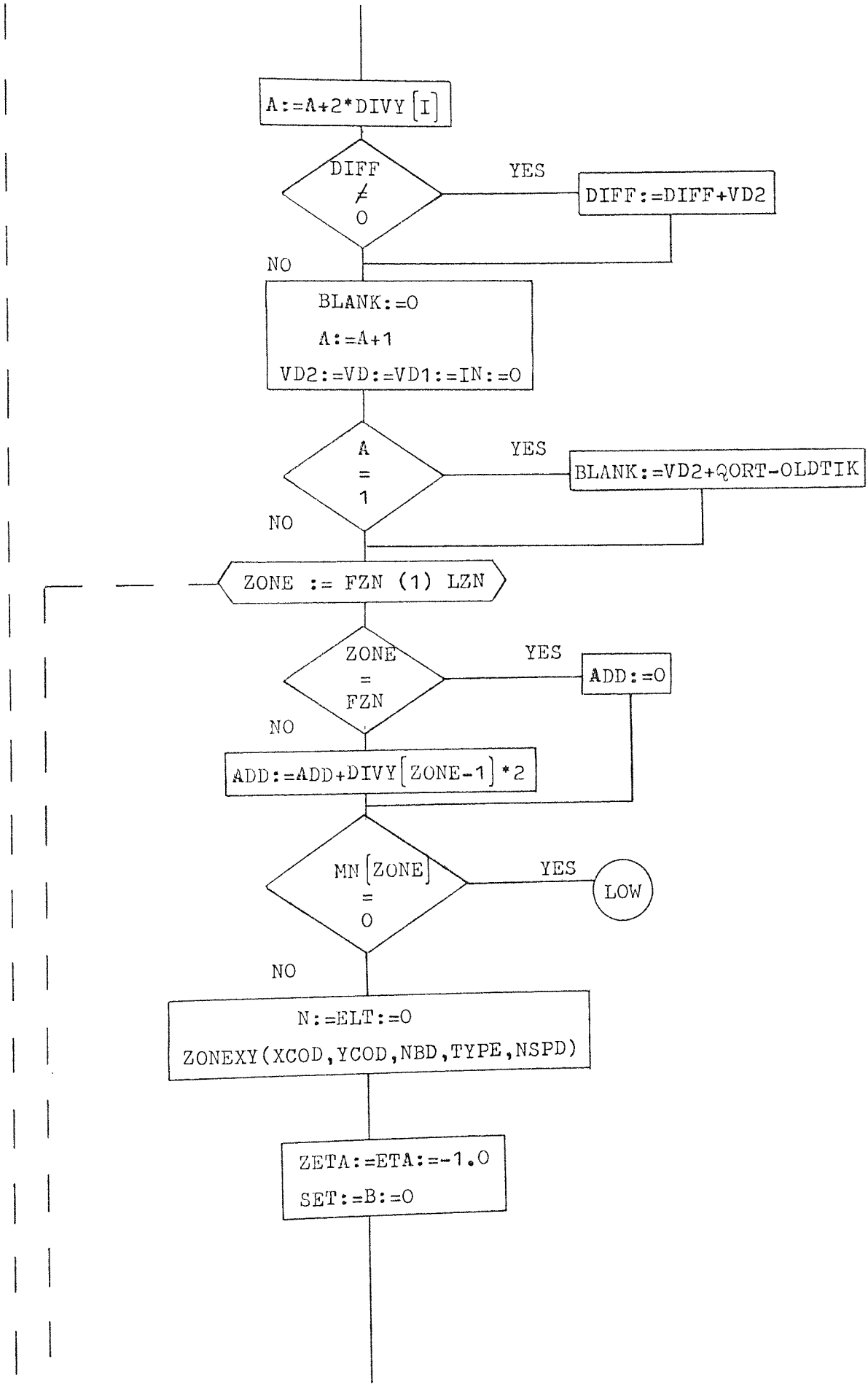
The final procedure, RESPLOT, plots the discretised component using the generated data. Each element is plotted individually, hence any errors in the nodal connections or co-ordinates can be traced visually.

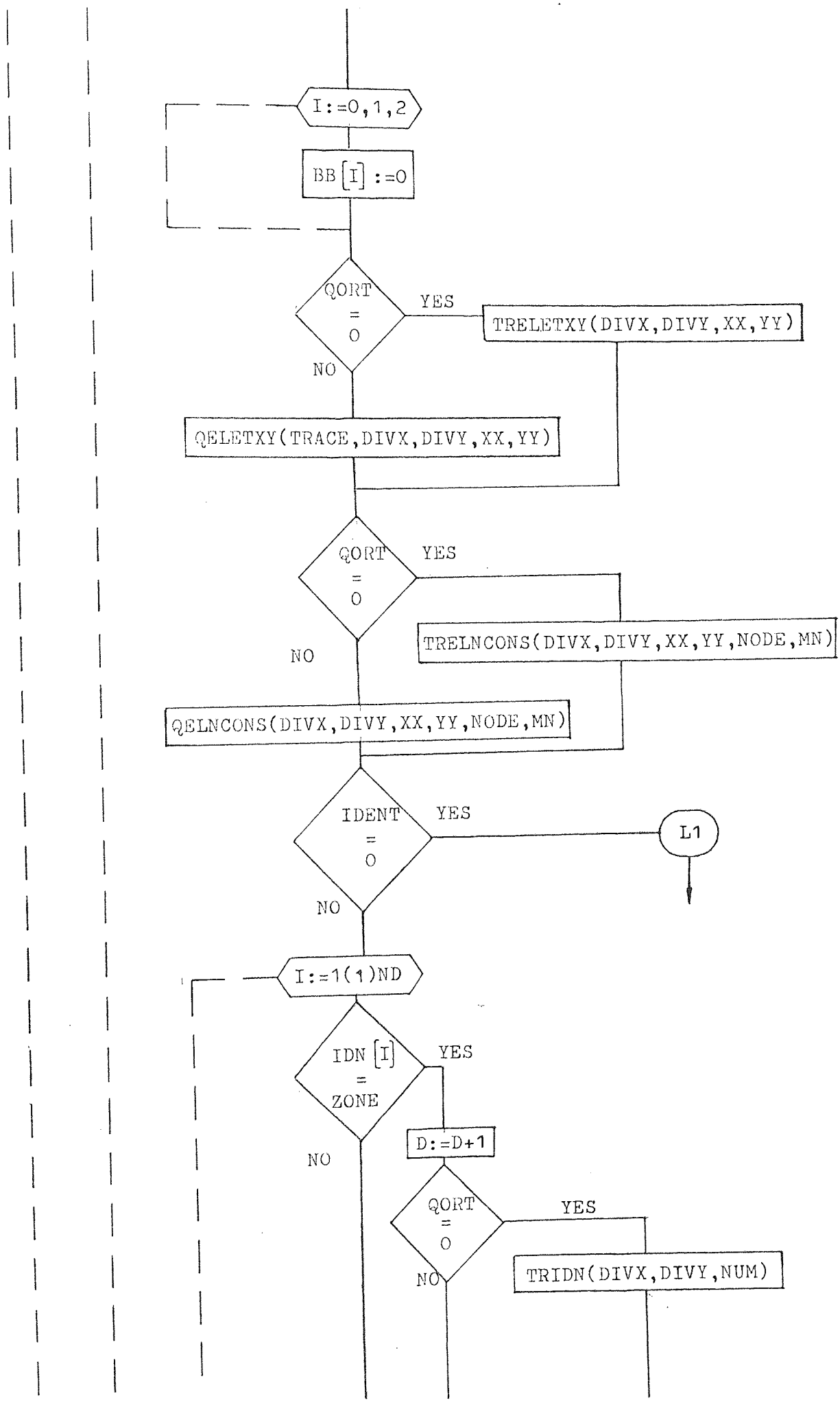
The sub-routine or procedures referred to in this section can be found together with a detailed description, in the Appendix (9.3).

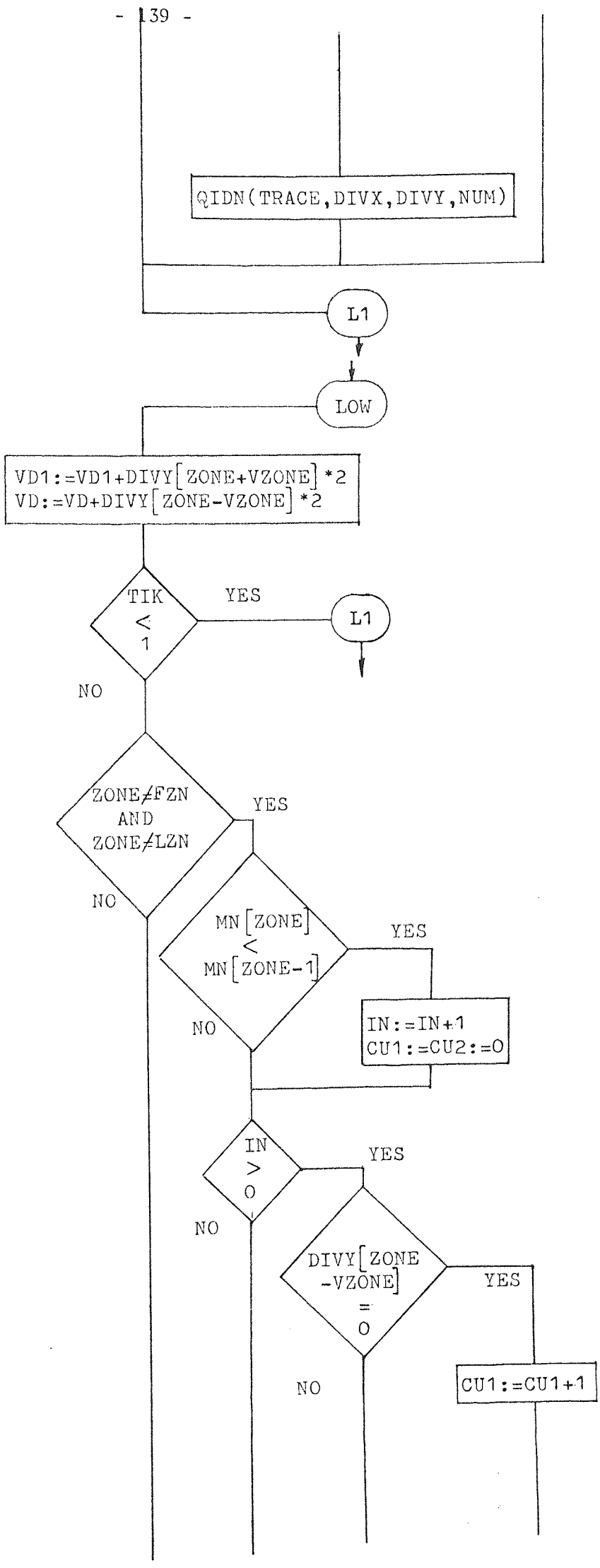
Flowchart for Mesh Generation Program :- STDAMG

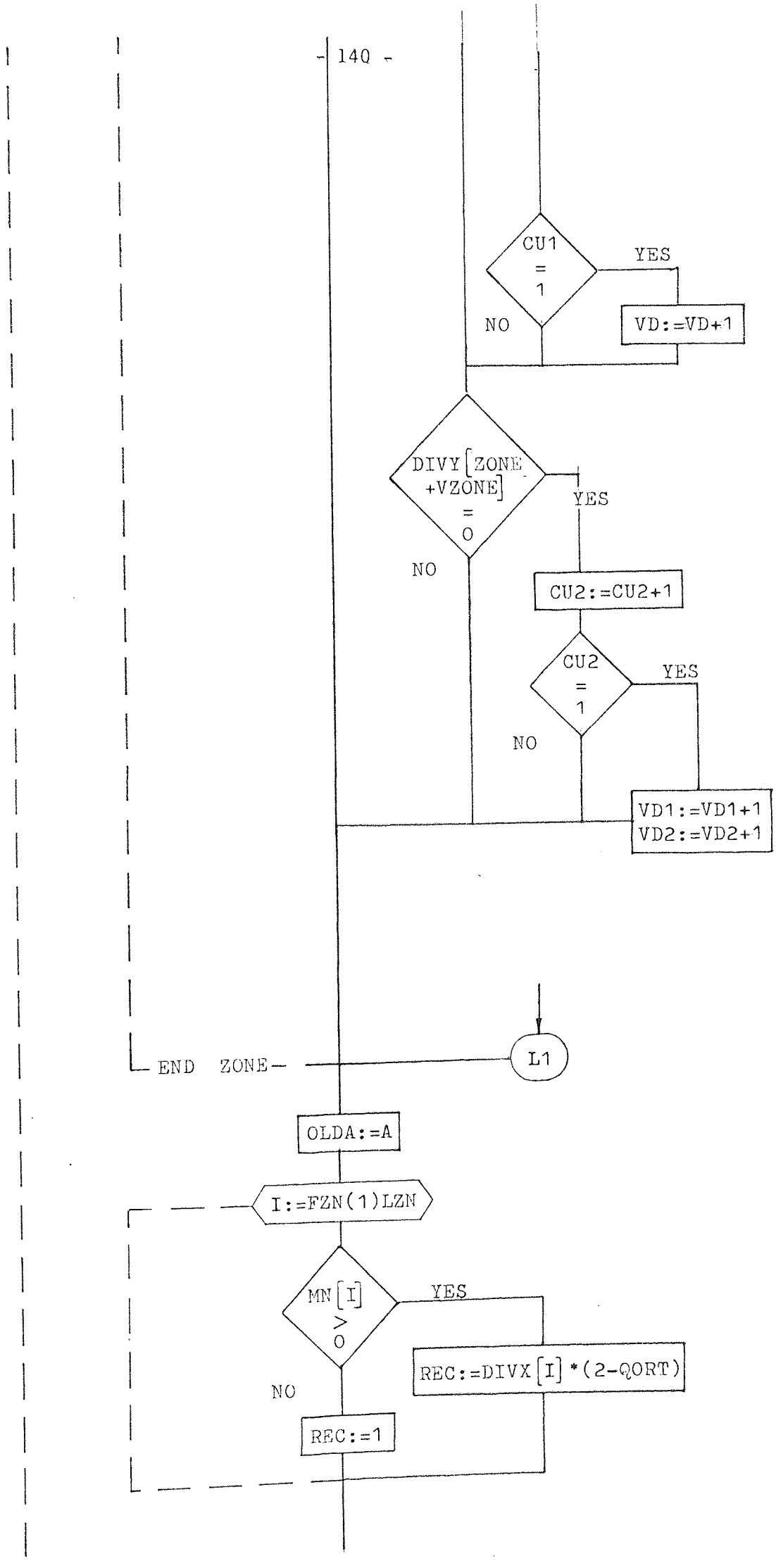


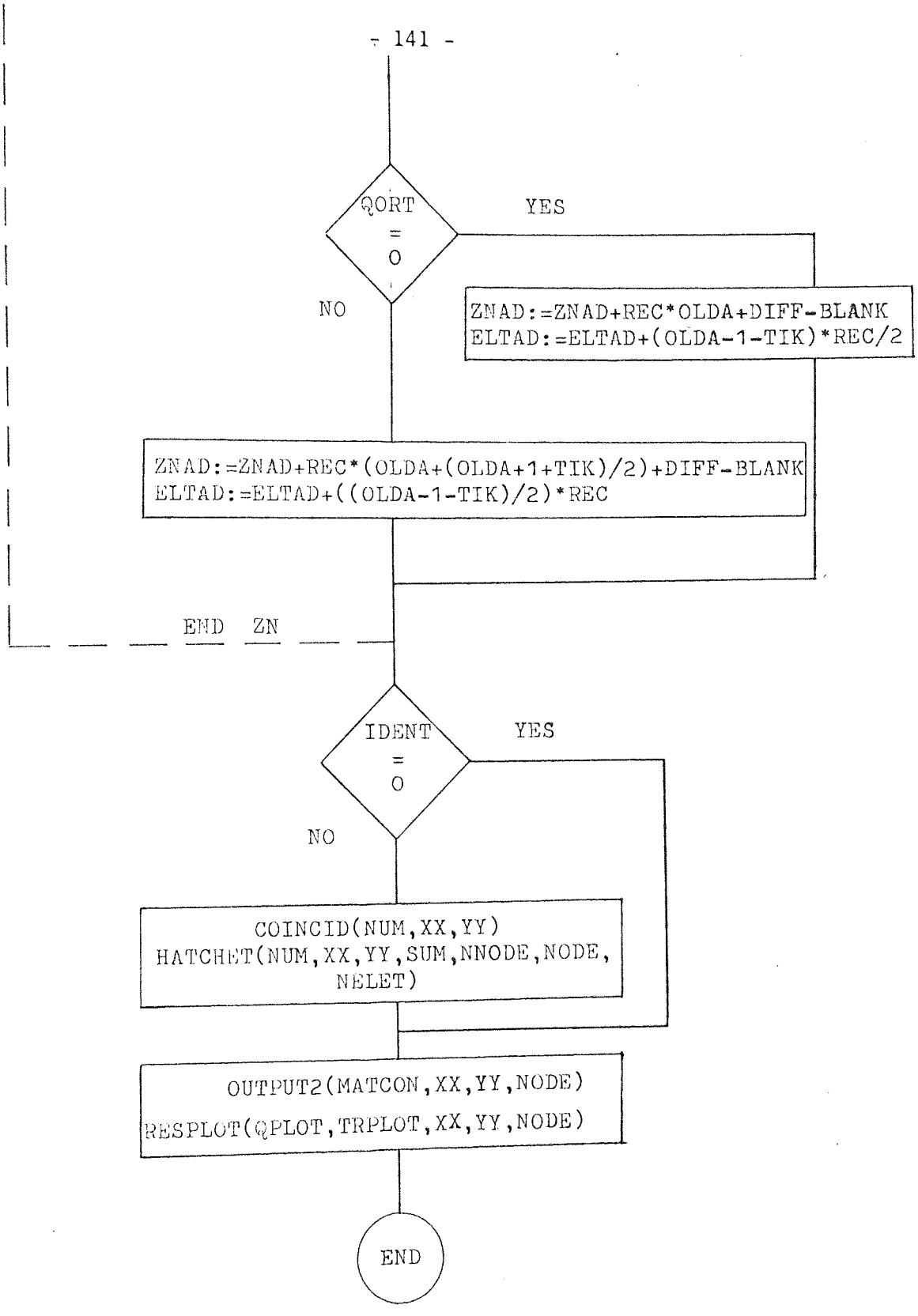

MN[I] < MN[I-1]
YES
NO
CU3:=CU3+1
CU3 = 1
YES
NO
MN[I] < MN[I+1]
YES
NO
CU3:=CU3+1
CU3 = 2
YES
NO
A:=A+1
DIFF:=DIFF-1
TIK:=TIK+1
DIFF < 0
YES
NO
DIFF:=0
CU3:=0


A:=A+2*DIVY[I]
DIFF ≠ 0
YES
DIFF:=DIFF+VD2
NO
BLANK:=0
A:=A+1
VD2:=VD:=VD1:=IN:=0
A = 1
YES
BLANK:=VD2+QORT-OLDTIK
NO
ZONE := FZN (1) LZN
ZONE = FZN
YES
ADD:=0
NO
ADD:=ADD+DIVY[ZONE-1]*2
MN[ZONE] = 0
YES
LOW
NO
N:=ELT:=0
ZONEXY(XCOD,YCOD,NBD,TYPE,NSPD)
ZETA:=ETA:=-1.0
SET:=B:=0


I:=0,1,2
BB[I]:=0
QORT = 0
YES
TRELETXY(DIVX,DIVY,XX,YY)
NO
QELETXY(TRACE,DIVX,DIVY,XX,YY)
QORT = 0
YES
TRELNCONS(DIVX,DIVY,XX,YY,NODE,MN)
NO
QELNCONS(DIVX,DIVY,XX,YY,NODE,MN)
IDENT = 0
YES
L1
NO
I:=1(1)ND
IDN[I] = ZONE
YES
D:=D+1
NO
QORT = 0
YES
TRIDN(DIVX,DIVY,NUM)
NO


QIDN(TRACE,DIVX,DIVY,NUM)
L1
LOW
VD1:=VD1+DIVY[ZONE+VZONE]*2
VD:=VD+DIVY[ZONE-VZONE]*2
TIK < 1
YES
L1
NO
ZONE≠FZN AND ZONE≠LZN
YES
NO
MN[ZONE] < MN[ZONE-1]
YES
NO
IN:=IN+1
CU1:=CU2:=0
IN > 0
YES
NO
DIVY[ZONE -VZONE] = 0
YES
NO
CU1:=CU1+1


CU1 = 1
YES
NO
VD:=VD+1
DIVY[ZONE +VZONE] = 0
YES
NO
CU2:=CU2+1
CU2 = 1
YES
NO
VD1:=VD1+1
VD2:=VD2+1
END ZONE
L1
OLDA:=A
I:=FZN(1)LZN
MN[I] > 0
YES
NO
REC:=DIVX[I]*(2-QORT)
REC:=1


QORT
=
0
YES
NO
ZNAD:=ZNAD+REC*OLDA+DIFF-BLANK
ELTAD:=ELTAD+(OLDA-1-TIK)*REC/2
ZNAD:=ZNAD+REC*(OLDA+(OLDA+1+TIK)/2)+DIFF-BLANK
ELTAD:=ELTAD+((OLDA-1-TIK)/2)*REC
END ZN
IDENT
=
0
YES
NO
COINCID(NUM,XX,YY)
HATCHET(NUM,XX,YY,SUM,NNODE,NODE,
NELET)
OUTPUT2(MATCON,XX,YY,NODE)
RESPLOT(QPLOT,TRPLOT,XX,YY,NODE)
END

5.5 EXAMPLES AND USERS GUIDE

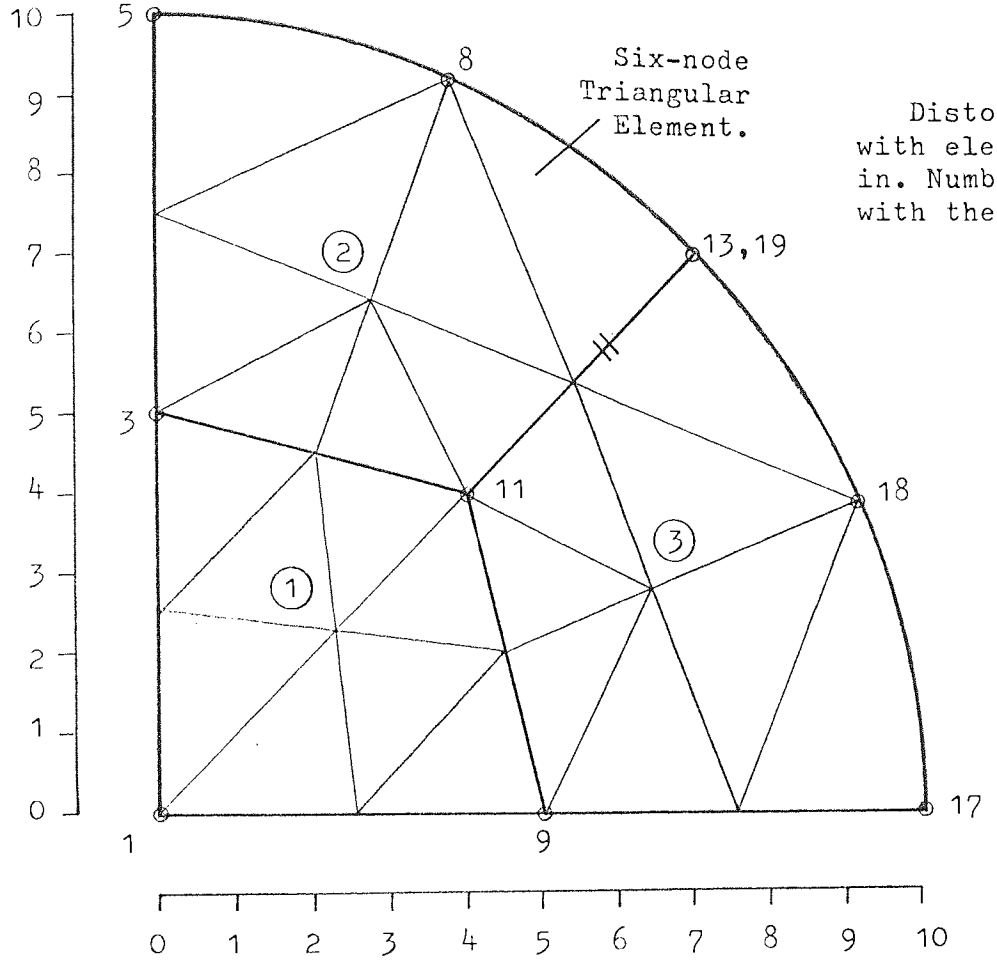
Some examples are given here to guide the user and illustrate the majority of the program's facilities. Extensive notes have been added to the users guide, at the end of this section, together with a glossary of terms. To reduce data input the super-node numbers are pre-determined from the grid dimensions of the zone array. Each zone can be automatically defined by a series of super-node numbers, i.e. zone connections, alleviating the need to specify them in the data input. Example 1 , figure (5.10) is that of a disc in compression: from symmetry only a quarter of the plate need be considered. In this case the six-noded triangular element has been employed. As stated earlier each zone is defined by eight super-nodes and the regular numbering sequence of the zone array is clearly illustrated. Note that the fourth zone is not required and is redundant. The 'zone' numbering scheme does not recognise voids, as this would make the program over complicated and simply numbers the entire grid. See procedure ZONEXY. From the diagram, each zone is defined by a minimum of four super-nodes, but two further mid-side nodes 8, 18 have been used so that the zones can be made to fit the curved boundary of the disc. To achieve a better element distribution zone four has been made void and zone faces 11-13 and 11-19 are joined. The node numbers on zone face 11-13 have been retained and this is clearly shown in the element array. Refer to procedures COINCID and HATCHET. The following data sequence in Table (5.11) corresponds to example .1. and can be compared with the users guide information.

A recessed bar under compression is considered in the second example and the idealized structure is given in figure (5.12). Eight-node

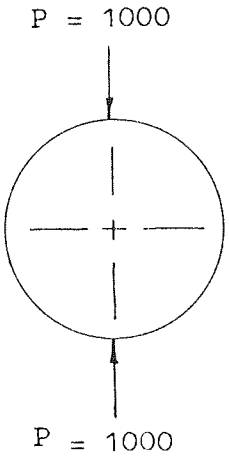
quadrilateral elements are used in this case and because the elements around the recess, form a regular mesh, the routine STDGEN can be used to generate the input data. Zones 1, 2 and 3 can be treated in this manner and a string of parameters, shown in Table (5.13), fully describe the three zones. As a result the zone array can be assembled as if these three zones were already specified, therefore only the super-nodes shown in the discretized figure need be declared. The data sequence is given in Table (5.13), note that the uniform loading on the ends of the bar follows a 1,4,1 distribution over the element.

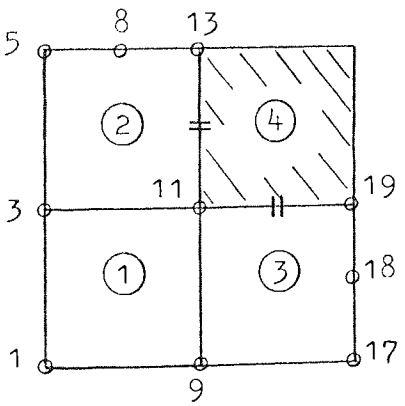
The third example depicts a plate in uniform tension containing a 90 degree edge crack. From symmetry only half the plate need be considered and the element subdivision is shown in figure (5.14). As this problem involves mode I fracture, the data format is adjusted appropriately using CODE = 2, i.e. program PCQTM1ST. From the diagram it will be seen that zones 9 and 12 are not used in the element construction, but they are still represented in the grid numbering scheme. Zones which are not declared in the data input are assumed to be void, except where the zones are automatically generated through procedures COREGEN and STDGEN. In this example the procedure COREGEN has been used to set up the special 'transition' elements around the core element. Refer to section (2.5.2). Zones 1 to 4 have been generated in this fashion and a set of parameters defining the shape of the local mesh around the crack tip are entered, as in Table (5.15). In order for the singularity core element to be formulated correctly the nodes on its boundary must be entered in an anti-clockwise direction, beginning with node one. Again the data input can be compared with the users guide.

Example (1) Disc in compression.



Distorted zone array with elements finely drawn in. Numbering corresponds with the super-nodes.





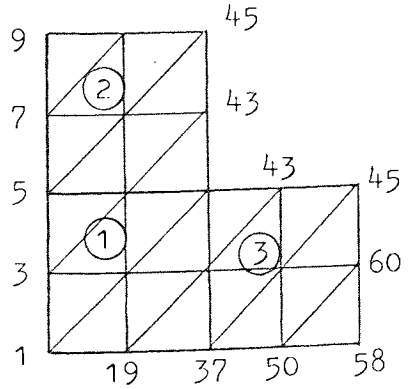
Mat'l cons:- $E = 3.0*10^7$

$v = 0.3$

$G = 1.2*10^7$

Zone array with super-nodes in position. Note closing faces ‖ and numbering scheme is continuous even with the void present.

◯ - Zone numbers.



Element array showing nodes used in the boundary conditions. Nb. Alteration in nodal pattern due to the closing faces.

NNODES =(9*5+5*4)-4 = 61.

NELEMT = 24.

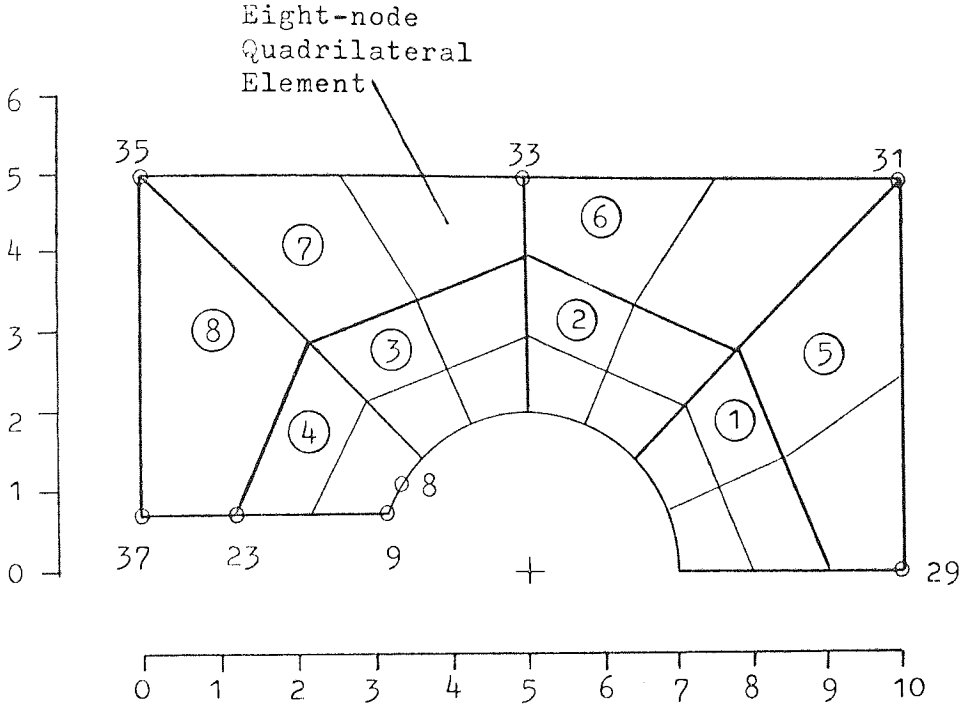
FIG 5.10.

TABLE 5.11. Data for example (1) a disc in compression. The data sequence can be followed from the users guide.

```
1,0,1
DISC IN COMPRESSION, T=1.0  END OF TITLE
24,61,1,3,0,0,1,1,1
9,3,2,2,1,0,0
1,0.0,0.0,1          1,0.0,5.0,3
1,0.0,10.0,5         1,3.82683,9.2388,8
1,5.0,0.0,9          1,4.0,4.0,11
2,7.07107,7.0717,13,19     1,10.0,0.0,17
1,9.2388,3.82683,18
3,1,2,2,1,2,3
1
2,3
0
17
1,3,0.0,0.0     2,1,0.0,0.0     3,1,0.0,0.0
4,1,0.0,0.0     5,1,0.0,0.0     6,1,0.0,0.0
7,1,0.0,0.0     8,1,0.0,0.0     9,1,0.0,-1.0&3
10,2,0.0,0.0    19,2,0.0,0.0    28,2,0.0,0.0
37,2,0.0,0.0    46,2,0.0,0.0    50,2,0.0,0.0
54,2,0.0,0.0    58,2,0.0,0.0
0,0.0,1.0
3.0&7,0.3,1.2&7,3.0&7,0.3
```

Mesh Generation data.

FIG 5.12. Example (2) A recessed bar in compression.



Distorted zone array showing super-node numbering.

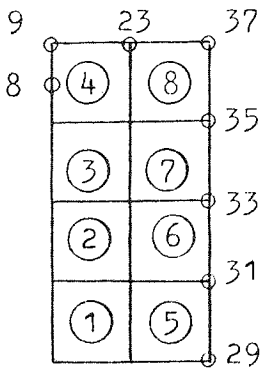
Mat'l cons:-

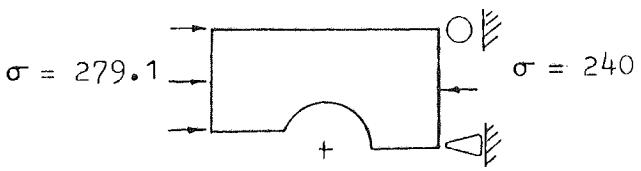
$E = 3.0*10^7$

$v = 0.3$

$G = 1.2*10^7$

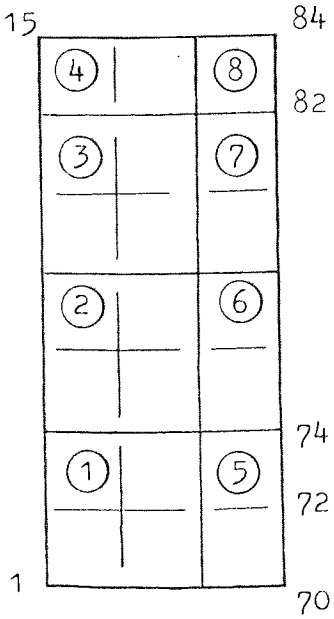
Compression load = 1200





Zones 1,2 & 3 will be generated using the sub-procedure STDGEN.

◯ - Zone numbering



Element array. Note the different numbering sequence compared with ex'l (1). This is due to the different element type.

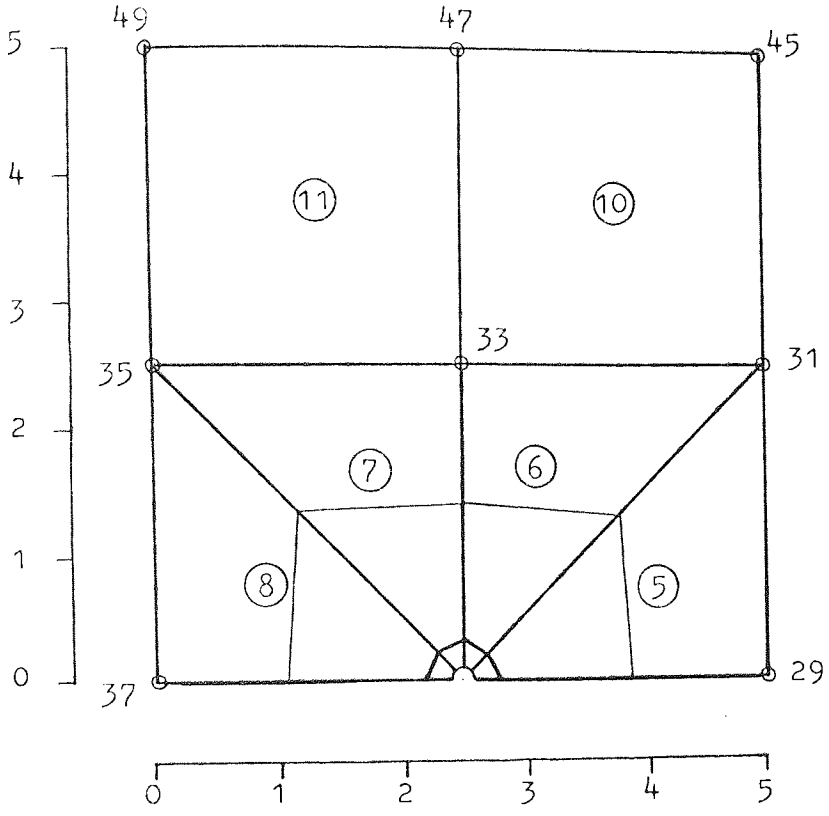
NNODES = 15*4 + 8*3 = 84.

NELEMT = 21.

TABLE 5.13. Example (2) A recessed bar in compression. Routine STDGEN is used in this example to form the mesh around the semi-circle and also reduce data input.

```
1,1,1
RECESSED BAR IN COMPRESSION
W=5, L=10, P=1000    END OF TITLE
21,84,1,3,0,0,1,1,1  ----------------------------------
8,5,4,2,1,0
1
1,1,7,5.0,0.0,2.0,3.0,4.0,0.0,22.50,2,2
1,3.1265,0.7,9       1,3.3178,1.0818,8          Mesh Generation
1,1.2,0.7,23         1,0.0,0.7,37                  data.
1,10.0,0.0,29        1,10.0,5.0,31
1,5.0,5.0,33         1,0.0,5.0,35
1,1,2,1,4
3,1,1,2,5,6,7
1,1,1,1,8
0                    ----------------------------------
8
74,1,0.0,0.0         73,0,-400.0,0.0
72,0,-200.0,0.0      71,0,-400.0,0.0
70,3,0.0,0.0         82,0,200.0,0.0
83,0,800.0,0.0       84,0,200.0,0.0
0,0.0,1.0
3.0&7,0.3,1.2&7,3.0&7,0.3
```

FIG 5.14. Example (3) Plate in uniform tension containing a 90 degree edge crack.



The super-node numbering corresponds with that of the zone array.

Core radius = Rc = a/30

Rc = 0.083

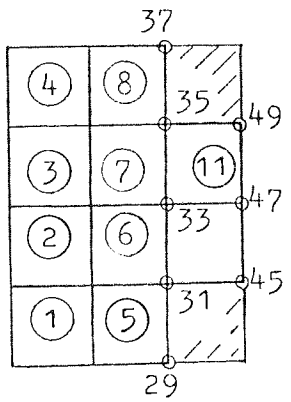
a = 2.5

Mat'l cons :-

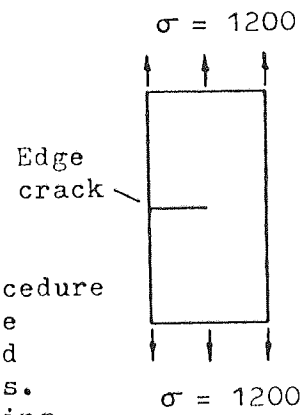
E = $3.0*10^7$

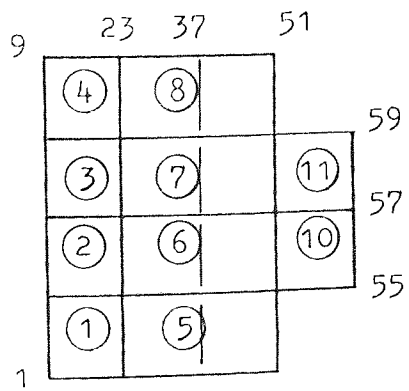
v = 0.3

G = $1.2*10^7$



Zones 1,2,3 & 4 are generated by sub-procedure COREGEN to obtain the correctly constructed 'transition' elements. Note that the numbering sequence is maintained despite the 'void' zones.





Quadrilateral elements are used in this example.

NNODES = 9*4 + 5*3 +8 =59.

NELEMT = 14.

◯ - Zone numbering

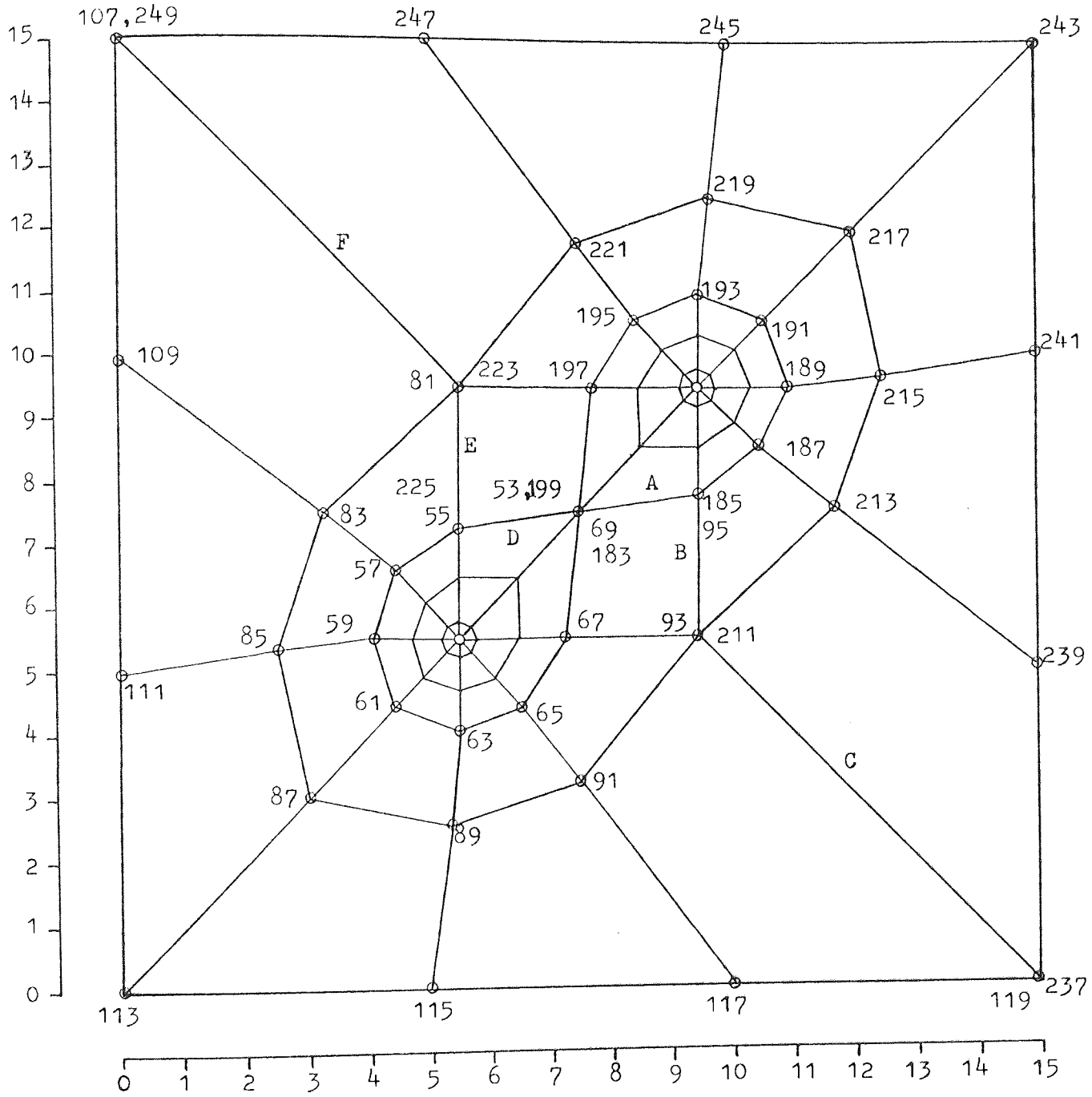
TABLE 5.15. Data for example (3). Plate in uniform tension with a 90 degree edge crack.

```
2,1,1
TESTING MODE 1 PROG PCQTM1ST. RC = 0.083  W=2H= 5.0
END OF TITLE
14,59,1.0,1,1
8,6,4,3,1,1
1,1,9,2.5,0.0,0.083,0.0,22.5,1,1
0
1,5.0,0.0,29      1,5.0,2.5,31
1,2.5,2.5,33      1,0.0,2.5,35
1,0.0,0.0,37      1,5.0,5.0,45
1,2.5,5.0,47      1,0.0,5.0,49
4,1,2,1,5,6,7,8
2,1,1,1,10,11
0
12
1,2,0.0,0.0       55,0,0.0,5.0&3
10,2,0.0,0.0      56,0,0.0,2.0&4
15,2,0.0,0.0      57,0,0.0,1.0&4
24,2,0.0,0.0      58,0,0.0,2.0&4
29,2,0.0,0.0      59,0,0.0,5.0&3
38,2,0.0,0.0      43,3,0.0,0.0
1
3.0&7,0.3,1.2&7,3.0&7,0.3
1
```

Mesh Generation data.

The last example, figure (5.16), is that of a square plate in uniform tension containing a 45 degree central crack. This is a mixed mode fracture problem involving two crack tips, the program CODE = 3. As in the last example procedure COREGEN has been used to generate the local crack tip mesh, i.e. zones 1-8 and 41-48. The node numbers around the core element must be numbered anti-clockwise, this simplifies the matrix manipulation considerably in the fracture program. It is possible to partition the zone array using a column of voids and then join the appropriate zone faces to form the final distorted zone array. This technique has been used in this example and the joining zone faces are lettered. The advantage of this technique is that it allows complicated structures to be meshed and is analogous to building a jigsaw puzzle, where the jigsaw pieces are equivalent to the zone array subsections. Note that when joining zone faces they must both have the same subdivision, since otherwise a miss-match will occur. This rule applies to all the zones and any subdivision can be specified provided continuity is maintained. A problem arises when joining multiple zone faces, as in this example, four super-nodes namely 53, 69, 99 and 183, all coincide at the centre of the plate. The method used to trace the redundant node numbers relies on the fact that their co-ordinates are unique. Hence in order that the correct node numbers are selected, a means of over-riding the selection process is afforded in the data input. See procedures COINCID and HATCHET.

For details of the job decks and operating methods refer to the Appendix (9.2).



Mat'l cons:- $E = 3.0*10^7$

$v = 0.3$

$G = 1.2*10^7$

Core radius, $Rc = a/30 = 0.1$

$a = 2.8$

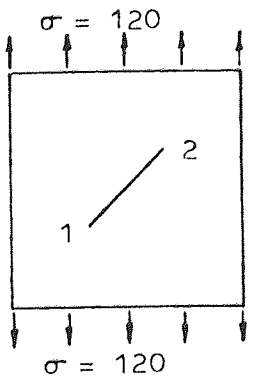
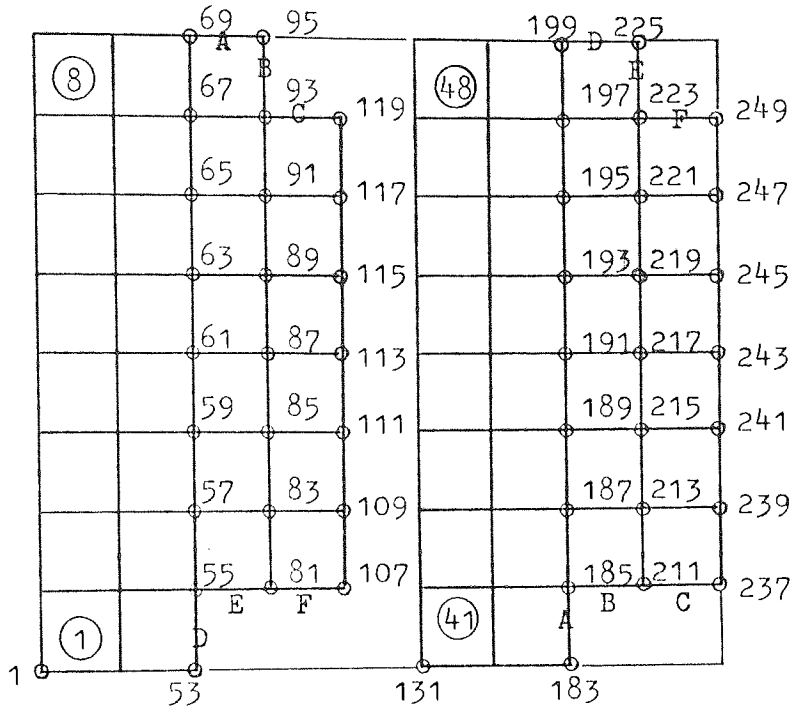
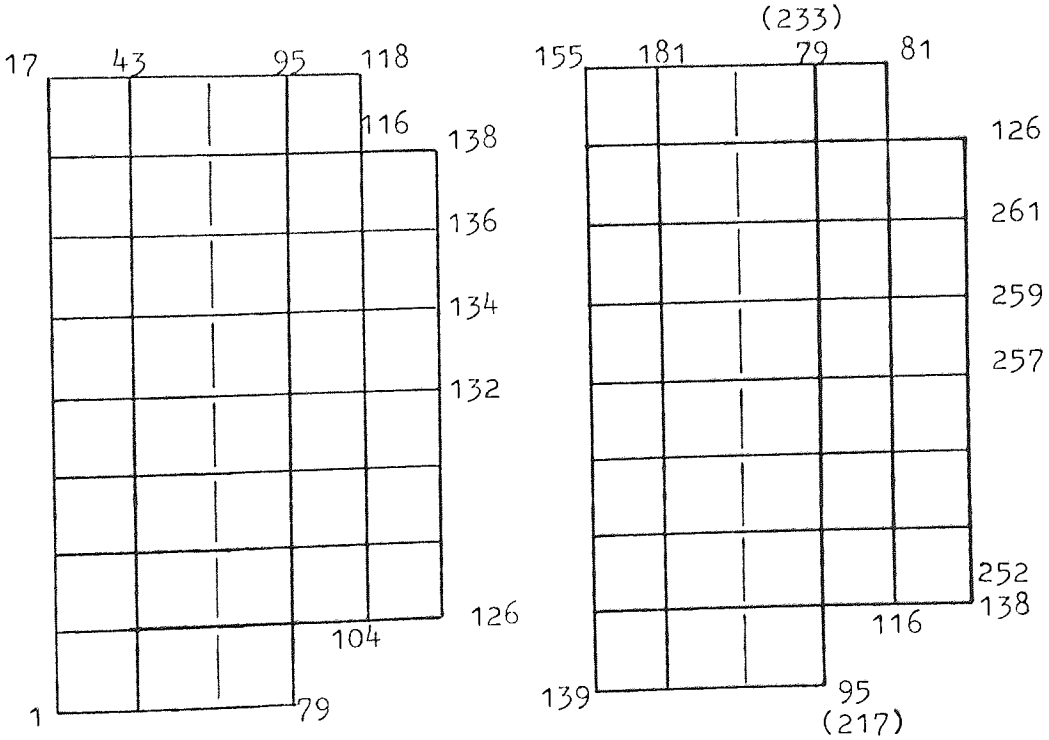


FIG 5.16. Plate in uniform tension with a 45 degree central crack. The zones are joined along faces A,B,C,D,E&F, also identified in the zone array.



Zone array. Again all zones are allocated super-node numbers irrespective of status.



Element array. NNODES = 17*8 + 9*6 + (8+15+7+13)*2 -14 = 262

NELEMT = (8*3 + 2*6 + 1) *2 = 74

FIG 5.17. Example (4)

TABLE 5.18. Data for example (4). Plate in uniform tension with a 45 degree central crack.

```
3,1,1
45 DEG CRACKED PLATE W=L=15 RC= 0.1, A = 2.8, P = 1800
END OF TITLE
74,262,1.0,1,0,0,1,1
39,42,8,9,1
2
1,1,17,5.5,5.5,0.1,45.0,22.5,1,1
131,41,17,9.5,9.5,0.1,-135.0,22.5,1,139
0
4,7.5,7.5,53,199,69,183     2,0.0,15.0,107,249     2,5.5,9.5,81,223
2,5.5,7.25,55,225           2,9.5,7.75,95,185      2,9.5,5.5,93,211
2,15.0,0.0,119,237          1,4.5,6.5,57           1,4.0,5.5,59
1,4.4,4.4,61                1,5.5,4.0,63           1,6.5,4.5,65
1,7.25,5.5,67               1,3.25,7.5,83          1,2.5,5.3,85
1,3.0,3.0,87                1,5.3,2.5,89           1,7.5,3.25,91
1,0.0,10.0,109              1,0.0,5.0,111          1,0.0,0.0,113
1,5.0,0.0,115               1,10.0,0.0,117         1,10.5,8.5,187
1,11.0,9.5,189              1,10.6,10.6,191        1,9.5,11.0,193
1,8.5,10.5,195              1,7.75,9.5,197         1,11.75,7.5,213
1,12.5,9.7,215              1,12.0,12.0,217        1,9.7,12.5,219
1,7.5,11.75,221             1,15.0,5.0,239         1,15.0,10.0,241
1,15.0,15.0,243             1,10.0,15.0,245        1,5.0,15.0,247
16,1,2,1,9,10,11,12,13,14,15,16,49,50,51,52,53,54,55,56
26,1,1,1,18,19,20,21,22,23,24,26,27,28,29,30,31
        58,59,60,61,62,63,64,66,67,68,69,70,71
6                                                   Mesh Generation
9,3,18,4,24,2,24,3,26,4,31,2                           data.
2
79,233,95,217
```

See overleaf for loading and boundary conditions

TABLE 5.18. cont'd

```
14
132,2,0.0,0.0                126,0,0.0,1.0&2
133,0,0.0,-4.0&2             262,0,0.0,4.0&2
134,0,0.0,-2.0&2             261,0,0.0,2.0&2
135,3,0.0,0.0                260,0,0.0,4.0&2
136,0,0.0,-2.0&2             259,0,0.0,2.0&2
137,0,0.0,-4.0&2             258,0,0.0,4.0&2
138,0,0.0,-1.0&2             257,0,0.0,1.0&2
1
3.0&7,0.3,1.2&7,3.0&7,0.3
```

5.5.1 USERS GUIDE

As explained earlier, the mesh generation program has been adapted to several finite element programs, and a list of the data required for each class of program is given in Table (5.19). The programs are described in the Appendix (9.3) and a quick reference can be found in the FILEINDEX, stored on the departments' files and also listed in the Appendix (9.3). If any difficulties arise in interpreting the control variables refer to (28, 29).

TABLE 5.19 A list of the input data for the various program classes is given below.

Note the variables are identified in Table (5.21).

PROGRAM CLASS	1	2	3	4
	CODE QORT NJOB JOB NAME NELEMTS NNODES NSETFS PRNT PRINC NSKEW NMAT NSETF NSETC	CODE QORT NJOB JOB NAME NELEMT NNODE THICK NSETFS NSETC	CODE QORT NJOB JOB NAME NELEMT NNODE THICK NSETFS NSKEW SERNO NSETC NSETF	CODE QORT NJOB JOB NAME NELEMT NNODE NSETFS PRINT SOLID NMAT NSETC NSETF
	MESH GENERATION DATA TABLE (5.20)			
	NSPEC			
B.C's	NODE NO. KODE ULX VLY	— — —	— — —	— — —
Mat'l Con's	CASE ANG THICK	CASE	CASE	CASE'
	E,ν,G,E,ν			
Possible 2nd set of forces		— — —	— — —	— — —
	NOSK ANGSK	HND	NOSK ANGSK	
Possible 2nd set of constraints		— — —	— — —	— — —

(1)*

* See notes

TABLE 5.20 Mesh generation data sequence.

1. CONTROL CARDS

TNSPDS - No. of declared super-nodes, i.e. not including standard generated nodes.

PZONE - No. of zones being used, i.e. not including voids or generated zones.

VZONE - No. of vertical zones
HZONE - No. of horizontal zones
} size of zone array

GH - Graphical output (1/yes, 0/no).

2. STANDARD GEOMETRIES

NTIP - No. of crack tips IF>0 then input the following core parameters:-

NSTART - Super-node no. starting the core.

ZNS - Zone no. starting the core.

N1 - No. of super-nodes on the core face.

X1,Y1 - Coordinates of the crack tip.

RC - Core radius (2)*

A - Starting angle (3)*

A1 - Increment angle (3)*

DY - Zone subdivision in the y-axis

NS - Node no. starting the core

NCM - No. of generated sections IF>0 then input the following parameters:

NSTART - As above

ZNS - " "

N1 - " "

X1,Y1 - As above

R1,R2,R3- Radii for the inner core, grading node and outer node respectively.

A - As above

A1 - " "

DX, DY - Zone's subdivision.

* see notes

3. X & Y COORDINATES OF SPECIFIED SUPER-NODES

Data sequence entered for each node

Q	- No. of super nodes occupying the position	(4)*
XCOD,YCOD	- x and y coordinates	(4)*
W	- String of super-node no.'s	(4)*

4. DEFINING ZONES

ZONE	- No. of like zones	(5)*
MN	- Material No.	(5)*
DIVX,DIVY	- Zone subdivision	(5)*
P	- String of like zone no's	

5. IDENTIFYING CLOSING SIDES

ND - No. of closing faces, IF>0 then input the following parameters for each face:-

ZN - Zone no.

SIDE - Side of face to be joined i.e.(1,2,3 or 4) (6)*

Extra parameters in cases where multiple zone faces are joined..

COIN - No. of coinciding nodes. IF>0 then input the following parameters for each pair of nodes:-

ND - Node No. retained

CND - Corresponding Node No. (7)*

* see notes

TABLE 5.21. Control variables used in the finite element program.

CODE	- Program classification number	(1)*
NJOB	- No. of jobs to be run.	
NELEMT	- No. of elements.	
NNODE	- No. of nodes	
NSETFS	- No. of sets of forces.	
NSETF	- No. of forces for set of forces input.	
PRNT	- Type of output i.e. 1 - stress/strain at nodes. 2 - " / " at element centroid 3 - " / " for both locations.	
PRINC	- Principal stresses & strains 1/yes, 0/no.	
NSKEW	- No. of nodes where skewed b.c's are applied	(8)*
NMAT	- No. of materials.	
NSETC	- No. of sets of constraints.	
THICK	- Thickness of plate.	
SOLID	- 0 - hollow axisymmetric structure. 1 - solid " "	
SURNO	- No. of nodes along the crack face.	(8)*
NSPEC	- No. of nodal points where b.c's are prescribed.	(8)*
KODE	- 0, prescribed loads in the x&y direction. 1, " disp't in the x & load in the y direction. 2, " load in the x & disp't in the y direction. 3, " disp't in both x & y directions	
ULX	- Value of prescribed load or disp't in the x direction.	
VLY	- " " " " " " y "	
CASE	- Type of problem (0-plane stress, 1-plane strain).	
CASE'	- 0-Isotropic, 1-orthotropic.	
ANG	- Angle of orthotropy.	
E,ν,G,E,ν	- Material constants.	
HND	- Direction of crack	(9)*
NOSK	- Node no. where skewed b.c's are applied	(10)*
ANGSK	- Corresponding angle of skew	(10)*
JOB NAME	- Insert job title followed by END OF TITLE.	
QORT	- 0-Triangular element type 1-Quadrilateral element type	* see notes

NOTES:-

1. Program classification:-

CODE

1. Plane stress, plane strain, basic finite element program.

   PCPLSS, PLSS, PCPLSEG.

2. As in (1) but modified to compute Mode I fracture problems.

   PCQTMIST

3. As in (1) but modified to compute mixed mode fracture problems and also crack closure.

   PCPOY, PCPOXY

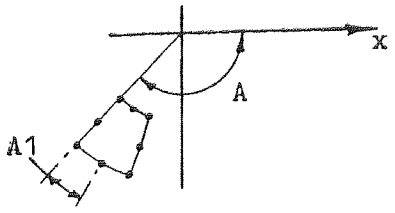
4. Axisymmetric finite element program

   IAAXMG

2. The core element radius is usually 1/30th of a crack length, for details see Chapter Seven - Transition elements.

3. The Starting angle can be defined as the angle between the first zone or crack face, and the +ve x-axis.

i.e.



The incremented angle refers to half the angle taken by the zone. See examples 3 and 4.

4. In the majority of problems zone faces are joined, hence it is possible for super-nodes to have the same co-ordinates. To reduce the data input, the number of super-nodes occupying the same co-ordinates is given, followed by the x & y co-ordinates and the corresponding super-node numbers.

5. A similar situation exists with the zone data, to reduce data input the number of like zones is entered, followed by its material type and sub-division, and finally by a string of corresponding zone numbers.

6. The zone faces are numbered, e.g.,

Side 1 - the left-hand vertical face.

Side 2 - the top horizontal face.

Side 3 - the right-hand vertical face.

Side 4 - the bottom horizontal face.

See examples 1 and 4.

7. Where four or more super-nodes coincide in joining zone faces, an overriding option is available which short circuits the selection process. Therefore the node to be retained and its corresponding node are entered. Any number of coinciding nodes can be entered. See example 4.

8. The variable SURNO is only specified when partial crack closure is expected. As the local node co-ordinates have to be skewed in order to detect the normal displacement between adjacent crack face nodes. Usually NSKEW = SURNO. If other nodes are skewed remote from the crack face NSKEW>SURNO.

9. If the crack tip faces to the right,

i.e.,  then HND = 1.

If the crack tip faces to the left,

i.e.,  then HND = 2

The core element numbering convention for the MIST program, is

See reference (28) for further details.

10. Extension of note 8. When skewed boundary conditions are applied a list of node numbers and corresponding angle of skew must be given. Usually this process is only used in the partial crack closure scheme. The skewed nodes located on the crack face are entered first and in adjacent pairs, specifying the upper node number and then the lower node number. This is necessary in order to detect an overlap condition. See Chapter Seven; Partial Crack Closure.

COMMON ERRORS

For the mesh generation program STDAMG:-

1. If the program fails due to an overflow condition this usually indicates,

   a, The number of super-nodes does not correspond with the input value.

   b, The number of zones does not correspond to the input value.

   c, An error in the super-node or zone numbering.

   d, The zones subdivisions are not compatible.

2. The graphical output is disjointed. Normally this type of error can easily be traced and may be due to one of the following faults:-

   a, An incorrect co-ordinate has been specified.

   b, Joining zone faces have been specified incorrectly.

   c, The number of nodes has been computed incorrectly.

For the general finite element programs:-

1. The program fails during the solving routine SYMVBSOL,

   a, Elements have been allowed to overlap.

   b, The stiffness equations are singular due to incorrectly defining the boundary conditions.

   c, The address sequence is incorrect, arising from incorrectly defined element nodal connections - check the data generating scheme (2c).

2. The program gives incorrect or unexpected results;

   a, Check the boundary conditions making sure they correspond to the correct co-ordinates.

   b, Check for equilibrium.

General Comments:

It always pays to check off each punched card against the written data input sequence, before running the job. Faults which are likely to be found are miss-prints and incorrectly placed commas and full stops.

CHAPTER 6

MODIFICATION OF THE FINITE ELEMENT METHOD TO ACCOMMODATE FRACTURE PROBLEMS

6.1 INTRODUCTION

It is the purpose of this chapter to describe the processes involved in modifying the finite element method to include the local crack tip singularity expressions.

Various methods have been used to obtain the stress distribution around a sharp crack, the majority of which utilise the Airy stress function. The method used here is that of Sih and Liebowitz[75], which is based on the method of Williams[49] combined with the complex function theory of Muskhelishvili[25]. The local crack tip field equations are derived in the following section and the final expressions are listed in the Appendix (9.1). The combination of the classical crack tip formulation and the finite element method developed by Hilton and Hutchinson[51], forms the second part of this chapter. The algebraic and matrix manipulations form the essential part of the numerical procedures presented in the following chapter.

6.2 DERIVATION OF THE LOCAL CRACK TIP FIELD EQUATIONS

The displacement and stress fields around the crack tip can be expressed in terms of the complex functions $\phi(z)$ and $\psi(z)$. According to Muskhelishvili[25], for isotropic, plane problems these expressions

are given as,

$$\sigma_x + \sigma_y = 2[\phi'(z) + \overline{\phi'(z)}] \tag{6.1}$$

$$\sigma_y - \sigma_x + j2\tau_{xy} = 2[\bar{z}\phi''(z) + \psi'(z)] \tag{6.2}$$

$$2\mu(u_x + ju_y) = \kappa\phi(z) - z\overline{\phi'(z)} - \overline{\psi(z)} \tag{6.3}$$

for Cartesian coordinates, and

$$\sigma_r + \sigma_\theta = 2[\phi'(z) + \overline{\phi'(z)}] \tag{6.4}$$

$$\sigma_\theta - \sigma_r + j2\tau_{r\theta} = 2e^{2j\theta}[\bar{z}\phi''(z) + \psi''(z)] \tag{6.5}$$

$$2\mu(u_r + ju_\theta) = e^{-j\theta}[\kappa\phi(z) - z\overline{\phi'(z)} - \overline{\psi'(z)}] \tag{6.6}$$

for Polar coordinates. See figures (6.1) and (6.2).

In these equations,

$$\begin{aligned} &\kappa = 3 - 4\nu \text{ for plane stress,} \\ &\kappa = (3 - \nu)/(1 + \nu) \text{ for plane strain} \\ &\mu = \text{Shear modulus} \end{aligned} \tag{6.7}$$

The complex number $z = x + jy = re^{j\theta}$

and $j = \sqrt{-1}$

The prime denotes the derivative with respect to z and the bar indicates the complex conjugate number, thus,

$\bar{z} = x - jy$ and $\overline{\phi(z)}$ is the equivalent to $\phi(z)$ with j replaced by -j.

In this analysis the Goursat functions are taken as suitable forms for the functions $\phi(z)$ and $\psi(z)$ in the same manner as reference (75).

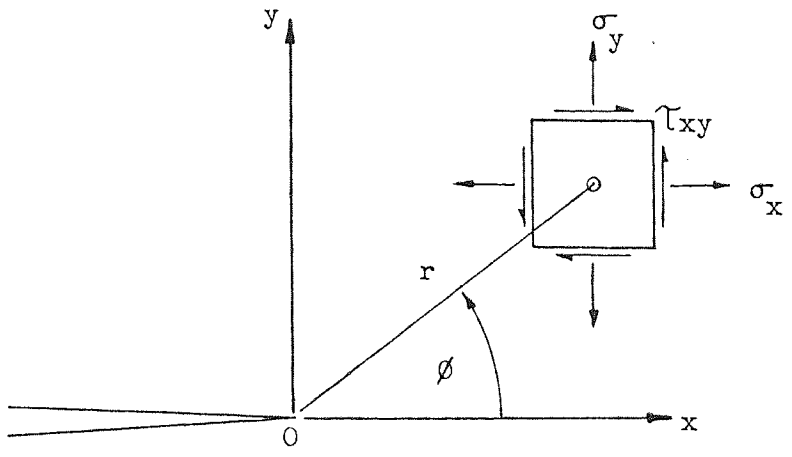


FIG 6.1 Notation for rectangular stress components near a crack tip.

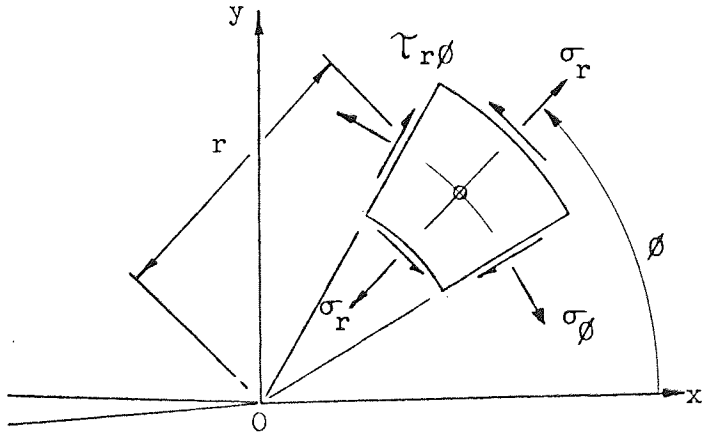


FIG 6.2 Notation for polar stress components near a crack tip.

Hence,

$$\phi(z) = \sum_{n=1}^{\infty} A_n z^{\lambda_n} \qquad (6.8)$$

$$\psi(z) = \sum_{n=1}^{\infty} B_n z^{\lambda_{n+1}}$$

where A_n and B_n are complex constants to be determined from the boundary conditions and λ_n are real eigenvalues. Applying the appropriate boundary conditions, see reference (28), yields,

$$\lambda_n = n/2 \qquad (6.9)$$

and

$$\lambda_n A_n + (-1)^n \bar{A}_n + (\lambda_n+1)B_n = 0 \qquad (6.10)$$

where $n = 1,.....,\infty$.

Substituting equation (6.9) into equations (6.8) gives

$$\phi(z) = \sum_{n=1}^{\infty} A_n z^{n/2} \qquad (6.11)$$

$$\psi(z) = \sum_{n=1}^{\infty} B_n z^{(n/2+1)} \qquad (6.12)$$

Introducing the constants α_{2n-1} and α_{2n} as the real and imaginary parts of A_n, we have,

$$A_n = \alpha_{2n-1} + j\alpha_{2n} \qquad (6.13)$$

also

$$\bar{A}_n = \alpha_{2n-1} - j\alpha_{2n}$$

$$n = 1,.....,\infty.$$

Substituting expression (6.13) into equation (6.10) yields,

$$(\alpha_{2n-1} + j\alpha_{2n})n/2 + (-1)^n(\alpha_{2n-1} - j\alpha_{2n}) + (\frac{n+2}{2})B_n = 0$$

or rearranging

$$B_n = \frac{2}{(n+2)} \{-\alpha_{2n-1}[\frac{n}{2} + (-1)^n] - j\alpha_{2n}[\frac{n}{2} - (-1)^n]\} \qquad (6.14)$$

$$n = 1,\ldots\infty$$

Replacing the complex constants A_n and B_n in equations (6.11) and (6.12), gives

$$\phi(z) = \sum_{n=1}^{\infty} [\alpha_{2n-1} + j\alpha_{2n}]z^{n/2} \qquad (6.15)$$

$$\psi(z) = \sum_{n=1}^{\infty} \frac{2}{n+2} \{-\alpha_{2n-1}[\frac{n}{2} + (-1)^n] - j\alpha_{2n}[\frac{n}{2} - (-1)^n]\} z^{(n+2)/2} \qquad (6.16)$$

Now differentiating $\phi(z)$ w.r.t.z, we have

$$\phi'(z) = \sum_{n=1}^{\infty} [\alpha_{2n-1} + j\alpha_{2n}] \frac{n}{2} z^{(\frac{n}{2}-1)} \qquad (6.17)$$

$$\phi''(z) = \sum_{n=1}^{\infty} [\quad " \quad]\frac{n}{2}(\frac{n}{2} - 1)z^{(\frac{n}{2}-2)} \qquad (6.18)$$

Similarly,

$$\psi'(z) = \sum_{n=1}^{\infty} \{-\alpha_{2n-1}[\frac{n}{2} + (-1)^n] - j\alpha_{2n}[\frac{n}{2} - (-1)^n]\} z^{n/2} \qquad (6.19)$$

$$\psi''(z) = \sum_{n=1}^{\infty} \{\quad " \quad \}\frac{n}{2} z^{(\frac{n}{2}-1)} \qquad (6.20)$$

The bar forms of these expressions can easily be determined. On adding equations (6.4) and (6.5) we have

$$\sigma_\theta + j\tau_{r\theta} = \phi'(z) + \overline{\phi'(z)} + e^{2j\theta}\,[\bar{z}\phi''(z) + \psi''(z)] \tag{6.21}$$

and similarly by subtraction,

$$\sigma_r - j\tau_{r\theta} = \phi'(z) + \overline{\phi'(z)} - e^{2j\theta}[\bar{z}\phi''(z) + \psi''(z)] \tag{6.22}$$

Utilizing equation (6.15) - (6.20) and separating the real and imaginary parts, yields the stress field equations. Therefore, after much algebraic manipulation,

$$\sigma_\theta = \sum_{n=1}^{\infty} \frac{n}{2} r^{\frac{n}{2}-1}\{\alpha_{2n-1}[(\tfrac{n}{2}+1)\cos\theta(\tfrac{n}{2}-1)-(\tfrac{n}{2}+(-1)^n)\cos\theta(\tfrac{n}{2}+1)]$$

$$+ \alpha_{2n}[-(\tfrac{n}{2}+1)\sin\theta(\tfrac{n}{2}-1)+(\tfrac{n}{2}-(-1)^n)\sin\theta(\tfrac{n}{2}+1)]\} \tag{6.23}$$

Similar expressions are found for σ_r and $\tau_{r\theta}$, refer to Appendix (9.1). The displacement expressions can be found in an analogous manner and therefore the corresponding strain expression can be determined, using the relations,

$$\varepsilon_\theta = \frac{1}{r}\frac{\partial u_\theta}{\partial \theta} + \frac{u_r}{r} \tag{6.24}$$

$$\varepsilon_r = \frac{\partial u_r}{\partial r} \tag{6.25}$$

and

$$\gamma_{r\theta} = \frac{1}{r}\frac{\partial u_r}{\partial \theta} + \frac{\partial u}{\partial r} - \frac{u_\theta}{r} \tag{6.26}$$

All relevant equations are listed in Appendix (9.1)

The constants α_1 and α_2 are related to Irwin's definition of the stress intensity factors K_I and K_{II} by,

$$\alpha_1 + j\alpha_2 = \frac{1}{\sqrt{2}} (K_I - jK_{II}) \qquad (6.27)$$

therefore

$$\alpha_1 = \frac{K_I}{\sqrt{2}} \qquad (6.28)$$

and

$$\alpha_2 = \frac{-K_{II}}{\sqrt{2}} \qquad (6.28)$$

In some cases the square root term is replaced by 2π, causing some confusion in the interpretation of the stress intensity values. Usually, however, the form used is quoted and appropriate action can be taken. The field equations are all expressed in terms of the α's and the stress intensity factors are extracted as part of the numerical procedure. The form given in expression (6.28) has been adopted in this analysis.

6.3 MODIFICATION OF THE FINITE ELEMENT METHOD TO INCORPORATE THE HILTON/HUTCHINSON SINGULAR CORE

The Hilton and Hutchinson[51] technique combines the flexibility of the finite element method with the inherent accuracy of the classical crack tip expressions. A circular sub-region surrounding the crack tip is assumed to behave in accordance with the local crack tip field equations, section (6.2) and (9.1) and this region, the 'core', is constrained to match the finite element nodal displacements on its boundary. The method of Lagrange multipliers, as used by Richards[76,77], is employed to formulate the stiffness equations.

Firstly the strain energy of the core region must be determined to complete the potential energy of the system. See figure (6.3), which illustrates a plate containing a single edge crack; the core is surrounded by a simplified finite element mesh. From the previous section the displacement field within the core can be described as,

$$u_x = \sum_{n=1}^{\infty} \frac{r^{n/2}}{2\mu} \{\alpha_{2n-1} f(\nu,\theta) + \alpha_{2n} g(\nu,\theta)\} \tag{6.29}$$

$$u_y = \sum_{n=1}^{\infty} \frac{r^{n/2}}{2\mu} \{\alpha_{2n-1} p(\nu,\theta) + \alpha_{2n} q(\nu,\theta)\} \tag{6.30}$$

or $$\{u\} = [N]_c\{\alpha\} \tag{6.31}$$

where the α's represent the unknown quantities and matrix $[N]_c$ contains the relevant functions. Using the strain displacement relationship we can write,

$$\{\varepsilon\} = [B]_c\{\alpha\} \tag{6.32}$$

Hence the strain energy of the core region is given by

$$U_c = \frac{1}{2} \int_{vol} \{\sigma\}^t\{\varepsilon\}dvol \tag{6.33}$$

Using the stress-strain relation $\{\sigma\} = [C]\{\varepsilon\}$, this becomes

$$U_c = \frac{1}{2} \{\alpha\}^t \int_{vol} [B]_c^t[C][B]_c dvol\{\alpha\} \tag{6.34}$$

or $$U_c = \frac{1}{2} \{\alpha\}^t[k_c]\{\alpha\} \tag{6.35}$$

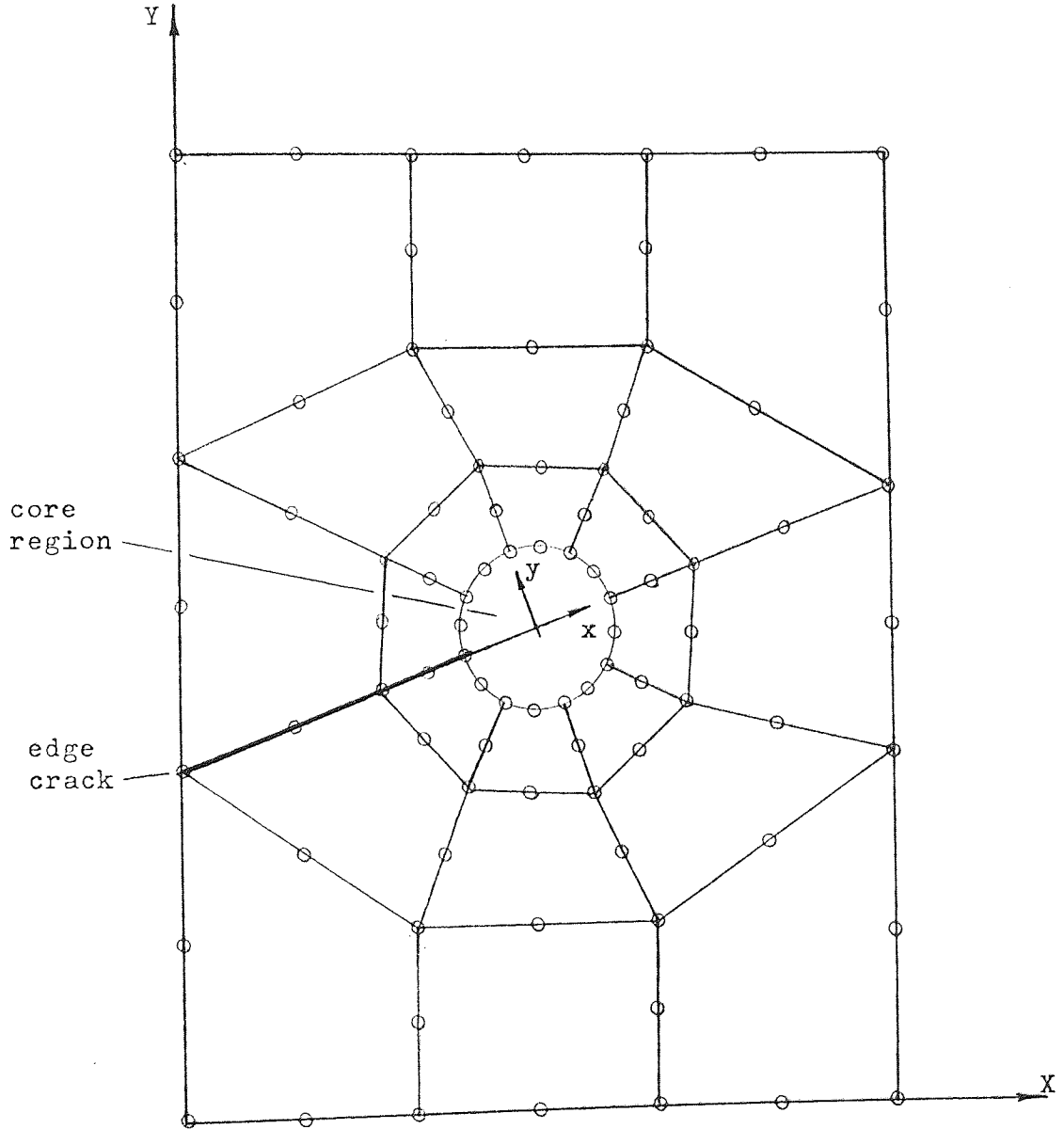


FIG 6.3 Plate with a single edge crack, showing the local crack tip coordinate system of the core element, surrounded by a simplified finite element network.

The circular core considerably eases the integration necessary to form $[k_c]$, in fact it will be shown later that the matrix can be expanded to any degree using a single expression, hence taking into account further terms in the series formulation.

The total potential energy of the core-mesh system is therefore,

$$V = \frac{1}{2}\{\alpha\}^t[k_c]\{\alpha\} + \frac{1}{2}\{q\}^t[K]\{q\} - \{q\}^t\{Q\} \qquad (6.36)$$

where $[K]$ is the stiffness matrix of the surrounding mesh, $\{q\}$ is the nodal displacement vector and $\{Q\}$ is the force vector. The displacements associated with the nodes on the core interface boundary can be expressed as a function of the unknown α's. Using equation (6.31) the displacements at these common nodes can be defined, relative to the local crack tip coordinates, as,

$$\{u\}_c = \begin{Bmatrix} \{u_1\}_c \\ \{u_2\}_c \\ \vdots \end{Bmatrix} = \begin{Bmatrix} [N]_c\ \{\alpha\} \\ [N]_c\ \{\alpha\} \\ \vdots \end{Bmatrix} = [\bar{A}]\{\alpha\} \qquad (6.37)$$

Referring the local coordinates to the global system yields,

$$\{q_1\} = [A]\{\alpha\} \qquad (6.38)$$

where $\{q_1\}$ represents the displacements of the nodes on the core interface. Using the Lagrange multiplier method to enforce continuity of displacement at the juncture of the core and the finite elements, the modified potential energy $\bar{V}$ is given by

$$\bar{V} = \frac{1}{2}\{\alpha\}^t[k_c]\{\alpha\} + \frac{1}{2}\{q\}^t[K]\{q\} - \{q\}^t\{Q\}$$

$$+(\{q_1\} - [A]\{\alpha\})^t\{\lambda\} \tag{6.39}$$

hence $\{\lambda\}$ represents a set of Lagrange multipliers. Variation $\delta\bar{V}$ is found by treating the α's and q's as free variables which may all receive arbitrary independent increments, so that for equilibrium,

$$\delta\bar{V} = 0 = \{\delta\alpha\}^t[k_c]\{\alpha\} + \{\delta q\}^t[K]\{q\} - \{\delta q\}^t\{Q\}$$

$$+ (\{\delta q_1\} - [A]\{\delta\alpha\})^t\{\lambda\} \tag{6.40}$$

Partitioning $\{q\}$ and $[K]$ such that,

$$\{q\} = \begin{Bmatrix} \{q_1\} \\ \{q_2\} \end{Bmatrix} \quad [K] = \begin{bmatrix} [K_{11}] & [K_{12}] \\ [K_{21}] & [K_{22}] \end{bmatrix} \tag{6.41}$$

equation (6.40) can be rewritten as,

$$\begin{aligned} 0 = {} & \{\delta\alpha\}^t([k_c]\{\alpha\} - [A]^t\{\lambda\}) \\ & + \{\delta q_1\}^t([K_{11}]\{q_1\} + [K_{12}]\{q_2\} - \{Q_1\} + \{\lambda\}) \\ & + \{\delta q_2\}^t([K_{21}]\{q_1\} + [K_{22}]\{q_2\} - \{Q_2\}) \end{aligned} \tag{6.42}$$

Since $\{\delta\alpha\}$,$\{\delta q_1\}$ and $\{\delta q_2\}$ are arbitrary, equation (6.42) yields three simultaneous matrix equations of equilibrium

$$[k_c]\{\alpha\} - [A]^t\{\lambda\} = 0 \tag{6.43}$$

$$[K_{11}]\{q_1\} + [K_{12}]\{q_2\} - \{Q_1\} + \{\lambda\} = 0 \tag{6.44}$$

$$[K_{21}]\{q_1\} + [K_{22}]\{q_2\} - \{Q_2\} = 0 \tag{6.45}$$

Premultiplying equation (6.44) by $[A]^t$ and combining with equation (6.43), yields,

$$[A]^t[K_{11}]\{q_1\} + [A]^t[K_{12}]\{q_2\} + [k_c]\{\alpha\} - [A]^t\{Q_1\} = 0 \quad (6.46)$$

Replacing vector $\{q_1\}$, results in the final expressions,

$$\begin{aligned} ([k_c] + [A]^t[K_{11}][A])\{\alpha\} + [A]^t[K_{12}]\{q_2\} - [A]^t\{Q_1\} &= 0 \\ [K_{21}][A]\{\alpha\} + [K_{22}]\{q_2\} - \{Q_2\} &= 0 \end{aligned} \quad (6.47)$$

These two equations can be expressed in one modified stiffness equation, thus,

$$[K^*]\{q^*\} = \{Q^*\} \quad (6.48)$$

where

$$[K^*] = \begin{bmatrix} [K_{22}] & , [K_{21}][A] \\ [A]^t[K_{12}], & [k_c] + [A]^t[K_{11}][A] \end{bmatrix} \quad (6.49)$$

$$\{Q^*\} = \begin{Bmatrix} \{Q_2\} \\ [A]^t\{Q_1\} \end{Bmatrix} \quad (6.50)$$

and

$$\{q^*\} = \begin{Bmatrix} \{q_2\} \\ \{\alpha\} \end{Bmatrix} \quad (6.51)$$

Equation (6.48) may be solved to determine the nodal displacements and the crack tip stress intensity factors. Comparing the stiffness matrices [K] and [K*] of expressions (6.41) and (6.49), gives an indication of the manipulations required in combining the core and finite element system.

CHAPTER 7

NUMERICAL PROCEDURES AND EXAMPLES FOR VARIOUS CRACK CONFIGURATIONS

7.1 INTRODUCTION

The fracture mechanics programs, forming the basis of the work presented here, were written by Robertson[28], and have been augmented in line with his suggestions for further program development. This chapter deals with the various additional developments, examining each new feature in turn together with a complementary set of numerical examples. Firstly, the mixed mode fracture program, originally designed to accept one or two crack tip core regions, has now been modified and is applicable to any crack configuration. The computer storage requirements of the modified stiffness matrix has also been minimized.

The number of terms retained in the singularity expression has been made optional, so that any appropriate degree of expansion can be chosen. The variable ELDOF controls the number of degrees of freedom taken in the core element. Unfortunately this area of work has not been finalized due to an unidentified program error. However, the program functions correctly when using the first two terms of the crack tip field equation.

The problem of partial crack closure, if treated using the standard finite element method, results in the physically inadmissible

situation where the free crack boundaries have crossed. Problems of this type were previously treated using an averaged displacement technique. An iterative approach has been developed, as part of the present investigation, which examines the free crack surfaces in order to determine the point of maximum overlap and subsequently condenses the corresponding adjacent nodal degrees of freedom, giving either a no-slip model or a frictionless model. A similar method was designed using bar elements to bridge adjacent nodes, but this was later abandoned in favour of the above technique. The scheme is also applicable to any general situation where free boundaries may overlap and has been mounted in the multi-tip program for the particular purpose of examining cracked structures.

The local mesh configuration around the core element has been examined in much detail by Robertson and to a lesser extent by Alsharqi[29], the mesh designs adopted are very similar forming a finely graded element distribution around the core. Following the work of Lynn and Ingraffea[58], it is possible to substitute this finely graded mesh by a ring of isoparametric quadrilateral elements, suitably constructed to sense the crack tip singularity. The elements have been termed 'transition' elements and were used originally in conjunction with the crack tip degenerate isoparametric elements, in order to improve the solution accuracy.

The concept of a strain-energy density factor, introduced by Sih[16], enables the crack propagation angle to be computed. This facility has been incorporated in a separate program owing to the limits of the small core memory (SCM) presently set at the Manchester

Regional Computer Centre. The SCM is used to store the compiled program plus any ancillary routines, such as the NAG library, together with the real and integer variables. Unfortunately the NAG routines, used in the crack propagation program, causes an overflow of the SCM, hence this facility cannot be added to the main fracture program. A request has been made to increase the SCM limits, so hopefully the crack propagation program can be added as a further routine to the main program. The multi-tip mixed mode program PCPOXY is listed in the Appendix (9.3) with an appropriate flowchart.

7.2 MULTI-TIP MIXED MODE PROCEDURE

7.2.1 INTRODUCTION

The adaptation of the general purpose finite element program for fracture mechanics studies, centres on the matrix manipulation referred to in Chapter (6.3). In order to illustrate the matrix operations used in the multi-tip mixed mode procedure, consider a hypothetical problem containing three crack tips. The form of the stiffness matrix is shown in figure (7.1). Because the matrix is symmetric about its diagonal, only half of the matrix is given. The position of the nodes lying on the core interface dictate the internal construction of the matrix. Note that the interface nodal numbering is consecutive and this eases the problems involved in manipulating the sub-matrices. As the nodes are entered in this regular fashion it is possible to forecast the regions within the matrix which will be zero.

From figure (7.4) it can be seen that the nodes on the core interface are not connected to any node numbers greater than (3*N1)

through the inter-element grid. Here, N1 represents the number of interface nodes. Using similar reasoning the matrix structure will have the general form of figure (7.1), taking into account all possible inter-element connections. Corresponding with the diagrams the following variables are defined as,

NS[I] - The value of the node number starting the core I.

N1[I] - The number of interface nodes on the core I.

NG - The number of d.o.f. of the original finite element mesh.

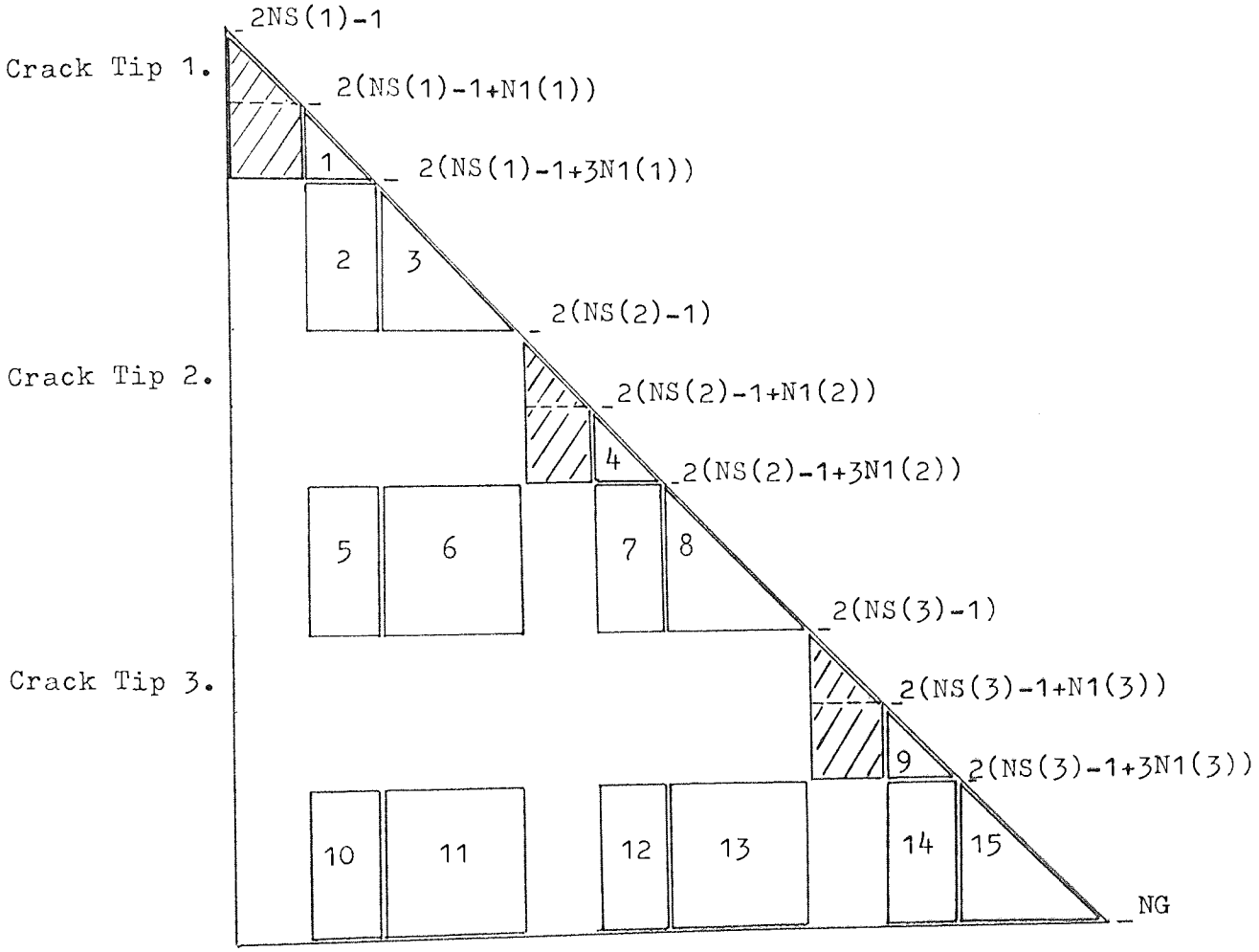
Returning to figure (7,1), the shaded areas of the matrix can be identified as the sub-matrices $[K_{11}]$ and $[K_{21}]$ of equation (6.41), and these matrices are operated on to form part of the modified stiffness matrix, i.e.

$$[K] = \begin{bmatrix} [K_{11}], & [K_{12}] \\ \\ [K_{21}], & [K_{22}] \end{bmatrix}$$

and the corresponding modified matrix,

$$[K^*] = \begin{bmatrix} [K_{22}], & [K_{21}][A] \\ [A]^t[K_{12}], & [k_c]+[A]^t[K_{11}][A] \end{bmatrix}$$

Clearly, from the above, the sub-matrix $[K_{22}]$ remains intact and is simply re-positioned in the modified stiffness matrix. The sub-matrix $[K_{22}]$ is analogous to the numbered sub-matrices of figure (7.1).

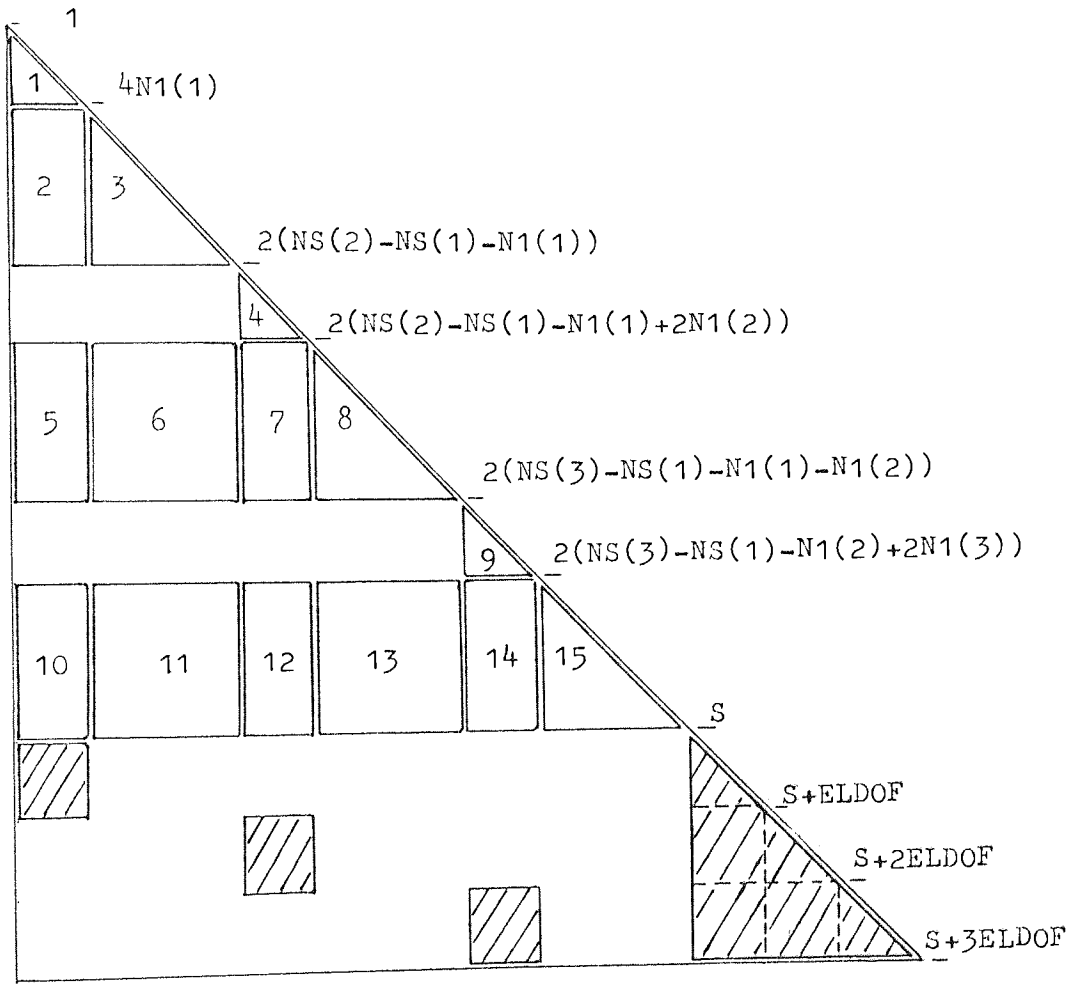


NS(I) - Node No. starting the core I.

N1(I) - No. of nodes on the core interface I.

NG - No. of d.o.f. in the original element mesh.

FIG 7.1 Symmetric stiffness matrix (K), showing the populated regions, for a hypothetical fracture problem containing three crack tips.



NS(I) - Node No. starting the core I.

N1(I) - No. of nodes on the core interface I.

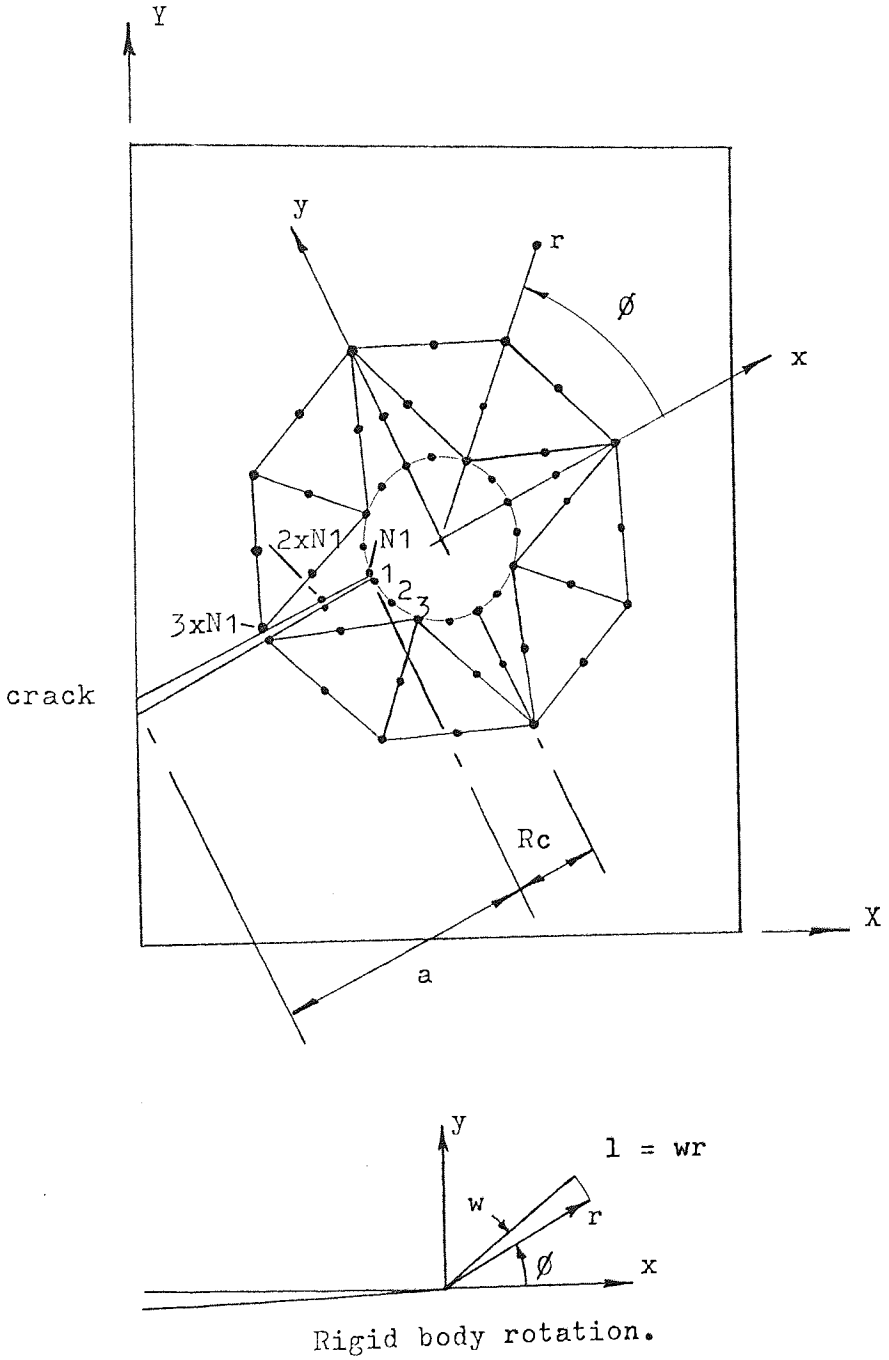
S = NG+2(1-NS(1)-N1(1)-N1(2)-N1(3))

NG - No. of d.o.f. in the original element mesh.

ELDOF - No. of terms taken in the core element.

FIG 7.2 Modified stiffness matrix (K*), showing the new positions of the sub-matrices given in figure(7.1).

FIG 7.3 Reassembled stiffness matrix representing a minimum storage configuration.



Rigid body rotation.

FIG 7.4 The nodes on the core element interface have no inter-element connections with node numbers greater than 3xN1.

Now consider the equivalent modified stiffness matrix [K*] of figure (7.1), as given in figure (7.2). This shows the new positions of the numbered sub-matrices, together with the redefined shaded areas The dimensions of the sub-matrices are given on the edge of the diagrams and are in terms of the core parameters NS, N1 and NG. Before examining the manipulation processes in detail, it is worth noting that the stiffness matrix [K] can be modified and reassembled to present a minimum storage configuration, as figure (7.3) illustrates. However, this arrangement was not adopted due to the complex programming operations involved in relation to the small computer storage space savings. It will be appreciated, later in this section, that the size of the storage area for the modified stiffness matrix will be larger than, or equal to, that of matrix [K] and cannot be smaller. Hence any storage minimization scheme of this kind may become redundant.

The operation of the program follows that of the conventional finite element method described in section (4.2), but with the following modifications. Firstly the storage requirements of the new stiffness matrix [K*] must be determined in relation to the size of the original stiffness matrix [K]. This operation is performed by procedure MMNTKBAND. Secondly the matrix [K] and corresponding vectors {ADD} and {Q} must be modified forming the new matrices [K*], {ADD*} and {Q*}. Note that matrix [K] is held in a one-dimensional array and is related to its accepted two-dimensional form through the address sequence {ADD}. Hence, as [K] is modified it follows that its address sequence is similarly modified. These operations are carried out by the procedure MMNT. The reassembly

of the stiffness matrix, in effect implies that the element node numbers have been redefined, but as this is not so some mechanism is needed to relate the old scheme with the new numbering system. This mechanism is required in the application of the geometric boundary conditions, and also in the print out information. The program control sequence is listed in the Appendix (9.3).

7.2.2 DESCRIPTION OF PROCEDURE MMNTKBAND

The storage requirements of the modified stiffness matrix [K*] may be greater than that computed by procedure ADDARRAY for the original stiffness matrix [K]. In order to provide for this eventually, procedure MMNTKBAND predetermines the storage requirements of the modified stiffness matrix. The numbered sub-matrices of figure (7.1) remain intact and are simply reassembled as in figure (7.2). Also, it can be seen, that the sub-matrix pattern is repeated for each crack tip, for example, sub-matrices 1, 4 and 9 occupy similar locations, together with bands 2-3, 5-6-7-8 and 10-11-12-13-14-15. In this procedure the numbered sub-matrices are scanned, examining the number of coefficients in each row and recording the summed total in variable BAND. It will be noticed that the numbered sub-matrices close-up in the reassembled stiffness matrix [K*], reducing the effective length of the rows. This is automatically taken into account in the summing process. The following steps refer to the flowchart overleaf and the stiffness matrices of figures (7.1) and (7.2):-

1. Various control variables are initialised.
2. The counter loop C is constructed to examine each sub-matrix block. Within the loop I, sub-matrices 1, 4 and 9, for example are scanned for each crack tip C. The row value O

is computed and used to find the new row length via the ADD array. If the row extends into the shaded areas, figure (7.1), then the row length is given by I, else the row is short and corresponds to 0.

3. The control variables R and P are preset. R is an accumulator finally used in step 6.

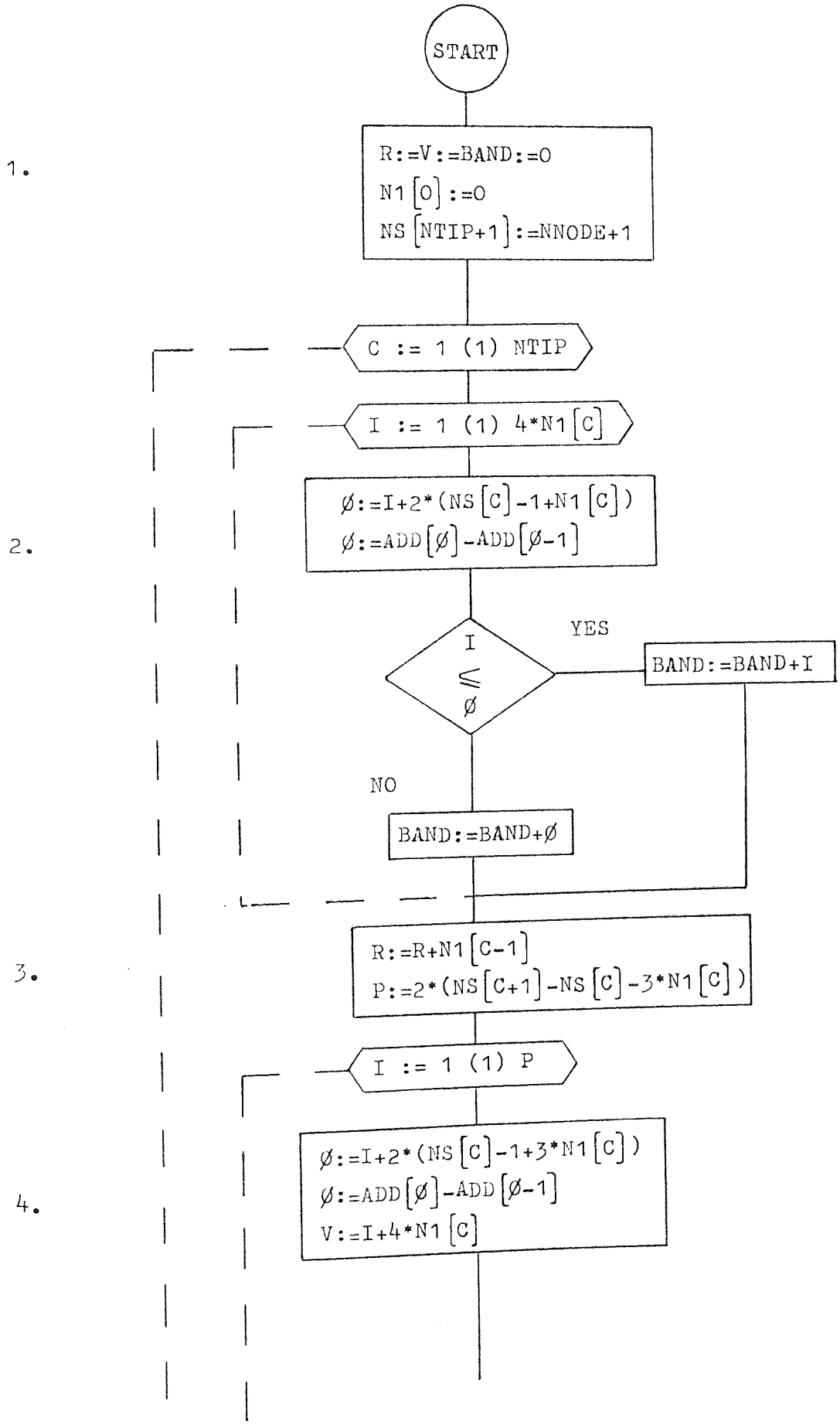
4. The second set of sub-matrices are examined, i.e. 2-3, 5-6-7-8 and 10-11-12-13-14-15, using the counter C. The rows are scanned using counter I, in a similar fashion to that of step 2, however, in this case the blank portions lying between sub-matrices 6 and 7, or 11 and 12, are taken into account within the loop J.

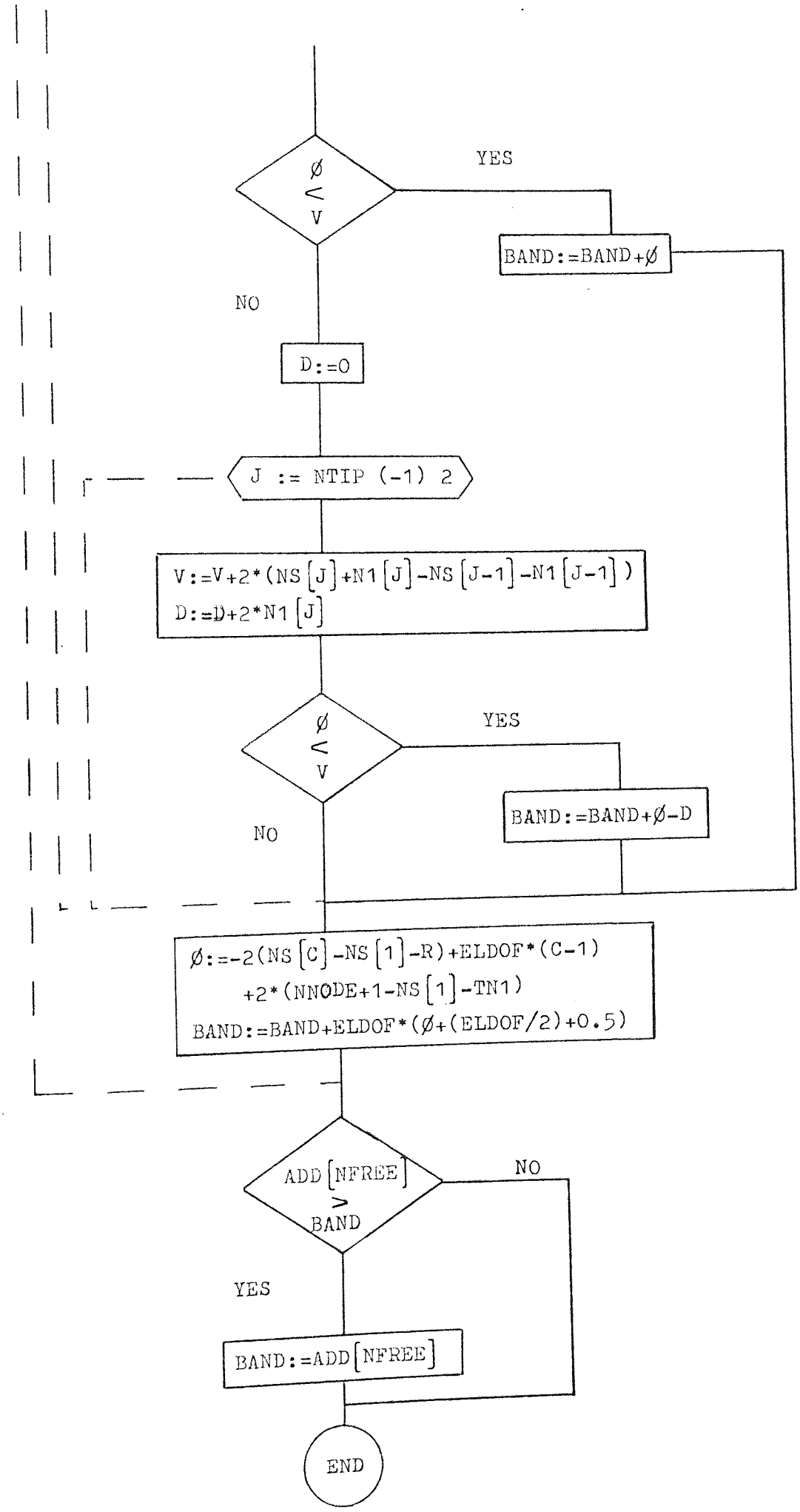
5. If the row length, determined by D, extends across a partitioning blank then its reassembled length, figure (7.2), is modified within loop J.

6. The size of the shaded regions of figure (7.2) are computed and added to the variable BAND. The steps 2-6 are again repeated for each crack-tip or sub-region.

7. Finally the storage requirements of the reassembled matrix [K*] is compared with that of matrix [K] and variable BAND is set to the greater of these values. The bounds of the stiffness matrix will later be declared as K[1:BAND].

Flowchart for procedure MMNTKBAND :-




Ø
<
V
YES
BAND:=BAND+Ø
NO
D:=0
J := NTIP (-1) 2
V:=V+2*(NS[J]+N1[J]-NS[J-1]-N1[J-1])
D:=D+2*N1[J]
5.
Ø
<
V
YES
BAND:=BAND+Ø-D
NO
Ø:=-2(NS[C]-NS[1]-R)+ELDOF*(C-1)
+2*(NNODE+1-NS[1]-TN1)
BAND:=BAND+ELDOF*(Ø+(ELDOF/2)+0.5)
6.
ADD[NFREE]
>
BAND
NO
7.
YES
BAND:=ADD[NFREE]
END

7.2.3 DESCRIPTION OF PROCEDURE MMNT

This procedure modifies the stiffness matrix [K] to form the matrix [K*], as shown in figures (7.1) and (7.2). To achieve this the matrix operations inferred by equations (6.47) are executed. Before the above equations can be implemented matrices [A] and $[k_c]$ must be determined. Matrix [A] connects the interface nodal displacements with the wanted stress intensity factors and rigid body modes. It is constructed using the rectangular displacement field equations of expression (6.29) and (6.30), plus the rigid body displacement terms, thus,

$$u_x = K_I f(r,\nu,\theta) + K_{II} g(r,\nu,\theta) + \delta x - \omega r\sin\theta$$
$$u_y = K_I p(r,\nu,\theta) + K_{II} q(r,\nu,\theta) + \delta y + \omega r\cos\theta \qquad (7.1)$$

where the functions $f(r,\nu,\theta)$, $g(r,\nu,\theta)$, etc., represent the expressions given in Appendix (9.1). The series has been truncated taking the first two terms only, so that n = 1, and

$$\alpha_{2n-1} = K_I/\sqrt{2}, \qquad \alpha_{2n} = -K_{II}/\sqrt{2}$$

δx, δy and ω represent the rigid body displacement components. By inserting the corresponding values of r and θ for each interface node the expression (6.37) can be formed. Referring the local crack tip displacement coordinates to the global coordinate system, yields,

$$\{q_1\} = [A]\{\alpha\} \qquad (7.2)$$

where $\{q_1\}$ is the vector of interface nodal displacements, and

$$\{\alpha\} = \begin{Bmatrix} \omega \\ \delta x \\ \delta y \\ K_I \\ K_{II} \end{Bmatrix} \qquad (7.3)$$

The number of terms taken in the series expression of Appendix (9.1) can be increased by adjusting the value of ELDOF, refer to section (7.3).

The core stiffness matrix $[k_c]$ can be formed using the series expression given in Appendix (9.1). It will be appreciated that the rigid body displacements have no effect on the strain energy stored in the core element and therefore the relevant coefficients of the matrix $[k_c]$ are zero. Procedure CORK evaluates matrix $[k_c]$ and is described in detail, in section (7.3). Here we shall use the first two terms in the series and the matrix $[k_c]$ takes the form,

$$[k_c] = \begin{bmatrix} & 0 & 0 \\ 0 & 0 & 0 \\ & 0 & 0 \\ & \frac{Rc\pi}{8\mu}(3+2\kappa) & 0 \\ 0 & & \\ & 0 & \frac{Rc\pi}{8\mu}(2\kappa-1) \end{bmatrix} \qquad (7.4)$$

Having evaluated the matrices [A] and $[k_c]$ the sub-matrices,

$$[A]^t[K_{12}]$$

and $$[k_c] + [A]^t[K_{11}][A] \qquad (7.5)$$

can be formed, as given in equation (6.47). These sub-matrices are represented as shaded areas in figure (7.2), and are held in array [KC] in a compact form during the operation of procedure MMNT. This extra storage space is unavoidable and there is no alternative but to store the expressions of (7.5) in array [KC], whilst the

overall stiffness matrix [K] is reassembled as [K*]. As stated earlier the numbered sub-matrices of figure (7.1) are relocated in the modified matrix as given by figure (7.2). As each row is relocated the address sequence {ADD} and force vector {Q} are similarly modified. Finally the sub-matrices of array [KC] are appropriately positioned in the new matrix [K*], overwriting the coefficients of the old matrix [K]. The procedure operations are described by the following steps and refer to the flowchart overleaf:-

1. Array [KC] is initialised.
2. Constants preset using the material properties in array [A].
3. The details of each core are printed giving the core radius, the number of interface nodes, its starting node number, and angle relative to the positive x-axis. The sizes of the [K] and [K*] matrices are also printed.
4. The variable KAP is determined depending on the particular case chosen, i.e. 0 = plane stress and 1 = plane strain.
5. The loop C is set up for each crack tip and generates the array [KC]. Firstly, the crack tip angle is defined and procedure CORK computes the stiffness matrix $[k_c]$ for the crack tip element under consideration. See section (7.3) for the operation of procedure CORK.
6. For each interface node the coefficients of equation (6.37) are formed. Here matrix $[F] = [N_i]_c$, where i = NS[C] to NS[C]+ N1[C]. The angle of each node with respect to the local coordinate system is represented by real TA. The expressions within the loop J are equivalent to the functions $f(r,\nu,\theta)$, $g(r,\nu,\theta)$ etc., of expressions (6.29) and (6.30).

7. The stress intensity factors are recognized and divided by $\sqrt{2}$.

8. The matrix [A] given in expression (6,38) is formed by referring the local crack tip coordinates to the global system. Here matrix [FI] = [A].

9. Step 9 represents the formation of sub-matrix $([k_c] + [A]^t[K_{11}][A])$. The assembly process operates within the loops Z, W, I and J, where the sub-matrix $[K_{11}]$ is scanned and the appropriate coefficients are recorded by variable KA. The array [KC] is partially formed summing on counters I and J, and by adding the core stiffness matrix [KT] with loops W and Z.

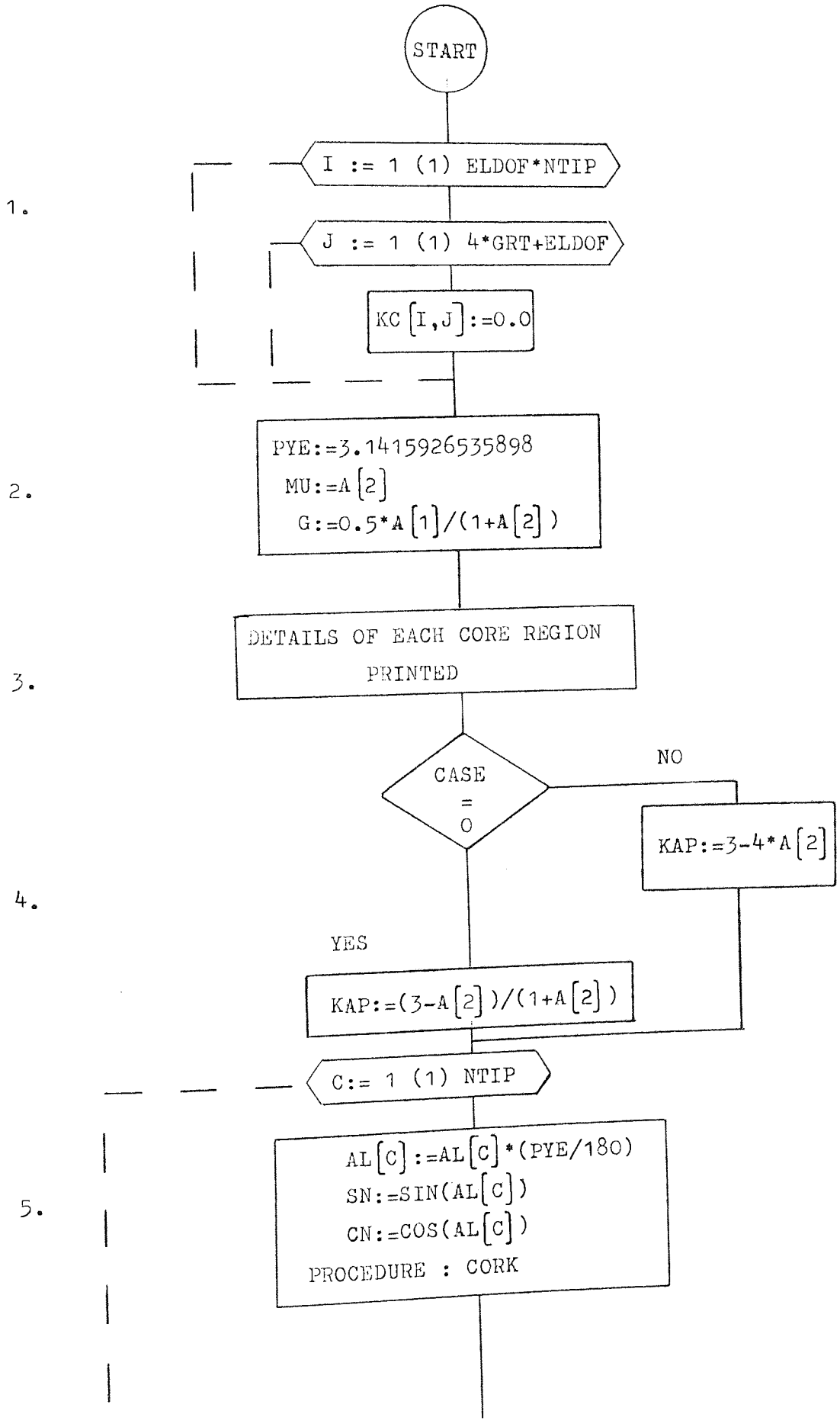
10. The array [KC] is computed by adding sub-matrices $([A]^t[K_{12}])$. This is achieved in a similar manner to step 9, where sub matrix $[K_{12}]$ is scanned within loops W, I and J, and the appropriate coefficient is assigned to variable KA. Array [KC] is then completed using the coefficients of array [FI].

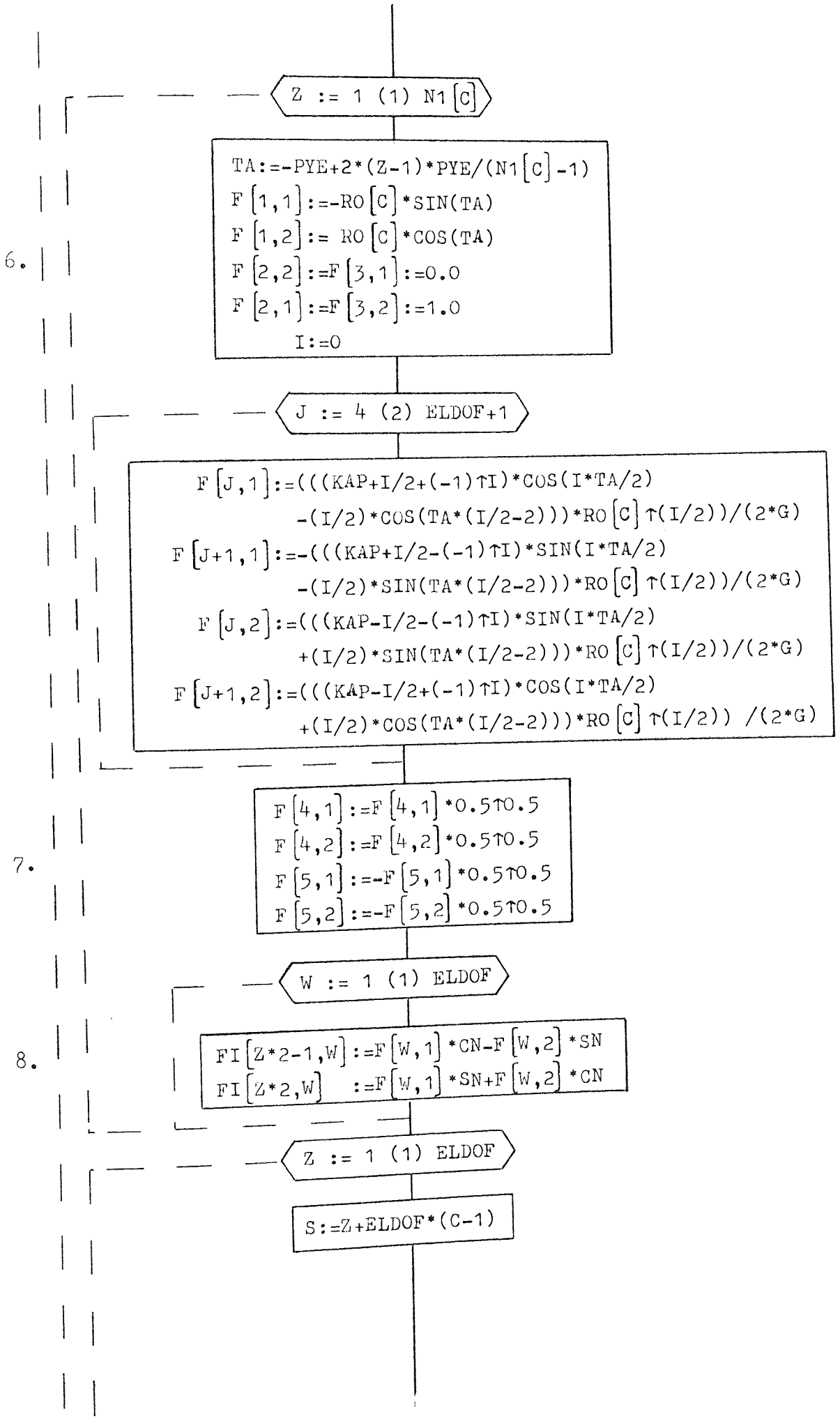
11. This section of the flowchart deals with the transfer of the numbered sub-matrices (1, 4 and 9) of figure (7.1) into their new positions given by figure (7.2). The force vector {Q} and the address sequence {ADD} are simultaneously adjusted. The rows within the sub-matrices 1, 4 or 9 could fall short of the shaded area and this is taken into account by the conditional statement.

12. This step is similar to the previous step (11) but further complicated by the blank sections of the stiffness matrix, as inbetween the sub-matrices 6-7 for example. Therefore further conditional statements are needed to determine the row length. Note that each sub-region is dealt with in turn within the loop C, i.e. 1, 2-3, 4, 5-6-7-8, etc.

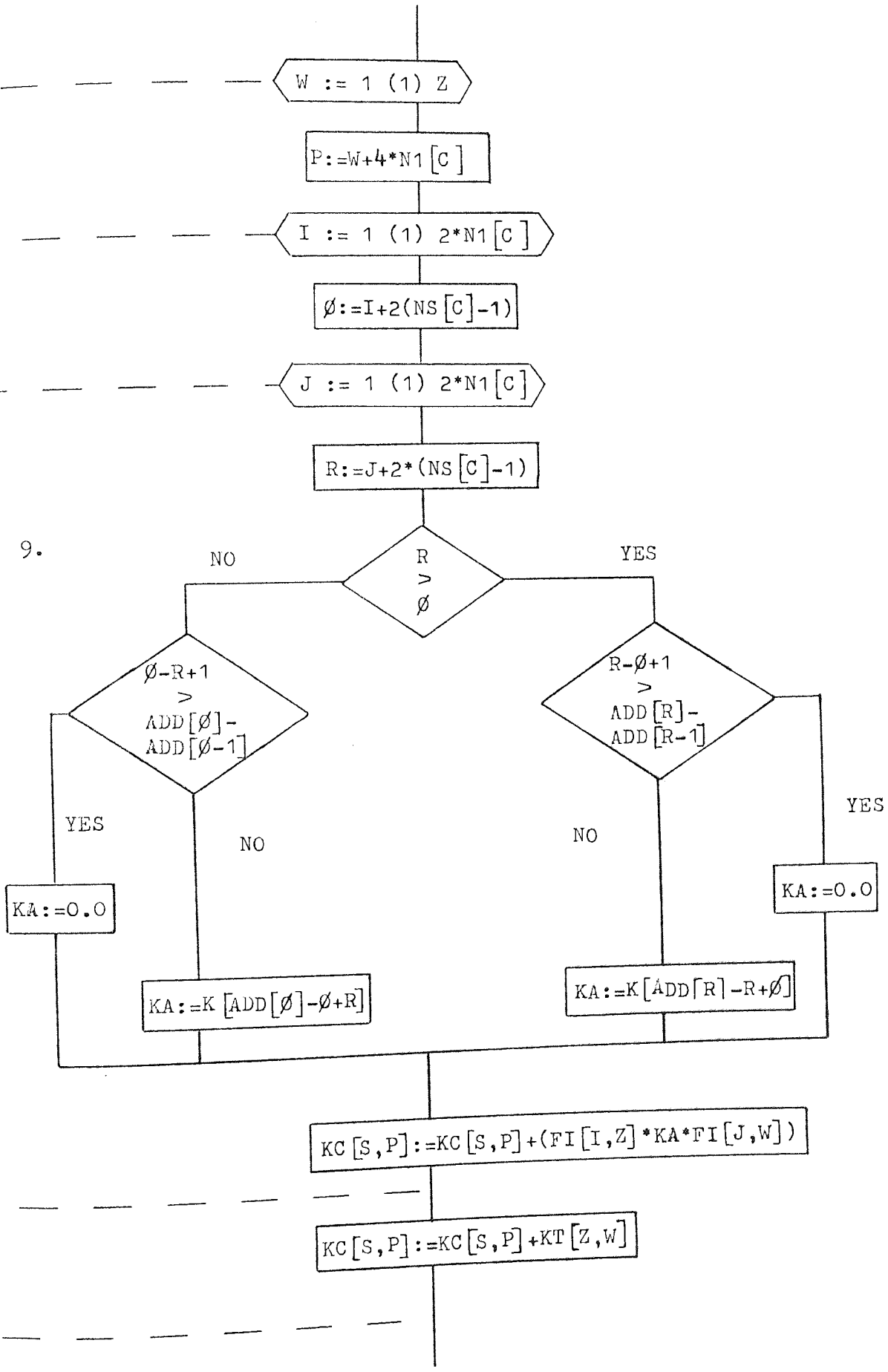
13. Finally after the numbered sub-matrices have been relocated the coefficients of array [KC] are transferred, completing array [K*], as shown in figure (7.2).

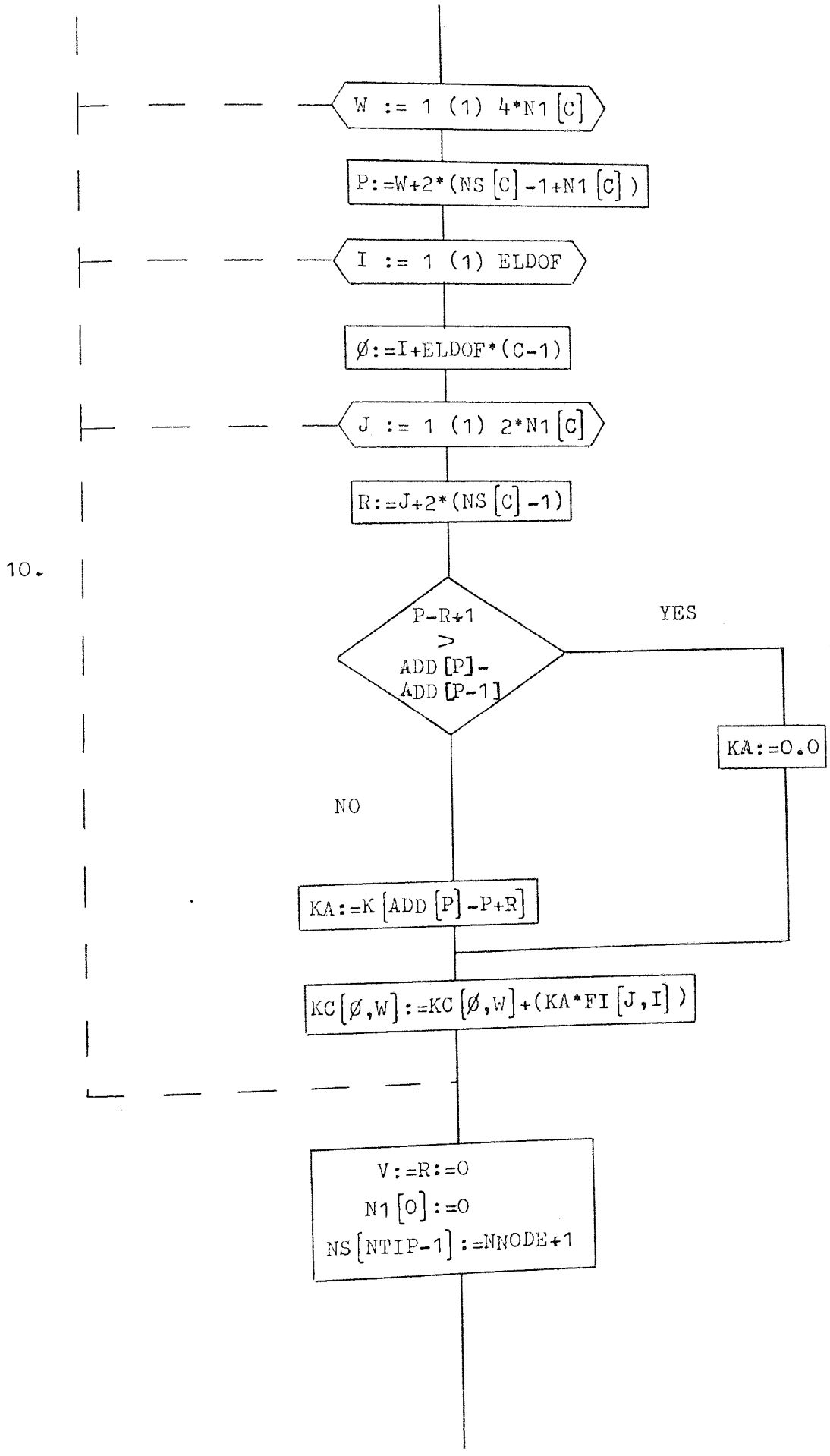
Flowchart for procedure MMNT :



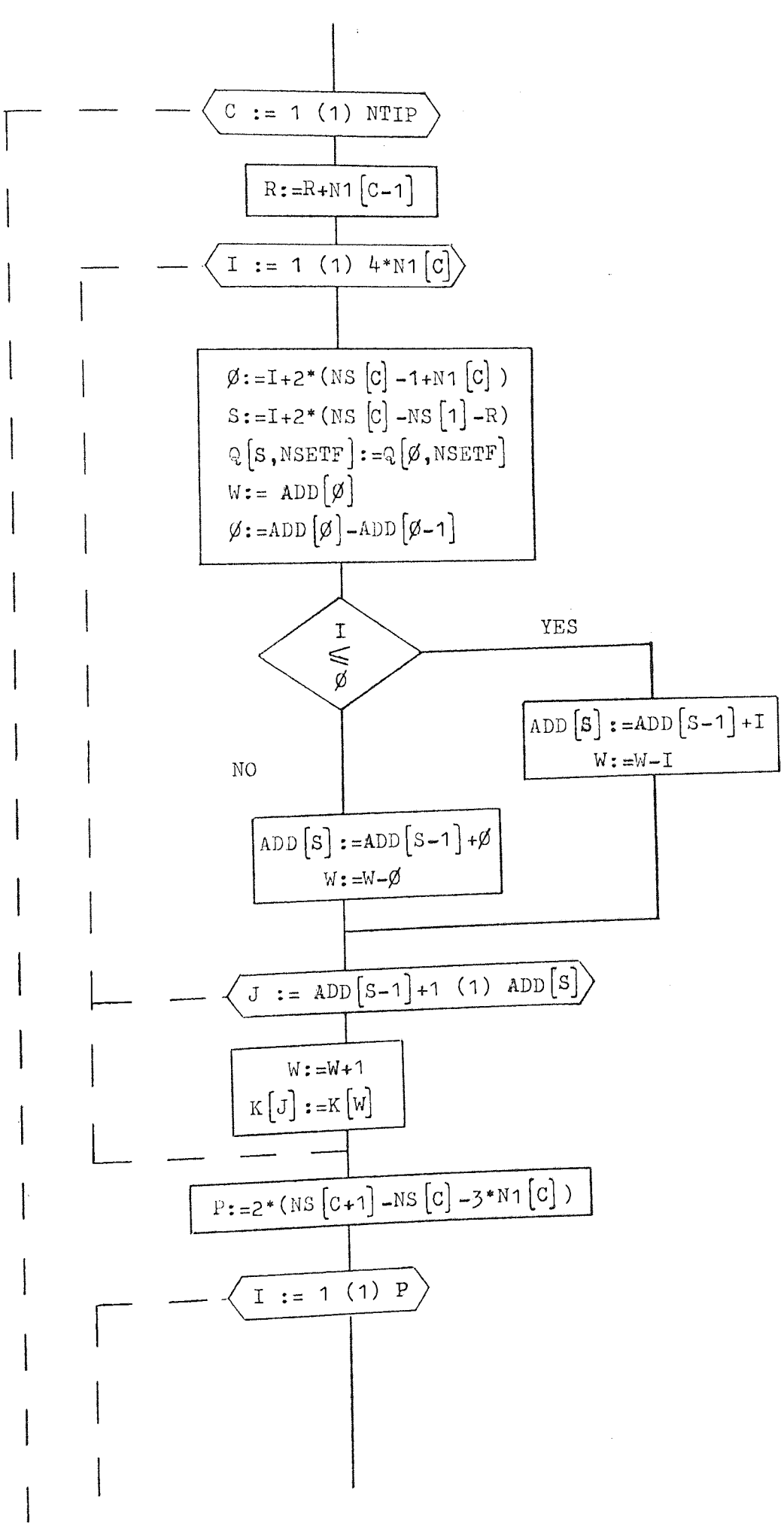

Z := 1 (1) N1[C]
TA:=-PYE+2*(Z-1)*PYE/(N1[C]-1)
F[1,1]:=-RO[C]*SIN(TA)
F[1,2]:= RO[C]*COS(TA)
F[2,2]:=F[3,1]:=0.0
F[2,1]:=F[3,2]:=1.0
I:=0
6.
J := 4 (2) ELDOF+1
F[J,1]:=(((KAP+I/2+(-1)↑I)*COS(I*TA/2)
-(I/2)*COS(TA*(I/2-2)))*RO[C]↑(I/2))/(2*G)
F[J+1,1]:=-(((KAP+I/2-(-1)↑I)*SIN(I*TA/2)
-(I/2)*SIN(TA*(I/2-2)))*RO[C]↑(I/2))/(2*G)
F[J,2]:=(((KAP-I/2-(-1)↑I)*SIN(I*TA/2)
+(I/2)*SIN(TA*(I/2-2)))*RO[C]↑(I/2))/(2*G)
F[J+1,2]:=(((KAP-I/2+(-1)↑I)*COS(I*TA/2)
+(I/2)*COS(TA*(I/2-2)))*RO[C]↑(I/2)) /(2*G)
F[4,1]:=F[4,1]*0.5↑0.5
F[4,2]:=F[4,2]*0.5↑0.5
F[5,1]:=-F[5,1]*0.5↑0.5
F[5,2]:=-F[5,2]*0.5↑0.5
7.
W := 1 (1) ELDOF
FI[Z*2-1,W]:=F[W,1]*CN-F[W,2]*SN
FI[Z*2,W] :=F[W,1]*SN+F[W,2]*CN
8.
Z := 1 (1) ELDOF
S:=Z+ELDOF*(C-1)

9.

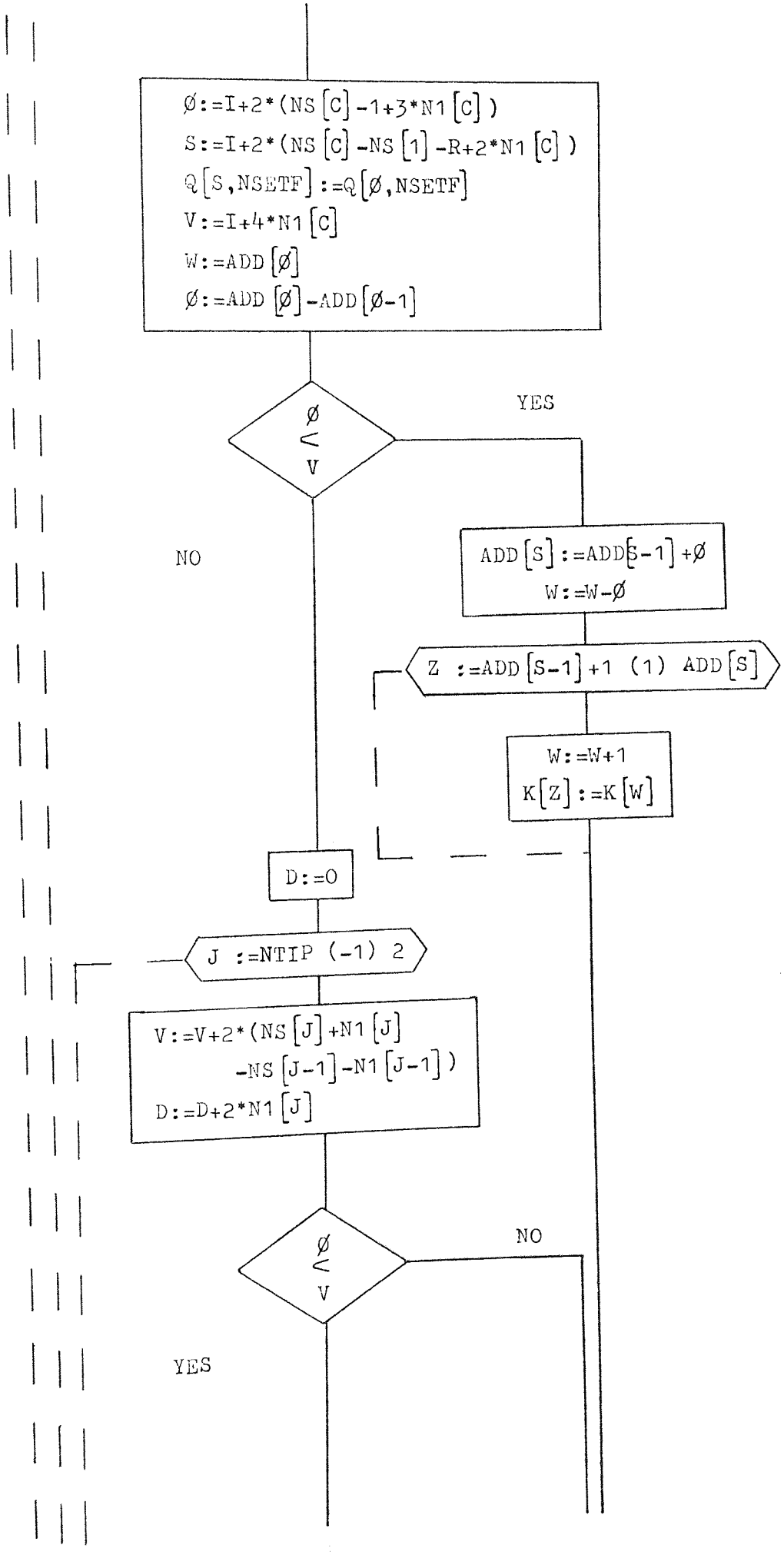

W := 1 (1) Z
P:=W+4*N1[C]
I := 1 (1) 2*N1[C]
Ø:=I+2(NS[C]-1)
J := 1 (1) 2*N1[C]
R:=J+2*(NS[C]-1)
R > Ø
NO
YES
Ø-R+1 > ADD[Ø]-ADD[Ø-1]
R-Ø+1 > ADD[R]-ADD[R-1]
YES
NO
NO
YES
KA:=0.0
KA:=K[ADD[Ø]-Ø+R]
KA:=K[ADD[R]-R+Ø]
KA:=0.0
KC[S,P]:=KC[S,P]+(FI[I,Z]*KA*FI[J,W])
KC[S,P]:=KC[S,P]+KT[Z,W]


W := 1 (1) 4*N1[C]
P:=W+2*(NS[C]-1+N1[C])
I := 1 (1) ELDOF
Ø:=I+ELDOF*(C-1)
J := 1 (1) 2*N1[C]
R:=J+2*(NS[C]-1)
10.
P-R+1 > ADD[P]-ADD[P-1]
YES
KA:=0.0
NO
KA:=K[ADD[P]-P+R]
KC[Ø,W]:=KC[Ø,W]+(KA*FI[J,I])
V:=R:=0
N1[0]:=0
NS[NTIP-1]:=NNODE+1

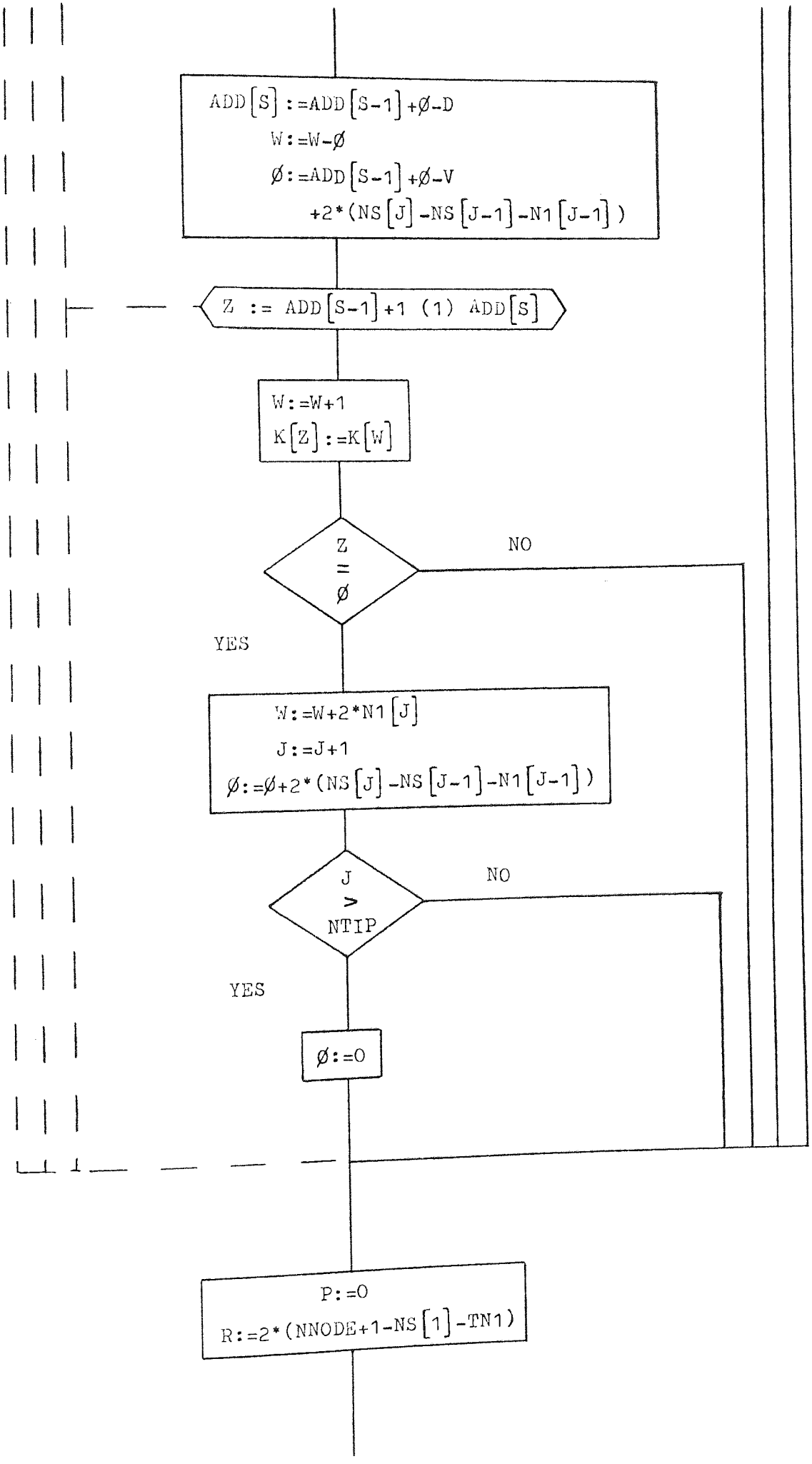
11.


C := 1 (1) NTIP
R:=R+N1[C-1]
I := 1 (1) 4*N1[C]
Ø:=I+2*(NS[C]-1+N1[C])
S:=I+2*(NS[C]-NS[1]-R)
Q[S,NSETF]:=Q[Ø,NSETF]
W:= ADD[Ø]
Ø:=ADD[Ø]-ADD[Ø-1]
I ≤ Ø
YES
ADD[S]:=ADD[S-1]+I
W:=W-I
NO
ADD[S]:=ADD[S-1]+Ø
W:=W-Ø
J := ADD[S-1]+1 (1) ADD[S]
W:=W+1
K[J]:=K[W]
P:=2*(NS[C+1]-NS[C]-3*N1[C])
I := 1 (1) P

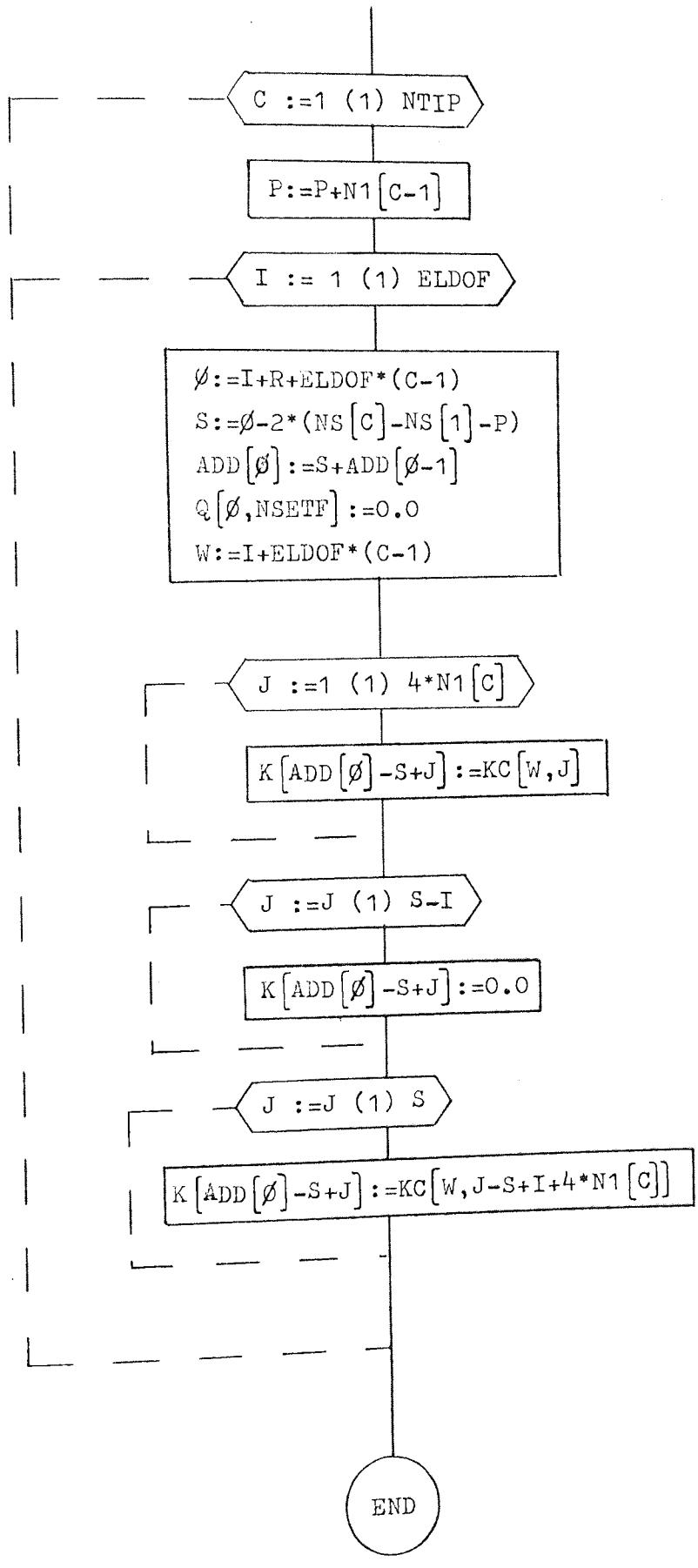
12.


Ø:=I+2*(NS[C]-1+3*N1[C])
S:=I+2*(NS[C]-NS[1]-R+2*N1[C])
Q[S,NSETF]:=Q[Ø,NSETF]
V:=I+4*N1[C]
W:=ADD[Ø]
Ø:=ADD[Ø]-ADD[Ø-1]
Ø ≤ V
YES
NO
ADD[S]:=ADD[S-1]+Ø
W:=W-Ø
Z :=ADD[S-1]+1 (1) ADD[S]
W:=W+1
K[Z]:=K[W]
D:=0
J :=NTIP (-1) 2
V:=V+2*(NS[J]+N1[J]
-NS[J-1]-N1[J-1])
D:=D+2*N1[J]
Ø ≤ V
NO
YES

12.


ADD[S] :=ADD[S-1] +Ø-D
W:=W-Ø
Ø:=ADD[S-1] +Ø-V
+2*(NS[J] -NS[J-1] -N1[J-1])
Z := ADD[S-1] +1 (1) ADD[S]
W:=W+1
K[Z] :=K[W]
Z = Ø
NO
YES
W:=W+2*N1[J]
J:=J+1
Ø:=Ø+2*(NS[J] -NS[J-1] -N1[J-1])
J > NTIP
NO
YES
Ø:=0
P:=0
R:=2*(NNODE+1-NS[1] -TN1)

13.


C :=1 (1) NTIP
P:=P+N1[C-1]
I := 1 (1) ELDOF
Ø:=I+R+ELDOF*(C-1)
S:=Ø-2*(NS[C]-NS[1]-P)
ADD[Ø]:=S+ADD[Ø-1]
Q[Ø,NSETF]:=0.0
W:=I+ELDOF*(C-1)
J :=1 (1) 4*N1[C]
K[ADD[Ø]-S+J]:=KC[W,J]
J :=J (1) S-I
K[ADD[Ø]-S+J]:=0.0
J :=J (1) S
K[ADD[Ø]-S+J]:=KC[W,J-S+I+4*N1[C]]
END

7.2.4 NUMERICAL EXAMPLES

It is the purpose of this section to illustrate the application of the multi-tip mixed mode program by obtaining numerical solutions for a number of specific crack problems, some of which have been solved previously by other methods. The problems are chosen to demonstrate the accuracy and flexibility of the technique for arbitrary two-dimensional component shapes.

The first example, shown in figure (7.5), is that of a rectangular plate subject to uniaxial tension and containing a 45 degree central crack. The theoretical solution for the infinite plate is taken from Rice[78], where, for tips 1 and 2,

$$K_I = \sigma\sqrt{a}\,\sin^2\alpha$$
$$K_{II} = \sigma\sqrt{a}\,\sin\alpha\,\cos\alpha \qquad (7.6)$$

The plate dimensions have been chosen such that they reasonably simulate an infinite plate. For the value of α = 45 degrees, equation (7.6), yields,

$$K_I/\sigma\sqrt{a} = K_{II}/\sigma\sqrt{a} = 0.5$$

The corresponding finite element results were,

$$K_{I_1}/\sigma\sqrt{a} = K_{I_2}/\sigma\sqrt{a} = 0.51$$

and

$$K_{II_1}/\sigma\sqrt{a} = K_{II_2}/\sigma\sqrt{a} = 0.51$$

where subscripts 1 and 2 refer to the crack tips and (a) is the half

crack length. The number of elements and nodes used in the discretised problem were 100 and 344 respectively. Using the 'transition' elements around the core element would have reduced these figures to 68 and 240, representing a considerable saving in computer storage space. Refer to section (7.5).

The second example, illustrated in figure (7.6), is that of a plate in uniform tension containing three 90 degree edge cracks. The problem was one of several used to prove the multi-tip procedure and the theoretical solutions are given in Rooke and Cartwright[79]. The value of the shearing stress intensity factor is given as zero and the mode I factors are, for the outer cracks,

$$K_I/\sigma\sqrt{\pi a} = 0.8925$$

and the inner crack,

$$K_I/\sigma\sqrt{\pi a} = 0.822$$

Note that Rooke and Cartwright's definition of Irwin's equation (6.28) contains a $1/\sqrt{\pi}$ factor. The corresponding finite element results were,

TIP 1. $K_I/\sigma\sqrt{a} = 0.8684$

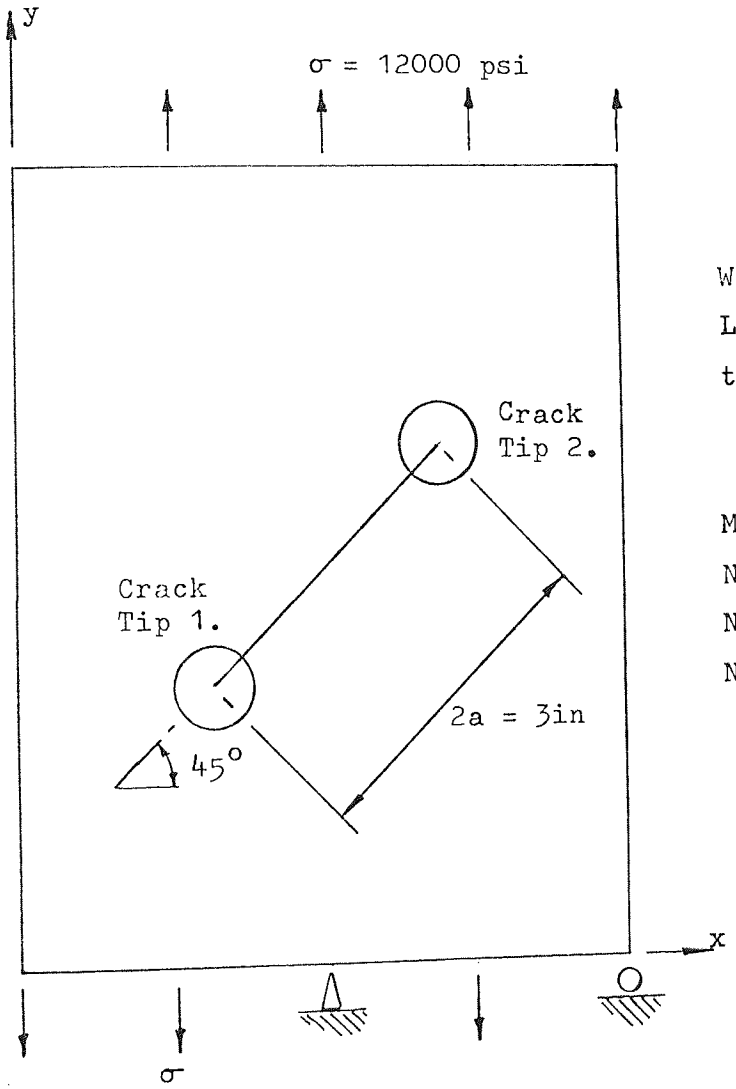
$K_{II}/\sigma\sqrt{a} = -0.0244$

TIP 2. $K_I/\sigma\sqrt{a} = 0.7981$

$K_{II}/\sigma\sqrt{a} = -0.00079$

TIP 3. $K_I/\sigma\sqrt{a} = 0.8585$

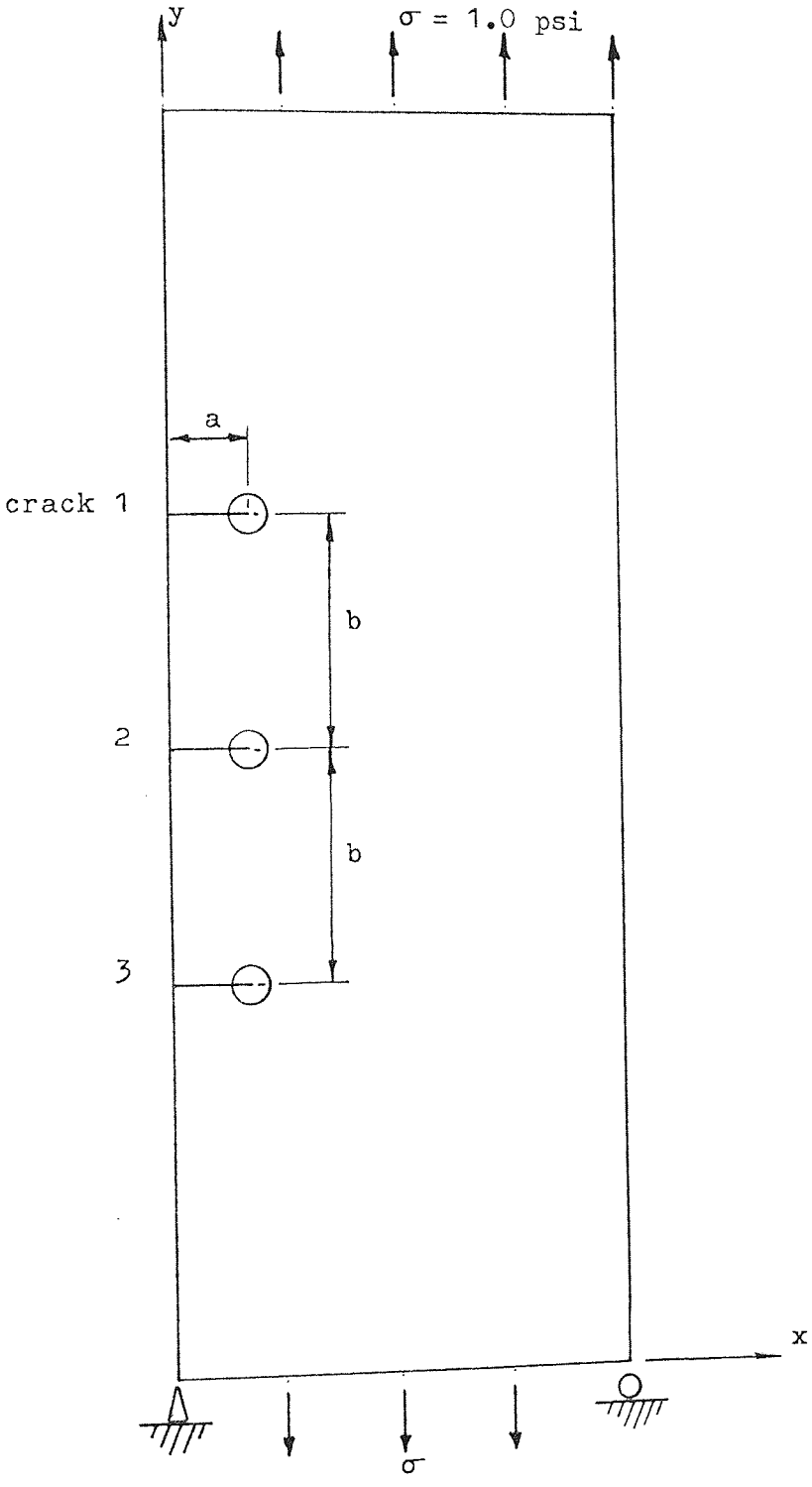
$K_{II}/\sigma\sqrt{a} = 0.0216$



W = 16 in
L = 20 in
t = 1.0 in

Mesh details
No. interface nodes - 17
No. elements - 100
No. nodes - 344

FIG 7.5 A rectangular plate in uniaxial tension, containing a 45^{o} degree central crack.



a = 0.9 in
b = 3.0 in
W = 6.0 in
L = 16 in
t = 1.0 in

Mesh details
No. interface nodes - 13
No. elements - 80
No. nodes - 299

FIG 7.6 A plate containing three 90^{o} degree edge cracks

The overall discrepancy is -3.0% despite using only 13 core/mesh interface nodes. Normally 17 interface nodes are used to reduce the displacement disconitnuity between the core and finite element mesh, refer to section (7.5). The finite element K_{II} values are extremely small compared with the K_I results, however, note that the outer cracks are subject to a higher shear stress as denoted by the larger K_{II} values.

A shouldered plate having a small crack on the fillet radius, is taken as the third example. The site of the crack has been chosen to correspond with the point of maximum stress concentration. The plate is in uniform tension and is illustrated in figure (7.7). An extremely fine mesh was used in this problem, resulting in 541 nodes and 244 elements; 21 interface nodes were taken around the core element. The normalized stress intensity factor for the finite element results were,

$$K_I/\sigma_n\sqrt{a} = 1.644$$

and

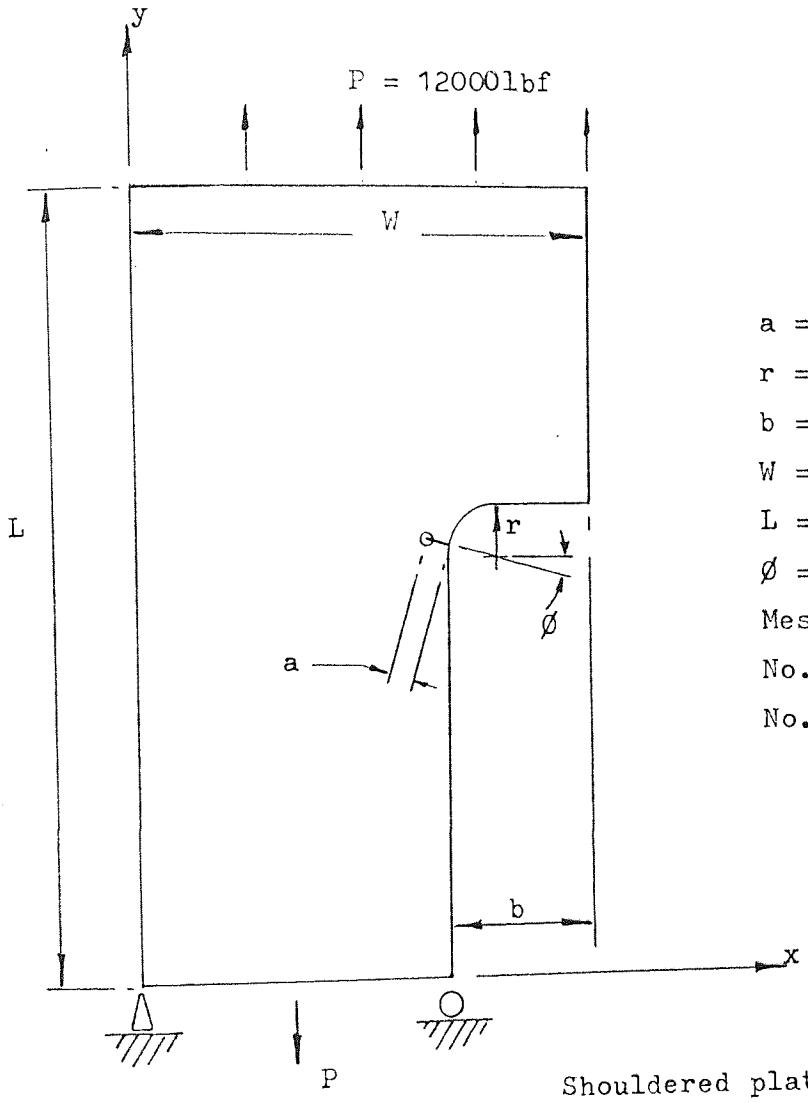
$$K_{II}/\sigma_n\sqrt{a} = 0.5169$$

where $a = 0.5$ and σ_n is the applied stress across the reduced cross-section. As there is no theoretical solution available for this case a comparison can be made with a rectangular plate containing a similar crack configuration, as shown in figure (7.7). The loading distribution corresponds with that found around the crack area, and the following solutions were obtained from reference (79),

$$K_I/\sigma_n\sqrt{a} = 1.821$$

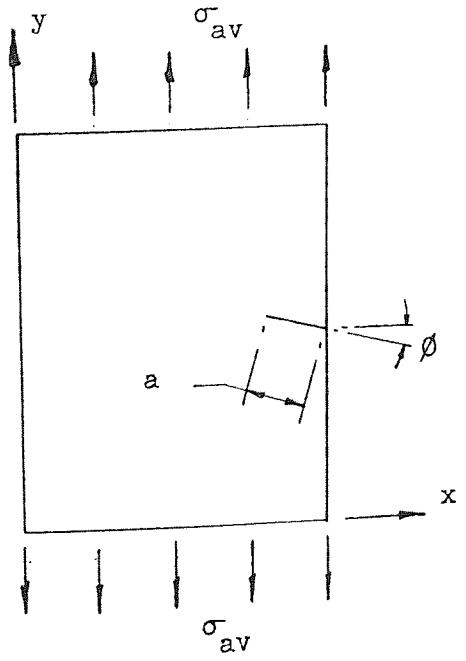
$$K_{II}/\sigma_n\sqrt{a} = 0.264$$

FIG 7.7



a = 0.5 in
r = 2.5 in
b = 5.0 in
W = 15 in
L = 25 in
$\emptyset = 15^{o}$
Mesh details
No. elements - 244
No. nodes - 541

Shouldered plate containing a small crack on the fillet radius.



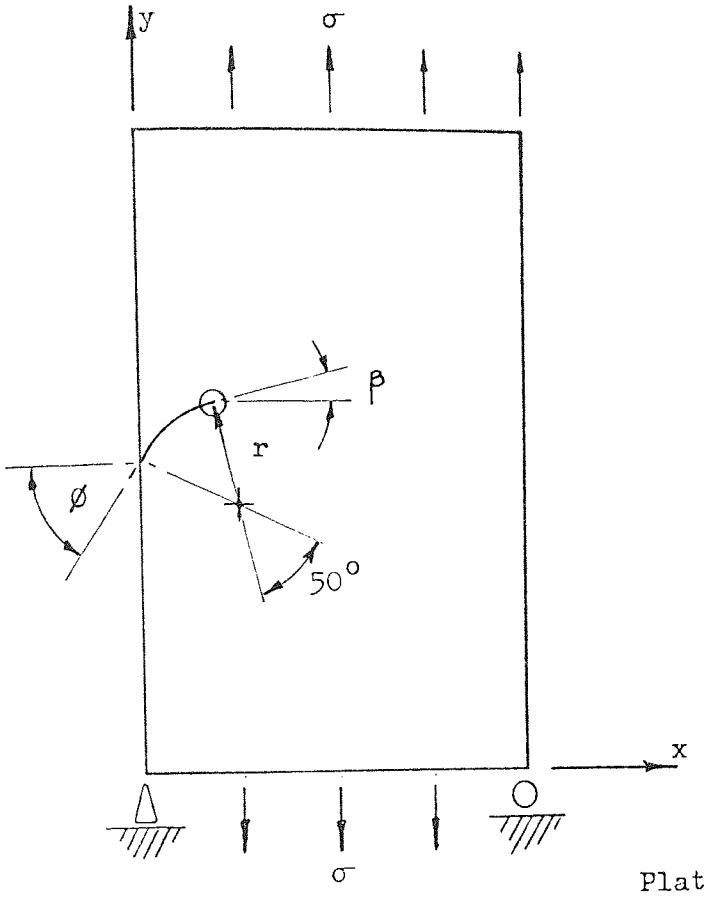
W = 5.0 in
L = 10.0 in
$\emptyset = 15^{o}$
$\sigma_{av} = 19 \times 10^{3}$ psi

Comparison idealized case.

The results from the idealized case indicate that the finite element solutions for the shouldered plate are at least of the same order of magnitude and gives some confidence in the finite element solution.

As a final example, the case of a rectangular plate containing an arcular edge crack, with various root angles, was examined. The plate is in uniform tension and the crack configuration is shown in figure (7.8). Again a relatively fine mesh has been chosen and the details are given in the diagram. Here, also, there is no theoretical solution and to gauge the validity of the results an idealized case has been chosen of a plate with a straight edge crack, with a root angle corresponding to the final inclination angle of the curved crack; that is angle β. The results determined using the finite element method and those of the idealized case are shown in Table (7.9) and are illustrated graphically in figure (7.10). Comparing the two sets of results it can be seen that they are of the same order and the plotted curves follow the same general pattern. From this information it is assumed that the finite element results are representative of the true solutions.

Other facilities, such as the partial crack closure scheme written into the program, are demonstrated in later examples.



σ = 12000 psi

a = 1.745329 in = crack length

r = 2.0 in

L = 2W = 10.0 in

t = 1.0 in

Mesh details

No. elements - 132

No. nodes - 299

Rc = 0.036 in

Plate in uniform tension containing an arcular edge crack.

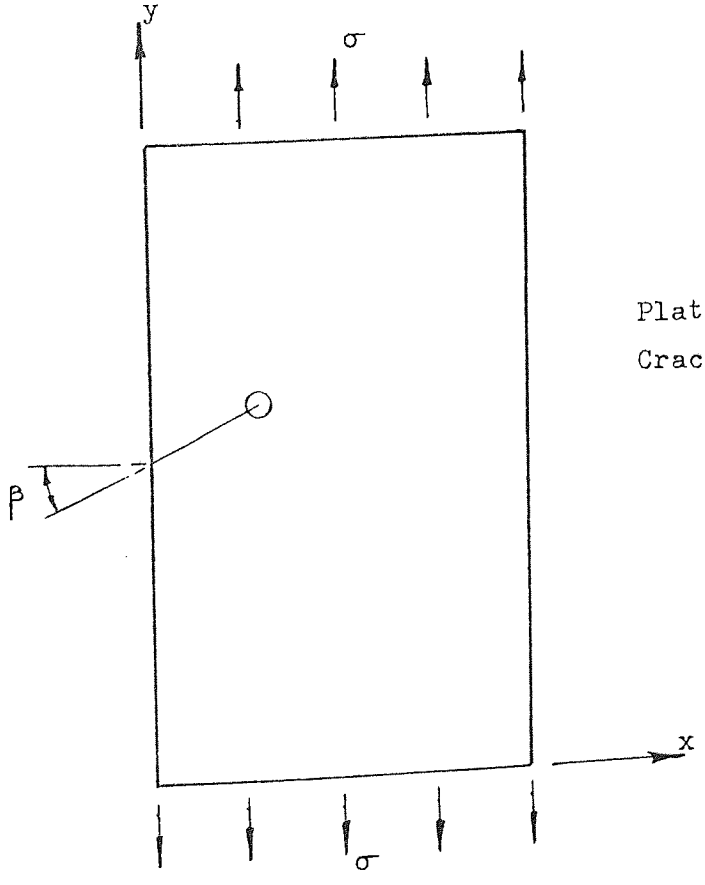


Plate dimensions as above.

Crack length = a

Idealized cracked structure.

FIG 7.8

Finite Element Results			
ϕ root angle degrees	$K_{I}/\sigma\sqrt{a}$	$K_{II}/\sigma\sqrt{a}$	β degrees
30^{o}	1.33203	0.21394	75^{o}
45^{o}	1.56809	-4.132×10^{-3}	90^{o}
60^{o}	1.64278	-0.2418	105^{o}
75^{o}	1.493	-0.4608	120^{o}
90^{o}	1.1896	-0.587	135^{o}
Comparison results for a straight crack of the same length, at root angle 'β'. REF(16)			
root angle degrees	$K_{I}/\sigma\sqrt{a}$	$K_{II}/\sigma\sqrt{a}$	
75^{o}	1.75	0.23	
90^{o}	1.909	0.0	
105^{o}	1.75	0.23	
120^{o}	1.45	0.44	
135^{o}	1.0	0.47	

TABLE 7.9. Normalized K_{I} and K_{II} values for a plate with a curved edge crack at various root angles. See figure(7.8).

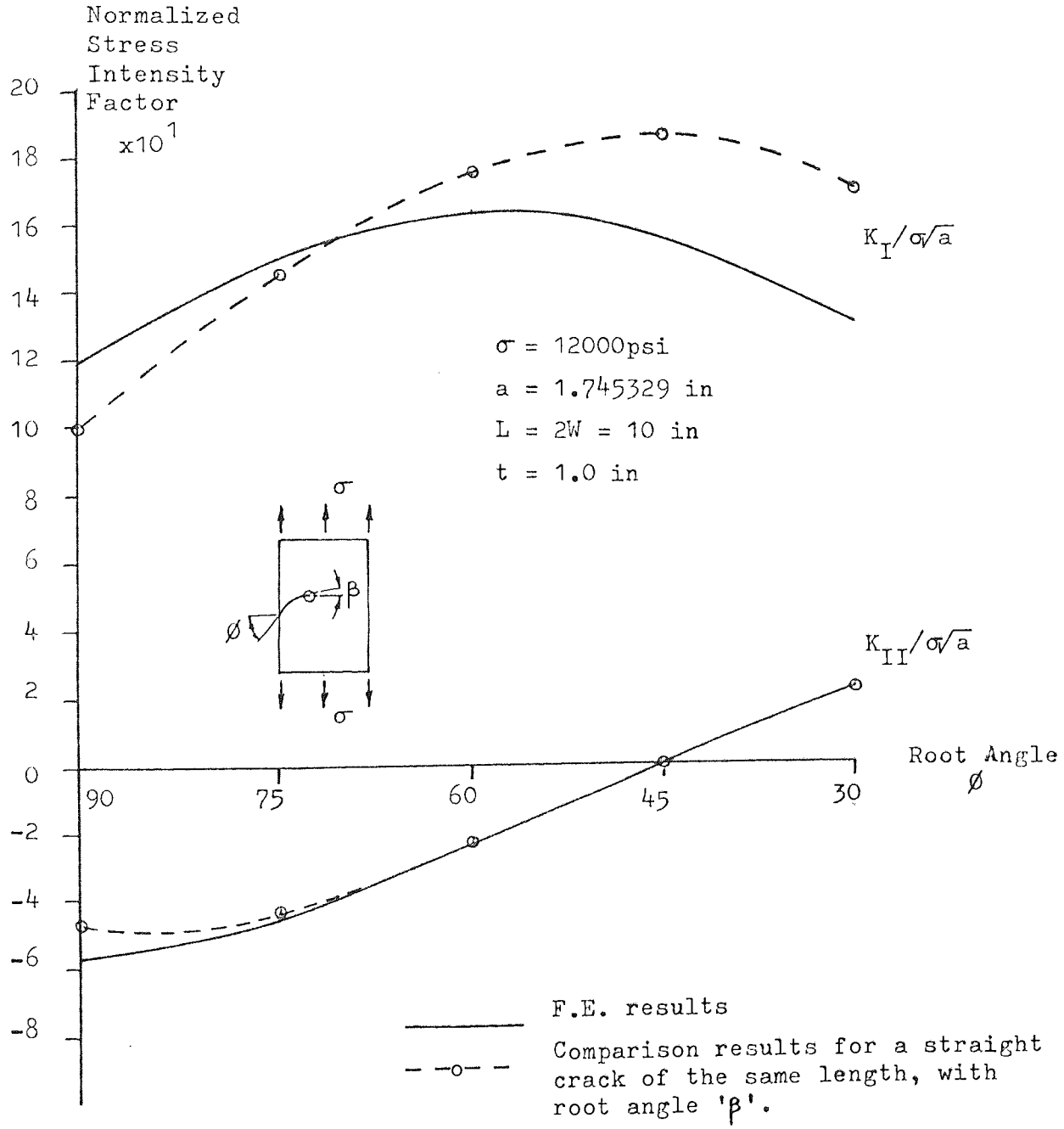


FIG 7.10 Non-dimensional K_I and K_{II} results for a plate in uniform tension containing an arcular edge crack.

7.3 REFINEMENT OF THE CORE ELEMENT DISPLACEMENT MODEL

7.3.1 INTRODUCTION

The fracture problems solved using finite element analysis, in the early investigations, suggested that a large number of nodes or degrees of freedom were required in the discretized model in order to obtain a satisfactory solution. As a result the computer storage space was overloaded and ways of overcoming this situation were examined. A direct approach was to introduce a segmented solving routine, where the large backing store could be utilized. Alternatively the size of the storage requirements could be reduced and methods which may bring about this situation are discussed in this section.

In general, it was found that the Hilton and Hutchinson method, as described in Chapter six, increased the size of the stiffness matrix. In other words the modified stiffness matrix [K*] was larger than the original [K]. This aggravates the computer storage problem and could be avoided by firstly, reducing the number of elements used in the finite element model and secondly, by changing the method of solution. The first proposal lead to the idea of enlarging the core region, and thus eliminating the fine local crack tip mesh. This would lead to a reduction in the core elements ability to yield accurate stress intensity factors and consequently, to maintain the solution accuracy, more terms would have to be taken in the singularity field equations. The second approach involves the modification of the matrix equations (6.47) and the introduction of what will be termed the 'pseudo inverse' method. A combination of both schemes would produce an efficient and accurate program for the solution of fracture problems. The two methods are described in the following sections.

7.3.2 PSEUDO INVERSE METHOD

This method has been briefly discussed in Chapter two, in connection with the classical solution/finite element techniques. Essentially the method hinges on the transformation of the matrix equation (6.38), given here as,

$$\{q_1\} = [A]\{\alpha\} \tag{7.7}$$

where vector $\{q_1\}$ represents the interface nodal displacements, and $\{\alpha\}$ contains the stress intensity factors. The connecting matrix [A] is generally not square and equation (7.7) represents an overdetermined system of equations. The transformation of equation (7.7) requires the inverse of matrix [A], and this can be found using a least squares technique similar to that of Businger and Golub[80] or Peters and Wilkinson[81]. Returning to the system equations (6.43 - 6.44), rewritten here as,

$$[k_c]\{\alpha\} - [A]^t\{\lambda\} = 0 \tag{7.8}$$

$$[K_{11}]\{q_1\} + [K_{12}]\{q_2\} - \{Q_1\} + \{\lambda\} = 0 \tag{7.9}$$

$$[K_{21}]\{q_1\} + [K_{22}]\{q_2\} - \{Q_2\} = 0 \tag{7.10}$$

Transposing the equation (7.7)and substituting the vector $\{\alpha\}$ into equation (7.8), yields

$$[A^*]^t[k_c][A^*]\{q_1\} = \{\lambda\} \tag{7.11}$$

where [A*] is the pseudo inverse of matrix [A], and the matrices are as given in Chapter six. Replacing the Lagrange multipliers $\{\lambda\}$ of equation (7.9), with that of equation (7.11), results in the modified system,

$$[K']\{q\} = \{Q\} \tag{7.12}$$

where

$$[K'] = \begin{bmatrix} ([A^*]^t[k_c][A^*] + [K_{11}]), & [K_{12}] \\ [K_{21}], & [K_{22}] \end{bmatrix} \qquad (7.13)$$

and vector $\{q\}$ and $\{Q\}$ are the nodal displacements and nodal forces respectively. The modified stiffness matrix $[K']$ has the same dimensions as the original matrix $[K]$, and is of a simpler form compared with $[K^*]$ of equation (6.49). The solution of equation (7.12) however, does not yield the wanted stress intensity factors directly and it is necessary to back substitute the interface nodal displacements into the transformed equation (7.7), that is,

$$\{\alpha\} = [A^*]\{q_1\}$$

This method has been successfully employed by Jones and Callinan[50].

The essential modifications to the fracture program are as follows:

1. The sub-matrix $[K_{11}]$ may be sparsely populated, hence, the addition of the sub-matrix $([A^*]^t[k_c][A^*])$ could violate the variable bandwidth storage scheme. To avoid this situation the address sequence is preset to accept the additional coefficients.
2. The pseudo inverse matrix $[A^*]$ must be determined and the sub-matrix $[A^*]^t[k_c][A^*]$ formed and added to the overall stiffness matrix.
3. Finally the interface nodal displacements are used to interpret the wanted stress intensity factors through the appropriate expression.

The following section examines the procedures required to implement these various steps.

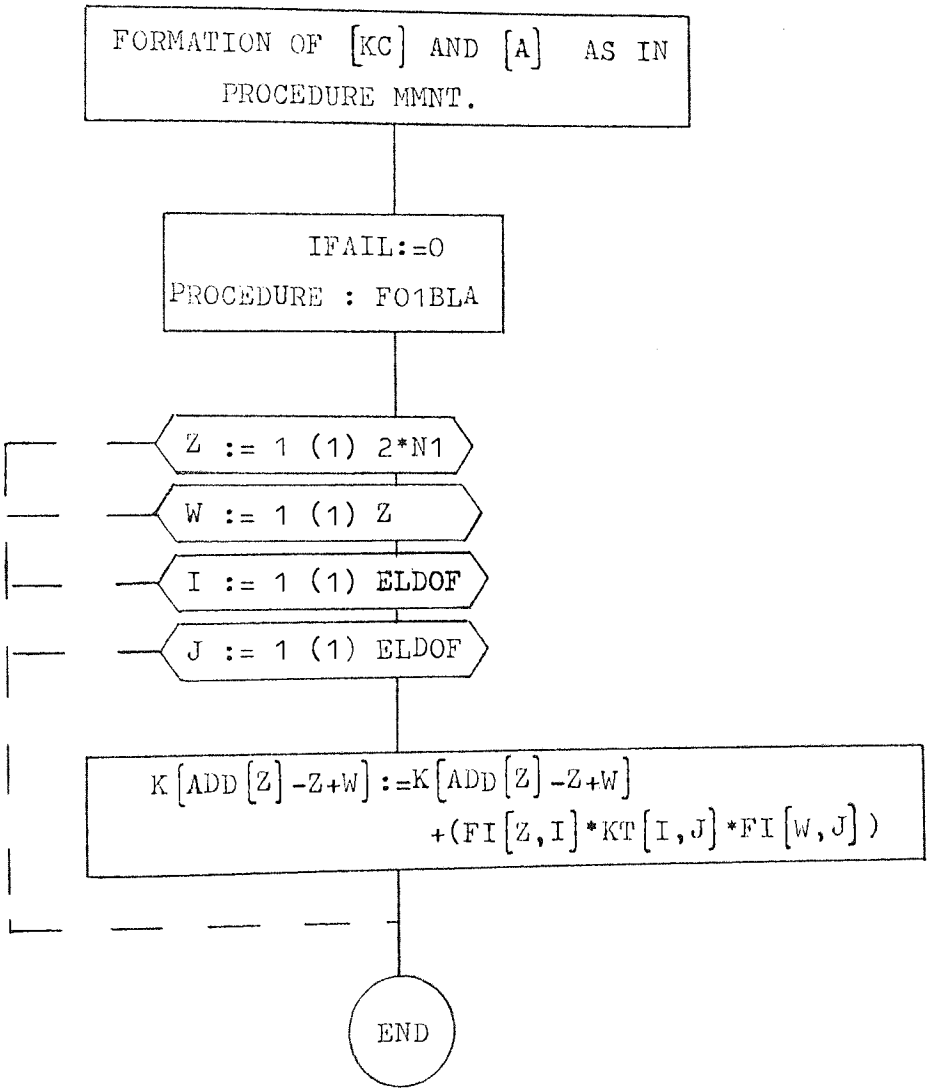
7.3.3 PSEUDO INVERSE PROCEDURE DESCRIPTIONS

PROCEDURE MIIST:

This procedure is equivalent to the multi-tip routine of section (7.2.3), and matrices $[k_c]$ and [A] are formed using the same computer coding. The matrix handling scheme is different however, and the NAG sub-routine F01BLA is used to determine the inverse matrix [A*]. The formation of the sub-matrices $([A^*]^t[k_c][A^*])$ and its subsequent addition to the overall stiffness matrix is described by the following steps which refer to the flowchart overleaf:-

1. The NAG subroutine F01BLA is called and computes the pseudo inverse matrix [A*], storing it in matrix [FI].
2. A looping system is constructed which effectively forms each coefficient of sub-matrix $([A^*]^t[k_c][A^*])$, and adds it to the overall stiffness matrix [K].

Flowchart for procedure MIIST :-



PROCEDURE SOLVIT:-

The solution of the modified stiffness equations (7.12) yields the nodal displacements of the finite element mesh. The interpretation if the interface nodal displacements, through the equation (7.7), gives the wanted stress intensity factors, thus,

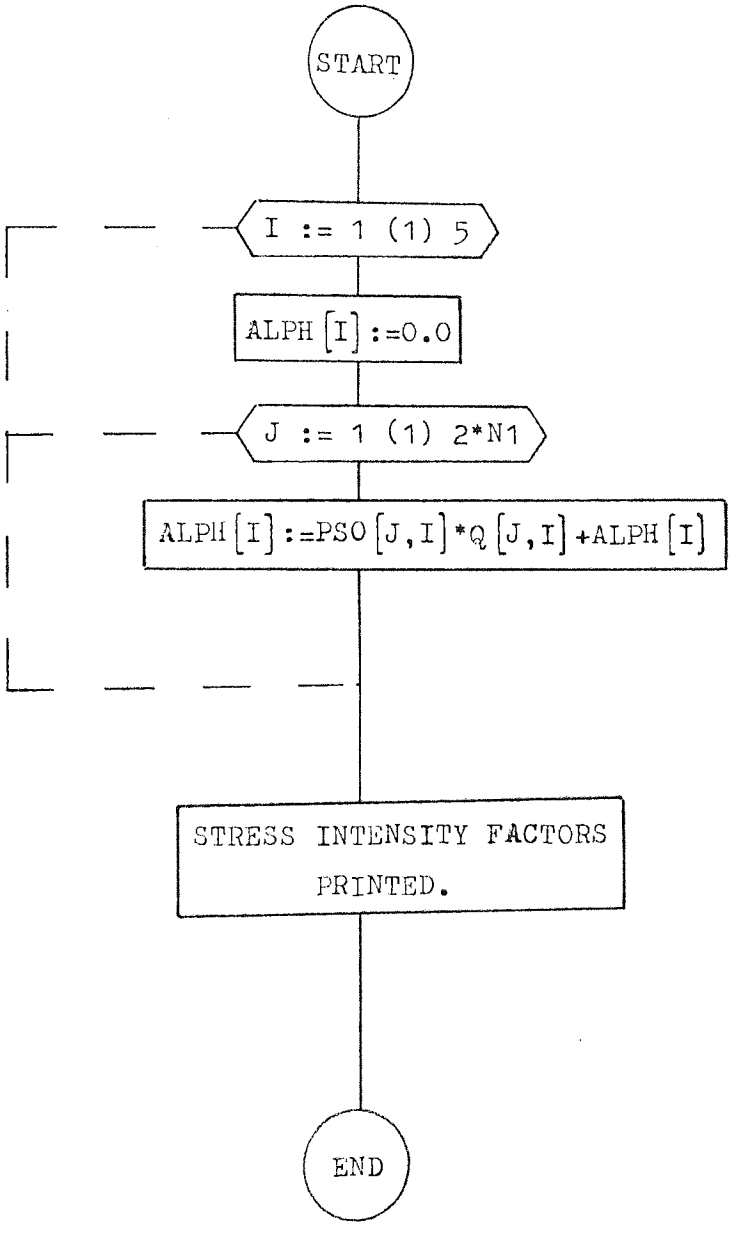
$$\{\alpha\} = [A^*]\{q_1\} \qquad (7.14)$$

The back-substitution process is relatively simple and is summarized in the following steps which correspond with the flowchart:-

1. The computed stress intensity factors and the rigid body displacement will be stored in matrix [ALPHA]. Array [PSO] represents the pseudo inverse matrix [A*] and the back substitution process is accomplished within the loops I and J.

2. The determined stress intensity factors are printed.

Flowchart for procedure SOLVIT :



1.

2.

7.3.4 CORE ELEMENT STIFFNESS MATRIX

Previously only the first two terms were used in the crack tip singularity expressions. By including extra terms, the solution accuracy will improve and a larger core element can be employed. The coefficients of the core stiffness matrix $[k_c]$ were found using the first two terms of the stress and strain near crack tip field expressions, and the strain energy integral equation. In refining the core element, the number of terms taken in the series expansion was made adjustable, hence, the stress and strain expression of equation (6.23) were used directly. The strain energy stored in the crack tip core element is given by the expression,

$$U_c = \int_{vol} \frac{1}{2}(\sigma_r\varepsilon_r + \sigma_\theta\varepsilon_\theta + \tau_{r\theta}\gamma_{r\theta})dvol \qquad (7.15)$$

where r and θ are the polar coordinates at the crack tip.

For the mixed mode case a circular core element is used and the integration limits are,

$$U_c = \frac{t}{2}\int_0^{2\pi}\int_0^{r}(\sigma_r\varepsilon_r + \sigma_\theta\varepsilon_\theta + \tau_{r\theta}\gamma_{r\theta})rdrd\theta \qquad (7.16)$$

Now consider the first term in the above expression and let subscripts n and m refer to σ_r and ε_r; we have, using the equations of Appendix (9.1),

$$\frac{t}{2}\int_0^{2\pi}\int_0^{r}(\sigma_r\varepsilon_r)rdrd\theta = \frac{t}{2}\sum_{n=1}^{\infty}\sum_{m=1}^{\infty}\{\frac{n}{2}r^{(n/2-1)}[\alpha_{2n-1}C - \alpha_{2n}D] \times \frac{mr}{4\mu}^{(m/2-1)}[\alpha_{2m-1}I + \alpha_{2m}J]\}rdrd\theta \qquad (7.17)$$

where C, D, I and J are functions of θ, μ is the shear modulus, r is the core element radius and the α's are constants. α_1 and α_2 are connected with the stress intensity factors via equation (6.28). Evaluating the first integral yields,

$$\sum_{n=1}^{\infty}\sum_{m=1}^{\infty}\frac{t}{8\mu}\frac{mn}{(n+m)}r^{(n+m)/2}\int_0^{2\pi}\{[\alpha_{2n-1}C-\alpha_{2n}D][\alpha_{2m-1}I+\alpha_{2m}J]\}d\theta \quad (7.18)$$

Examining the expression under the integral and expanding, we have,

$$\int_0^{2\pi}\{\alpha_{2n-1}\alpha_{2m-1}C.I+\alpha_{2n-1}\alpha_{2m}C.J - \alpha_{2n}\alpha_{2m-1}D.I-\alpha_{2n}\alpha_{2m}D.J\}\,d\theta \quad (7.19)$$

Expanding the first term C.I, using the definitions given in Appendix (9.1), we have

$$\alpha_{2n-1}\alpha_{2m-1}\int_0^{2\pi}\{[C_4\cos\theta(C_5)+C_3\cos\theta(C_1)]\times[C_6\cos\theta(C_{10})+C_8\cos\theta(C_{11})]\}d\theta \quad (7.20)$$

where the functions $C_i(n,m)$ are also tabulated in the Appendix (9.1) Multiplying the terms and integrating, yields,

$$\begin{aligned}\alpha_{n-1}\alpha_{2m-1}\int_0^{2\pi}(C.I)d\theta = \\ \frac{1}{2}\{C_4C_6[\frac{1}{C_5+C_{10}}\sin\theta(C_5+C_{10})+\frac{1}{C_5-C_{10}}\sin\theta(C_5-C_{10})] \\ +C_4C_8[\frac{1}{C_5+C_{11}}\sin\theta(C_5+C_{11})+\frac{1}{C_5-C_{11}}\sin\theta(C_5-C_{11})] \\ +C_3C_6[\frac{1}{C_1+C_{10}}\sin\theta(C_1+C_{10})+\frac{1}{C_1-C_{10}}\sin\theta(C_1-C_{10})] \\ +C_3C_8[\frac{1}{C_1+C_{11}}\sin\theta(C_1+C_{11})+\frac{1}{C_1-C_{11}}\sin\theta(C_1-C_{11})]\}_0^{2\pi}\end{aligned} \quad (7.21)$$

Repeating the process for each term in equation (7.16), an expression can be found in terms of the integers m and n for the strain energy of the crack tip core elements. Thus by simply substituting the appropriate values of m and n in the strain energy expression, any coefficient of matrix $[k_c]$ can be found. The strain energy expression is listed in the Appendix (9.1). The embodiment of this expression within the sub-routine CORK is given in the next section.

7.3.5 DESCRIPTION OF PROCEDURE CORK

The final expression for the core element strain energy, is rather cumbersome and wherever possible repetitive terms are assigned the same variable name. The functions C_i of expression (7.21) may be computed as zero subject to the values of integers n and m. In this case a cosine term with argument C_i, under the integration sign, would yield unity and on integration θ. Similarly a sine term with a zero argument would be left out of the expression. As the values of the functions $C_i(n,m)$ are dependent on the values of integers n and m, a conditional statement is used to detect any zero terms and override the sine and cosine expressions. Specifying integers n and m generates a 2 x 2 sub-matrix of the core stiffness array $[k_c]$ and its position within the array is dependent on the position of the α's in the vector $\{\alpha\}$.

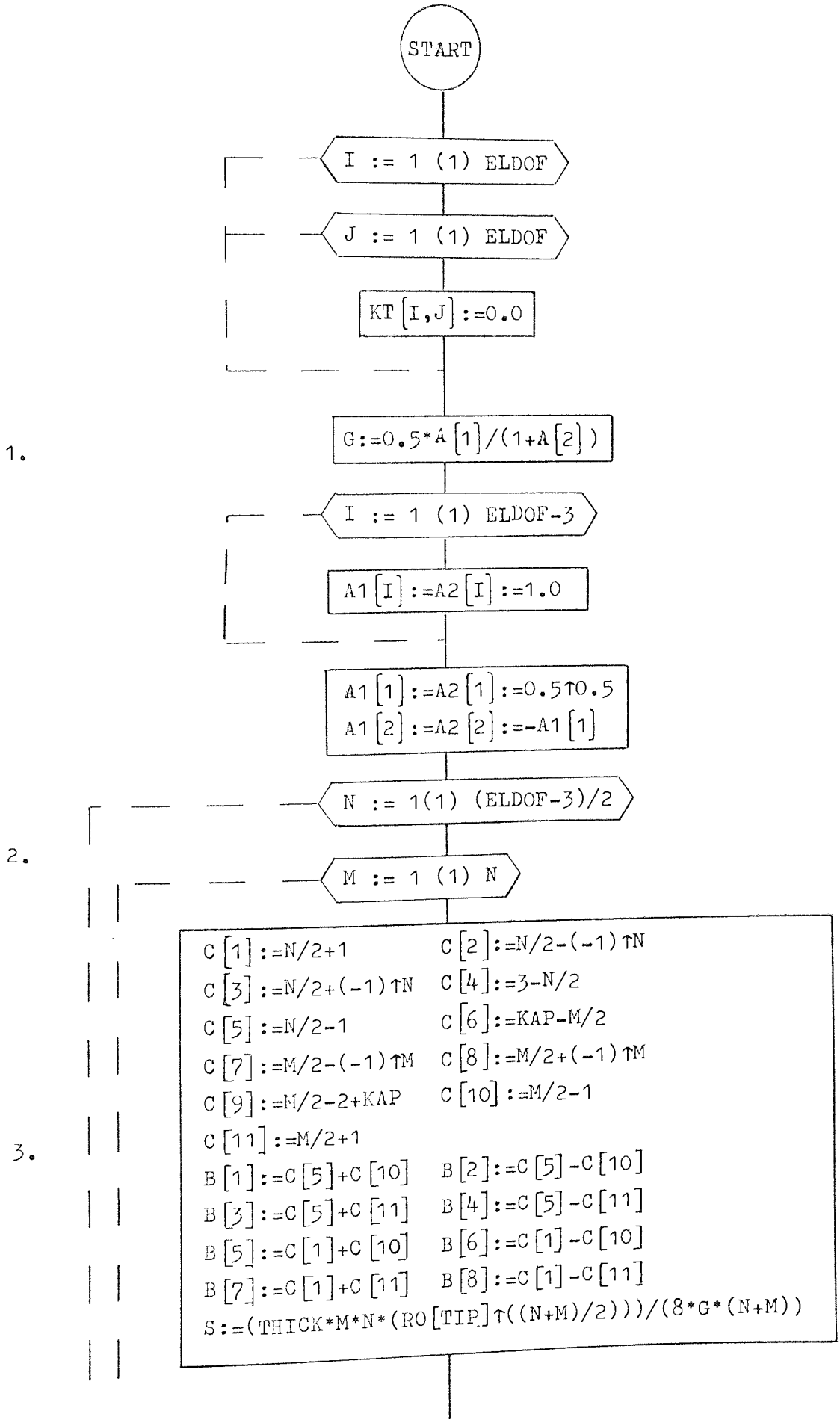
There is no reason why this formulation cannot be used in the case of mode I fracture, that is with a semi-circular core element, simply by changing the upper integral limit to π. In a general fracture program this procedure could be used to examine mode I

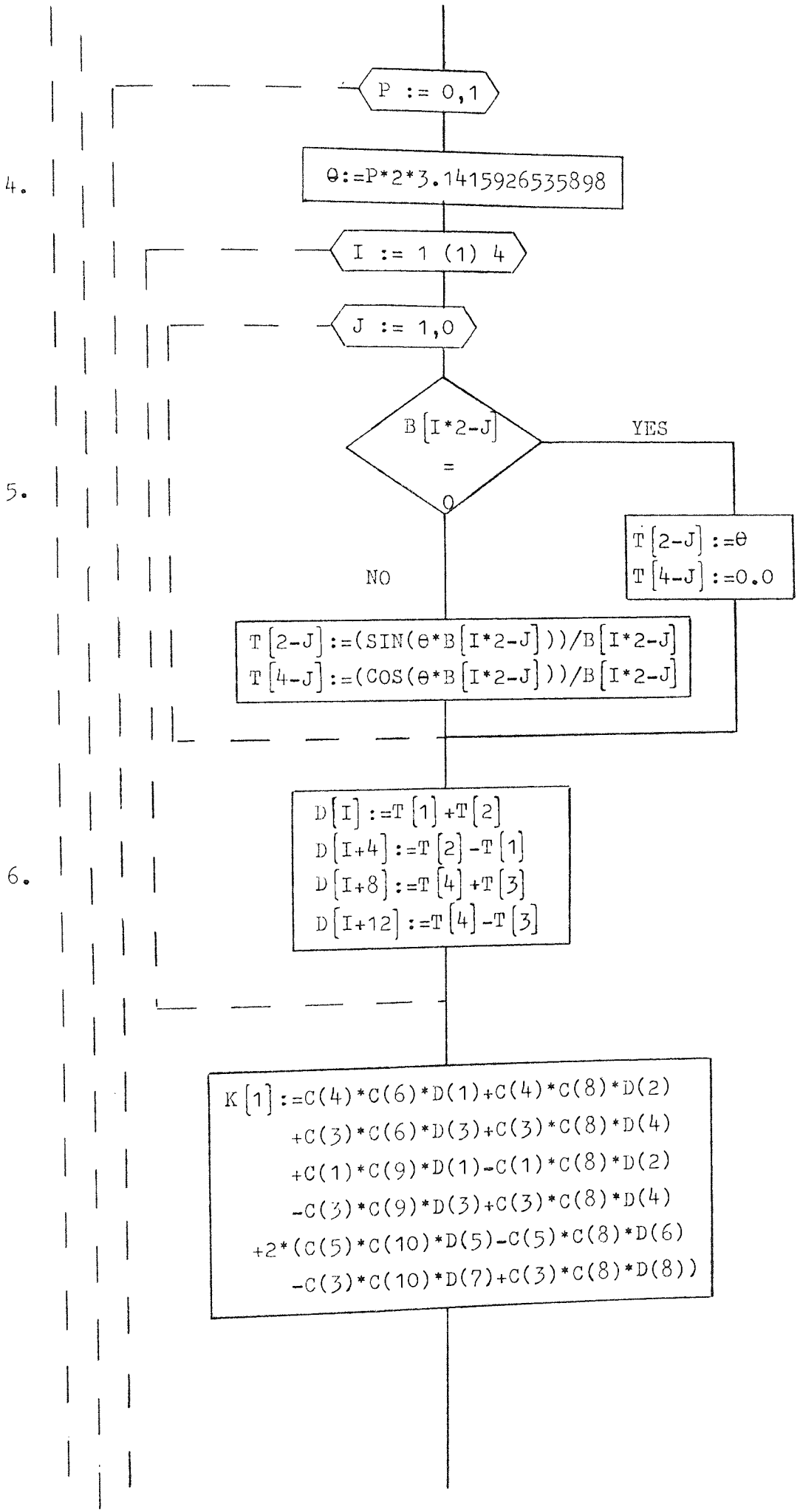
and mixed mode crack configurationswith minor adjustments. The following notes correspond with the steps in the flowchart and relate the program coding to the theory;-

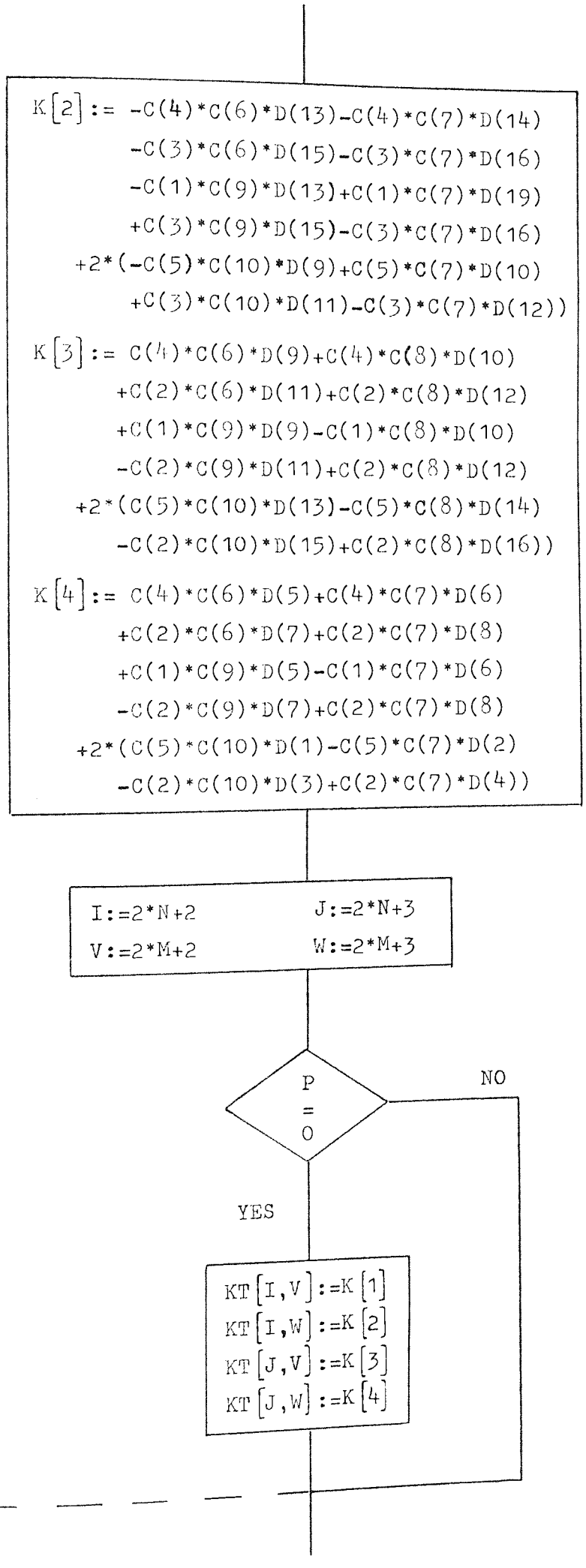
1. The core stiffness matrix [KT] is initialised and various parameters are preset.
2. Loops N and M are constructed to generate the coefficients of the core stiffness matrix. As this matrix is symmetric about its diagonal only the lower triangular coefficients are determined.
3. The functions $C_i(n,m)$ are found and correspond with the functions given in Appendix (9.1). Combinations of these terms, which are common, are grouped in variable B_i.
4. The counter P is formed and represents the upper and lower limits of equation (7.19).
5. To further simplify the strain energy expression the terms are further reduced within the loops I and J. The conditional statement detects the zero B_i terms and accordingly assigns the intermediate constants T_i.
6. The variables T_i are appropriately assigned to the constant D_i.
7. Each coefficient of the stiffness matrix is formed, collecting the evaluated terms. Note that each coefficient K_i is made up of three basic groups arising from the three terms of equation (7.16). The last group is multiplied by 2 in order to balance the multiplying parameter S. (see step 3).
8. The sub-matrix coefficients are stored in matrix [KT] for the first value of P and then carried forward to the final expression in step 9.

9. This constitutes the final expression for the coefficients of $[k_c]$ and each term is multiplied by the common factor S. The array [A] terms carry out the adjustment of the α coefficients to give the stress intensity factors. See equation (6.28).

Flowchart for procedure CORK :

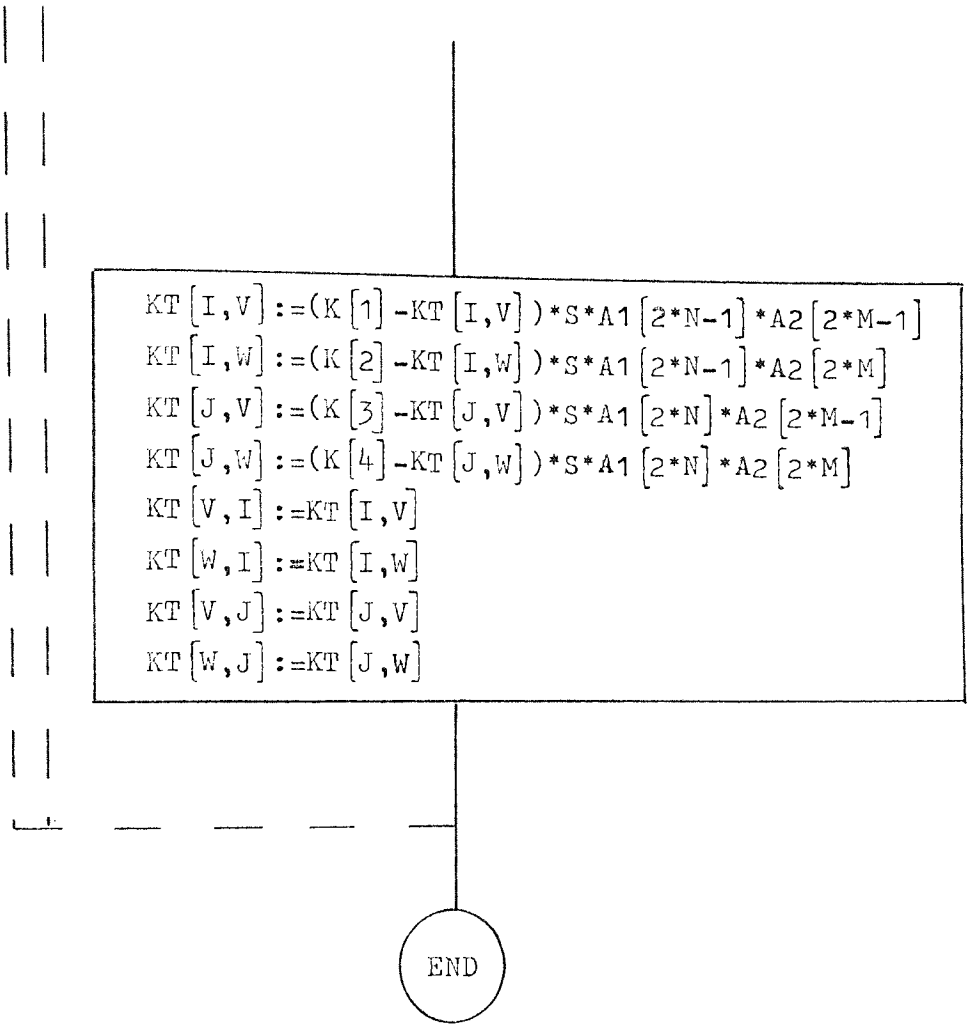



P := 0,1
4.
Θ:=P*2*3.1415926535898
I := 1 (1) 4
J := 1,0
B[I*2-J]
=
0
YES
5.
T[2-J] :=Θ
T[4-J] :=0.0
NO
T[2-J] :=(SIN(Θ*B[I*2-J]))/B[I*2-J]
T[4-J] :=(COS(Θ*B[I*2-J]))/B[I*2-J]
D[I] :=T[1] +T[2]
D[I+4] :=T[2] -T[1]
D[I+8] :=T[4] +T[3]
6.
D[I+12] :=T[4] -T[3]
K[1] :=C(4)*C(6)*D(1)+C(4)*C(8)*D(2)
+C(3)*C(6)*D(3)+C(3)*C(8)*D(4)
+C(1)*C(9)*D(1)-C(1)*C(8)*D(2)
-C(3)*C(9)*D(3)+C(3)*C(8)*D(4)
+2*(C(5)*C(10)*D(5)-C(5)*C(8)*D(6)
-C(3)*C(10)*D(7)+C(3)*C(8)*D(8))


K[2]:= -C(4)*C(6)*D(13)-C(4)*C(7)*D(14)
-C(3)*C(6)*D(15)-C(3)*C(7)*D(16)
-C(1)*C(9)*D(13)+C(1)*C(7)*D(19)
+C(3)*C(9)*D(15)-C(3)*C(7)*D(16)
+2*(-C(5)*C(10)*D(9)+C(5)*C(7)*D(10)
+C(3)*C(10)*D(11)-C(3)*C(7)*D(12))
K[3]:= C(4)*C(6)*D(9)+C(4)*C(8)*D(10)
+C(2)*C(6)*D(11)+C(2)*C(8)*D(12)
+C(1)*C(9)*D(9)-C(1)*C(8)*D(10)
-C(2)*C(9)*D(11)+C(2)*C(8)*D(12)
+2*(C(5)*C(10)*D(13)-C(5)*C(8)*D(14)
-C(2)*C(10)*D(15)+C(2)*C(8)*D(16))
K[4]:= C(4)*C(6)*D(5)+C(4)*C(7)*D(6)
+C(2)*C(6)*D(7)+C(2)*C(7)*D(8)
+C(1)*C(9)*D(5)-C(1)*C(7)*D(6)
-C(2)*C(9)*D(7)+C(2)*C(7)*D(8)
+2*(C(5)*C(10)*D(1)-C(5)*C(7)*D(2)
-C(2)*C(10)*D(3)+C(2)*C(7)*D(4))
I:=2*N+2
J:=2*N+3
V:=2*M+2
W:=2*M+3
P = 0
NO
YES
KT[I,V]:=K[1]
KT[I,W]:=K[2]
KT[J,V]:=K[3]
KT[J,W]:=K[4]

7.

8.


KT[I,V]:=(K[1]-KT[I,V])*S*A1[2*N-1]*A2[2*M-1]
KT[I,W]:=(K[2]-KT[I,W])*S*A1[2*N-1]*A2[2*M]
KT[J,V]:=(K[3]-KT[J,V])*S*A1[2*N]*A2[2*M-1]
KT[J,W]:=(K[4]-KT[J,W])*S*A1[2*N]*A2[2*M]
KT[V,I]:=KT[I,V]
KT[W,I]:=KT[I,W]
KT[V,J]:=KT[J,V]
KT[W,J]:=KT[J,W]
END

9.

7.3.6 NUMERICAL EXAMPLES AND CLOSING REMARKS

To assess the performance of the pseudo inverse method the case of a plate in uniaxial tension, containing an edge crack with various root angles, was investigated. The test results are compared with those obtained using the Hilton and Hutchinson technique in Table (7.11). It can be seen that the stress intensity factors and the rigid body displacements of both methods compare favourably, except for the rigid body displacement δx, for the 90 degree edge crack case, where there is a sign difference. On further investigation it was discovered that a spurious negative displacement, in an essentially positive displacement field, occurred along the crack faces. This could not be accounted for, despite a thorough investigation of the numerical procedures. A listing of the residual nodal forces gave no indication of any applied tractions along the crack surfaces, to explain this phenomenon. The technique was subsequently abandoned due to time restrictions and the need to investigate more immediately important topics.

A similar set of problems were chosen to test the refined core element facility, provided by procedure CORK. This procedure was used in conjunction with the Hilton and Hutchinson method of Chapter six. Selecting only the first two terms in the series expression, gave identical results to those obtained with the original program written by Robertson. However, taking a further two terms in the series did not enhance the solution accuracy as anticipated. For the particular case of a rectangular plate in uniform tension, the shearing mode stress intensity factor is zero; the corresponding finite element result gave $K_{II} = 2.75 \times 10^{3}$, clearly incorrect. A

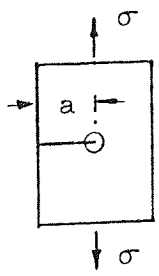
table of results, giving the finite element solutions with the corresponding core element degrees of freedom, is presented in Table (7.12). No explanation has arisen to account for these discrepancies, after considerable efforts were expended to locate the error. Robertson[28] derived the expressions for the first four x four coefficients of the core stiffness matrix $[k_c]$, and the numerical values of these coefficients, generated by the procedure CORK, are identical. This problem will be discussed further in Chapter eight.

ϕ degrees	Pseudo Inverse Method	Correct solution (Hilton/Hutchinson)
90°	$K_I = 5.51585 \times 10^4$ $K_{II} = 3.95$ $\delta x = -1.3154 \times 10^{-4}$ $\delta y = 4.999 \times 10^{-3}$ $w = -2.22 \times 10^{-3}$	$K_I = 5.1821 \times 10^4$ $K_{II} = 8.315$ $\delta x = 1.52025 \times 10^{-4}$ $\delta y = 4.46388 \times 10^{-3}$ $w = -1.96687 \times 10^{-3}$
45°	$K_I = 1.31226 \times 10^4$ $K_{II} = -6.64098 \times 10^3$ $\delta x = -2.553 \times 10^{-3}$ $\delta y = 1.78086 \times 10^{-3}$ $w = -3.1694 \times 10^{-4}$	$K_I = 1.271419 \times 10^4$ $K_{II} = -6.54955 \times 10^3$ $\delta x = -2.5348 \times 10^{-3}$ $\delta y = 1.781233 \times 10^{-3}$ $w = -3.609 \times 10^{-4}$
30°	$K_I = 1.89017 \times 10^4$ $K_{II} = -5.99618 \times 10^3$ $\delta x = -2.307 \times 10^{-3}$ $\delta y = 2.6699 \times 10^{-3}$ $w = -4.749203 \times 10^{-4}$	$K_I = 1.798 \times 10^4$ $K_{II} = -5.904 \times 10^3$ $\delta x = -2.286 \times 10^{-3}$ $\delta y = 2.663 \times 10^{-3}$ $w = -5.1012 \times 10^{-4}$

Units for K_I, K_{II} - $psi\sqrt{in}$
Displacement - in

TABLE 7.11 Results obtained using the 'Pseudo Inverse' Method, for various edge crack problems.

2-DOF	4-DOF
$w = -2.199\times10^{-3}$ $\delta x = -8.80889\times10^{-4}$ $\delta y = 4.9809\times10^{-3}$ $K_I = 5.233\times10^{4}$ $K_{II} = -1.7458\times10^{-5}$	$w = -2.4303\times10^{-3}$ $\delta x = -8.654\times10^{-4}$ $\delta y = 5.0071\times10^{-3}$ $K_I = 5.16645\times10^{4}$ $K_{II} = 2.749\times10^{3}$ $a_1 = 1.112\times10^{4}$ $a_2 = 5.3377\times10^{3}$
6-DOF	**8-DOF**
$w = -2.77049\times10^{-3}$ $\delta x = -8.69072\times10^{-4}$ $\delta y = 5.0102\times10^{-3}$ $K_I = 5.09236\times10^{4}$ $K_{II} = 3.2433\times10^{3}$ $a_1 = 1.4797\times10^{4}$ $a_2 = 4.81659\times10^{3}$ $a_3 = -1.66356\times10^{4}$ $a_4 = 1.93744\times10^{4}$	$w = -2.0983\times10^{-3}$ $\delta x = -8.6241\times10^{-4}$ $\delta y = 5.0102\times10^{-3}$ $K_I = 5.2119\times10^{4}$ $K_{II} = 3.7119\times10^{3}$ $a_1 = 1.2561\times10^{4}$ $a_2 = 3.652\times10^{3}$ $a_3 = -2.5124\times10^{4}$ $a_4 = 7.036\times10^{3}$ $a_5 = 9.07748\times10^{4}$ $a_6 = 5.578\times10^{4}$



a = 2.5in
L = 10.0in
W = 5.0in
t = 1.0in
σ = 12000 ibf/in^2

TABLE 7.12. Results obtained using the procedure CORK, for various core element d.o.f.

Units for K_I, K_{II} - $psi\sqrt{in}$
Displacement - in

7.4 PARTIAL CRACK CLOSURE

7.4.1 INTRODUCTION

In the application of the finite element method, the equilibrium displacement solution is obtained from a minimization of the total potential energy of the elastic body. For the normal mesh, inter-element compatibility is maintained by assuming suitable displacement functionswithin each element. No allowance is made, however, with respect to the compatibility requirements of free boundaries in close proximity to each other, as in the case of crack surfaces. Therefore, without special precautions the minimum energy configuration can result in the physically inadmissible situation where the free boundaries have crossed. Figure (7.13) illustrates the case of a rectangular plate with a 90 degree edge crack, under an applied bending moment M, and also the crack profiles for the correct and incorrect solutions. In this example the incorrect solution yields a negative mode I stress intensity factor which implies that the crack surfaces have crossed. Because of the symmetry of this problem, overlapping adjacent nodes can be constrained to move together, but for the mixed-mode problem, where the crack may be inclined to the global axis, corrective action is rather more difficult. In this situation the adjacent crack surfaces can move relative to each other, as shown in figure (7.14). Coalescing adjacent crack face nodes is equivalent to a no-slip finite element model.

The problem of partial crack closure was briefly examined by Robertson[28], who used an averaged displacement scheme to prevent the crack faces overlapping. The problem was not fully investigated

however, due to time restrictions, and only the mode I case was examined. Essentially the averaged displacement technique examines the overlapped crack configuration and computes the necessary displacement pattern to bring the overlapped crack faces into line. This is achieved simply by specifying the average displacements of adjacent nodes as geometric boundary conditions and re-solving the modified stiffness equations. This approach functions correctly for the symmetric mode I case, where only half the plate is considered and overlapping nodes are assigned to have zero δy displacements. See figure (7.13). In mixed-mode crack configurations this approach is invalid, as nodal displacements are specified using information from the overlapped component shape, thus violating the condition which allows the elastic body to take up the shape of minimum potential energy.

One possible method of overcoming this situation, was to use bar elements, and bridge the overlapped crack surfaces at the adjacent nodal points. An iterative scheme was developed which operated as follows:-

1. A bar element was attached to the pair of nodes corresponding to the point of maximum overlap.
2. The modified system of equations were solved.
3. Steps 1 and 2 were repeated until the crack surfaces no longer crossed over.

The bar element type is given in reference (82), section (8.6) and working through the matrix manipulation its stiffness matrix is

given by,

$$[k]^e = \frac{EA}{L}\begin{bmatrix} s^2 & -sc & -s^2 & sc \\ -sc & c^2 & -sc & -c^2 \\ -s^2 & -sc & s^2 & -sc \\ sc & -c^2 & -sc & c^2 \end{bmatrix} \qquad (7.22)$$

where E represents the modulus of elasticity, L and A represent the element length and cross-sectional area, and, s and c represent sinα and cosineα. The element details are given in figure (7.15). There are problems associated with the use of bar elements in this fashion. Theoretically the element length is zero, hence the value of the constants EA/L is arbitrary. The value of this group of constants can effect the position of the final closed crack surfaces. For example, using the particular case of figure (7.13), a 90 degree edge crack, the bar element stiffness matrix becomes,

$$[k]^e = \frac{EA}{L}\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 1 \end{bmatrix} \qquad (7.23)$$

where $\alpha = 0$. Now by changing the value of the constants EA/L the coefficients of the matrix $[k]^e$ can become large relative to the stiffness coefficients of the overall system. Thus, on solving the modified set of equations, the corresponding constrained nodal displacement tends to zero, due to the large diagonal coefficient introduced when adding expression (7.23). This tendency is illustrated by an example given in section (7.4.3).

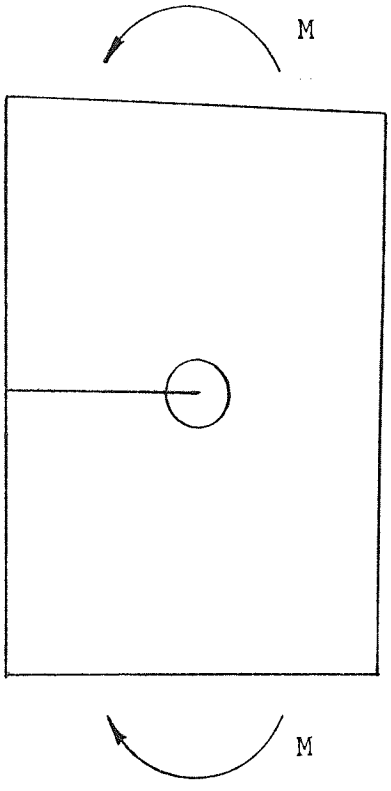
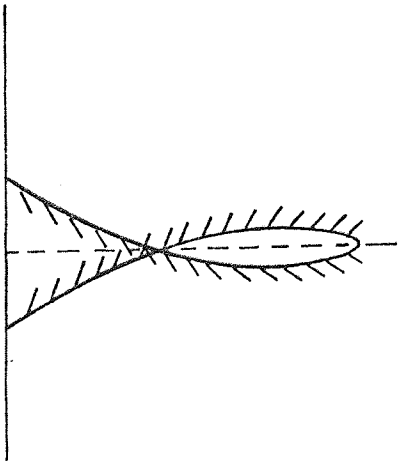
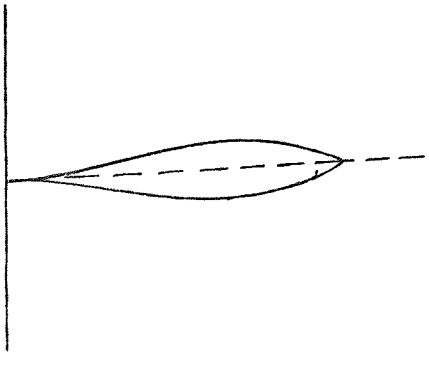


Plate containing a 90^{o} degree edge crack subject to in-plane bending.

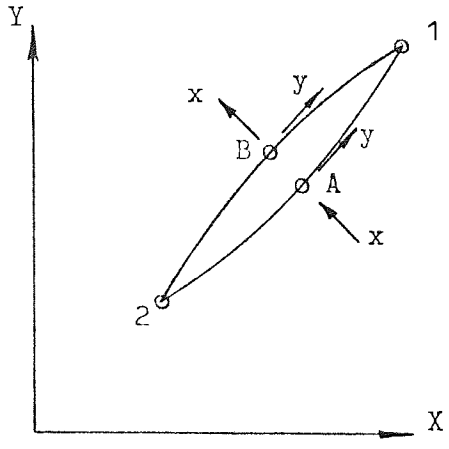


Incorrect crack profile where the crack surfaces have crossed.

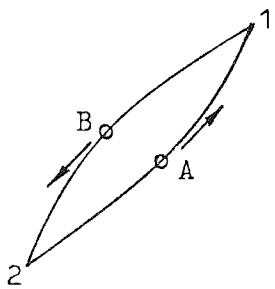


Correct crack profile.

FIG 7.13



Inclined crack subject to closure forces.



Sliding of crack surfaces for a non-zero K_{II}.

FIG 7.14

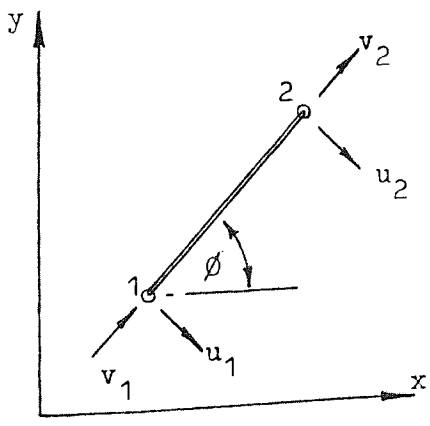


FIG 7.15 Bar Element details.

The inserted bar element technique does not provide a representative model for the problem of partial crack closure, hence, a method used by Iverson[83], for determining the stresses in screw threads was adopted. Here again the problem of free boundaries in close proximity is encountered and Iversen treated the situation by equating the values of the components of displacement normal to the crack face for adjacent crack face nodes. Overlap of crack faces is thus prevented whilst relative sliding is allowed. The method is described as enforcing equality of displacements and is further discussed by Cook[3]. Consider a set of simultaneous equations, which do not represent any particular structure,

$$\begin{aligned} 3x + 8y + 9z &= 30 \\ 8x + 20y + 11z &= 15 \\ 9x + 11y + 1z &= 6 \end{aligned} \tag{7.24}$$

or in matrix form

$$\begin{bmatrix} 3 & 8 & 9 \\ 8 & 20 & 11 \\ 9 & 11 & 1 \end{bmatrix} \begin{Bmatrix} x \\ y \\ z \end{Bmatrix} = \begin{Bmatrix} 30 \\ 15 \\ 6 \end{Bmatrix}$$

If we require $x = z$ then one of the three equations in (7.24) is redundant and we reduce the number of equations by addition, thus,

$$\begin{aligned} 8x + 20y + 11z &= 15 \\ 12x + 19y + 10z &= 36 \end{aligned} \tag{7.25}$$

and as $x = z$

$$\begin{aligned} 20y + 19z &= 15 \\ 19y + 22z &= 36 \end{aligned} \tag{7.26}$$

or in matrix form, forcing the redundant x to equal zero,

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 20 & 19 \\ 0 & 19 & 22 \end{bmatrix} \begin{Bmatrix} x \\ y \\ z \end{Bmatrix} = \begin{Bmatrix} 0 \\ 15 \\ 36 \end{Bmatrix} \qquad (7.27)$$

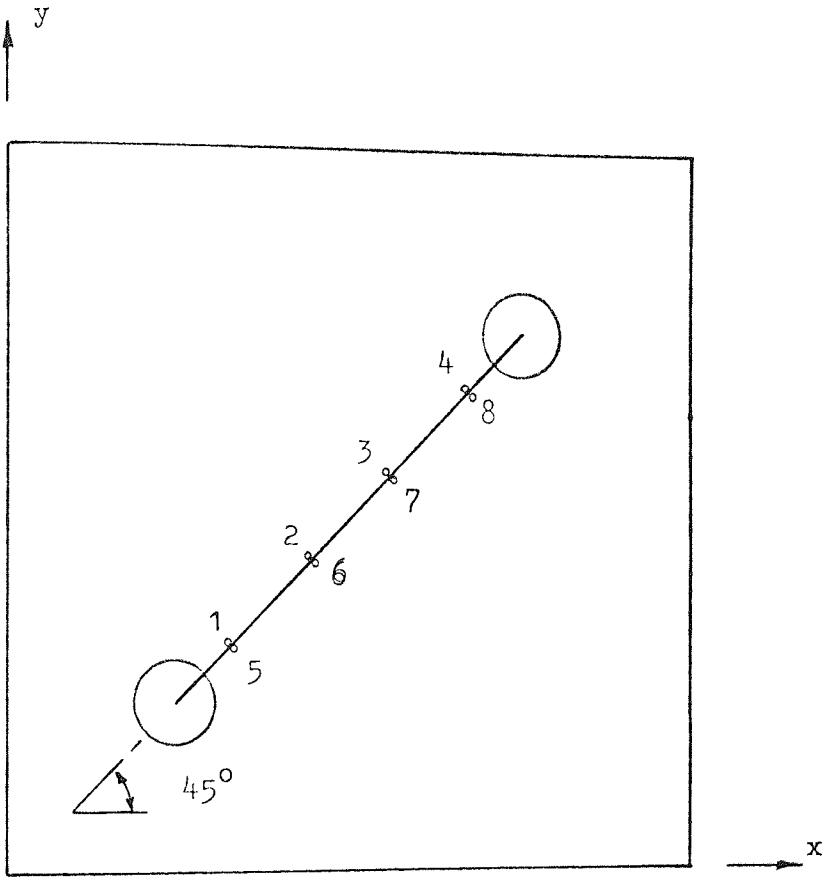
This result implies that the rows and columns of the redundant displacement are added to the rows and columns of the retained displacement. Both sets of adjacent nodal displacements can be condensed in this manner to represent a no-slip model. Various crack configurations are examined in section (7.4.3) where partial closure exists and both of the above models are used to obtain a solution.

7.4.2 THE NODAL COUPLING TECHNIQUE APPLIED TO PARTIAL CRACK CLOSURE

Before describing the numerical procedures a few notes on the method of data input are appropriate.

a) The nodes lying on opposite crack faces must have the same coordinates.

b) The normal displacements of the crack surface nodes are used to determine whether crack closure has occurred, hence, in the case of inclined cracks the nodal displacements must be expressed in local skew coordinates. This is achieved using the procedures SKEWEDCON and SKEWLOAD. The data input sequence is given in the users guide Chapter five. Now, as the paired crack surface nodes have the same coordinates, in order to differentiate between the



SURNO = 8 NSKEW = 8 .

Crack Face	Upper		Lower
1	45.0	5	45.0
2	45.0	6	45.0
3	45.0	7	45.0
4	45.0	8	45.0

FIG 7.16 Corresponding crack closure input data.

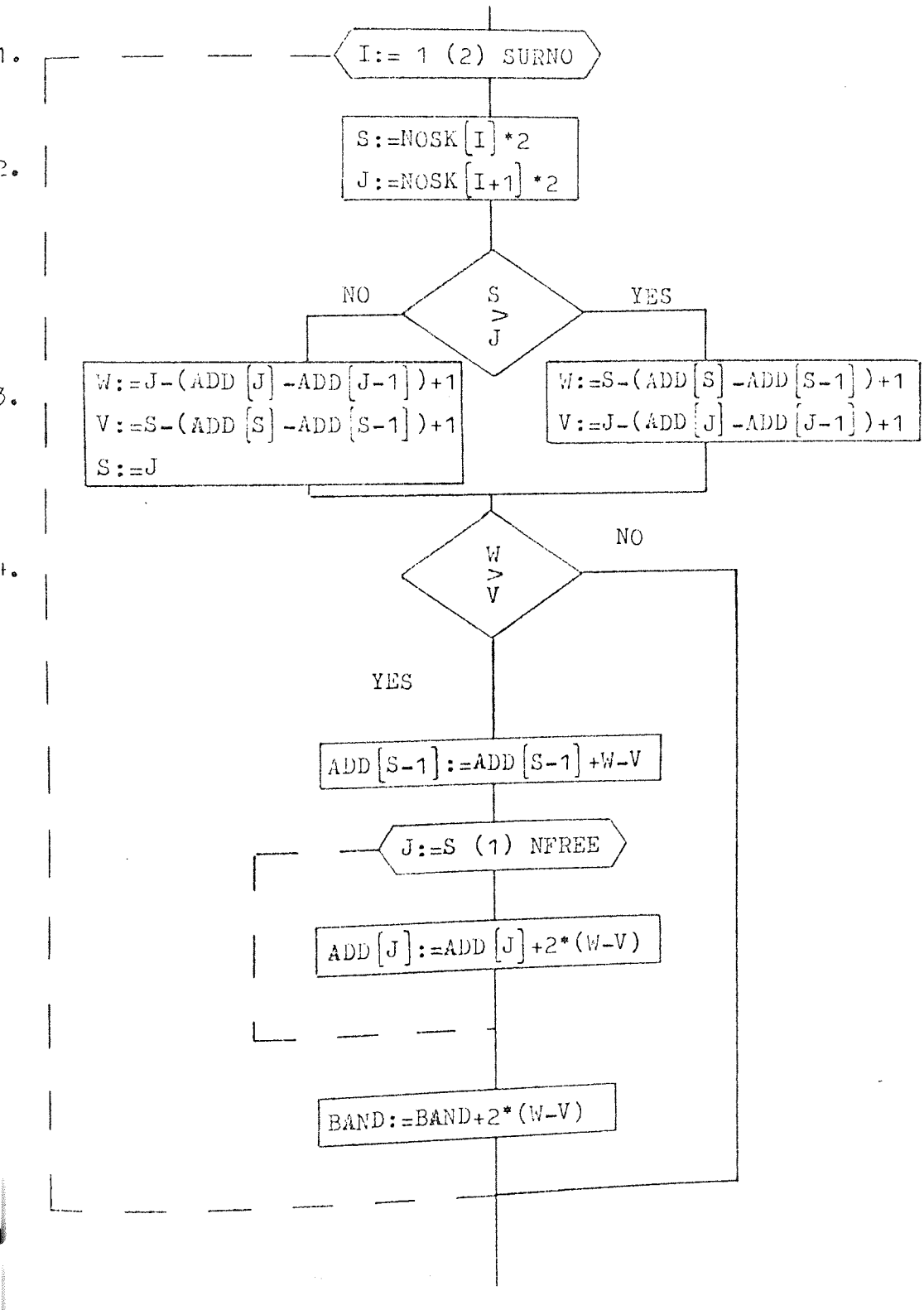
upper and lower crack faces, the node numbers are entered as pairs giving the upper crack face node number first and then the lower crack face node number. Note that these nodes must be accompanied by the appropriate angle of skew. Figure (7.16) shows the data input sequence for an inclined crack.

Implementing the node coupling technique discussed earlier, may increase the size of the stiffness matrix [K], due to the row and column interchanges. The address sequence, which relates the one-dimensional stiffness matrix to the coefficients within the two-dimensional matrix, is adjusted using the skewed node input data. This process is carried out as part of the main control program, before the assembly procedure is called. Its operation is described in the following steps, which refer to the flowchart overleaf:-

1. The loop I is constructed so that each pair of nodes on the crack faces can be scanned.
2. The integers S and J are assigned the values of the pair of node numbers.
3. The smaller of the two node numbers is made redundant and the conditional statement determines relative values of S and J. Making the smaller node number redundant, automatically ensures that there is room in the stiffness matrix to carry out the column summation. However, the row summation does not possess this guarantee, hence the integers W and V are computed, and represent the remaining unused space in each row, corresponding with the nodes S and J. S represents the larger node number on exiting step 3.

4. If integer W ≥ V then there is room within the stiffness matrix to carry out the row summation, otherwise the appropriate changes to the address sequence are carried out.

Excerpt from the main program PCPOXY flowchart :-



The iterative scheme, discussed in the introduction, requires that the continually modified system equations be repeatedly solved. As the solving routine overwrites both the stiffness matrix and force vector, a method of storing the modified arrays is introduced, whereby the computer backing store is utilized. Essentially each time the system equations are modified they are saved in the computer's backing store, using the transfer routine PUTARRAY, and similarly recalled using GETARRAY. The complete program listing is given in the Appendix (9.3).

PROCEDURE CLOSURE:-

The CLOSURE procedure implements the nodal coupling technique discussed in the introduction and its operation is described by the following steps, which correspond with the flowchart:-

1. Various controlling integers are preset and the crack face nodal displacements are transferred to arrays {X} and {Y}. Note that the force vector {Q} is overwritten by the displacements in the solving routine.
2. The counter V records the number of times procedure CLOSURE has been accessed. Also the array {OVP} records the coefficient number of the paired nodes which have been coupled, and this information is used so that the redundant normal displacement, which is zero, can be replaced by its twin nodal displacement.
3. The control loop I is constructed so that each crack face node pair can be scanned to detect overlap. The conditional statements determine the point of maximum overlap which is recorded by W.

4. The crack surface nodal displacements are printed out, together with the computed stress intensity factors.

5. If the crack faces do not overlap then the matrix modification process is not implemented and the stress intensity factors are scanned. See step 10.

6. If the crack faces are overlapped then the system equations are recalled from the backing store, using GETARRAY.

7. Integer J and I take the values of the crack face node numbers which are at the point of maximum overlap, that is W; J taking the larger node number.

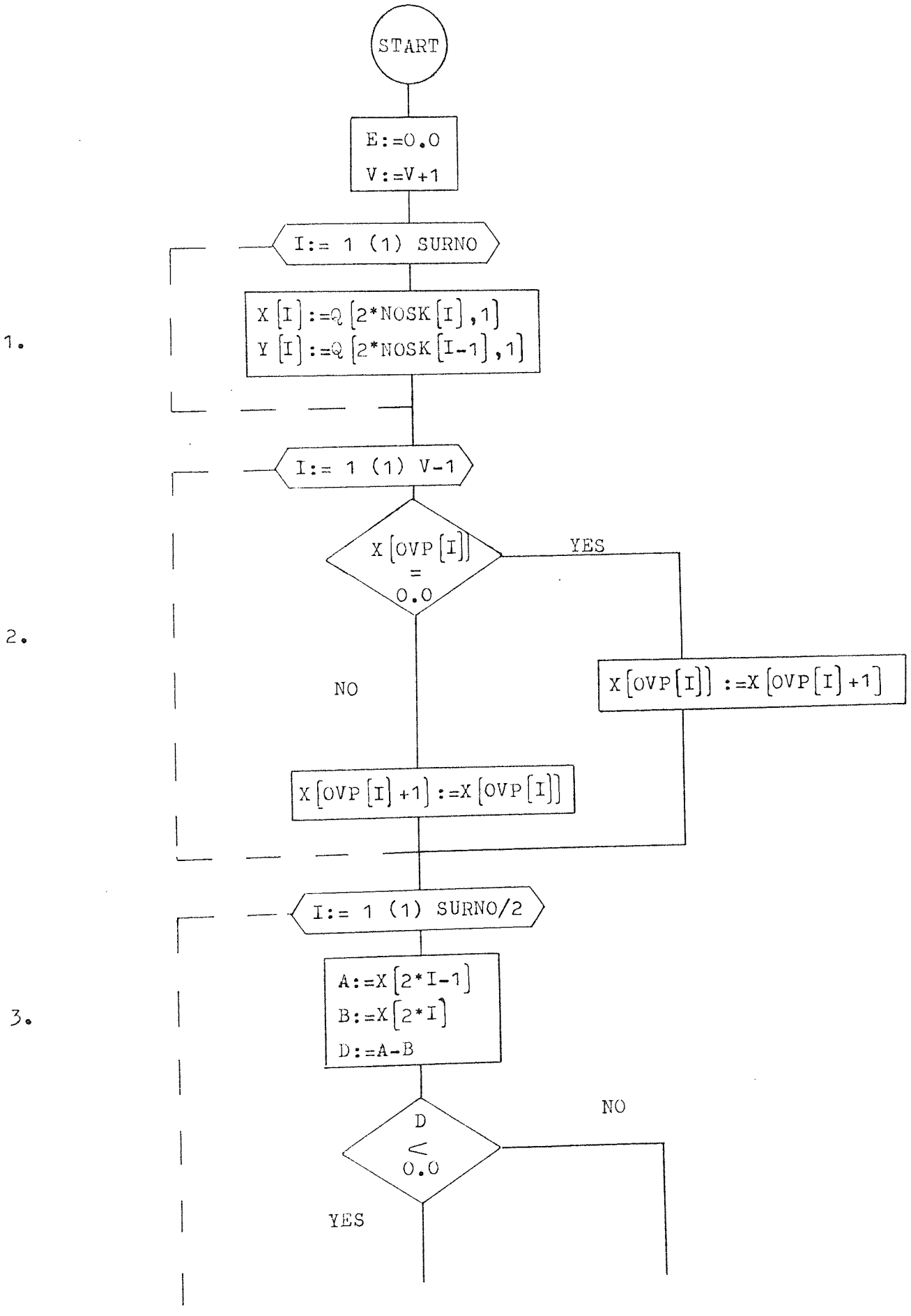
8. This step represents the coupling process, where the coefficients of the rows and columns corresponding to the redundant node, are added to the coefficients of the rows and columns of the twin node. In this case both the x and y d.o.f. are coupled giving a no-slip model. The counter Z operates on the integers I and J, so that the paired crack face nodes are coupled in the δx and δy sense respectively. The frictionless model results from coupling the normal displacements only and this is achieved by equating Z to zero. Note that the program PCPOY contains the frictionless closure procedure. The coefficients of the stiffness matrix along row I are transferred to the corresponding row J, and

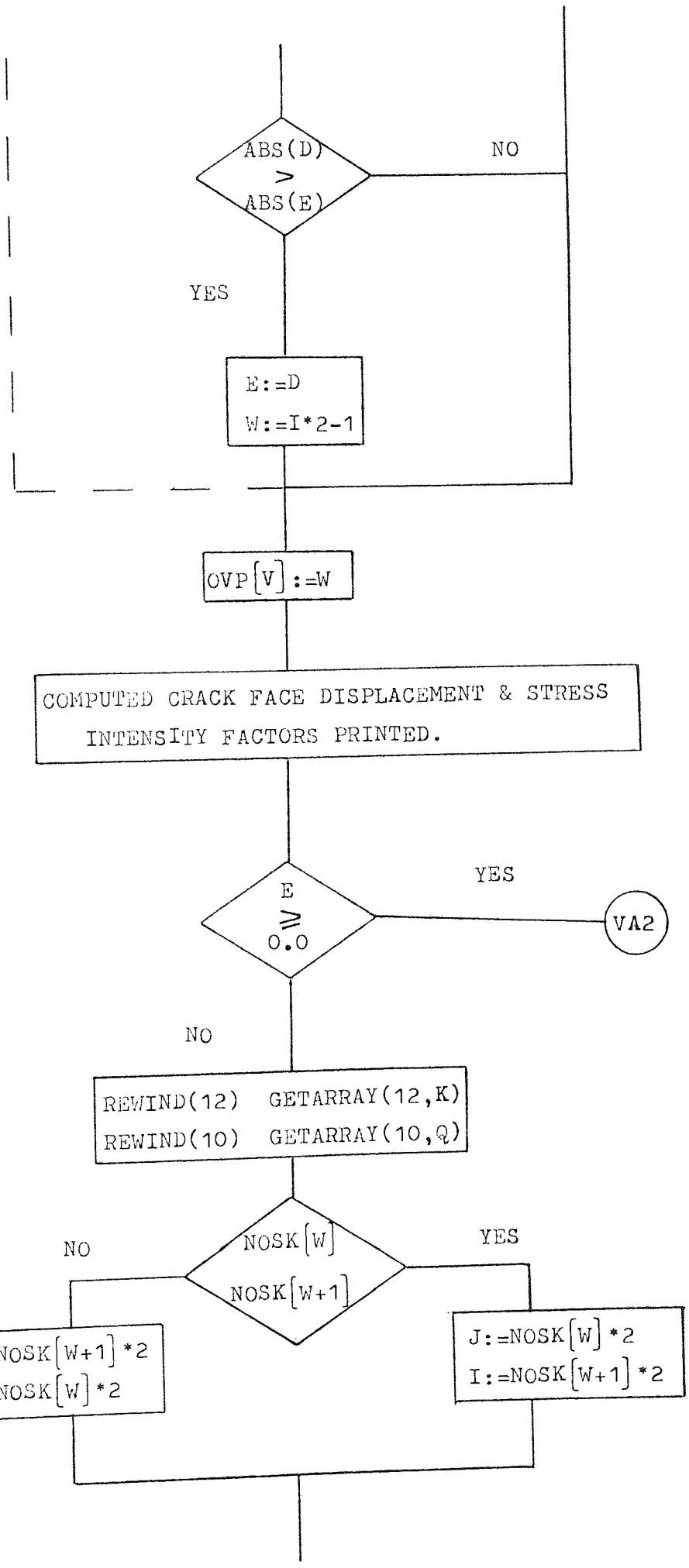
similarly the coefficients lying in column I are added to the coefficients of column J. Finally the diagonal coefficients of [K] and the force vector coefficients corresponding to I and J, are added. The sub-procedure GEOMBC is called so that displacement I is zero on solving the system equations, as given by equation (7.27).

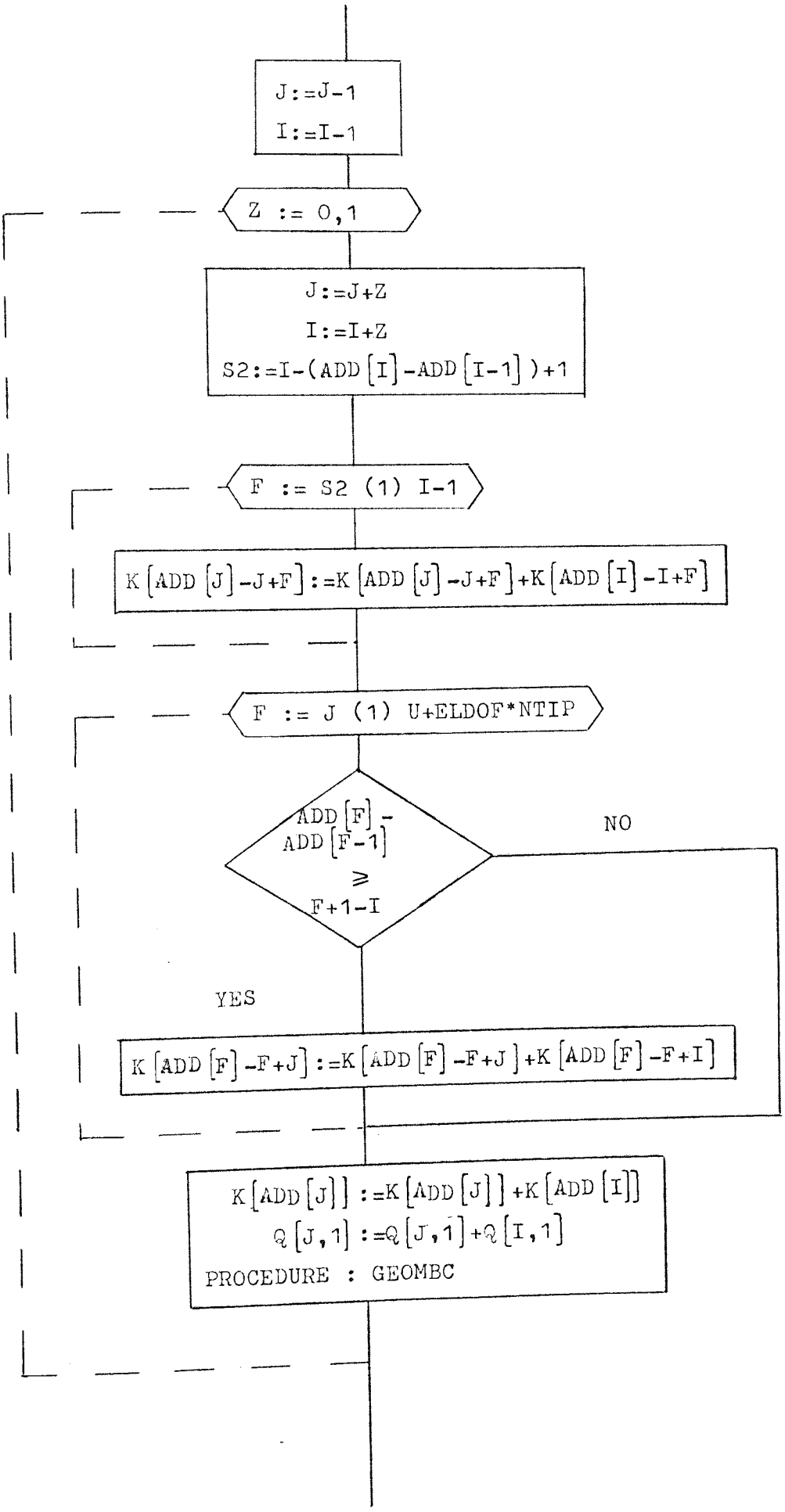
9. After the stiffness equations have been modified the main control program is retraced from position RESOLVE.

10. If the crack faces have not overlapped then the mode I stress intensity factors are scanned to determine whether they are still negative. If this is the case, then they are forced to be zero by the sub-procedure GEOMBC.

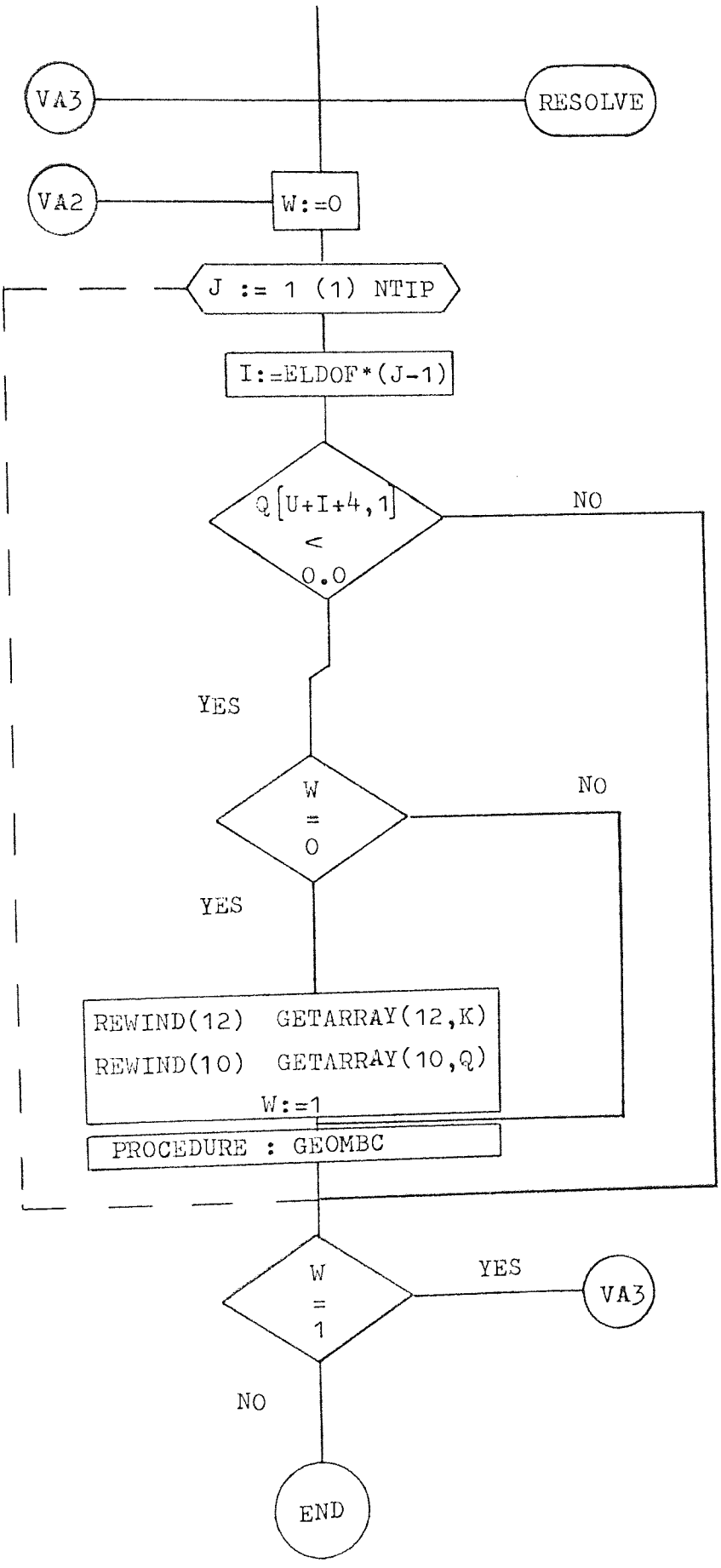
Flowchart for procedure CLOSURE :-




3.
ABS(D) > ABS(E)
NO
YES
E:=D
W:=I*2-1
OVP[V]:=W
COMPUTED CRACK FACE DISPLACEMENT & STRESS INTENSITY FACTORS PRINTED.
4.
E ≥ 0.0
YES
VA2
5.
NO
REWIND(12) GETARRAY(12,K)
REWIND(10) GETARRAY(10,Q)
6.
NOSK[W] NOSK[W+1]
NO
YES
J:=NOSK[W+1]*2
I:=NOSK[W]*2
J:=NOSK[W]*2
I:=NOSK[W+1]*2
7.


J:=J-1
I:=I-1
Z := 0,1
J:=J+Z
I:=I+Z
S2:=I-(ADD[I]-ADD[I-1])+1
8.
F := S2 (1) I-1
K[ADD[J]-J+F]:=K[ADD[J]-J+F]+K[ADD[I]-I+F]
F := J (1) U+ELDOF*NTIP
ADD[F]-ADD[F-1] ≥ F+1-I
NO
8.
YES
K[ADD[F]-F+J]:=K[ADD[F]-F+J]+K[ADD[F]-F+I]
K[ADD[J]]:=K[ADD[J]]+K[ADD[I]]
Q[J,1]:=Q[J,1]+Q[I,1]
PROCEDURE : GEOMBC

9.


VA3
RESOLVE
VA2
W:=0
J := 1 (1) NTIP
I:=ELDOF*(J-1)
Q[U+I+4,1] < 0.0
NO
YES
W = 0
NO
YES
REWIND(12) GETARRAY(12,K)
REWIND(10) GETARRAY(10,Q)
W:=1
PROCEDURE : GEOMBC
W = 1
YES
VA3
NO
END

10.

7.4.3 NUMERICAL EXAMPLES FOR THE CASE OF PARTIAL CRACK CLOSURE

The first example has been chosen to contrast the results obtained for the three techniques discussed in the introduction. The problem is that of a rectangular plate containing a 90 degree edge crack, subjected to a bending moment M. The results are shown in Table (7.17) and the final crack profiles are illustrated in figure (7.18). The example has been treated as a mix-mode fracture problem, and therefore the shearing mode II can be used as a measure of the method's accuracy. In this case the shearing stress intensity factor K_{II}, should strictly, be zero. As expected the average displacement method yields a poor K_{II} value (6.7% of K_I), and this indicates that a shear stress distribution has been set up around the crack tip, resulting from the assumed nodal displacements. A similar situation arises with the inserted bar element technique, here $K_{II} = -68.0$ lb/in$^{3/2}$ or 2,5% of K_I. The nodal coupling technique yields an identical set of results for both the frictionless and no-slip models, which would be expected from the symmetry of the problem and the fact that no shear is present. The K_{II} values, in these results were 0.1% of the K_I values. The same problem was solved using various bar element stiffnesses, and the resulting crack profiles are shown in figure (7,19). It can be seen that as the EA/L value is increased, the joined nodal displacements tend to zero. Clearly this is not a representative model of the partial crack closure phenomenon, and the nodal coupling technique is used in the remaining examples,

Average Displacement Technique.

Iteration 4.

$\delta x = -1.4778 \times 10^{-3}$
$\delta y = 4.574 \times 10^{-4}$
$K_I = 2.0499 \times 10^{3}$
$K_{II} = 1.3728 \times 10^{2}$
$w = 6.1152 \times 10^{-4}$

Inserted Bar Element solution.

Iteration 3.

$\delta x = -1.2119 \times 10^{-3}$
$\delta y = 7.626 \times 10^{-4}$
$K_I = 2.7002 \times 10^{3}$
$K_{II} = -6.7916 \times 10^{1}$
$w = 4.4712 \times 10^{-4}$

Units K_I & K_{II} - $psi\sqrt{in}$
Displacement - in

Node Coupling Method.

No-Friction Model Iteration 3.

$\delta x = -1.4307 \times 10^{-3}$
$\delta y = 7.2239 \times 10^{-4}$
$K_I = 2.4403 \times 10^{3}$
$K_{II} = -2.222 \times 10^{0}$
$w = 5.5087 \times 10^{-4}$

No-Slip Model Iteration 3.

$\delta x = -1.4307 \times 10^{-3}$
$\delta y = 7.2238 \times 10^{-4}$
$K_I = 2.4403 \times 10^{3}$
$K_{II} = -2.304 \times 10^{0}$
$w = 5.5087 \times 10^{-4}$

TABLE 7.17. Comparison of results for the three partial crack closure models. Example(1), see accompanying figure(7.18).

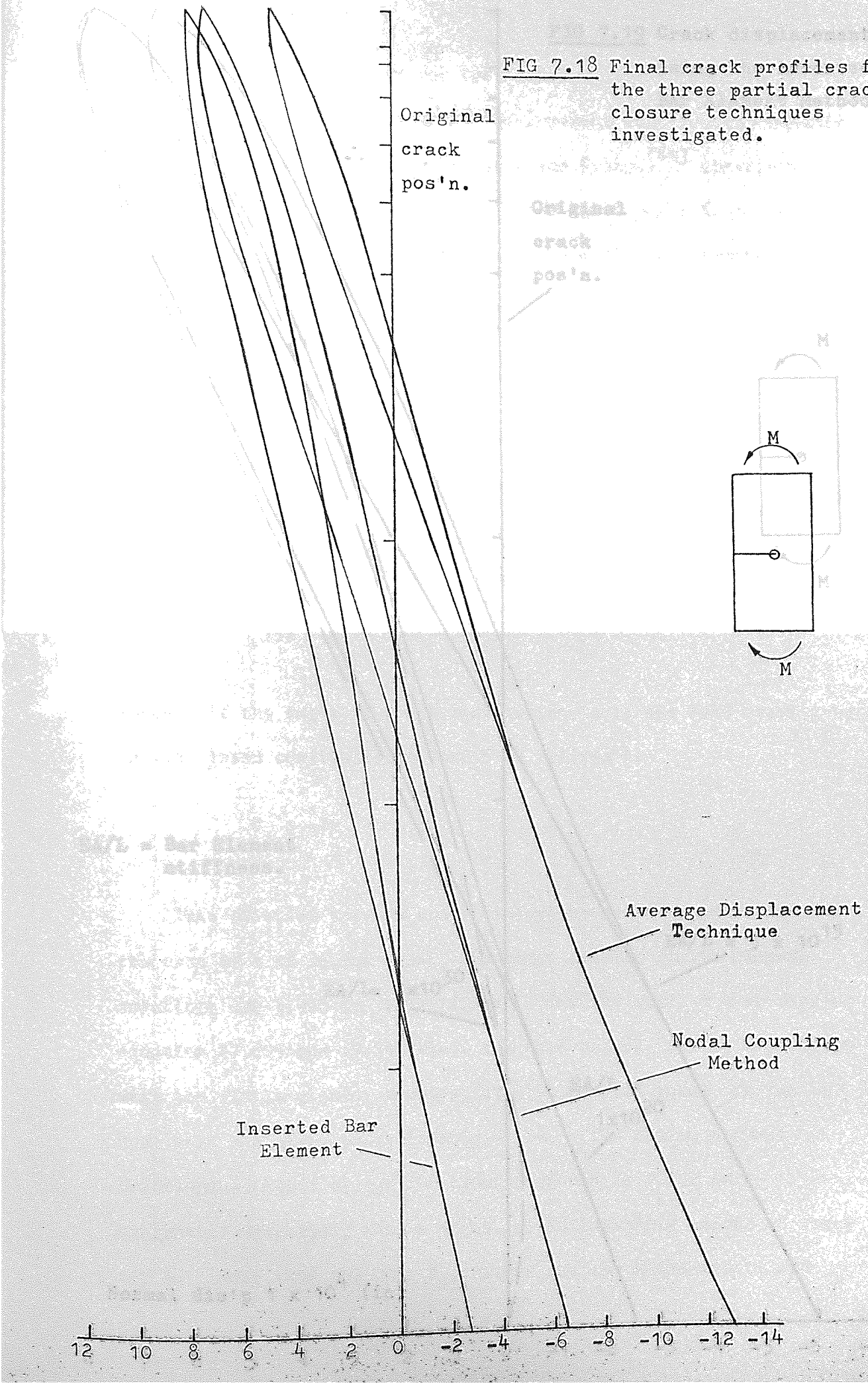


FIG 7.18 Final crack profiles for the three partial crack closure techniques investigated.

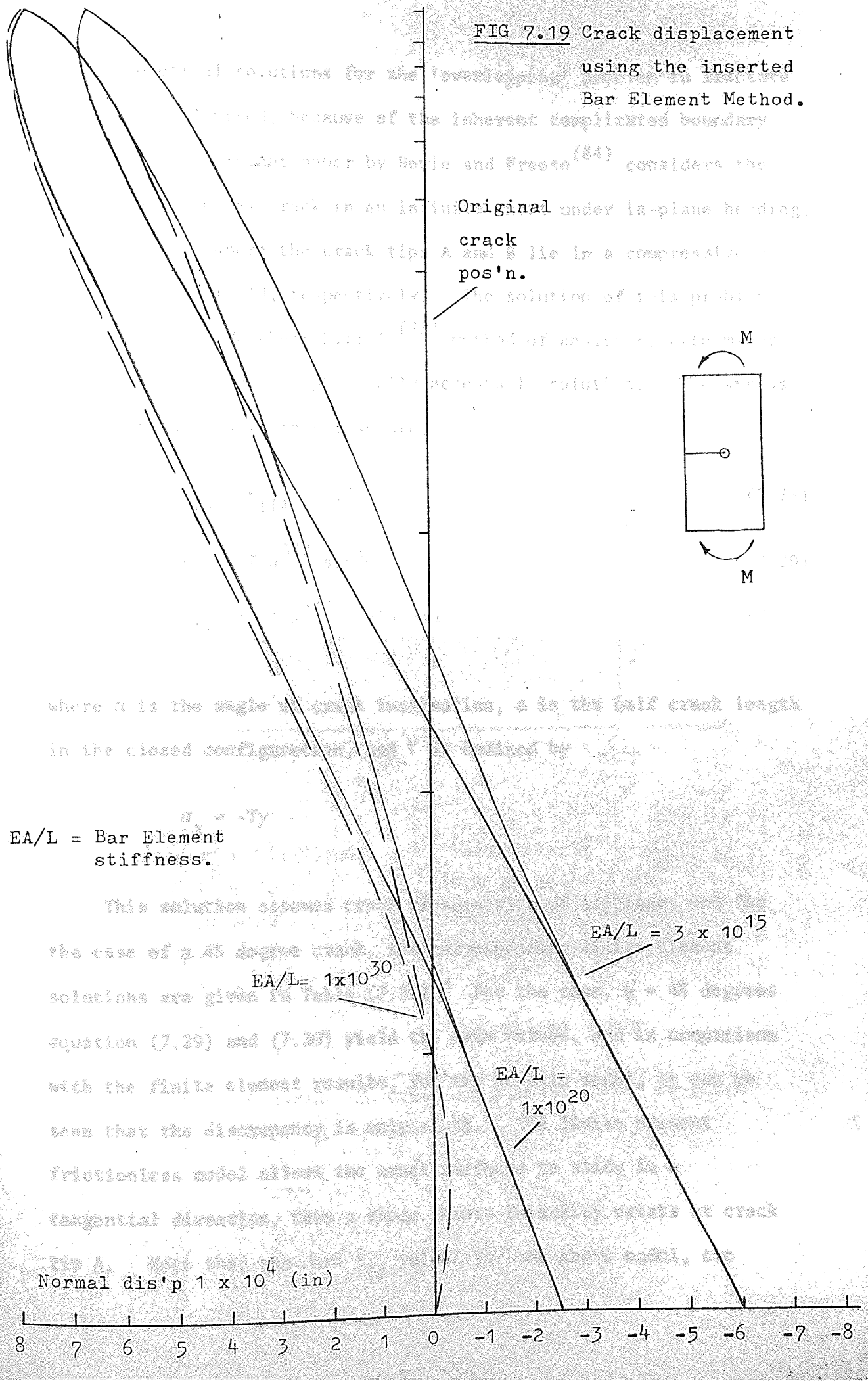


FIG 7.19 Crack displacement using the inserted Bar Element Method.

Theoretical solutions for the 'overlapping' problem in fracture analysis are limited, because of the inherent complicated boundary conditions. A recent paper by Bowie and Freese[84] considers the case of an internal crack in an infinite sheet under in-plane bending, figure (7.20), where the crack tips A and B lie in a compressive and tensile stress field, respectively. The solution of this problem was found using Muskhelishvili's[25] method of analysis, with minor modifications to give a physically acceptable solution. The stress intensity factors in this case are,

$$K_{IA} = K_{IIA} = 0.0 \tag{7.28}$$

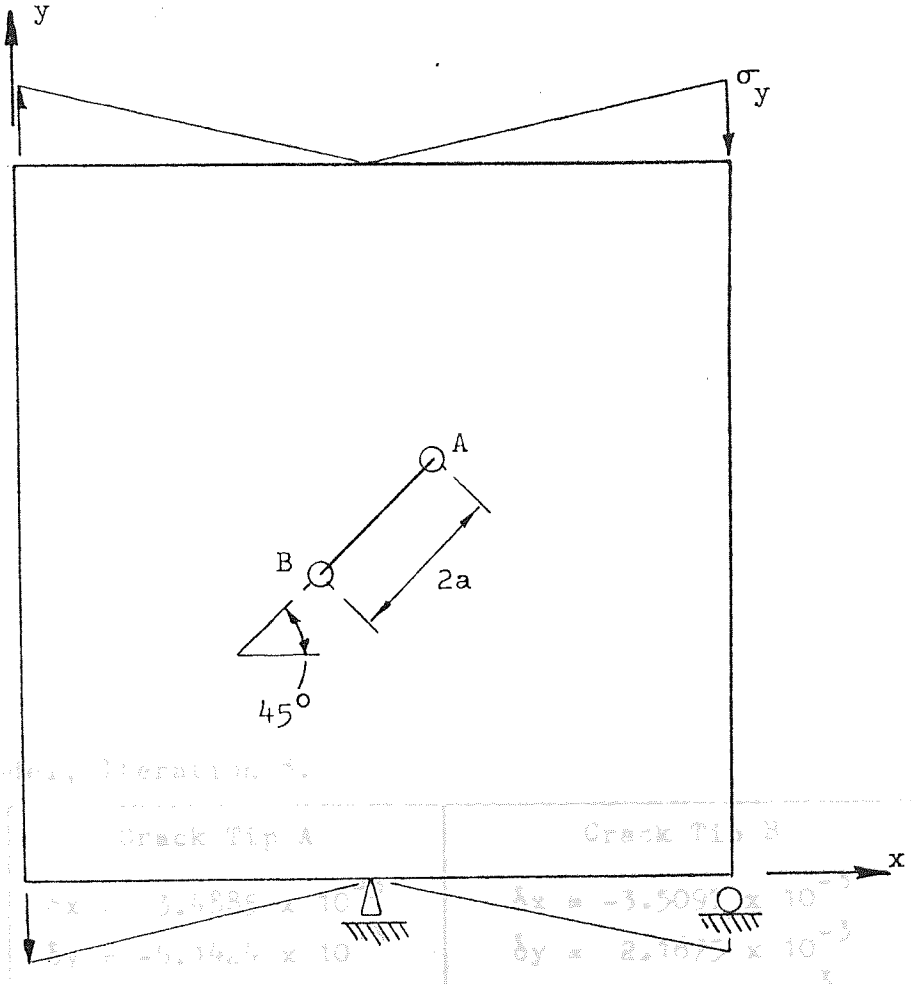
$$K_{IB} = T\, a^{3/2} \sin^3\alpha \tag{7.29}$$

$$K_{IIB} = -T\, a^{3/2} \sin^2\alpha\cos\alpha \tag{7.30}$$

where α is the angle of crack inclination, a is the half crack length in the closed configuration, and T is defined by

$$\sigma_x = -Ty$$

This solution assumes crack closure without slippage, and for the case of a 45 degree crack, the corresponding finite element solutions are given in Table (7.21). For the case, α = 45 degrees equation (7.29) and (7.30) yield the same values, and in comparison with the finite element results, for the no-slip model, it can be seen that the discrepancy is only -1.3%. The finite element frictionless model allows the crack surfaces to slide in a tangential direction, thus a shear stress intensity exists at crack tip A. Note that the two K_{II} values, for the above model, are



σ_y = T(x-8)psi
T = 3000
L = 20
W = 16
t = 1.0
a = 1.5

Mesh details
No. elements - 100
No. nodes - 344
Rc = 0.05

Dimensions - ins

FIG 7.20 Central 45° crack in a plate under in-plane bending. Example(2)

Theoretical solution from Bowie and Freese(84).
No-Slip.

Crack Tip A $K_I = K_{II} = 0.0$

Crack Tip B $K_I = K_{II} = 1060.62$

Finite Element results.

No-Friction Model, Iteration 8.

Crack Tip A	Crack Tip B
$\delta x = 3.4833 \times 10^{-3}$	$\delta x = -3.5045 \times 10^{-3}$
$\delta y = -5.1574 \times 10^{-3}$	$\delta y = 2.1626 \times 10^{-3}$
$K_I = 0.0$	$K_I = 1.0466 \times 10^{3}$
$K_{II} = -9.6642 \times 10^{2}$	$K_{II} = 9.6444 \times 10^{2}$
$w = -1.1955 \times 10^{-4}$	$w = -8.0094 \times 10^{-4}$

No-Slip Model, Iteration 8.

Crack Tip A	Crack Tip B	
$\delta x = 3.4885 \times 10^{-3}$	$\delta x = -3.5097 \times 10^{-3}$	
$\delta y = -5.1424 \times 10^{-3}$	$\delta y = 2.1675 \times 10^{-3}$	
$K_I = 0.0$	$K_I = 1.0458 \times 10^{3}$	dis'y -1.3%
$K_{II} = 0.0$	$K_{II} = 1.0467 \times 10^{3}$	
$w = -1.1391 \times 10^{-3}$	$w = -8.0064 \times 10^{-4}$	

Units K_I & K_{II} - $psi\sqrt{in}$
Displacement - ins

TABLE 7.21. Comparison of results for a 45° centrally cracked plate under in-plane bending, see figure(7.20) example(2).

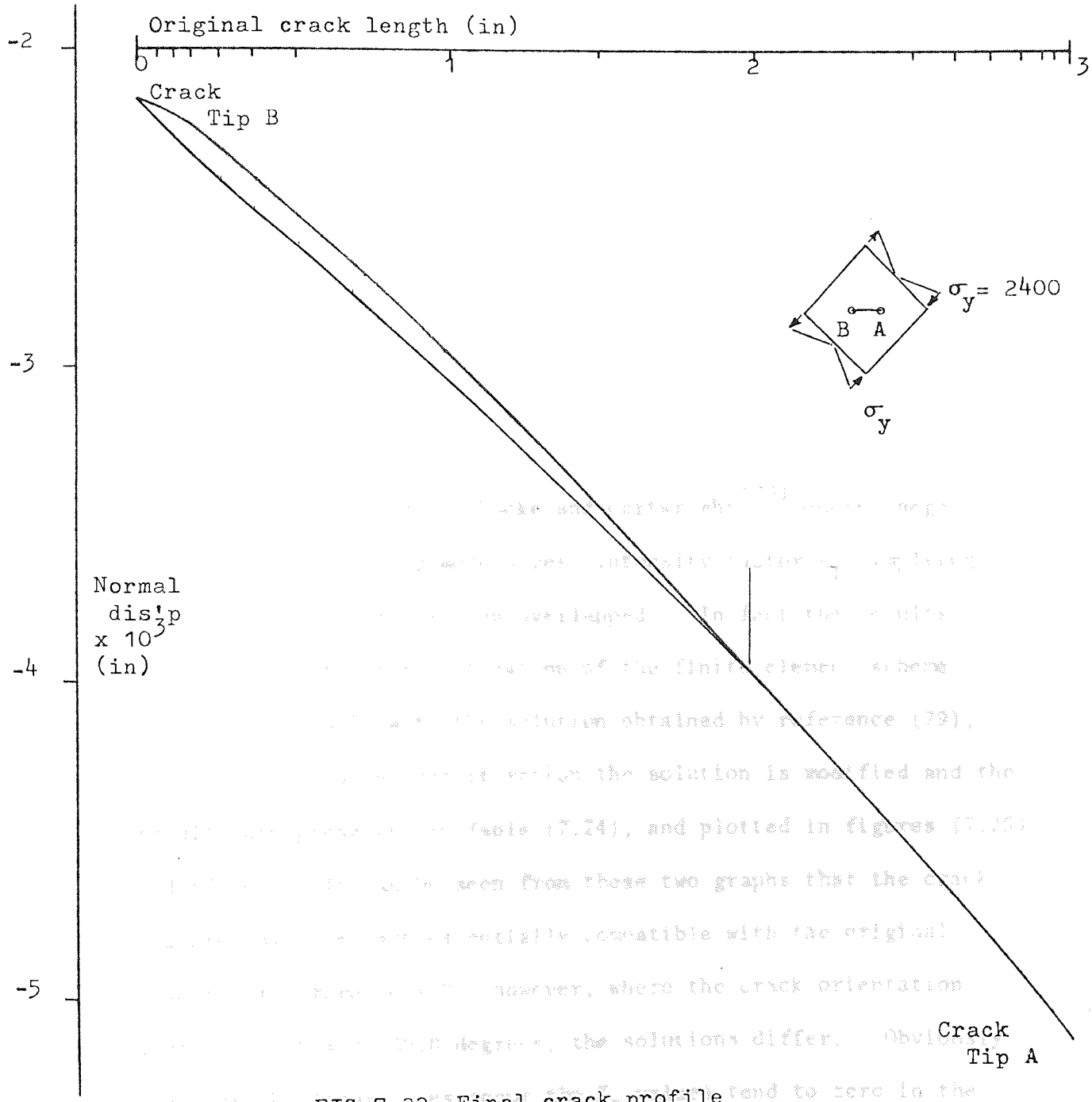


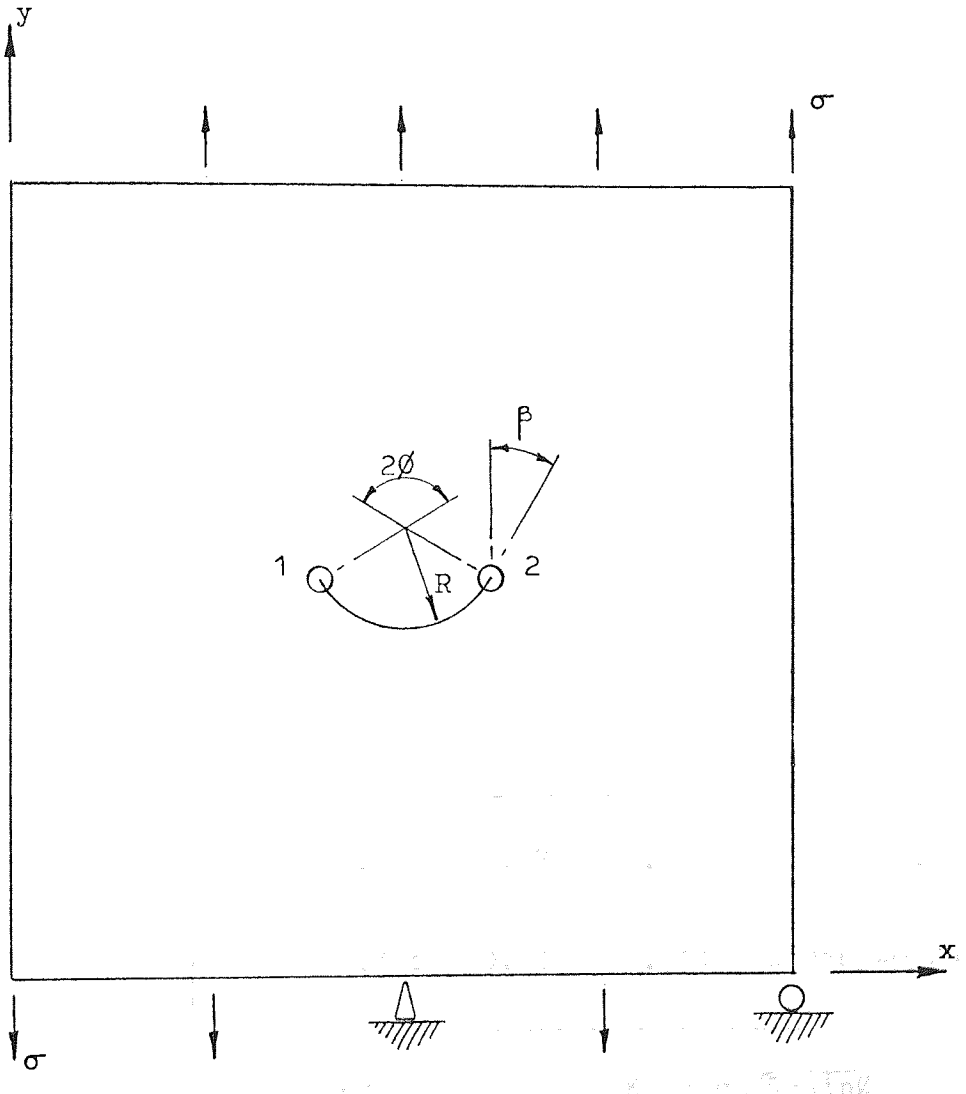
FIG 7.22 Final crack profile example(2).

identical and that their value is 8.0% lower than found in the no-slip model, indicating that the shear stress has been relieved due to the relative movement of the crack faces. Figure (7.22) illustrates the final crack profile, which is applicable to both finite element models.

Having shown that the finite element method provides a physically acceptable solution to the problem of partial crack closure, the remaining examples illustrate the method's versatility. The problem of a circular arc crack of radius, R, in a plate subjected to a uniform uniaxial stress, σ, has been chosen as the third example, see figure (7.23). There have been several theoretical solutions proposed for this problem, none of which are entirely satisfactory. The solution presented by Rooke and Cartwright[79] quotes negative values for the opening mode stress intensity factor K_I, implying that the crack surfaces have overlapped. In fact the results obtained from the first iteration of the finite element scheme compare favourably with the solution obtained by reference (79), however after subsequent iteration the solution is modified and the results are presented in Table (7.24), and plotted in figures (7.25) and (7.26). It can be seen from these two graphs that the crack closure solutions are essentially compatible with the original results of reference (79), however, where the crack orientation angle $\beta = 0.0$ and -30.0 degrees, the solutions differ. Obviously where crack closure does occur the K_I values tend to zero in the finite element solution, as illustrated in figure (7.25). Also it would be reasonable to assume that, in the no-slip model, where K_I tends to zero K_{II} will similarly tend to zero, due to the restricted

relative crack surface displacement. In the frictionless model the crack faces are allowed to slide over one another, thus, whereas K_I may tend to zero the unrestricted tangential displacements can induce a large K_{II} value. This is the case in figure (7.26), where the K_{II} values in the frictionless model are comparable with the first iteration solutions. In practice it would be difficult to assess which model is applicable, from a theoretical point of view a frictional force would have to be applied to the closed crack faces corresponding with the normal crack surface loading. The final crack profiles are illustrated in figures (7.27, 7.28, 7.29 and 7.30), both the frictionless and no-slip solutions give similar crack profiles, which are represented in the above figures.

A similar problem has been chosen as the fourth example. Here, a rectangular plate contains a crack near a circular hole of radius R. The crack is on a line which passes through the centre of the hole and the plate is subjected to a linearly varying stress perpendicular to the crack direction, see figure (7.31). In this example the sliding mode stress intensity factors K_{II} are zero and therefore the finite element models yield the same result. The solutions obtained from reference (79), are given, with the results obtained using the finite element scheme, in Table (7.32). It was found that the crack overlaps at crack tip A, resulting in a negative K_I value. For the first iteration the opening mode K_{IB} value was within 4.0% of the theoretical solution and after completing the iteration cycles, its value had not changed significantly. This situation can be explained using the final crack profile, shown in figure (7.33), only a small section of the crack has actually overlapped due to the large displacements of the relatively weak region around the hole.



	L = W = 10 in	Mesh details
ø = 60°	t = 1.0 in	No. elements - 98
	R = 1.0 in	No. nodes - 338
	σ = 12000 psi	Rc = 0.023 in

FIG 7.23 Curved crack in a uniaxial stress field. Example (3).

Case		49C	49A	48B	49A	49C	48C	49C
Angle 'β' degrees		90°	60°	30°	0.0°	-30°	-60°	-90°
Theoretical sol'n Rooke & Cartwright(79)	K_I^*	0.791	0.7445	0.256	-0.163	-0.102	0.381	0.791
	K_{II}^*	-0.116	0.303	0.624	0.488	0.056	-0.233	-0.116
F.E. results 1st Iteration	K'_I	0.794	0.7453	0.271	-0.153	-0.102	0.371	0.794
	K'_{II}	-0.115	0.323	0.634	0.506	0.067	-0.243	-0.115
F.E. results Final Iteration No-Slip Model	K'_I	0.69	0.702	0.271	0.0	0.043	0.372	0.69
	K'_{II}	-0.052	0.334	0.634	0.0	-0.074	-0.24	-0.052
F.E. results Final Iteration No-Friction Model	K'_I	0.779	0.739	0.271	0.0	0.0	0.372	0.779
	K'_{II}	-0.095	0.328	0.634	0.511	0.086	-0.24	-0.095

$$K_I^* = K_I x\sqrt{\sin\phi} \;/\; K_o \qquad K_o = \sigma\sqrt{\pi R \sin\phi}$$

$$K_{II}^* = K_{II} x\sqrt{\sin\phi} \;/\; K_o$$

$$K'_I = K_I/\sigma\sqrt{R}$$

$$K'_{II} = K_{II}/\sigma\sqrt{R}$$

TABLE 7.24. Normalized K_I and K_{II} results for an arcular crack in a uniaxial stress field. Example(3), see figure(7.23)

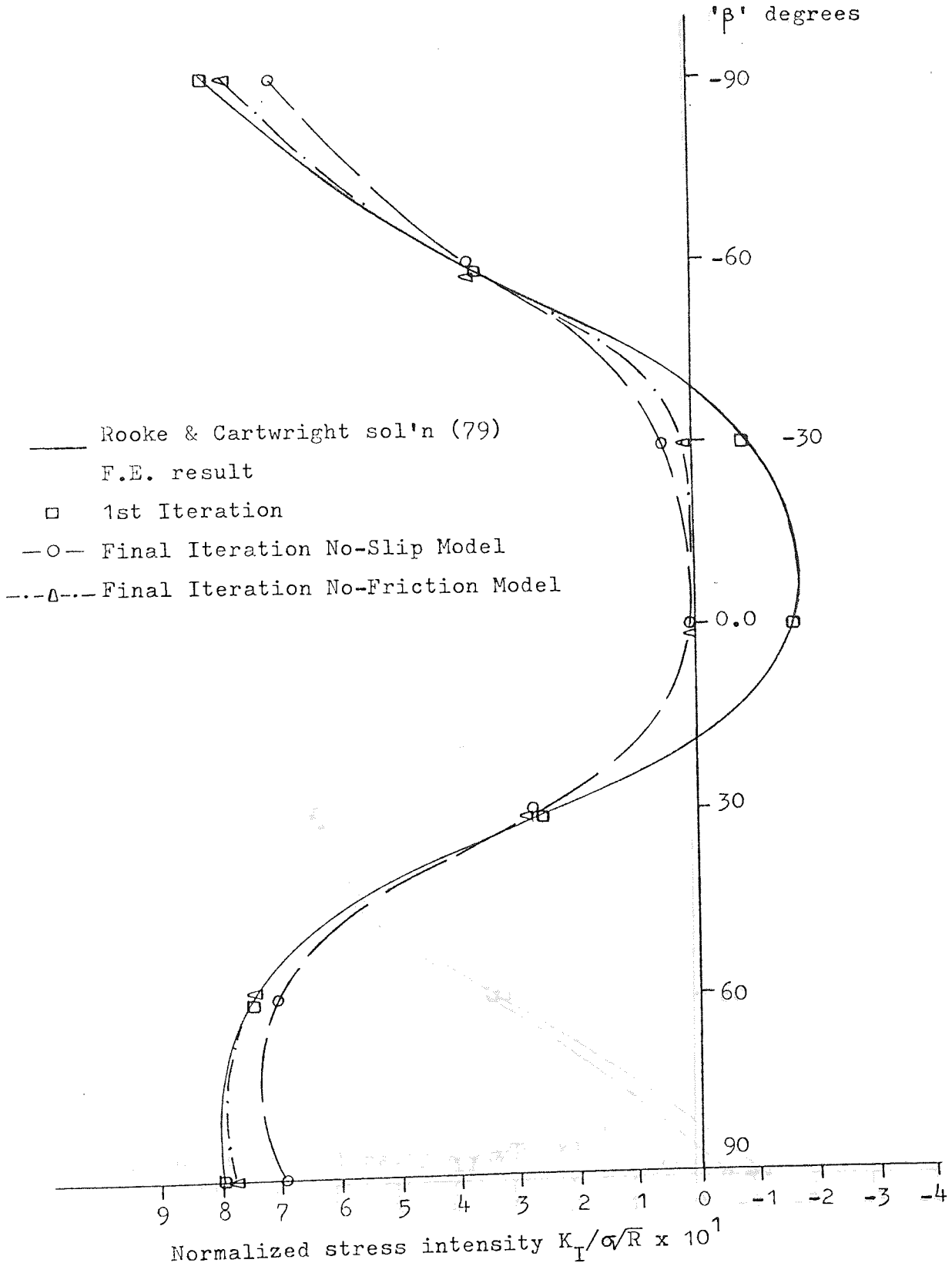
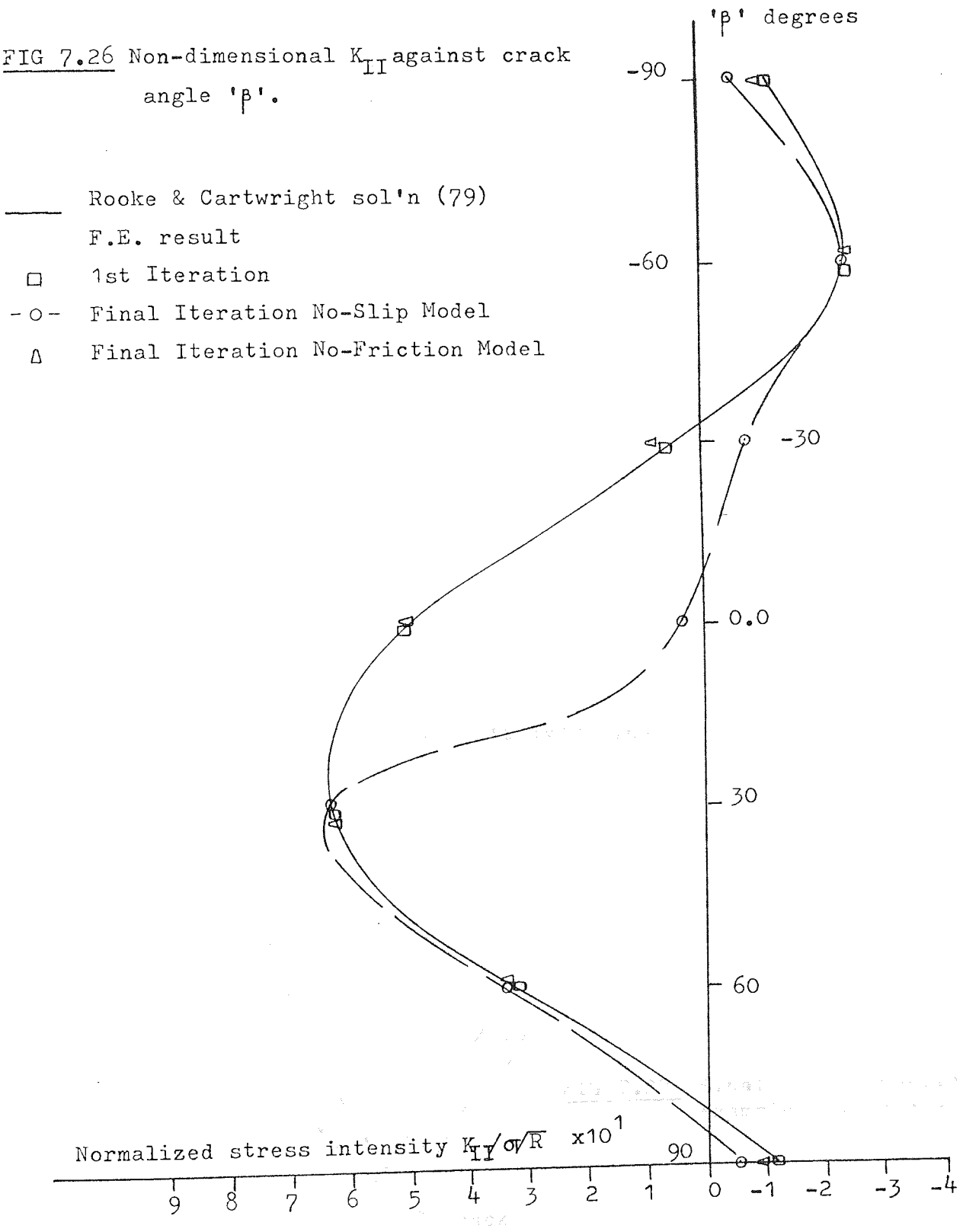


FIG 7.25 Non-dimensional K_I plotted against crack angle 'β'.

FIG 7.26 Non-dimensional K_{II} against crack angle 'β'.



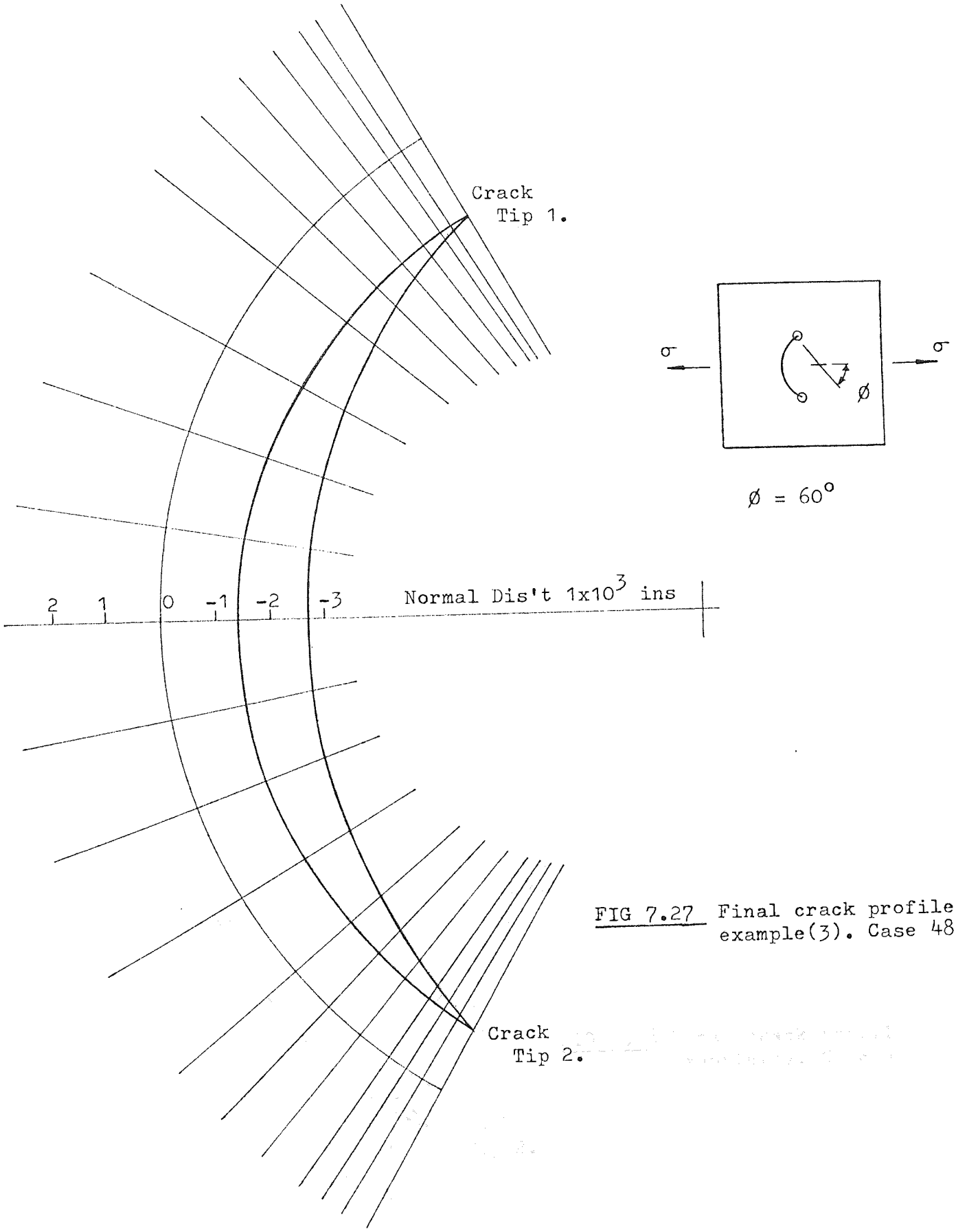


FIG 7.27 Final crack profile example(3). Case 48A

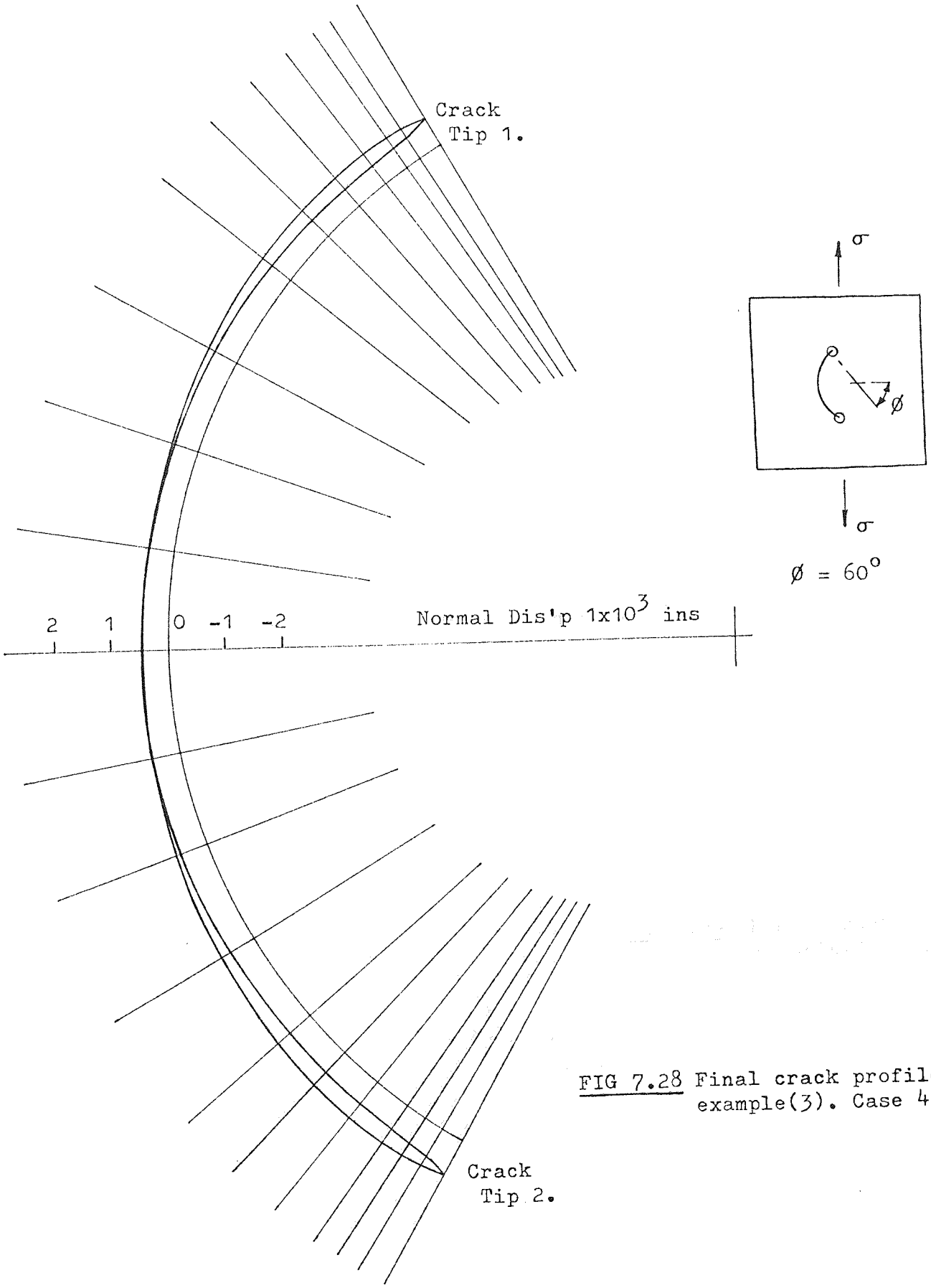


FIG 7.28 Final crack profile example(3). Case 48C

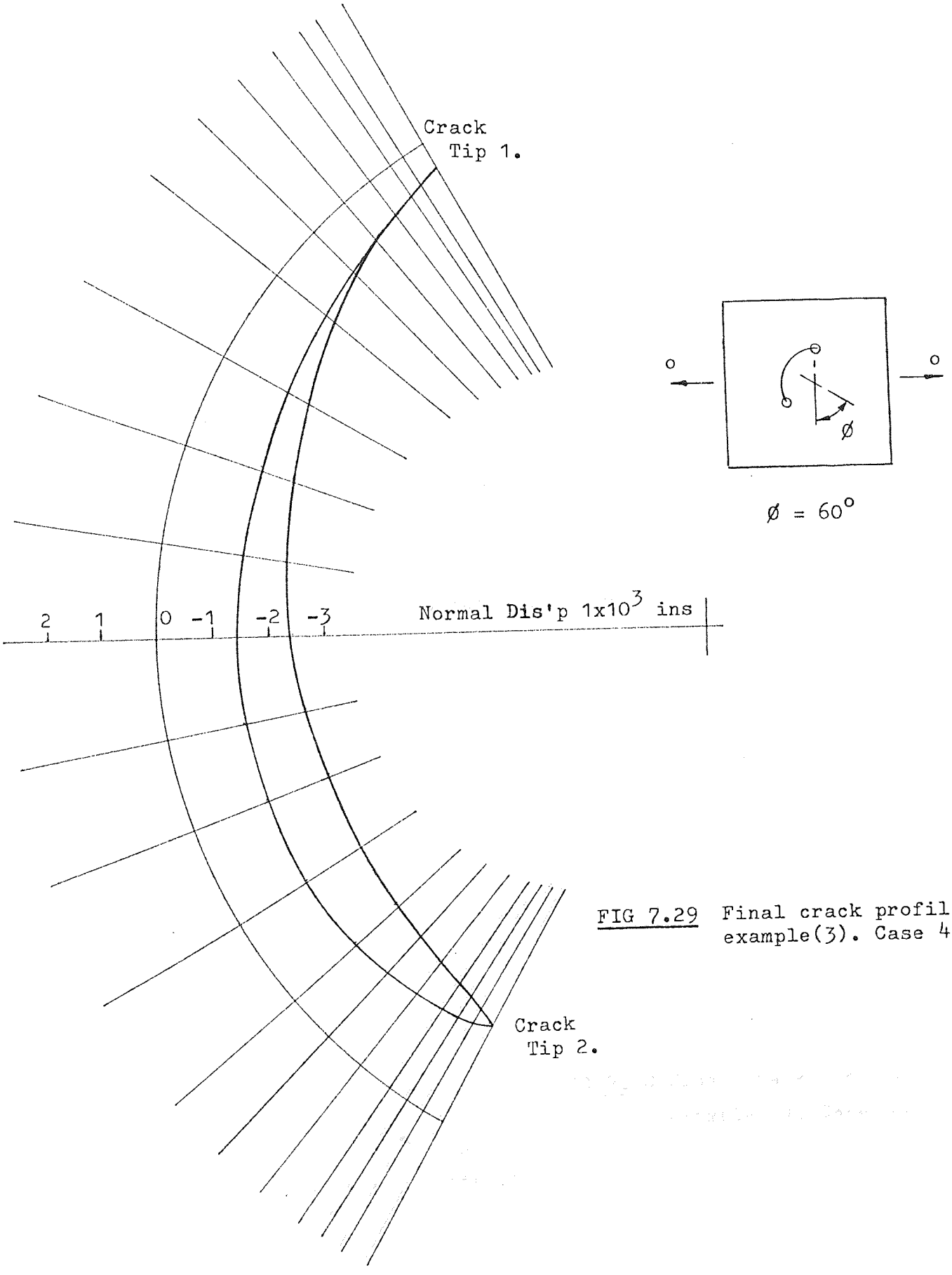


FIG 7.29 Final crack profile example(3). Case 49A

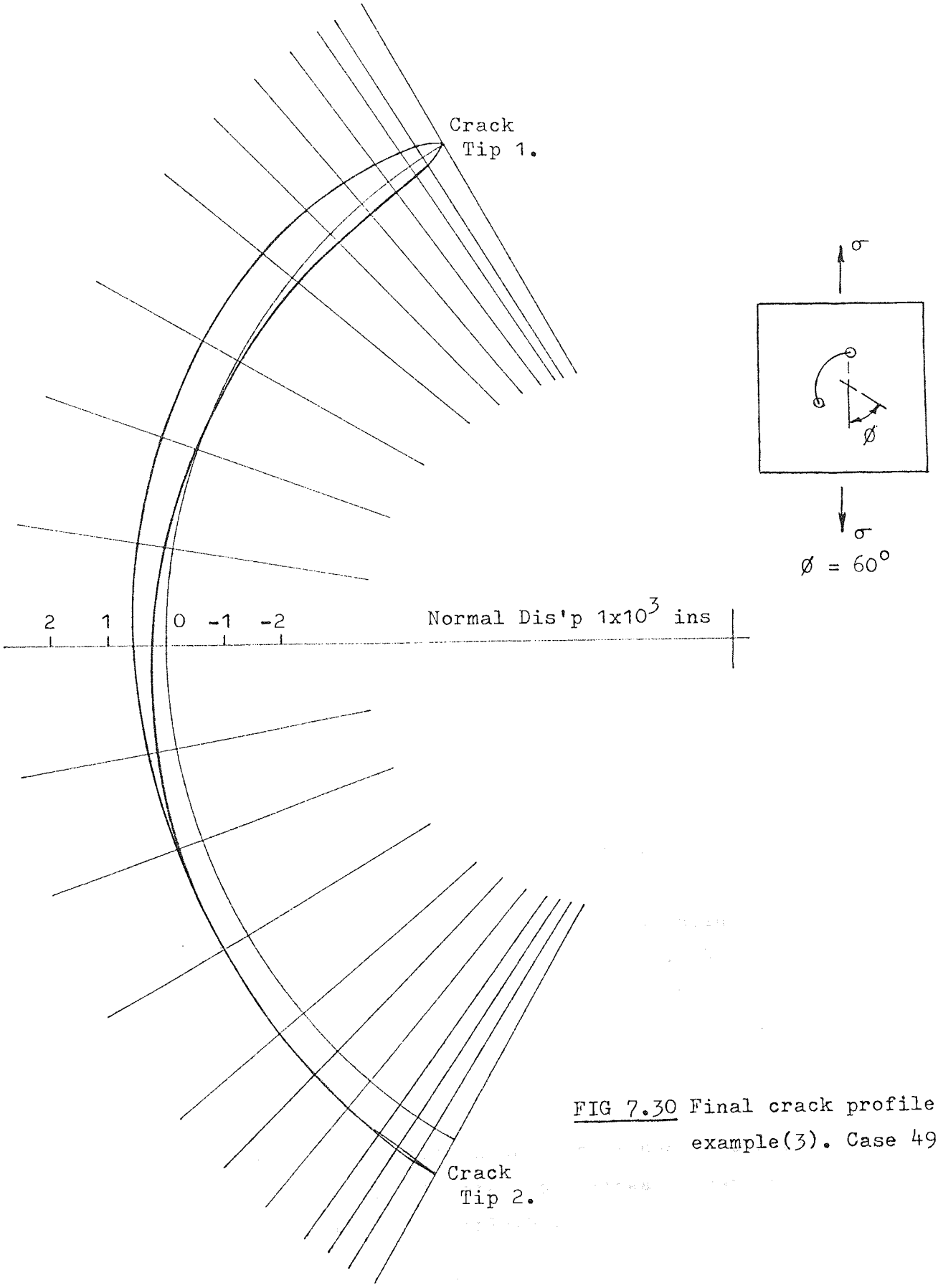
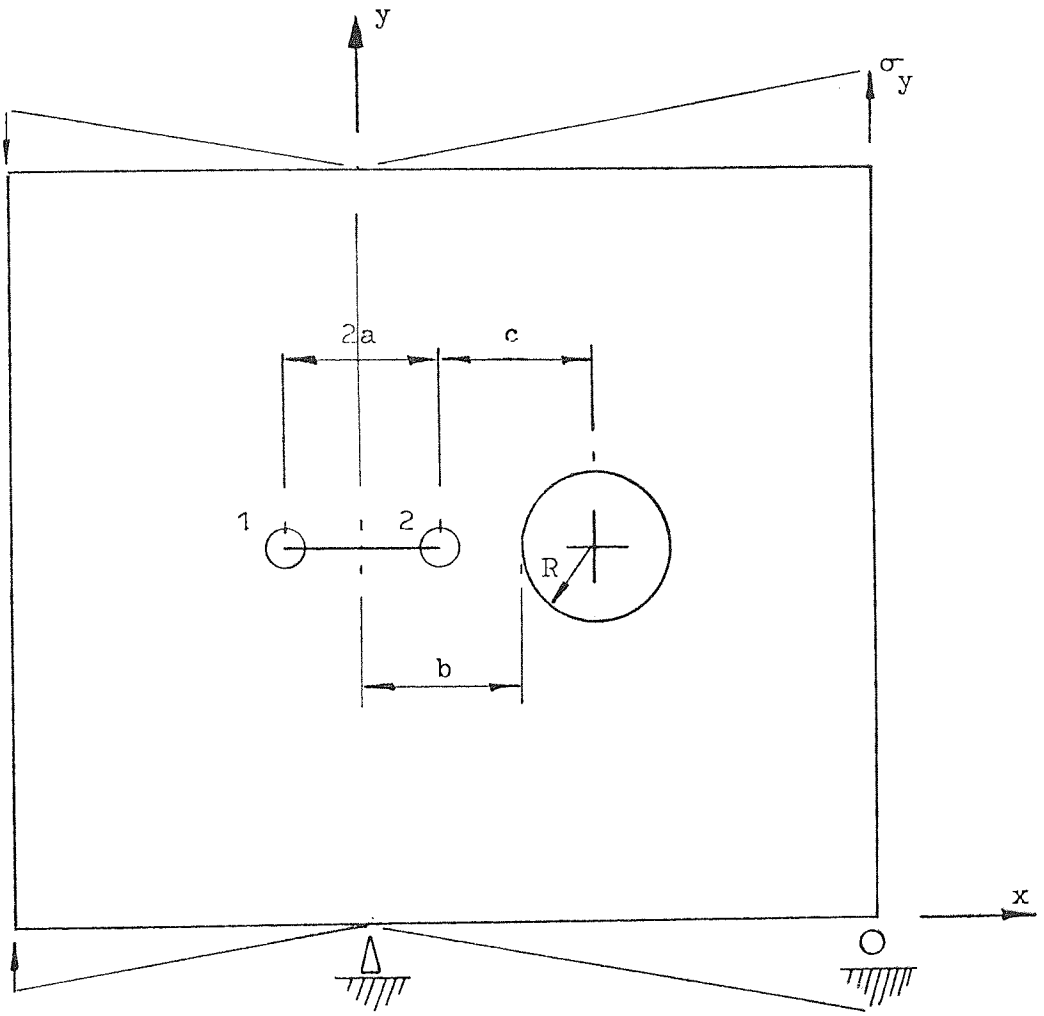


FIG 7.30 Final crack profile example(3). Case 49C



σ_y = Tx

T = 4000

L = 11.0

W = 9.5

2a = 2.4

R = 1.0

b = 1.5

c = 1.3

Mesh details

Rc = 0.024

No. elements - 124

No. nodes - 435

FIG 7.31. Plate containing a crack near a circular hole, subjected to a linearly varying stress. Example(4).

Theoretical solution from Rooke and Cartwright(79). $K_{I1} = 0.0$ $K_{I2} = 0.74$		
Finite element solution	TIP 1.	TIP 2.
Iteration 1.	$\delta x = 1.595 \times 10^{-3}$ $\delta y = 5.610 \times 10^{-4}$ $w = 7.231 \times 10^{-4}$ $K_I = -0.053$ $K_{II} = 0.0$	$\delta x = -1.687 \times 10^{-3}$ $\delta y = 1.174 \times 10^{-3}$ $w = 7.231 \times 10^{-4}$ $K_I = 0.77$ $K_{II} = 0.0$
Iteration 7.	$\delta x = 1.591 \times 10^{-3}$ $\delta y = 5.599 \times 10^{-4}$ $w = 7.233 \times 10^{-4}$ $K_I = 0.0$ $K_{II} = 0.0$	$\delta x = -1.689 \times 10^{-3}$ $\delta y = 1.176 \times 10^{-3}$ $w = 7.233 \times 10^{-4}$ $K_I = 0.772$ $K_{II} = 0.0$

Displacement - ins

TABLE 7.32. Normalized K_I and K_{II} results for example(4), figure(7.31).

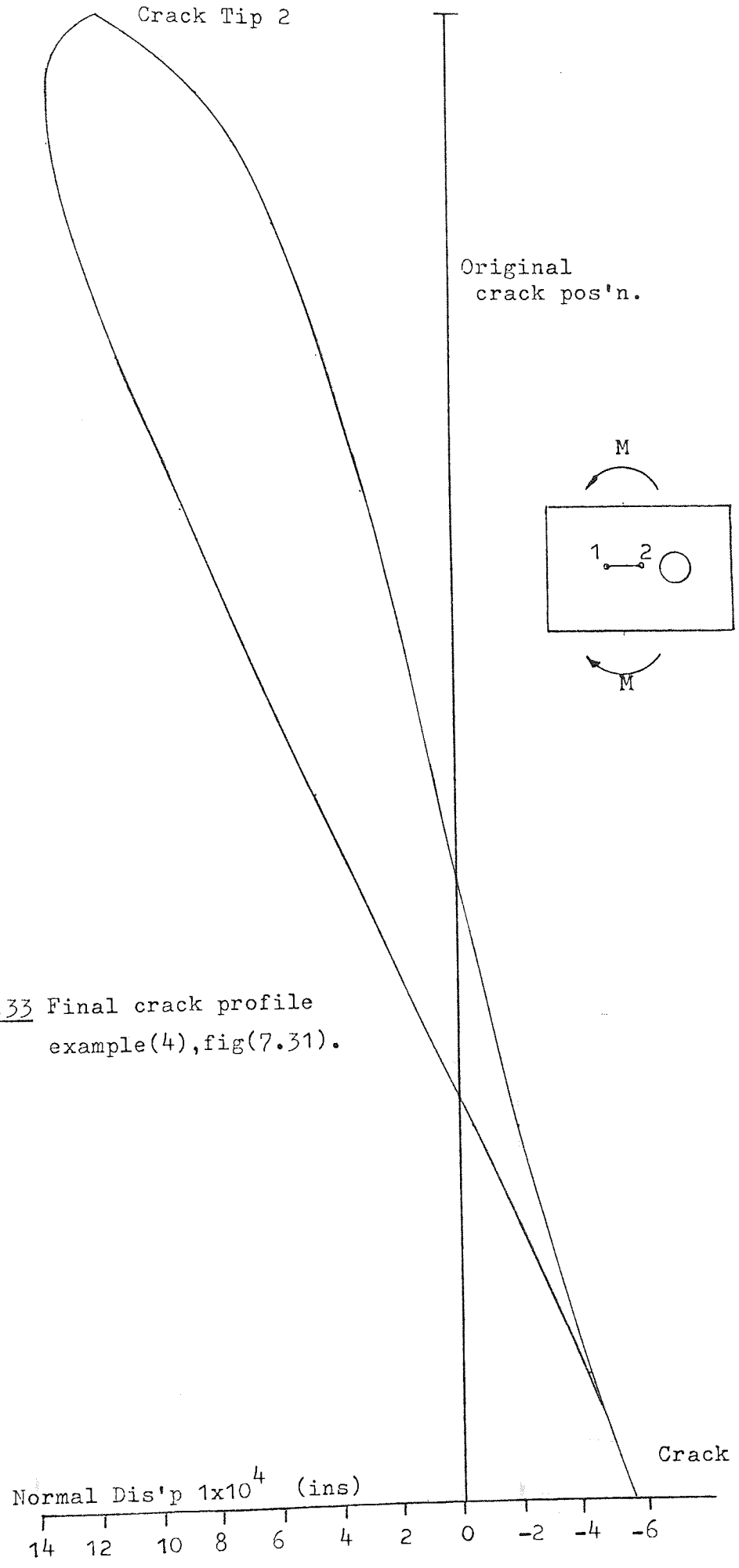
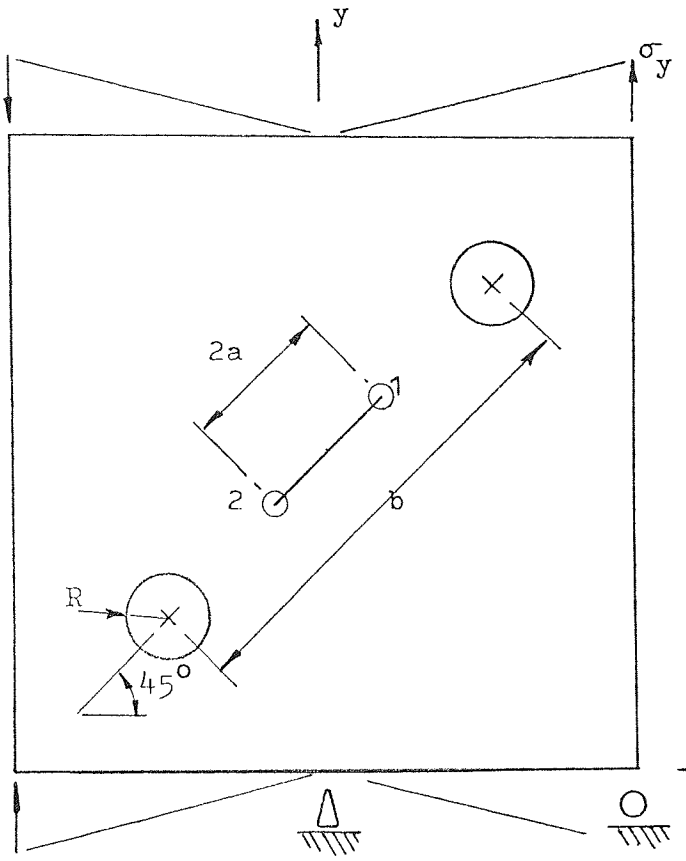


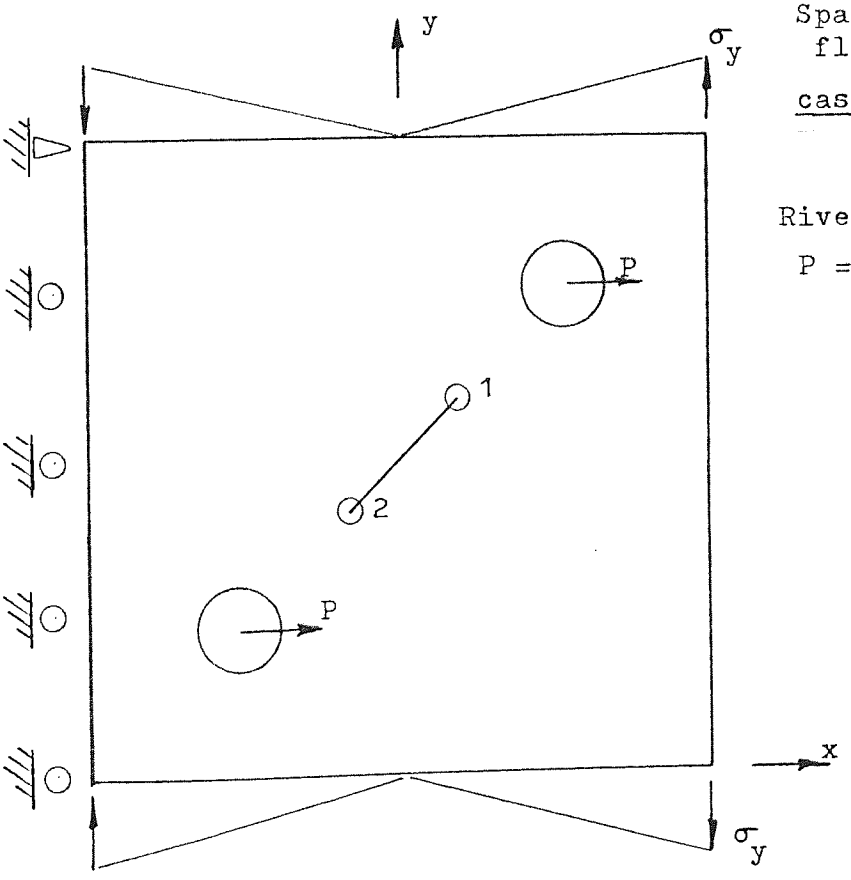
FIG 7.33 Final crack profile
example(4),fig(7.31).



Loading dist'n
case 1.

t = 1.0
W = L = 17
R = 1.2
2a = 3.0
b = 6.8
Rc = 0.05
σ_y = Tx
T = 1 x 10^4

Mesh details No. elements - 116
No. nodes - 414



Spar boom
flange.
case 2.

Rivet loading
P = $30x10^4$ lb

FIG 7.34 Example(5).

Case.1.

Theoretical solution from Bowie and Freese(84)
no holes present.

TIP 1. $K_I = K_{II} = 3199.872$ lbf $in^{\frac{1}{2}}/in^2$

TIP 2. $K_I = K_{II} = 0.0$

Finite element result no-slip model.

	Iteration 1.	Iteration 9.
TIP 1.	$\delta x = -8.525413 \times 10^{-3}$ in $\delta y = 1.147756 \times 10^{-2}$ in $K_I = 4.611397 \times 10^{3}$ psi$\sqrt{in}$ $K_{II} = 4.553698 \times 10^{3}$ " $w = 3.134445 \times 10^{-3}$	$\delta x = -8.572025 \times 10^{-3}$ in $\delta y = 1.159465 \times 10^{-2}$ in $K_I = 5.116828 \times 10^{3}$ psi$\sqrt{in}$ $K_{II} = 5.087265 \times 10^{3}$ " $w = 3.142822 \times 10^{-3}$
TIP 2.	$\delta x = 8.525415 \times 10^{-3}$ in $\delta y = -3.587750 \times 10^{-3}$ in $K_I = -4.611381 \times 10^{3}$ psi$\sqrt{in}$ $K_{II} = 4.55365 \times 10^{3}$ " $w = 2.444495 \times 10^{-3}$	$\delta x = 8.464289 \times 10^{-3}$ in $\delta y = -3.587074 \times 10^{-3}$ in $K_I = 0.0$ $K_{II} = 0.0$ $w = 2.846451 \times 10^{-3}$

Case.2. no-closure.	
TIP 1.	TIP 2.
$\delta x = -1.698936 \times 10^{-3}$ in $\delta y = -3.375524 \times 10^{-2}$ in $K_I = 3.64 \times 10^{4}$ psi$\sqrt{in}$ $K_{II} = -5.9647 \times 10^{4}$ " $w = -5.715322 \times 10^{-5}$	$\delta x = -1.37099 \times 10^{-4}$ in $\delta y = 3.13133 \times 10^{-2}$ in $K_I = 3.1744 \times 10^{4}$ psi$\sqrt{in}$ $K_{II} = -7.091816 \times 10^{4}$ " $w = 9.72579 \times 10^{-4}$

TABLE 7.35. Finite element results for example(5), illustrated in figure(7.34).

The final example consists of a rectangular sheet containing a 45 degree central crack and two offset circular holes of radius R, as shown in figure (7.34). Two loading configurations were considered: firstly, the plate was subjected to a linearly varying stress, and secondly, the plate was considered to be a section from a spar boom flange of an aircraft wing. In the latter problem, it was assumed that the inner face of the flange was clamped, representing the rigid spar centre. The section was subjected to a linearly varying stress and the holes were given simulated rivet loads. For the first loading configuration the results can be compared with the theoretical solution obtained for a 45 degree central crack without holes, as in example two. Table (7.35) shows that the theoretical solution is approximately 37% lower than the finite element result. The proximity of the holes to the crack tip increases the stress concentration giving rise to the larger stress intensity factors derived using the finite element method. The second case, which represents a section from the spar boom flange, can be thought of as a practical application and its solution revealed that, in fact, the crack faces did not close.

7.5 SPECIAL ELEMENTS USED AROUND THE CORE ELEMENT

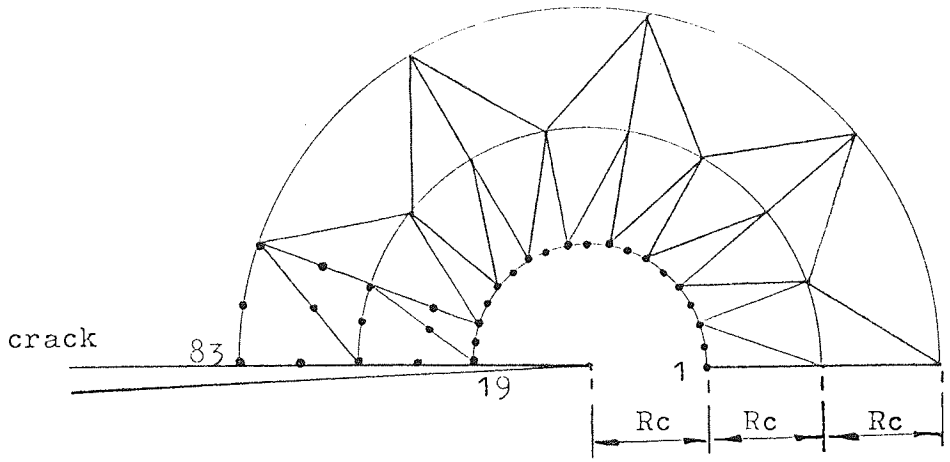
The core element is surrounded by a fine mesh so that the rapid stress changes, in this region, can be accurately interpreted. A series of tests were carried out by Robertson[28] in order to examine the influence of core radii and the number of interface core nodes, on the stress intensity factors. Figure (7.36) illustrates the mode I and mixed-mode local core mesh designs resulting from this investigation. The core radius Rc, was linked with the crack length a, and optimum results were obtained within the limits $a/30 > Rc > a/50$. A similar

local core mesh design was adopted by Alsharqi[29] in the investigation of axisymmetric cracked structures and these are shown in figure (7.37). It was found that a minimum of 17-19 nodes on the core interface produced both consistent and accurate results. If the number of interface nodes is reduced then problems with interface displacement compatibility arise. Because of the standard design of the surrounding core mesh, it can be generated from a few input parameters, as explained in Chapter five. The local core mesh design used in the majority of the previous examples is shown in figure (7.38) and is based on the results of the previous research work.

With the need to reduce the number of nodes used in the finite element mesh idealization, the possibility of reducing the element density around the core region was appealing. A method proposed by Lynn and Ingraffea[58], takes the form of a band of 'transition' elements around the degenerate isoparametric crack tip element. The 'transition' element is described in Chapter (2.5.2) and is essentially an isoparametric quadrilateral element with the mid-side nodes positioned so as to respond to the crack tip singularity. This method has been used in the present work in conjunction with the core element, replacing the three bands of conventional elements by a single ring of 'transition' elements, as illustrated in figure (7.39). The overall effect is a significant reduction in the number of nodes used in the finite element idealization. To determine the optimum 'transition' element configuration, a series of tests were carried out, using the problem of a plate containing a 90 degree edge crack, subjected to a uniform tensile load. The problem was treated as a

mixed-mode case, so that the shearing mode II stress intensity factors can also be monitored. The relevant dimensions of the surrounding transition elements are given in figure (7.39). The outer radius Ro of the ring of elements was related to the core radius Rc by Ro/Rc = constant. Various core radii were selected and the results are plotted in figure (7.40). It can be seen that the core radius is directly linked to the crack length, a, hence the results are applicable to any crack configuration. The transition elements have not been fully tested but the graphical results indicate that the ratio Ro/Rc = 3, produces accurate results within the limits $a/30 < Ro < a/13$. The equivalent mode I case was solved using the semi-circular core element, shown in figure (7.39), with only 9 interface nodes, and the results were identical with those of the mixed-mode example. Hence doubling the number of nodes used on the core interface, when using the mode I facility, as in previous mesh designs, serves no real purpose.

An isolated mixed-mode case of a 45 degree edge crack problem was solved using the new mesh configuration and the results were within 2.0% of those given by Bowie[16]. It is hoped that this modification will overcome the difficulties encountered with insufficient computer storage space.



Mode-I crack tip mesh.

Both designs employ the six-node triangular element.

Mixed-Mode crack tip mesh.

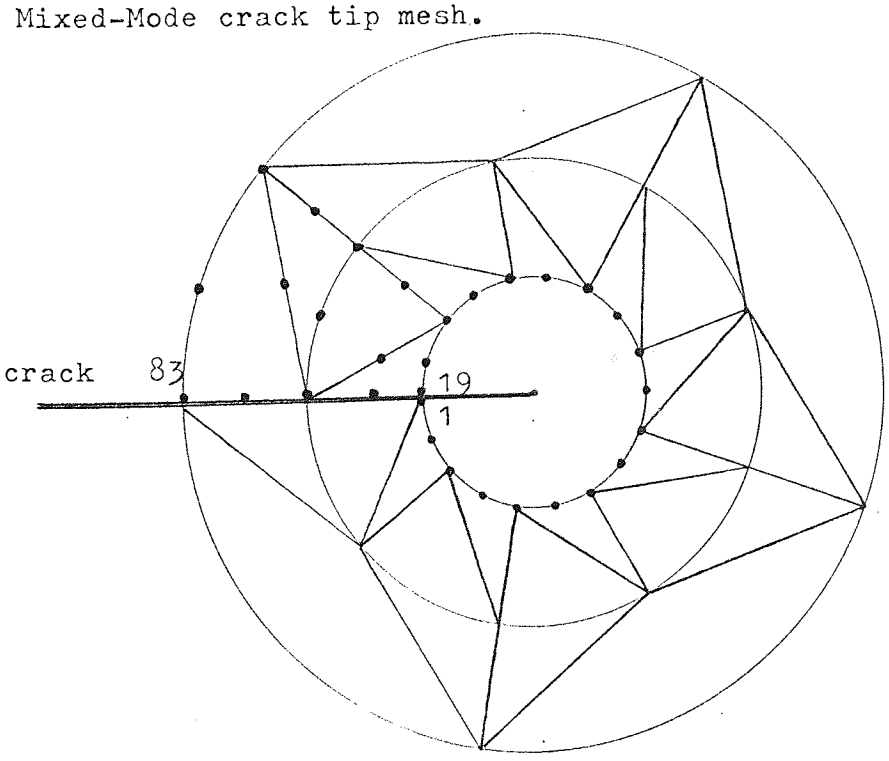
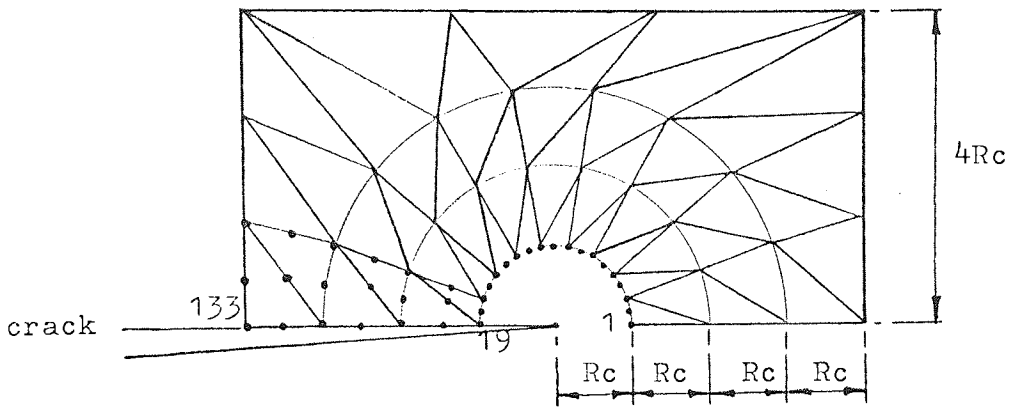


FIG 7.36 Local crack tip mesh design, as used by Robertson(28).



Mode-I mesh design.

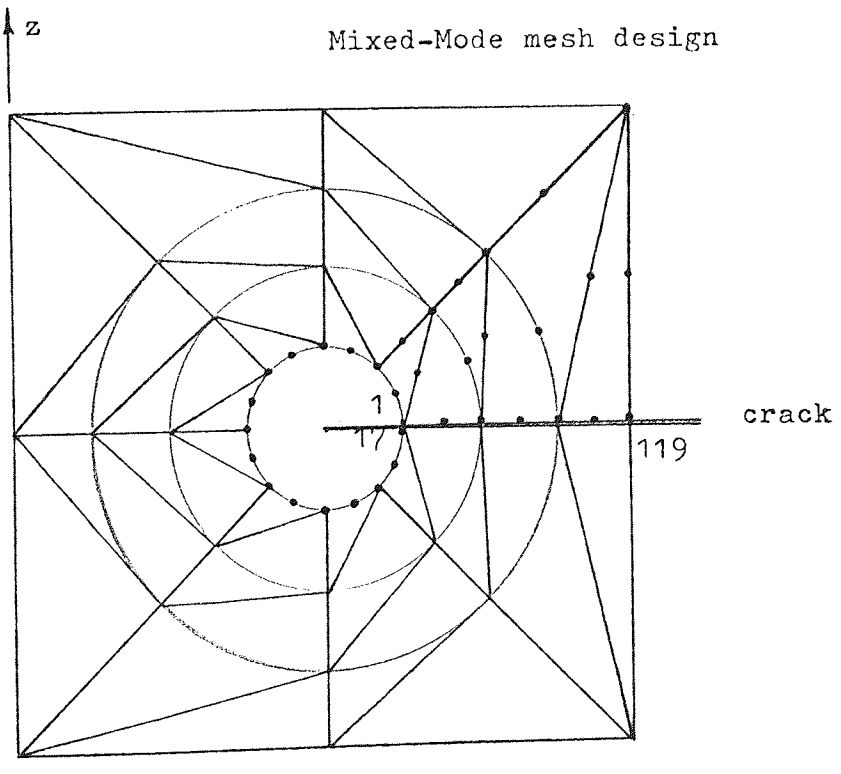
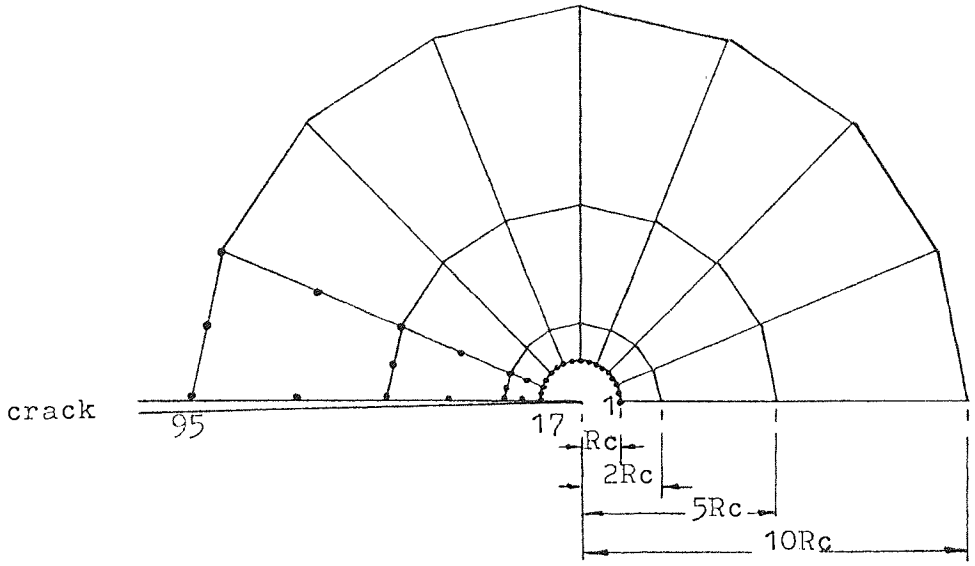
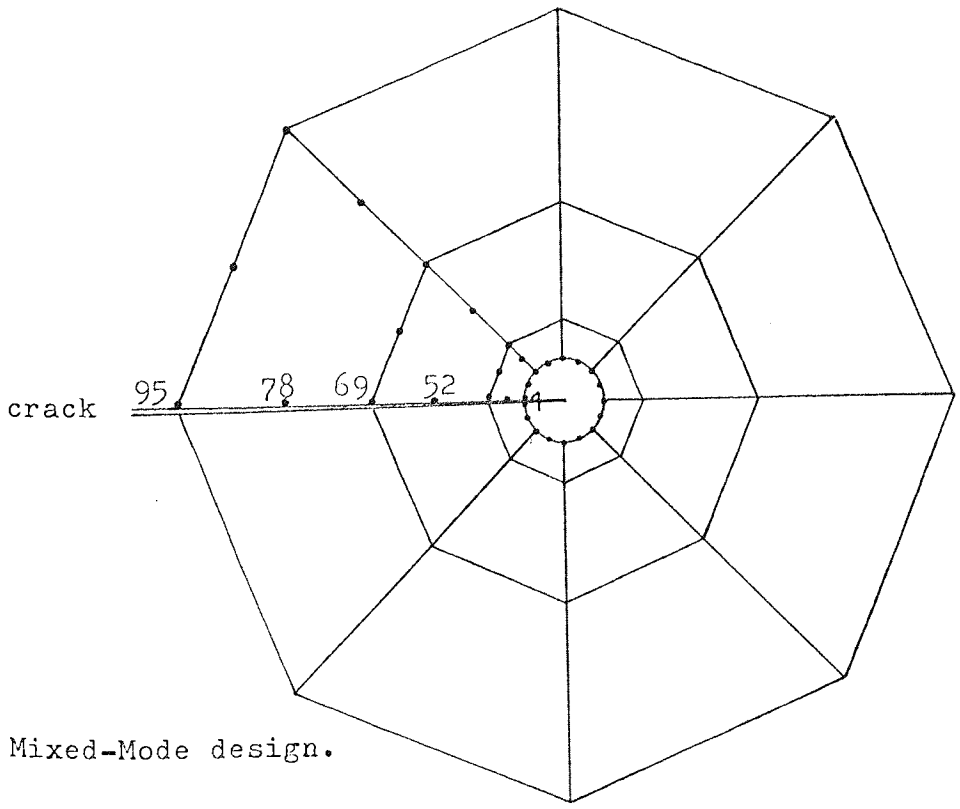


FIG 7.37. Local crack tip mesh design, as used by Alsharqi(29).



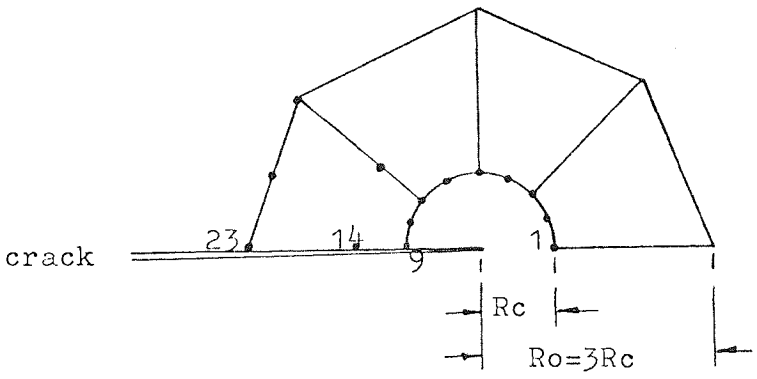
Mode-I design.

Both configurations employ 17 nodes on the core element interface.

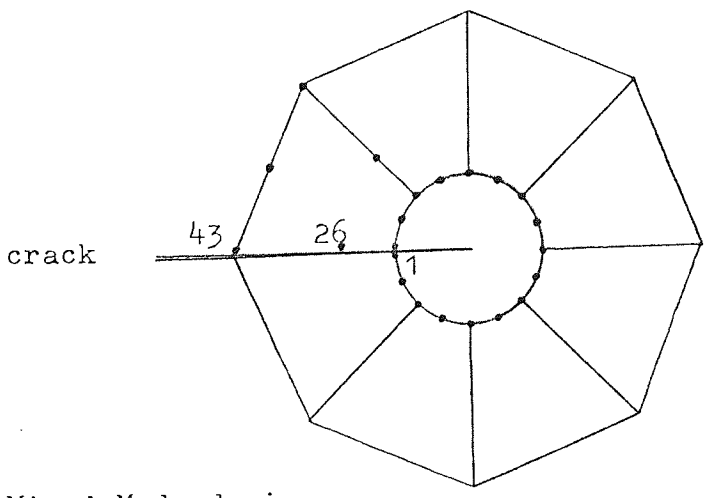


Mixed-Mode design.

FIG 7.38 Local crack tip element configuration using the eight-node quadrilateral element.



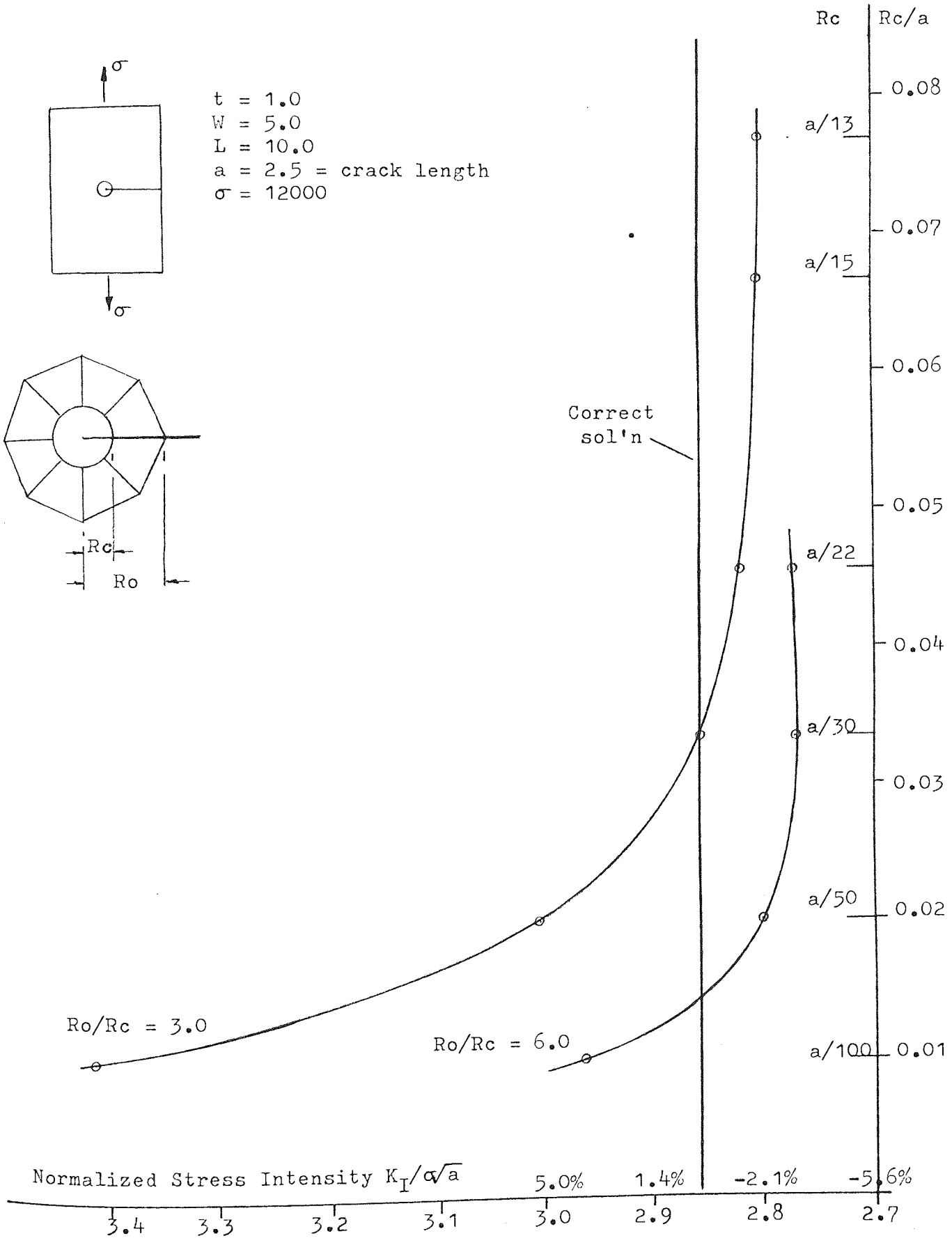
Mode I configuration



Mixed-Mode design

FIG 7.39. Local crack tip element configuration, employing the 'transition' element. Note the displaced mid-side nodes.

FIG 7.40. Normalized stress intensity factor K_I against crack length/core element dimensions.



7.6 CRACK PROPAGATION

The concept of a strain-energy density factor S, has been presented in section (3.2.3), where it was shown that the stationary values of this factor can predict the direction of crack growth under mixed-mode conditions. The expression for the strain-energy density factor is given by equation (3.8), and is a function of the stress intensity factors K_I, K_{II} and K_{III}. The ability to predict the direction of crack propagation is essential in fatigue studies and the finite element method provides a versatile model. Various investigators have used the technique, namely, Ingraffea[85,86] and Fukuda et al.[87].

The incorporation of the crack propagation facility in the finite element method presents no difficulties and it has been done as part of the current project. A standard routine can be called to determine the stationary value of the strain-energy density factor and hence the angle of crack propagation. With the development of the multi-tip and crack closure procedures, of the previous sections, it was found that the program length prohibited the introduction of the crack propagation facility. Basically the method utilized the standard routine of the NAG library and this exceeded the small core memory limits, hence a separate program was written. The small core memory limit is presently being revised and this may allow the crack propagation procedure to be added to the main program. The crack propagation program is given overleaf and the following steps describe its operation:-

1. The program data input sequence is given in a comment statement at the beginning of the program.

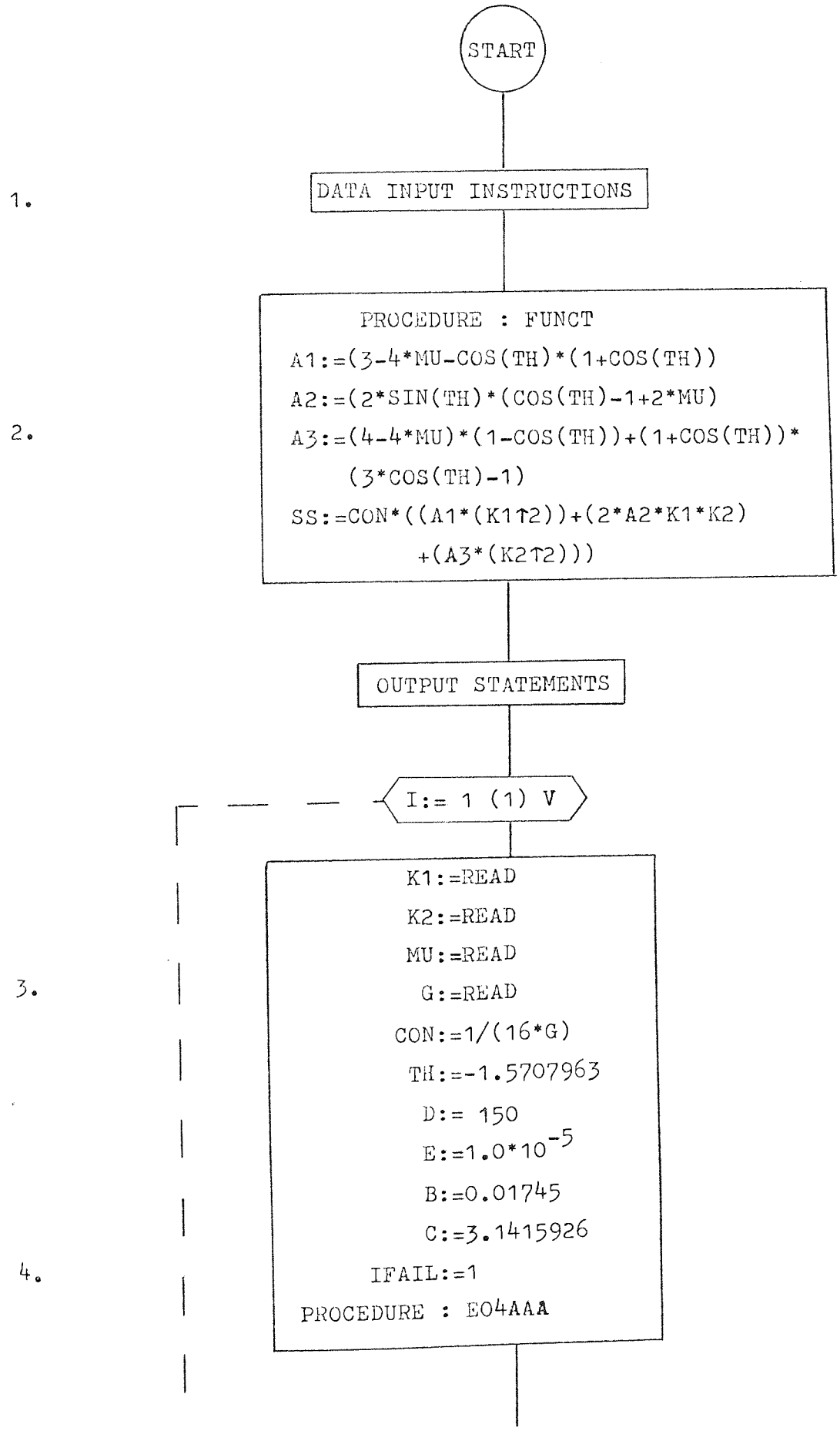
2. Sub-routine FUNCT contains the function to be minimized and is called by the standard NAG routine E04AAA.

3. The data is read into the computer within the control loop I.

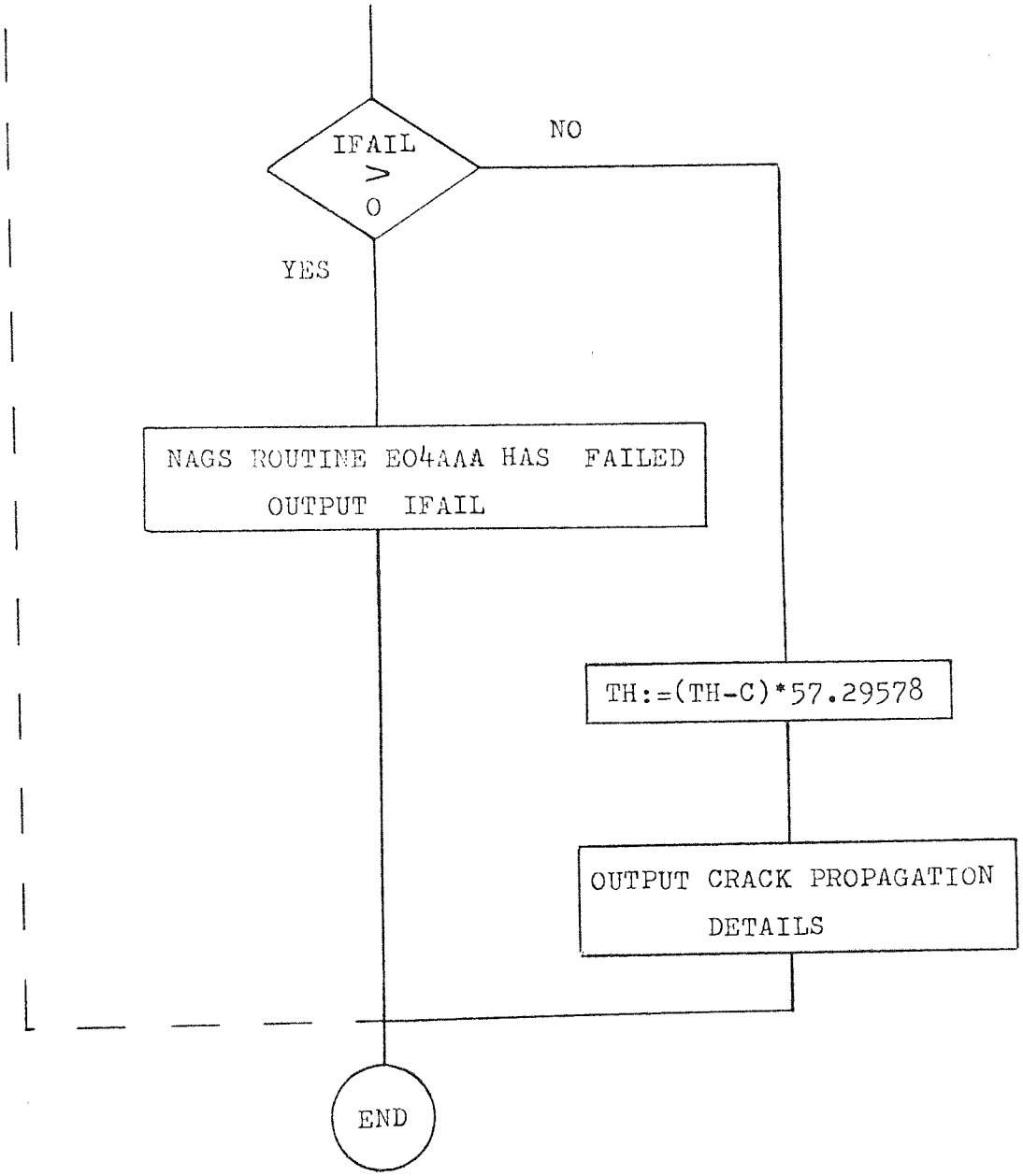
4. Various control variables are defined which are applicable to the NAG routine E04AAA.

5. On exit from the routine E04AAA if IFAIL is zero then the routine has successfully determined the angle of crack propagation and all details are output.

Flowchart for program PROPAG :-



5.


IFAIL
>
0
NO
YES
NAGS ROUTINE E04AAA HAS FAILED
OUTPUT IFAIL
TH:=(TH-C)*57.29578
OUTPUT CRACK PROPAGATION
DETAILS
END

7.6.1 CRACK PROPAGATION NUMERICAL EXAMPLES

The results from some of the previous examples have been used to illustrate the operation of the crack propagation program. The first group of examples are various edge crack configurations, and they are shown with the corresponding propagation angles in Tables (7.41) and (7.42). The fracture angle θ, has been plotted against the crack angle β, and the results are given in figure (7.44). The graphical results have been compared with those obtained by Sih[16], for a plate in uniform tension containing a central crack. The edge crack solutions, for both the curved and straight crack configurations, lie on the same path, and follow the general pattern given by Sih's results.

The second set of examples are taken from section (7.4.3) and represent the central circular cracked plates, subjected to a loading configuration which induces partial crack closure. The crack propagation results are shown diagrammatically in figure (7.43), with the corresponding stress intensity factors. The results have also been plotted in figure (7.44), but it is difficult to interpret any general trend without taking further intermediate cases. The overall results show that the program is suitable for crack growth studies and can be used as a procedure for finite element analysis as given by references (85 - 87).

Root angle 'β'	Angle of propagation Ø
90°	0.0°
54°	35.3°
135°	-44.58°

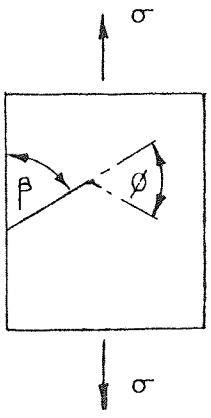


TABLE 7.41. Crack propagation results for a plate with an oblique edge crack

Root angle '∝'	Angle 'β'	Angle of propagation Ø
90	135	-44.0
75	120	-31.14
60	105	-16.9
45	90	0.29
30	75	17.64

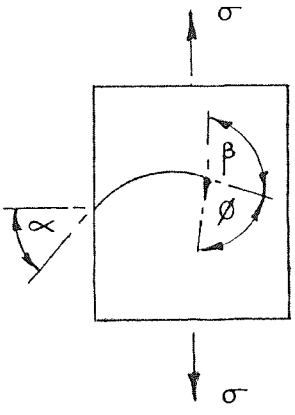
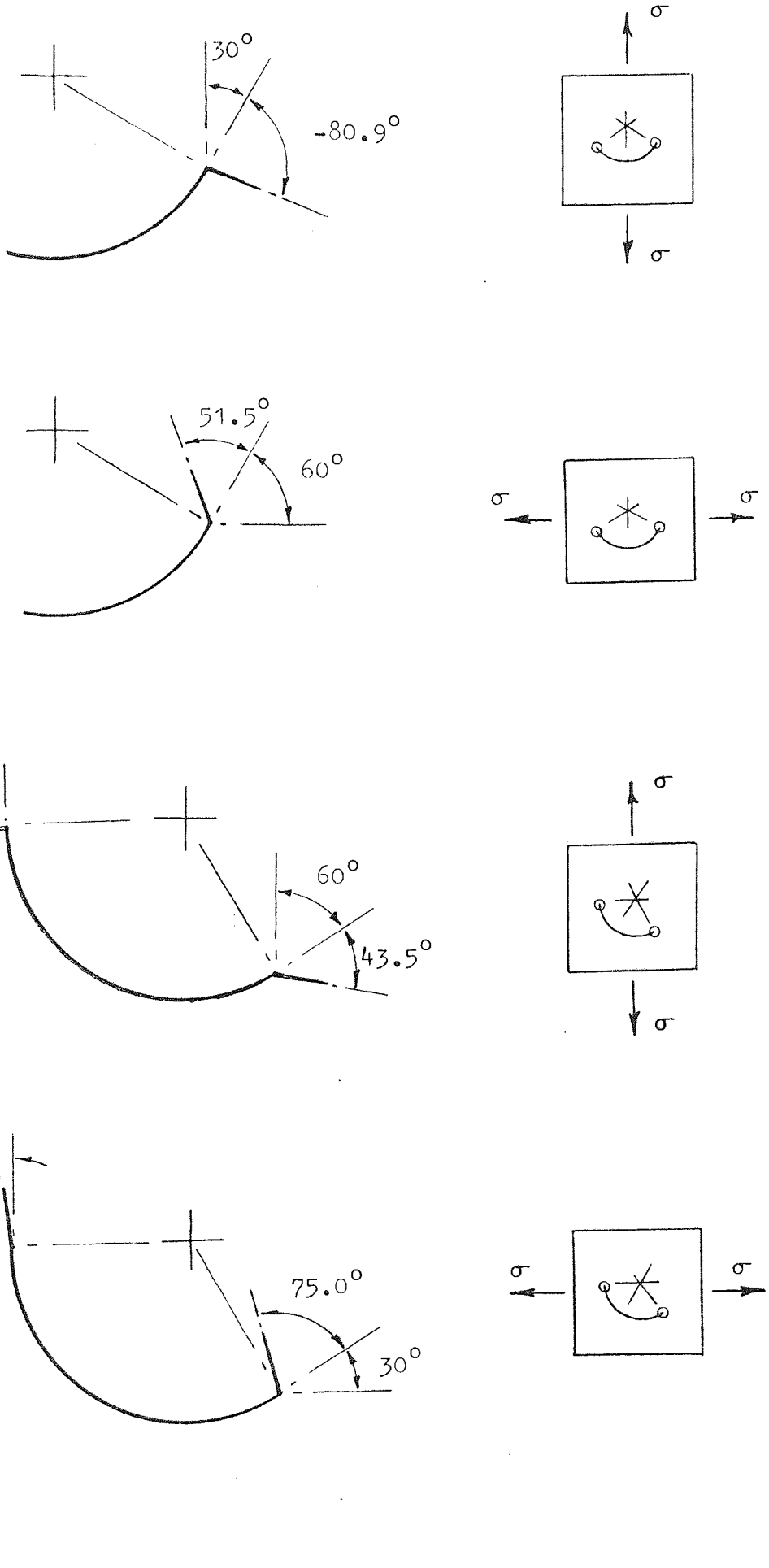
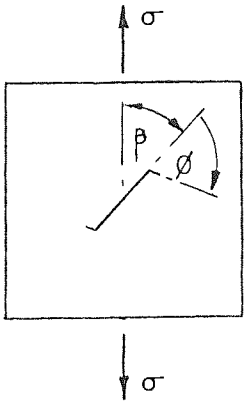


TABLE 7.42. Crack propagation results for a plate with a curved edge crack

FIG 7.43. Computed crack propagation angles for the fracture problems involving partial crack closure of section(7.4.3).





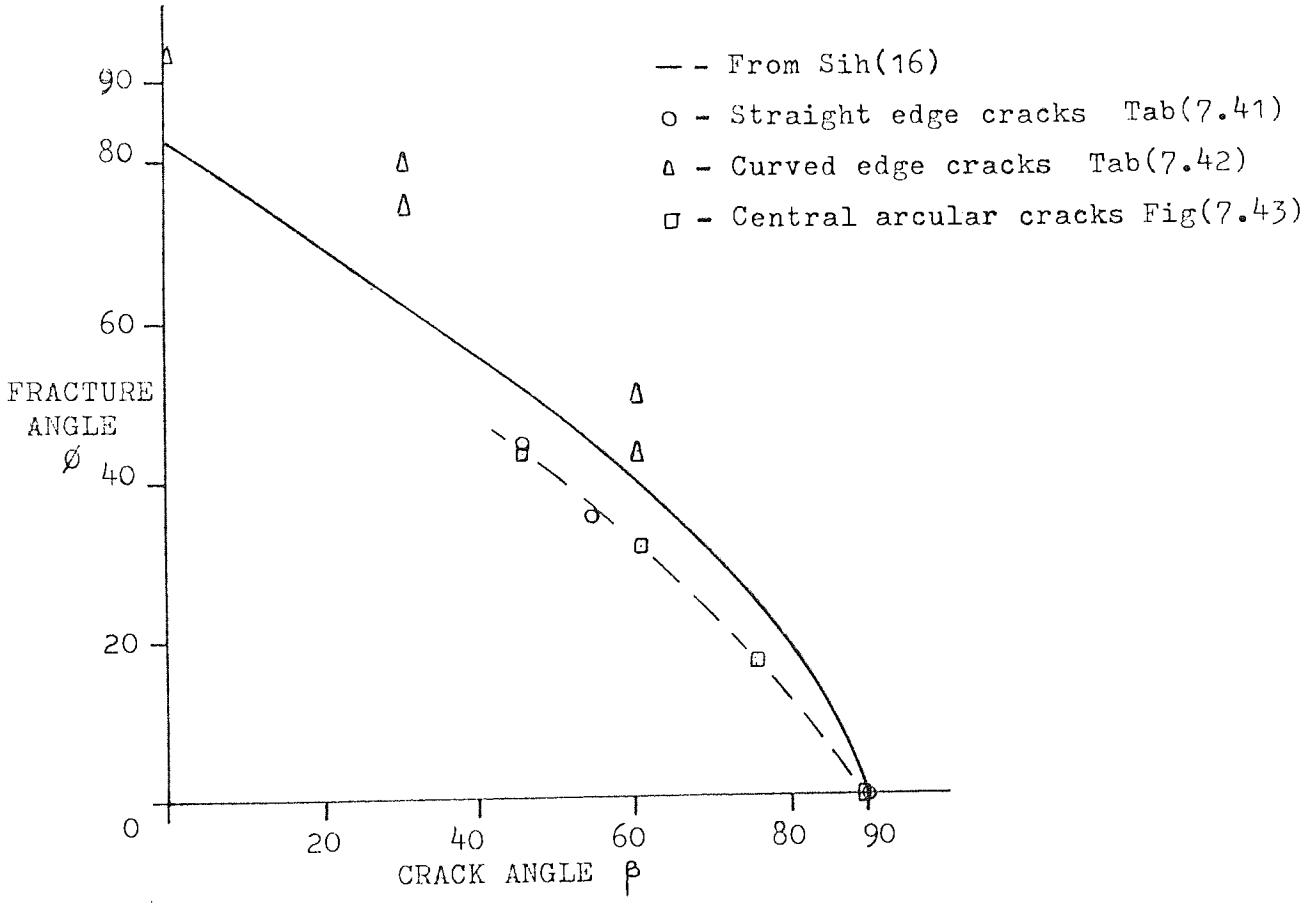


FIG 7.44. Crack angle against fracture angle for various crack configurations.

CHAPTER 8

DISCUSSION AND CONCLUSION

8.1 DISCUSSION

The application of the finite element method to determine crack tip stress fields has seen rapid progress in the past decade. The technique has been finding increasing use in all aspects of fracture mechanics, providing a versatile and efficient engineering tool. As recounted earlier, the Hilton and Hutchinson[51] crack tip element has been employed in this study of cracked structures. Although a small plastic zone forms around the crack tip, it has been shown[14], that this does not significantly affect the system's strain energy and consequently assuming linearity remains a good approximation. Three modes of fracture have been defined in Chapter two: as only two-dimensional cases are considered here the appropriate modes of fracture are the opening mode K_I and the inplane shearing or tearing mode K_{II}.

The work presented in this thesis is founded on the programs written by Robertson[28], whose original objective was to develop a general two-dimensional fracture package. This objective was not fully realised and several problem areas were identified in his suggestions for future research. The suggested work programme has been virtually accomplished and the various difficulties met with and overcome are reviewed here.

The introduction of an automatic mesh generation program, was a primary objective, and has enabled larger problems involving greater element detail, to be investigated with a minimum of input data. The scheme has proven to be flexible and reasonably easy to implement. However, in order to realise the potential of the scheme it is recommended that the user implement the examples given in Section (5.4). The scheme functions as follows: the component is appropriately divided into sub-regions termed 'zones' and an element mesh is generated within each zone to obtain the final discretized component. This process is described in greater detail in Chapter 5. Computational experiments showed that element distortion incurred loss of accuracy in isoparametric formulations and in order to minimize this effect, the program has been designed to make all internal element faces straight with the side node positioned mid-way along the element face. Elements which have faces lying on the zone boundary are allowed to take up the boundary configuration, be it curved or otherwise. This leads on to the problem of element grading, where the mesh generated by a zone can be graded by off-setting the mid-side super node. The zone is an eight-node isoparametric quadrilateral element and in displacing the side super-node all the boundary face element nodes are displaced. Therefore, considering a single element face lying on the zone boundary, it is clear that its mid-side node will also be displaced. This situation is detrimental to the element's performance, as explained in Chapter 2, and thus using the zone's mid-side node to grade the element mesh, is not recommended. Figure (8.1) illustrates the problems involved with the combined facilities of curved zone boundaries and element gradation. Steinmueller (88) has observed this effect and the corresponding

restriction in the application of auto-mesh generation using isoparametric co-ordinates. To overcome this effect it is suggested that the co-ordinates of the element corner nodes are found and then the element mid-side node co-ordinates can be determined using a quadratic function based on the shape of the zone boundary. In this way loss of accuracy can be minimized and element mesh grading can be used to greater effect.

The various finite element programs have been modified to accept the new input data format presented by the auto-mesh generation scheme, and a program index is given in Appendix (9.3). Each program requires a different data format and the mesh generation data output is controlled by a program 'code' number. The only programs which have not been updated in this manner are the axisymmetric fracture programs, and these need to be updated to bring them in line with the basic axisymmetric finite element program IAAXMG. See Chapter 4.

During the initial developments of the two-dimensional finite element programs, spurious shear components were found on stress free boundaries and this was first thought to be due to a programming error. It was shown, after subsequent investigation, to be due to element distortion which degenerated the element shape function causing loss of accuracy. This aspect of element behaviour has been dealt with in Chapter 2, and it is worth noting here that in problems of this kind, the program should be tested against the simplest element configuration possible. Thus high-lighting any malfunctions, which can subsequently be checked by hand calculation.

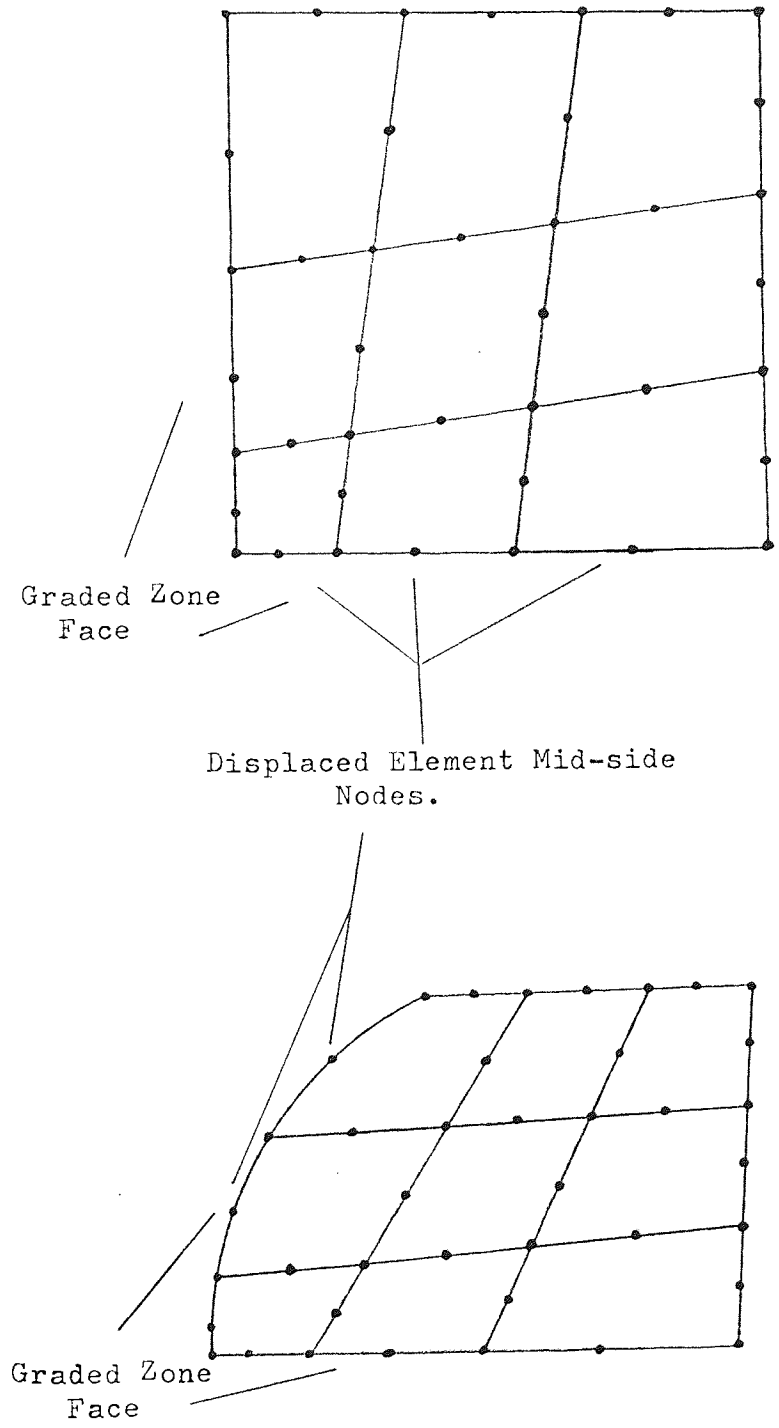


FIG 8.1 The two figures show how the element mid-side nodes on the zone boundary, are displaced when the grading facility is used.

A second element type was introduced to the two-dimensional finite element programs, namely the eight-node isoparametric quadrilateral element. Few problems were encountered with the introduction of the new element type and Chapter 4 describes the modifications necessary for the incorporation of any element type. With reference to the assembly routines, it is possible, with hindsight, to say that the routine can be condensed to a much greater extent if the matrix manipulations were to be carried out within a set of control loops, rather than in the more explicit form presently used. This would allow the two assembly procedures, which exist at the moment, to be combined and also save computer storage space. As the trend is moving toward the use of mini-computers, where storage space is at a premium, the same criterion will apply, that is, where explicit matrix manipulations are carried out, these should be condensed as previously indicated.

With the ever increasing use of the computer centre facilities at Aston University, users running large programs were advised to move to the capacious computing facilities at Manchester University, to which Aston is linked. Having transferred the programs, the initial difficulties of adjusting to the new system were overcome and consisted basically of print and conditional statement variations, and coping with the CDC 7600 operations. Appendix (9.2) elaborates on the use of channel cards, with respect to data transfer using the backing store and 1900 computer. Having established the programs, it was found that, in some cases, the size of the stiffness matrix could not be held on the large core memory and several methods were investigated, in order to overcome this difficulty.

Firstly a more flexible solving routine was introduced based on the scheme proposed by Jennings and Tuff[69], and provided a very compact method of solving large sets of simultaneous equations, stored in a segmented form. The scheme was hampered by the method of assembling the stiffness matrix and also by the matrix manipulations implicit in the fracture formulation. Both of these procedures require the matrix segments to be transformed to and from the backing store in a random manner, and severely detract from the solving routine's efficiency. The assembly procedure cannot be performed on a segment by segment basis, as the nodal numbering of each element dictates the location of the stiffness coefficient within the overall array. As the numbering system is random, so the location of each coefficient is random, therefore it is not possible to reduce the inefficiency of this process. The re-structuring of the stiffness matrix to incorporate the singularity element, however, could be examined more closely and methods of optimising this procedure should be examined. An example of the increase in cost incurred when using the segmented solving routine as opposed to the SYMVBSOL procedure, can be taken from a simple case of a notched bar specimen, having 600 degrees of freedom in the discretized model; the cost was 9 units in the segmented scheme as against 5 units using the SYMVBSOL procedure. The scheme could easily be adapted to small computers with limited core space, but with backing store facilities.

Secondly, two indirect methods were examined which would solve the computer storage problem, namely the 'pseudo-inverse' technique and refinement of the crack tip core element. The first method has

been described in detail, see Section (7.3.2), and functioned correctly for a number of test cases considered. Unfortunately it was found that for the particular case of a 90 degree edge crack, the displacement field was not interpreted correctly, and although efforts were made to resolve this situation, the malfunction was not satisfactorily explained. All ancillary routines functioned correctly. The pseudo-inverse matrix, denoted by [A*], was obtained via the Nottingham routine F01BLA. This is a 'black box' routine and the solution was checked using the relationship [A*][A] = [I], and subsequently listing the coefficients of the identity matrix [I]. The diagonal coefficients were all found to be equal to unity and the largest off-diagonal terms were of the order 3.0×10^{-7}, which would indicate that the pseudo-inverse routine was operating successfully. A listing of the residual forces gave no indication of any applied tractions which would account for the negative displacements registered in an essentially positive displacement field along the crack faces. Further examination of the method was precluded due to time restrictions and the need to investigate more immediately important topics, and was subsequently abandoned with the development of the revised Hilton and Hutchinson technique. The second, of the indirect methods, proposed, benefits the fracture program in two ways, firstly the core element would provide a more accurate estimate of the stress intensity factors and secondly the core element could be enlarged so that the dense local crack tip mesh could be dispensed with. To this end a procedure was developed and has been explained in detail in Section (7.3.4). As this development will be looked at again in the context of the multi-tip procedure, only its advantages will be given here.

To ease the problems encountered with computer storage space, the element density around the crack tip core element was relaxed and instead of using 21 interface nodes, only 17 were employed. This allowed further work to continue and the problem of partial crack closure was investigated. As recounted in Chapter 7 the problem of overlapping crack surfaces had been looked at by Robertson[28], but a satisfactory model had not been devised for the general case. Robertson used an average displacement scheme, whereby, on solving the system equations and identifying that crack closure had occurred, then the adjacent nodes, on the crack surface, would be specified displacements corresponding to their average overlapped displacements. Therefore, using the standard geometric boundary condition routine GEOMBC, the adjacent crack face nodes would have the same displacement on solving the modified system equations. The scheme, however, is only valid for mode I crack configurations and breaks down in mixed-mode fracture problems. A second method which is conceptually valid for the mixed-mode crack closure configuration, is based on the use of bar elements. These are used to bridge the closed adjacent crack surface nodes and the method is discussed in greater detail in Chapter 7. Subsequent test cases show that in assuming various material properties and physical dimensions, for the inserted element, the final crack profile can be arbitary. As the bar element stiffness is increased the resulting closed crack nodal displacements tend to zero and this is due to the large diagonal stiffness coefficient, corresponding to the bar element, dominating the smaller coefficients in the same row. This tendency is shown clearly in figure (7.19).

As both of the above schemes proved inadequate for mixed-mode fracture problems, a technique used by Iversen[83] was adopted. This method essentially couples the adjacent nodes normal displacements, preventing element overlap, but at the same time allowing the crack faces to slide in a tangential sense. The technique has been used to provide a no-slip model and a frictionless model for the partial crack closure problem. The no-slip model was used to compare the results obtained using the finite element method with those obtained theoretically by Bowie and Freese[84] for the mixed-mode crack closure problem. The finite element solutions are set out in Section (7.4.3) and correspond with the theoretical solutions to within 1.3%, consequently this method was used to examine further crack closure problems. In particular, the problem of a circular crack sited at the centre of a sheet in uniform tension, had not previously been solved satisfactorily. The solution presented by Rooke and Cartwright[79] gives negative values for the mode I stress intensity factors, implying that the crack surfaces have overlapped. Both finite element models were used to examine the problem and the results are plotted in figures (7.25, 7.26). The figures show that there is good correlation between the theoretical solution and the finite element results, provided crack closure does not occur. In the latter case, the new finite element results are considered to be most satisfactory.

The technique is not restricted to fracture cases involving overlapping element faces, the scheme can be used in any situation where element boundaries may overlap, indeed it can be seen as a further boundary condition and could be implemented as such in further program developments.

In cases where the crack tip became closed, the stress intensity factors did not, in the majority of cases, assume a value of zero. In the no-slip model, for example, K_I and K_{II} are theoretically zero in the above case. The reason for this behaviour stems from the fact that the core element is in effect subjected to the local shearing and compressive loads, and therefore interprets this loading in the usual manner. This problem was overcome by simply forcing the appropriate stress intensity factors to be zero, using the standard method of boundary condition treatment via procedure GEOMBC. It was noticed that in forcing the stress intensity factors to be zero, there was very little change in the overall solution.

It is interesting to examine the validity of the two crack closure models. Firstly the no-slip model couples the crack face adjacent nodes in a complete sense, in effect creating false crack tips, see figure (8.2). Obviously the mode I stress intensity factor of such a false crack tip would be very small due to the local compressive stress field, but in problems where a high shearing stress exists then the false crack tip might generate a stress field which could affect the overall solution. Secondly, the frictionless finite element model allows crack faces to move freely in a tangential direction and so the problem of a false crack tip does not arise. However, partially closed cracks are subject to frictional forces, and therefore this model represents the other extreme given by the no-slip case. In order to progress with the partial crack closure model it is necessary to gain experimental information, either through the literature[89] or from laboratory tests. In this way frictional effects assumed in the finite element model can be directly related to experimental data.

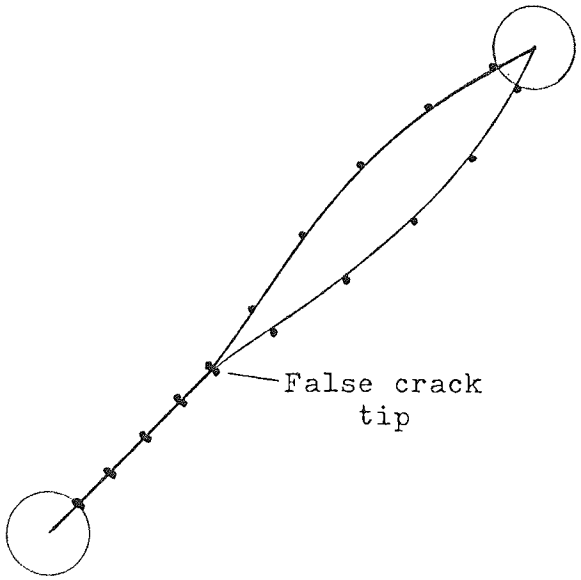


FIG 8.2 Final crack profile using the no-slip finite element model.

Clearly equal and opposite frictional loads can be applied to the closed adjacent nodes, but, without knowing the normal crack face loading this frictional force would be arbitrary. To obtain the normal crack face loading it would be necessary to determine the normal stresses and infer the loading conditions from the closed crack area. This process could be 'triggered' at the end of the iteration cycle and the system equations could then be re-solved using the modified force vector. A subsequent check would be necessary to determine if the applied friction forces are representative of the final crack configuration.

As part of the general development of the fracture package, a multi-tip procedure was written and tested against a wide range of problems, see Section (7.2.4). In the single and double crack tip configurations, the results obtained were identical to those determined using the specialized fracture programs. Also in problems containing more than two crack tips, the results obtained using the multi-crack tip procedure, were comparable to the theoretically determined solutions. Coupled with the multi-tip procedure was the development of the crack tip core element, whereby the number of terms taken in the singularity expression was made optional. This facility has been mentioned earlier in context with computer storage space limitations, and it was explained that the procedure did not operate correctly when more than two terms were taken in the singularity expression. To determine the reason for this situation, various steps were taken. The first four x four coefficients of the core strain energy matrix $[k_c]$, had been evaluated, explicitly, by Robertson and

the numerical values generated by the procedure CORK were identical to those obtained using the explicit expressions. Refer to Section (7.3.4) for a detailed description of procedure CORK. The CORK procedure was overridden and the explicit terms used to form the matrix $[k_c]$, in an attempt to trace the malfunction, but this produced the same solution discrepancies. The matrix [A] connecting the core element parameters with the interface nodal displacements, was printed out and checked by hand calculation, and found to be correctly compiled. Similarly the various matrix assembly stages were checked by printing the relevant matrices, and here again the operations were found to be functioning correctly. In order to resolve this problem, two possible courses of action are suggested; firstly, in cases of this kind where no fault can be found in the implementation of the theory, then the theoretical working should be checked, and secondly, if this proves negative, then the appropriate procedures should be tested.

With the need to reduce the number of degrees of freedom in the idealized component, the possibility of reducing the element density around the crack tip region, would be immensely helpful. The 'transition' element, introduced by Lynn and Ingraffea[58], allows the number of elements in crack tip region to be reduced significantly, saving 104 d.o.f. in the normal local crack mesh configuration. This represents a considerable reduction in the overall number of nodes required in any multi-tip problem, for the same solution accuracy. Further tests need to be undertaken before this new element construction can be used with confidence. From the results of Section (7.5), there are indications that a smaller outer mesh radius to inner core radius R_o/R_c ratio would allow the dimension of the core element radius to be

more flexible. From figure (7.40) it can be seen that the solution for the two R_o/R_c ratios chosen, is essentially straight over a wide range of core sizes, and the trend indicates, that this section of the curve, could be made to coincide with the theoretical solution, if a smaller R_o/R_c ratio was used. If this is the case then obviously a larger core radius would be used.

To supplement the suggestions which have arisen in the course of the discussion, a list of further program developments and research topics suggested for future study are given below:

(a) It has already been suggested that where possible explicit matrix manipulation should be condensed in order to save computer storage space. For example in the assembly procedure the matrix operation $[B]^t[C][B]$ can be carried out as follows:

```
FOR P:=1(1)12+4*QORT
FOR N:=1(1)P
FOR I:=1(1)3
FOR J:=1(1)3

K[P,N]:=K[P,N]+(B[J,P]*C[J,I]*B[I,N])
        *W[U,3]*W[U,4]*DETJ*TH;
```

where QORT is dependent on the element type. In using this format both of the assembly procedures Q/FEASSEMBLY could be combined. Similarly procedures FENOSTR and FEELSTR should be modified in this fashion.

(b) An investigation of the degenerate isoparametric crack tip element should be undertaken, as the method has proven to be efficient and accurate[57,58]. With respect to mini-computer applications, the scheme requires no modification of the conventional finite element method, except for an additional procedure for interpreting the local crack tip displacement. Because of the technique's simplicity it would be ideally suited to work requiring three dimensional analysis and also in the investigation of crack propagation.

(c) The basic two-dimensional finite element program is presently being modified to study heat conduction problems. It could readily be applied to cracked structures in future studies.

(d) The problem of crack propagation has been investigated recently using the finite element method as a fatigue model. The local mesh must be altered to follow the progress of the crack, which in turn means that the stiffness matrix has to be modified to allow for the various geometric alterations and also the creation of further crack face nodes. The iteration processes involved would be extremely complicated and readers are referred to Igraffea's work[85,86].

8.2 CONCLUSIONS

Significant progress has been made towards establishing a general purpose fracture analysis program. Features which can now be studied include multi-crack tip, mixed-mode problems, of any complexity, involving partial crack closure. A separate program has been written which calculates the crack propagation angles and strain-energy-density factors from the computed stress intensity factors.

To complement the finite element analysis programs, an auto-mesh generation program has been written, which significantly reduces the amount of effort required in data preparation. The mesh generation program has proven to be versatile, reasonably easy to implement and generally applicable to the range of finite element programs available, including the general axisymmetric program.

The existing general two-dimensional finite element program has been modified to include the eight-node isoparametric quadrilateral element. Also, in order to accommodate problems involving a large number of elements, a flexible solving routine was developed, providing a very compact method of solving large sets of simultaneous equations stored in a segmented form.

The various examples illustrate the wide range of component shapes which can be examined and the accuracy of the solution. Generally values for the stress intensity factors are obtained within 5.0% of theoretical solutions, where they are available.

A great deal of experience has been obtained from the various problems encountered in applying the finite element technique, and one of the most important factors governing the solution accuracy is the size and proportion of the elements used in the discretized component. Some guidelines can be found in reference (5), which may help in this respect.

The main objectives have been achieved, but there are areas of work which still need clarifying as discussed above.

CHAPTER 9

APPENDICES

9.1.1 FIELD EQUATIONS FOR STRAINS, STRESSES AND DISPLACEMENTS NEAR A CRACK TIP

The derivation of the local crack tip field equations can be found in Section (6.2). The constants α_1 and α_2 are related to Irwin's definition of the stress intensity factors K_I and K_{II} by,

$$\alpha_1 + j\alpha_2 = \frac{1}{\sqrt{2}} (K_I - jK_{II}) \tag{9.1}$$

The field equations for both polar and cartesian coordinate systems are listed below. Refer to figures (6.1, 6.2). The field equations can be simplified if we assume

$$C_1 = (\frac{n}{2} + 1) \qquad C_6 = (\kappa - \frac{m}{2})$$

$$C_2 = (\frac{n}{2} - (-1)^n) \qquad C_7 = (\frac{m}{2} - (-1)^m)$$

$$C_3 = (\frac{n}{2} + (-1)^n) \qquad C_8 = (\frac{m}{2} + (-1)^m)$$

$$C_4 = (3 - \frac{n}{2}) \qquad C_9 = (\frac{m}{2} - 2 + \kappa)$$

$$C_5 = (\frac{n}{2} - 1) \qquad C_{10} = (\frac{m}{2} - 1)$$

$$C_{11} = (\frac{m}{2} + 1) \qquad C_{11} = (\kappa + \frac{m}{2}) \tag{9.2}$$

The expressions are given in terms of integers n and m, for later use in the derivations of the strain energy of the core element.

Polar coordinates

$$\sigma_{r\theta} = \sum_{n=1}^{\infty} \frac{n}{2} r^{(\frac{n}{2}-1)} \{ \alpha_{2n-1}[C_1\cos\theta(C_5) - C_3\cos\theta(C_1)] + \alpha_{2n}[-C_1\sin\theta(C_5) + C_2\sin\theta(C_1)]\} \quad (9.3)$$

$$\sigma_r = \sum_{n=1}^{\infty} \frac{n}{2} r^{(\frac{n}{2}-1)} \{\alpha_{2n-1}[C_4\cos\theta(C_5) + C_3\cos\theta(C_1)] - \alpha_{2n}[C_4\sin\theta(C_5) + C_2\sin\theta(C_1)]\} \quad (9.4)$$

$$\tau_{r\theta} = \sum_{n=1}^{\infty} \frac{n}{2} r^{(\frac{n}{2}-1)} \{\alpha_{2n-1}[C_5\sin\theta(C_5) - C_3\sin\theta(C_1)] + \alpha_{2n}[C_5\cos\theta(C_5) - C_2\cos\theta(C_1)]\} \quad (9.5)$$

$$\varepsilon_\theta = \sum_{m=1}^{\infty} \frac{m}{4\mu} r^{(\frac{m}{2}-1)} \{\alpha_{2m-1}[C_9\cos\theta(C_{10}) - C_8\cos\theta(C_{11})] + \alpha_{2m}[-C_9\sin\theta(C_{10}) + C_7\sin\theta(C_{11})]\} \quad (9.6)$$

$$\varepsilon_r = \sum_{m=1}^{\infty} \frac{m}{4\mu} r^{(\frac{m}{2}-1)} \{\alpha_{(2m-1)}[C_6\cos\theta(C_{10}) + C_8\cos\theta(C_{11})] + \alpha_{2m}[-C_6\sin\theta(C_{10}) - C_7\sin\theta(C_{11})]\} \quad (9.7)$$

$$\gamma_{r\theta} = \sum_{m=1}^{\infty} \frac{m}{2\mu} r^{(\frac{m}{2}-1)} \{\alpha_{2m-1}[C_{10}\sin\theta(C_{10}) - C_8\sin\theta(C_{11})] + \alpha_{2m}[C_{10}\cos\theta(C_{10}) - C_7\cos\theta(C_{11})]\} \quad (9.8)$$

$$u_r = \sum_{m=1}^{\infty} \frac{r}{2\mu} \frac{m}{2} \{\alpha_{2m-1}[C_6\cos\theta(C_{10}) - C_8\cos\theta(C_{11})] + \alpha_{2m}[-C_6\sin\theta(C_{10}) - C_7\sin\theta(C_{11})]\} \quad (9.9)$$

$$u_\theta = \sum_{m=1}^{\infty} \frac{r}{2\mu} \frac{m}{2} \{\alpha_{2m-1}[C_{12}\sin\theta(C_{10}) - C_8\sin\theta(C_{11})] + \alpha_{2m}[C_{12}\cos\theta(C_{10}) - C_8\cos\theta(C_{11})]\} \quad (9.10)$$

Cartesian coordinates

$$u_x = \sum_{n=1}^{\infty} \frac{r}{2\mu}^{\frac{n}{2}} \{\alpha_{2n-1}[(\kappa + \frac{n}{2} + (-1)^n)\cos\theta\frac{n}{2} - \frac{n}{2}\cos\theta(\frac{n}{2} - 2)]$$

$$+ \alpha_{2n}[-(\kappa + \frac{n}{2} - (-1)^n)\sin\theta\frac{n}{2} + \frac{n}{2}\sin\theta(\frac{n}{2} - 2)]\} \qquad (9.11)$$

$$u_y = \sum_{n=1}^{\infty} \frac{r}{2\mu}^{\frac{n}{2}} \{\alpha_{2n-1}[(\kappa - \frac{n}{2} - (-1)^n)\sin\theta\frac{n}{2} + \frac{n}{2}\sin\theta(\frac{n}{2} - 2)]$$

$$+ \alpha_{2n}[(\kappa - \frac{n}{2} + (-1)^n)\cos\theta\frac{n}{2} + \frac{n}{2}\cos\theta(\frac{n}{2} - 2)]\} \qquad (9.12)$$

where the symbols have their usual meaning. The stress and strain equations for the cartesian coordinate system are not required and hence were not derived, however reference (48) provides a listing.

9.1.2 STRAIN ENERGY OF THE CORE ELEMENT

The strain energy of the crack tip core element has been derived using the strain, displacement and stress expressions of the previous section. The algebraic manipulationsinvolved are given in Section (7.3.4) and the strain energy can be expressed as

$$U_c = \int_{vol} \frac{1}{2}(\sigma_r\varepsilon_r + \sigma_\theta\varepsilon_\theta + \tau_{r\theta}\gamma_{r\theta})\,dvol \qquad (9.13)$$

For the full circular core or the mixed mode case, figure (9.1), the integral equation becomes,

$$U_c = \frac{t}{2} \int_0^{2\pi}\int_0^{r} (\sigma_r\varepsilon_r + \sigma_\theta\varepsilon_\theta + \tau_{r\theta}\gamma_{r\theta})\,r\,dr\,d\theta \qquad (9.14)$$

Completing the necessary algebraic manipulations the strain energy for the core element can be expressed as a function of θ, the core radius r, and integers m and n.

Again in order to reduce the size of the strain energy expression, repetative terms are grouped as follows

$$T_1 = (\sin\theta(C_5 + C_{10}))/(C_5 + C_{10})$$

$$T_2 = (\sin\theta(C_5 - C_{10}))/(C_5 - C_{10})$$

$$T_3 = (\sin\theta(C_5 + C_{11}))/(C_5 + C_{11})$$

$$T_4 = (\sin\theta(C_5 - C_{11}))/(C_5 - C_{11})$$

$$T_5 = (\sin\theta(C_1 + C_{10}))/(C_1 + C_{10})$$

$$T_6 = (\sin\theta(C_1 - C_{10}))/(C_1 - C_{10})$$

$$T_7 = (\sin\theta(C_1 + C_{11}))/(C_1 + C_{11})$$

$$T_8 = (\sin\theta(C_1 - C_{11}))/(C_1 - C_{11}) \qquad (9.15)$$

Expressions T_9 - T_{16} are identical with those above but the sine is replaced by cosine, e.g.

$$T_9 = (\cos\theta(C_5 + C_{10}))/(C_5 + C_{10})$$

$$T_{10} = (\cos\theta(C_5 - C_{10}))/(C_5 - C_{10}) \quad \text{etc.} \qquad (9.16)$$

The function C_i are listed in the previous section. We can therefore write the strain energy expressions, as,

$$U_c = \frac{1}{2} \sum_{n=1}^{\infty} \sum_{m=1}^{\infty} \frac{t}{8\mu} \frac{mn}{(n+m)} r^{(n+m)/2}$$

$$\Big[\alpha_{2n-1}\alpha_{2m-1}\{C_4C_6(T_1 + T_2) + C_4C_8(T_3 + T_4)$$
$$+ C_3C_6(T_5 + T_6) + C_3C_8(T_7 + T_8)$$
$$+ C_1C_9(T_1 + T_2) - C_1C_8(T_3 + T_4)$$
$$- C_3C_9(T_5 + T_6) + C_3C_8(T_7 + T_8)$$
$$+ 2 \times [C_5C_{10}(T_2 - T_1) + C_5C_8(T_4 - T_3)$$
$$-C_3C_{10}(T_6 - T_5) + C_3C_8(T_8 - T_7)]\}$$

$$+ \alpha_{2n-1}\alpha_{2m}\{-C_4C_6(T_{10} - T_9) - C_4C_7(T_{12} - T_{11})$$
$$- C_3C_6(T_{14} - T_{13}) - C_3C_7(T_{16} - T_{15})$$
$$- C_1C_9(T_{10} - T_9) + C_1C_7(T_{12} - T_{11})$$
$$+ C_3C_9(T_{14} - T_{13}) - C_3C_7(T_{16} - T_{15})$$
$$+ 2 \times [-C_5C_{10}(T_9 + T_{10}) + C_5C_7(T_{11} + T_{12})$$
$$+ C_3C_{10}(T_{13} + T_{14}) - C_3C_7(T_{15} + T_{16})]\}$$

$$+ \alpha_{2n}\alpha_{2m-1}\{C_4C_6(T_9 + T_{10}) + C_4C_8(T_{11} + T_{12})$$
$$+ C_2C_6(T_{13} + T_{14}) + C_2C_8(T_{15} + T_{16})$$
$$+ C_1C_9(T_9 + T_{10}) - C_1C_8(T_{11} + T_{12})$$
$$- C_2C_9(T_{13} + T_{14}) + C_2C_8(T_{15} + T_{16})$$
$$+ 2 \times [C_5C_{10}(T_{10} - T_9) - C_5C_8(T_{12} - T_{11})$$
$$- C_2C_{10}(T_{14} - T_{13}) + C_2C_8(T_{16} - T_{15})]\}$$

$$+ \alpha_{2n}\alpha_{2m}\{C_4C_6(T_2 - T_1) + C_4C_7(T_4 - T_3)$$
$$+ C_2C_6(T_6 - T_5) + C_2C_7(T_8 - T_7)$$
$$+ C_1C_9(T_2 - T_1) - C_1C_7(T_4 - T_3)$$
$$- C_2C_9(T_6 - T_5) + C_2C_7(T_8 - T_7)$$
$$+ 2 \times [C_5C_{10}(T_1 + T_2) - C_5C_7(T_3 + T_4)$$
$$-C_2C_{10}(T_5 + T_6) + C_2C_7(T_7 + T_8)]\}\Big]_0^{2\pi} \qquad (9.17)$$

9.2 JOB CONTROL CARDS

A few selected job decks are given overleaf, to aid the user, and represent the basic requirements necessary to run programs at Aston and Manchester Computer Centres. Programs run on the Manchester system are controlled from job decks input at Aston. Data stored on file at Aston cannot be transferred to Manchester under the present system and only card information can be transferred directly. This situation necessitates that the data generation program operates at Aston and also at the Manchester Computer centre. Therefore, having generated the mesh data successfully using the Aston system, the same input data can be used again to run the Manchester version of the auto-mesh generation program, knowing that the data generated is correct. It is hoped that in the future information stored on file can be transferred directly to Manchester, saving this duplication in data generation. The sample job decks overleaf, provide the basic requirements to operate the programs, and information on the use of compiled programs, own program descriptions, standard library routines, etc., can be obtained from the advisory service. The University of Manchester Regional Computer Centre provides a Joint System Mini-Manual, which presents the essential information necessary to operate the system.

The first example shows the job control cards required to run the mesh generation program STDAMG, on the Aston ICL 1904S computer. The first card specifies the user and job title, and also the estimated job time and size. The second card indicates the program name and its requirements, in this case the standard ICL graph plotting routines are required, hence *GP ICL. Also as a record of the generated data is necessary, an output filename is provided, thus *LP1 TEST1. Where LP1

refers to the line printer 1 and acts as a transfer channel for the generated data, and TEST1 is the appointed filename. Note that LP1 is a standard channel built into the STDAMG program.

The second example, performs the same function as the previous example on the joint system at Manchester. The joint system refers to the 1904S/7600 computers, where the 1904S manages the input and output from the larger CDC 7600 computer. Jobs run on the 7600 are normally input on the 1900, passed across the hardware coupler, and run on the 7600, and the output is then passed back to the 1900 for listing or storing in the filestore. Returning to example two, the first card identifies the user and specifies the job size and priority. In this case the job will be run at priority 4000 for, at least 10 decimal seconds. The ATTACH control statement copies the 1900 file PCSTDAMG into a local file ACE on the 7600. The 7600 filename ACE is optional. The RFL cards request a field length (i.e. an amount of core store) for, respectively the compilation and execution of the program. The ALGOL card calls the ALGOL compiler, and the compiler options are selected by the various parameters. The most important are,

I = filename Source program on named file.

S = 2 All arrays in large Core Memory.

The LGO card loads and executes the program using the data which follows the second end-of-section card (####S). The bracketed (C) warns the computer that channel cards are being used. As in example (1) an output channel is specified, which is built into the auto-mesh generation program, namely channel (11) and in this case the file DATAFIL receives the generated data. After the program has been executed the file DATAFIL is re-wound and transferred to the 1900 file-store using the CATALOG command. Here the generated data is stored on file PCTEST1.

Having generated the data file PCTEST1 this can be run against the appropriate finite element program. In example three a job deck is shown, which uses the general two-dimensional finite element program PCPLSS and runs it against the data stored in file PCTEST1. The first card identifies the job and specifies the priority, etc., as in the previous example. Again the ATTACH statement is used in order to copy the 1900 files PCPLSS and PCTEST1 into the local filenames PROG and DATAFIL, respectively, on the 7600. The remaining commands are similar to that of example two, except that the results are output using the standard channels to the line printer. Note, however, that channel (11) must correspond with the data file DATAFIL, as this channel number is used to transfer the data to the finite element program and this channel number has been used in all of the finite element programs stored on the Manchester system. The RETURN command returns the uncompiled program to the 1900 store and saves space on the 7600.

The fourth example illustrates the job deck which would be used with the program PCPLSEG. It is identical to that of the previous example, but as the segmented solving routine is used the appropriate channels must be presented. The largest problem which can be run using the standard solving routine, would be of approximately 1 K d.o.f. or have an equivalent matrix length of 80 K words, above this limit computer storage space becomes a problem and hence the need for a segmented solving routine. Each segment can hold 35 K words and in this example three channels have been selected, which means that a stiffness matrix of length 105 K words could be handled. Any number of segments can be chosen depending on the size of the problem. Refer to Chapter 4 for further information on the segmented solving routine.

Example 1. Job deck for running the Auto-Mesh Generation program STDAMG, on the Aston I.C.L.1904S, storing the generated data in filename TEST1.

```
JOB :(username),(jobname),JD(JT60,MZ40K)
UAALGOL PROG STDAMG,*GP ICL,*LP1 TEST1
```

} Mesh Generation Data

```
****
```

EXAMPLE 2. This set of control cards runs the Mesh Generation program PCSTDAMG, stored at Manchester and is equivalent to example 1. No graphical output is generated in this case.

```
JOB (jobname),:(username),CP76(P4000,TD10,SP)
ATTACH(ACE,PCSTDAMG,ST=S4S)
RFL(45000)
ALGOL(S=2,I=ACE,L,R=0,O=0,C=0,P=0,F=0)
RFL(50000,L=40)
LGO(C)
REWIND(DATAFIL)
CATALOG(DATAFIL,PCTEST1,ST=S4S)
####S
CHANNEL,31=DATAFIL
CHANNEL,11=31
####S
```

} Mesh Generation Data

```
****
```

EXAMPLE 3. Job control card sequence for running the general finite element program PCPLSS at Manchester. For continuity the program is run using the generated data of the previous example, PCTEST1.

```
JOB (jobname),:(username),CP76(P3000,TD40,SP)
ATTACH(DATAFIL,PCTEST1,ST=S4S)
ATTACH(PROG,PCPLSS,ST=S4S)
RFL(45000)
ALGOL(S=2,I=PROG,L,R=0,O=1,C=0,P=0,F=0)
RETURN(PROG)
RFL(60000,L=240)
LGO(C)
####S
CHANNEL,11=DATAFIL
****
```

EXAMPLE 4. Job control cards for running the basic finite element program PCPLSEG, which contains the segmented solving routine. Again the data is stored on file PCTEST1.

```
Job (jobname),:(username),CP76(P3000,TD50,SP)
ATTACH(DATAFIL,PCTEST1,ST=S4S)
ATTACH(PROG,PCPLSEG,ST=S4S)
RFL(45000)
ALGOL(S=2,I=PROG,L,R=0,O=1,C=0,P=0,F=0)
RETURN(PROG)
RFL(60000,L=240)
LGO(C)
####S
CHANNEL,11=DATAFIL
CHANNEL,1=SEG1,B
CHANNEL,2=SEG2,B
CHANNEL,3=SEG3,B
CHANNEL,21=1
CHANNEL,22=2
CHANNEL,23=3
****
```

EXAMPLE 5. JOB control card sequence for operating the multi-tip, crack closure programs PCPOY and PCPOXY, again the data is stored on file PCTEST1.

```
JOB (jobname),:(username),CP76(P3000,TD50,SP)
REQUEST(ARRAYQ,T)
REQUEST(DATAFIL,T)
REQUEST(LGO,T)
ATTACH(DATAFIL,PCTEST1,ST=S4S)
ATTACH(PROG,PCPOY,ST=S4S)
RFL(45000)
ALGOL(S=2,I=PROG,L,R=0,O=1,C=0,P=0,F=0)
RETURN(PROG)
RFL(60000,L=240)
LGO(C)
####S
CHANNEL,11=DATAFIL
CHANNEL,10=ARRAYQ,B
CHANNEL,12=ARRAYK,B
CHANNEL,30=10
CHANNEL,32=12
****
```

The final example shows how to implement the multi-tip fracture program PCPOY, which has a facility to investigate partial crack closure problems. Again the basic commands are as in the previous examples, however to reduce the amount of core taken in the transfer operations, i.e. 5 K_8 words, a REQUEST statement is used and this reduces the allocated core space to 1 K_8 words. In effect this means that the data transfer process will be that much slower, as the information can only be transferred in 1 K_8 blocks. Also, because the data transfer channels which are suppressed in this manner, are only carried out once in the program cycle, the use of the REQUEST command can be seen as a core space saving of 8 K_8 words. The system equations are repeatedly modified and solved in the partial crack closure iterative scheme. As the solving routine overwrites the force vector and stiffness matrix, a current copy must be saved on the backing store, hence the need for the two extra channels - ARRAYQ and ARRAYK. Further information on the use of channel cards can be found in the Joint System Mini-Manual.

9.3 PROGRAM AND PROCEDURE LISTINGS

9.3.1 PROGRAM INDEX

The following list gives a brief description of the various programs modified or generated in the course of this work.

Notes: 1. Programs prefixed with 'PC' are designed to operate on the Manchester system.

2. All programs are written in ALGOL 60, unless stated otherwise.

3. Where applicable a program 'code' is given and this corresponds to the mesh generation CODE number.

4. All 2-D finite element programs have the facility of two element types, namely the 6-node triangle and the 8-node quadrilateral isoparametric element.

PROGRAM TITLE — DESCRIPTION

STDAMG - This is an automatic mesh generation program, it generates the x and y nodal coordinates, element nodal connections and the ancillary control variables. It services most of the finite element programs. Generated data is stored on a specified filename and a plot of the resulting mesh configuration is given. See Chapter five.

PCSTDAMG - This is the Manchester version of STDAMG and it does not provide graphical information.

QTPLSS - This is a 2-D finite element program, and apart from the nodal displacements this program outputs the stresses and strains at the nodal points and/or at the element centroids. Refer to Chapter four. CODE = 1.

PCQTPLSS - This is the Manchester version of program QTPLSS. CODE = 1.

QPLSSR - This program is identical to QTPLSS, but has an extra routine which outputs the residual forces after a successful run. CODE = 1.

PCPLSEG - This is a Manchester program, based on the QTPLSS program. It differs in that it has a segmented solving routine capable of encompassing very large problems, i.e. above 1 K d.o.f. or [K] lengths > 100 K words. Each segment can handle 35 K words. See Chapter four. CODE = 1.

IAAXMG - This is an axisymmetric finite element program, employing the 6-node triangular element only. It outputs the nodal displacements and the stresses and strains at the element centroids and/or nodes. See reference (29) and Chapter four. CODE = 4.

PCQTM1ST - This is a 2-D finite element program, used to investigate mode I, single crack tip problems. It has no crack closure facility and the elements must all have the same material property. CODE = 2.

PCPOY and

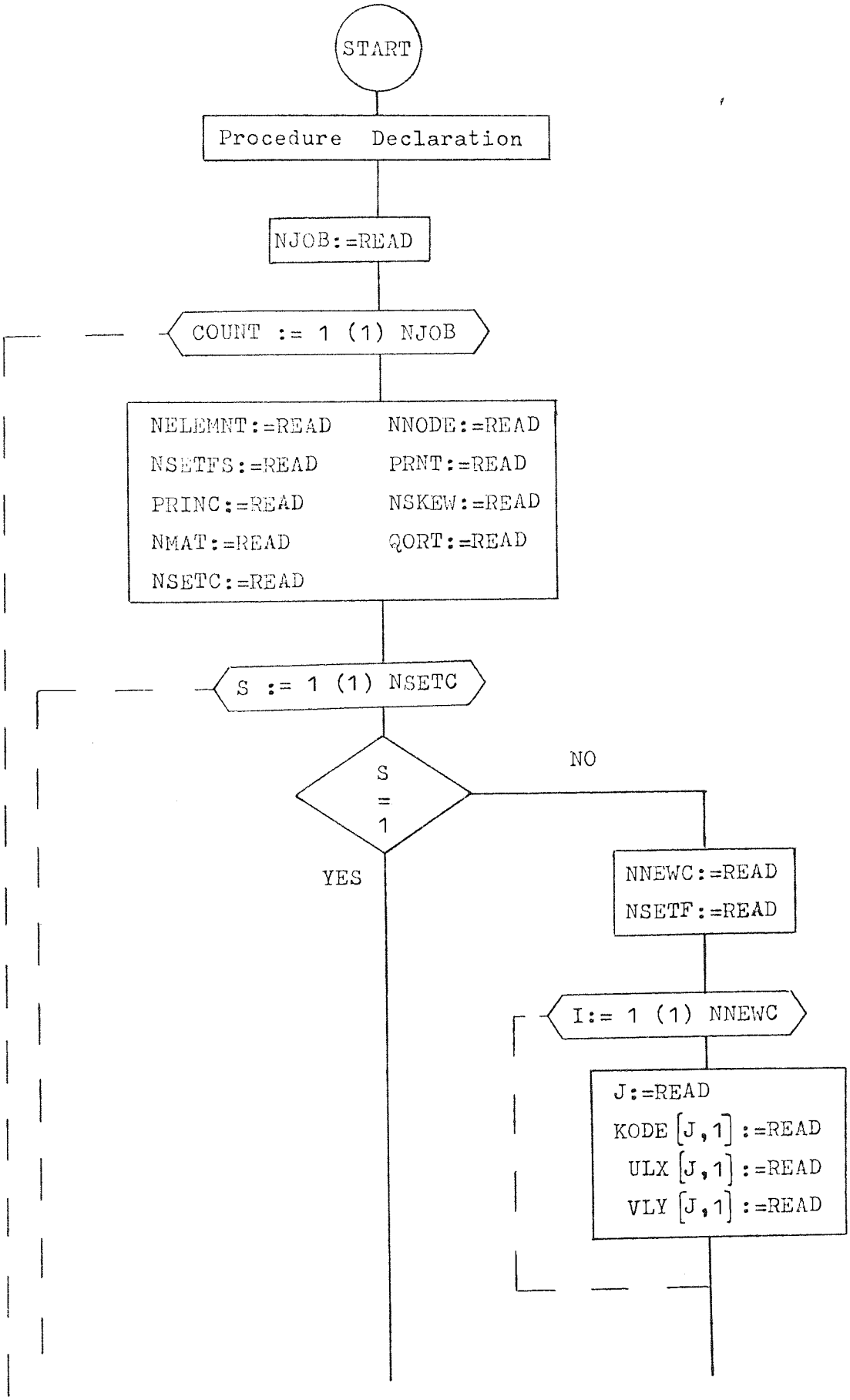
PCPOXY - These are 2-D finite element programs, which can accommodate mixed-mode, multi-crack tip problems, with partial crack closure. PCPOY and PCPOXY contain closure routines which give a no-friction and no-slip finite element model respectively. Elements must have the same material properties. See Chapter seven. CODE = 3.

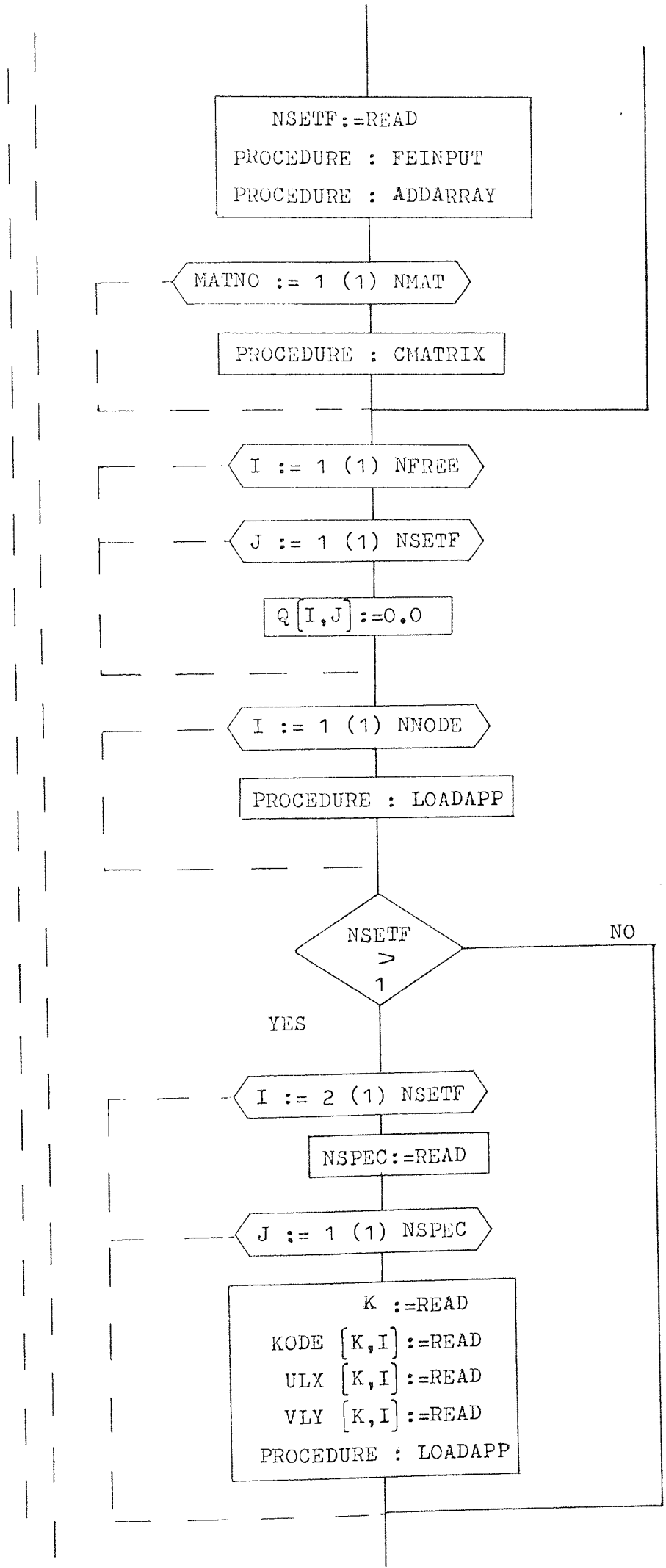
PROPAG - This program calculates the strain-energy density factor and angle of crack propagation, using the computed stress intensity factors. The input data sequence is given at the head of the program. Refer to Chapter seven.

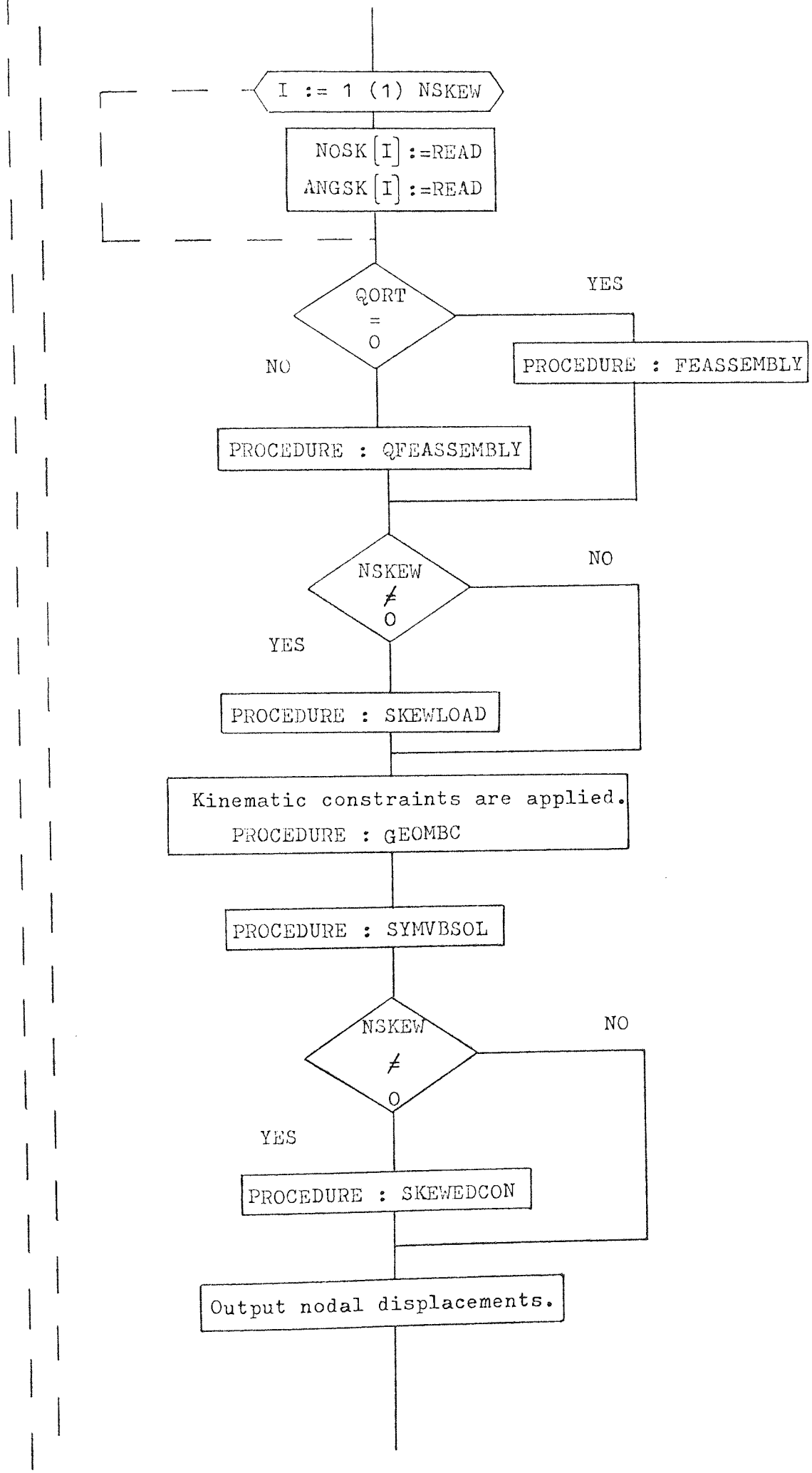
9.3.2 PROGRAM LISTINGS

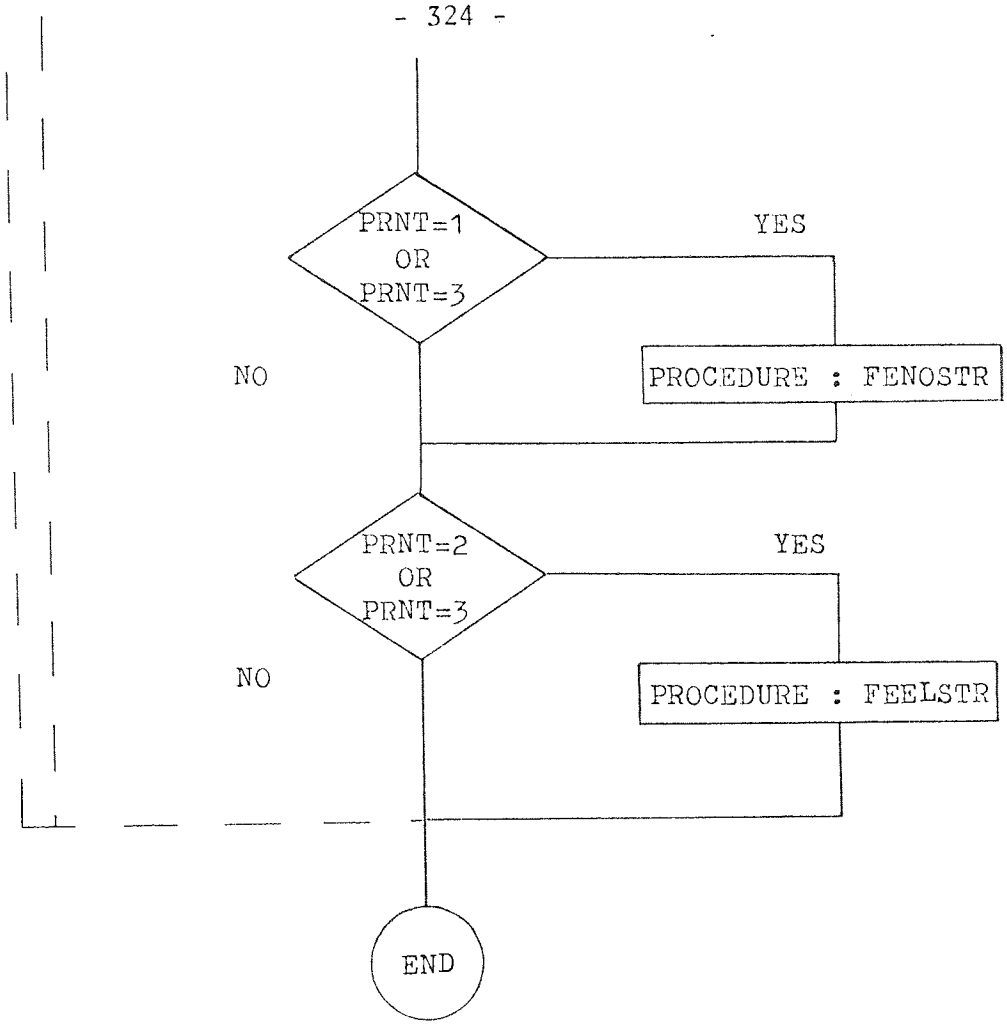
i. The general 2-D F.E. program QTPLSS.

Program flowchart.




NSETF:=READ
PROCEDURE : FEINPUT
PROCEDURE : ADDARRAY
MATNO := 1 (1) NMAT
PROCEDURE : CMATRIX
I := 1 (1) NFREE
J := 1 (1) NSETF
Q[I,J]:=0.0
I := 1 (1) NNODE
PROCEDURE : LOADAPP
NSETF > 1
NO
YES
I := 2 (1) NSETF
NSPEC:=READ
J := 1 (1) NSPEC
K :=READ
KODE [K,I] :=READ
ULX [K,I] :=READ
VLY [K,I] :=READ
PROCEDURE : LOADAPP


I := 1 (1) NSKEW
NOSK[I] :=READ
ANGSK[I] :=READ
QORT = 0
YES
NO
PROCEDURE : FEASSEMBLY
PROCEDURE : QFEASSEMBLY
NSKEW ≠ 0
NO
YES
PROCEDURE : SKEWLOAD
Kinematic constraints are applied.
PROCEDURE : GEOMBC
PROCEDURE : SYMVBSOL
NSKEW ≠ 0
NO
YES
PROCEDURE : SKEWEDCON
Output nodal displacements.


PRNT=1
OR
PRNT=3
YES
NO
PROCEDURE : FENOSTR
PRNT=2
OR
PRNT=3
YES
NO
PROCEDURE : FEELSTR
END

```
DOCUMENT        'OTPISS

'BEGIN' 'COMMENT'  THIS PROGRAM EMPLOYS THE ISOPARAMETRIC
                   FORMULATION FOR 2-D PROBLEMS OF PLANE STRESS/STR
                   TWO ELEMENTS CAN BE SELECTED, VIA
                  THE QORT PARAMETER - EITHER A SIX NODE TRIANGLE
                  OR AN EIGHT-NODE QUADRILATERAL;
'COMMENT' CHANGES IN INPUT:-
              NODAL CONNECTIONS,MATERIAL NO.
              EL. PROPS AND EL. THICKNESS;
'INTEGER' NELEMT,NNODE,NSETFS,NFREE,I,J,K,V,U,W,CASE,NSETC,NEWC,S,
NJOB,COUNT,PRNT,NMAT,MATNO,NSKEW,PRINC,QORT;
'REAL' DELTA,DETJ,ANG;

'PROCEDURE' DECLARATIONS

NJOB:=READ;
'FOR' COUNT:=1 'STEP' 1 'UNTIL' NJOB 'DO'
'BEGIN'
NEWLINE(2);
WRITETEXT('('JOB%NAME%------')');
COPYTEXT('('END%OF%TITLE')');
NELEMT:=READ;
NNODE:=READ;
NEWLINE(2);
WRITETEXT('('NO%OF%ELEMENTS%----')');
PRINT(NELEMT,3,0);
NEWLINE(2);
WRITETEXT('('NO%OF%NODES%-------')');
PRINT(NNODE,3,0);
NFREE:=NNODE*2;
NSETFS:=READ;
PRNT:=READ;PRINC:=READ; NSKEW:=READ;
NMAT:=READ; QORT:=READ;
NEWLINE(2);
WRITETEXT('('ELEMENT%SELECTED%------%')');
'IF'QORT=0'THEN'WRITETEXT('('TRIANGULAR')')
          'ELSE'WRITETEXT('('QUADRILATERAL')');
'BEGIN'
'INTEGER' BAND,NSPEC,Z,COMPA,NSETF;
'REAL' 'ARRAY' XX,YY[1:NNODE],ULX,VLY[1:NNODE,1:NSETFS],
     Q[1:NFREE,1:NSETFS],C[1:NMAT,1:6],A[1:5],TH[1:NMAT],
ANGSK[1:NSKEW+1];
'INTEGER''ARRAY' NODE[1:NELEMT,1:7+2*QORT],KODE[1:NNODE,1:NSETFS],
NOSK[1:NSKEW+1],ADD[0:NFREE];
NSETC:=READ;
'FOR' S:=1 'STEP' 1 'UNTIL' NSETC 'DO'
'BEGIN'
'IF' S=1 'THEN' 'BEGIN'
NSETF:=READ;
FEINPUT(C,ADD,XX,YY,NFREE,NNODE,NSETF,KODE,NSPEC,NSETFS,
        ULX,VLY,NELEMT,NODE,NMAT);
```

```
ADDARRAY(NELEMT,NNODE,ADD,NODE);
'FOR' MATNO:=1 'STEP' 1 'UNTIL' NMAT 'DO'
CMATRIX(C,CASE,A,MATNO,TH,ANG);
'END' 'ELSE'
'BEGIN'  NNEUC:=READ;
      NSETF:=READ;
 NEWLINE(10);
WRITETEXT('('CONSTRAINT%SET---')');
PRINT(S,2,0);
'FOR' I:=1 'STEP' 1 'UNTIL' NNEUC 'DO'
'BEGIN' J:=READ;
      KODE[J,1]:=READ;  NLX[J,1]:=READ;  VLY[J,1]:=READ;
'END';
NEWLINE(2);WRITETEXT('('GODAL%POINT%DATA')');
NEWLINE(2);SPACE(4);WRITETEXT('('NODE')');SPACE(5);
WRITETEXT('('X%COORD')'); SPACE(5);WRITETEXT('('Y%COORD')');
SPACE(5);WRITETEXT('('TYPE')');SPACE(5);
WRITETEXT('('X-DISP')');SPACE(5);WRITETEXT('('Y-DISP')');
NEWLINE(1);SPACE(46);WRITETEXT('('OR%LOAD')');
SPACE(4); WRITETEXT('('OR%LOAD')');
'FOR' I:=1 'STEP' 1 'UNTIL' NNODE 'DO'
'BEGIN'  NEWLINE(1); SPACE(3);
         PRINT(I,3,0);  SPACE(3);
         PRINT(XX[I],0,3);
         PRINT(YY[I],0,3);  SPACE(2);
         PRINT(KODE[I,1],3,0);  SPACE(2);
         PRINT(ULX[I,1],0,3);
         PRINT(VLY[I,1],0,3);
'END';
'END';
'FOR' I:=1 'STEP' 1 'UNTIL' NFREE 'DO'
'FOR' J:=1 'STEP' 1 'UNTIL' NSETF 'DO' Q[I,J]:=0.0;
'FOR' I:=1 'STEP' 1 'UNTIL' NNODE 'DO'
      LOADAPP(KODE[I,1],ULX[I,1],VLY[I,1],0,I,1);
'IF' NSETF 'GT' 1 'THEN'
'BEGIN'
'FOR' I:=2 'STEP' 1 'UNTIL' NSETF 'DO'
'BEGIN'
NSPEC:=READ;
NEWLINE(2);
WRITETEXT('('FORCE%SET-----')');
PRINT(I,3,0);
NEWLINE(2);WRITETEXT('('NODE')');
SPACE(5);WRITETEXT('('TYPE')');SPACE(3);
WRITETEXT('('X-DISP')');SPACE(7);WRITETEXT('('Y-DISP')');
NEWLINE(1);SPACE(21);WRITETEXT('('OR%LOAD')');
SPACE(6);WRITETEXT('('OR%LOAD')');
'FOR' J:=1 'STEP' 1 'UNTIL' NSPEC 'DO'
'BEGIN'
   K:=READ;  KODE[K,I]:=READ;  ULX[K,I]:=READ;  VLY[K,I]:=READ;
LOADAPP(KODE[K,I],ULX[K,I],VLY[K,I],0,K,I);
NEWLINE(2);
PRINT(K,3,0);  SPACE(2);
PRINT(KODE[K,I],3,0);  SPACE(2);
PRINT(ULX[K,I],0,4);
```

```
PRINT(VLY[K,I],0,4);
'END';
'END';
'END';
'FOR' I:=1 'STEP' 1 'UNTIL' NSKEW 'DO'
'BEGIN' NOSK[I]:=READ;
        ANGSK[I]:=READ;
'END';
'BEGIN' 'REAL' 'ARRAY' KC1:ADD[NFREE];
'IF'QORT=0'THEN'
FEASSEMBLY(NELEMT,K,XX,YY,DETJ,NODE,C,TH,ADD,NFREE,AUX,SKEWEDCON,
           NSKEW,NSETF,ANGSK,NOSK)
'ELSE'
QFEASSEMBLY(NELEMT,K,XX,YY,DETJ,NODE,C,TH,ADD,NFREE,QAUX,
           SKEWEDCON,NSKEW,NSETF,ANGSK,NOSK);
'IF'NSKEW'NE'0'THEN' SKEWLOAD(NSETF,Q,NOSK,ANGSK,NSKEW,1);
'COMMENT' INTRODUCTION OF KINEMATIC CONSTRAINTS;
'FOR' I:=1 'STEP' 1 'UNTIL' NNODE 'DO'
'BEGIN'
'IF' KODE[I,1]=0 'THEN' 'GOTO' KC1;
'IF' KODE[I,1]=2 'THEN' 'GOTO'KC2;
GEOMBC(VLX[I,1],2*I-1,Q,K,NFREE,1,ADD);
'FOR' J:=2 'STEP' 1 'UNTIL' NSETF 'DO' Q[I,J]:=Q[I,1];
'IF' KODE[I,1]=1 'THEN' 'GOTO' KC1;
KC2; GEOMBC(VLY[I,1],2*I,Q,K,NFREE,1,ADD);
'FOR' J:=2 'STEP' 1 'UNTIL' NSETF 'DO' Q[I,J]:=Q[I,1];
KC1;  'END';
NEWLINE(4);
WRITETEXT('('SYMMRSOL%%BEGINS')');
SYMMRSOL(K,K,ADD,0,NFREE,NSETF,FAIL);
'IF' NSKEW 'NE' 0 'THEN' SKEWLOAD(NSETF,Q,NOSK,ANGSK,NSKEW,-1);
'FOR' I:=1 'STEP' 1 'UNTIL' NSETF 'DO''BEGIN'
NEWLINE(4);
 WRITETEXT('('NODAL%DISPLACEMENTS%FORM FORCES%SET')');
        PRINT(I,2,0);
NEWLINE(3);SPACE(1);WRITETEXT('('NODE')');
SPACE(5);WRITETEXT('('X-DIRECTION')');SPACE(8);
WRITETEXT('('Y-DIRECTION')');SPACE(10);
WRITETEXT('('NODE')');SPACE(5);WRITETEXT('('X-DIRECTION')');
SPACE(8);WRITETEXT('('Y-DIRECTION')');
      W:=2*(NNODE'/'2);
'FOR' J:=2 'STEP' 2 'UNTIL' W 'DO'
  'BEGIN' NEWLINE(2);  V:=2*(J-1);
PRINT((J=1),3,0); SPACE(2); PRINT(Q[V-1,I],0,8); SPACE(2);
PRINT(Q[V,I],0,8); SPACE(13); PRINT(J,3,0); SPACE(2);
PRINT(Q[V+1,I],0,8); SPACE(2); PRINT(Q[V+2,I],0,8);
'END';
'IF' NNODE 'GT' W 'THEN'
'BEGIN' NEWLINE(2); PRINT(NNODE,3,0); SPACE(2);
        V:=2*NNODE;
        PRINT(Q[V-1,I],0,8); SPACE(2); PRINT(Q[V,I],0,8);
'END';
'END';
'IF' PRNT=1 'OR' PRNT=3'THEN'
FENDSTR(NSETF,NNODE,NELEMT,NODE,XX,YY,DETJ,Q,TH,C,AUX,QAUX);
```

```
'IF' PRNT=2 'OR' PRNT=3 'THEN'
FEELSTR(NSETF,Q,NELEMT,NODE,XX,YY,DETJ,TH,C,AUX,QAUX);
'END';
'END' OF CONSTRAINT LOOP;
'END';
'GOTO' FINE;
FAIL:NEWLINE(4);
WRITETEXT('('PROGRAM%FAILED%IN%PROCEDURE%SYMMVSOL')');
'GOTO'EXIT;
FINE:'END'; NEWLINE(4);
WRITETEXT('('END%OF%PROGRAM=SUBSERIAL"EXIT')');
EXIT:'END' OF PROGRAM;
****
```

(ii) The general 2-D finite element program containing the segmented solving routine. PCPLSEG

The program flowchart is identical to that of program QTPLSS, but procedures ADDARRAY and SYMVBSOL are replaced by ADDSEQ and SEGSOL respectively. A formal program listing is given overleaf in order to show the use of sub-procedure TEST and all procedures which are different to those of program QTPLSS, are shown in a later section.

```
DOCUMENT     PCPLSEG

'BEGIN' 'COMMENT'  THIS PROGRAM EMPLOYS THE ISOPARAMETRIC
                   FORMULATION FOR 2-D PROBLEMS OF PLANE STRESS/STR
                   THE ELEMENT USED IS A SIX-NODE TRIANGLE;
'COMMENT' CHANGES IN INPUT:-
             NODAL CONNECTIONS,MATERIAL NO.
             EL. PROPS AND EL. THICKNESS;
'INTEGER' NELEMT,NNODE,NSETFS,NFREE,I,J,K,V,U,O,CASE,NSETC,NNEWC,S,
NJOB,COUNT,PRNT,NMAT,MATNO,NSKEW,PRINC,CK1,CK2,CK3,CK4,CHTA,CHTP,
SIZE,CK,SEG,QORT;
'REAL' DELTA,DETJ,ANG;
'INTEGER''ARRAY'LIM,COL,ROW[0:15];

'PROCEDURE' DECLARATIONS

SELECT INPUT(11);
NJOB:=READ;
'FOR' COUNT:=1 'STEP' 1 'UNTIL' NJOB 'DO'
'BEGIN'
NEWLINE(2);
WRITETEXT('('JOB%NAME%------')');
COPYTEXT('('END%OF%TITLE')');
NELEMT:=READ;
NNODE:=READ;
NEWLINE(2);
WRITETEXT('('NO%OF%ELEMENTS%----')');
PRINT(NELEMT,3,0);
NEWLINE(2);
WRITETEXT('('NO%OF%NODES%-------')');
PRINT(NNODE,3,0);
NFREE:=NNODE*2;
NSETFS:=READ;
PRNT:=READ;PRINC:=READ; NSKEW:=READ;
NMAT:=READ; QORT:=READ;
NEWLINE(2);
WRITETEXT('('ELEMENT%SELECTED%------%')');
'IF'QORT=0'THEN'WRITETEXT('('TRIANGULAR')')
          'ELSE'WRITETEXT('('QUADRILATERAL')');
'BEGIN'
'INTEGER' BAND,NSPEC,Z,COMPA,NSETF;
'REAL' 'ARRAY' XX,YY[1:NNODE],ULX,VLY[1:NNODE,1:NSETFS],
     Q[1:NFREE,1:NSETFS],C[1:NMAT,1:6],A[1:5],TH[1:NMAT],
ANGSK[1:NSKEW+1];
'INTEGER''ARRAY' NODE[1:NELEMT,1:7+2*QORT],KODE[1:NNODE,1:NSETFS],
NOSK[1:NSKEW+1],ADD[0:NFREE];
NSETC:=READ;
'FOR' S:=1 'STEP' 1 'UNTIL' NSETC 'DO'
'BEGIN'
'IF' S=1 'THEN' 'BEGIN'
NSETF:=READ;
FEINPUT(C,ADD,XX,YY,NFREE,NNODE,NSETF,KODE,NSPEC,NSETFS,
```

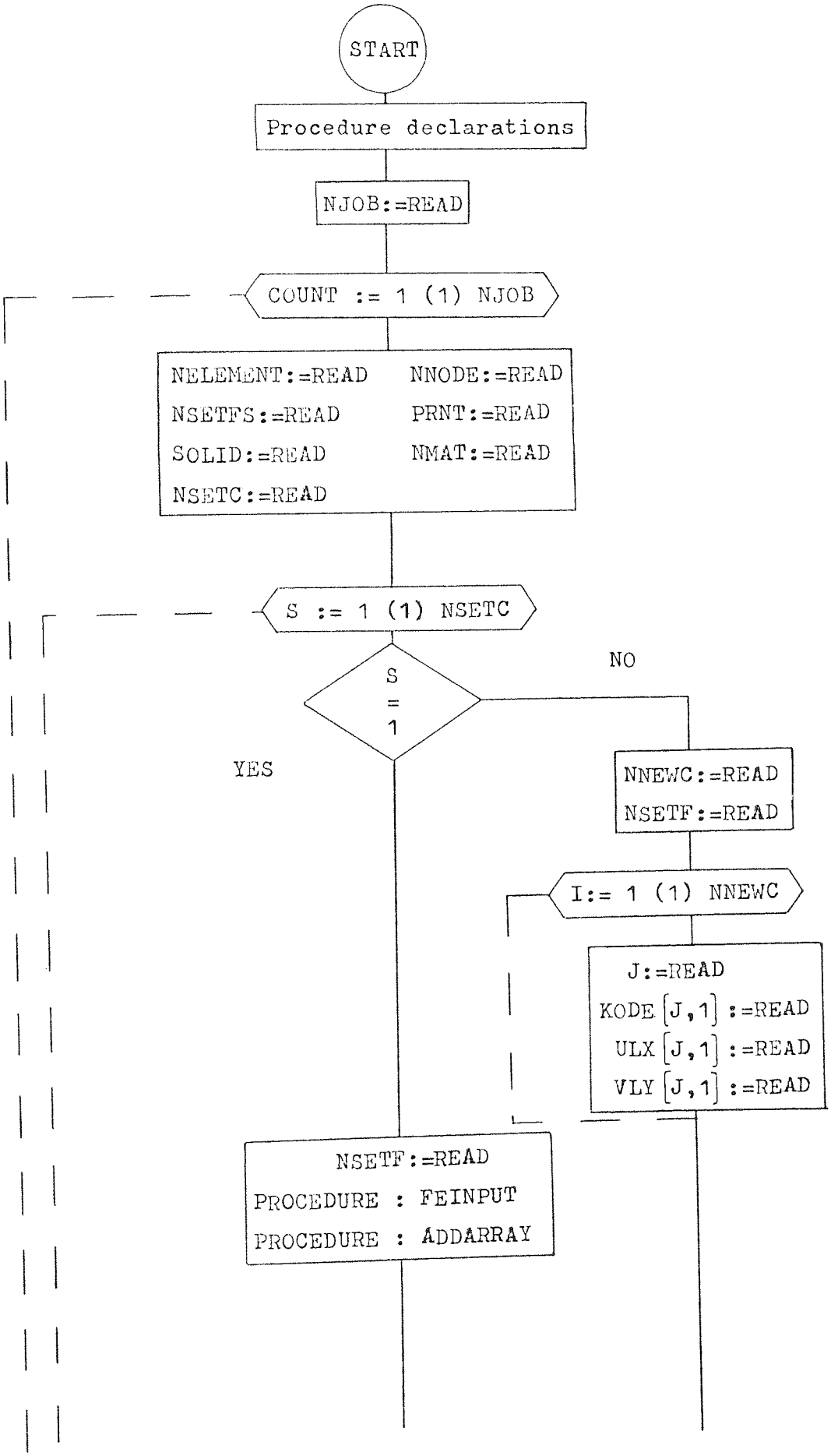
```
        ULX,VLY,NELEMT,NODE,NMAT);
ADDSER(NELEMT,NNODE,ADD,NODE);
'FOR' MATNO:=1 'STEP' 1 'UNTIL' NMAT 'DO'
CMATRIX(C,CASE,A,MATNO,TH,AUG);
'END' 'ELSE'
'BEGIN'  NNEWC:=READ;
      NSETF:=READ;
 NEWLINE(10);
WRITETEXT('('CONSTRAINT%SET---')');
PRINT(S,2,0);
'FOR' I:=1 'STEP' 1 'UNTIL' NNEWC 'DO'
'BEGIN' J:=READ;
      KODE[J,1]:=READ;  ULX[J,1]:=READ;  VLY[J,1]:=READ;
'END';
NEWLINE(2);WRITETEXT('('NODAL%POINT%DATA')');
NEWLINE(2);SPACE(4);WRITETEXT('('NODE')');SPACE(5);
WRITETEXT('('X%COORD')'); SPACE(5);WRITETEXT('('Y%COORD')');
SPACE(5);WRITETEXT('('TYPE')');SPACE(5);
WRITETEXT('('X-DISP')');SPACE(5);WRITETEXT('('Y-DISP')');
NEWLINE(1);SPACE(46);WRITETEXT('('OR%LOAD')');
SPACE(4); WRITETEXT('('OR%LOAD')');
'FOR' I:=1 'STEP' 1 'UNTIL' NNODE 'DO'
'BEGIN'  NEWLINE(1); SPACE(3);
         PRINT(I,3,0);  SPACE(3);
         PRINT(XX[I],0,3);
         PRINT(YY[I],0,3);  SPACE(2);
         PRINT(KODE[I,1],3,0);  SPACE(2);
         PRINT(ULX[I,1],0,3);
         PRINT(VLY[I,1],0,3);
'END';
'END';
'FOR' I:=1 'STEP' 1 'UNTIL' NFREE 'DO'
'FOR' J:=1 'STEP' 1 'UNTIL' NSETF 'DO' Q[I,J]:=0.0;
'FOR' I:=1 'STEP' 1 'UNTIL' NNODE 'DO'
     LOADAPP(KODE[I,1],ULX[I,1],VLY[I,1],Q,I,1);
'IF' NSETF 'GT' 1 'THEN'
'BEGIN'
'FOR' I:=2 'STEP' 1 'UNTIL' NSETF 'DO'
'BEGIN'
NSPEC:=READ;
NEWLINE(2);
WRITETEXT('('FORCE%SET-----')');
PRINT(I,3,0);
NEWLINE(2);WRITETEXT('('NODE')');
SPACE(5);WRITETEXT('('TYPE')');SPACE(3);
WRITETEXT('('X-DISP')');SPACE(7);WRITETEXT('('Y-DISP')');
NEWLINE(1);SPACE(21);WRITETEXT('('OR%LOAD')');
SPACE(6);WRITETEXT('('OR%LOAD')');
'FOR' J:=1 'STEP' 1 'UNTIL' NSPEC 'DO'
'BEGIN'
   K:=READ;  KODE[K,I]:=READ;  ULX[K,I]:=READ;  VLY[K,I]:=READ;
LOADAPP(KODE[K,I],ULX[K,I],VLY[K,I],Q,K,I);
NEWLINE(2);
PRINT(K,3,0);  SPACE(2);
PRINT(KODE[K,I],3,0);  SPACE(2);
```

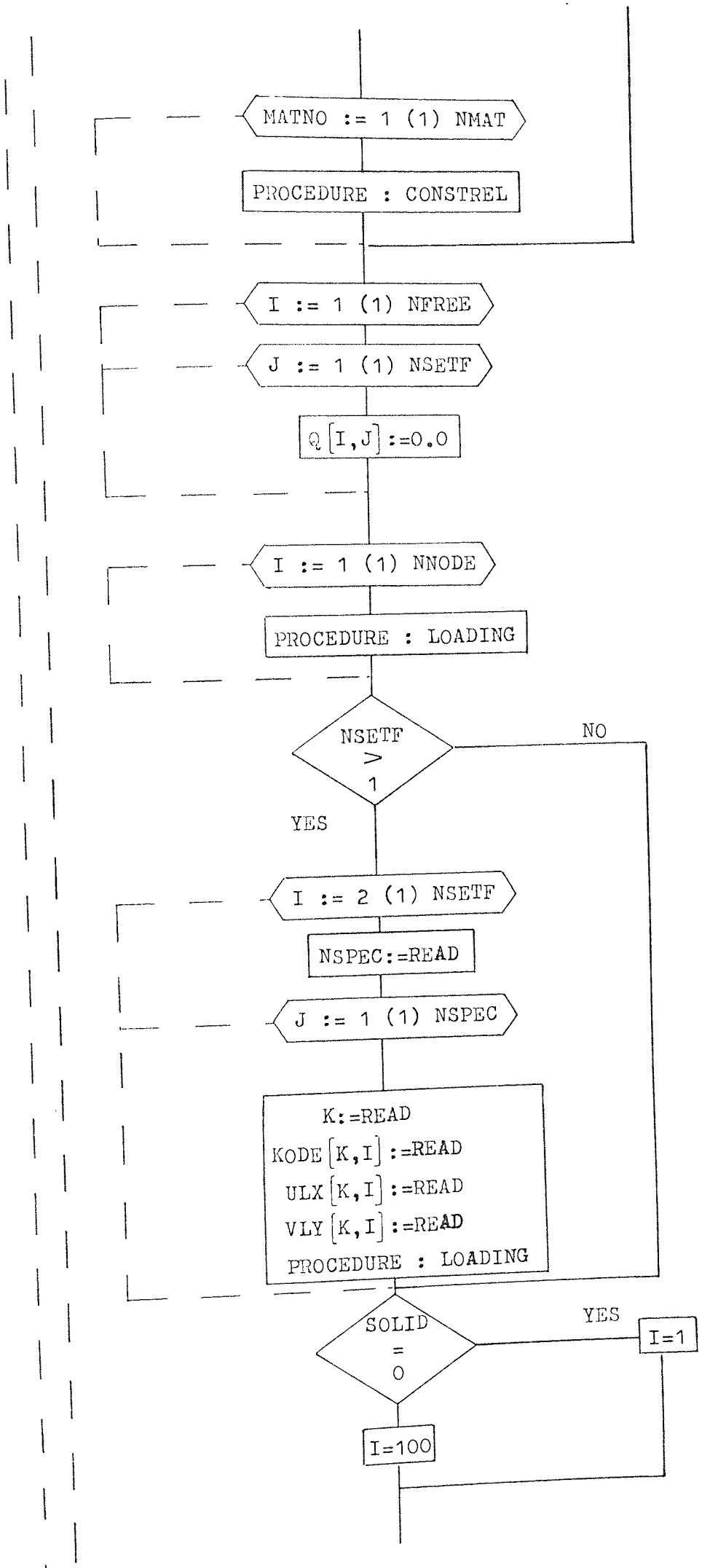
```
PRINT(ULX[K,I],0,4);
PRINT(VLY[K,I],0,4);
'END';
'END';
'END';
'FOR' I:=1 'STEP' 1 'UNTIL' NSKEW 'DO'
'BEGIN' NOSK[I]:=READ;
        ANGSK[I]:=READ;
'END';
'BEGIN' 'REAL' 'ARRAY' K[1:SIZE];
'IF' QOKT=0 'THEN'
FEASSEMBLY(NELEMT,K,XX,YY,DETJ,NODE,C,TH,ADD,NFREE,AUX,SKEWEDCON,
           NSKEW,NSETF,ANGSK,NOSK,TEST);
'ELSE'
QFEASSEMBLY(NELEMT,K,XX,YY,DETJ,NODE,C,TH,ADD,NFREE,QAUX,
           SKEWEDCON,NSKEW,NSETF,ANGSK,NOSK,TEST);
'IF'NSKEW'NE'0'THEN' SKEWLOAD(NSETF,Q,NOSK,ANGSK,NSKEW,1);
'COMMENT' INTRODUCTION OF KINEMATIC CONSTRAINTS;
'FOR' I:=1 'STEP' 1 'UNTIL' NNODE 'DO'
'BEGIN'
'IF' KODE[I,1]=0 'THEN' 'GOTO' KC1;
'IF' KODE[I,1]=2 'THEN' 'GOTO'KC2;
GEOMBC(ULX[I,1],2*I-1,0,K,NFREE,1,ADD,TEST);
'FOR' J:=2 'STEP' 1 'UNTIL' NSETF 'DO' Q[I,J]:=Q[I,1];
'IF' KODE[I,1]=1 'THEN' 'GOTO' KC1;
KC2; GEOMBC(VLY[I,1],2*I,0,K,NFREE,1,ADD,TEST);
'FOR' J:=2 'STEP' 1 'UNTIL' NSETF 'DO' Q[I,J]:=Q[I,1];
KC1:  'END';
NEWLINE(4);
WRITETEXT('('SEGSOL%%BEGINS')');
SEGSOL(K,ADD,Q,NFREE,NSETF,FAIL);
'IF' NSKEW 'NE' 0 'THEN' SKEWLOAD(NSETF,Q,NOSK,ANGSK,NSKEW,-1);
'FOR' I:=1 'STEP' 1 'UNTIL' NSETF 'DO''BEGIN'
NEWLINE(4);
 WRITETEXT('('NODAL%DISPLACEMENTS%FOR% FORCE%SET')');
          PRINT(I,2,0);
NEWLINE(3);SPACE(1);WRITETEXT('('NODE')');
SPACE(5);WRITETEXT('('X-DIRECTION')');SPACE(8);
WRITETEXT('('Y-DIRECTION')');SPACE(18);
WRITETEXT('('NODE')');SPACE(5);WRITETEXT('('X-DIRECTION')');
SPACE(8);WRITETEXT('('Y-DIRECTION')');
       W:=2*(NNODE'/'2);
'FOR' J:=2 'STEP' 2 'UNTIL' W 'DO'
  'BEGIN' NEWLINE(2);  V:=2*(J-1);
PRINT((J-1),3,0); SPACE(2); PRINT(Q[V-1,I],0,8); SPACE(2);
PRINT(Q[V,I],0,8); SPACE(13); PRINT(J,3,0); SPACE(2);
PRINT(Q[V+1,I],0,8); SPACE(2); PRINT(Q[V+2,I],0,8);
'END';
'IF' NNODE 'GT' W 'THEN'
'BEGIN' NEWLINE(2); PRINT(NNODE,3,0); SPACE(2);
        V:=2*NNODE;
        PRINT(Q[V-1,I],0,8); SPACE(2); PRINT(Q[V,I],0,8);
'END';
'END';
```

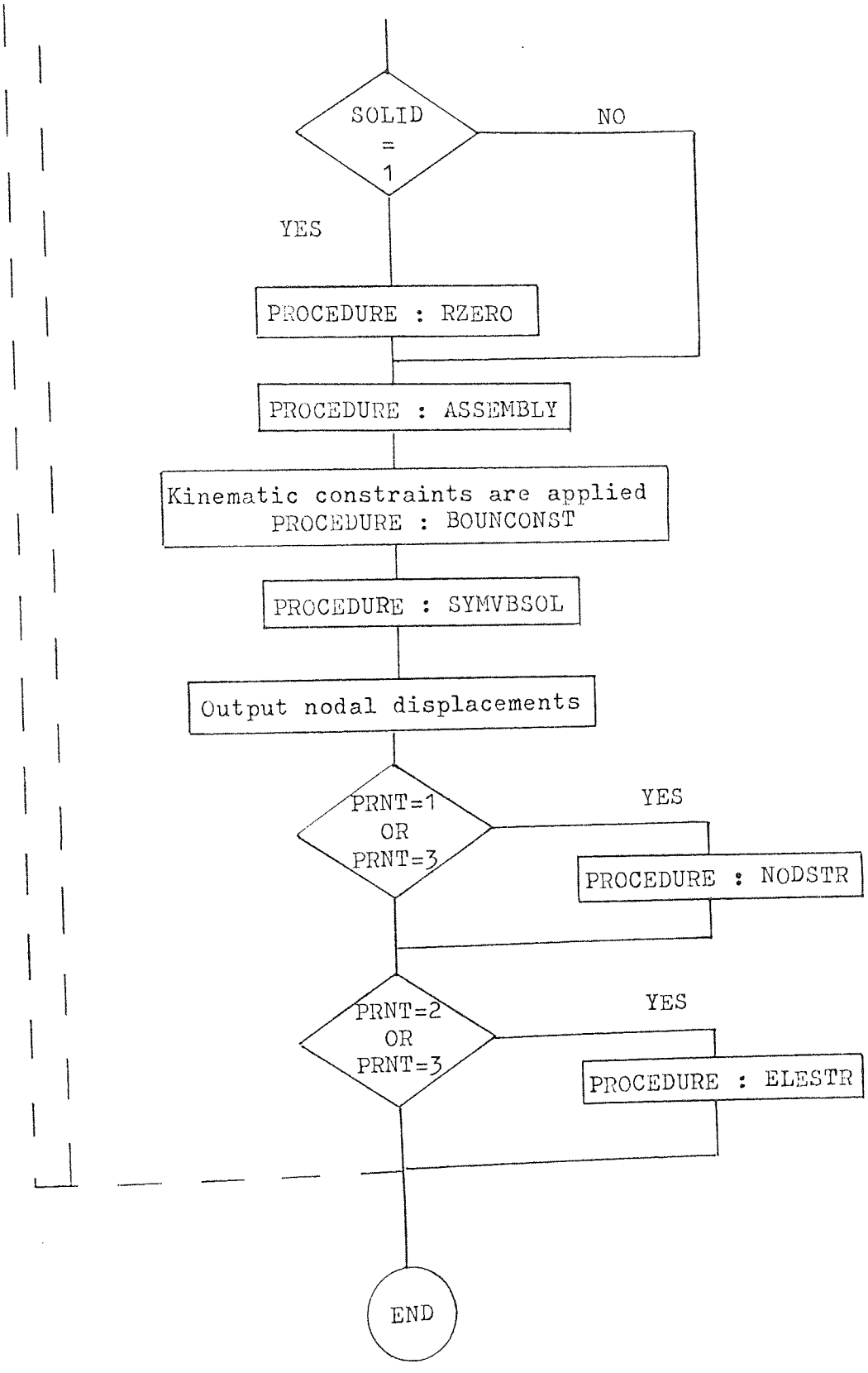
```
'IF' PRNT=1 'OR' PRNT=3'THEN'
FENOSTR(NSETF,NNODE,NELEMT,NODE,XX,YY,DETJ,Q,TH,C,AUX,QAUX);
'IF' PRNT=2 'OR'PRNT=3'THEN'
FEELSTR(NSETF,Q,NELEMT,NODE,XX,YY,DETJ,TH,C,AUX,QAUX);
'END';
'END' OF CONSTRAINT LOOP;
'END';
'GOTO' FINE;
FAIL:NEWLINE(4);
WRITETEXT('('PROGRAM%FAILED%IN%PROCEDURE%SEGSOL')');
'GOTO'EXIT;
FINE;'END'; NEWLINE(4);
WRITETEXT('('END%OF%PROGRAM%-%NORMAL%EXIT')');
EXIT:'END' OF PROGRAM;
****
```

iii. The general axisymmetric program IAAXMG.

Program flowchart




MATNO := 1 (1) NMAT
PROCEDURE : CONSTREL
I := 1 (1) NFREE
J := 1 (1) NSETF
Q[I,J]:=0.0
I := 1 (1) NNODE
PROCEDURE : LOADING
NSETF > 1
NO
YES
I := 2 (1) NSETF
NSPEC:=READ
J := 1 (1) NSPEC
K:=READ
KODE[K,I]:=READ
ULX[K,I]:=READ
VLY[K,I]:=READ
PROCEDURE : LOADING
SOLID = 0
YES
I=1
I=100


SOLID = 1
NO
YES
PROCEDURE : RZERO
PROCEDURE : ASSEMBLY
Kinematic constraints are applied
PROCEDURE : BOUNCONST
PROCEDURE : SYMVBSOL
Output nodal displacements
PRNT=1 OR PRNT=3
YES
PROCEDURE : NODSTR
PRNT=2 OR PRNT=3
YES
PROCEDURE : ELESTR
END

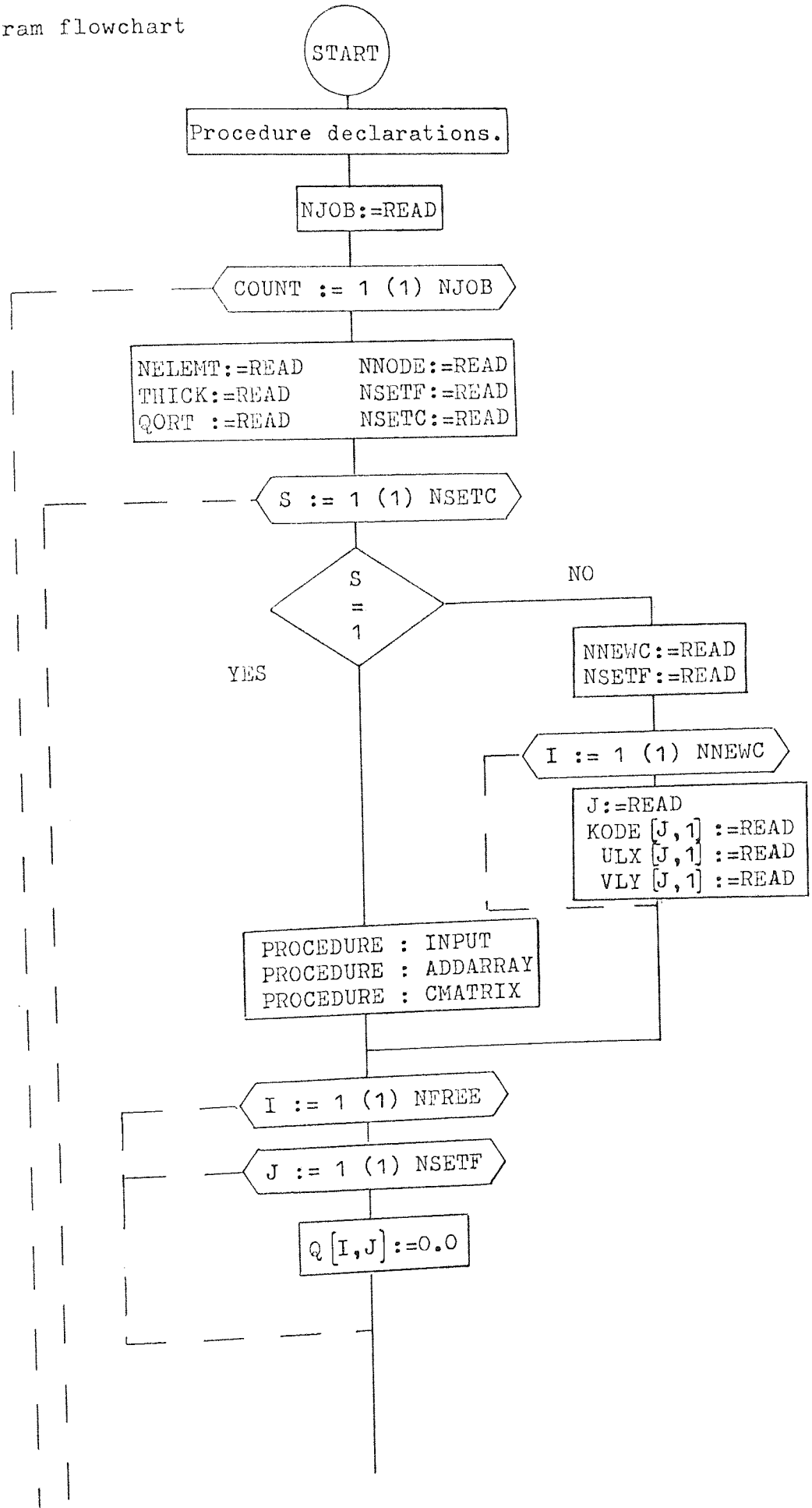
```
DOCUMENT     AAXMG


'PROGRAM'(AXXX)
'INPUT'0=CR0
'OUTPUT'0=LP0
'EXTENDED DATA'
'EXTENDED'
'BEGIN' 'COMMENT' THIS PROG EMPLOYS THE ISOPARAMETRIC
                  FORMULATION FOR AXISYMETRIC PROBLEMS
                   THE ELEMENT USED IS A SIX-NODE TRIANGLE;
'COMMENT' CHANGES IN INPUT:-
             NODAL CONNECTIONS,MATERIAL NO.
             AND ELEMENT PROPERTIES;
'INTEGER' NELEMT,NNODE,NSETFS,NFREE,I,J,K,V,W,U,NSETC,NNEWC,S,
NJOB,COUNT,PRNT,NMAT,MATNO,SYM,EC,ER,BSHP,SOLID,CASE;
'REAL' DELTA,DETJ,RMEAN,RAVG,XS,XF,YS,YF;

'PROCEDURE' DECLARATIONS


NJOB:=READ;
'FOR' COUNT:=1 'STEP' 1 'UNTIL' NJOB 'DO'
'BEGIN'
WRITETEXT('('('2C')'JOB%NAME%------')');
COPYTEXT('('END%OF%TITLE')');
NELEMT:=READ;
NNODE:=READ;
WRITETEXT('('('2C')'NO%OF%ELEMENTS%---')');
PRINT(NELEMT,3,0);
WRITETEXT('('('2C')'NO%OF%NODES%------')');
PRINT(NNODE,3,0);
NFREE:=NNODE*2;
NSETFS:=READ;
PRNT:=READ;  SOLID:=READ;
 NMAT:=READ;
'BEGIN'
'INTEGER' BAND,NSPEC,Z,COMPA,NSETF;
'REAL' 'ARRAY' XX,YY[1:NNODE],ULX,VLY[1:NNODE,1:NSETFS],
     Q[1:NFREE,1:NSETFS],C[1:NMAT,1:10],A[1:5];
'INTEGER''ARRAY' NODE[1:NELEMT,1:7],KODE[1:NNODE,1:NSETFS],ADD[0:NF
NSETC:=READ;
'FOR' S:=1 'STEP' 1 'UNTIL' NSETC 'DO'
'BEGIN'
'IF' S=1 'THEN' 'BEGIN'
NSETF:=READ;
FEINPUT(C,ADD,XX,YY,NFREE,NNODE,NSETF,KODE,NSPEC,NSETFS,
ULX,VLY,NELEMT,NODE,NMAT);   ADDARRAY(NELEMT,NNODE,ADD,NODE);
'FOR' MATNO:=1 'STEP' 1 'UNTIL' NMAT 'DO'
CONSTREL (C,A,MATNO,CASE);
'END' 'ELSE'
'BEGIN'  NNEWC:=READ;
      NSETF:=READ;
WRITETEXT('('('10C')'CONSTRAINT%SET---')');
```

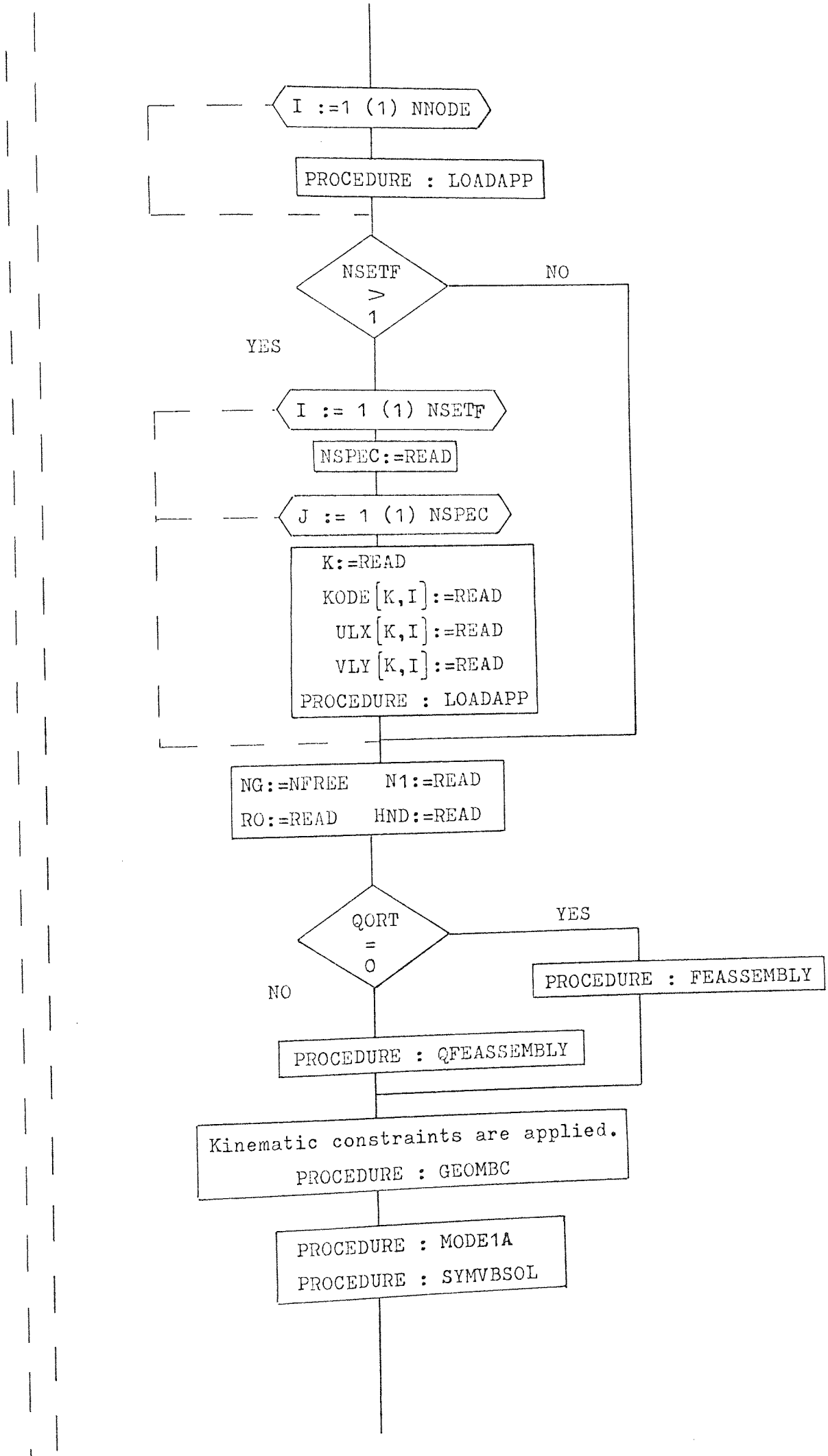
```
PRINT(S,2,0);
'FOR' I:=1 'STEP' 1 'UNTIL' NNEWC 'DO'
'BEGIN' J:=READ;
      KODE[J,1]:=READ;  ULX[J,1]:=READ;  VLY[J,1]:=READ;
'END';
WRITETEXT('('('2C')'NODAL%POINT%DATA'('2C4S')'NODE'('5S')'X%COORD
'('5S')'Y%COORD'('5S')'TYPE'('5S')'X-DISP'('5S')'Y-DISP'('C46S')'
OR%LOAD'('4S')'OR%LOAD')');
'FOR' I:=1 'STEP' 1 'UNTIL' NNODE 'DO'
'BEGIN'  NEWLINE(1); SPACE(3);
         PRINT(I,3,0);  SPACE(3);
         PRINT(XX[I],0,3);
         PRINT(YY[I],0,3);  SPACE(2);
         PRINT(KODE[I,1],3,0);  SPACE(2);
         PRINT(ULX[I,1],0,3);
         PRINT(VLY[I,1],0,3);
'END';
'END';
'FOR' I:=1 'STEP' 1 'UNTIL' NFREE 'DO'
'FOR' J:=1 'STEP' 1 'UNTIL' NSETF 'DO' Q[I,J]:=0.0;
'FOR' I:=1 'STEP' 1 'UNTIL' NNODE 'DO'
     LOADING(KODE[I,1],ULX[I,1],VLY[I,1],Q,I,1);
'IF' NSETF 'GT' 1 'THEN'
'BEGIN'
'FOR' I:=2 'STEP' 1 'UNTIL' NSETF 'DO'
'BEGIN'
NSPEC:=READ;
WRITETEXT('('('2C')'FORCE%SET-----')');
PRINT(I,3,0);
WRITETEXT('('('2C')'NODE'('5S')'TYPE'('3S')'X-DISP'('7S')'Y-DISP
'('C21S')'OR%LOAD'('6S')'OR%LOAD')');
'FOR' J:=1 'STEP' 1 'UNTIL' NSPEC 'DO'
'BEGIN'
   K:=READ;  KODE[K,I]:=READ;  ULX[K,I]:=READ;  VLY[K,I]:=READ;
LOADING(KODE[K,I],ULX[K,I],VLY[K,I],Q,K,I);
NEWLINE(2);
PRINT(K,3,0);  SPACE(2);
PRINT(KODE[K,I],3,0);  SPACE(2);
PRINT(ULX[K,I],0,4);
PRINT(VLY[K,I],0,4);
'END';
'END';
'END';
'IF'SOLID=0'THEN'I:=1'ELSE'I:=100;
'BEGIN'
 'REAL' 'ARRAY' K[1:ADD[NFREE]];
'INTEGER''ARRAY'ZER[1:I];
'IF'SOLID=1'THEN'
RZERO(NODE,XX,YY,ZER);
ASSEMBLY(NELEMT,K,XX,YY,DETJ,NODE,C,ADD,NFREE,NSETF,STRDIS,ZER);
'COMMENT' INTRODUCTION OF KINEMATIC CONSTRAINTS;
'FOR' I:=1 'STEP' 1 'UNTIL' NNODE 'DO'
'BEGIN'
'IF' KODE[I,1]=0 'THEN' 'GOTO' KC1;
'IF' KODE[I,1]=2 'THEN' 'GOTO'KC2;
```

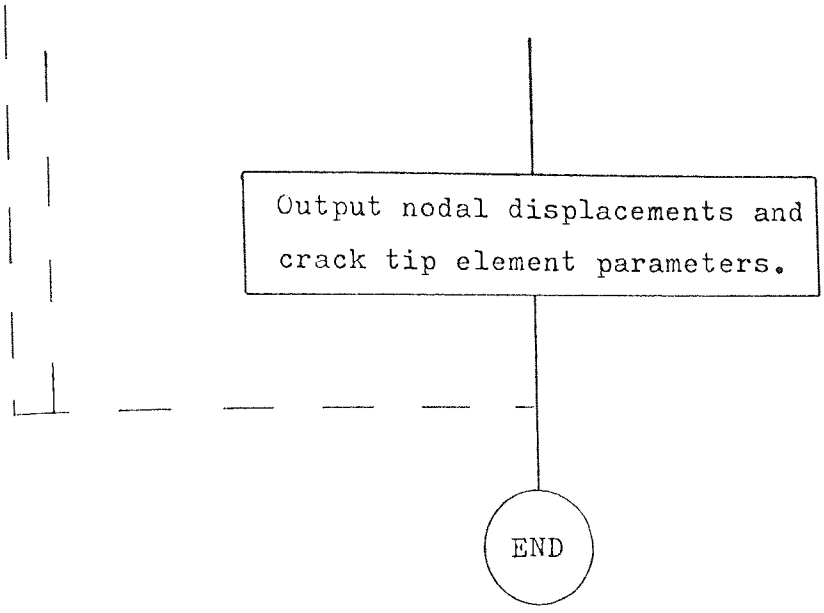
```
BOUNCONST(ULX[I,1],2*I-1,Q,K,NFREE,1,ADD);
'FOR' J:=2 'STEP' 1 'UNTIL' NSETF 'DO' Q[I,J]:=Q[I,1];
'IF' KODE[I,1]=1 'THEN' 'GOTO' KC1;
KC2: BOUNCONST(VLY[I,1],2*I,Q,K,NFREE,1,ADD);
'FOR' J:=2 'STEP' 1 'UNTIL' NSETF 'DO' Q[I,J]:=Q[I,1];
KC1:  'END';
WRITETEXT('('('4C')'SYNVBSOL%%BEGINS')');
SYMVBSOL(K,K,ADD,Q,NFREE,NSETF,FAIL);
'FOR' I:=1 'STEP' 1 'UNTIL' NSETF 'DO'
'BEGIN' WRITETEXT('('('4C')'NODAL%DISPLACEMENTS%FOR% FORCE%SET')');
        PRINT(I,2,0);
WRITETEXT('('('3CS')'NODE'('5S')'R-DIRECTION'('8S')'Z-DIRECTION
'('18S')'NODE'('5S')'R-DIRECTION'('8S')'Z-DIRECTION')');
      W:=2*(NNODE'/'2);
'FOR' J:=2 'STEP' 2 'UNTIL' W 'DO'
  'BEGIN' NEWLINE(2);  V:=2*(J-1);
PRINT((J-1),3,0); SPACE(2); PRINT(Q[V-1,I],0,8); SPACE(2);
PRINT(Q[V,I],0,8); SPACE(13); PRINT(J,3,0); SPACE(2);
PRINT(Q[V+1,I],0,8); SPACE(2); PRINT(Q[V+2,I],0,8);
'END';
'IF' NNODE 'GT' W 'THEN'
'BEGIN' NEWLINE(2); PRINT(NNODE,3,0); SPACE(2);
        V:=2*NNODE;
        PRINT(Q[V-1,I],0,8); SPACE(2); PRINT(Q[V,I],0,8);
'END';
'END';
'IF' PRNT=1 'OR' PRNT=3'THEN'
NODSTR(NSETF,NNODE,NELEMT,NODE,XX,YY,DETJ,Q,C,STRDIS,ZER);
'IF' PRNT=2 'OR'PRNT=3'THEN'
ELESTR(NSETF,Q,NELEMT,NODE,XX,YY,DETJ,C,STRDIS,ZER);
'END';
'END' OF CONSTRAINT LOOP;
'END';
FAIL: 'END' OF JOB LOOP;
'END' OF PROGRAM;
****
```

iv. 2-D F.E. mode I fracture program PCQTM1ST.

Program flowchart




I :=1 (1) NNODE
PROCEDURE : LOADAPP
NSETF > 1
NO
YES
I := 1 (1) NSETF
NSPEC:=READ
J := 1 (1) NSPEC
K:=READ
KODE[K,I]:=READ
ULX[K,I]:=READ
VLY[K,I]:=READ
PROCEDURE : LOADAPP
NG:=NFREE N1:=READ
RO:=READ HND:=READ
QORT = 0
YES
NO
PROCEDURE : FEASSEMBLY
PROCEDURE : QFEASSEMBLY
Kinematic constraints are applied.
PROCEDURE : GEOMBC
PROCEDURE : MODE1A
PROCEDURE : SYMVBSOL


Output nodal displacements and crack tip element parameters.
END

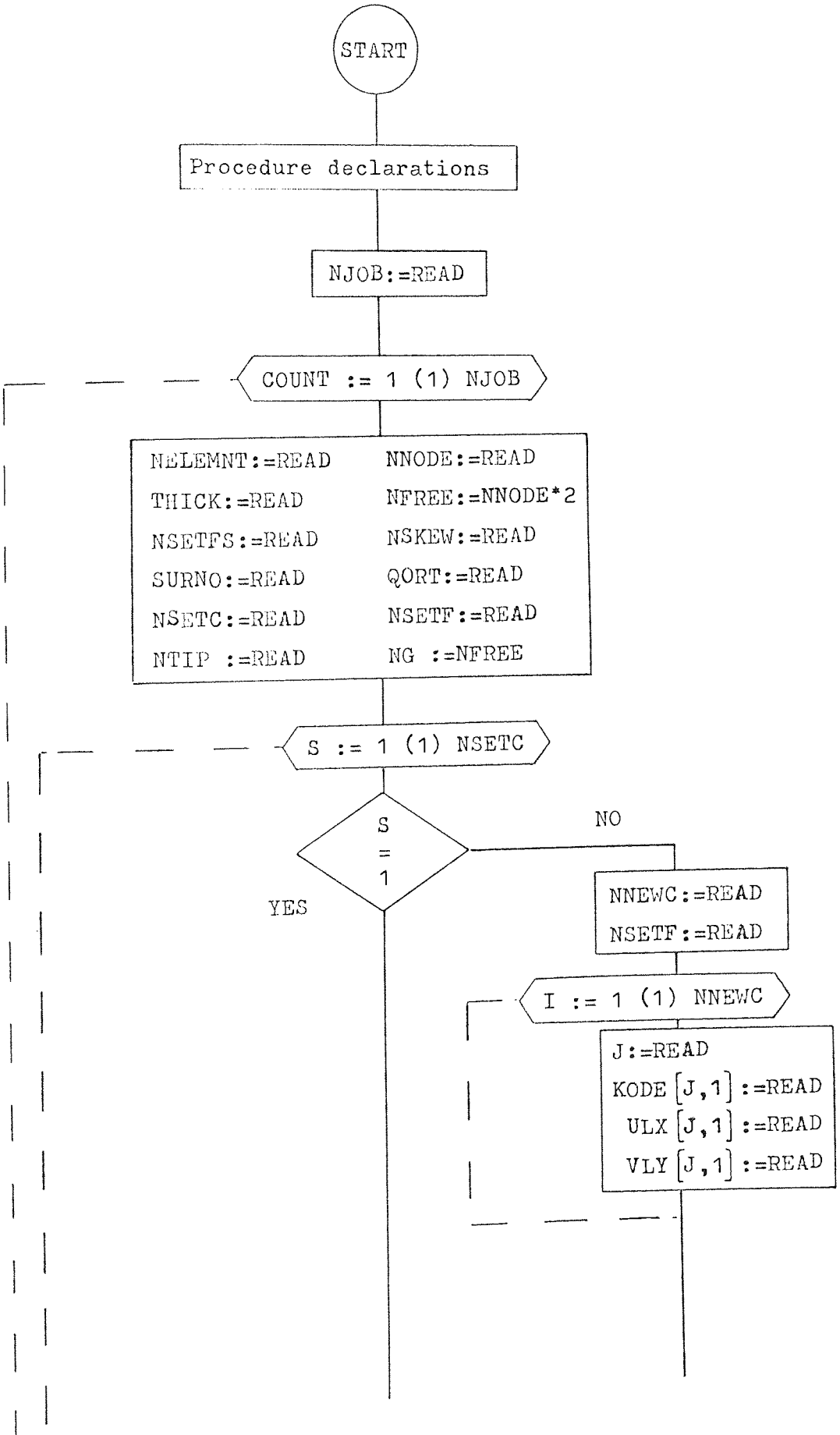
```
DOCUMENT     PCOTH4ST

'BEGIN' 'COMMENT'  THIS PROGRAM EMPLOYS THE ISOPARAMETRIC
                   FORMULATION FOR 2-D PROBLEMS OF PLANE STRESS/STR
                   TWO ELEMENTS CAN BE SELECTED, VIA THE
                  QORT PARAMETER - EITHER A SIX-NODE TRIANGLE
                  OR AN EIGHT NODE QUADRILATERAL.
                  THIS PROGRAM ACCOMODATES MODE 1, SINGLE TIP
                  CRACK PROBLEMS;
'INTEGER' NELEMT,NNODE,NSETF,NFREE,I,J,V,U,P,K,CASE,NSETC,NFREC,S,
NJOB,COUNT,QORT;
'REAL' DELTA,DETJ,THICK;
'ARRAY' C[1:3,1:3],A[1:5];

'PROCEDURE' DECLARATIONS


    SELECT INPUT(11);

NJOB:=READ;
'FOR' COUNT:=1 'STEP' 1 'UNTIL' NJOB 'DO'
'BEGIN'
NEWLINE(2);
WRITETEXT('('JOB%NAME%------')');
COPYTEXT('('END%OF%TITLE')');
NELEMT:=READ;NNODE:=READ;THICK:=READ;NSETF:=READ;
NEWLINE(2);
WRITETEXT('('NO%OF%ELEMENTS%----')');
PRINT(NELEMT,3,0);
NEWLINE(2);
WRITETEXT('('NO%OF%NODES%-------')');
PRINT(NNODE,3,0);
QORT:=READ;
NEWLINE(2);
WRITETEXT('('ELEMENT%SELECTED%--------%')');
'IF'QORT=0'THEN'WRITETEXT('('TRIANGULAR')')
          'ELSE'WRITETEXT('('QUADRILATERAL')');
NFREE:=NNODE*2;
'BEGIN'  'INTEGER' NSPEC,Z,COMPA;
'REAL' 'ARRAY' XX,YY[1:NNODE],ULX,VLY[1:NNODE,1:NSETF],
                Q[1:NFREE,1:NSETF];
'INTEGER' 'ARRAY' NODE[1:NELEMT,1:6+2*QORT],KODE[1:NNODE,1:NSETF],
 ADD[0:NFREE];
NSETC:=READ;
'FOR' S:=1 'STEP' 1 'UNTIL' NSETC 'DO'
'BEGIN'  'IF' S=1 'THEN'
'BEGIN'
INPUT(ADD,XX,YY,NFREE,NNODE,NSETF,KODE,NSPEC,ULX,VLY,NELEMT,NODE);
ADDARRAY(NELEMT,NNODE,ADD,NODE);
CMATRIX(C,CASE,A);
'END' 'ELSE'
'BEGIN'  NNEWC:=READ;  NSETF:=READ;
'FOR' I:=1 'STEP' 1 'UNTIL' NNEWC 'DO'
```

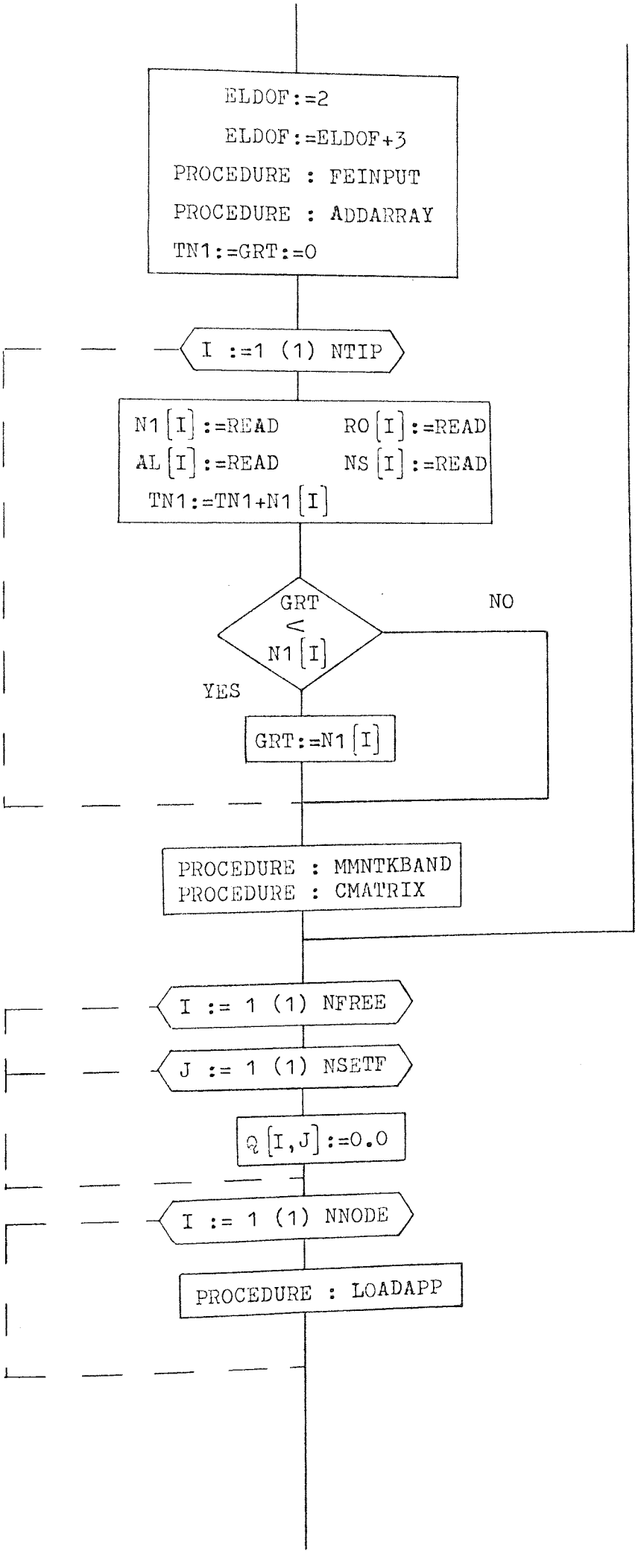
```
'BEGIN' J:=READ;
      KODE[J,1]:=READ;  ULX[J,1]:=READ;  VLY[J,1]:=READ;
'END';
'END';
NEWLINE(2);
WRITETEXT('('NO%OF%DEGREES%OF%FREEDOM----')');
PRINT(NFREE,3,0);
NEWLINE(2);
'FOR' I:=1 'STEP' 1 'UNTIL' NFREE 'DO'
'FOR' J:=1 'STEP' 1 'UNTIL' NSETF 'DO' Q[I,J]:=0.0;
'FOR' I:=1 'STEP' 1 'UNTIL' NNODE 'DO'
     LOADAPP(KODE[I,1],ULX[I,1],VLY[I,1],Q,I,1);
'IF' NSETF 'GT' 1 'THEN'
'BEGIN'
'FOR' I:=1 'STEP' 1 'UNTIL' NSETF 'DO'
'BEGIN'
NSPEC:=READ;
NEWLINE(2);
WRITETEXT('('FORCE%SET-----')');
PRINT(I,3,0);
NEWLINE(2);WRITETEXT('('NODE')');
SPACE(5);WRITETEXT('('TYPE')');SPACE(5);
WRITETEXT('('X-DISP')');SPACE(5);WRITETEXT('('Y-DISP')');
NEWLINE(1);SPACE(10);WRITETEXT('('OR%LOAD')');
SPACE(4);WRITETEXT('('OR%LOAD')');
'FOR' J:=1 'STEP' 1 'UNTIL' NSPEC 'DO'
'BEGIN'
   K:=READ;  KODE[K,I]:=READ;  ULX[K,I]:=READ;  VLY[K,I]:=READ;
LOADAPP(KODE[K,I],ULX[K,I],VLY[K,I],Q,K,I);
NEWLINE(2);
PRINT(K,3,0);  SPACE(2);
PRINT(KODE[K,I],0,4);  SPACE(2);
PRINT(ULX[K,I],0,4);
PRINT(VLY[K,I],0,4);
'END';
'END';
'END';
NEWLINE(6);
'BEGIN'
'INTEGER' N1,NG,NNO;
'REAL' RO;
'REAL' 'ARRAY' K[1:ADD[NFREE]];
NG:=NFREE;
N1:=READ;  RO:=READ;  NNO:=READ;
'IF'QORT=0'THEN'
FEASSEMBLY(NELEMT,K,XX,YY,DETJ,NODE,C,THICK,ADD)
          'ELSE'
QFEASSEMBLY(NELEMT,K,XX,YY,DETJ,NODE,C,THICK,ADD);
'COMMENT' INTRODUCTION OF KINEMATIC CONSTRAINTS;
'FOR' I:=1 'STEP' 1 'UNTIL' NNODE 'DO'
'BEGIN'
'IF' KODE[I,1]=0 'THEN' 'GOTO' KC1;
'IF' KODE[I,1]=2 'THEN' 'GOTO'KC2;
GEOMBC(ULX[I,1],2*I-1,Q,K,NFREE,1,ADD);
'FOR' J:=2 'STEP' 1 'UNTIL' NSETF 'DO' Q[I,J]:=Q[I,1];
```

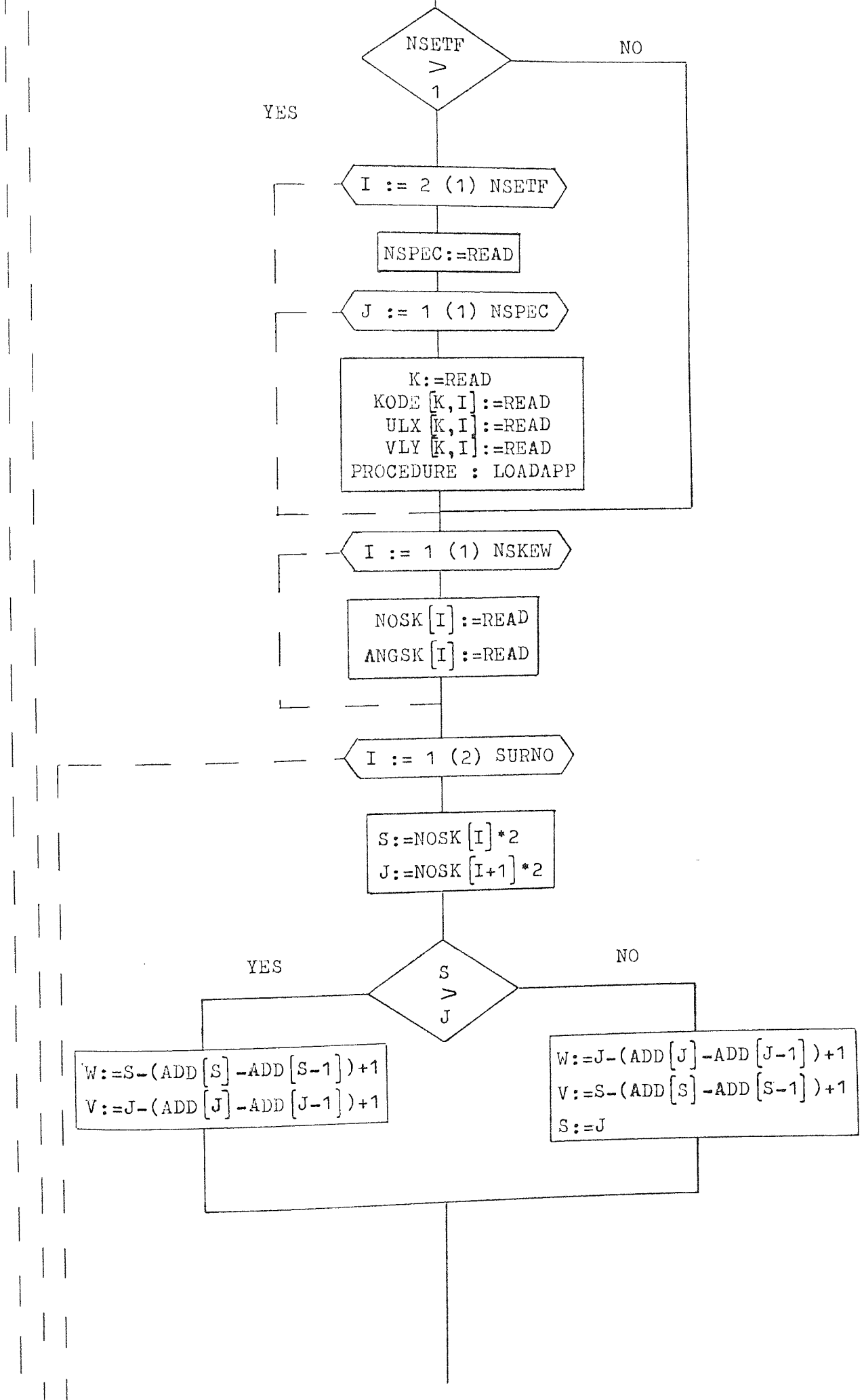
```
'IF' KODE[I,1]=1 'THEN' 'GOTO' KC1;
KC2; GEOMBC(VLY[I,1],2*I,0,K,NFREE,1,ADD);
'FOR' J:=2 'STEP' 1 'UNTIL' NSETF 'DO' G[I,J]:=Q[I,1];
KC1;  'END';
NEWLINE(1);SPACE(2);
WRITETEXT('('MODE1A%BEGINS')');
MODE1A(N1,RO,NG,NSETF,A,ADD,K,0,CASE,END);
NEWLINE(2);
NEWLINE(2);
WRITETEXT('('SYMVBSOL%%BEGINS')');
SYMVBSOL(K,K,ADD,0,NG-2*N1+2,NSETF,FAIL);
NEWLINE(6);
'FOR' I:=1 'STEP' 1 'UNTIL' NSETF 'DO'
'BEGIN'
NEWLINE(4);
        WRITETEXT('('NODAL%DISPLACEMENTS%FOR%FORCE%SET')');
        PRINT(I,2,0);
NEWLINE(3);SPACE(1);WRITETEXT('('NODE')');
SPACE(5);WRITETEXT('('X-DIRECTION')');SPACE(8);
WRITETEXT('('Y-DIRECTION')');SPACE(10);
WRITETEXT('('NODE')');SPACE(5);WRITETEXT('('X-DIRECTION')');
SPACE(8);WRITETEXT('('Y-DIRECTION')');
      U:=2*((NNODE-N1)'/'2);
      'FOR' J:=2 'STEP' 2 'UNTIL' U 'DO'
      'BEGIN' NEWLINE(2);  V:=2*(J-1);  Z:=J+N1;
         PRINT(Z-1,3,0);  SPACE(2);  PRINT(Q[V-1,I],0,8);
         SPACE(2);  PRINT(Q[V,I],0,8);  SPACE(13);
         PRINT(Z,3,0);  SPACE(2);  PRINT(Q[V+1,I],0,8);
         SPACE(2);  PRINT(Q[V+2,I],0,8);
      'END';
      'IF' NNODE-N1 'GT' U 'THEN'
      'BEGIN' NEWLINE(2);  PRINT(NNODE-N1,3,0);  SPACE(2);
         V:=2*(NNODE-N1);
         PRINT(Q[V-1,I],0,8);SPACE(2);PRINT(Q[V,I],0,8);
      'END';
      V:=NG-2*N1;
 NEWLINE(2);
        WRITETEXT('('CRACK%TIP%DISPLACEMENT%
        IN%X-DIRECTION=')');
        PRINT(Q[V+1,I],0,10);
 NEWLINE(2);
        WRITETEXT('('MODE1%STRESS%INTENSITY%FACTOR,K1=')');
        PRINT(Q[V+2,I],0,10);
 'END';
 'END';
 'END';
 'END';
 'GOTO'  FINE;
 FAIL;NEWLINE(4);
 WRITETEXT('('PROGRAM%FAILED%IN%PROCEDURE%SYMBVSOL')');
 'GOTO'EXIT;
 FINE;'END'; NEWLINE(4);
 WRITETEXT('('END%OF%PROGRAM%-%NORMAL%EXIT')');
 EXIT;'END' OF PROGRAM;
```

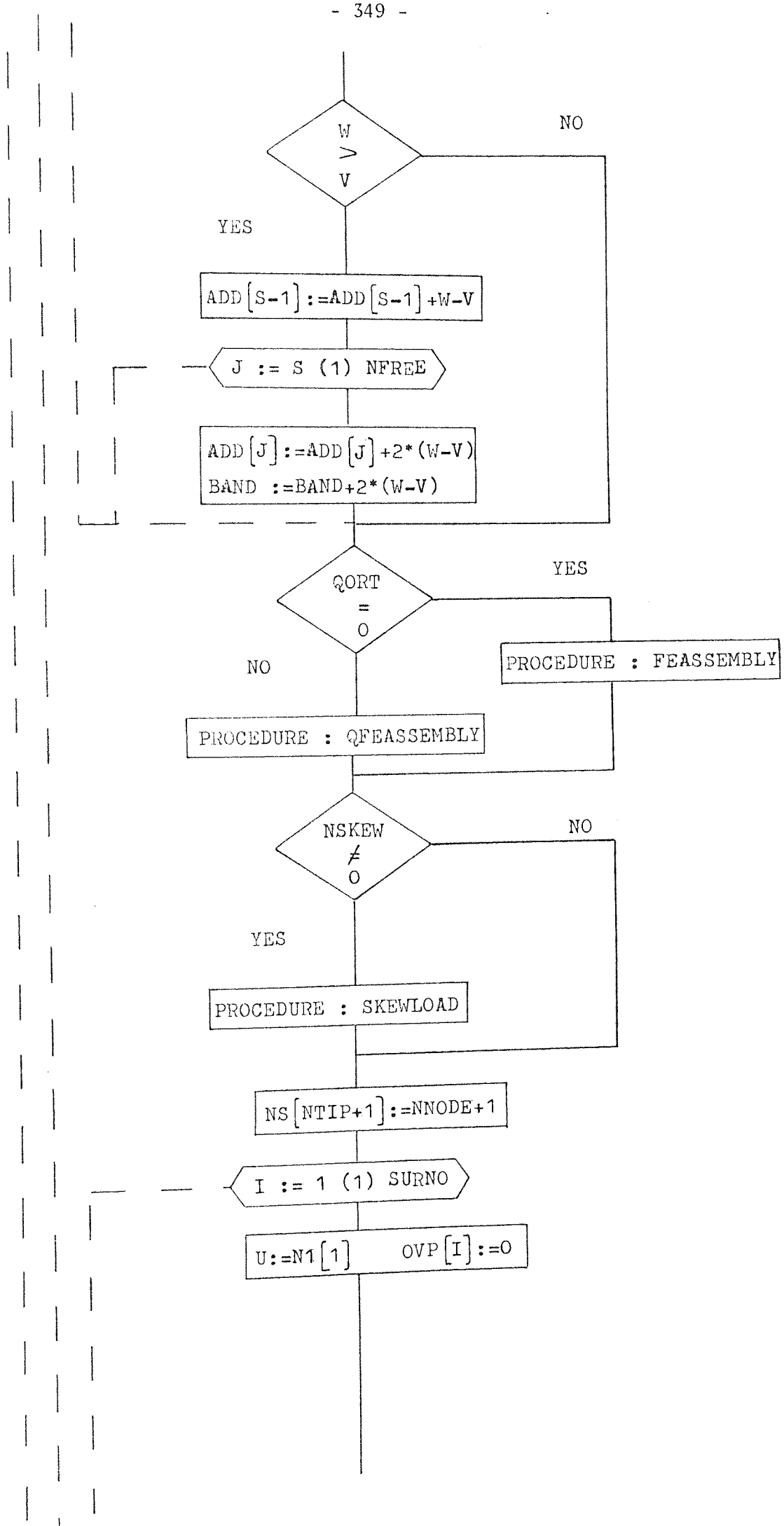
v. 2-D F.E. program for multi-tip,mixed-mode fracture problems, incorporating a partial crack closure facility. PCPOY & PCPOXY.

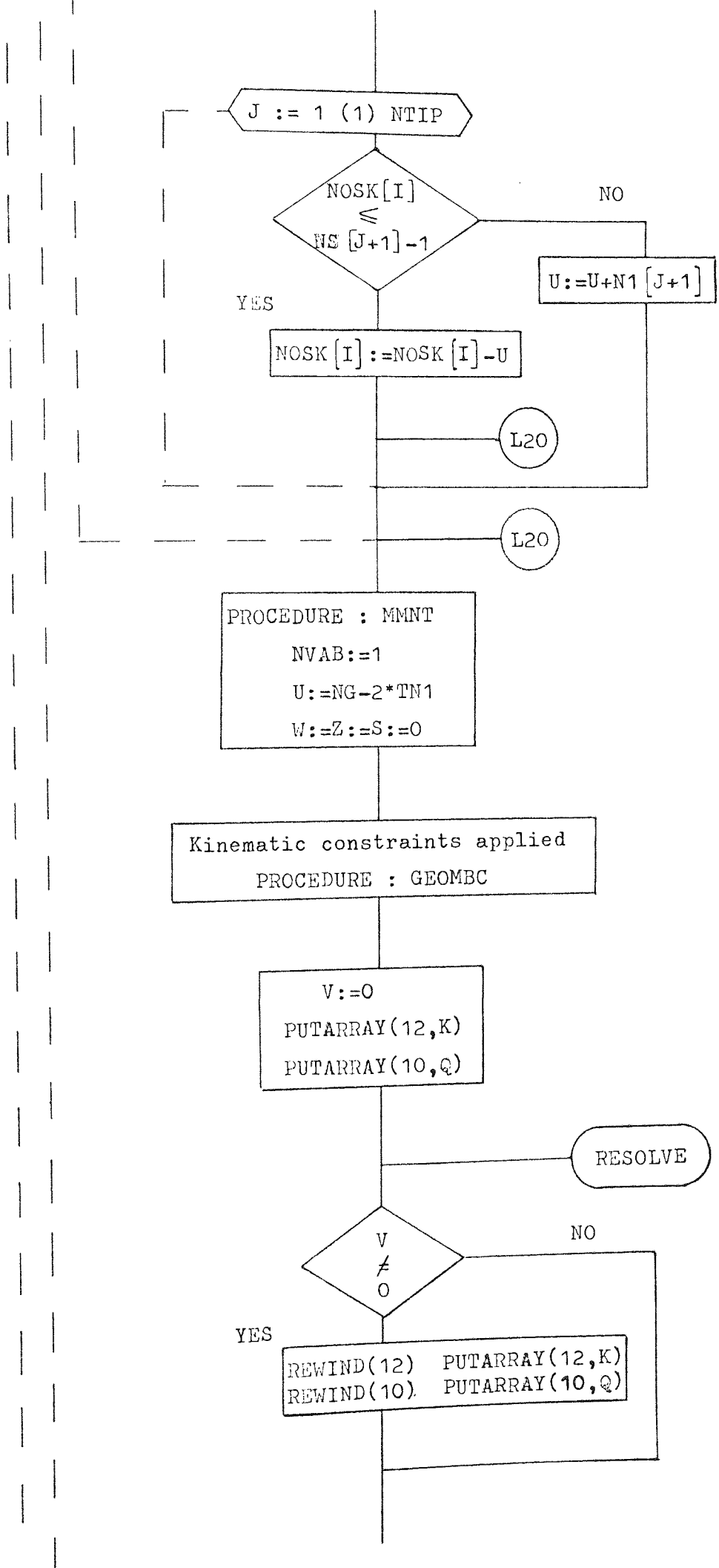
Program flowchart

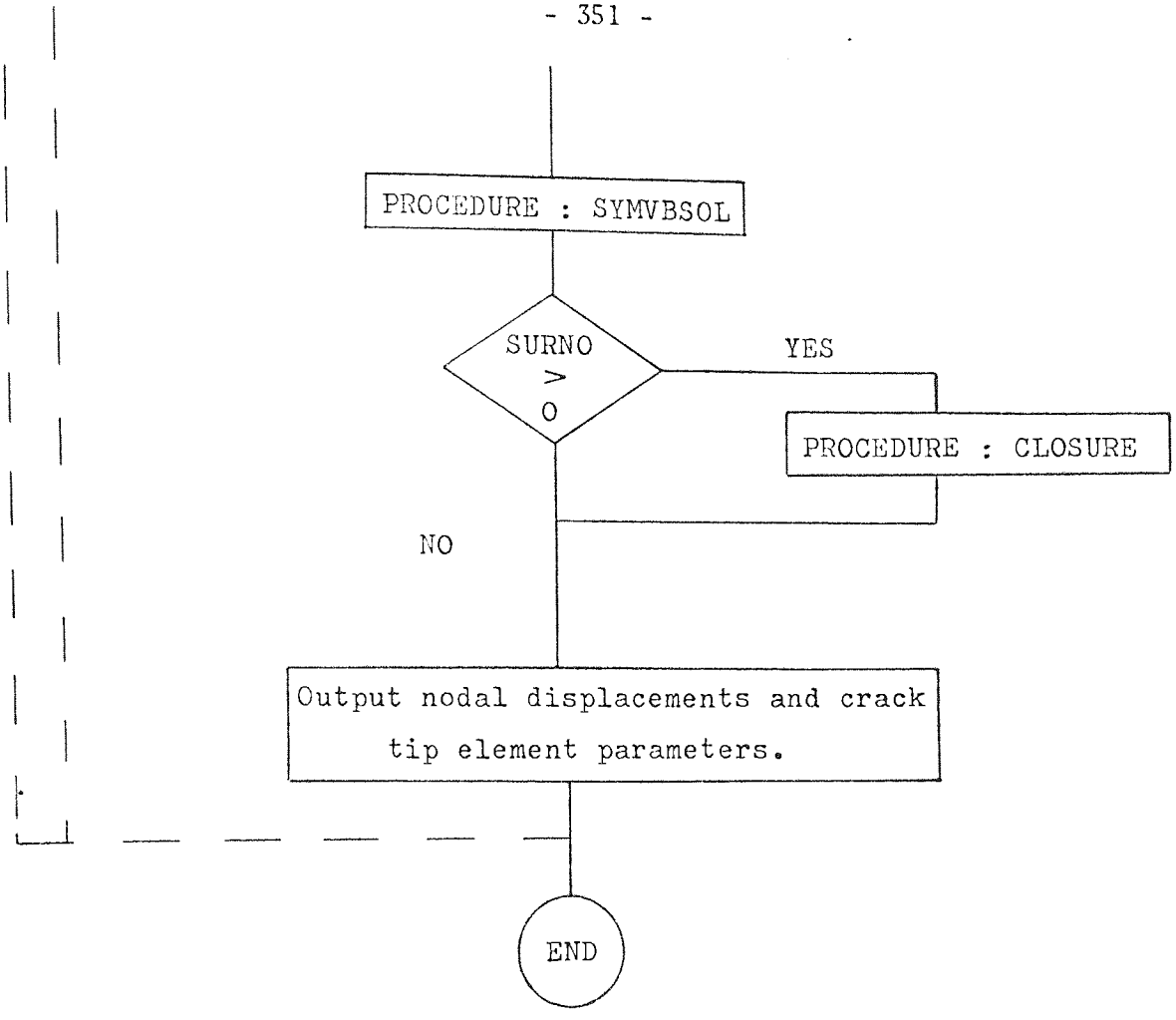



ELDOF:=2
ELDOF:=ELDOF+3
PROCEDURE : FEINPUT
PROCEDURE : ADDARRAY
TN1:=GRT:=0
I :=1 (1) NTIP
N1[I]:=READ
RO[I]:=READ
AL[I]:=READ
NS[I]:=READ
TN1:=TN1+N1[I]
GRT < N1[I]
NO
YES
GRT:=N1[I]
PROCEDURE : MMNTKBAND
PROCEDURE : CMATRIX
I := 1 (1) NFREE
J := 1 (1) NSETF
Q[I,J]:=0.0
I := 1 (1) NNODE
PROCEDURE : LOADAPP


NSETF > 1
NO
YES
I := 2 (1) NSETF
NSPEC:=READ
J := 1 (1) NSPEC
K:=READ
KODE [K,I]:=READ
ULX [K,I]:=READ
VLY [K,I]:=READ
PROCEDURE : LOADAPP
I := 1 (1) NSKEW
NOSK [I]:=READ
ANGSK [I]:=READ
I := 1 (2) SURNO
S:=NOSK [I]*2
J:=NOSK [I+1]*2
S > J
YES
NO
W:=S-(ADD [S]-ADD [S-1])+1
V:=J-(ADD [J]-ADD [J-1])+1
W:=J-(ADD [J]-ADD [J-1])+1
V:=S-(ADD [S]-ADD [S-1])+1
S:=J


W > V
NO
YES
ADD[S-1] :=ADD[S-1] +W-V
J := S (1) NFREE
ADD[J] :=ADD[J] +2*(W-V)
BAND :=BAND+2*(W-V)
QORT = 0
YES
NO
PROCEDURE : FEASSEMBLY
PROCEDURE : QFEASSEMBLY
NSKEW ≠ 0
NO
YES
PROCEDURE : SKEWLOAD
NS[NTIP+1] :=NNODE+1
I := 1 (1) SURNO
U:=N1[1] OVP[I] :=0


J := 1 (1) NTIP
NOSK[I] ⩽ NS[J+1]-1
NO
U:=U+N1[J+1]
YES
NOSK[I]:=NOSK[I]-U
L20
L20
PROCEDURE : MMNT
NVAB:=1
U:=NG-2*TN1
W:=Z:=S:=0
Kinematic constraints applied
PROCEDURE : GEOMBC
V:=0
PUTARRAY(12,K)
PUTARRAY(10,Q)
RESOLVE
V ≠ 0
NO
YES
REWIND(12) PUTARRAY(12,K)
REWIND(10) PUTARRAY(10,Q)


PROCEDURE : SYMVBSOL
SURNO
>
0
YES
PROCEDURE : CLOSURE
NO
Output nodal displacements and crack tip element parameters.
END

```
DOCUMENT     PCPUY


'BEGIN' 'COMMENT'  THIS PROGRAM EMPLOYS THE ISOPARAMETRIC
                   FORMULATION FOR 2-D PROBLEMS OF PLANE STRESS/STR
                   TWO ELEMENTS CAN BE SELECTED, VIA
                  THE QORT PARAMETER - EITHER A SIX-NODE TRIANGLE
                  OR AN EIGHT-NODE QUADRILATERAL.
                  THIS PROGRAM ACCOMMODATES MIXED-NODE MULTI
                  CRACK TIP PROBLEMS ,WITH PARTIAL CRACK CLOSURE;
'INTEGER' NELEMT,NNODE,NSETFS,NFREE,I,J,K,V,U,W,CASE,NSETC,NNEWC,S,
NJOB,COUNT,PRNT,NVAB1,NSKEW,SURNO,C1,C2,NSETF,QORT,BAND,NTIP,TN1,ELI
'REAL' DELTA,DETJ,THICK,EQ,GG,COH,K1,K2,TH,SS;
'ARRAY' C[1:3,1:31,A[1:5];

'PROCEDURE' DECLARATIONS
-----------------------------------------------------------------


 SELECT INPUT(11);
NJOB:=READ;
'FOR' COUNT:=1 'STEP' 1 'UNTIL' NJOB 'DO'
'BEGIN'
PAPERTHROW;
NEWLINE(2);
WRITETEXT('('JOB%NAME%------')');
COPYTEXT('('END%OF%TITLE')');
NELEMT:=READ;
NNODE:=READ;
THICK:=READ;
NEWLINE(2);
WRITETEXT('('NO%OF%ELEMENTS%----')');
PRINT(NELEMT,3,0);
NEWLINE(2);
WRITETEXT('('NO%OF%NODES%-------')');
PRINT(NNODE,3,0);
NFREE:=NNODE*2;
NSETFS:=READ;
  NSKEW:=READ;  SURNO:=READ;
QORT:=READ;
NSETC:=READ;  NSETF:=READ;
NTIP:=READ;
NEWLINE(2);
WRITETEXT('('ELEMENT%SELECTED%--------%')');
'IF'QORT=0'THEN'WRITETEXT('('TRIANGULAR')')
          'ELSE'WRITETEXT('('QUADRILATERAL')');
'BEGIN'
'INTEGER' NSPEC,Z,COMPA,GRT,NG;
'REAL' 'ARRAY' XX,YY[1:NNODE],ULX,VLY[1:NNODE,1:NSETFS],
                Q[1:NFREE,1:NSETFS],ANGSK[1:NSKEW+1],AL,RO[1:NTIP];
'INTEGER''ARRAY' NODE[1:NELEMT,1:6+2*QORT],KODE[1:NNODE,1:NSETFS]
,ADDLO:NFREE],
NOSK[1:NSKEW+1],N1,NS[0:NTIP+1];
  NG:=NFREE;
'FOR' S:=1 'STEP' 1 'UNTIL' NSETC 'DO'
```

```
'BEGIN'
'IF' S=1 'THEN' 'BEGIN'
ELDOF:=2; ELDOF:=ELDOF+3;
FEINPUT(ADD,XX,YY,NFREE,NNODE,NSETF,KODE,NSPEC,NSETFS,ULX,VLY,NELEM
        NODE);
ADDARRAY(NELEMT,NNODE,ADD,NODE);
TN1:=GRT:=0;
'FOR'I:=1'STEP'1'UNTIL'NTYP'DO'
'BEGIN' N1[I]:=READ;  RO[I]:=READ;  AL[I]:=READ;
        NS[I]:=READ; TN1:=TN1+N1[I];
        'IF' GRT<N1[I]'THEN'GRT:=N1[I];
'END';
MMNTKBAND(NFREE,N1,BAND,ADD,NS);
CMATRIX(C,CASE,A);
NEWLINE(2);
WRITETEXT('('NO%OF%DEGREES%OF%FREEDO'1----')');
PRINT(NFREE,3,0);
NEWLINE(2);
'END' 'ELSE'
'BEGIN'  NNEVC:=READ;
      NSETF:=READ;
'FOR' I:=1 'STEP' 1 'UNTIL' NNEVC 'DO'
'BEGIN' J:=READ;
      KODE[J,1]:=READ;  ULX[J,1]:=READ;  VLY[J,1]:=READ;
'END';
'END';
'FOR' I:=1 'STEP' 1 'UNTIL' NFREE 'DO'
'FOR' J:=1 'STEP' 1 'UNTIL' NSETF 'DO' Q[I,J]:=0.0;
'FOR' I:=1 'STEP' 1 'UNTIL' NNODE 'DO'
     LOADAPP(KODE[I,1],ULX[I,1],VLY[I,1],Q,I,1);
'IF' NSETF 'GT' 1 'THEN'
'BEGIN'
'FOR' I:=2 'STEP' 1 'UNTIL' NSETF 'DO'
'BEGIN'
NSPEC:=READ;
NEWLINE(2);
WRITETEXT('('FORCE%SET-----')');
PRINT(I,3,0);
NEWLINE(2);WRITETEXT('('NODE')');
SPACE(5);WRITETEXT('('TYPE')');SPACE(8);
WRITETEXT('('X-DISP')');SPACE(7);WRITETEXT('('Y-DISP')');
NEWLINE(1);SPACE(21);WRITETEXT('('OR%LOAD')');
SPACE(6);WRITETEXT('('OR%LOAD')');
'FOR' J:=1 'STEP' 1 'UNTIL' NSPEC 'DO'
'BEGIN'
   K:=READ;  KODE[K,I]:=READ;  ULX[K,I]:=READ;  VLY[K,I]:=READ;
LOADAPP(KODE[K,I],ULX[K,I],VLY[K,I],Q,K,I);
NEWLINE(2);
PRINT(K,3,0);  SPACE(2);
PRINT(KODE[K,I],3,0);  SPACE(2);
PRINT(ULX[K,I],0,4);
PRINT(VLY[K,I],0,4);
'END';
'END';
'END';
```

```
'FOR' I:=1 'STEP' 1 'UNTIL' NSKEW 'DO'
'BEGIN' NOSK[I]:=READ;
        ANGSK[I]:=READ;
'END';
'FOR'I:=1'STEP'2'UNTIL'SURNO'DO'
'BEGIN'
   S:=NOSK[I]*2;  J:=NOSK[I+1]*2;
   'IF' S>J 'THEN' 'BEGIN'
      W:=S-(ADD[S]-ADD[S-1])+1;
      V:=J-(ADD[J]-ADD[J-1])+1;
      'END' 'ELSE''BEGIN'
      W:=J-(ADD[J]-ADD[J-1])+1;
      V:=S-(ADD[S]-ADD[S-1])+1;
      S:=J;
     'END';
   'IF' U > V 'THEN'
   'BEGIN'
      ADD[S-1]:=ADD[S-1]+U-V;
       'FOR'J:=S'STEP'1'UNTIL'NFREE'DO'
         ADD[J]:=ADD[J]+2*(U-V);
         BAND:=BAND+2*(W-V);
   'END';
'END' I;
NEWLINE(6);
'BEGIN' 'INTEGER' SUB1,SUB2,SUB3,SUBSCP;
'REAL''ARRAY' K[1:BAND];
'INTEGER''ARRAY' OVP[0:SURNO];
'IF'QORT=0'THEN'
FEASSEMBLY(NELEMT,K,XX,YY,DETJ,NODE,C,THICK,ADD,NFREE,
           AUX,SKEWEDCON,NSKEW,NSETF,ANGSK,NOSK)
          'ELSE'
QFEASSEMBLY(NELEMT,K,XX,YY,DETJ,NODE,C,THICK,ADD,NFREE,
            QAUX,SKEWEDCON,NSKEW,NSETF,ANGSK,NOSK);
'IF' NSKEW 'NE' 0 'THEN'
      SKEWLOAD(NSETF,0,NOSK,ANGSK,NSKEW,1);
NS[NTIP+1]:=NNODE+1;
'FOR'I:=1'STEP'1'UNTIL'SURNO'DO'
'BEGIN'  U:=N1[1];   OVP[I]:=0;
   'FOR'J:=1'STEP'1'UNTIL'NTIP'DO'
   'BEGIN'
   'IF'NOSK[I]'LE'NS[J+1]-1'THEN'
      'BEGIN' NOSK[I]:=NOSK[I]-U;
              'GOTO' L20;
      'END'
      'ELSE'
         U:=U+N1[J+1];
   'END'J;
L20;'END'I;
NEWLINE(6);
MMNT(NG,NSETF,A,ADD,K,Q,CASE,THICK,N1,RO,GRT,AL,NS);
NVAB1:=1;
U:=NG-2*TN1;
'COMMENT' INTRODUCTION OF KINEMATIC CONSTRAINTS;
W:=Z:=S:=0;
'FOR' I:=1 'STEP' 1 'UNTIL' NNODE-TN1 'DO'
```

```
'BEGIN'
'IF' I>W 'THEN'
              'BEGIN'  S:=S+1;
                       W:=NS[S+1]-1-N1[S];
                       Z:=Z+N1[S];
        'END';     Z:=Z+1;
'IF' KODE[Z,1]=0 'THEN' 'GOTO' KC1;
'IF' KODE[Z,1]=2 'THEN' 'GOTO'KC2;
GEOMBC(ULX[Z,1],2*I-1,Q,K,U+ELDOF*NTIP,1,ADD);
'FOR' J:=2 'STEP' 1 'UNTIL' NSETF 'DO' Q[I,J]:=Q[I,1];
'IF' KODE[Z,1]=1 'THEN' 'GOTO' KC1;
KC2; GEOMBC(VLY[Z,1],2*I,Q,K,U+ELDOF*NTIP,1,ADD);
'FOR' J:=2 'STEP' 1 'UNTIL' NSETF 'DO' Q[I,J]:=Q[I,1];
KC1;  'END';
 V:=0;
PUTARRAY(12,K); PUTARRAY(10,Q);
 RESOLVE;
  'IF' V 'NE' 0 'THEN' 'BEGIN'
   REWIND(12); PUTARRAY(12,K);
   REWIND(10); PUTARRAY(10,Q);
               'END';
NEWLINE(2);
WRITETEXT('('SYMVBSOL%%BEGINS')');
SYMVBSOL(K,K,ADD,Q,U+ELDOF*NTIP,NSETF,FAIL);
NEWLINE(2);
'IF'SURNO > 0 'THEN''BEGIN'
 WRITETEXT('('CLOSURE%BEGINS**********')');
 CLOSURE(K,NOSK,ANGSK,ADD,Q,RESOLVE,BAND,NG,U,OVP);
'END';
NEWLINE(6);
'FOR' I:=1 'STEP' 1 'UNTIL' NSETF 'DO''BEGIN'
NEWLINE(4);
 WRITETEXT('('NODAL%DISPLACEMENTS%FOR%FORCE%SET')');
PRINT(I,2,0);
NEWLINE(3);SPACE(1);WRITETEXT('('NODE')');
SPACE(5);WRITETEXT('('X-DIRECTION')');SPACE(3);
WRITETEXT('('Y-DIRECTION')');SPACE(10);
WRITETEXT('('NODE')');SPACE(5);WRITETEXT('('X-DIRECTION')');
SPACE(8);WRITETEXT('('Y-DIRECTION')');
S:=V:=0;
'FOR'Z:=1'STEP'1'UNTIL'NTIP'DO'
'BEGIN'   V:=NS[Z+1]-NS[Z]-N1[Z];
   'FOR'J:=2'STEP'2'UNTIL'V'DO'
   'BEGIN'   U:=J+NS[Z]-2+N1[Z];
             S:=S+4;
          NEWLINE(2);
          PRINT(U,3,0); SPACE(2); PRINT(Q[S-3,I],0,8);
          SPACE(2); PRINT(Q[S-2,I],0,3); SPACE(13);
          PRINT(U+1,3,0); SPACE(2); PRINT(Q[S-1,I],0,8);
          SPACE(2); PRINT(Q[S,I],0,8);
   'END'J;
'IF'J-2 < V 'THEN'
   'BEGIN' NEWLINE(2);   S:=S+2;
          PRINT(U+2,3,0); SPACE(2); PRINT(Q[S-1,I],0,8);
          SPACE(2); PRINT(Q[S,I],0,3);
```

```
  'END';
'END'Z;
U:=NG-2*TN1; NEWLINE(6);
'FOR'J:=1'STEP'1'UNTIL'NTIP'DO'
'BEGIN'    V:=ELDOF*(J-1);
NEWLINE(8);  SPACE(20);
WRITETEXT('('CRACK%TIP%NUMBER%%**********')');
PRINT(J,3,0);
NEWLINE(2);  SPACE(13);
WRITETEXT('('CRACK%TIP%DISPLACEMENT%IN%X-DIRECTION%=%')');
PRINT(QLU+2+V,1],0,10);
NEWLINE(2);  SPACE(13);
WRITETEXT('('CRACK%TIP%DISPLACEMENT%IN%Y-DIRECTION%=%')');
PRINT(QLU+3+V,1],0,10);
NEWLINE(2);  SPACE(18);
WRITETEXT('('MODE1%STRESS%INTENSITY%FACTOR%KI%=%')');
PRINT(QLU+4+V,1],0,10);
NEWLINE(2);  SPACE(17);
WRITETEXT('('MODE2%STRESS%INTENSITY%FACTOR%KII%=%')');
PRINT(QLU+5+V,1],0,10);
NEWLINE(2);  SPACE(29);
WRITETEXT('('RIGID%BODY%ROTATION%U%=%')');
PRINT(QLU+1+V,1],0,10);
'END';
'END';
'END';
'END';
'END';
'GOTO'FINE;
FAIL; NEWLINE(4);
      WRITETEXT('('PROGRAM%FAILED%IN%ROUTINE%SYNVBSOL')');
NEWLINE(2);
WRITETEXT('('FAILURE%DUE%TO%:-')'); NEWLINE(1); SPACE(17);
WRITETEXT('('UNREAL%PROBLEM%I.E.%WRONG%BOUNDARY%CONDITIONS')');
NEWLINE(1); SPACE(17);WRITETEXT('('ELEMENTS%OVERLAPPING')');
NEWLINE(2);
WRITETEXT('('SEE%DUMP%----%DUE%TO%DIVISION%BY%ZERO')');
  V:=0; U:=1; V:=U/V;
'GOTO'EXIT;
FINE:'END';
NEWLINE(4);
WRITETEXT('('END%OF%PROGRAM%-%NORMAL%EXIT')');
EXIT:'END' OF PROGRAM;
```

(vi) Program PROPAG has been described in Chapter seven and therefore only a formal program listing is given overleaf.

```
DOCUMENT     PROPAG


'BEGIN''COMMENT'PROGRAM FOR DETERMINING THE DIRECTION OF
       CRACK PROPAGATION & STRAIN ENERGY DENSITY FACTOR
       DATA INPUT:-  JOB TITLE ENDING WITH A 'SEMI-COLON'
                     NUMBER OF CRACK TIPS.
                            DATA FOR EACH TIP:-
                             KI,KII,
                             MU- POISSONS RATIO,
                             G - SHEAR MODULUS.
                                                ;
'INTEGER'IFAIL,I,V,J;
'REAL'E,B,C,D,MU,G,K1,K2,CON,TH,SS;
'PROCEDURE'E04AAA(X,F,ABSACC,RELACC,XSTEP,FUNCT,MAXFUN,IFAIL);
'VALUE'ABSACC,RELACC,XSTEP,MAXFUN;
'REAL'X,F,ABSACC,RELACC,XSTEP;
'INTEGER'IFAIL,MAXFUN;
'PROCEDURE'FUNCT; 'ALGOL';

'PROCEDURE'FUNCT(TH,SS);
'VALUE'TH; 'REAL'TH,SS;
'BEGIN''REAL'A1,A2,A3;
   A1:=(3-4*MU-COS(TH))*(1+COS(TH));
   A2:=(2*SIN(TH))*(COS(TH)-1+2*MU);
   A3:=(4-4*MU)*(1-COS(TH))*(1+COS(TH))*(3*COS(TH)-1);
   SS:=CON*((A1*(K1↑2))+(2*A2*K1*K2)+(A3*(K2↑2)));
'END' OF PROCEDURE FUNCT;

NEWLINE(8);
COPYTEXT('('';')');  NEWLINE(5);
V:=READ;
WRITETEXT('('CRACK%NO%%%%%ANGLE%OF%PROPAGATION')');
WRITETEXT('('%%%%%STRAIN%ENERGY%DENSITY%FACTOR')');
WRITETEXT('('%%%%%%%%%%KI%%%%%%%%%%KII')');
'FOR'I:=1'STEP'1'UNTIL'V'DO'
'BEGIN'
K1:=READ;   K2:=READ;   MU:=READ;   G:=READ;
CON:=1/(16*G);    TH:=-1.5707963;   D:=150;   E:=1.0&-5;
B:=0.01745;   C:=3.1415926;   IFAIL:=1;
E04AAA(TH,SS,B,F,C,FUNCT,D,IFAIL);
'IF' IFAIL>0 'THEN''BEGIN'
NEWLINE(5);
WRITETEXT('('NAGS%ROUTINE%E04AAA%HAS%FAILED')');
NEWLINE(2);
WRITETEXT('('IFAIL%=%')'); PRINT(IFAIL,2,0);
'END'   'ELSE'   'BEGIN'
TH:=(TH-C)*57.29578;
NEWLINE(2);  PRINT(I,3,0);
SPACE(10); PRINT(TH,0,5);  SPACE(13);  PRINT(SS,0,10);
SPACE(8);  PRINT(K1,0,5);  PRINT(K2,0,5);
'END';
'END' LOOP ;
PAPERTHROW;
'END' OF PROG PROPAG;
```

(vii) The auto-mesh generation program STDAMGhas been described in Chapter 5, and only a formal program listing is given overleaf.

```
DOCUMENT     STDAMG


'BEGIN'
'COMMENT' AUTO-MESH GENERATION PROGRAM. GENERATES DATA FOR
            6-NODE TRIANGULAR & 8-NODE QUADRILATERAL ISOPARAMETRIC
            ELEMENTS. SERVICES ALL THE 2D F.E. PROGRAMS AND THE
            GENERAL AXISYMMETRIC PROGRAM (IAAXMG) VIA A CODE NUMBER

'INTEGER'TNSPDS,NZ,NZONES,NNODE,NELET,HZONE,VZONE,CODE
        ,GH,CORP,QORT,NSPEC,CASE,N1,NNEWC,HND,
         ZN,ZONE,A,LZN,ADD,ELT,CON,ZNAD,FZN,B,I,J,O,N,REC,
         VD1,CU1,CU2,CU3,C1,C2,C3,C4,C5,C6,C7,NTIP,ALT,
         C,P,E,SUB,IDENT,D,DIFF,ELTAD,OLDA,SUM,VD,IN,TIK,
         ND,NNE,AA,B2,VD2,SET,SP,NSETFS,NMAT,NSKEW,W,BLANK,
         OLDTIK,SNODE,PZONE,NGH;
'REAL'DIAG1,DIAG2,ZETA,ETA,THICK,EE1,EE2,NU1,NU2,GE,ANG,RC,
      SCALE;
'INTEGER''ARRAY'BR[0:2],SIDE,IDN[1:20],NI,NS[1:5];
'REAL''ARRAY'NN,QUADX,QUADY[1:8],RO,AL[1:5];


'PROCEDURE' DECLARATIONS

  SELECT OUTPUT(4);
      OUTPUT1;
'BEGIN'
'REAL''ARRAY'XCOD,YCOD[1:SNODE];
'INTEGER''ARRAY'VARN[1:SNODE],MN[1:NZONES],
DIVX,DIVY[-VZONE:NZONES*VZONE],
 NODE[1:NELET,1:(7+QORT*2)];
   'FOR'I:=1'STEP'1'UNTIL'NZONES'DO'
      MN[I]:=0;
  MGINPUT(XCOD,YCOD,MN,DIVX,DIVY,VARN,NODE);
'BEGIN''INTEGER''ARRAY'NUM[0:SUM,1:2];
'REAL''ARRAY'XX,YY[1:NNODE+SUM];
'FOR'I:=1'STEP'1'UNTIL'SUM'DO''BEGIN'
     XX[NNODE+I]:=YY[NNODE+I]:=0.0;  'END';
D:=DIFF:=OLDA:=ELTAD:=0; AA:=1;
A:=LZN:=ADD:=ELT:=CON:=ZNAD:=DIVY[0]:=0;
'FOR'ZN:=1'STEP'1'UNTIL'NZONE'DO''BEGIN'
FZN:=(ZN-1)*VZONE+1;
LZN:=VZONE*ZN;
CU3:=DIFF:=TIK:=A:=ADD:=ELT:=0;
'FOR'I:=FZN'STEP'1'UNTIL'LZN  'DO''BEGIN'
'IF'MN[I]=0'THEN''BEGIN'
DIFF:=DIFF+DIVY[I-VZONE]*2;
'IF'I'NE'FZN'AND'I'NE'LZN'THEN''BEGIN'
'IF'MN[I]<MN[I-1]'THEN'CU3:=CU3+1;
      'IF'CU3=1'THEN''BEGIN'
'IF'MN[I]<MN[I+1]'THEN'CU3:=CU3+1;
'IF'CU3=2'THEN''BEGIN'
A:=A+1;DIFF:=DIFF-1;TIK:=TIK+1;
'IF'DIFF<0'THEN'DIFF:=0;CU3:=0;
'END';  'END';'END';'END';'END';
```

```
A:=A+2*DIVY[I];'END' OF I LOOP;
'IF'DIFF!NE!0'THEN'DIFF:=DIFF+VD2;
A:=A+1;
BLANK:=0;
'IF'A=1'THEN' BLANK:=VD2+QORT-OLDTIK;
VD2:=VD:=VD1:=IN:=0;
'FOR'ZONE:=FZN'STEP'1'UNTIL'LZN'DO''BEGIN'
'IF'ZONE=FZN'THEN'ADD:=0'ELSE'
ADD:=DIVY[ZONE-1]*2+ADD;
'IF'MN[ZONE]=0'THEN''GOTO'LOW;
N:=ELT:=0;
'COMMENT' QUADRILATERAL NODAL PTS GENERATED;
  ZONEXY(XCOD,YCOD,VARN);
ZETA:=ETA:=-1.0;  SET:=0;
 B:=0;
      'FOR'I:=0,1,2'DO'BB[I]:=0;
'COMMENT' XRY ELEMENT NODAL COORDS OBTAINED;
'IF'QORT=0'THEN'  TRELETXY(DIVX,DIVY,XX,YY)
        'ELSE'  QELETXY(TRACE,DIVX,DIVY,XX,YY);
'COMMENT' ELEMENT NODAL CONNECTIONS DETERMINED;
'IF' QORT = 0'THEN'  TRELNCONS(DIVX,DIVY,XX,YY,NODE,MN)
           'ELSE'   QELNCONS(DIVX,DIVY,XX,YY,NODE,MN);
'COMMENT' CLOSING SIDE NODE NUMBERS STORED;
'IF'IDENT=0'THEN''GOTO'L1;
'FOR'I:=1'STEP'1'UNTIL'IDENT'DO''BEGIN'
'IF'IDN[I]=ZONE'THEN'
'BEGIN'D:=D+1;
'IF' QORT=0'THEN'  TRIDN(DIVX,DIVY,NUM)
          'ELSE'  QIDN(TRACE,DIVX,DIVY,NUM);
'END';
'END' OF I LOOP;
'GOTO'L1;
LOW:VD1:=VD1+DIVY[ZONE+VZONE]*2;
VD:=VD+DIVY[ZONE-VZONE]*2;
'IF'TIK<1'THEN''GOTO'L1;
'IF'ZONE!NE!FZN'AND'ZONE'NE'LZN'THEN''BEGIN'
'IF'MN[ZONE]<MN[ZONE-1]'THEN''BEGIN'
      IN:=IN+1;  CU1:=CU2:=0;  'END';
          'IF'IN>0'THEN''BEGIN'
    'IF'DIVY[ZONE-VZONE]=0'THEN''BEGIN'   CU1:=CU1+1;
      'IF'CU1=1'THEN'VD:=VD+1;  'END';
    'IF'DIVY[ZONE+VZONE]=0'THEN''BEGIN'    CU2:=CU2+1;
      'IF'CU2=1'THEN''BEGIN'VD1:=VD1+1;    VD2:=VD2+1;'END';
'END';'END';'END';
L1:'END' OF ZONE LOOP;
 OLDA:=A; OLDTIK:=TIK;
 'FOR'I:=FZN'STEP'1'UNTIL'LZN'DO''BEGIN'
'IF'MN[I]>0'THEN''BEGIN'
REC:=DIVX[I]*(2-QORT);'GOTO'LIN99;
'END''ELSE' REC:=1;'END';
LIN99:'IF' QORT=0'THEN''BEGIN'
                ZNAD:=ZNAD+REC*OLDA+DIFF-BLANK;
                ELTAD:=ELTAD+(OLDA-1-TIK)*REC/2;
               'END'
          'ELSE''BEGIN'
```

```
  ZNAD:=ZNAD+REC*(OLDA+(OLDA+1+TIK)/2)+DIFF-BLANK;
ELTAD:=ELTAD+((OLDA-1-TIK)/2)*REC;
'END' OF ZN LOOP;
'END';
'COMMENT' ELEMENT NODAL CONNECTIONS ADJUSTED;
'IF' IDENT=0 'THEN' 'GOTO' L5;
COINCID(NUM,XX,YY);
    HATCHET(NUM,XX,YY,SUM,NNODE,NODE,NELET);
L5:OUTPUT2(MATCON,XX,YY,NODE);
SELECT OUTPUT(0);
RESPLOT(QPLOT,TRPLOT,XX,YY,NODE);
'END';
'END';
'END' OF PROCEDURE MESHGEN;
****
```

9.3.3 F.E.PROGRAM PROCEDURE LISTINGS

The following procedures are listed overleaf:-

General F.E. routines

AUX.

QAUX

GEOMBC

LOADAPP

SYMVBSOL

CMATRIX

SKEWDCON

SKEWLOAD

FEINPUT

FEASSEMBLY

QFEASSEMBLY

FENOSTR

FEELSTR

ADDARRAY

RESIDUAL

Multi-crack tip routines

MMNT

MMNTKBAND

CORK

CLOSURE

Segmented Solving routines

SEGSOL

TEST

ADDSEQ

GEOMBC

FEASSEMBLY

QFEASSEMBLY

Pseudo inverse routines

MIIST

SOLVIT

Modified IAAXMG routines

STRDIS

RZERO

```
'PROCEDURE' AUX(L1,L2,L3,B,X,Y,U,N,Z);
'VALUE' L1,L2,L3,Z;
'INTEGER' Z;
'REAL' L1,L2,L3,U;
'INTEGER' 'ARRAY' N;
'ARRAY' X,Y,B;
'BEGIN'
'INTEGER' I,V;
'REAL' CHANGE;
'REAL' 'ARRAY' P[1:2,1:6],J[1:2,1:2];
'COMMENT' THIS PROCEDURE EVALUATES THE JACOBIAN J ITS DETERMINANT U
          AND THE STRAIN-DISP ARRAY B;
J[1,1]:=X[N[Z,1]]*(4*L1-1)+X[N[Z,3]]*(4*L1+4*L2-3)+4*L2*X[N[Z,4]]-
4*L2*X[N[Z,5]]+4*X[N[Z,6]]*(1-2*L1-L2);
J[1,2]:=Y[N[Z,1]]*(4*L1-1)+Y[N[Z,3]]*(4*L1+4*L2-3)+4*L2*Y[N[Z,4]]-
4*L2*Y[N[Z,5]]+4*Y[N[Z,6]]*(1-2*L1-L2);
J[2,1]:=X[N[Z,2]]*(4*L2-1)+X[N[Z,3]]*(4*L1+4*L2-3)+4*L1*X[N[Z,4]]+
4*X[N[Z,5]]*(1-L1-2*L2)-4*L1*X[N[Z,6]];
J[2,2]:=Y[N[Z,2]]*(4*L2-1)+Y[N[Z,3]]*(4*L1+4*L2-3)+4*L1*Y[N[Z,4]]+
4*Y[N[Z,5]]*(1-L1-2*L2)-4*L1*Y[N[Z,6]];
'COMMENT' U REPLACES DETJ;
U:=J[1,1]*J[2,2]-J[1,2]*J[2,1];
'COMMENT' THE COEFFS OF [J] ARE REPLACED BY THOSE OF [J]-1;
CHANGE:=J[1,1];
J[1,1]:=J[2,2]/U;
J[1,2]:=-J[1,2]/U;
J[2,1]:=-J[2,1]/U;
J[2,2]:=CHANGE/U;
'COMMENT' THE P ARRAY REPS THE DIFF COEFFS OF N[I] WRT L1 AND L2;
'FOR' I:=1 'STEP' 1 'UNTIL' 6 'DO'
'FOR' V:=1,2 'DO'  P[V,I]:=0.0;
'FOR' I:=1,2 'DO'
'BEGIN' P[I,1]:=J[I,1]*(4*L1-1);
        P[I,2]:=J[I,2]*(4*L2-1);
        P[I,3]:=J[I,1]*(1-4*L3)+J[I,2]*(1-4*L3);
        P[I,4]:=4*(L2*J[I,1]+L1*J[I,2]);
        P[I,5]:=4*(J[I,2]*L3-L2*(J[I,2]+J[I,1]));
        P[I,6]:=4*(J[I,1]*L3-L1*(J[I,1]+J[I,2]));
'END';
'FOR' I:=1 'STEP' 1 'UNTIL' 12 'DO'
'FOR' V:=1,2,3 'DO' B[V,I]:=0.0;
'FOR' I:=1 'STEP' 1 'UNTIL' 6 'DO'
'BEGIN'
       B[1,(I*2-1)]:=B[3,(I*2)]:=P[1,I];
        B[3,(I*2-1)]:=B[2,(I*2)]:=P[2,I];
'END';
'END' OF AUX;
```

```
'PROCEDURE' QAUX(ETA,ZETA,B,X,Y,U,N,Z);
'VALUE'ETA,ZETA,Z;
'INTEGER'Z;
'REAL'ETA,ZETA,U;
'INTEGER''ARRAY'N;
'ARRAY'X,Y,B;
'BEGIN'
'INTEGER'I,V;
'REAL'CHANGE;
'REAL''ARRAY'P[1:2,1:8],J[1:2,1:2];
'COMMENT' THIS PROCEDURE EVALUATES THE JACOBIAN J, ITS
          DETERMINANT U, AND THE STRAIN DISP ARRAY B = FOR
          A QUADRILATERAL 'ISOPARAMETRIC' 8-NODE ELEMENT;
'COMMENT' U REPLACES DETJ;
P[1,1]:=(1-ZETA)*(2*ETA+ZETA)/4;
P[1,2]:=(1-ZETA)*(2*ETA-ZETA)/4;
P[1,3]:=(1+ZETA)*(2*ETA+ZETA)/4;
P[1,4]:=(1+ZETA)*(2*ETA-ZETA)/4;
P[1,5]:=-ETA*(1-ZETA);
P[1,6]:=(1-ZETA↑2)/2;
P[1,7]:=-ETA*(1+ZETA);
P[1,8]:=-(1-ZETA↑2)/2;
P[2,1]:=(1-ETA)*(2*ZETA+ETA)/4;
P[2,2]:=(1+ETA)*(2*ZETA-ETA)/4;
P[2,3]:=(1+ETA)*(2*ZETA+ETA)/4;
P[2,4]:=(1-ETA)*(2*ZETA-ETA)/4;
P[2,5]:=-(1-ETA↑2)/2;
P[2,6]:=-ZETA*(1+ETA);
P[2,7]:=(1-ETA↑2)/2;
P[2,8]:=-ZETA*(1-ETA);
J[1,1]:=J[1,2]:=J[2,1]:=J[2,2]:=0.0;
'FOR'I:=1'STEP'1'UNTIL'8'DO'
'BEGIN'
J[1,1]:=J[1,1]+P[1,I]*X[N[Z,I]];
J[1,2]:=J[1,2]+P[1,I]*Y[N[Z,I]];
J[2,1]:=J[2,1]+P[2,I]*X[N[Z,I]];
J[2,2]:=J[2,2]+P[2,I]*Y[N[Z,I]];
'END';
U:=J[1,1]*J[2,2]-J[1,2]*J[2,1];
'COMMENT' THE COEFFS OF [J] ARE REPLACED BY THOSE OF [J] INVERSE;
CHANGE:=J[1,1];
J[1,1]:=J[2,2]/U;
J[1,2]:=-J[1,2]/U;
J[2,1]:=-J[2,1]/U;
J[2,2]:=CHANGE/U;
'COMMENT' THE B ARRAY REPRESENTS THE DIFF COEFFS OF N[I] WITH RESPE
          TO ETA & ZETA;
'FOR'I:=1'STEP'1'UNTIL'16'DO'
'FOR'V:=1,2,3'DO' B[V,I]:=0.0;
'FOR'I:=1'STEP'1'UNTIL'8'DO'
'BEGIN'
B[1,(I*2-1)]:=B[3,(I*2)]:=J[1,1]*P[1,I]+J[1,2]*P[2,I];
B[3,(I*2-1)]:=B[2,(I*2)]:=J[2,1]*P[1,I]+J[2,2]*P[2,I];
'END';
'END' OF PROCEDURE QAUX;
```

```
'PROCEDURE' GEOMBC(U,N,R,AK,NEQ,F,A);
'VALUE' U,N,NEQ,F;
'REAL' U;
'INTEGER' N,NEQ,F;
'ARRAY' R,AK;
'INTEGER' 'ARRAY' A;
'BEGIN' 'INTEGER' M,K,CJ;
'IF' N=1 'THEN' CJ:=1 'ELSE' CJ:=N-(A[N]-A[N-1])+1;
'FOR' K:=CJ 'STEP' 1 'UNTIL' N 'DO'
'BEGIN'
        R[K,F]:=R[K,F]-AK[A[N]-N+K]*U;
        AK[A[N]-N+K]:=0.0;
'END';
'IF' N+1 'GT' NEQ 'THEN' 'GOTO' LZ;
'FOR' K:=N+1 'STEP' 1 'UNTIL' NEQ 'DO'
'BEGIN' CJ:=K-(A[K]-A[K-1])+1;
        'IF' CJ 'LE' N 'THEN'
        'BEGIN'
                R[K,F]:=R[K,F]-AK[A[K]-K+N]*U;
                AK[A[K]-K+N]:=0.0;
        'END' 'ELSE'
'END';
LZ: AK[A[N]]:=1.0;
    R[N,F]:=U;
'END';
```

```
'PROCEDURE' LOADAPP(A,B,C,D,E,F);
'VALUE' A,B,C,E,F;
'INTEGER' A,E,F;
'REAL' B,C;
'ARRAY' D;
'BEGIN'
'INTEGER' K;
'IF' A=3 'THEN' 'GOTO' KLAB1;
K:=2*E;
'IF' A=1 'THEN' 'GOTO' KLAB2;
  D[K-1,F]:=D[K-1,F]+B;
'IF' A 'NE' 0 'THEN' 'GOTO' KLAB1;
KLAB2:  D[K,F]:=D[K,F]+C;
KLAB1:  'END' OF LOADAPP;
```

```
'PROCEDURE' SYMVBSOL(A,L,S,B)DIMENSIONS:(N,R)FAILURE EXIT:(FAIL);
'VALUE' N,R; 'ARRAY' A,L,B; 'INTEGER' 'ARRAY' S; 'INTEGER' N,R;
'LABEL' FAIL;
'BEGIN'
'INTEGER' G,H,I,J,K,M,P,Q,T,U,V;
'REAL' Y;
H:=0;
'FOR' I:=1 'STEP' 1 'UNTIL' N 'DO'
  'BEGIN'
  T:=I+H-S[I]+1; G:=H+1;
  P:=S[T]-1;
  'FOR' J:=T 'STEP' 1 'UNTIL' I-1 'DO'
    'BEGIN'
    Q:=P+1; H:=H+1;
    P:=S[J]; K:=J+Q-P;
    V:=H-P; U:=G;
    Y:=A[H];
    'IF' K 'GT' T 'THEN' U:=U+K-T;
    'FOR' U:=U 'STEP' 1 'UNTIL' H-1 'DO'
    Y:=Y-L[U]*L[U-V];
    Y:=Y/L[H-V]; L[H]:=Y;
    'FOR' M:=1 'STEP' 1 'UNTIL' R 'DO'
    B[I,M]:=B[I,M]-B[J,M]*Y;
    'END' J;
  Y:=A[H+1];
  'FOR' U:=G 'STEP' 1 'UNTIL' H 'DO'
  Y:=Y-L[U]↑2;
  H:=H+1; Y:=SQRT(Y);
  L[H]:=Y;
  'FOR' M:=1 'STEP' 1 'UNTIL' R 'DO'
  B[I,M]:=B[I,M]/Y
  'END' I;
'COMMENT' REDUCTION COMPLETE;
'FOR' I:=N 'STEP' -1 'UNTIL' 1 'DO'
  'BEGIN'
  Y:=L[H];
  'FOR' M:=1 'STEP' 1 'UNTIL' R 'DO'
  B[I,M]:=B[I,M]/Y;
  'IF' I=1 'THEN' 'GOTO' COMPLETE;
  J:=I; P:=S[I-1];
  'FOR' H:=H-1 'STEP' -1 'UNTIL' P+1 'DO'
    'BEGIN'
    J:=J-1; Y:=L[H];
    'FOR' M:=1 'STEP' 1 'UNTIL' R 'DO'
    B[J,M]:=B[J,M]-B[I,M]*Y;
    'END' H;
  H:=P;
COMPLETE; 'END' I;
'END' SYMVBSOL;
```

```
'PROCEDURE' MATRIX(Z,CASE,A,MATNO,TH,ANG);
'VALUE'MATNO;'INTEGER' MATNO,CASE;'REAL'ANG;'ARRAY'A,Z,TH;
'BEGIN' 'INTEGER'I;'REAL'S,C,S2,C2,S3,C3,S4,C4,C11,C12,C22,C23,C33,C
'COMMENT' CALCULATION OF ELASTIC CONSTANTS WHERE SEVERAL MATERIALS
          ARE PRESENT EACH WITH VARYING ANGLES OF ANISOTROPY;
CASE:=READ; ANG:=READ; TH[MATNO]:=READ;
'FOR' I:=1 'STEP' 1 'UNTIL' 5 'DO' A[I]:=READ;
'IF' MATNO=1 'THEN'
'BEGIN' NEWLINE(4);
WRITETEXT('('MATERIAL%PROPERTIES:=')'); NEWLINE(1);
WRITETEXT('('(NOTE,IF%CASE=0,THEN%PLANE%STRESS%ELSE')');
NEWLINE(1); SPACE(6);
WRITETEXT('('THE%PROBLEM%IS%ONE%OF%PLANE%STRAIN.)')');
NEWLINE(2);
WRITETEXT('('MATERIAL%%%%%%ANGLE%OF%%%%%%CASE%%%%THICKNESS')');
SPACE(21);
WRITETEXT('('ELASTIC%%PROPERTIES')'); NEWLINE(1); SPACE(1);
WRITETEXT('('NUMBER%%%%%%ANISOTROPY')'); SPACE(30);
WRITETEXT('('EXX')'); SPACE(11);
WRITETEXT('('EYY')'); SPACE(9);
WRITETEXT('('VXY%%%%%%%VY%%%%%%%%%%GXY')');
'END'; NEWLINE(2); SPACE(1);
PRINT(MATNO,2,0); SPACE(7); PRINT(ANG,0,2); SPACE(3);PRINT(CASE,1
SPACE(3);PRINT(TH[MATNO],0,4);SPACE(2);PRINT(A[1],0,3);SPACE(2);
PRINT(A[4],0,3);SPACE(2);PRINT(A[2],1,2);SPACE(2);PRINT(A[5],1,2);
SPACE(2);PRINT(A[3],0,3);
ANG:=0.0174533*ANG; C23:=C13:=C33:=0.0;
'IF' CASE=0 'THEN'
'BEGIN' C11:=A[1]/(1-(A[4]/A[1])*A[2]*A[2]);
C12:=A[4]*A[2]/(1-(A[4]/A[1])*A[2]*A[2]);
C22:=A[4]/(1-(A[4]/A[1])*A[2]*A[2]);
C33:=A[5];
'END' 'ELSE'
'BEGIN'C33:=A[3];
A[5]:=A[1]/((1+A[5])*(1-A[5]-2*(A[4]/A[1])*A[2]*A[2]));
C11:=A[5]*(1-A[5]*A[5]);
C12:=A[5]*(A[4]/A[1])*A[2]*(1+A[5]);
C22:=(A[4]/A[1])*(1-(A[4]/A[1])*A[2]*A[2])*A[3];
'END';
'IF' ANG 'NE' 0.0 'THEN'
'BEGIN' S:=SIN(ANG);C:=COS(ANG);S2:=S*S;C2:=C*C;S3:=S2*S;C3:=C2*C;
S4:=S2*S2; C4:=C2*C2;
Z[MATNO,1]:=C11*C4+2*(C12+2*C33)*C2*S2-4*(C13*C2+C23*S2)*C*S+C22*S4
Z[MATNO,2]:=(S4+C4)*C12+(C11+C22-4*C33)*C2*S2+2*(C13-C23)*C3*S+2*(C
-C13)*C*S3;
Z[MATNO,3]:=(C11-C12-2*C33)*C3*S+(C12-C22+2*C33)*C*S3+3*(C23-C13)*S
+C13*C4-C23*S4;
Z[MATNO,4]:=C11*S4+C22*C4+4*C13*C*S3+4*C23*C3*S+2*(C12+2*C33)*C2*S2
Z[MATNO,5]:=(C11-C12-2*C33)*S3*C+(C12-C22+2*C33)*C3*S+3*(C13-C23)*C
+C23*C4-C13*S4;
Z[MATNO,6]:=(C11+C22-2*C12-2*C33)*S2*C2+2*(C13-C23)*C3*S+2*(C23-C13
C+C33*(S4+C4);
'END' 'ELSE'
'BEGIN' Z[MATNO,1]:=C11; Z[MATNO,2]:=C12;
Z[MATNO,3]:=C13; Z[MATNO,4]:=C22;
Z[MATNO,5]:=C23; Z[MATNO,6]:=C33;
'END';
'END' OF PROCEDURE MATRIX;
```

```
'PROCEDURE' SKEWEDCON(Z,NODE,ANGSK,NOSK,NSKEW,KE,NSETF);
'VALUE' Z,NSKEW,NSETF;  'INTEGER' Z,NSKEW,NSETF;
'INTEGER' 'ARRAY' NODE,NOSK; 'REAL' 'ARRAY' ANGSK,KE;
'BEGIN' 'INTEGER' I,J,V,K,L;  'REAL' M,N,O,P,Q1,R,S1,T;
'REAL' 'ARRAY' S[1:12+4*QORT];
'FOR' I:=1 'STEP' 2 'UNTIL' 11+4*QORT 'DO' 'BEGIN'
 S[I]:=1; S[I+1]:=0; 'END';
V:=0;
'FOR' I:=1 'STEP' 1 'UNTIL' 6+2*QORT 'DO'
'BEGIN' 'FOR' J:=1 'STEP' 1 'UNTIL' NSKEW 'DO'
'IF' NODE[Z,I]=NOSK[J] 'THEN'
'BEGIN' S[2*I-1]:=COS(ANGSK[J]*0.0174533);
        S[2*I]:=SIN(ANGSK[J]*0.0174533);
V:=V+1;  'GOTO' SKLAB;
'END';
SKLAB: 'END';
'IF' V 'NL' 0 'THEN'
'BEGIN' 'FOR' I:=1 'STEP' 1 'UNTIL' 6+2*QORT 'DO'
'BEGIN'  K:=2*I;
'FOR' J:=1 'STEP' 1 'UNTIL' I 'DO'
'BEGIN'   L:=2*J;
M:=KE[K-1,L-1];  N:=KE[K-1,L];
O:=KE[K,L-1];  P:=KE[K,L];
Q1:=S[K-1]*N+S[K]*O;  R:=S[K-1]*N+S[K]*P;
S1:=-S[K]*M+S[K-1]*O;  T:=-S[K]*N+S[K-1]*P;
KE[K-1,L-1]:=KE[L-1,K-1]:=Q1*S[L-1]+R*S[L];
KE[K-1,L]:=KE[L,K-1]:=-Q1*S[L]+R*S[L-1];
KE[K,L-1]:=KE[L-1,K]:=S1*S[L-1]+T*S[L];
KE[K,L]:=KE[L,K]:=-S1*S[L]+T*S[L-1];
'END';
'END';
'END';
'END' OF PROCEDURE SKEWEDCON;



'PROCEDURE' SKEWLOAD(NSETF,Q,NOSK,ANGSK,NSKEW,OP);
'VALUE' NSETF,NSKEW,OP;  'INTEGER' NSETF,NSKEW,OP;
'INTEGER' 'ARRAY' NOSK;  'REAL' 'ARRAY' Q,ANGSK;
'BEGIN' 'INTEGER' I,J;  'REAL' SN,CN,M,N;
'FOR' I:=1 'STEP' 1 'UNTIL' NSKEW 'DO'
'BEGIN' SN:=SIN(ANGSK[I]*0.0174533);
        CN:=COS(ANGSK[I]*0.0174533);
  'FOR' J:=1 'STEP' 1 'UNTIL' NSETF 'DO'
   'BEGIN' M:=Q[2*NOSK[I]-1,J];
           N:=Q[2*NOSK[I],J];
           Q[2*NOSK[I]-1,J]:=CN*M+OP*SN*N;
           Q[2*NOSK[I],J]:=-OP*SN*M+CN*N;
'END';
'END';
'END' OF PROCEDURE SKEWLOAD;
```

```
'PROCEDURE' FEINPUT(C,ADD,XX,YY,NFREE,NNODE,NSETF,KODE,NSPEC,NSETFS
                    ULX,VLY,NELEMT,NODE,NMAT);
'VALUE' NFREE,NNODE,NSETF,NSPEC,NELEMT,NSETFS,NMAT;
'INTEGER' NFREE,NNODE,NSETF,NSPEC,NELEMT,NSETFS,NMAT;
'INTEGER' 'ARRAY' ADD,KODE,NODE;  'REAL' 'ARRAY' XX,YY,ULX,VLY,C;
'BEGIN' 'INTEGER' I,J,W;
'FOR' I:=0 'STEP' 2 'UNTIL' NFREE 'DO' ADD[I]:=0;
'FOR' I:=1 'STEP' 1 'UNTIL' NNODE 'DO'
'FOR' J:=1 'STEP' 1 'UNTIL' NSETFS'DO'  KODE[I,J]:=0;
'FOR' I:=1 'STEP' 1 'UNTIL' NNODE 'DO'
'FOR' J:=1 'STEP' 1 'UNTIL' NSETFS 'DO'
ULX[I,JJ:=VLY[I,J]:=0.0;
'FOR' I:=1 'STEP' 1 'UNTIL' NMAT 'DO'
'FOR' J:=1 'STEP' 1 'UNTIL' 6 'DO'C[I,J]:=0.0;
'FOR'I:=1'STEP'1'UNTIL'NNODE'DO''BEGIN'
   XX[I]:=READ;YY[I]:=READ;
'END';
'FOR'I:=1'STEP'1'UNTIL'NELEMT'DO''BEGIN'
'FOR'J:=1'STEP'1'UNTIL'7+2*NORT'DO'
NODE[I,J]:=READ;
'END';
NSPEC:=READ;
'FOR' I:=1 'STEP' 1 'UNTIL' NSPEC 'DO'
'BEGIN'
    J:=READ;      KODE[J,1]:=READ;  ULX[J,1]:=READ;  VLY[J,1]:=READ;
'END';
'COMMENT' THE NODE NUMBER MUST BE SPECIFIED ALONG WITH
          THE COORDS OF THE NODE;
NEWLINE(4);WRITETEXT('('NODAL%POINT%DATA')');
NEWLINE(2);SPACE(4);WRITETEXT('('NODE')');SPACE(5);
WRITETEXT('('X%COORD')'); SPACE(5);WRITETEXT('('Y%COORD')');
SPACE(5);WRITETEXT('('TYPE')');SPACE(7);
WRITETEXT('('X-DISP')');SPACE(6);WRITETEXT('('Y-DISP')');
NEWLINE(1);SPACE(48);WRITETEXT('('OR%LOAD')');
SPACE(5); WRITETEXT('('OR%LOAD')');
'FOR' I:=1 'STEP' 1 'UNTIL' NNODE 'DO'
'BEGIN'  NEWLINE(1); SPACE(3);
         PRINT(I,3,0);  SPACE(3);
         PRINT(XX[I],0,3);
         PRINT(YY[I],0,3);  SPACE(2);
         PRINT(KODE[I,1],3,0);  SPACE(2);
         PRINT(ULX[I,1],0,3);
         PRINT(VLY[I,1],0,3);
'END';
NEWLINE(4);WRITETEXT('('ELEMENT%DATA')');NEWLINE(2);
WRITETEXT('('ELEMENT')');SPACE(18+11*NORT);
WRITETEXT('('NODAL%CONNECTIONS')');
SPACE(17+5*NORT); WRITETEXT('('MATERIAL')');
'FOR' W:=1 'STEP' 1 'UNTIL' NELEMT 'DO'
'BEGIN'  NEWLINE(1);
PRINT(W,3,0); SPACE(6);
'FOR' J:=1 'STEP' 1 'UNTIL' 7+2*NORT 'DO'
'BEGIN'  PRINT(NODE[W,J],3,0);
SPACE(2);
'END';
'END';
'END' OF PROCEDURE FEINPUT;
```

```
'PROCEDURE' FEASSEMBLY(NELEMT,K,XX,YY,DETJ,NODE,C,TH,ADD,NFREE,
                       AUX,SKEWEDCOM,NSKEW,NSETF,ANGSK,NOSK);
'VALUE' NELEMT,NFREE,NSKEW,NSETF;
'INTEGER' NELEMT,NFREE,NSKEW,NSETF; 'REAL' DETJ;
'REAL''ARRAY'K,XX,YY,C,TH,ANGSK;'INTEGER' 'ARRAY' NODE,ADD,NOSK;
'PROCEDURE' AUX,SKEWEDCOM;
'BEGIN' 'INTEGER' I,J,U,Z,V,SUB1,SUB2,SUB3;
'REAL' 'ARRAY' B[1:3,1:12],KE[1:12,1:12],W[1:6,1:4];
'FOR' I:=1 'STEP' 1 'UNTIL' 6 'DO' W[I,1]:=0.33333333;
W[1,2]:=W[1,3]:=W[2,3]:=W[2,4]:=W[3,2]:=W[3,4]:=0.5;
W[1,4]:=W[2,2]:=W[3,3]:=W[4,4]:=W[5,2]:=W[6,3]:=0.0;
W[4,2]:=W[4,3]:=W[5,3]:=W[5,4]:=W[6,2]:=W[6,4]:=1.0;
'FOR' I:=1 'STEP' 1 'UNTIL' ADD[NFREE] 'DO' K[I]:=0.0;
'FOR' Z:=1 'STEP' 1 'UNTIL' NELEMT 'DO'
'BEGIN'
'FOR'I:=1 'STEP' 1 'UNTIL' 12 'DO'
'FOR' J:=1 'STEP' 1 'UNTIL' 12 'DO' KE[I,J]:=0.0;
'COMMENT' THE LOOP FOR THE NUMBER OF INT PTS IS CONSTRUCTED;
'FOR' U:=1 'STEP' 1 'UNTIL' 3 'DO'
'BEGIN'
AUX(W[U,2],W[U,3],W[U,4],B,XX,YY,DETJ,NODE,Z);
      'FOR' J:=1 'STEP' 2 'UNTIL' 11 'DO'
      'FOR' I:=J 'STEP' 2 'UNTIL' 11 'DO'
      'BEGIN'
KE[J,I]:=KE[I,J]:=KE[I,J]+W[U,1]*(B[1,J]*(C[NODE[Z,7],1]*B[1,I]+
C[NODE[Z,7],3]*B[3,I])+B[3,J]*(C[NODE[Z,7],3]*B[1,I]+C[NODE[Z,7],6]*
B[3,I]))*DETJ*0.5*TH[NODE[Z,7]];
KE[J,I+1]:=KE[I+1,J]:=KE[I+1,J]+W[U,1]*(B[1,J]*(C[NODE[Z,7],2]*
B[2,I+1]+C[NODE[Z,7],3]*B[3,I+1])+B[3,J]*(C[NODE[Z,7],5]*B[2,I+1]+
C[NODE[Z,7],6]*B[3,I+1]))*DETJ*0.5*TH[NODE[Z,7]];
      'END';
      'FOR' J:=2 'STEP' 2 'UNTIL' 12 'DO'
      'FOR' I:=J 'STEP' 2 'UNTIL' 12 'DO'
      'BEGIN'
KE[J,I]:=KE[I,J]:=KE[I,J]+W[U,1]*(B[2,J]*(C[NODE[Z,7],4]*B[2,I]+
C[NODE[Z,7],5]*B[3,I])+B[3,J]*(C[NODE[Z,7],5]*B[2,I]+C[NODE[Z,7],6]*
B[3,I]))*DETJ*0.5*TH[NODE[Z,7]];
      'IF' I=12 'THEN' 'GOTO' L6 'ELSE'
KE[J,I+1]:=KE[I+1,J]:=KE[I+1,J]+W[U,1]*(B[2,J]*(C[NODE[Z,7],2]*
B[1,I+1]+C[NODE[Z,7],5]*B[3,I+1])+B[3,J]*(C[NODE[Z,7],3]*B[1,I+1]+
C[NODE[Z,7],6]*B[3,I+1]))*DETJ*0.5*TH[NODE[Z,7]];
   L6: 'END';
'END';
'IF'NSKEW'NE'0'THEN'SKEWEDCOM(Z,NODE,ANGSK,NOSK,NSKEW,KE,NSETF);
'COMMENT' ASSEMBLY OF OVERALL STIFFNESS MATRIX AS A
          ONE-DIMENSIONAL ARRAY;
'FOR' I:=1 'STEP' 1 'UNTIL' 6 'DO'
'FOR' J:=1 'STEP' 1 'UNTIL' 6 'DO'
'FOR' V:=1,0 'DO'
'BEGIN'  SUB1:=NODE[Z,I]*2-1;
         SUB2:=NODE[Z,J]*2-V;
         SUB3:=NODE[Z,I]*2;
'IF' SUB1 'LT' SUB2 'THEN' 'GOTO' LABA;
K[ADD[SUB1]-SUB1+SUB2]:=K[ADD[SUB1]-SUB1+SUB2]+KE[I*2-1,J*2-V];
LABA; 'IF' SUB3 'LT' SUB2 'THEN' 'GOTO' LABB;
K[ADD[SUB3]-SUB3+SUB2]:=K[ADD[SUB3]-SUB3+SUB2]+KE[I*2,J*2-V];
LABB; 'END';
'END';
'END' OF PROCEDURE FEASSEMBLY;
```

```
'PROCEDURE' QFEASSEMBLY(NELEMT,K,XX,YY,DETJ,NODE,C,TH,ADD,NFREE,
                        QAUX,SKEWEDCOD,NSKEW,NSETF,ANGSK,DOSK);
'VALUE' NELEMT,NFREE,NSKEW,NSETF;
'INTEGER' NELEMT,NFREE,NSKEW,NSETF; 'REAL' DETJ;
'REAL''ARRAY'K,XX,YY,C,TH,ANGSK;'INTEGER' 'ARRAY' NODE,ADD,DOSK;
'PROCEDURE' QAUX,SKEWEDCOD;
'BEGIN' 'INTEGER' I,J,U,Z,V,SUB1,SUB2,SUB3;
'REAL' 'ARRAY' B[1:3,1:16],KE[1:16,1:16],W[1:4,1:4];
'FOR'I:=1'STEP'1'UNTIL'4'DO''BEGIN'W[I,4]:=W[I,3]:=1.0;
                  'END';
W[1,1]:=W[1,2]:=W[2,2]:=W[3,1]:=-0.577350269189;
W[2,1]:=W[3,2]:=W[4,1]:=W[4,2]:=0.577350269189;
'FOR' I:=1 'STEP' 1 'UNTIL' ADD[NFREE] 'DO' K[I]:=0.0;
'FOR' Z:=1 'STEP' 1 'UNTIL' NELEMT 'DO'
'BEGIN'
'FOR'I:=1 'STEP' 1 'UNTIL' 16 'DO'
'FOR' J:=1 'STEP' 1 'UNTIL' 16 'DO' KE[I,J]:=0.0;
'COMMENT' THE LOOP FOR THE NUMBER OF INT PTS IS CONSTRUCTED;
'FOR' U:=1 'STEP' 1 'UNTIL' 4 'DO'
'BEGIN'
QAUX(W[U,1],W[U,2],B,XX,YY,DETJ,NODE,Z);
      'FOR' J:=1 'STEP' 2 'UNTIL' 15 'DO'
      'FOR' I:=J 'STEP' 2 'UNTIL' 15 'DO'
      'BEGIN'
KE[J,I]:=KE[I,J]:=KE[I,J]+W[U,3]*W[U,4]*(B[1,J]*(C[NODE[Z,9],1]*B[1
C[NODE[Z,9],3]*B[3,I])+B[3,J]*(C[NODE[Z,9],3]*B[1,I]+C[NODE[Z,9],6]*
B[3,I]))*DETJ*TH[NODE[Z,9]];
KE[J,I+1]:=KE[I+1,J]:=KE[I+1,J]+W[U,3]*W[U,4]*(B[1,J]*(C[NODE[Z,9],
B[2,I+1]+C[NODE[Z,9],3]*B[3,I+1])+B[3,J]*(C[NODE[Z,9],5]*B[2,I+1]+
C[NODE[Z,9],6]*B[3,I+1]))*DETJ*TH[NODE[Z,9]];
      'END';
      'FOR' J:=2 'STEP' 2 'UNTIL' 16 'DO'
      'FOR' I:=J 'STEP' 2 'UNTIL' 16 'DO'
      'BEGIN'
KE[J,I]:=KE[I,J]:=KE[I,J]+W[U,3]*W[U,4]*(B[2,J]*(C[NODE[Z,9],4]*B[2
C[NODE[Z,9],5]*B[3,I])+B[3,J]*(C[NODE[Z,9],5]*B[2,I]+C[NODE[Z,9],6]*
B[3,I]))*DETJ*TH[NODE[Z,9]];
      'IF' I=16 'THEN' 'GOTO' L6 'ELSE'
KE[J,I+1]:=KE[I+1,J]:=KE[I+1,J]+W[U,3]*W[U,4]*(B[2,J]*(C[NODE[Z,9],
B[1,I+1]+C[NODE[Z,9],5]*B[3,I+1])+B[3,J]*(C[NODE[Z,9],3]*B[1,I+1]+
C[NODE[Z,9],6]*B[3,I+1]))*DETJ*TH[NODE[Z,9]];
   L6:  'END';
'END';
'IF'NSKEW'NE'0'THEN'SKEWEDCOD(Z,NODE,ANGSK,DOSK,NSKEW,KE,NSETF);
'COMMENT' ASSEMBLY OF OVERALL STIFFNESS MATRIX AS A
          ONE-DIMENSIONAL ARRAY;
'FOR' I:=1 'STEP' 1 'UNTIL' 8 'DO'
'FOR' J:=1 'STEP' 1 'UNTIL' 8 'DO'
'FOR' V:=1,0 'DO'
'BEGIN'  SUB1:=NODE[Z,I]*2-1;
         SUB2:=NODE[Z,J]*2-V;
         SUB3:=NODE[Z,I]*2;
'IF' SUB1 'LT' SUB2 'THEN' 'GOTO' LABA;
K[ADD[SUB1]+SUB1-SUB2]:=K[ADD[SUB1]+SUB1-SUB2]+KE[I*2-1,J*2-V];
LABA: 'IF' SUB3 'LT' SUB2 'THEN' 'GOTO' LABB;
K[ADD[SUB3]+SUB3-SUB2]:=K[ADD[SUB3]+SUB3-SUB2]+KE[I*2,J*2-V];
LABB: 'END';
'END';
'END' OF PROCEDURE QFEASSEMBLY;
```

```
'PROCEDURE' FENGSTR(NSETF,NNODE,NELEMT,NODE,XX,YY,DETJ,Q,TH,C,AUX,QAU
'VALUE' NSETF,NNODE,NELEMT; 'INTEGER' NSETF,NNODE,NELEMT;
'REAL' DETJ; 'INTEGER' 'ARRAY' NODE;
'REAL' 'ARRAY' XX,YY,Q,C,TH; 'PROCEDURE' AUX,QAUX;
'BEGIN' 'COMMENT' CALCULATION OF STRAINS AND STRESSES AT NODAL
                  POINTS BY AVERAGING CONTRIBUTIONS FROM
                  ADJACENT ELEMENTS;
'INTEGER' CNT1,CNT2,V,Z,U,J;
'REAL' SIGXX,SIGYY,SIGXY,EXX,EYY,EXY,SIG1,SIG2,SIG12,E1,E2,E12,PH,SET
'REAL' 'ARRAY' STR[1:3],W[1:6*2*QORT,1:4*2*QORT],B[1:3,1:12*4*QORT];
'IF'QORT=0'THEN''BEGIN'
'FOR'V:=1'STEP'1'UNTIL'6'DO'
'FOR'Z:=2'STEP'1'UNTIL'4'DO'W[V,Z]:=0.0;
W[1,2]:=W[2,3]:=W[3,4]:=1.0;
W[4,2]:=W[4,3]:=W[5,3]:=W[5,4]:=W[6,2]:=W[6,4]:=0.5;
'END''ELSE''BEGIN'
W[1,1]:=W[1,2]:=W[2,2]:=W[4,1]:=W[5,2]:=W[8,1]:=-1.0;
W[6,2]:=W[8,2]:=W[5,1]:=W[7,1]:=0.0;
W[2,1]:=W[3,1]:=W[3,2]:=W[4,2]:=W[6,1]:=W[7,2]:=1.0;
'END';
'FOR' V:=1 'STEP' 1 'UNTIL' NSETF 'DO'
'BEGIN'
'IF'PRINC=1'THEN''GOTO'LINE5;
 NEWLINE(5);SPACE(20);
WRITETEXT('('STRAINS')'); SPACE(43);
WRITETEXT('('STRESSES')'); NEWLINE(1);
WRITETEXT('('NODE%%%EXX')'); SPACE(10);
WRITETEXT('('EYY')'); SPACE(10);
WRITETEXT('('EXY')'); SPACE(21);
WRITETEXT('('SIGMA-X%%%%%%SIGMA-Y%%%%%%SIGMA XY')');
'GOTO'LINE6;
 LINE5: NEWLINE(5); SPACE(15);
WRITETEXT('('PRINCIPAL%STRAINS')'); SPACE(5);
WRITETEXT('('PRINCIPAL%STRESSES')'); NEWLINE(1);
WRITETEXT('('NODE%%E11%%%%%%%%E22')'); SPACE(10);
WRITETEXT('('E12')'); SPACE(21);
WRITETEXT('('SIGMA 1%%%%%%SIGMA 2%%%%%%SIGMA 12%%%%%%DIRECTION
%%%%%SIGMA EFF')');
LINE6:'FOR' U:=1 'STEP' 1 'UNTIL' NNODE 'DO'
'BEGIN' SIGXX:=SIGYY:=SIGXY:=EXX:=EYY:=EXY:=CNT2:=0.0;
'FOR' Z:=1 'STEP' 1 'UNTIL' NELEMT 'DO'
'BEGIN'
'FOR' CNT1 :=1 'STEP' 1 'UNTIL' 6*2*QORT 'DO'
'BEGIN' 'IF' NODE[Z,CNT1]=U 'THEN'
'BEGIN'
'IF'QORT=0'THEN'
AUX(W[CNT1,2],W[CNT1,3],'IF'CNT1=1'THEN'1'ELSE'2,XX,YY,DETJ,NODE,Z)
'ELSE'
QAUX(W[CNT1,1],W[CNT1,2],B,XX,YY,DETJ,NODE,Z);
              'FOR' J:=1 'STEP' 1 'UNTIL' 3 'DO'
               'BEGIN'
              STR[J]:=(B[J,1]*Q[NODE[Z,1]*2-1,V]+B[J,2]*Q[NODE[Z,1]*2,V]
               +B[J,3]*Q[NODE[Z,2]*2-1,V]+B[J,4]*Q[NODE[Z,2]*2,V]
               +B[J,5]*Q[NODE[Z,3]*2-1,V]+B[J,6]*Q[NODE[Z,3]*2,V]
```

```
                +B[J,7]*Q[NODE[Z,4]*2-1,V]+B[J,8]*Q[NODE[Z,4]*2,V]
                +B[J,9]*Q[NODE[Z,5]*2-1,V]+B[J,10]*Q[NODE[Z,5]*2,V]
                +B[J,11]*Q[NODE[Z,6]*2-1,V]+B[J,12]*Q[NODE[Z,6]*2,V]);
'IF'QORT=0'THEN'
    STR[J]:=STR[J]*TH[NODE[Z,7]]
    'ELSE'
    STR[J]:=(STR[J]
                *B[J,13]*Q[NODE[Z,7]*2-1,V]+B[J,14]*Q[NODE[Z,7]*2,V]
                 +B[J,15]*Q[NODE[Z,8]*2-1,V]+B[J,16]*Q[NODE[Z,8]*2,V])*
                TH[NODE[Z,8]];
                 'FI.D';
J:=NODE[Z,7+2*QORT];
                 SIGXX:=SIGXX+(C[J,1]*STR[1]+C[J,2]*STR[2]+C[J,3]*STR[3]
                 SIGYY:=SIGYY+(C[J,2]*STR[1]+C[J,4]*STR[2]+C[J,5]*STR[3]
                 SIGXY:=SIGXY+(C[J,3]*STR[1]+C[J,5]*STR[2]+C[J,6]*STR[3]
                 EXX:=EXX+STR[1];
                 EYY:=EYY+STR[2];
                 EXY:=EXY+STR[3];
CNT2:=CNT2+1;
'END';
'END';
'END';
SIGXX:=SIGXX/CNT2;  SIGYY:=SIGYY/CNT2;  SIGXY:=SIGXY/CNT2;
EXX:=EXX/CNT2;  EYY:=EYY/CNT2;  EXY:=EXY/CNT2;
NEWLINE(1);
PRINT(U,2,0);
'IF'PRINC=1'THEN''GOTO'LINE7;
PRINT(EXX,0,4);
PRINT(EYY,0,4);
PRINT(EXY,0,4); SPACE(10);
PRINT(SIGXX,0,4);
PRINT(SIGYY,0,4);
PRINT(SIGXY,0,4);
'GOTO'LINE8;
LINE7:E12:=((EXX-EYY)↑2+(EXY)↑2)↑0.5;
E1:=(EXX+EYY)/2+E12;
E2:=(EXX+EYY)/2-E12;
SIG12:=(((SIGXX+SIGYY)/2)↑2+(SIGXY)↑2)↑0.5;
SIG1:=(SIGXX+SIGYY)/2+SIG12;
SIG2:=(SIGXX+SIGYY)/2-SIG12;
'IF'ABS(SIGYY-SIGXX)<(SIGXY*2/6.3)'THEN'DN:=0.0'ELSE'
DN:=(ARCTAN(2*SIGXY/(SIGYY-SIGXX)))*28.6478;
SEFF:=ABS(SIG1-SIG2)/1.4142135;
PRINT(E1,0,4);
PRINT(E2,0,4);
PRINT(E12,0,4);SPACE(10);
PRINT(SIG1,0,4);
PRINT(SIG2,0,4);
PRINT(SIG12,0,4);
PRINT(DN,0,4);
PRINT(SEFF,0,4);
LINE8:'END';
'END';
'END' OF PROCEDURE FELOSTR;
```

```
'PROCEDURE' ELELSTR(NSETF,Q,NELEMT,NODE,XX,YY,DETJ,TH,C,AUX,QAUX);
'VALUE' NSETF,NELEMT;  'INTEGER' NSETF,NELEMT;
'REAL' DETJ;  'INTEGER' 'ARRAY' NODE;
'REAL' 'ARRAY' XX,YY,Q,C,TH;  'PROCEDURE' AUX,QAUX;
'BEGIN' 'COMMENT' CALCULATION OF ELEMENT STRESSES AND STRAINS;
'INTEGER' V,Z,J;
'REAL' SIGXX,SIGYY,SIGXY,EXX,EYY,EXY,SIG1,SIG2,SIG12,E1,E2,E12,DD,SI
'REAL' 'ARRAY' STR[1:3],B[1:3,1:12+4*QORT];
'FOR' V:=1 'STEP' 1 'UNTIL' NSETF 'DO'
'BEGIN'
'IF'PRINC=1'THEN''GOTO'LINE1;
 NEWLINE(5);SPACE(20);
WRITETEXT('('STRAINS')'); SPACE(43);
WRITETEXT('('STRESSES')'); NEWLINE(1);
WRITETEXT('('ELET%%%EXX')'); SPACE(10);
WRITETEXT('('EYY')'); SPACE(10);
WRITETEXT('('EXY')'); SPACE(21);
WRITETEXT('('SIGMA-X%%%%%%SIGMA-Y%%%%%%SIGMA XY')');
'GOTO'LINE3;
 LINE1: NEWLINE(5); SPACE(15);
WRITETEXT('('PRINCIPAL%STRAINS')'); SPACE(30);
WRITETEXT('('PRINCIPAL%STRESSES')'); NEWLINE(1);
WRITETEXT('('ELET%%%E11%%%%%%%%%%E22')'); SPACE(10);
WRITETEXT('('E12')'); SPACE(21);
WRITETEXT('('SIGMA 1%%%%%%SIGMA 2%%%%%%SIGMA 12%%%%%%DIRECTION
%%%%%SIGMA EFF')');
LINE3:'FOR' Z:=1 'STEP' 1 'UNTIL' NELEMT 'DO'
'BEGIN'
        SIGXX:=SIGYY:=SIGXY:=EXX:=EYY:=EXY:=0.0;
'IF'QORT=0'THEN'
                AUX(0.3333,0.3333,0.3333,B,XX,YY,DETJ,NODE,Z)
'ELSE'
                QAUX(0.0,0.0,B,XX,YY,DETJ,NODE,Z);
                'FOR' J:=1 'STEP' 1 'UNTIL' 3 'DO'
               'BEGIN'
              STR[J]:=(B[J,1]*Q[NODE[Z,1]*2-1,V]+B[J,2]*Q[NODE[Z,1]*2,V
               +B[J,3]*Q[NODE[Z,2]*2-1,V]+B[J,4]*Q[NODE[Z,2]*2,V]
               +B[J,5]*Q[NODE[Z,3]*2-1,V]+B[J,6]*Q[NODE[Z,3]*2,V]
               +B[J,7]*Q[NODE[Z,4]*2-1,V]+B[J,8]*Q[NODE[Z,4]*2,V]
               +B[J,9]*Q[NODE[Z,5]*2-1,V]+B[J,10]*Q[NODE[Z,5]*2,V]
               +B[J,11]*Q[NODE[Z,6]*2-1,V]+B[J,12]*Q[NODE[Z,6]*2,V]);
'IF'QORT=0'THEN'
   STR[J]:=STR[J]*TH[NODE[Z,7]]
    'ELSE'
   STR[J]:=(STR[J]
            +B[J,13]*Q[NODE[Z,7]*2-1,V]+B[J,14]*Q[NODE[Z,7]*2,V]
            +B[J,15]*Q[NODE[Z,8]*2-1,V]+B[J,16]*Q[NODE[Z,8]*2,V])
             *TH[NODE[Z,9]];
              'END';
              J:=NODE[Z,7+2*QORT];
               SIGXX:=SIGXX+(C[J,1]*STR[1]+C[J,2]*STR[2]+C[J,3]*STR[3
               SIGYY:=SIGYY+(C[J,2]*STR[1]+C[J,4]*STR[2]+C[J,5]*STR[3
               SIGXY:=SIGXY+(C[J,3]*STR[1]+C[J,5]*STR[2]+C[J,6]*STR[3
               EXX:=EXX+STR[1];
               EYY:=EYY+STR[2];
               EXY:=EXY+STR[3];
```

```
NEWLINE(1);
PRINT(Z,2,0);
'IF'PRINC=1'THEN''GOTO'LINE2;
PRINT(EXX,0,4);
PRINT(EYY,0,4);
PRINT(EXY,0,4); SPACE(10);
PRINT(SIGXX,0,4);
PRINT(SIGYY,0,4);
PRINT(SIGXY,0,4);
'GOTO'LINE4;
LINE2:E12:=((EXX-EYY)↑2+(EXY)↑2)↑0.5;
E1:=(EXX+EYY)/2+E12;
E2:=(EXX+EYY)/2-E12;
SIG12:=(((SIGXX+SIGYY)/2)↑2+(SIGXY)↑2)↑0.5;
SIG1:=(SIGXX+SIGYY)/2+SIG12;
SIG2:=(SIGXX+SIGYY)/2-SIG12;
'IF'ABS(SIGYY-SIGXX)<(SIGXY*2/0.5)'THEN'DN:=90.0'ELSE'
DN:=(ARCTAN(2*SIGXY/(SIGYY-SIGXX)))*28.64789;
SEFF:=ABS(SIG1-SIG2)/1.4142135;
PRINT(E1,0,4);
PRINT(E2,0,4);
PRINT(E12,0,4):SPACE(10);
PRINT(SIG1,0,4);
PRINT(SIG2,0,4);
PRINT(SIG12,0,4);
PRINT(DN,0,4);
PRINT(SEFF,0,4);
LINE4:'END';
'END';
'END' OF PROCEDURE FEELSTR;



'PROCEDURE' ADDARRAY(NELEMT,NNODE,ADD,NODE);
'INTEGER' NELEMT,NNODE;
'INTEGER' 'ARRAY' NODE,ADD;
'BEGIN' 'INTEGER' W,CH,I,ADDTEMP;
'FOR' W:=1 'STEP' 1 'UNTIL' NELEMT 'DO'
'BEGIN' CH:=NODE[W,1];
        'FOR' I:=2 'STEP' 1 'UNTIL' 6+2*QORT 'DO'
        'IF' NODE[W,I]'LT'CH 'THEN' CH:=NODE[W,I];
        'FOR' I:=1 'STEP' 1 'UNTIL' 6+2*QORT 'DO'
        'BEGIN' ADDTEMP:=NODE[W,I]-CH+1;
        'IF'ADDTEMP'GT'ADD[NODE[W,I]*2]'THEN'ADD[NODE[W,I]*2]:=ADDT
        'END';
'END';
'FOR' I:=1 'STEP' 1 'UNTIL' NNODE 'DO'
'BEGIN' CH:=2*ADD[2*I]; W:=2*I;
        ADD[W-1]:=ADD[W-2]+CH-1;
        ADD[W]:=ADD[W-1]+CH;
'END';
'END' OF PROCEDURE ADDARRAY;
```

```
'PROCEDURE' RESIDUAL(K,Q,ADD,NFREE,NSETF);
'VALUE' NFREE,NSETF;
'INTEGER'NFREE,NSETF;
'INTEGER' 'ARRAY' ADD;
'REAL' 'ARRAY' K,Q;
'BEGIN' 'INTEGER' I,J,Z,DI,DJ,CI,CJ;
'ARRAY' FL1:[NFREE,1:NSETF];
'FOR' I:=1 'STEP' 1 'UNTIL' NFREE 'DO'
'FOR' J:=1 'STEP' 1 'UNTIL' NSETF 'DO' FL[I,J]:=0.0;
'FOR' Z:=1 'STEP' 1 'UNTIL' NSETF 'DO'
'FOR' I:=1 'STEP' 1 'UNTIL' NFREE 'DO'
'BEGIN' CI:=I-ADD[I]+ADD[I-1]+1;
'FOR' J:=CI 'STEP'1 'UNTIL' NFREE 'DO'
'BEGIN' 'IF' J 'LE' I 'THEN'
'BEGIN' DI:=I;  DJ:=J;
        'GOTO' LAB1;
'END' 'ELSE'
'BEGIN' CJ:=J-ADD[J]+ADD[J-1]+1;
'IF' CJ 'GT' I 'THEN' 'GOTO' LAB2;
DI:=J;  DJ:=I;
'END';
LAB1; FL[I,Z]:=FL[I,Z]+Q[J,Z]*K[ADD[DI]-DI+DJ];
LAB2; 'END';
'END';
NEWLINE(4); WRITETEXT('('RESIDUALSFORCES')');
NEWLINE(2);SPACE(1);WRITETEXT('('NODE')');  SPACE(7);
WRITETEXT('('X')');  SPACE(12); WRITETEXT('('Y')');SPACE(10);
WRITETEXT('('NODE')'); SPACE(7);WRITETEXT('('X')');SPACE(12);
WRITETEXT('('Y')');SPACE(10);WRITETEXT('('NODE')');SPACE(7);
WRITETEXT('('X')');SPACE(12);WRITETEXT('('Y')'); NEWLINE(2);
'FOR' I:=1 'STEP' 1 'UNTIL' NSETF 'DO'
'BEGIN'
'FOR'J:=0'STEP'3'UNTIL'(NFREE/2)-1'DO''BEGIN'
'FOR'Z:=1'STEP'1'UNTIL'3'DO''BEGIN'
'IF'Z+J>NFREE/2'THEN''GOTO'L16;
PRINT(Z+J,4,0);PRINT(FL[(Z+J)*2-1,1],0,5);
PRINT(FL[Z+J)*2,1],0,5);
'END' OF Z LOOP;  NEWLINE(2);
'END' OF J LOOP; L16:
'END';
'END' OF PROCEDURE RESIDUAL;
```

```
'PROCEDURE' HMNT(NG,NSETF,A,ADD,K,Q,CASE,THICK,N1,RO,GRT,AL,NS);
'VALUE' NG,NSETF,CASE; 'INTEGER' NG,NSETF,CASE,GRT;
'REAL' THICK; 'INTEGER''ARRAY' ADD,N1,NS;'REAL''ARRAY' K,Q,A,RO,AL;
'BEGIN' 'COMMENT' MODIFICATION OF OVERALL STIFFNESS MATRIX FOR THE
                  MIXED MODE CASE,D-NUMBER OF CRACK TIPS;
'INTEGER' I,J,U,C,D,Z,S,P,O,R,V,IA;
 'REAL'TA,KAP,SN,CN,MU,G,PYE,KA;
'ARRAY' KT[1:ELDOF,1:ELDOF],KC[1:NTIP*ELDOF,1:4*GRT*ELDOF],
        FI[1:2*GRT,1:ELDOF],F[1:ELDOF,1:2];
'FOR' I:=1 'STEP' 1 'UNTIL' ELDOF*NTIP 'DO'
'FOR' J:=1 'STEP' 1 'UNTIL' 4*GRT*ELDOF 'DO'  KC[I,J]:=0.0;
  PYE:=3.14159265358;
MU:=A[2];  G:=0.5*A[1]/(1+A[2]);
NEWLINE(2);
WRITETEXT('('PROCEDURE%HMNT%BEGINS')');NEWLINE(1);
WRITETEXT('('THIS%PROCEDURE%MODIFIES%THE%OVERALL%STIFFNESS%MATRIX
%FOR%MIXED%MODE%FRACTURE,N-NUMBER%OF%CRACK%TIPS%PRESENT')');
'FOR'I:=1'STEP'1'UNTIL'NTIP'DO'
'BEGIN'
NEWLINE(3);WRITETEXT('('DETAILS%OF%TIP%NUMBER%-%')');
PRINT(I,2,0);NEWLINE(2); SPACE(13);
WRITETEXT('('ANGLE%OF%TIP%TO%+VE%X-AXIS%DEG%=%')'); PRINT(AL[I],0,4)
NEWLINE(1); SPACE(12);
WRITETEXT('('NUMBER%OF%NODES%ON%CORE%ELEMENT%=%')');
PRINT(N1[I],3,0);NEWLINE(1); SPACE(21);
WRITETEXT('('RADIUS%OF%CORE%ELEMENT%=%')');
PRINT(RO[I],0,4);
NEWLINE(1); SPACE(5);
WRITETEXT('('NODE%NUMBER%STARTING%THE%CORE%INTERFACE%=%')');
PRINT(NS[I],3,0);
'END';
NEWLINE(3);WRITETEXT('('MODULUS%OF%RIGIDITY,G%=%')'); PRINT(G,0,4);
NEWLINE(1);WRITETEXT('('POISSON'S%RATIO,MU%=%')');
PRINT(MU,0,4);
 NEWLINE(2);
 WRITETEXT('('1-D%K-ARRAY%LENGTH-BAND%OR%ADD[NFREE]%=%')');
 PRINT(BAND,6,0); PRINT(ADD[NFREE],10,0);
'COMMENT' NS IS THE NODE NUMBER STARTING THE  CORE;
'IF' CASE=0 'THEN' KAP:=(3-A[2])/(1+A[2])
             'ELSE' KAP:=3-4*A[2];
'FOR'C:=1'STEP'1'UNTIL'NTIP'DO'
'BEGIN'
AL[C]:=AL[C]*(PYE/180); SN:=SIN(AL[C]); CN:=COS(AL[C]);
   CORK(KT,RO,C,KAP);
'FOR' Z:=1 'STEP' 1 'UNTIL' N1[C] 'DO'
'BEGIN'
      TA:=-PYE+2*(Z-1)*PYE/(N1[C]-1);
    F[1,1]:=-RO[C]*SIN(TA);
    F[1,2]:=RO[C]*COS(TA);
    F[2,2]:=F[3,1]:=0.0;
    F[2,1]:=F[3,2]:=1.0;
    I:=0;
    'FOR'J:=4'STEP'2'UNTIL'ELDOF*1'DO'
    'BEGIN'
     I:=I+1;
```

```
      F[J,1]:=(((KAP+I/2+(-1)↑I)*COS(I*TA/2)
                -(I/2)*COS(TA*(I/2-2)))*RO[C]↑(I/2))/(2*G);
      F[J+1,1]:=-(((KAP+I/2-(-1)↑I)*SIN(I*TA/2)
                 -(I/2)*SIN(TA*(I/2-2)))*RO[C]↑(I/2))/(2*G);
      F[J,2]:=(((KAP-I/2-(-1)↑I)*SIN(I*TA/2)
                +(I/2)*SIN(TA*(I/2-2)))*RO[C]↑(I/2))/(2*G);
      F[J+1,2]:=(((KAP-I/2+(-1)↑I)*COS(I*TA/2)
                 +(I/2)*COS(TA*(I/2-2)))*RO[C]↑(I/2))/(2*G);
  'END'J;
   F[4,1]:=F[4,1]*0.5↑0.5;  F[4,2]:=F[4,2]*0.5↑0.5;
   F[5,1]:=-F[5,1]*0.5↑0.5;  F[5,2]:=-F[5,2]*0.5↑0.5;
                             'FOR'W:=1'STEP'1'UNTIL'ELDOF'DO'
                             'BEGIN'
                              FI[Z*2-1,W]:=F[W,1]*CN-F[W,2]*SN;
                             FI[Z*2,W]:=F[W,1]*SN+F[W,2]*CN;
                                 'END'W;
'END' OF LOOP Z;
'COMMENT'    [K22] = [KC] + [A]TRAN[K11][A];
'FOR'Z:=1'STEP'1'UNTIL'ELDOF'DO'
'BEGIN'    S:=Z+ELDOF*(C-1);
   'FOR'W:=1'STEP'1'UNTIL'Z'DO'
   'BEGIN'    P:=W+4*N1[C];
      'FOR'I:=1'STEP'1'UNTIL'2*N1[C]'DO'
      'BEGIN'    Q:=I+2*(NS[C]-1);
         'FOR'J:=1'STEP'1'UNTIL'2*N1[C]'DO'
         'BEGIN'    R:=J+2*(NS[C]-1);
               'IF'R>Q'THEN''BEGIN'
                    'IF'(R-Q+1)>ADD[R]-ADD[R-1]
                    'THEN' KA:=0.0
                    'ELSE' KA:=K[ADD[R]-R+Q];
               'END' 'ELSE' 'BEGIN'
                    'IF'(Q-R+1) > ADD[Q]-ADD[Q-1]
                    'THEN' KA:=0.0
                    'ELSE' KA:=K[ADD[Q]-Q+R];
               'END';
                    KC[S,P]:=KC[S,P]+(FI[I,Z]*KA*FI[J,W]);
         'END'J;
      'END'I;
          KC[S,P]:=KC[S,P]+KT[Z,W];
   'END'W;
'END'Z;
'COMMENT'       [K21]TRAN*  =  [K21][A];
'FOR'W:=1'STEP'1'UNTIL'4*N1[C]'DO'
'BEGIN'    P:=W+2*(NS[C]-1+N1[C]);
   'FOR'I:=1'STEP'1'UNTIL'ELDOF'DO'
   'BEGIN'    Q:=I+ELDOF*(C-1);
      'FOR'J:=1'STEP'1'UNTIL'2*N1[C]'DO'
      'BEGIN'    R:=J+2*(NS[C]-1);
                 'IF' P-R+1 > ADD[P]-ADD[P-1]
                 'THEN' KA:=0.0
                 'ELSE'  KA:=K[ADD[P]-P+R];
                 KC[Q,W]:=KC[Q,W]+(KA*FI[J,I]);
      'END'J;
   'END'I;
'END'W;
```

```
'END' C TIP COUNTER;
'COMMENT' MOVING [K22]   INTO  [K11]*  , ADJUSTING [ADD]  AND [Q];
V:=R:=0; N1[0]:=0; NS[NTIP+1]:=NNODE+1;
'FOR'C:=1'STEP'1'UNTIL'NTIP'DO'
'BEGIN'   R:=R+N1[C-1];
   'FOR'I:=1'STEP'1'UNTIL'4*N1[C]'DO'
'BEGIN'  O:=I+2*(NS[C]-1+N1[C]);
         S:=I+2*(NS[C]-NS[1]-R);   Q[S,NSETF]:=Q[O,NSETF];
         U:=ADD[O];
          O:=ADD[O]-ADD[O-1];
         'IF' I 'LE' O 'THEN''BEGIN' ADD[S]:=ADD[S-1]+I;
                                U:=U-I;  'END'
                       'ELSE''BEGIN' ADD[S]:=ADD[S-1]+O;
                                U:=U-O; 'END';
      'FOR'J:=ADD[S-1]+1'STEP'1'UNTIL'ADD[S]'DO'
      'BEGIN'  W:=W+1; K[J]:=K[U];
      'END'J;
   'END'I;
   P:=2*(NS[C+1]-NS[C]-3*N1[C]);
   'FOR'I:=1'STEP'1'UNTIL'P'DO'
   'BEGIN'   O:=I+2*(NS[C]-1+3*N1[C]);
             S:=I+2*(NS[C]-NS[1]-R+2*N1[C]);  Q[S,NSETF]:=Q[O,NSETF
      V:=I+4*N1[C];  U:=ADD[O];
      O:=ADD[U]-ADD[O-1];
      'IF' O < V 'THEN'
      'BEGIN' ADD[S]:=ADD[S-1]+O;          U:=U-O;
         'FOR'Z:=ADD[S-1]+1'STEP'1'UNTIL'ADD[S]'DO'
         'BEGIN' U:=U+1; K[Z]:=K[U];'END';
      'END'  'ELSE'
      'BEGIN'  D:=0;
      'FOR'J:=NTIP'STEP'-1'UNTIL'2'DO'
      'BEGIN'  V:=V+2*(NS[J]+N1[J]-NS[J-1]-N1[J-1]);
               D:=D+2*N1[J];
         'IF' O < V 'THEN''BEGIN'
              ADD[S]:=ADD[S-1]+O-D;  U:=U-O;
              O:=ADD[S-1]+O-V+2*(NS[J]-NS[J-1]-N1[J-1]);
        'FOR'Z:=ADD[S-1]+1'STEP'1'UNTIL'ADD[S]'DO'
        'BEGIN'  U:=U+1;   K[Z]:=K[U];
                 'IF' Z = O 'THEN''BEGIN'
                            U:=U+2*N1[J];  J:=J+1;
                            O:=O+2*(NS[J]-NS[J-1]-N1[J-1]);
                            'IF' J > NTIP 'THEN' O:=0;
                            'END';

         'END'Z; 'GOTO'I1;
      'END';
   'END'J;
'END';
   I1:'END'I;
'END' C TIP COUNTER;
'COMMENT'  [K21]*, [K22]*  AND  [O2]*   FORMED;
P:=U;   R:=2*(NNODE+1-NS[1]-TH1);
'FOR'C:=1'STEP'1'UNTIL'NTIP'DO'
'BEGIN'   P:=P+N1[C-1];
   'FOR'I:=1'STEP'1'UNTIL'ELDOF'DO'
```

```
   'BEGIN'    O:=I+R+ELDOF*(C-1);
              S:=O-2*(NS[C]-NS[1]-D);
              ADD[O]:=S+ADD[O-1];    Q[O,NSETP]:=0.0;
              O:=I+ELDOF*(C-1);
      'FOR'J:=1'STEP'1'UNTIL'4*N1[C]'DO'
         K[ADD[O]-S+J]:=KC[O,J];
      'FOR'J:=J'STEP'1'UNTIL'S-I'DO'
          K[ADD[O]-S+J]:=0.0;
      'FOR'J:=J'STEP'1'UNTIL'S'DO'
         K[ADD[O]-S+J]:=KC[O,J-S+I+4*N1[C]];
   'END'];
'END' C COUNTER;
'END' OF PROCEDURE MAKT;




'PROCEDURE' MAKTKBAND(NFREE,N1,BAND,ADD,NS);
'INTEGER'BAND,NFREE;
'INTEGER''ARRAY'N1,NS,ADD;
'BEGIN''INTEGER' I,V,C,P,O,D,R;
   R:=V:=BAND:=0;    N1[0]:=0; NS[NTIP+1]:=NNODE+1;
'FOR'C:=1'STEP'1'UNTIL'NTIP'DO'
'BEGIN'
   'FOR'I:=1'STEP'1'UNTIL'4*N1[C]'DO'
   'BEGIN'    O:=I+2*(NS[C]-1+N1[C]);
              O:=ADD[O]-ADD[O-1];
   'IF' I 'LE' O 'THEN' BAND:=BAND+I
                 'ELSE' BAND:=BAND+O;
   'END'I;
   R:=R+N1[C-1];    P:=2*(NS[C+1]-NS[C]-3*N1[C]);
   'FOR'I:=1'STEP'1'UNTIL'P'DO'
   'BEGIN'
      O:=I+2*(NS[C]-1+3*N1[C]);    O:=ADD[O]-ADD[O-1];
      V:=I+4*N1[C];
      'IF'O<V'THEN' BAND:=BAND+O
            'ELSE'  'BEGIN'
         D:=0;
      'FOR'J:=NTIP'STEP'-1'UNTIL'2'DO'
      'BEGIN'  V:=V+2*(NS[J]+N1[J]-NS[J-1]-N1[J-1]);
               D:=D+2*N1[J];
            'IF'O<V'THEN' 'BEGIN' BAND:=BAND+O-D;
                          'GOTO'I2; 'END';
      'END'J;
      'END';
   I2:'END';
      O:=-2*(NS[C]-NS[1]-R)+(ELDOF)*(C-1)+2*(NNODE+1-NS[1]-TN1);
      BAND:=BAND+ELDOF*(O+(ELDOF/2)*0.5);
'END' OF C LOOP;
      'IF' ADD[NFREE] > BAND 'THEN'
           BAND:=ADD[NFREE];
'END' OF PROCEDURE MAKTKBAND;
```

```
'PROCEDURE' CORK(KT,RO,TIP,KAP);
'VALUE'TIP;
'INTEGER'TIP;
'REAL' KAP;
'REAL''ARRAY'KT,RO;
'BEGIN' 'INTEGER'H,N,P;
        'REAL'G,S,O;
        'REAL''ARRAY'C[1:11],B[1:8],T[1:4],D[1:16],
          A1,A2[1:ELDOF-3],K[1:4];
'FOR'I:=1'STEP'1'UNTIL'ELDOF'DO'
'FOR'J:=1'STEP'1'UNTIL'ELDOF'DO' KT[I,J]:=0.0;
 G:=0.5*A[1]/(1+Ar2]);
'FOR'I:=1'STEP'1'UNTIL'ELDOF-3'DO'
   A1[I]:=A2[I]:=1.0;
   A1[1]:=A2[1]:=0.5↑0.5;
   A1[2]:=A2[2]:=-A1[1];
'FOR'N:=1'STEP'1'UNTIL'(ELDOF-3)/2'DO'
'BEGIN'
   'FOR'H:=1'STEP'1'UNTIL'N'DO'
   'BEGIN'
   C[1]:=N/2+1;    C[2]:=N/2-(-1)↑N;    C[3]:=N/2+(-1)↑N;
   C[4]:=3-N/2;    C[5]:=N/2-1;    C[6]:=KAP-H/2;
   C[7]:=H/2-(-1)↑H;    C[8]:=H/2+(-1)↑H;    C[9]:=H/2-2+KAP;
   C[10]:=N/2-1;    C[11]:=N/2+1;
   B[1]:=C[5]+C[10];    B[2]:=C[5]-C[10];
   B[3]:=C[5]+C[11];    B[4]:=C[5]-C[11];
   B[5]:=C[1]+C[10];    B[6]:=C[1]-C[10];
   B[7]:=C[1]+C[11];    B[8]:=C[1]-C[11];
   S:=(THICK*H*N*(RO[TIP]↑((N+H)/2)))/(8*G*(N+H));
'FOR'P:=0,1 'DO'
'BEGIN'
O:=P*2*3,141592653589;
'FOR'I:=1'STEP'1'UNTIL'4'DO'
'BEGIN'
   'FOR'J:=1,0 'DO'
   'BEGIN'
   'IF' B[I*2-J]=0 'THEN''BEGIN'
                   T[2-J]:=0;  T[4-J]:=0.0;
                 'END'  'ELSE''BEGIN'
         T[2-J]:=(SIN(O*B[I*2-J]))/B[I*2-J];
         T[4-J]:=(COS(O*B[I*2-J]))/B[I*2-J];
                 'END';
   'END'J;
D[I]:=T[1]+T[2];    D[I+4]:=T[2]-T[1];
D[I+8]:=T[4]+T[3];    D[I+12]:=T[4]-T[3];
'END'I;
K[1]:=C[4]*C[6]*D[1]+C[4]*C[8]*D[2]+C[3]*C[6]*D[3]+C[3]*C[8]*D[4]
     +C[1]*C[9]*D[1]-C[1]*C[8]*D[2]-C[3]*C[9]*D[3]+C[3]*C[8]*D[4]
     +2*(C[5]*C[10]*D[5]-C[5]*C[8]*D[6]-C[3]*C[10]*D[7]+
       C[3]*C[8]*D[8]);
K[2]:=-C[4]*C[6]*D[13]-C[4]*C[7]*D[14]-C[3]*C[6]*D[15]
      -C[3]*C[7]*D[16]-C[1]*C[9]*D[13]+C[1]*C[7]*D[14]+C[3]*C[9]*D[1
      -C[3]*C[7]*D[16]+2*(-C[5]*C[10]*D[9]+C[5]*C[7]*D[10]
      +C[3]*C[10]*D[11]-C[3]*C[7]*D[12]);
```

```
K[3]:=C[4]*C[6]*D[9]+C[4]*C[8]*D[10]+C[2]*C[6]*D[11]+C[2]*C[8]*D[12
     +C[1]*C[9]*D[9]-C[1]*C[8]*D[10]-C[2]*C[9]*D[11]+C[2]*C[8]*D[12]
   +2*(C[5]*C[10]*D[13]-C[5]*C[8]*D[14]-C[2]*C[10]*D[15]+
      C[2]*C[8]*D[16]);
K[4]:=C[4]*C[6]*D[5]+C[4]*C[7]*D[6]+C[2]*C[6]*D[7]+C[2]*C[7]*D[8]
     +C[1]*C[9]*D[5]-C[1]*C[7]*D[6]-C[2]*C[9]*D[7]+C[2]*C[7]*D[8]
  +2*(C[5]*C[10]*D[1]-C[5]*C[7]*D[2]-C[2]*C[10]*D[3]
      +C[2]*C[7]*D[4]);

I:=2*N+2;
J:=2*N+3;
V:=2*M+2;
W:=2*M+3;
'IF' P = 0 'THEN'
'BEGIN' KT[I,V]:=K[1];
        KT[I,W]:=K[2];
        KT[J,V]:=K[3];
        KT[J,W]:=K[4];
'END';
'END'P;
KT[I,V]:=(K[1]-KT[I,V])*S*A1[2*M-1]*A2[2*N-1];
KT[I,W]:=(K[2]-KT[I,W])*S*A1[2*M-1]*A2[2*N];
KT[J,V]:=(K[3]-KT[J,V])*S*A1[2*M]*A2[2*N-1];
KT[J,W]:=(K[4]-KT[J,W])*S*A1[2*M]*A2[2*N];
KT[V,I]:=KT[I,V];
KT[W,I]:=KT[I,W];
KT[V,J]:=KT[J,V];
KT[W,J]:=KT[J,W];
'END'M;
'END'N;
'END' OF PROCEDURE CORK;
```

```
'PROCEDURE' CLOSURE(K,NOSK,ANGSK,ADD,Q,RESOLVE,BAND,NG,U,OVP);
'INTEGER' BAND,NG,U;
'INTEGER''ARRAY'NOSK,ADD,OVP;
'REAL''ARRAY'K,ANGSK,Q;
'LABEL' RESOLVE;
'BEGIN''INTEGER'U,I,J,F,Z,S,S2;
        'REAL'A,B,D,E;
          'REAL''ARRAY'X,Y[1:SURNO];
'COMMENT'  NO-FRICTION MODEL IE, CRACK FACES COUPLED IN
                    THE NORMAL DISPT SENSE ONLY;
 E:=0.0; V:=V+1;
'FOR'I:=1'STEP'1'UNTIL'SURNO'DO''BEGIN'
  X[I]:=Q[2*NOSK[I],1]; Y[I]:=Q[2*NOSK[I]-1,1]; 'END';
'FOR'I:=1'STEP'1'UNTIL'V-1'DO'
'BEGIN'
      'IF' X[OVP[I]] = 0.0 'THEN'
           X[OVP[I]]:= X[OVP[I]+1] 'ELSE'
           X[OVP[I]+1]:= X[OVP[I]];
'END';
'FOR'I:=1'STEP'1'UNTIL'SURNO/2'DO'
'BEGIN' A:=X[2*I-1]; B:=X[2*I]; D:=A-B;
   'IF' D < 0.0 'THEN'
  'BEGIN'
   'IF' ABS(D) > ABS(E) 'THEN'
   'BEGIN' E:=D; U:=I*2-1;  'END';
  'END';
'END';
 OVP[V]:=U;
 NEWLINE(4); WRITETEXT('('%%%ITERATION%%---%')');
     PRINT(V,3,0);
NEWLINE(3); SPACE(26); WRITETEXT('('UPPER%CRACK%FACE')');
SPACE(38); WRITETEXT('('LOWER%CRACK%FACE')');
NEWLINE(2); SPACE(20);
WRITETEXT('('NORMAL%%%%%%%%%%%%%TANGENTIAL')'); SPACE(27);
WRITETEXT('('NORMAL%%%%%%%%%%%%%TANGENTIAL')');
NEWLINE(1);
WRITETEXT('('%PUSH%%%%%%NODE%%%DISPLACEMENT%%%%%%%%DISPLACEMENT')')
SPACE(18);WRITETEXT('('NODE%%%DISPLACEMENT%%%%%%%%DISPLACEMENT')');
NEWLINE(2);
'FOR'I:=1'STEP'2'UNTIL'SURNO'DO'
'BEGIN'
PRINT(I,4,0); SPACE(3); PRINT(NOSK[I],4,0);
PRINT(X[I],0,6); SPACE(5); PRINT(Y[I],0,6);
SPACE(15); PRINT(NOSK[I+1],4,0); PRINT(X[I+1],0,6);
SPACE(5); PRINT(Y[I+1],0,6);
 NEWLINE(1);
'END';
 NEWLINE(3);
WRITETEXT('('MAXIMUM%OVERLAP%=%')');
PRINT(E,0,6); SPACE(6); WRITETEXT('(',AT%POSITION%=%')');
PRINT(W,4,0);
NEWLINE(4);
WRITETEXT('('CRACK')'); NEWLINE(1);
WRITETEXT('('%NO%%%%%%%%%%%%%X-DISP%%%%%%%%%%%%%Y-DISP')');
WRITETEXT('('%%%%%%%%%%%%%%%%KI%%%%%%%%%%%%%%%KII')');
```

```
SPACE(15);WRITETEXT('('ROTATION')');
'FOR'J:=1'STEP'1'UNTIL'NTIP'DO'
'BEGIN'  I:=ELDOF*(J-1);
NEWLINE(2); PRINT(J,4,0);
SPACE(5);  PRINT(Q[U+I+2,1],0,6);
SPACE(5);  PRINT(Q[U+I+3,1],0,6);
SPACE(5);  PRINT(Q[U+I+4,1],0,6);
SPACE(5);  PRINT(Q[U+I+5,1],0,6);
SPACE(5);  PRINT(Q[U+I+1,1],0,6);
 'END'  J ;
NEWLINE(4);
'IF' E 'GE' 0.0 'THEN''GOTO' VA2;
REWIND(12);  GETARRAY(12,K);
 REWIND(10); GETARRAY(10,Q);
'IF'NOSK[W] > NOSK[U+1] 'THEN'
'BEGIN' J:=NOSK[W]*2;  I:=NOSK[U+1]*2;
'END' 'ELSE'
'BEGIN' J:=NOSK[W+1]*2;  I:=NOSK[U]*2;
'END';
J:=J-1;  I:=I-1;
'FOR'Z:=1 'DO'
'BEGIN'  J:=J+Z;   I:=I+Z;
 S2:=I-(ADD[I]-ADD[I-1])+1;
'FOR'F:=S2'STEP'1'UNTIL'I-1'DO'
      K[ADD[J]-J+F]:=K[ADD[J]-J+F]+K[ADD[I]-I+F];
'FOR'F:=I+1'STEP'1'UNTIL'J'DO'
'BEGIN'
   'IF'ADD[F]-ADD[F-1] 'GE' F+1-I 'THEN'
      K[ADD[J]-J+F]:=K[ADD[J]-J+F]+K[ADD[F]-F+I];
'END'F;
'FOR'F:=J'STEP'1'UNTIL'U+ELDOF*NTIP'DO'
'BEGIN'
   'IF' ADD[F]-ADD[F-1] 'GE' F+1-I 'THEN'
      K[ADD[F]-F+J]:=K[ADD[F]-F+J]+K[ADD[F]-F+I];
'END'F;
      K[ADD[J]]:=K[ADD[J]]+K[ADD[I]];
      Q[J,1]:=Q[J,1]+Q[I,1];
      GEOMBC(0.0,1,Q,K,U+ELDOF*NTIP,1,ADD);
'END'Z;
VA3;'GOTO' RESOLVE;
VA2;  W:=0;
    'FOR'J:=1'STEP'1'UNTIL'NTIP'DO'
    'BEGIN'  I:=ELDOF*(J-1);
    'IF'  Q[U+I+4,1]  <  0.0  'THEN'
      'BEGIN'
      'IF'  U=0 'THEN' 'BEGIN'
       REWIND(12);  GETARRAY(12,K);
       REWIND(10); GETARRAY(10,Q);
       W:=1;  'END';
     GEOMBC(0.0,U+I+4,0,K,U+ELDOF*NTIP,1,ADD);
      'END';
    'END'J;
    'IF'  W=1  'THEN' 'GOTO' VA3;
'END' PROCEDURE CLOSURE;
```

```
'PROCEDURE' SEGSOL(A,S,B)DIMENSIONS:(N,R)FAILURE EXIT:(FAIL);
'VALUE'N,R; 'ARRAY'A,B; 'INTEGER''ARRAY'S;
'INTEGER'N,R;
'LABEL' FAIL;
'BEGIN'
'INTEGER'G,H,I,J,K,M,P,Q,T,U,V,W,Z,P1;
'REAL'Y;
'REAL''ARRAY'L[1:SIZE];
M:=0;
REWIND(CHITA); PUTARRAY(CHITA,A);
  REWIND(1); GETARRAY(1,A);
'FOR'I:=1'STEP'1'UNTIL'SEG'DO'
 'BEGIN'
   'FOR'W:=COL[I]'STEP'1'UNTIL'I'DO'
   'BEGIN'
   'IF' W 'NE' I 'THEN' 'BEGIN'
          REWIND(W); GETARRAY(W,L);
                'END';
      'FOR'Z:=ROW[I-1]+1'STEP'1'UNTIL'ROW[I]'DO'
      'BEGIN'
      P:=S[Z-1];
      T:=Z-(S[Z]-P)+1;
      G:=P+1;
      Q:=ROW[W-1]+1;
      'IF' (Z+1-Q) 'LT' (Z+1-T)
                  'THEN' H:= Q-T  'ELSE' H:=0;
      H:=H+P;
      'IF' W 'EQ' I 'THEN'P1:=Z-1'ELSE'P1:=ROW[W];
      'IF' T 'GT' Q 'THEN'Q:=T;
        'FOR'J:=Q'STEP'1'UNTIL'P1'DO'
        'BEGIN'
        U:=G;
        M:=M+1;
        K:=J-(S[J]-S[J-1])+1;
        V:=M-S[J];
        Y:=A[M-LIM[I-1]];
        'IF' K 'GT' T 'THEN'U:=U+K-T;
        'FOR'U:=U'STEP'1'UNTIL'M-1'DO'
         'IF'W 'EQ' I 'THEN'Y:=Y-A[U-LIM[I-1]]*A[U-V-LIM[I-1]]
                      'ELSE'Y:=Y-A[U-LIM[I-1]]*L[U-V-LIM[W-1]];
         'IF' W 'EQ' I 'THEN'Y:=Y/A[M-V-LIM[I-1]]
                       'ELSE'Y:=Y/L[M-V-LIM[W-1]];
        A[M-LIM[I-1]]:=Y;
        'FOR'M:=1'STEP'1'UNTIL'K'DO'
         B[Z,M]:=B[Z,M]-B[J,M]*Y;
        'END'J;
      'IF' W 'EQ' I 'THEN''BEGIN'
             Y:=A[M+1-LIM[I-1]];
        'FOR'U:=U'STEP'1'UNTIL'M'DO'
             Y:=Y-A[U-LIM[I-1]]↑2;
        'IF' Y 'LE' 0 'THEN''GOTO' FAIL;
        Y:=SQRT(Y);
  M:=M+1;
```

```
        A[M-LIM[I-1]]:=Y;
        'FOR'N:=1'STEP'1'UNTIL'R'DO'
         B[Z,N]:=B[Z,N]/Y;
                        'END';
      'END'Z;
  'END'W;
'IF'I 'EQ' SEG 'THEN''GOTO'REDUCED;
 REWIND(I); PUTARRAY(I,A);
 REWIND(I+1); GETARRAY(I+1,A);
REDUCED:'END'I;
'COMMENT' REDUCTION COMPLETE;
M:=S[N];
'FOR'J:=SEG'STEP'-1'UNTIL'1'DO'
'BEGIN'
    'FOR'I:=ROW[J]'STEP'-1'UNTIL'ROW[J-1]+1'DO'
    'BEGIN'
    Y:=A[M-LIM[J-1]];
      'FOR'M:=1'STEP'1'UNTIL'R'DO'
      B[I,N]:=B[I,N]/Y;
    'IF' I 'EQ' 1 'THEN''GOTO'COMPLETE;
    V:=I;
    P:=S[I-1];
      'FOR'M:=M-1'STEP'-1'UNTIL'P+1'DO'
      'BEGIN'
      V:=V-1;
      Y:=A[M-LIM[J-1]];
        'FOR'N:=1'STEP'1'UNTIL'R'DO'
        B[V,N]:=B[V,N]-B[I,N]*Y;
      'END'M;
    M:=P;
    'END'I;
  REWIND(J-1); GETARRAY(J-1,A);
COMPLETE;'END'J;
'END' OF PROD SEGSOL;
```

```
'PROCEDURE'TEST(A,C,D,P,K,CK);
'VALUE'A,C,D,CK;
'INTEGER'A,C,D,CK;
'REAL''ARRAY'K,P;
'BEGIN''INTEGER'Z,I;
        'IF'A'LE'C
             'OR'A'GT'D
'THEN''BEGIN'
'FOR'Z:=1'STEP'1'UNTIL'SEG'DO''BEGIN'
   'IF'A'LE'LIM[Z1'THEN''BEGIN'
     'IF'CK=1'THEN''BEGIN'
REWIND(CHTA);
        PUTARRAY(CHTA,K); REWIND(Z);  GETARRAY(Z,K);
        CHTA:=Z; CK1:=LIM[Z-1]; CK2:=LIM[Z];
'GOTO'SEGEND;
             'END'
        'ELSE' 'BEGIN'
         REWIND(Z); GETARRAY(Z,P);
        CHTP:=Z; CK3:=LIM[Z-1]; CK4:=LIM[Z]; 'GOTO'SEGEND;
             'END';
        'END';
      'END';
   'END';
SEGEND:'END' OF PROD TEST;
```

```
'PROCEDURE' ADDSEQ(NELEMT,NNODE,ADD,NODE);
'INTEGER' NELEMT,NNODE;
'INTEGER' 'ARRAY' NODE,ADD;
'BEGIN' 'INTEGER' W,CH,I,ADDTEMP,CHANGE;
'FOR' W:=1 'STEP' 1 'UNTIL' NELEMT 'DO'
'BEGIN'  CH:=NODE[W,1];
         'FOR' I:=2 'STEP' 1 'UNTIL' 6+2*QORT 'DO'
         'IF' NODE[W,I]'LT'CH 'THEN' CH:=NODE[W,I];
         'FOR' I:=1 'STEP' 1 'UNTIL' 6+2*QORT 'DO'
         'BEGIN' ADDTEMP:=NODE[W,I]-CH+1;
         'IF'ADDTEMP'GT'ADD[NODE[W,I]*2]'THEN'ADD[NODE[W,I]*2]:=ADDT
         'END';
'END';
'FOR' I:=1 'STEP' 1 'UNTIL' NNODE 'DO'
'BEGIN' CH:=2*ADD[2*I];  W:=2*I;
        ADD[W-1]:=ADD[W-2]+CH-1;
        ADD[W]:=ADD[W-1]+CH;
'END';
LIM[0]:=0;  ROW[0]:=0;
SEG:=ENTIER(ADD[NFREE]/35000)+1;
W:=ENTIER(ADD[NFREE]/SEG);
'FOR'I:=1'STEP'1'UNTIL'SEG'DO'
  'IF'I=SEG'THEN'LIM[I]:=ADD[NNODE*2]
        'ELSE''BEGIN'
    LIM[I]:=W+LIM[I-1];
    'FOR'J:=0'STEP'2'UNTIL'NNODE*2'DO'
    'BEGIN'
    'IF'LIM[I] 'LT' ADD[(ENTIER(NFREE/(SEG*2))*2)*I+J]
    'THEN' 'BEGIN'
    LIM[I]:=ADD[(ENTIER(NFREE/(SEG*2))*2)*I+J];
    ROW[I]:=ENTIER(NFREE/(SEG*2))*2*I+J;
    'GOTO' AD2; 'END';
    'END'OF LOOP J;
AD2:'END';
ROW[SEG]:=NFREE;
SIZE:=LIM[1];
'FOR'I:=2'STEP'1'UNTIL'SEG'DO'
   'IF'SIZE'LT'LIM[I]-LIM[I-1]
        'THEN'SIZE:=LIM[I]-LIM[I-1];

'FOR'I:=1'STEP'1'UNTIL'SEG'DO'
'BEGIN'
CHANGE:=NFREE;
   'FOR'J:=ROW[I-1]+1'STEP'1'UNTIL'ROW[I]'DO'
   'BEGIN'
   COL[I]:=J-(ADD[J]-ADD[J-1])+1;
   'IF'COL[I] 'LT' CHANGE 'THEN'CHANGE:=COL[I];
   'END'J;
   'FOR'J:=1'STEP'1'UNTIL'SEG'DO'
   'IF' CHANGE 'LE' ROW[J] 'THEN'
        'BEGIN' COL[I]:=J; 'GOTO'AD3; 'END';
AD3:'END'I;
CK1:=CK3:=LIM[0];  CK2:=CK4:=LIM[1];  CHTA:=CHTP:=1;
CK:=1;
'END' OF PROCEDURE ADDSEQ;
```

The following standard procedures have been slightly modified to accommodate the segmented stiffness matrix.

```
'PROCEDURE' GEOMBC(U,N,R,AK,NEQ,F,A,TEST);
'VALUE' U,N,NEQ,F;
'REAL' U;
'INTEGER' N,NEQ,F;
'ARRAY' R,AK;
'INTEGER' 'ARRAY' A;
'PROCEDURE'TEST;
'BEGIN' 'INTEGER' M,K,CJ;
'IF' N=1 'THEN' CJ:=1 'ELSE' CJ:=N-(A[N]-A[N-1])+1;
'FOR' K:=CJ 'STEP' 1 'UNTIL' N 'DO'
'BEGIN'
TEST(A[N]-N+K,CK1,CK2,AK,AK,1);
        R[K,F]:=R[K,F]-AK[A[N]-N+K-CK1]*U;
        AK[A[N]-N+K-CK1]:=0.0;
'END';
'IF' N+1 'GT' NEQ 'THEN' 'GOTO' LZ;
'FOR' K:=N+1 'STEP' 1 'UNTIL' NEQ 'DO'
'BEGIN' CJ:=K-(A[K]-A[K-1])+1;
        'IF' CJ 'LE' N 'THEN'
        'BEGIN'
TEST(A[K]-K+N,CK1,CK2,AK,AK,1);
                R[K,F]:=R[K,F]-AK[A[K]-K+N-CK1]*U;
                AK[A[K]-K+N-CK1]:=0.0;
        'END' 'ELSE'
'END';
 LZ;TEST(A[N],CK1,CK2,AK,AK,1);
 AK[A[N]-CK1]:=1.0;
    R[N,F]:=U;
'END';
```

Excerpt from procedure FEASSEMBLY showing the inserted sub-procedure TEST.

```
'IF' SUB1 'LT' SUB2 'THEN' 'GOTO' LABA;
A1:=ADD[SUB1]-SUB1+SUB2;
TEST(A1,CK1,CK2,K,K,1);
  K[A1-CK1]:=K[A1-CK1]+KE[I*2-1,J*2-1];
LABA: 'IF' SUB3 'LT' SUB2 'THEN' 'GOTO' LABB;
 A2:=ADD[SUB3]-SUB3+SUB2;
  TEST(A2,CK1,CK2,K,K,1);
 K[A2-CK1]:=K[A2-CK1]+KE[I*2,J*2-1];
LABB; 'END';
'END';
'END' OF PROCEDURE FEASSEMBLY;
```

```
'PROCEDURE' MIIST(N1,RO,NG,NSETF,A,ADD,K,Q,CASE,THICK,FO1BLA,FI);
Formulation of [KC] and [A] as in procedure  MMNT

IFAIL:=0;
FO1BLA(2*N1,ELDOF,1.0&-10,FI,UP,RANK,IFAIL);
'FOR'Z:=1'STEP'1'UNTIL'2*N1'DO'
'FOR'W:=1'STEP'1'UNTIL'Z'DO'
'FOR'I:=1'STEP'1'UNTIL'ELDOF'DO'
'FOR'J:=1'STEP'1'UNTIL'ELDOF'DO'
K[ADD[Z]-Z+W]:=K[ADD[Z]-Z+W]+(FI[Z,I]*KT[I,J]*FI[W,J]);
'END'OF PROD MIIST;
```

```
'PROCEDURE' SOLVIT(Q,PSO,N1);
'INTEGER'N1;
'ARRAY'Q,PSO;
'BEGIN''INTEGER'I,J;
      'ARRAY'ALPH[1:ELDOF];
'FOR'I:=1'STEP'1'UNTIL'5'DO'
'BEGIN'ALPH[I]:=0.0;
'FOR'J:=1'STEP'1'UNTIL'2*N1'DO'
ALPH[I]:=PSO[J,I]*Q[J,1]*ALPH[I];
'END' OF I;
NEWLINE(4);
WRITETEXT('('CRACK%TIP%DISPLACEMENT%IN%X-DIRECTION=')');
PRINT(ALPH[2],0,10);
NEWLINE(2);
WRITETEXT('('CRACK%TIP%DISPLACEMENT%IN%Y-DIRECTION=')');
PRINT(ALPH[3],0,10);
NEWLINE(4);
WRITETEXT('('MODE1%STRESS%INTENSITY%FACTOR%KI=')');
PRINT(ALPH[4],0,10);
NEWLINE(2);
WRITETEXT('('MODE2%STRESS%INTENSITY%FACTOR%KII=')');
PRINT(ALPH[5],0,10);
NEWLINE(4);
WRITETEXT('('RIGID%BODY%ROTATION%U=')');
PRINT(ALPH[1],0,10);
'END' OF PROD SOLVIT;
```

```
'PROCEDURE' STRDIS(L1,L2,L3,B,X,Y,U,N,Z,ER,SOLID,ZER);
'VALUE' L1,L2,L3,Z;
'INTEGER' Z,ER,SOLID;
'REAL' L1,L2,L3,U;
'INTEGER' 'ARRAY' N,ZER;
'ARRAY' X,Y,B;
'BEGIN'
'INTEGER' I,V;
'REAL' CHANGE,C;
'REAL''ARRAY'J[1:2,1:2],NL[1:6],P[1:2,1:6];
'COMMENT' THIS PROCEDURE EVALUATES THE JACOBIAN J ITS DETERMINANT U
          AND THE STRAIN-DISP ARRAY B;
J[1,1]:=X[N[Z,1]]*(4*L1-1)+X[N[Z,3]]*(4*L1+4*L2-3)+4*L2*X[N[Z,4]]-
4*L2*X[N[Z,5]]+4*X[N[Z,6]]*(1-2*L1-L2);
J[1,2]:=Y[N[Z,1]]*(4*L1-1)+Y[N[Z,3]]*(4*L1+4*L2-3)+4*L2*Y[N[Z,4]]-
4*L2*Y[N[Z,5]]+4*Y[N[Z,6]]*(1-2*L1-L2);
J[2,1]:=X[N[Z,2]]*(4*L2-1)+X[N[Z,3]]*(4*L1+4*L2-3)+4*L1*X[N[Z,4]]+
4*X[N[Z,5]]*(1-2*L1-2*L2)-4*L1*X[N[Z,6]];
J[2,2]:=Y[N[Z,2]]*(4*L2-1)+Y[N[Z,3]]*(4*L1+4*L2-3)+4*L1*Y[N[Z,4]]+
4*Y[N[Z,5]]*(1-2*L1-2*L2)-4*L1*Y[N[Z,6]];
'COMMENT' U REPLACES DETJ;
U:=J[1,1]*J[2,2]-J[1,2]*J[2,1];
'COMMENT' THE COEFFS OF [J] ARE REPLACED BY THOSE OF [J]-1;
CHANGE:=J[1,1];
J[1,1]:=J[2,2]/U;
J[1,2]:=-J[1,2]/U;
J[2,1]:=-J[2,1]/U;
J[2,2]:=CHANGE/U;
'COMMENT' NL IS THE DIS FUNCTION DNS&DNT ITS DERIVATIVES WRT S&T;
NL[1]:=L1*(2*L1-1);
NL[2]:=L2*(2*L2-1);
NL[3] :=L3*(2*L3-1);
NL[4]:=4*L1*L2;
NL[5]:=4*L2*L3;
NL[6]:=4*L3*L1;
'IF'SOLID=1'THEN''BEGIN'
   'FOR'I:=1'STEP'1'UNTIL'6'DO''BEGIN'
'IF'Z=ZER[I]'THEN'
   'BEGIN'
    RMEAN:=(X[N[Z,1]]+X[N[Z,2]]+X[N[Z,3]]+X[N[Z,4]]+X[N[Z,5]]+
            X[N[Z,6]])/6;
    RAVG:=RMEAN;
    'GOTO'SL1;
   'END';'END';
'END';
RAVG:=0.0;
'FOR'I:=1'STEP'1'UNTIL'6'DO'
   RAVG:=RAVG+(X[N[Z,I]]*NL[I]);
SL1:
'FOR' I:=1 'STEP' 1 'UNTIL' 6 'DO'
'FOR' V:=1,2 'DO' P[V,I]:=0.0;
'FOR' I:=1,2 'DO'
'BEGIN'  P[I,1]:=J[I,1]*(4*L1-1);
P[I,2]:=J[I,2]*(4*L2-1);
P[I,3]:=J[I,1]*(1-4*L3)+J[I,2]*(1-4*L3);
```

```
P[I,4]:=4*(12*J[I,1]+L1*J[I,2]);
P[I,5]:=4*(J[I,2]*L3-L2*(J[I,2]+J[I,1]));
P[I,6]:=4*(J[I,1]*L3-L1*(J[I,1]+J[I,2]));
'END';
'FOR' I:=1 'STEP' 1 'UNTIL' 12 'DO'
'FOR' V:=1,2,3,4 'DO' B[V,I]:=0.0;
'FOR' I:=1 'STEP' 1 'UNTIL' 6 'DO'
'BEGIN'
B[1,(I*2-1)]:=B[4,(I*2)]:=P[1,I];
B[4,(I*2-1)]:=B[3,(I*2)]:=P[2,I];
B[2,(I*2-1)]:=N[I]/RAVG;
'END';
'END' OF STRDIS;
```

```
'PROCEDURE'RZERO(N,X,Y,ZER);
'INTEGER''ARRAY'N,ZER;
'REAL''ARRAY'X,Y;
'BEGIN' EC:=0;
'FOR'I:=1'STEP'1'UNTIL'NELEMT'DO'
   'BEGIN'
    'FOR'J:=1'STEP'1'UNTIL'6'DO'
     'BEGIN'
      'IF'X[N[I,J]]=0.0'THEN'
          'BEGIN'
          EC:=EC+1; ZER[EC]:=I;  'GOTO'NEXT;
          'END';
    'END'OF LOOP J;
NEXT;'END' OF LOOP I;
'END' OF PROD RZERO;
```

9.3.4 DESCRIPTION OF PROCEDURES USED IN THE AUTO-MESH GENERATION PROGRAM

The procedures are described with the aid of flowcharts and a corresponding formal listing is given.

PROCEDURE INDEX:-

COINCID

TR/QELENCONS

TRACE

TR/QELETXY

MATCON

HATCHET

RESPLOT

TR/QPLOT

TR/QIDN

MGINPUT

OUTPUT1

OUTPUT2

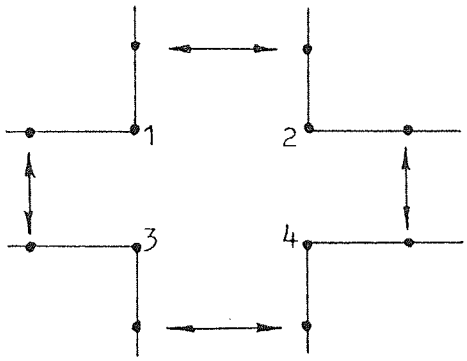
COREGEN

STDGEN

ZONEXY

PROCEDURE COINCID

This procedure performs a rather complicated function, in that situations may arise where two opposing zone faces may be joined to two other zone faces, causing three of the corner nodes to be redundant. As an example of this situation see the figure below.

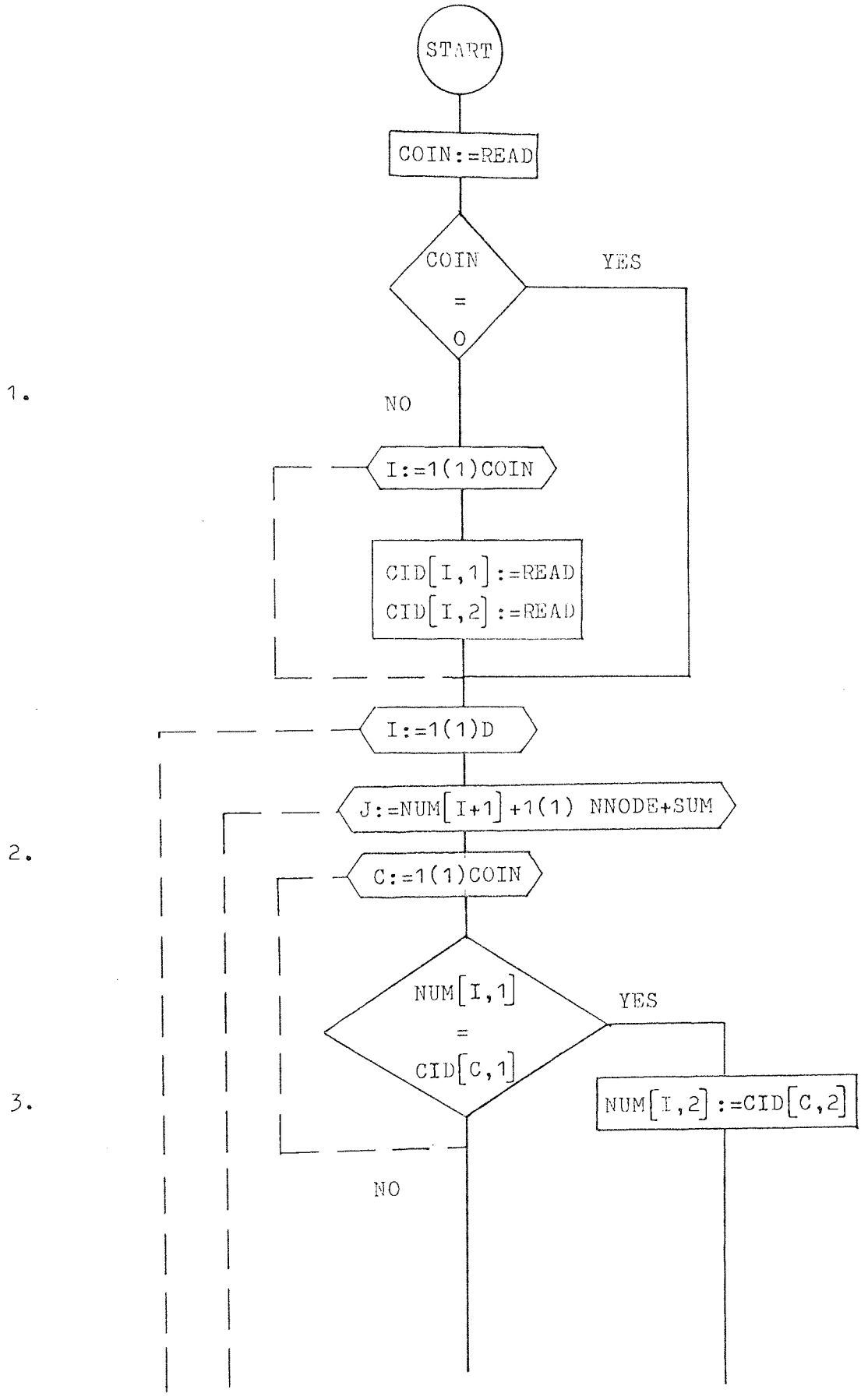


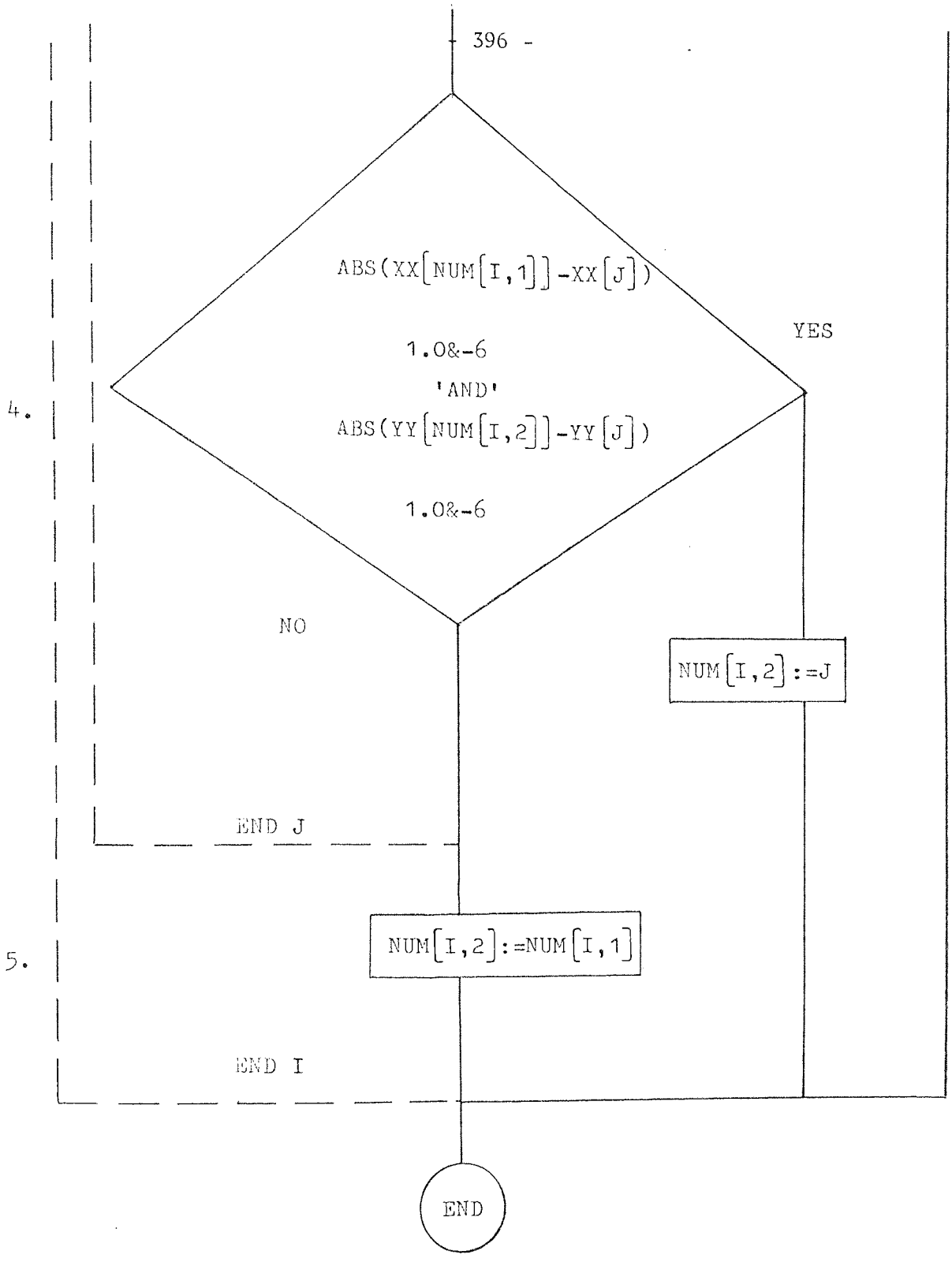
When the four zone faces are joined the nodes 2, 3 and 4, say, become redundant. It will be appreciated that the four zones are not in the above positions in the zone array, they may be quite remote from each other. The problem in this case arises, from the fact that all four nodes have the same coordinates. Hence when the redundant nodes are being scanned, using the coordinates of the node to be retained, any one of the remaining three node numbers could be selected. Therefore the procedure allows for the node numbers to be altered by the operator and overrides the scanning facility.

The scanning operation is carried out in order to obtain the node numbers which will be made redundant and this information is used to alter the x and y coordinate listing and also the element nodal connections. The following steps indicate how procedure COINCID operates and correspond to the flowchart overleaf:

1. If the variable COIN is zero then the coinciding node numbers are not read and the procedure continues as normal.
2. The control loops I and J are constructed. The I loop limits being from 1 to the total number of redundant nodes (D). The J loop allows the coordinates to be scanned. Note that the lower limit of counter J, starts at the node number to be retained hence the redundant node number selected is assumed to be of a greater value.
3. The inner loop C checks the coinciding node numbers, and bypasses the scanning operation if the node number under consideration, corresponds to the coincidential nodes. Also if COIN is zero this step is bypassed.
4. The scanning operation is performed within the control loops I and J, it simply compares the difference in the x and y coordinates and if the results lie within a certain tolerance then the redundant node number is assumed to have been found.
5. If in the scanning process the corresponding redundant node is not found then it is assumed that it does not exist and records the redundant node as being the same as the retained node number.

Flowchart for Procedure Coincid:-

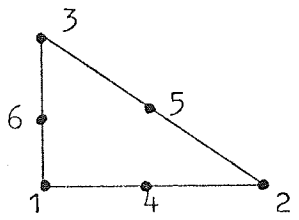


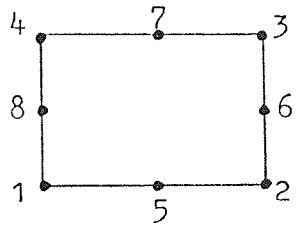

ABS(XX[NUM[I,1]]-XX[J])
1.0&-6
'AND'
ABS(YY[NUM[I,2]]-YY[J])
1.0&-6
YES
NO
NUM[I,2]:=J
END J
NUM[I,2]:=NUM[I,1]
END I
END
4.
5.

```
'PROCEDURE' COINCID(NUM,XX,YY);
'INTEGER''ARRAY'NUM;
'REAL''ARRAY'XX,YY;
'BEGIN''INTEGER'COIN;'INTEGER''ARRAY'CID[1:10,1:2];
    COIN:=READ;
    'IF'COIN=0'THEN''GOTO'L16;
      'FOR'I:=1'STEP'1'UNTIL'COIN'DO''BEGIN'
      CID[I,1]:=READ; CID[I,2]:=READ;'END';

  L16:
'FOR'I:=1'STEP'1'UNTIL'D'DO''BEGIN'
'FOR'J:=NUM[I,1]+1'STEP'1'UNTIL'NNODE+SUM'DO''BEGIN'
   'FOR'C:=1'STEP'1'UNTIL'COIN'DO''BEGIN'
       'IF'NUM[I,1]=CID[C,1]'THEN''BEGIN'
            NUM[I,2]:=CID[C,2]; 'GOTO'L7;'END';
                          'END';
'IF'ABS(XX[NUM[I,1]]-XX[J])<1.0&-6'AND'
ABS(YY[NUM[I,1]]-YY[J])<1.0&-6'THEN''BEGIN'NUM[I,2]:=J;'GOTO'L7;'END
'END' OF J LOOP;NUM[I,2]:=NUM[I,1];
L7:'END' OF I LOOP;
                  'END' OF PROCEDURE COINCID;
```

PROCEDURE QELENCONS AND TRELENCONS:

These procedures operate in the same fashion and determine the element nodal connections for the quadrilateral and triangular finite elements. The elements are defined by six nodes in the triangular element and eight nodes in the quadrilateral element. The numbering system is shown below.





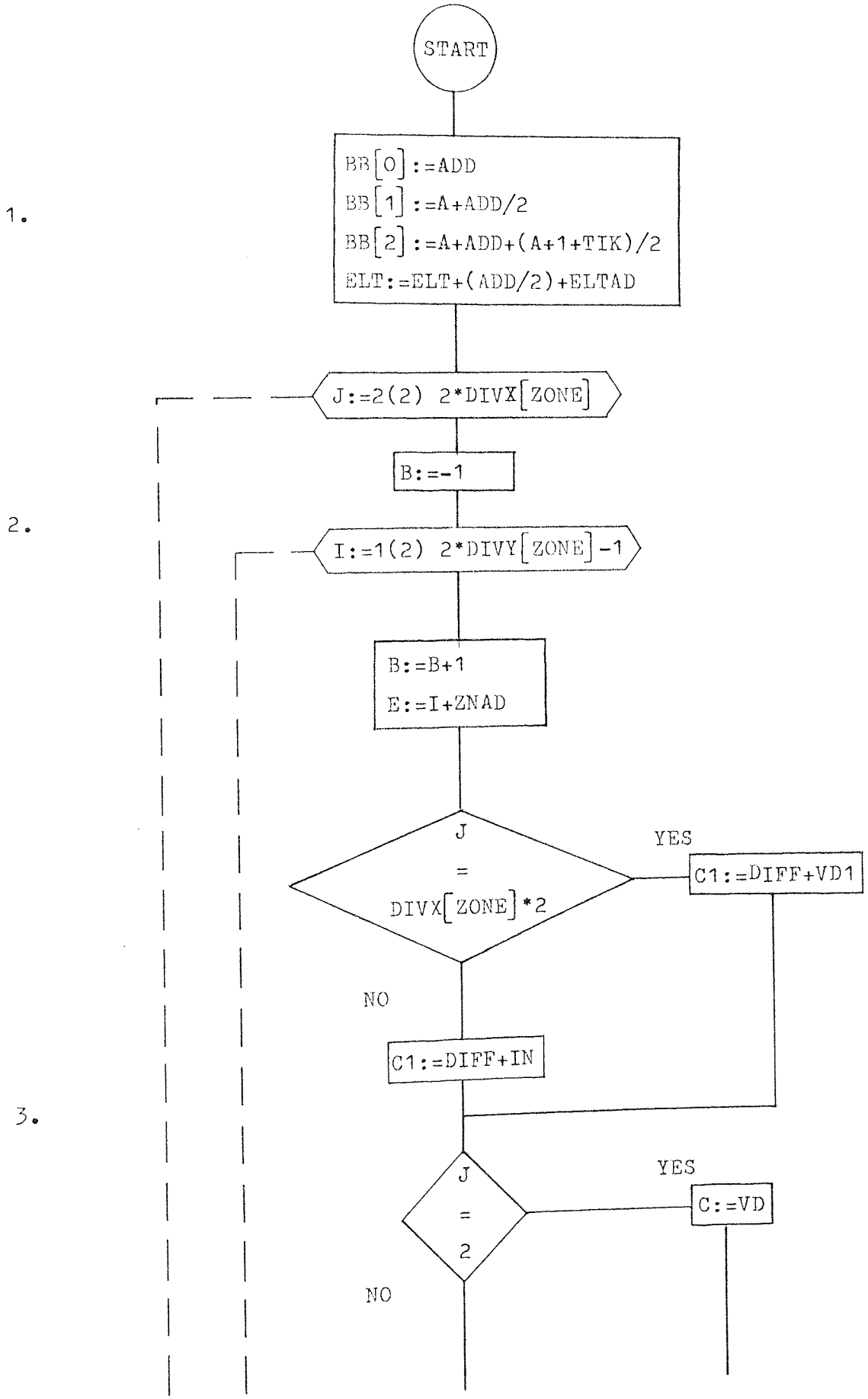
The numbering sequence is important and is used to identify the shape of the element, via its nodal coordinates. The element nodal connections are determined using the numbering scheme, referred to earlier and a similar system is used in numbering the elements. These procedures also interpolate the remaining nodal coordinates, which are determined at this stage to avoid boundary curves propagating into the mesh. Also, in the triangular element the position of the diagonal affects the element nodal numbering and the diagonal nodes coordinates.

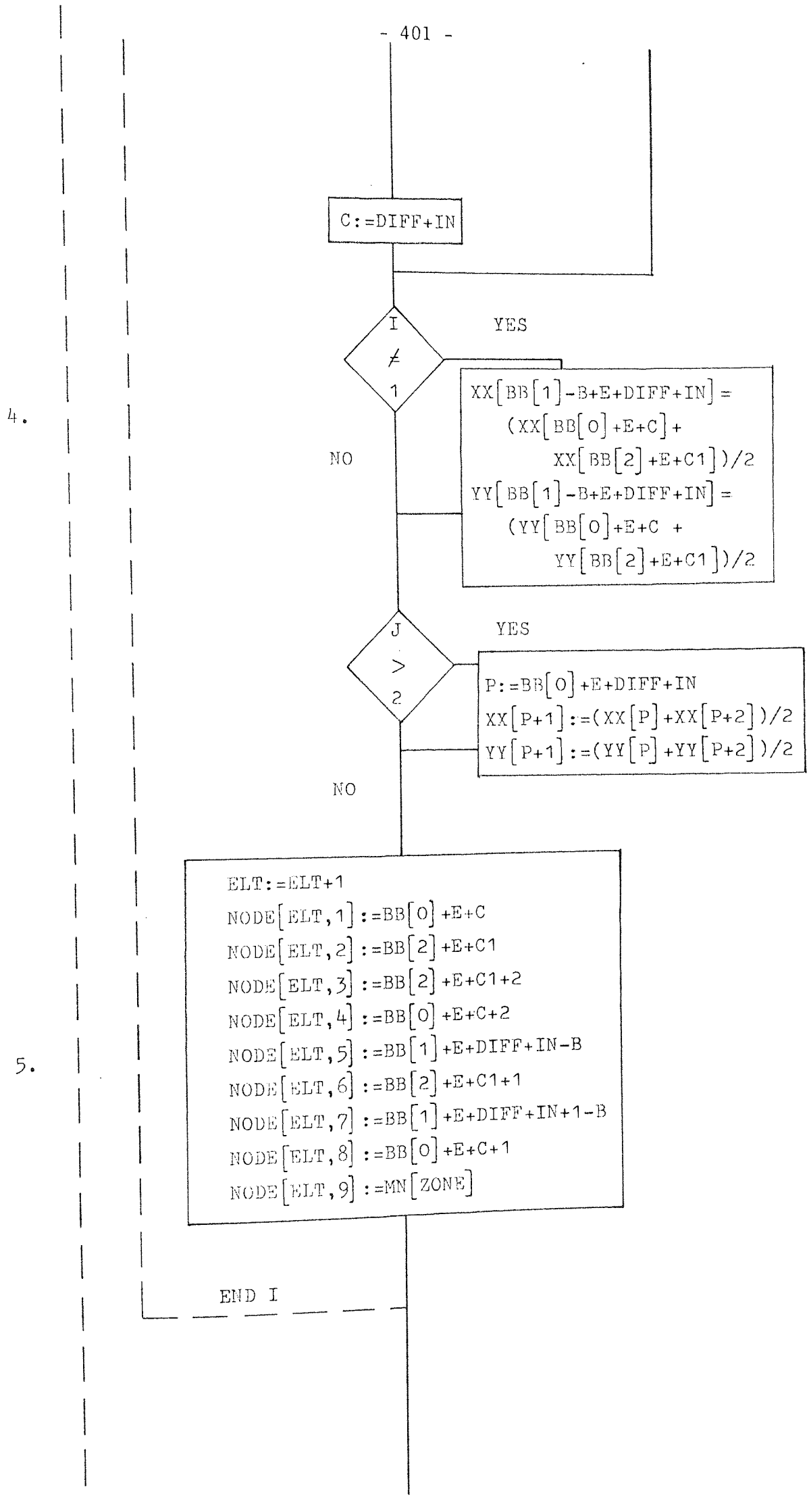
The following steps correspond to the flowchart for procedure QELENCONS:-

1. The vector BB is initialised and is used later to determine the node numbering sequence. The first element number is computed, being incremented accordingly within the procedure.

2. The counter loops J and I are constructed allowing each element to be scanned from column to column.
3. Depending on the value of the J counter, the variables C and C1 are found and form a part in the numbering scheme.
4. To prevent curves boundaries propagating into the inner mesh some of the node coordinates are not computed. See the section on errors in distorted isoparametric elements. It is convenient to calculate these coordinates within this procedure by interpolation.
5. The nodal element connections are held in array NODE and the appropriate node numbers are presented to the array via the numbering scheme.
6. At the end of each column of elements within the zone, the vector BB is incremented together with the element number.

Flowchart for Procedure Qelncons:-

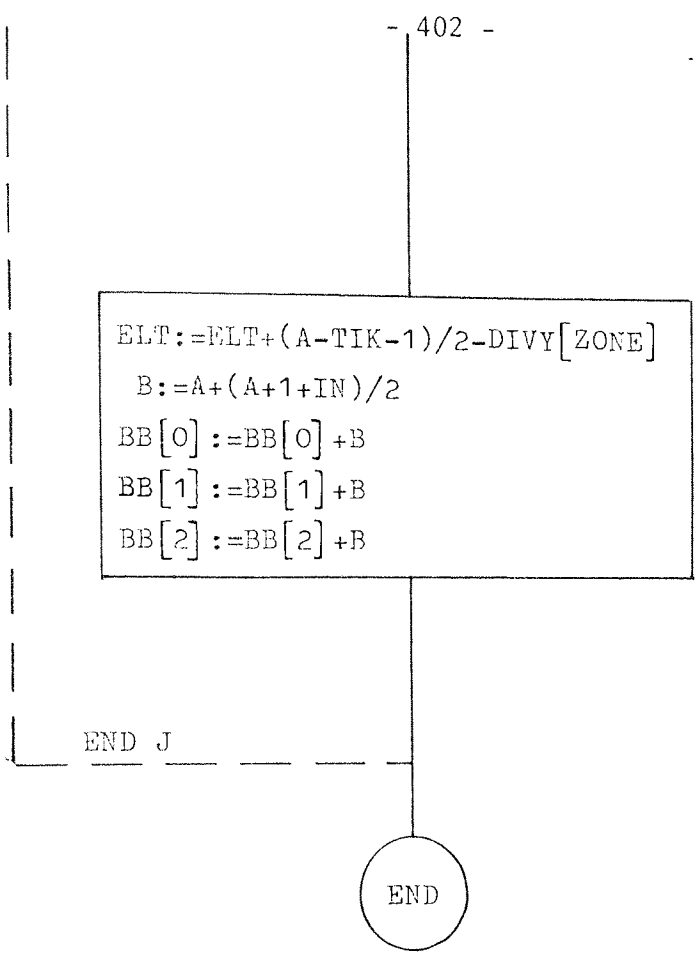



C:=DIFF+IN
I ≠ 1
YES
NO
XX[BB[1]-B+E+DIFF+IN] =
(XX[BB[0]+E+C] +
XX[BB[2]+E+C1])/2
YY[BB[1]-B+E+DIFF+IN] =
(YY[BB[0]+E+C +
YY[BB[2]+E+C1])/2
J > 2
YES
NO
P:=BB[0]+E+DIFF+IN
XX[P+1]:=(XX[P]+XX[P+2])/2
YY[P+1]:=(YY[P]+YY[P+2])/2
ELT:=ELT+1
NODE[ELT,1]:=BB[0]+E+C
NODE[ELT,2]:=BB[2]+E+C1
NODE[ELT,3]:=BB[2]+E+C1+2
NODE[ELT,4]:=BB[0]+E+C+2
NODE[ELT,5]:=BB[1]+E+DIFF+IN-B
NODE[ELT,6]:=BB[2]+E+C1+1
NODE[ELT,7]:=BB[1]+E+DIFF+IN+1-B
NODE[ELT,8]:=BB[0]+E+C+1
NODE[ELT,9]:=MN[ZONE]
END I

4.

5.

6.

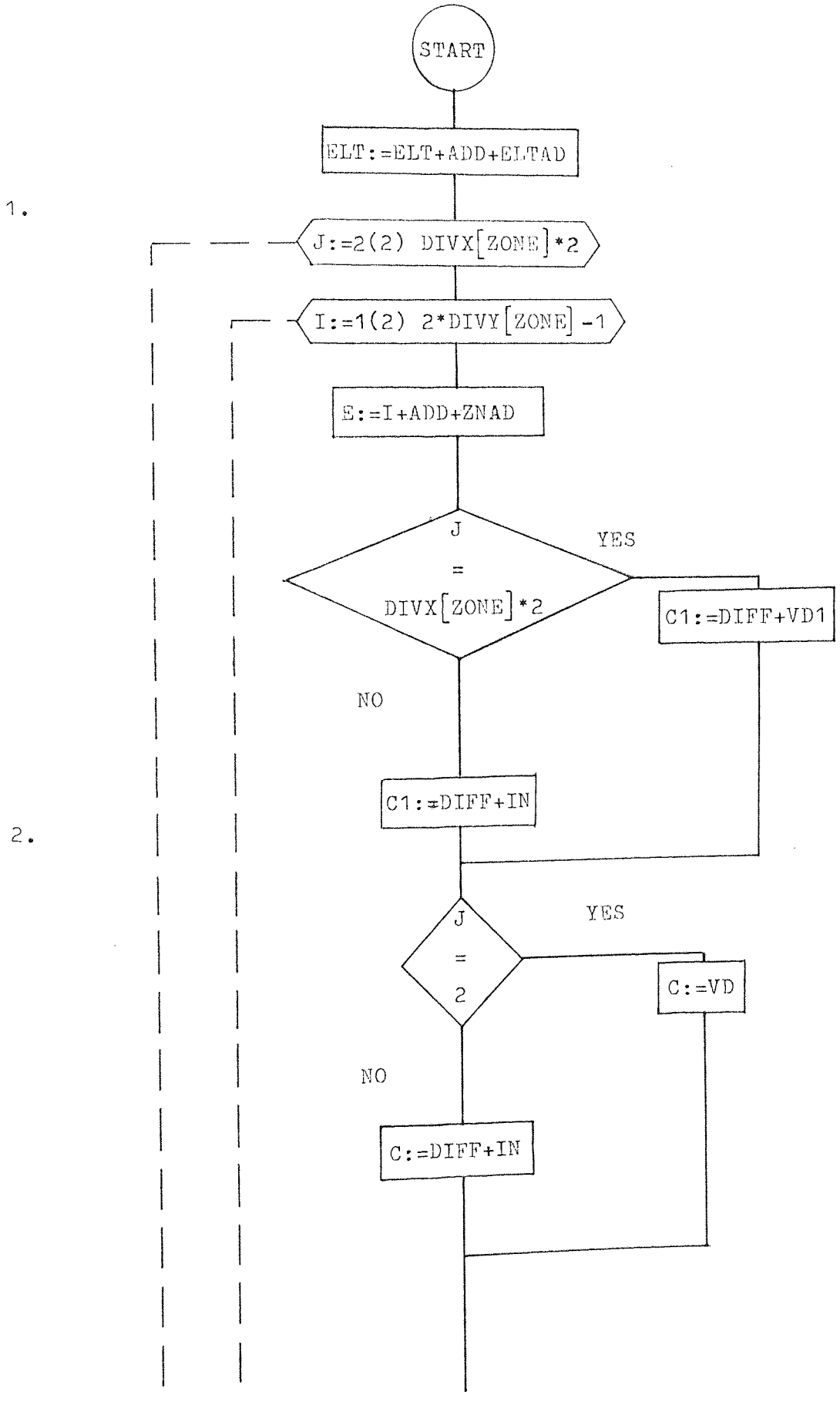

ELT:=ELT+(A-TIK-1)/2-DIVY[ZONE]
B:=A+(A+1+IN)/2
BB[0]:=BB[0]+B
BB[1]:=BB[1]+B
BB[2]:=BB[2]+B
END J
END

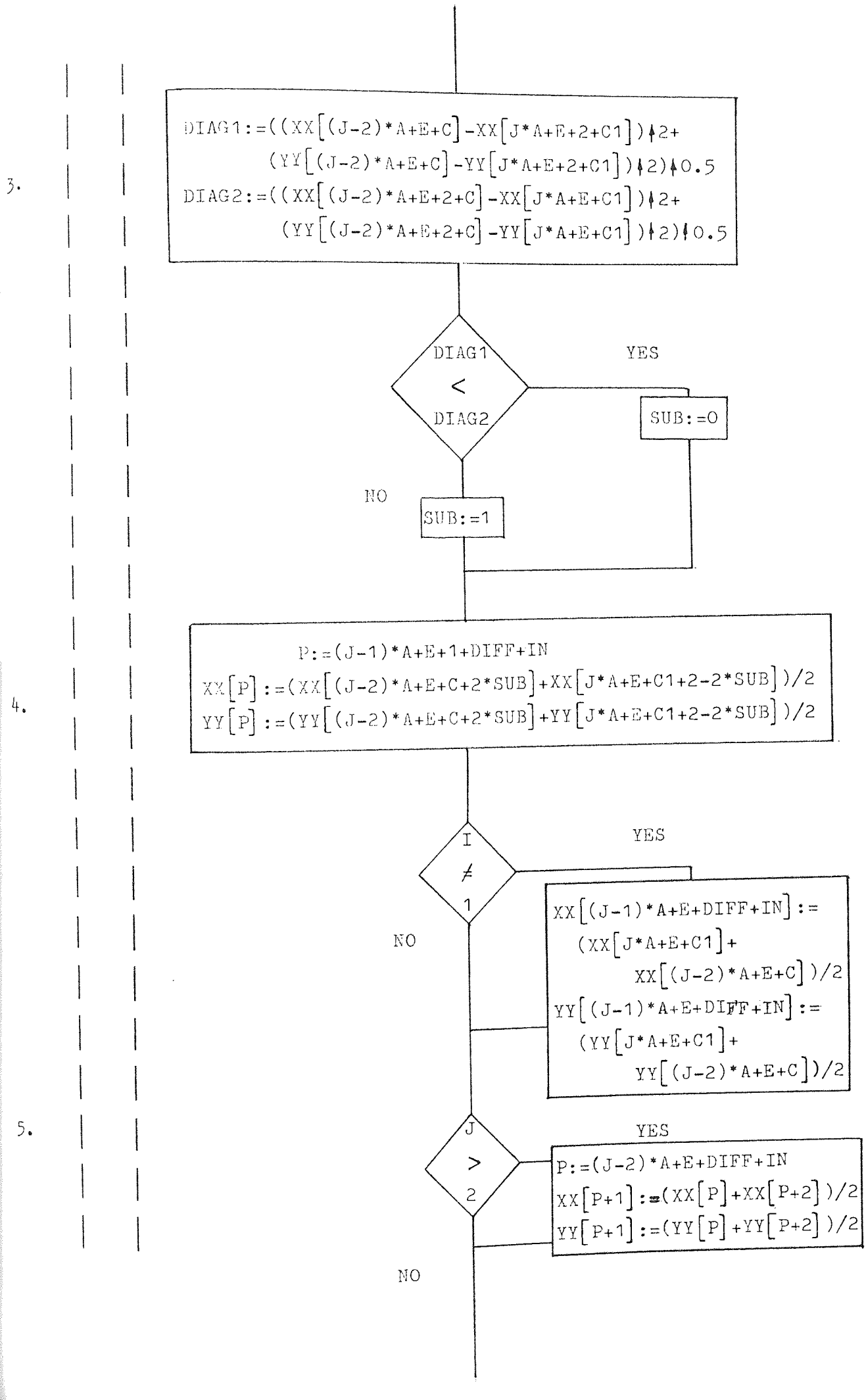
```
'PROCEDURE' QELNCONS(DIVX,DIVY,XX,YY,NODE,MN);
'INTEGER''ARRAY'DIVX,DIVY,NODE,MN;
'REAL''ARRAY'XX,YY;
'BEGIN'
   BB[0]:=ADD;  BB[1]:=A+ADD/2;  BB[2]:=A+ADD+(A+1+TIK)/2;
   ELT:=ELT+(ADD/2)+ELTAD;
'FOR'J:=2'STEP'2'UNTIL'DIVX[ZONE]*2'DO'
'BEGIN'
B:=-1;
'FOR'I:=1'STEP'2'UNTIL'(2*DIVY[ZONE]-1)'DO'
'BEGIN'
B:=B+1;
E:=I+ZNAD;
'IF'J=DIVX[ZONE]*2'THEN'C1:=DIFF+VD1
'ELSE'C1:=DIFF+IN;
'IF'J=2'THEN'C:=VD'ELSE'C:=DIFF+IN;
  'IF'I'NE'1'THEN''BEGIN'
   XX[BB[1]-B+E+DIFF+IN]:=(XX[BB[0]+E+C]+XX[BB[2]+E+C11)/2;
   YY[BB[1]-B+E+DIFF+IN]:=(YY[BB[0]+E+C]+YY[BB[2]+E+C11)/2;
   'END';
   'IF'J>2'THEN''BEGIN'
       P:=BB[0]+E+DIFF+IN;
          XX[P+1]:=(XX[P]+XX[P+2])/2;
          YY[P+1]:=(YY[P]+YY[P+2])/2;
                   'END';
ELT:=ELT+1;
NODE[ELT,1]:=BB[0]+E+C;
NODE[ELT,2]:=BB[2]+E+C1;
NODE[ELT,3]:=BB[2]+E+C1+2;
NODE[ELT,4]:=BB[0]+E+C+2;
NODE[ELT,5]:=BB[1]+E+DIFF+IN-B;
NODE[ELT,6]:=BB[2]+E+C1+1;
NODE[ELT,7]:=BB[1]+E+DIFF+IN+1-B;
NODE[ELT,8]:=BB[0]+E+C+1;
NODE[ELT,9]:=MN[ZONE];
'END' OF LOOP I; ELT:=ELT+(A-TIK-1)/2-DIVY[ZONE];
B:=A+(A+1+IN)/2;
BB[0]:=BB[0]+B;  BB[1]:=BB[1]+B;  BB[2]:=BB[2]+B;
'END' OF J LOOP;
'END' OF PROD QELNCONS;
```

The procedure TRELECONS is explained by the following steps which correspond to the flowchart:

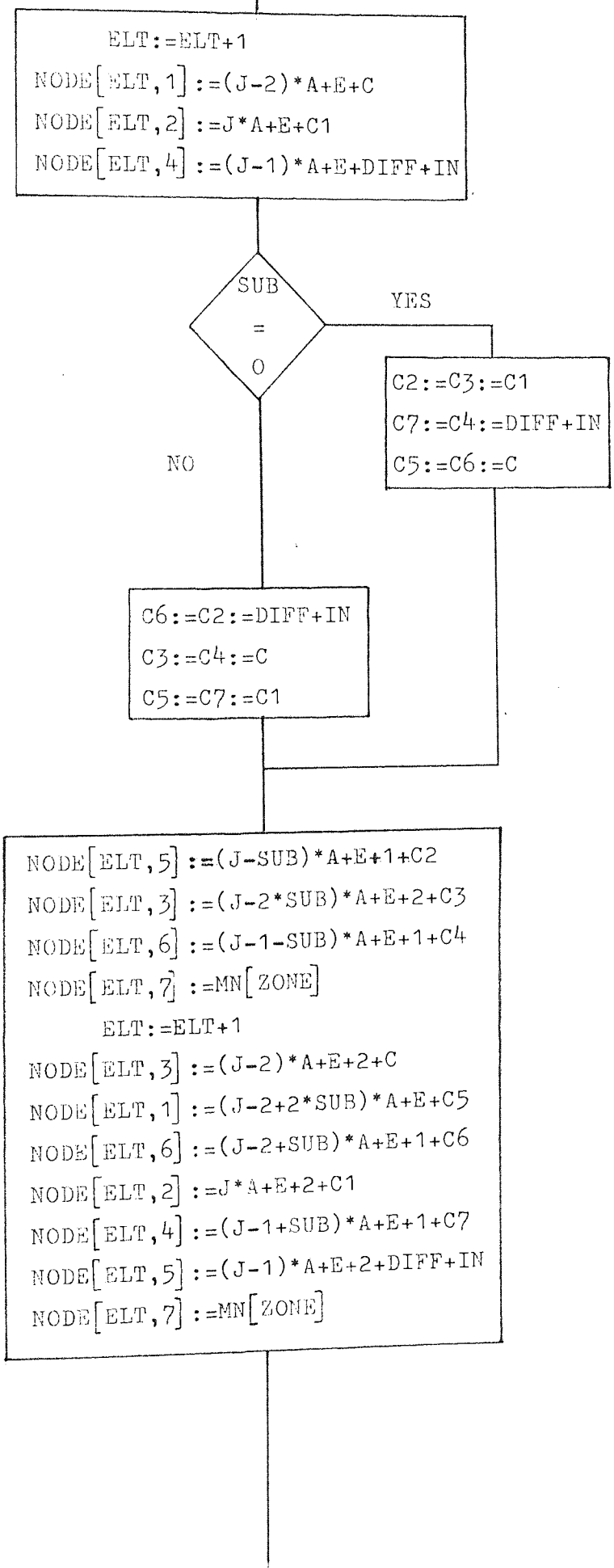
1. The element numbering variable, ELT, is initialised and loops J and I are constructed. The loops allow each element to be scanned in a column to column fashion, similar to the node numbering pattern.
2. Variables C and C1, part of the numbering scheme, are set, depending on the value of counter J.
3. This procedure determines the element nodal connections for the triangular element and to form a uniform mesh the shortest diagonal of the sub-quadrilateral is adopted. The diagonal lengths are computed as DIAG1 and DIAG2, and are used to compute the variable SUB.
4. The central node coordinates are computed using the variable SUB and node number P.
5. As before to prevent curved boundaries propagating into the inner mesh, some of the node coordinates are not formulated and it was found convenient to interpolate the remaining coordinates within this procedure.
6. Each sub-quadrilateral breaks down into two triangular elements, thus two sets of element nodal connections need to be computed. Also as the shortest diagonal, splitting the sub-quadrilateral, is chosen, the element nodal connections have to be adjusted accordingly. Hence the variables C2----C7 are set depending on the parameter SUB.
7. The element number ELT is incremented at the end of each column within the zone.

Flowchart for Procedure Trelncons:-

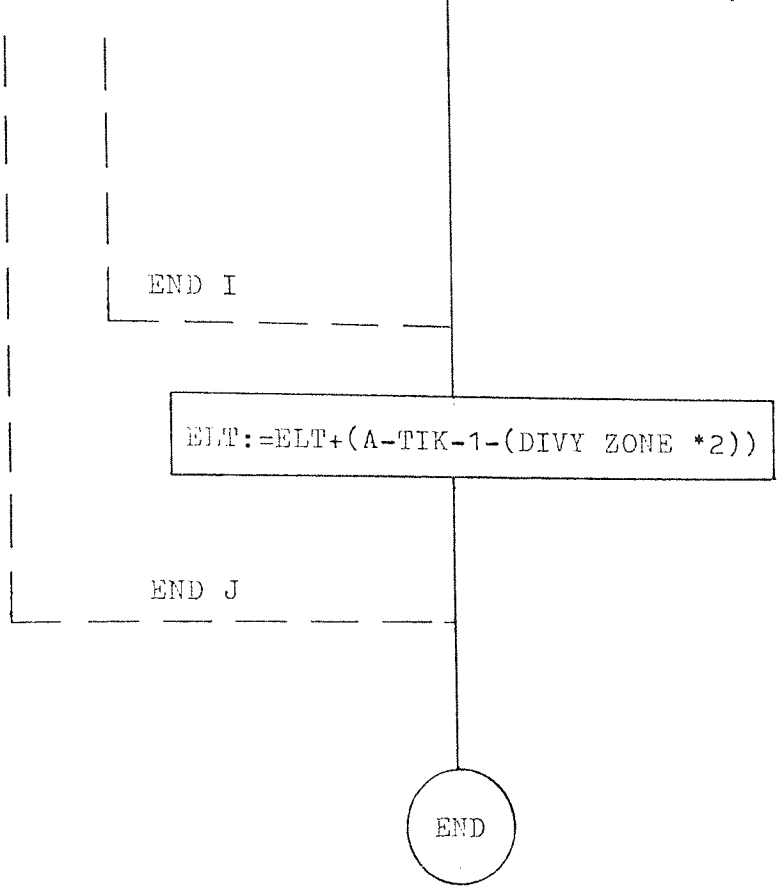



3.
DIAG1:=((XX[(J-2)*A+E+C]-XX[J*A+E+2+C1])↑2+
(YY[(J-2)*A+E+C]-YY[J*A+E+2+C1])↑2)↑0.5
DIAG2:=((XX[(J-2)*A+E+2+C]-XX[J*A+E+C1])↑2+
(YY[(J-2)*A+E+2+C]-YY[J*A+E+C1])↑2)↑0.5
DIAG1
<
DIAG2
YES
SUB:=0
NO
SUB:=1
4.
P:=(J-1)*A+E+1+DIFF+IN
XX[P]:=(XX[(J-2)*A+E+C+2*SUB]+XX[J*A+E+C1+2-2*SUB])/2
YY[P]:=(YY[(J-2)*A+E+C+2*SUB]+YY[J*A+E+C1+2-2*SUB])/2
I
≠
1
YES
NO
XX[(J-1)*A+E+DIFF+IN]:=
(XX[J*A+E+C1]+
XX[(J-2)*A+E+C])/2
YY[(J-1)*A+E+DIFF+IN]:=
(YY[J*A+E+C1]+
YY[(J-2)*A+E+C])/2
5.
J
>
2
YES
NO
P:=(J-2)*A+E+DIFF+IN
XX[P+1]:=(XX[P]+XX[P+2])/2
YY[P+1]:=(YY[P]+YY[P+2])/2

6.


ELT:=ELT+1
NODE[ELT,1]:=(J-2)*A+E+C
NODE[ELT,2]:=J*A+E+C1
NODE[ELT,4]:=(J-1)*A+E+DIFF+IN
SUB = 0
YES
NO
C2:=C3:=C1
C7:=C4:=DIFF+IN
C5:=C6:=C
C6:=C2:=DIFF+IN
C3:=C4:=C
C5:=C7:=C1
NODE[ELT,5]:=(J-SUB)*A+E+1+C2
NODE[ELT,3]:=(J-2*SUB)*A+E+2+C3
NODE[ELT,6]:=(J-1-SUB)*A+E+1+C4
NODE[ELT,7]:=MN[ZONE]
ELT:=ELT+1
NODE[ELT,3]:=(J-2)*A+E+2+C
NODE[ELT,1]:=(J-2+2*SUB)*A+E+C5
NODE[ELT,6]:=(J-2+SUB)*A+E+1+C6
NODE[ELT,2]:=J*A+E+2+C1
NODE[ELT,4]:=(J-1+SUB)*A+E+1+C7
NODE[ELT,5]:=(J-1)*A+E+2+DIFF+IN
NODE[ELT,7]:=MN[ZONE]

7.



```
'PROCEDURE' TREINCONS(DIVX,DIVY,XX,YY,NODE,IN);
'INTEGER''ARRAY'DIVX,DIVY,NODE,IN;
'REAL''ARRAY'XX,YY;
'BEGIN'
ELT:=ELT+ADD+ELTAD;
'FOR'J:=2'STEP'2'UNTIL'DIVX[ZONE]*2'DO'
'BEGIN'
'FOR'I:=1'STEP'2'UNTIL'(2*DIVY[ZONE]-1)'DO'
'BEGIN'
E:=I+ADD+ZNAD;
'IF'J=DIVX[ZONE]*2'THEN'C1:=DIFF+VD1
'ELSE'C1:=DIFF+IN;
'IF'J=2'THEN'C:=VD'ELSE'C:=DIFF+IN;
DIAG1:=((XX[(J-2)*A+E+C]-XX[J*A+E+2+C1])↑2+
  (YY[(J-2)*A+E+C]-YY[J*A+E+2+C1])↑2)↑0.5;
DIAG2:=((XX[(J-2)*A+E+2+C]-XX[J*A+E+C1])↑2+
 (YY[(J-2)*A+E+2+C]-YY[J*A+E+C1])↑2)↑0.5;
'IF'DIAG1'LE'DIAG2'THEN'SUB:=0 'ELSE'SUB:=1;
P:=(J-1)*A+E+1+DIFF+IN;
XX[P]:=(XX[(J-2)*A+E+C+2*SUB]+XX[J*A+E+C1+2-2*SUB])/2;
YY[P]:=(YY[(J-2)*A+E+C+2*SUB]+YY[J*A+E+C1+2-2*SUB])/2;
  'IF'I'NE'1'THEN''BEGIN'
   XX[(J-1)*A+E+DIFF+IN]:=(XX[J*A+E+C1]+XX[(J-2)*A+E+C1)/2;
   YY[(J-1)*A+E+DIFF+IN]:=(YY[J*A+E+C1]+YY[(J-2)*A+E+C1)/2;
   'END';
   'IF'J>2'THEN''BEGIN'
       P:=(J-2)*A+E+DIFF+IN;
          XX[P+1]:=(XX[P]+XX[P+2])/2;
          YY[P+1]:=(YY[P]+YY[P+2])/2;
                    'END';
ELT:=ELT+1;
NODE[ELT,1]:=(J-2)*A+E+C;
NODE[ELT,2]:=J*A+E+C1;
NODE[ELT,4]:=(J-1)*A+E+DIFF+IN;
'IF'SUB=0'THEN'
'BEGIN'C2:=C3:=C1;C7:=C4:=DIFF+IN;
       C5:=C6:=C;
'END'
          'ELSE'
'BEGIN'C6:=C2:=DIFF+IN;C3:=C4:=C;
        C5:=C7:=C1;
'END';
NODE[ELT,5]:=(J-SUB)*A+E+1+C2;
NODE[ELT,3]:=(J-2*SUB)*A+E+2+C3;
NODE[ELT,6]:=(J-1-SUB)*A+E+1+C4;
NODE[ELT,7]:=MN[ZONE];
ELT:=ELT+1;
NODE[ELT,3]:=(J-2)*A+E+2+C;
NODE[ELT,1]:=(J-2+2*SUB)*A+E+C5;
NODE[ELT,6]:=(J-2+SUB)*A+E+1+C6;
NODE[ELT,2]:=J*A+E+2+C1;
NODE[ELT,4]:=(J-1+SUB)*A+E+1+C7;
NODE[ELT,5]:=(J-1)*A+E+2+DIFF+IN;
NODE[ELT,7]:=MN[ZONE];
'END' OF LOOP I; ELT:=ELT+(A-TIK-1-(DIVY[ZONE]*2));
'END' OF J LOOP;
'END' OF PROD TREINCONS;
```

PROCEDURE TRACE:

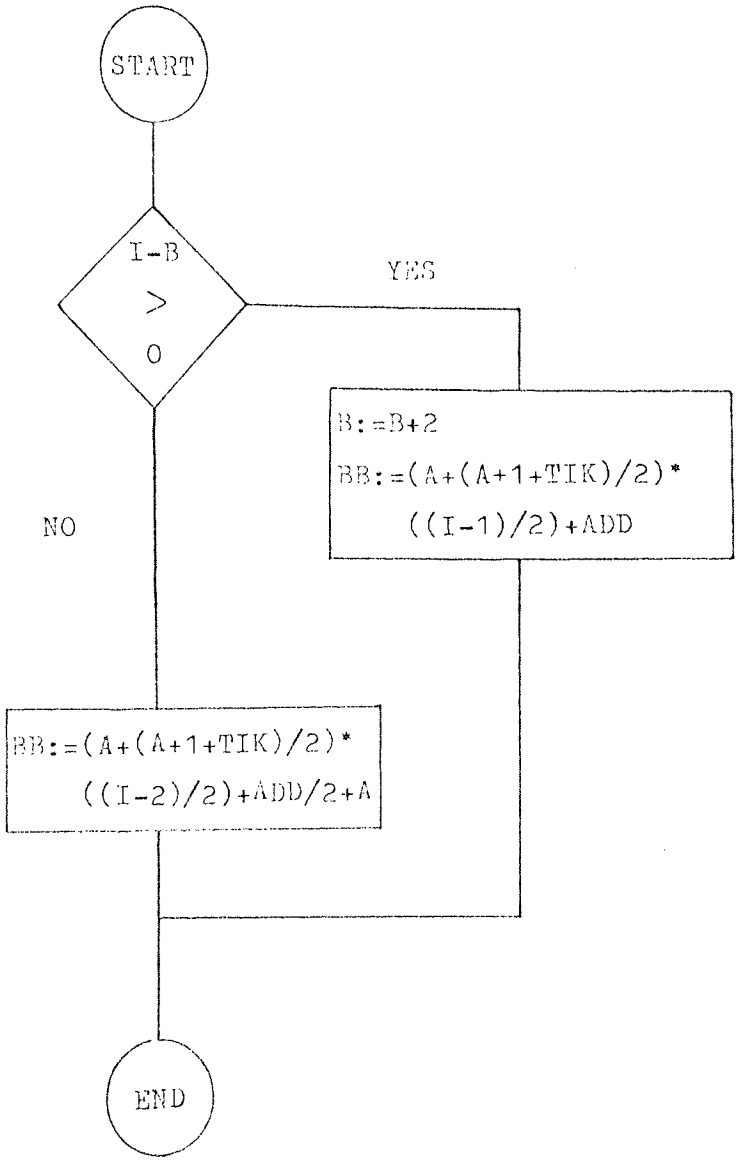
The TRACE procedure is an ancillary routine used by QELETXY and QIDN, its function is to determine the first node number starting each column in the zone, when generating quadrilateral elements.

When accessed, variables from the node numbering system, A, TIK, ADD, I and B are presented, and on exit the required node number is given by variable BB.

Procedure steps correspond to the flowchart:

1. A conditional statement is set up whereby, if (I-B) is greater than zero, the variable B is incremented by two, and the first column number of the zone is evaluated and held in BB. If the reverse is the case then BB is determined for a sparse nodal column.

Flowchart for Procedure Trace:-




```
'PROCEDURE' TRACE(BB,A,TIK,ADD,I,B);
'VALUE'A,ADD,TIK;
'INTEGER'A,TIK,ADD,I,B,BB;
'BEGIN'
'IF'(I-B)>0'THEN''BEGIN'
          B:=B+2;
          BB:=(A+(A+1+TIK)/2)*((I-1)/2)+ADD;
          'END' 'ELSE'
          BB:=A+(A+(A+1+TIK)/2)*((I-2)/2)+ADD/2;
'END' OF PROCEDURE TRACE;
```

PROCEDURES QELETXY AND TRELETXY:

Both of these procedures function in the same way and compute the x and y coordinates for each node generated within the zone under consideration. The mapping technique relates a unit square in local coordinates (ξ and η) to the quadrilateral in global coordinates (x and y), the shape of which depends on the eight super nodes given in the input data. The mapping function can be written as,

$$x = \sum_{i=1}^{8} N_i(\xi,\eta)x_i,$$

$$y = \sum_{i=1}^{8} N_i(\xi,\eta)y_i.$$

The vector $[N(\xi,\eta)]$ is the isoparametric shape function for a quadrilateral element. The mapping technique is comparatively simple and any point within the zone can be defined in global coordinates by inserting the corresponding local coordinates into the shape function $[N(\xi,\eta)]$.

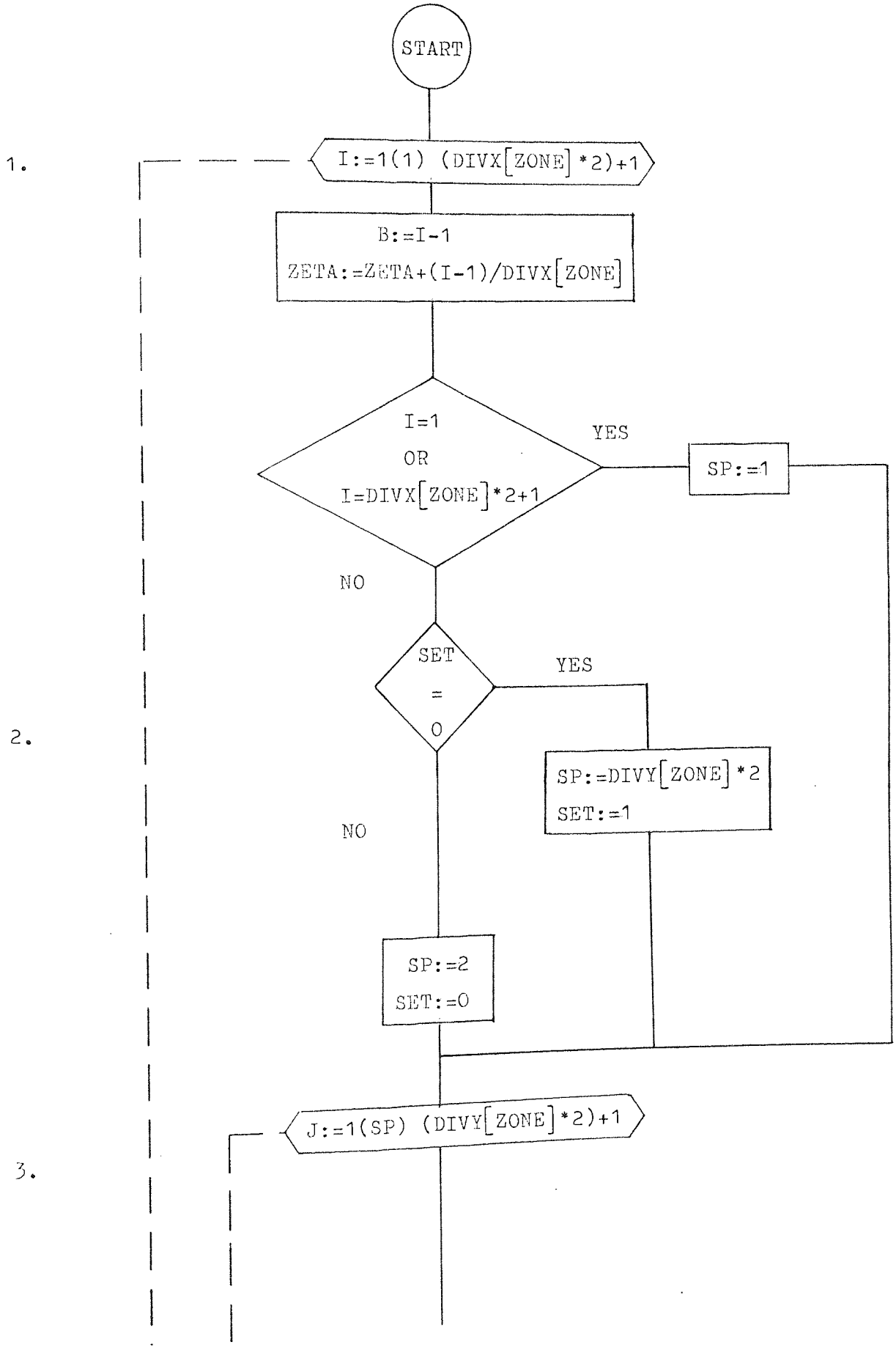
Each coordinate generated must be given a node number as part of the finite element model. The numbering scheme is explained in Chapter 5. The system is rather complicated by the fact that in meshing a real component it is convenient to make some of the zones 'void', that is no mesh generated in the zone. It can be appreciated therefore, that to have maximum flexibility in the generating process, voids can be placed anywhere in the zone array. Thus the node numbering scheme must be such that it can operate for any combination of voids that may arise. To avoid curved boundaries propagating into the inner mesh, only the finite element corner node coordinates are found, the remaining nodes being evaluated by interpolation in procedures QELNCONS and TRELNCONS.

The quadrilateral element has a different node pattern to that of the triangular element, being consistant but non-uniform. This presents problems in the node numbering scheme and a separate sub-procedure TRACE is used to determine the correct node number.

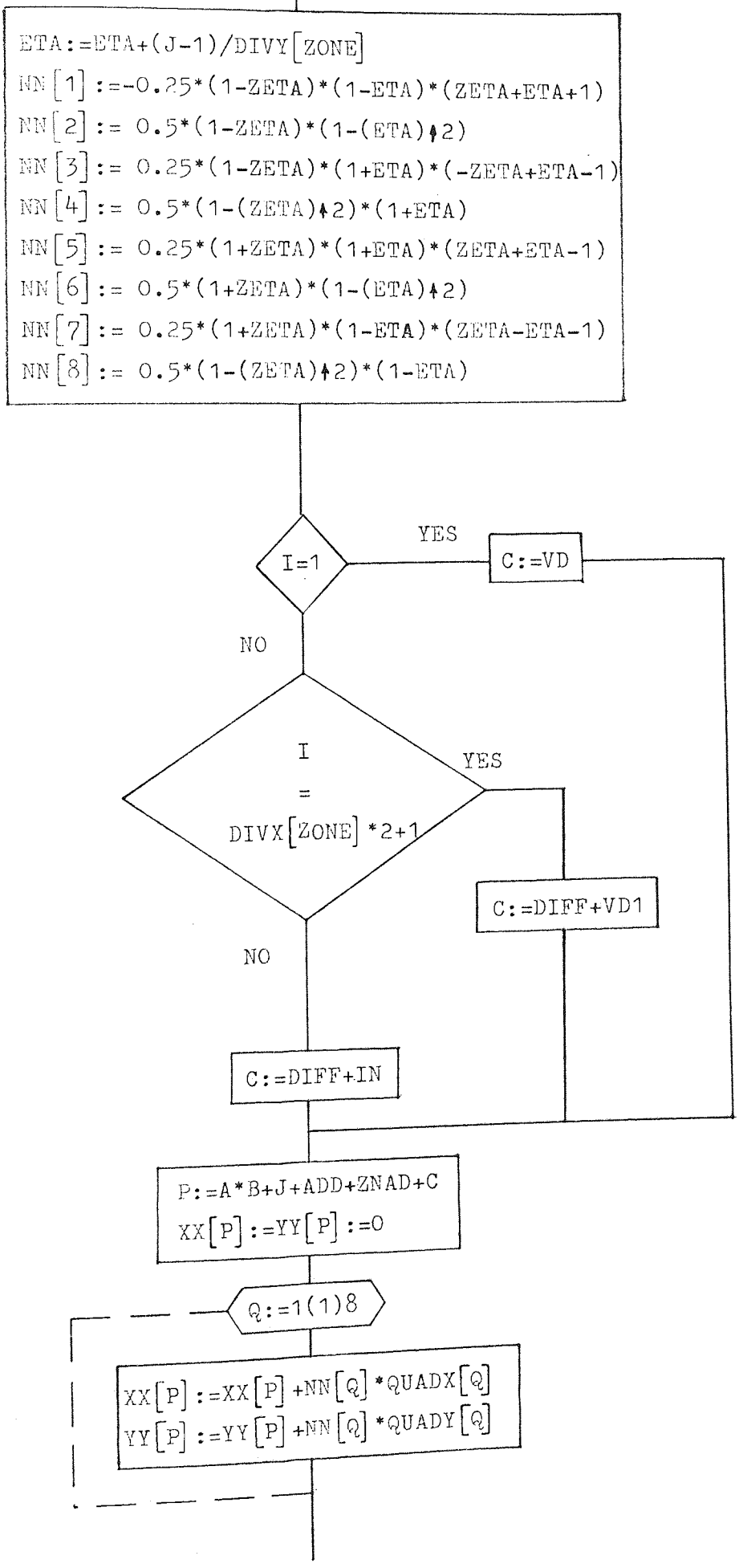
The procedure steps are as follows and apply to both QELETXY and TRELETXY:

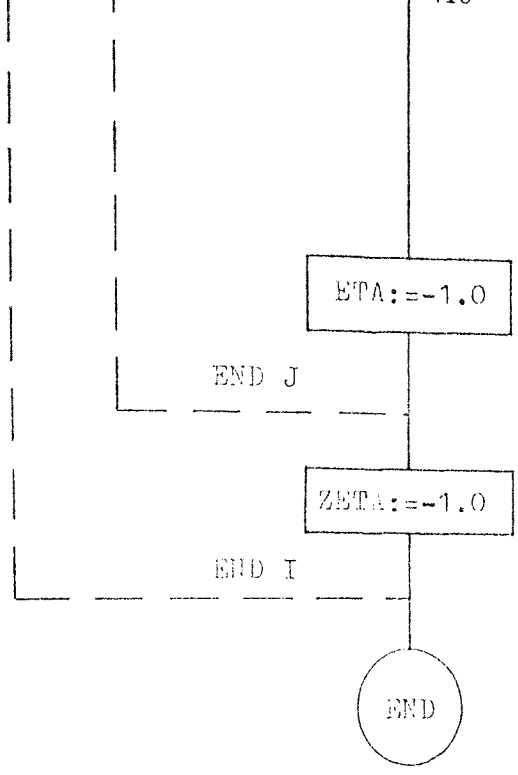
1. The counter loop I is constructed and represent the column of node points in the zone. For the quadrilateral element procedure TRACE is accessed and on exit variable BB[1] contains the first node number in the column. Similarly for the triangular element, B represents the column multiple. The local coordinate ZETA is determined.
2. The second step evaluates the variable SP and this controls the number of nodes evaluated in the column by stepping the J counter. The variable SET is used simply as a control flag.
3. The loop J is constructed, within this loop the local coordinate ETA is determined and in conjunction with the shape function $N(\xi,\eta)$ the global coordinates of each node are found for the column (I). The node number is represented by the variable P, which is controlled by the numbering scheme variables. (See Chapter 5). The loop Q evaluates the x and y coordinates, calling the individual shape function terms and zone coordinates, $N_i(\xi,\eta)$ and QUAD(x,y) respectively.

Flowchart for Procedure Treletxy:-



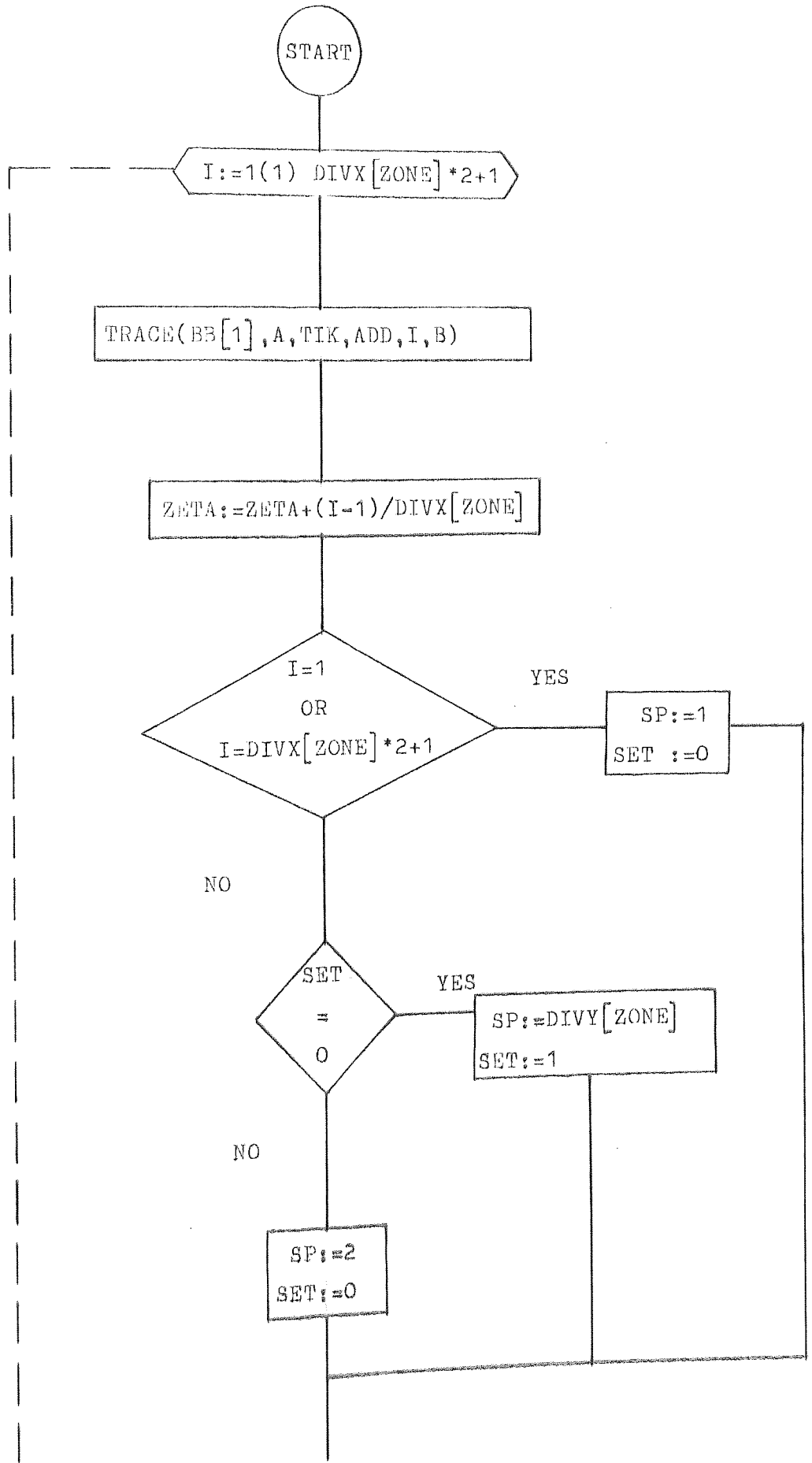
3.

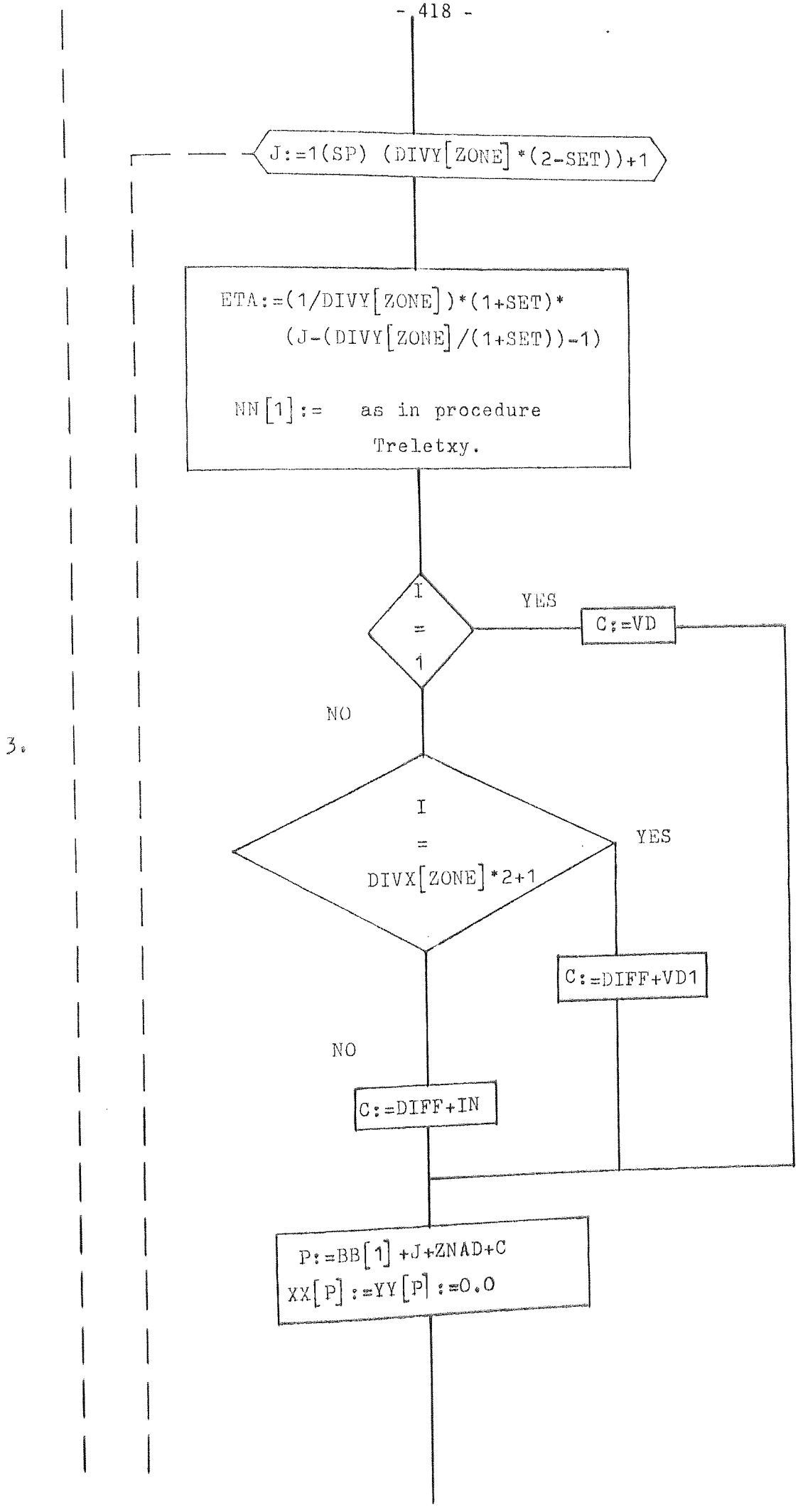

ETA:=ETA+(J-1)/DIVY[ZONE]
NN[1]:=-0.25*(1-ZETA)*(1-ETA)*(ZETA+ETA+1)
NN[2]:= 0.5*(1-ZETA)*(1-(ETA)↑2)
NN[3]:= 0.25*(1-ZETA)*(1+ETA)*(-ZETA+ETA-1)
NN[4]:= 0.5*(1-(ZETA)↑2)*(1+ETA)
NN[5]:= 0.25*(1+ZETA)*(1+ETA)*(ZETA+ETA-1)
NN[6]:= 0.5*(1+ZETA)*(1-(ETA)↑2)
NN[7]:= 0.25*(1+ZETA)*(1-ETA)*(ZETA-ETA-1)
NN[8]:= 0.5*(1-(ZETA)↑2)*(1-ETA)
I=1
YES
C:=VD
NO
I
=
DIVX[ZONE]*2+1
YES
C:=DIFF+VD1
NO
C:=DIFF+IN
P:=A*B+J+ADD+ZNAD+C
XX[P]:=YY[P]:=0
Q:=1(1)8
XX[P]:=XX[P]+NN[Q]*QUADX[Q]
YY[P]:=YY[P]+NN[Q]*QUADY[Q]

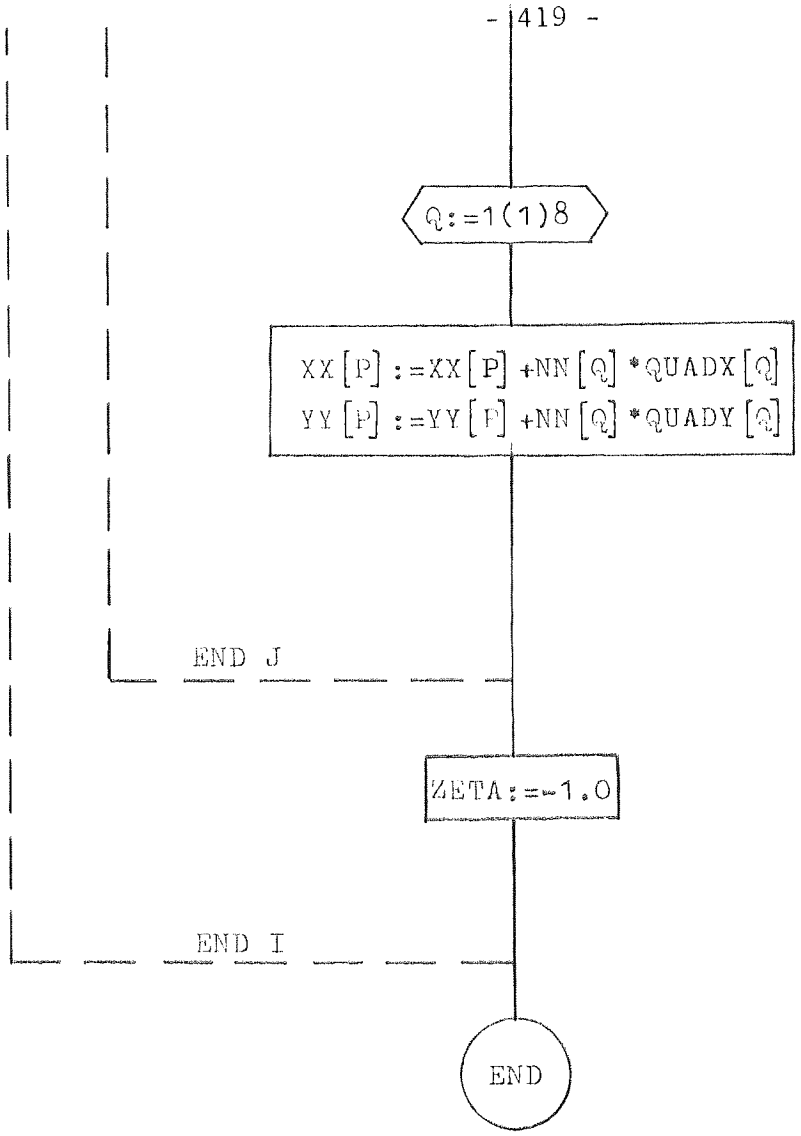



```
'PROCEDURE' TRELETXY(DIVX,DIVY,XX,YY);
'INTEGER''ARRAY'DIVX,DIVY;
'REAL''ARRAY'XX,YY;
'BEGIN'
'FOR'I:=1'STEP'1'UNTIL'(DIVX[ZONE]*2)+1'DO'
'BEGIN' R:=I-1;
ZETA:=ZETA+(I-1)/DIVX[ZONE];
      'IF'I=1'OR'I=DIVX[ZONE]*2+1
          'THEN' SP:=1
      'ELSE' 'BEGIN'
          'IF'SET=0'THEN''BEGIN'
                SP:=DIVY[ZONE]*2;
              SET:=1;  'END'
          'ELSE'
                'BEGIN'
                SP:=2; SET:=0;
            'END';
       'END';
'FOR'J:=1'STEP'SP'UNTIL'(DIVY[ZONE]*2)+1'DO'
'BEGIN'
ETA:=ETA+(J-1)/DIVY[ZONE];
NN[1]:=-0.25*(1-ZETA)*(1-ETA)*(ZETA+ETA+1);
NN[2]:=0.5*(1-ZETA)*(1-(ETA)↑2);
NN[3]:=0.25*(1-ZETA)*(1+ETA)*(-ZETA+ETA-1);
NN[4]:=0.5*(1-(ZETA)↑2)*(1+ETA);
NN[5]:=0.25*(1+ZETA)*(1+ETA)*(ZETA+ETA-1);
NN[6]:=0.5*(1+ZETA)*(1-(ETA)↑2);
NN[7]:=0.25*(1+ZETA)*(1-ETA)*(ZETA-ETA-1);
NN[8]:=0.5*(1-(ZETA)↑2)*(1-ETA);
'IF'I=1'THEN'C:=VD'ELSE''IF'I=DIVX[ZONE]*2+1'THEN'
C:=DIFF+VD'ELSE'C:=DIFF+IN;
P:=A*R+J+ADR+ZNAD+C;
XX[P]:=YY[P]:=0.0;
'FOR'Q:=1'STEP'1'UNTIL'8'DO'
'BEGIN'
XX[P]:=XX[P]+NN[Q]*QUADX[Q];
YY[P]:=YY[P]+NN[Q]*QUADY[Q];
'END'; ETA:=-1.0;
'END' OF J LOOP; ZETA:=-1.0;
'END' OF I LOOP;
'END' OF PROC TRELETXY;
```

Flowchart for Procedure Seletxy:-




J:=1(SP) (DIVY[ZONE]*(2-SET))+1
ETA:=(1/DIVY[ZONE])*(1+SET)*
(J-(DIVY[ZONE]/(1+SET))-1)
NN[1]:= as in procedure
Treletxy.
I
=
1
YES
C:=VD
NO
I
=
DIVX[ZONE]*2+1
YES
C:=DIFF+VD1
NO
C:=DIFF+IN
P:=BB[1]+J+ZNAD+C
XX[P]:=YY[P]:=0.0

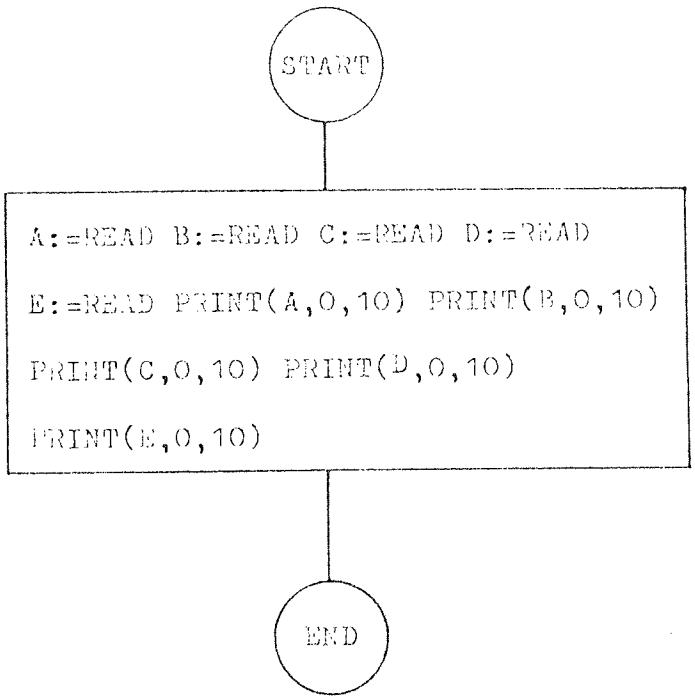

Q:=1(1)8
XX[P]:=XX[P]+NN[Q]*QUADX[Q]
YY[P]:=YY[P]+NN[Q]*QUADY[Q]
END J
ZETA:=-1.0
END I
END

```
'PROCEDURE' QFLFTXY(TRACE,DIVX,DIVY,XX,YY);
'INTEGER''ARRAY'DIVX,DIVY;
'REAL''ARRAY'XX,YY;
'PROCEDURE' TRACE;
'BEGIN'
'FOR'I:=1'STEP'1'UNTIL'(DIVX[ZONE]*2)+1'DO'
'BEGIN'
TRACE(BB[1],A,TIK,ADD,I,B);
ZETA:=ZETA+(I-1)/DIVX[ZONE];
      'IF'I=1'OR'I=DIVX[ZONE]*2+1
          'THEN''BEGIN' SP:=1;SET:=0;'END'
      'ELSE' 'BEGIN'
          'IF'SET=0'THEN''BEGIN'
                SP:=DIVY[ZONE];
              SET:=1;  'END'
          'ELSE'
                'BEGIN'
                SP:=2; SET:=0;
            'END';
       'END';
'FOR'J:=1'STEP'SP'UNTIL'(DIVY[ZONE]*(2-SET))+1'DO'
'BEGIN'
ETA:=(1/DIVY[ZONE])*(1+SET)*(J-(DIVY[ZONE]/(1+SET))-1);
NN[1]:=0.25*(1-ZETA)*(1-ETA)*(ZETA+ETA+1);
NN[2]:=0.5*(1-ZETA)*(1-(ETA)↑2);
NN[3]:=0.25*(1+ZETA)*(1+ETA)*(-ZETA+ETA-1);
NN[4]:=0.5*(1-(ZETA)↑2)*(1+ETA);
NN[5]:=0.25*(1+ZETA)*(1+ETA)*(ZETA+ETA-1);
NN[6]:=0.5*(1+ZETA)*(1-(ETA)↑2);
NN[7]:=0.25*(1+ZETA)*(1-ETA)*(ZETA-ETA-1);
NN[8]:=0.5*(1-(ZETA)↑2)*(1-ETA);
'IF'I=1'THEN'C:=VD'ELSE''IF'I=DIVX[ZONE]*2+1'THEN'
C:=DIFF+VD1'ELSE'C:=DIFF+IN;
P:=BB[1]+J+ZNAD+C;
XX[P]:=YY[P]:=0.0;
'FOR'Q:=1'STEP'1'UNTIL'8'DO'
'BEGIN'
XX[P]:=XX[P]+NN[Q]*QUADX[Q];
YY[P]:=YY[P]+NN[Q]*QUADY[Q];
'END';
'END' OF J LOOP; ZETA:=-1.0;
'END' OF I LOOP;
'END' OF PROD QFLFTXY;
```

PROCEDURE MATCON:

This procedure, as explained in procedure OUTPUT2, serves to read in and print out the material constants. As this operation is required repeatedly it has been adapted into a small sub-procedure.

Flowchart for Procedure Matcon:-




```
'PROCEDURE' MATCON(A,B,C,D,E);
  'REAL'A,B,C,D,E;
 'BEGIN'A:=READ; B:=READ; C:=READ; D:=READ; E:=READ;
   NEWLINE(1); PRINT(A,0,10); PRINT(B,0,10); PRINT(C,0,10);
  NEWLINE(1); PRINT(D,0,10); PRINT(E,0,10); NEWLINE(1);
'END'  OF PROCEDURE MATCON;
```

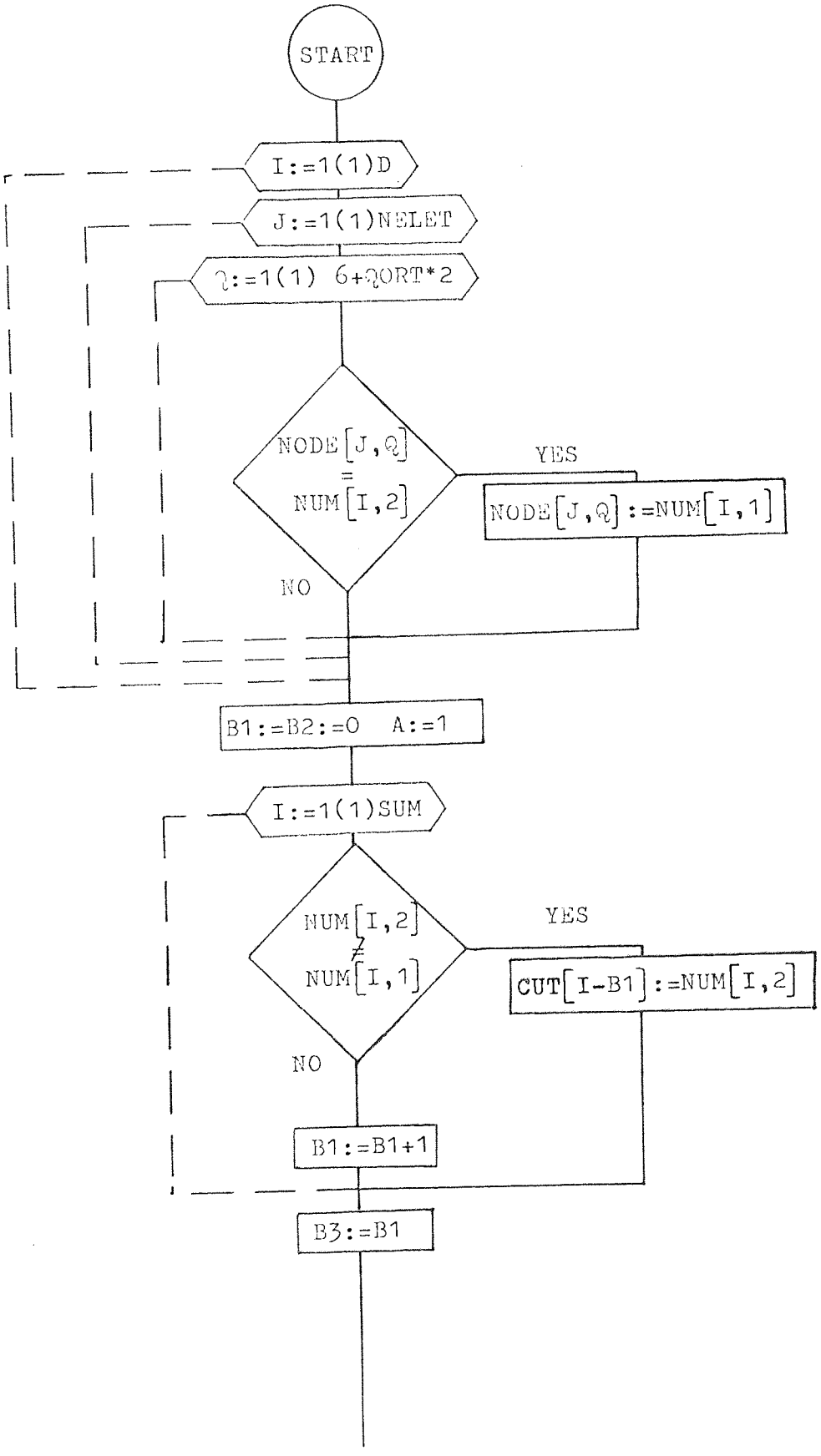
PROCEDURE HATCHET:

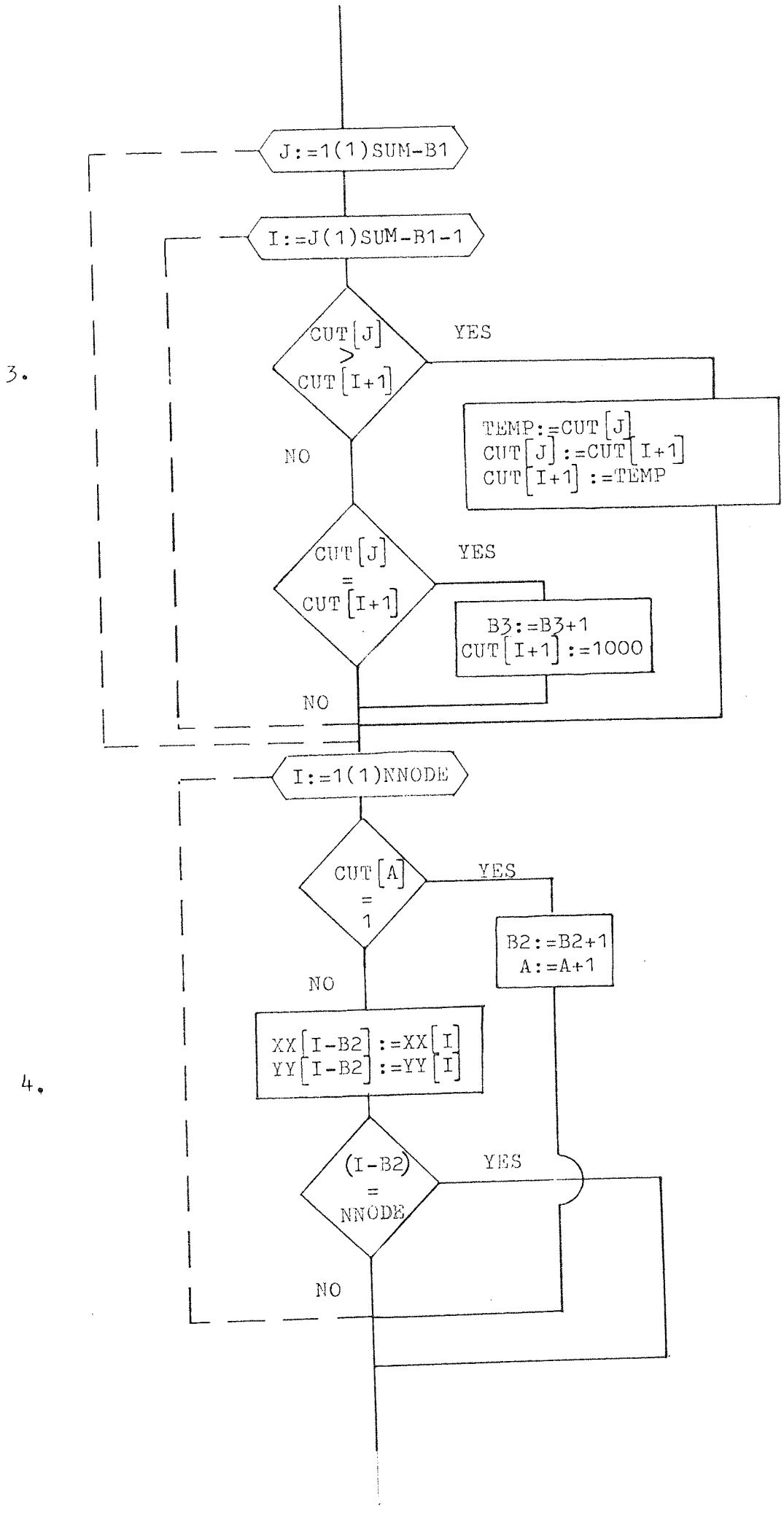
As explained in the introduction, zone faces can be joined, being originally open in the zone array. This facility causes the nodes on one of the zone faces to become redundant. The purpose of this procedure is to rearrange the element nodal connections, eradicating the redundant nodes and condensing the nodal numbering scheme. Hence if a node number becomes redundant any nodes of greater value are reduced by 1, etc. Similarly the x and y nodal coordinates are condensed, so that they correspond to the rearranged element nodal connections.

The following steps show how the procedure HATCHET operates:

1. The nested loops I, J and Q are constructed, within which the element nodal connections are rearranged. At this stage the redundant nodes are simply replaced by their counter parts, in array NUM.
2. The array NUM is modified and held in array CUT. This array now contains all the redundant node numbers excluding the node numbers which happen to be identical.
3. The array CUT is again modified so that the node numbers are in numerical order, this allows the re-addressing scheme to function correctly.
4. The nodal coordinates are condensed, using array CUT.
5. The element nodal connections are condensed, using array CUT.

Flowchart for Procedure Hatchet:-

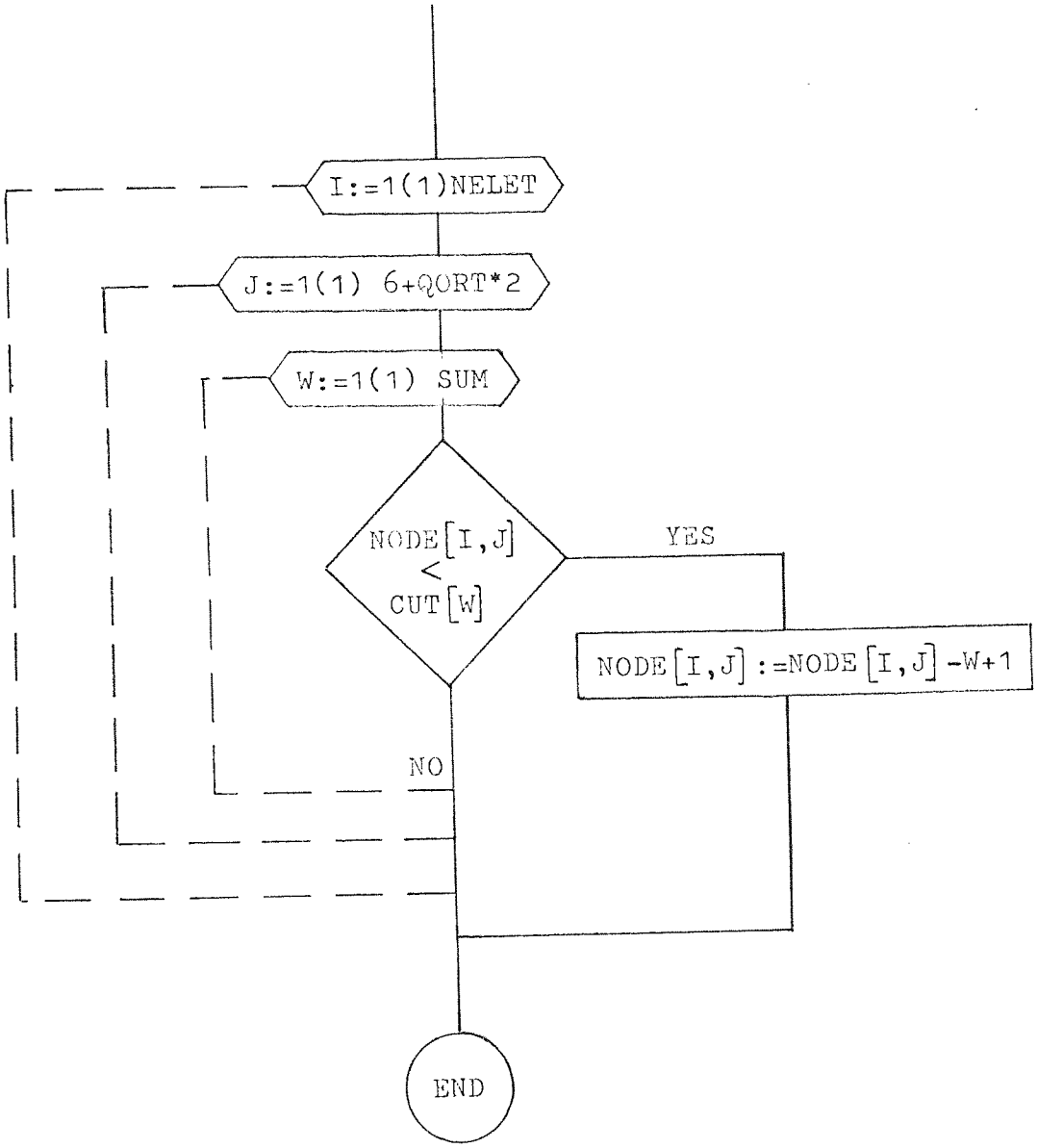



J:=1(1)SUM-B1
I:=J(1)SUM-B1-1
CUT[J] > CUT[I+1]
YES
NO
TEMP:=CUT[J]
CUT[J]:=CUT[I+1]
CUT[I+1]:=TEMP
CUT[J] = CUT[I+1]
YES
NO
B3:=B3+1
CUT[I+1]:=1000
I:=1(1)NNODE
CUT[A] = 1
YES
NO
B2:=B2+1
A:=A+1
XX[I-B2]:=XX[I]
YY[I-B2]:=YY[I]
(I-B2) = NNODE
YES
NO

3.

4.

5.


I:=1(1)NELET
J:=1(1) 6+QORT*2
W:=1(1) SUM
NODE[I,J]
<
CUT[W]
YES
NODE[I,J]:=NODE[I,J]-W+1
NO
END

```
'PROCEDURE' HATCHET(NUM,XX,YY,SUM,NNODE,NODE,NELET);
     'VALUE'SUM,NELET,NNODE; 'INTEGER'SUM,NELET,NNODE;
'INTEGER''ARRAY'NUM,NODE;  'REAL''ARRAY'XX,YY;
   'BEGIN''INTEGER'J,I,W,A,B1,B2,TEMP,B3;
          'INTEGER''ARRAY'CUT[1:SUM+1];
CUT[SUM+1]:=10000;
  'FOR'I:=1'STEP'1'UNTIL'D'DO'
   'FOR'J:=1'STEP'1'UNTIL'NELET'DO'
   'FOR'Q:=1'STEP'1'UNTIL'(6+QORT*2)'DO'
     'IF' NODE[J,Q]=NUM[I,2]'THEN'
      NODE[J,Q]:=NUM[I,1];
          B1:=B2:=0; A:=1;
'FOR'I:=1'STEP'1'UNTIL'SUM'DO'
   'IF'NUM[I,2]'NE'NUM[I,1]
          'THEN' CUT[I-B1]:=NUM[I,2] 'ELSE' B1:=B1+1;
      B3:=B1;
  'FOR'J:=1'STEP'1'UNTIL'SUM-B1'DO'
       'FOR'I:=J'STEP'1'UNTIL'SUM-B1-1'DO'
            'IF' CUT[J]>CUT[I+1] 'THEN''BEGIN'
                   TEMP:=CUT[J]; CUT[J]:=CUT[I+1]; CUT[I+1]:=TEMP;
                 'END'
               'ELSE''IF' CUT[J]=CUT[I+1]
                 'THEN''BEGIN'B3:=B3+1;CUT[I+1]:=1000;'END';
'FOR'I:=1'STEP'1'UNTIL'NNODE+SUM'DO''BEGIN'
     'IF'CUT[A]=I 'THEN'
                  'BEGIN'B2:=B2+1; A:=A+1;  'END'
     'ELSE'
          'BEGIN' XX[I-B2]:=XX[I]; YY[I-B2]:=YY[I];
            'IF'(I-B2)=NNODE 'THEN''GOTO'TU01;
          'END';  'END' OF LOOP I;
TU01;'FOR'I:=1'STEP'1'UNTIL'NELET'DO'
          'FOR'J:=1'STEP'1'UNTIL'(6+QORT*2)'DO''BEGIN'
            'FOR'W:=1'STEP'1'UNTIL'SUM+1'DO'
             'IF'NODE[I,J]<CUT[W]  'THEN' 'BEGIN'
                   NODE[I,J]:=NODE[I,J]-W+1;  'GOTO'TEX;
             'END';
TEX;'END' OF LOOP J;
 'END'  OF  PROCEDURE  HATCHET;
```

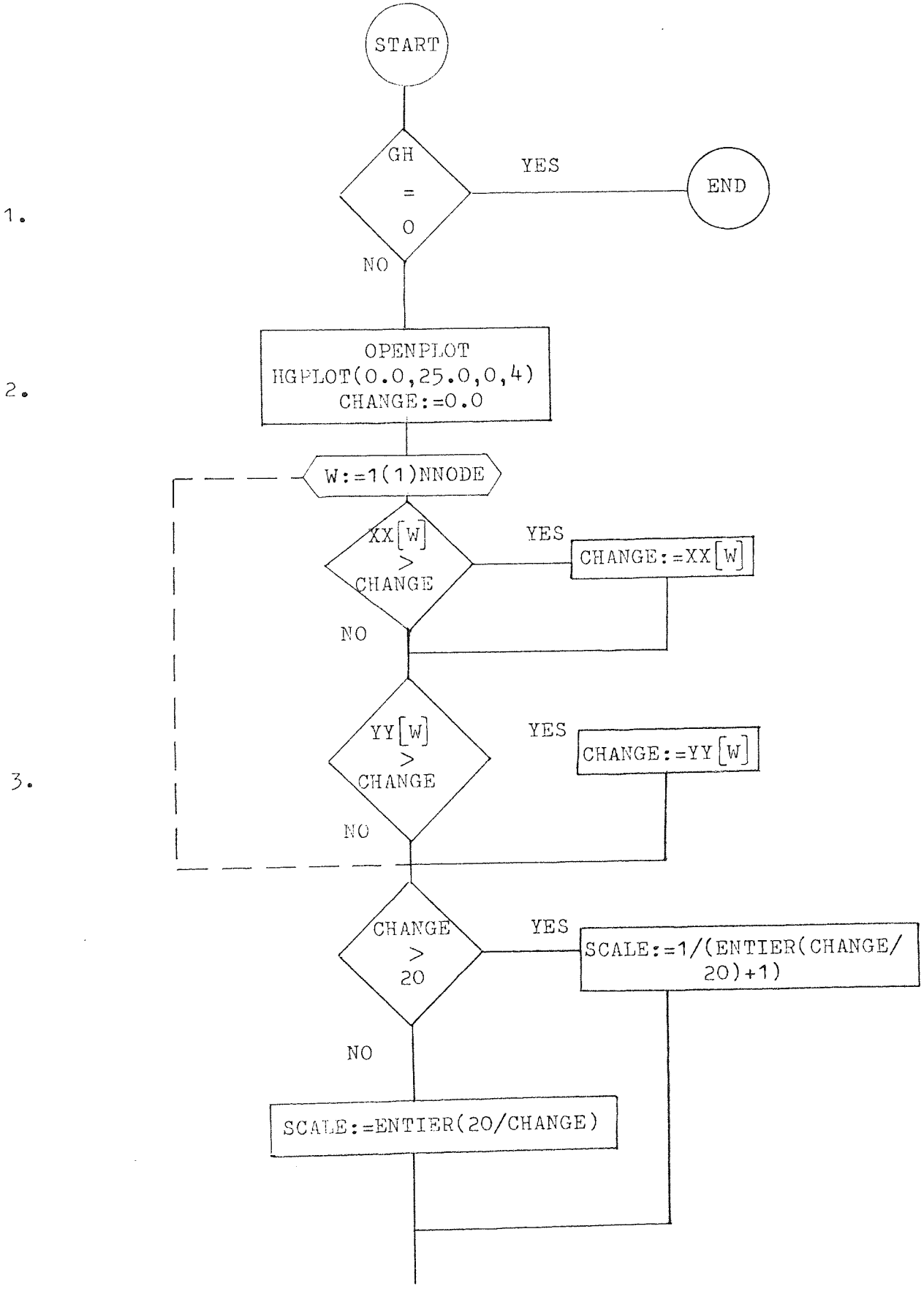
PROCEDURE RESPLOT AND SUB-PROCEDURES QPLOT AND TRPLOT

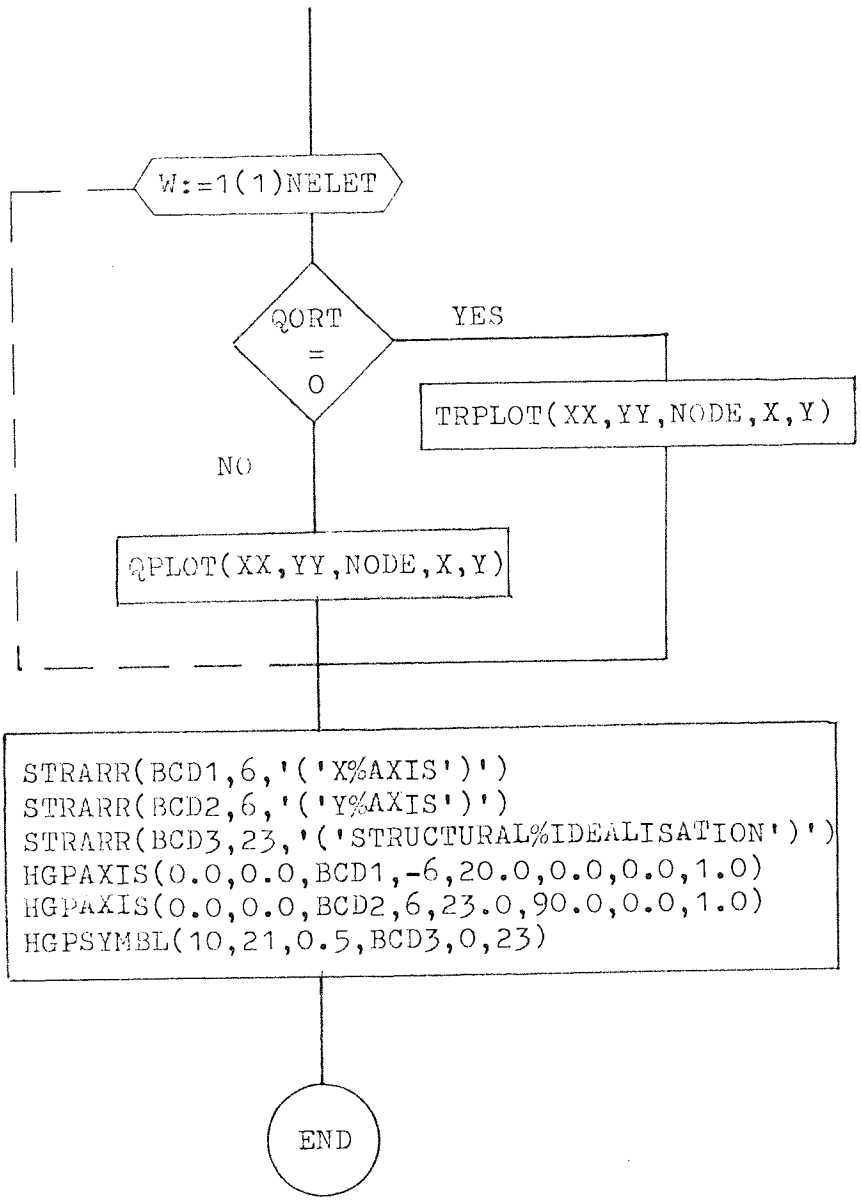
These procedures plot the final discretised component, and the graphical representation of the generated data provides a useful check on the element nodal connections and the nodal coordinates. Each element is plotted individually from its nodal connections, hence any errors would become apparent and could be traced readily.

The sub-procedures QPLOT and TRPLOT plot the quadrilateral and triangular elements respectively. The following steps describe how the plotting procedure RESPLOT and sub-proceduresQPLOT and TRPLOT function:-

1. Graphical output can be bypassed by declaring GH to be zero.
2. The plotting procedures are opened giving the size of the plotting area.
3. The scale of the graphical plot is determined by identifying the largest dimension of the structure.
4. The element plotting sub-proceduresTRPLOT and QPLOT are called depending on the control variable QORT. This variable is used throughout the mesh generation program and distinguishes between the triangular and quadrilateral elements, it is also passed onto some of the bi-element finite element programs. The sub-proceduresfunction by storing the element coordinates in arrays X and Y, which are then used in the line plotting procedure HGPLINE. This simply draws a straight line from one node to the next, hence where curved boundaries are met a simplified view is given and no attempt has been made to plot the resulting quadratic curves.
5. The last steps involve the construction of the axis and various labels.

Flowchart for Procedure Resplot :-



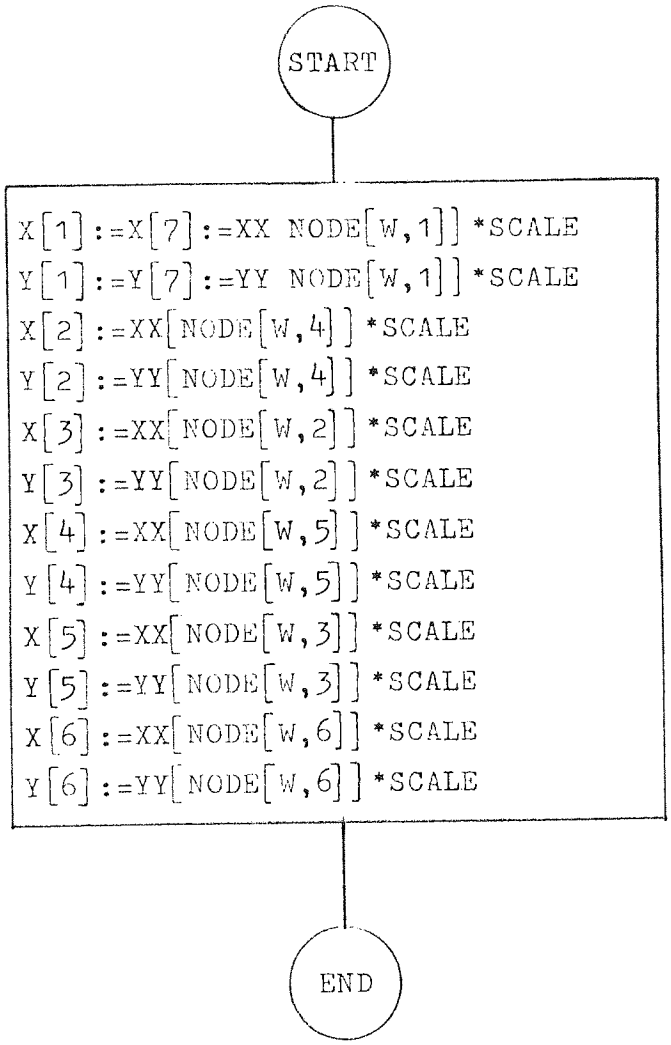

W:=1(1)NELET
QORT
=
0
YES
TRPLOT(XX,YY,NODE,X,Y)
NO
QPLOT(XX,YY,NODE,X,Y)
STRARR(BCD1,6,'('X%AXIS')')
STRARR(BCD2,6,'('Y%AXIS')')
STRARR(BCD3,23,'('STRUCTURAL%IDEALISATION')')
HGPAXIS(0.0,0.0,BCD1,-6,20.0,0.0,0.0,1.0)
HGPAXIS(0.0,0.0,BCD2,6,23.0,90.0,0.0,1.0)
HGPSYMBL(10,21,0.5,BCD3,0,23)
END

4,

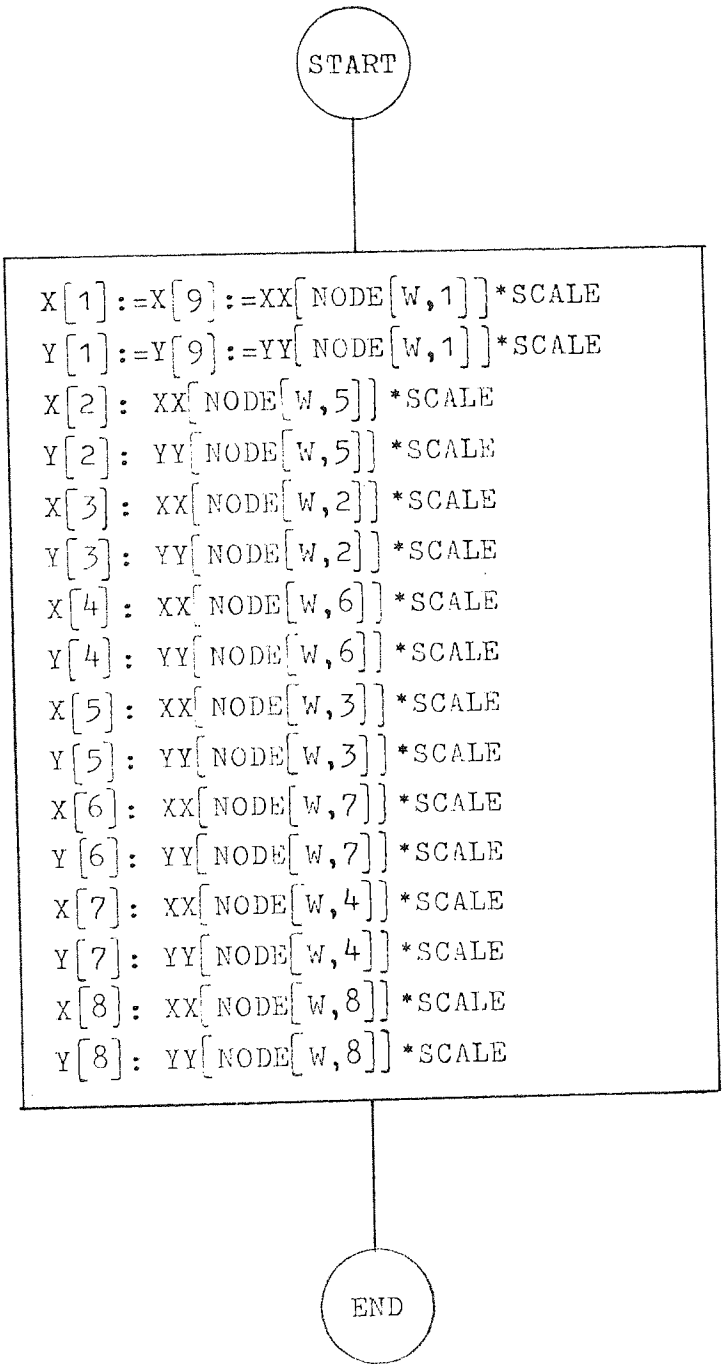
5,

```
'PROCEDURE' RESPLOT(QPLOT,TRPLOT,XX,YY,NODE);
'INTEGER''ARRAY'NODE;
'REAL''ARRAY'XX,YY;
'PROCEDURE' QPLOT,TRPLOT;
'BEGIN' 'COMMENT' GRAPHICAL DISPLAY OF INPUT DATA;
'REAL' XO,YO,CHANGE;
'INTEGER' 'ARRAY' BCD1,BCD2,BCD3[1:10];
'REAL' 'ARRAY' X,Y[1:9];
'IF'GH=0'THEN''GOTO'SKIP;
'BEGIN'
OPENPLOT;
HGPLOT(0.0,25.0,0.4);
CHANGE:=0.0;
'FOR' W:=1 'STEP' 1 'UNTIL' NNODE 'DO'
'BEGIN'
        'IF' XX[W]>CHANGE 'THEN' CHANGE:=XX[W];
        'IF' YY[W]>CHANGE 'THEN' CHANGE:=YY[W];
'END';
'IF'CHANGE>20'THEN'
    SCALE:=1/(ENTIER(CHANGE/20)+1)  'ELSE'
SCALE:=ENTIER(20/CHANGE);
'FOR' W:=1 'STEP' 1 'UNTIL' NELET 'DO'
  'IF'QORT=0'THEN'TRPLOT(XX,YY,NODE,X,Y)'ELSE'
                QPLOT(XX,YY,NODE,X,Y);
STRARR(BCD1,6,'('X%AXIS')');
STRARR(BCD2,6,'('Y%AXIS')');
STRARR(BCD3,23,'('STRUCTURAL%IDEALISATION')');
HGPAXIS(0.0,0.0,BCD1,-6,20.0,0.0,0.0,1.0);
HGPAXIS(0.0,0.0,BCD2,6,23.0,90.0,0.0,1.0);
HGPSYMBL(10,21,0.5,BCD3,0,23);
'END' OF GRAPH PLOTTER PROCEDURES;
CLOSEPLOT;
SKIP;'END' OF PROD RESPLOT;
```

Flowchart for Procedure Trplot:-



Flowchart for Procedure Qplot:-



```
'PROCEDURE' TRPLOT(XX,YY,NODE,X,Y);
'INTEGER''ARRAY'NODE;
'REAL''ARRAY'XX,YY,X,Y;
'BEGIN'
X[1]:=X[7]:=XX[NODE[W,1]]*SCALE;
                          Y[1]:=Y[7]:=YY[NODE[W,1]]*SCALE;
X[2]:=XX[NODE[W,4]]*SCALE;
                          Y[2]:=YY[NODE[W,4]]*SCALE;
X[3]:=XX[NODE[W,2]]*SCALE;
                          Y[3]:=YY[NODE[W,2]]*SCALE;
X[4]:=XX[NODE[W,5]]*SCALE;
                          Y[4]:=YY[NODE[W,5]]*SCALE;
X[5]:=XX[NODE[W,3]]*SCALE;
                          Y[5]:=YY[NODE[W,3]]*SCALE;
X[6]:=XX[NODE[W,6]]*SCALE;
                          Y[6]:=YY[NODE[W,6]]*SCALE;
HGPLINE(X,Y,7,1);
'END' OF TRIANGULAR ELEMENT;
```

```
'PROCEDURE' QPLOT(XX,YY,NODE,X,Y);
'INTEGER''ARRAY'NODE;
'REAL''ARRAY'XX,YY,X,Y;
'BEGIN'
X[1]:=X[9]:=XX[NODE[W,1]]*SCALE;
                          Y[1]:=Y[9]:=YY[NODE[W,1]]*SCALE;
X[2]:=XX[NODE[W,5]]*SCALE;
                          Y[2]:=YY[NODE[W,5]]*SCALE;
X[3]:=XX[NODE[W,2]]*SCALE;
                          Y[3]:=YY[NODE[W,2]]*SCALE;
X[4]:=XX[NODE[W,6]]*SCALE;
                          Y[4]:=YY[NODE[W,6]]*SCALE;
X[5]:=XX[NODE[W,3]]*SCALE;
                          Y[5]:=YY[NODE[W,3]]*SCALE;
X[6]:=XX[NODE[W,7]]*SCALE;
                          Y[6]:=YY[NODE[W,7]]*SCALE;
X[7]:=XX[NODE[W,4]]*SCALE;
          Y[7]:=YY[NODE[W,4]]*SCALE;
X[8]:=XX[NODE[W,8]]*SCALE;
          Y[8]:=YY[NODE[W,8]]*SCALE;
HGPLINE(X,Y,9,1);
'END' OF QUAD ELEMENT;
```

PROCEDURES QIDN AND TRIDN:

As explained in the introduction, to make the mesh generation scheme as general as possible a facility has been incorporated into the program which allows zone faces, originally open in the zone array, to be joined. This process causes the nodes on one of the zone faces to become redundant. In order to trace the redundant nodes, a copy of the nodes which need to be retained is made and by subsequent comparison of the x and y coordinates, the redundant nodes are found.

The object of these procedures is simply to record the node numbers which need to be retained when two zone faces are closed and these are stored in vector NUM.

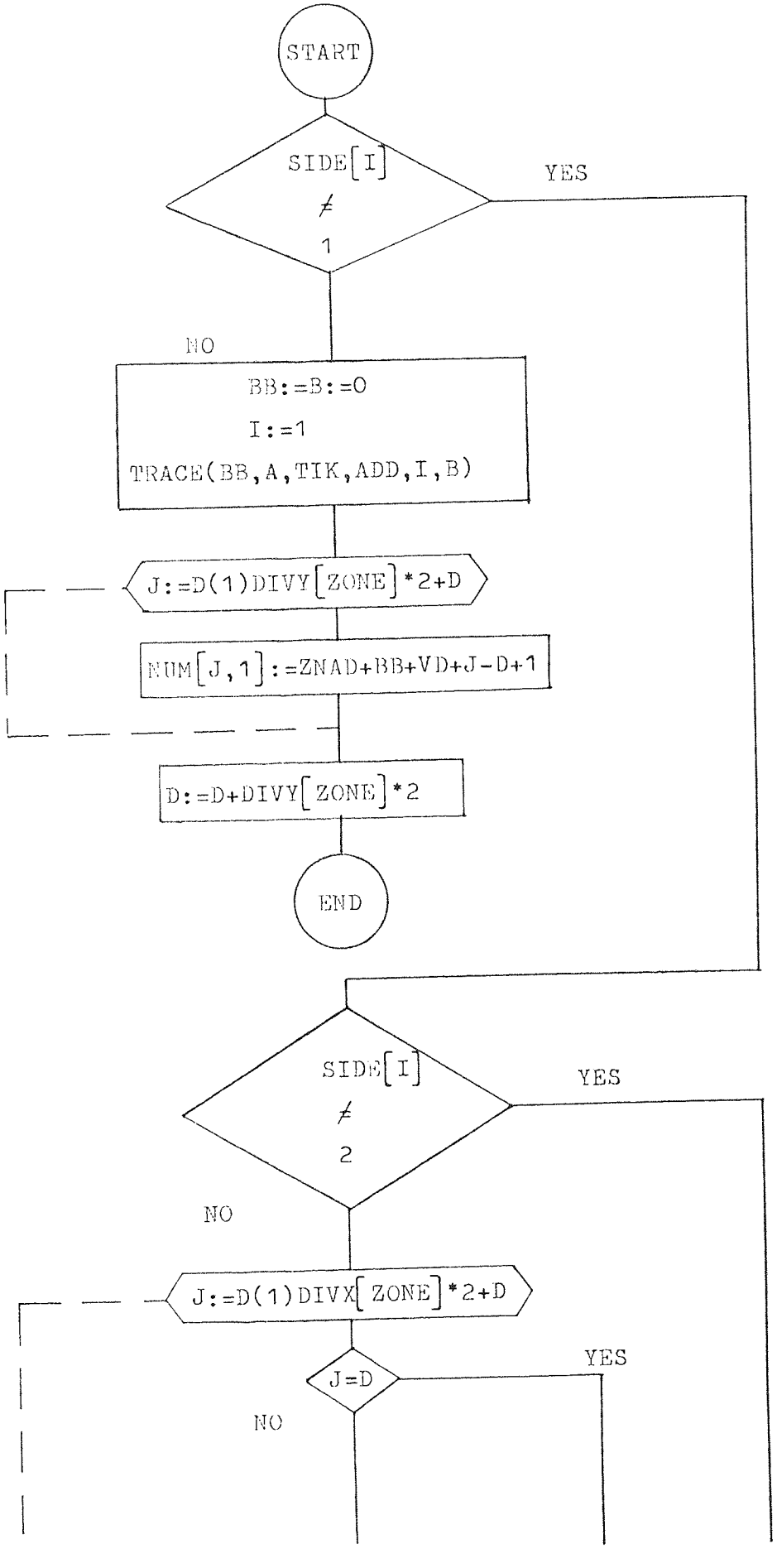
The following steps indicate how procedures QIDN and TRIDN function:

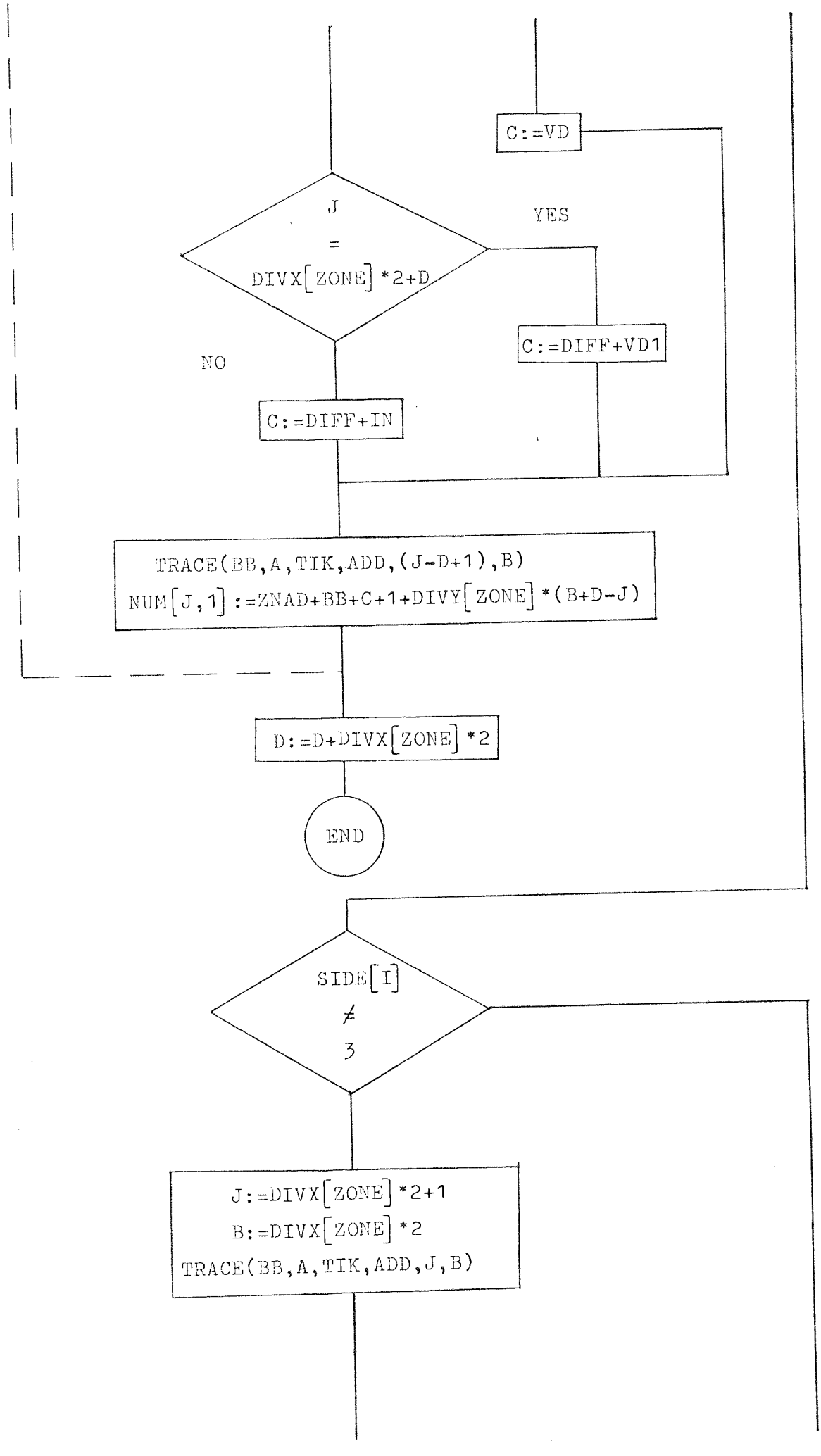
1. The zone face which is to be closed or joined, in the zone array, is given a side number, i.e. 1-4. This side number, together with the zone number is used to evaluate the nodes which are to be retained. Each side number is tested until the correct side has been found. Once the side has been identified the node numbering scheme is evoked and each node number is recorded in array NUM.

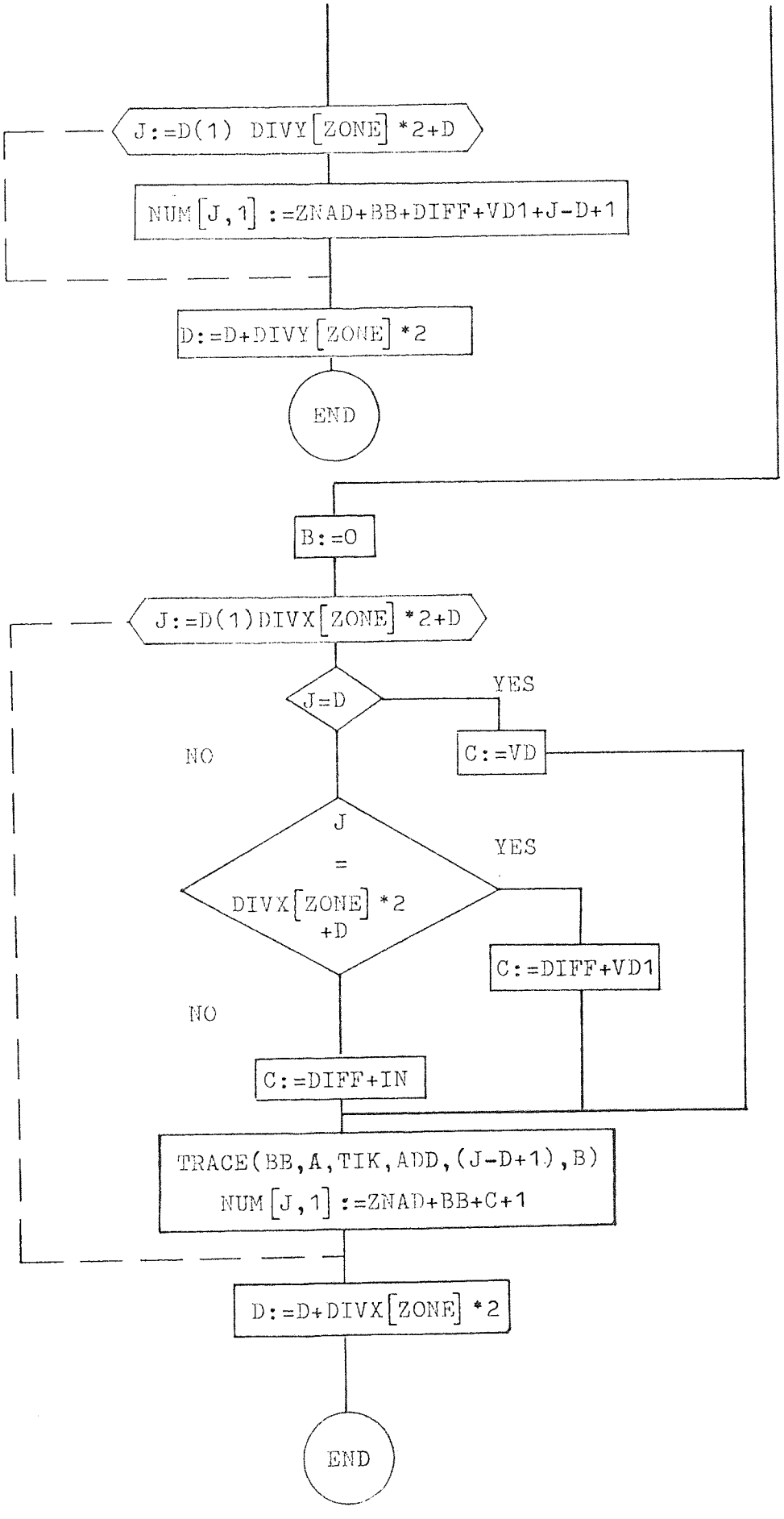
Note that the procedure TRACE is used in the case of the quadrilateral element.

If the zone has more than one face which is to be closed then the procedures are recalled by the main controlling procedure.

Flowchart for Procedure Qidn :-

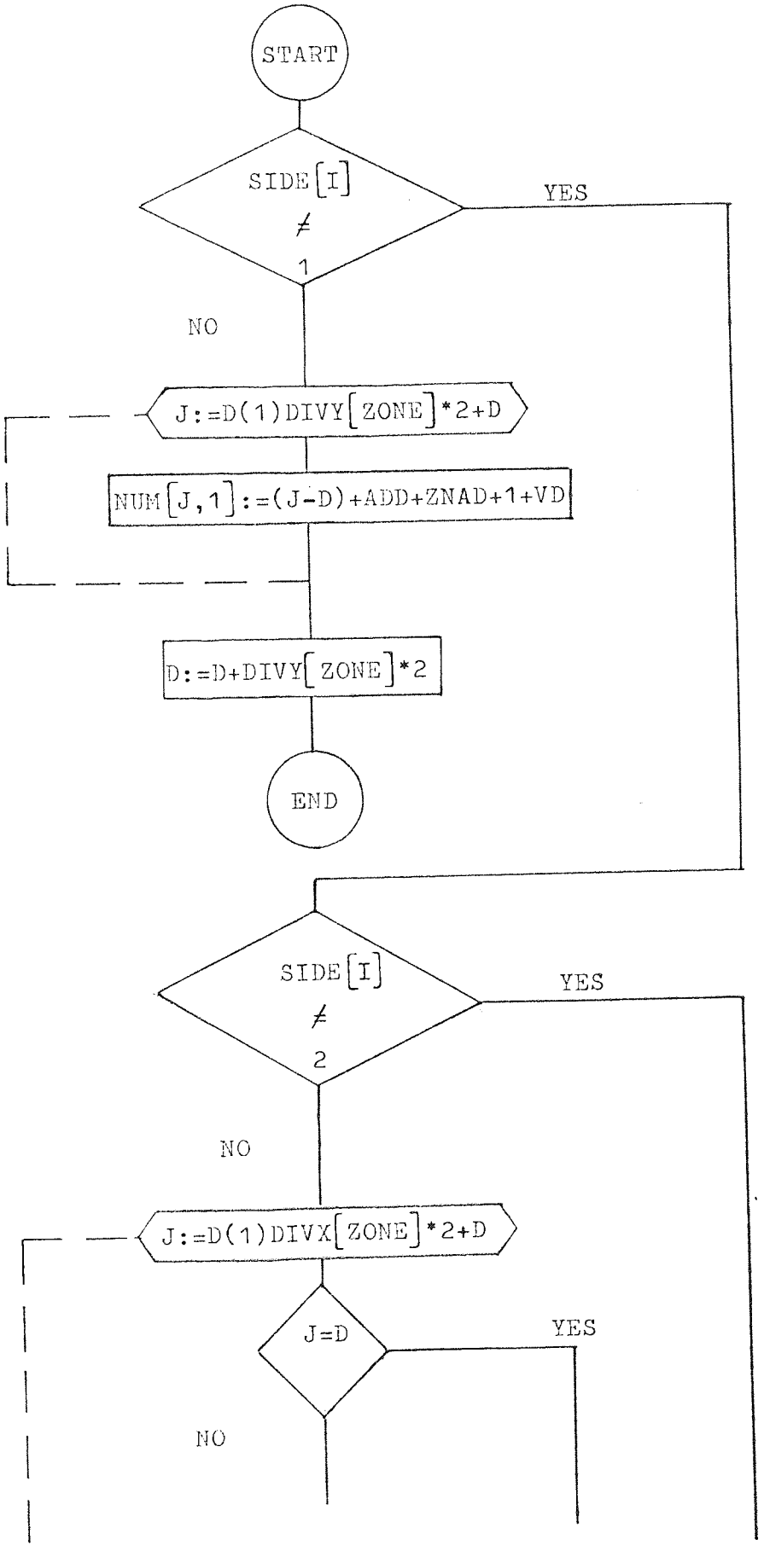


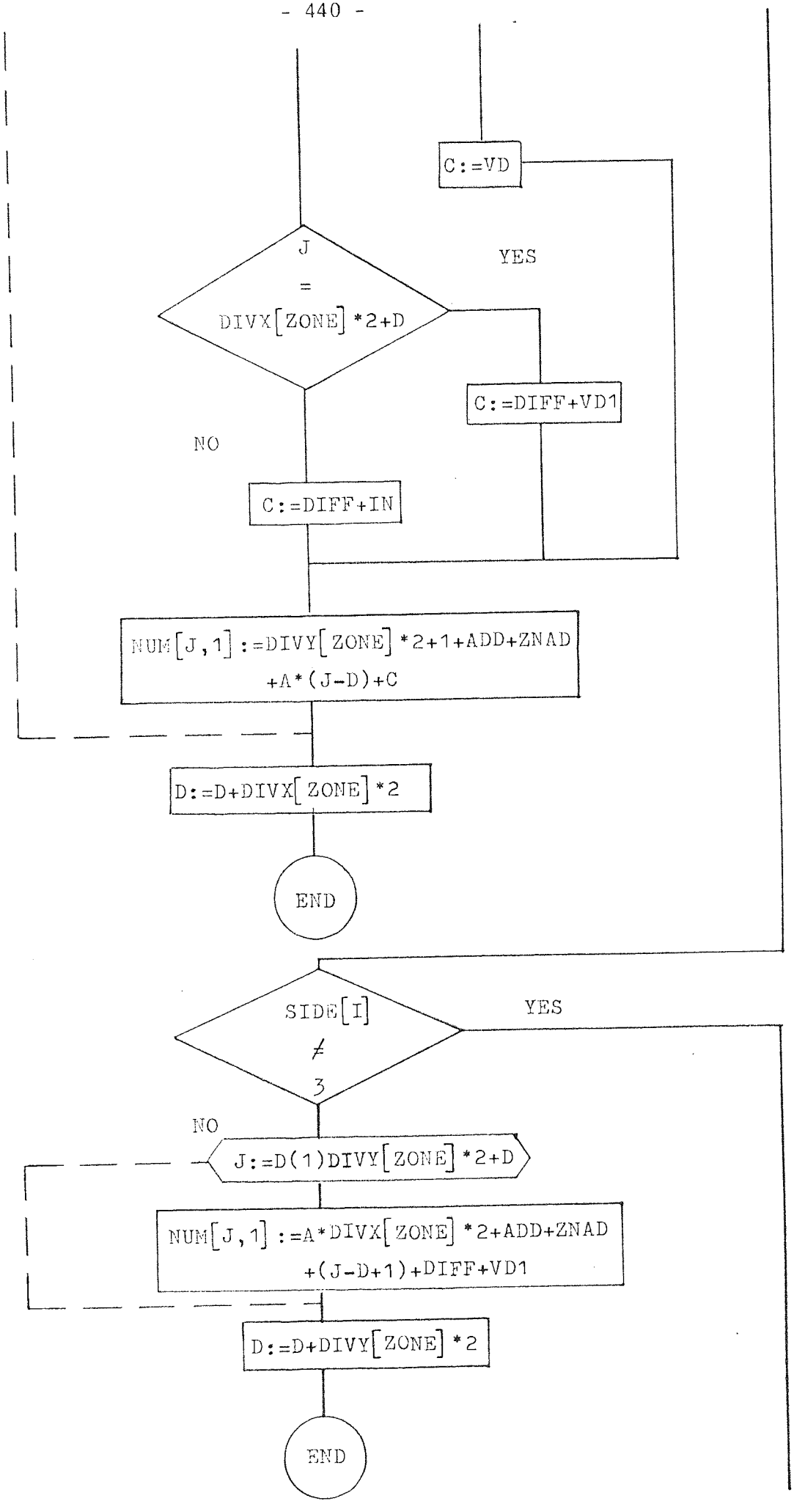

C:=VD
J
=
DIVX[ZONE]*2+D
YES
C:=DIFF+VD1
NO
C:=DIFF+IN
TRACE(BB,A,TIK,ADD,(J-D+1),B)
NUM[J,1]:=ZNAD+BB+C+1+DIVY[ZONE]*(B+D-J)
D:=D+DIVX[ZONE]*2
END
SIDE[I]
≠
3
J:=DIVX[ZONE]*2+1
B:=DIVX[ZONE]*2
TRACE(BB,A,TIK,ADD,J,B)

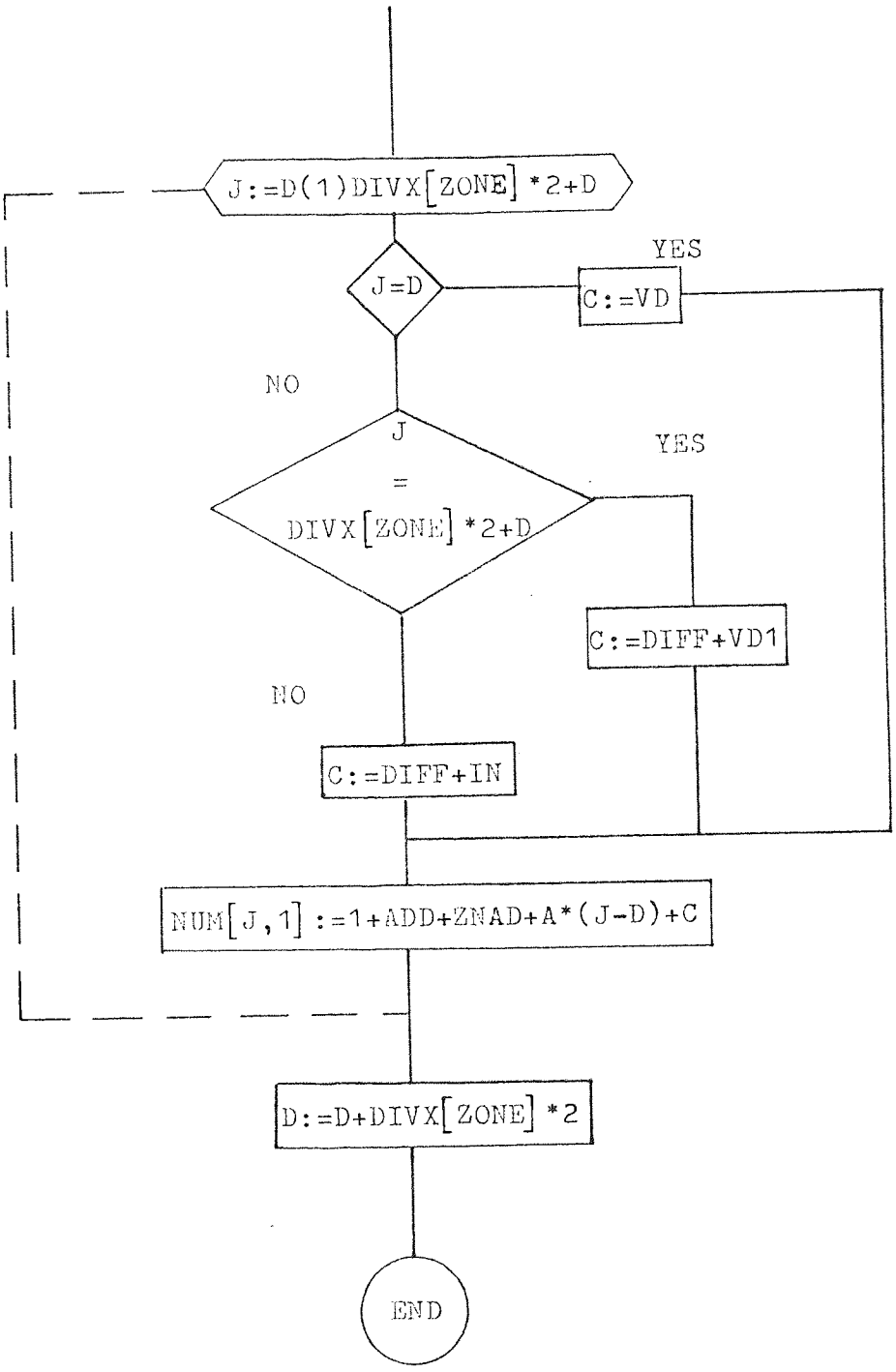

J:=D(1) DIVY[ZONE] *2+D
NUM[J,1] :=ZNAD+BB+DIFF+VD1+J-D+1
D:=D+DIVY[ZONE] *2
END
B:=0
J:=D(1)DIVX[ZONE] *2+D
J=D
YES
NO
C:=VD
J
=
DIVX[ZONE] *2
+D
YES
C:=DIFF+VD1
NO
C:=DIFF+IN
TRACE(BB,A,TIK,ADD,(J-D+1),B)
NUM[J,1] :=ZNAD+BB+C+1
D:=D+DIVX[ZONE] *2
END

```
'PROCEDURE'QIDN(TRACE,DIVX,DIVY,NUM);
'INTEGER''ARRAY'DIVX,DIVY,NUM;
'PROCEDURE' TRACE;
'BEGIN''INTEGER'B,BB,J,C;
'IF'SIDE[I]'NE'1'THEN'
'BEGIN'
   'IF'SIDE[I]'NE'2'THEN'
   'BEGIN'
      'IF'SIDE[I]'NE'3'THEN'
      'BEGIN'
B:=0;
      'FOR'J:=D'STEP'1'UNTIL'DIVX[ZONE]*2+D'DO'
      'BEGIN'
      'IF'J=D'THEN'C:=VD
            'ELSE''IF'J=DIVX[ZONE]*2+D
                  'THEN'C:=DIFF+VD1
                  'ELSE'C:=DIFF+IN;
    TRACE(BB,A,TIK,ADD,(J-D+1),B);
      NUM[J,1]:=ZNAD+BB+C+1;
      'END';
      D:=D+DIVX[ZONE]*2; 'GOTO'L80;
      'END'
            'ELSE'
   J:=DIVX[ZONE]*2+1;  B:=DIVX[ZONE]*2;
   TRACE(BB,A,TIK,ADD,J,B);
   'FOR'J:=D'STEP'1'UNTIL'DIVY[ZONE]*2+D'DO'
   NUM[J,1]:=ZNAD+BB+DIFF+VD1+J-D+1;
   D:=D+DIVY[ZONE]*2;  'GOTO'L80;
   'END'
         'ELSE'
B:=0;
'FOR'J:=D'STEP'1'UNTIL'DIVX[ZONE]*2+D'DO'
'BEGIN'
'IF'J=D'THEN'C:=VD
        'ELSE''IF'J=DIVX[ZONE]*2+D
              'THEN'C:=DIFF+VD1
              'ELSE'C:=DIFF+IN;
   TRACE(BB,A,TIK,ADD,(J-D+1),B);
   NUM[J,1]:=ZNAD+BB+C+1+DIVY[ZONE]*(B+D-J);
'END';
D:=D+DIVX[ZONE]*2; 'GOTO'L80;
'END'
'ELSE'
BB:=B:=0; I:=1;
   TRACE(BB,A,TIK,ADD,I,B);
   'FOR'J:=D'STEP'1'UNTIL'DIVY[ZONE]*2+D'DO'
   NUM[J,1]:=ZNAD+BB+VD+J-D+1;
D:=D+DIVY[ZONE]*2;
L80:'END' OF PROD QIDN;
```

Flowchart for Procedure Tridn:-




C:=VD
J
=
DIVX[ZONE]*2+D
YES
C:=DIFF+VD1
NO
C:=DIFF+IN
NUM[J,1]:=DIVY[ZONE]*2+1+ADD+ZNAD
+A*(J-D)+C
D:=D+DIVX[ZONE]*2
END
SIDE[I]
≠
3
YES
NO
J:=D(1)DIVY[ZONE]*2+D
NUM[J,1]:=A*DIVX[ZONE]*2+ADD+ZNAD
+(J-D+1)+DIFF+VD1
D:=D+DIVY[ZONE]*2
END


J:=D(1)DIVX[ZONE]*2+D
J=D
YES
C:=VD
NO
J
=
DIVX[ZONE]*2+D
YES
C:=DIFF+VD1
NO
C:=DIFF+IN
NUM[J,1]:=1+ADD+ZNAD+A*(J-D)+C
D:=D+DIVX[ZONE]*2
END

```
'PROCEDURE' TRIDN(DIVX,DIVY,NUM);
'INTEGER''ARRAY'DIVX,DIVY,NUM;
'BEGIN'
'IF'SIDE[I]'NE'1'THEN'
'BEGIN''IF'SIDE[I]'NE'2'THEN'
'BEGIN''IF'SIDE[I]'NE'3'THEN''BEGIN'
'FOR'J:=D'STEP'1'UNTIL'DIVX[ZONE]*2+D'DO'
'BEGIN''IF'J=D'THEN'C:=VD'ELSE''IF'J=DIVX[ZONE]*2+D'THEN'
C:=DIFF+VD1'ELSE'C:=DIFF+IN;
NUM[J,1]:=1+ADD+ZNAD+A*(J-D)+C;'END';
D:=D+DIVX[ZONE]*2;'GOTO'L8;'END'
'ELSE'
'FOR'J:=D'STEP'1'UNTIL'DIVY[ZONE]*2+D'DO'
        NUM[J,1]:=A*DIVX[ZONE]*2+ADD+ZNAD+(J-D+1)+DIFF+VD1;
D:=D+DIVY[ZONE]*2;'GOTO'L8;
'END'
'ELSE''FOR'J:=D'STEP'1'UNTIL'DIVX[ZONE]*2+D'DO''BEGIN'
'IF'J=D'THEN'C:=VD'ELSE''IF'J=DIVX[ZONE]*2+D'THEN'
C:=DIFF+VD1'ELSE'C:=DIFF+IN;
        NUM[J,1]:=DIVY[ZONE]*2+1+ADD+ZNAD+A*(J-D)+C; 'END';
D:=D+DIVX[ZONE]*2;'GOTO'L8;
'END'
'ELSE''FOR'J:=D'STEP'1'UNTIL'DIVY[ZONE]*2+D'DO'
        NUM[J,1]:=(J-D)+ADD+ZNAD+1+VD;
D:=D+DIVY[ZONE]*2;
L8:'END' OF PROD TRIDN;
```

PROCEDURE MGINPUT:

This procedure reads in all the mesh generation data, and the form in which this data is presented to the program can be found in the users guide (5.5.1). The program listing is self-explanatory, and the only point of interest is the sub-procedures COREGEN and STDGEN, which generate an ancillary set of data for the zones around circular regions. See procedures COREGEN and STDGEN.

```
'PROCEDURE' MGINPUT(XCOD,YCOD,MN,DIVX,DIVY,VARN,NODE);
'INTEGER''ARRAY'MN,DIVX,DIVY,VARN,NODE;
'REAL''ARRAY'XCOD,YCOD;
'BEGIN'
'FOR'J:=1'STEP'1'UNTIL'SNODE'DO'
 VARN[J]:=0;

'FOR'J:=VZONE'STEP'1'UNTIL'NZONES+VZONE'DO'
'BEGIN'DIVX[J]:=DIVY[J]:=0;'END';NZ:=0;
'IF'NTIP>0'THEN'COREGEN(XCOD,YCOD,VARN,NTIP,
VZONE,MN,DIVX,DIVY,NZ);
NGM:=READ;
'IF' NGM>0 'THEN' STDGEN(XCOD,YCOD,VARN,VZONE,MN,DIVX,DIVY);
'FOR'I:=1'STEP'1'UNTIL'TNSPOS'DO'
'BEGIN'
Q:=READ;
MU1:=READ;
MU2:=READ;
    'FOR'J:=1'STEP'1'UNTIL'Q'DO'
    'BEGIN'   W:=READ;   VARN[W]:=1;
              XCOD[W]:=MU1;
              YCOD[W]:=MU2;
    'END'J;
'END' OF I LOOP;
'FOR'ZN:=1'STEP'1'UNTIL'PZONE'DO'
'BEGIN'
Q:=READ; I:=READ; ND:=READ; NGM:=READ;
'FOR'J:=1'STEP'1'UNTIL'Q'DO'
'BEGIN'  P:=READ; MN[P]:=I;  DIVX[P]:=ND;   DIVY[P]:=NGM;
'END';
  ZN:=ZN+Q-1;
'END'OF ZN LOOP;SUM:=0;
IDENT:=READ;'IF'IDENT=0'THEN''GOTO'L4;
'FOR'I:=1'STEP'1'UNTIL'IDENT'DO''BEGIN'
IDN[I]:=READ;SIDE[I]:=READ;
'IF'SIDE[I]=2'OR'SIDE[I]=4'THEN'SUM:=SUM+DIVX[IDN[I]]*2+1
'ELSE'SUM:=SUM+DIVY[IDN[I]]*2+1;'END';
L4:'END' OF PROC MGINPUT;
```

PROCEDURE OUTPUT1:

There are several finite element programs in existence and each require a different data format. This input data consists of control variables which are local to each program. The node coordinates, element nodal connections and boundary condition data has the same format in each program and represents the bulk of the data.

To distinguish which program is going to be accessed a 'CODE' is used and this controls the data output. There is no need to add any further comments on this procedure, except to say that further detailed information on data presentation can be found in the users guide. Note that the last block of read statements are control variables required in the mesh generation scheme and set the array sizes, etc.

```
'PROCEDURE' OUTPUT1;
'BEGIN'
'INTEGER'NJOB,PRNT,PRINC,NSETF,NSETC,SURNO;
CODE:=READ; QORT:=READ;   NJOB:=READ;
PRINT(NJOB,3,0); NEWLINE(2); COPYTEXT('('END%OF%TITLE')');
WRITETEXT('('END%OF%TITLE')');
NEWLINE(1);  NELET:=READ;  NNODE:=READ;
PRINT(NELET,3,0);NSKEW:=0; PRINT(NNODE,3,0);
'IF'CODE=1'THEN'
'BEGIN'
NSETFS:=READ; PRNT:=READ;PRINC:=READ; NSKEW:=READ; NMAT:=READ;
NSETF:=READ; NSETC:=READ; PRINT(NSETFS,3,0);
PRINT(PRNT,3,0);PRINT(PRINC,3,0); PRINT(NSKEW,3,0); PRINT(NMAT,3,0);
PRINT(QORT,3,0); PRINT(NSETF,3,0);
PRINT(NSETC,3,0);  'END';
'IF'CODE=3'THEN''BEGIN'
THICK:=READ;NSETFS:=READ;
PRINT(THICK,0,10);PRINT(NSETFS,3,0);
NSKEW:=READ; SURNO:=READ;
            PRINT(NSKEW,3,0); PRINT(SURNO,3,0);
            PRINT(QORT,3,0);
           NSETC:=READ; PRINT(NSETC,3,0);
      NSETF:=READ;  PRINT(NSETF,3,0);
 'END';
'IF'CODE=2'THEN''BEGIN'
  THICK:=READ;  NSETFS:=READ;  NSETC:=READ;
 PRINT(THICK,0,10); PRINT(NSETFS,3,0); PRINT(QORT,3,0);
  PRINT(NSETC,3,0);
'END';
'IF' CODE=4'THEN''BEGIN'
  'FOR'I:=1'STEP'1'UNTIL'6'DO'
   'BEGIN'  PRNT:=READ;
      'IF' I=4'THEN' NMAT:=PRNT;
    PRINT(PRNT,3,0);
   'END';
'END';
            TNSPDS:=READ;
           PZONE:=READ;
            VZONE:=READ;
            HZONE:=READ;
            GH:=READ;
            NTIP:=READ;
            NZONES:=VZONE*HZONE;
            SNODE:=(VZONE*2+1)*(HZONE+1)+(VZONE+1)*HZONE;
'END' OF PROD OUTPUT1;
```

PROCEDURE OUTPUT2:

This procedure prints out the bulk of the data, that is the nodal coordinates, element nodal connections and boundary conditions, together with information necessary for the control of the receiving programs.

A 'CODE' is used to distinguish the various programs, as in procedure OUTPUT1, and this controls the output format. The following steps indicate how procedure OUTPUT2 functions:-

1. The x and y coordinates are printed out.
2. The element nodal connections are printed, and depending on the code, the material number is either included or left out.
3. The boundary conditions are printed.
4. The control variables are printed out for each program depending on the 'CODE' number. See the users guide for further information. The sub-procedure MATCON simply reads in the material constants and re-prints them.
5. The four asterisks are printed and are necessary as a data block terminator.

```
'PROCEDURE' OUTPUT2(MATCON,XX,YY,NODE);
'INTEGER''ARRAY'NODE;
'REAL''ARRAY'XX,YY;
'PROCEDURE' MATCON;
'BEGIN'
'FOR'I:=0'STEP'2'UNTIL'NNODE'DO''BEGIN'NEWLINE(1);
'FOR'J:=1'STEP'1'UNTIL'2'DO''BEGIN'
'IF'I+J>NNODE'THEN''GOTO'TOWN;
PRINT(XX[I+J],0,8);PRINT(YY[I+J],0,8);
'END';  'END'; TOWN:NEWLINE(1);
'FOR'I:=1'STEP'2'UNTIL'NELET'DO''BEGIN'
'IF'CODE=1 'OR' CODE=4'THEN''BEGIN'
'FOR'J:=1'STEP'1'UNTIL'(7+QORT*2)'DO'
PRINT(NODE[I,J],3,0);'IF'NELET<I+1'THEN''GOTO'LEAV;
   NEWLINE(1);
'FOR'J:=1'STEP'1'UNTIL'(7+QORT*2)'DO'PRINT(NODE[I+1,J],3,0);
'END' 'ELSE'
  'BEGIN'
'FOR'J:=1'STEP'1'UNTIL'(6+QORT*2)'DO'
PRINT(NODE[I,J],3,0);'IF'NELET<I+1'THEN''GOTO'LEAV;
  NEWLINE(1);
'FOR'J:=1'STEP'1'UNTIL'(6+QORT*2)'DO'PRINT(NODE[I+1,J],3,0);
  'END';
LEAV;NEWLINE(1); 'END';
NSPEC:=READ; PRINT(NSPEC,3,0);NEWLINE(1);
'BEGIN'
'INTEGER''ARRAY'KODE[1:NNODE,1:NSETFS],NOSK[1:NSKEW+1];
'REAL''ARRAY'ULX,VLY[1:NNODE,1:NSETFS],ANGSK[1:NSKEW+1];
'FOR'I:=1'STEP'1'UNTIL'NSPEC'DO''BEGIN'
   J:=READ; KODE[J,1]:=READ; ULX[J,1]:=READ; VLY[J,1]:=READ;
PRINT(J,3,0);PRINT(KODE[J,1],3,0);PRINT(ULX[J,1],0,10);
PRINT(VLY[J,1],0,10); NEWLINE(1); 'END';
'IF'CODE=1'THEN''BEGIN'
     'FOR'I:=1'STEP'1'UNTIL'NMAT'DO''BEGIN'
 CASE:=READ; PRINT(CASE,3,0);
ANG:=READ; THICK:=READ;PRINT(ANG,0,10);PRINT(THICK,0,10);
   MATCON(EE1,MU1,GE,EE2,MU2);
 'END';
 'FOR'I:=1'STEP'1'UNTIL'NSKEW'DO''BEGIN'
      NOSK[I]:=READ; ANGSK[I]:=READ;PRINT(NOSK[I],3,0);
    PRINT(ANGSK[I],0,10);  NEWLINE(1);
    'END';
'END';
'IF'CODE=2'THEN''BEGIN'
    CASE:=READ; PRINT(CASE,3,0);
    MATCON(EE1,MU1,GE,EE2,MU2);
HND:=READ;PRINT(NI[1],3,0);PRINT(RO[1],0,10);  PRINT(HND,3,0);
   'END';
    'IF'CODE=3'THEN''BEGIN'
   'FOR'I:=1'STEP'1'UNTIL'NTIP'DO'
   'BEGIN'
        PRINT(NI[I],3,0);PRINT(RO[I],0,10);PRINT(AL[I],0,10);
        PRINT(NS[I],4,0); NEWLINE(1);
        'END';
```

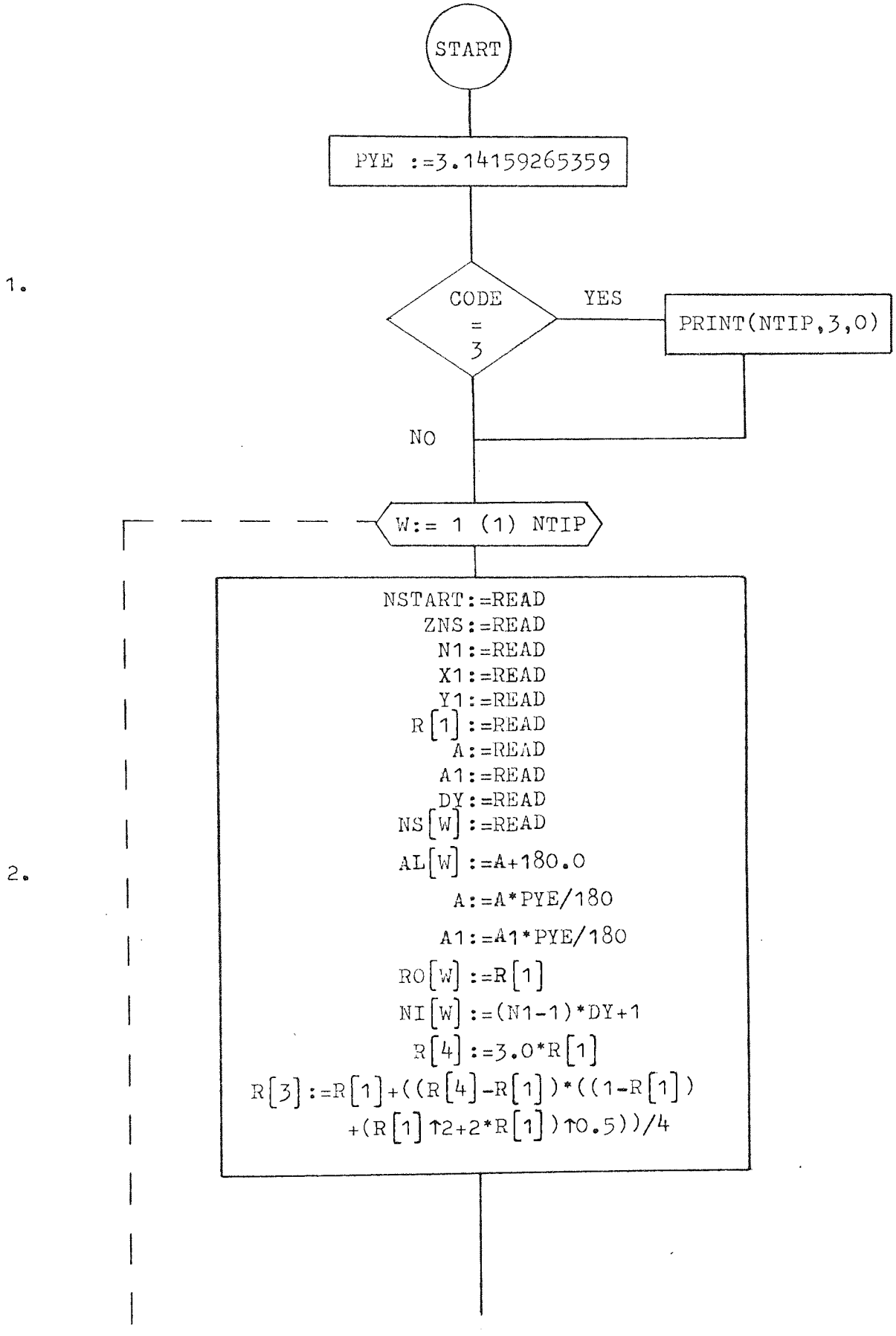
```
      CASE:=READ;PRINT(CASE,3,0); NEWLINE(1);
      MATCON(EE1,MU1,GE,EE2,MU2);
'FOR'I:=1'STEP'1'UNTIL'NSKEW'DO''BEGIN'
       NOSK[I]:=READ; ANGSK[I]:=READ;
PRINT(NOSK[I],3,0);PRINT(ANGSK[I],0,10);NEWLINE(1);
'END'; 'END' CODE3;
'IF' CODE=4 'THEN''BEGIN'
   'FOR'I:=1'STEP'1'UNTIL'NMAT'DO'
   'BEGIN' CASE:=READ; PRINT(CASE,3,0);
   MATCON(EE1,MU1,GE,EE2,MU2);
   'END';
'END' CODE4;
'END';NEWLINE(1);   WRITETEXT('('****')');
'END' OF PROD OUTPUT2;
```

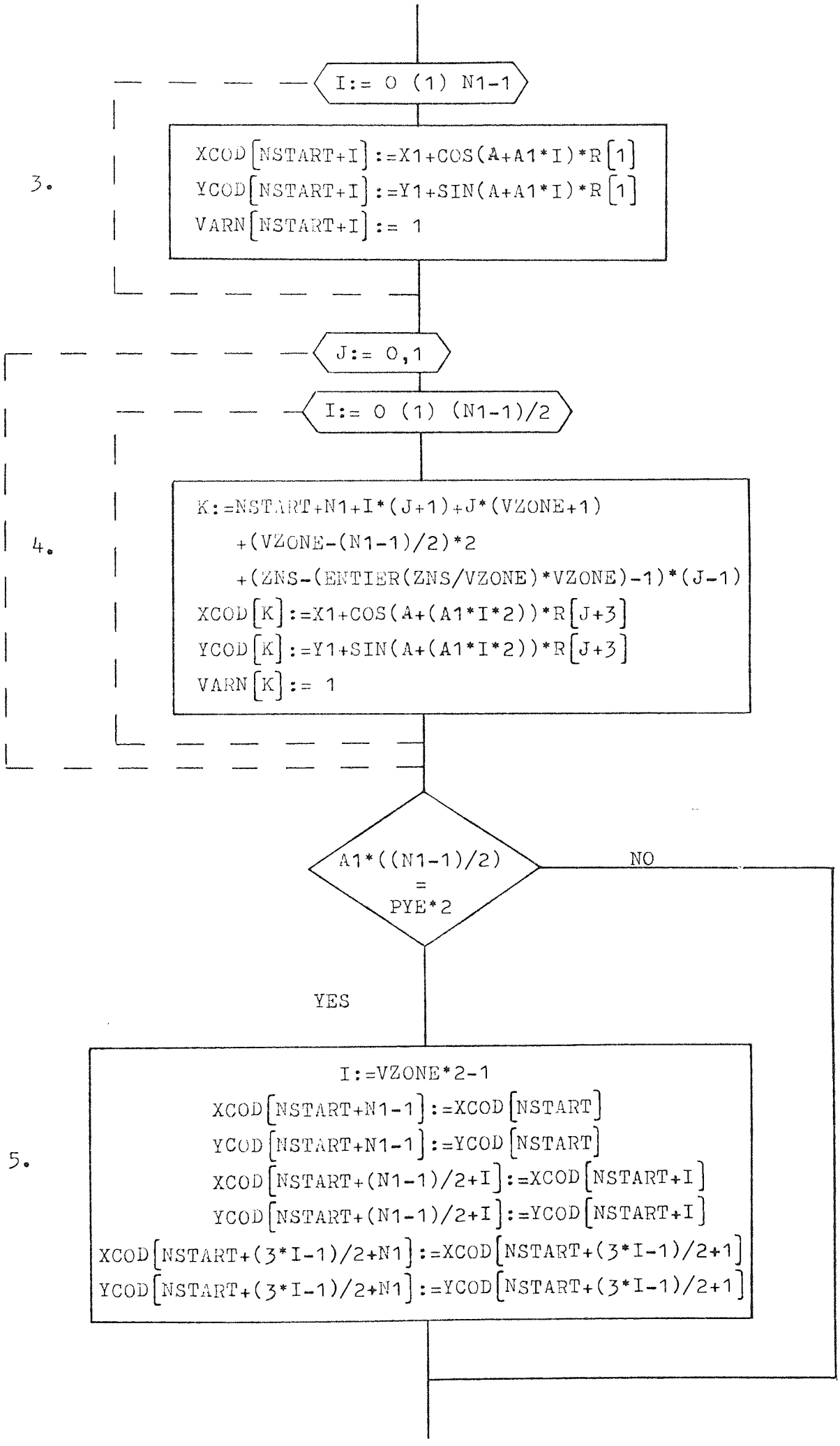
PROCEDURE COREGEN:

This procedure has been designed specifically to generate the 'transition' elements around the crack tip core element. It performs the same operations as procedure STDGEN. Basically a column of zones is generated from a string of input data, which is effectively 'wrapped' around the core element. Refer to examples (3) and (4) in section 5.5, for details on the data input and zone configuration. Section 2.5.2 examines the theoretical aspects of 'transition' elements, which are further discussed in Chapter six. The following points refer to the flowchart segments:-

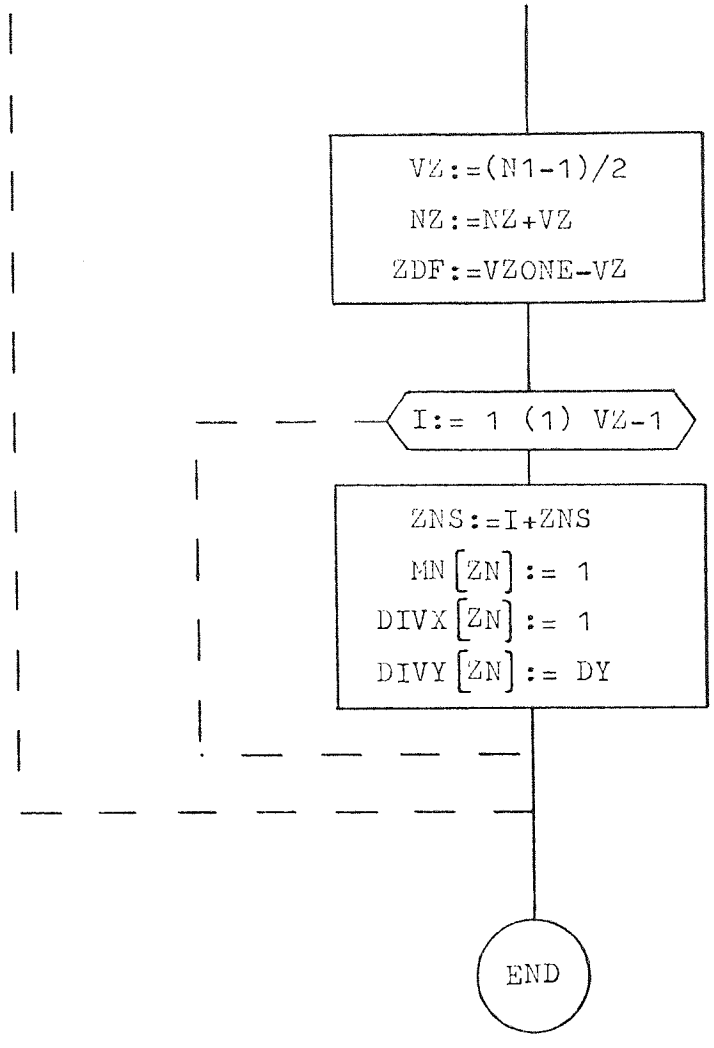
1. The number of crack tips is printed if the multi-tip program has been selected i.e. CODE=3.
2. The string of parameters are read into the computer and modified for later use in the F.E.program.
3. The coordinates of the super-nodes lying on the circular boundary are specified.
4. The coordinates of the super-nodes on the normal zone faces and also the remaining corner nodes are determined.
5. If the core element takes the form of a full circle, then the zones will in effect close. To avoid any miss-match, due to the inaccuracy of π, the closing node coordinates are equated.
6. Finally the zone's material type and sub-division is specified.

Flowchart for Procedure Coregen :-




I:= 0 (1) N1-1
3.
XCOD[NSTART+I]:=X1+COS(A+A1*I)*R[1]
YCOD[NSTART+I]:=Y1+SIN(A+A1*I)*R[1]
VARN[NSTART+I]:= 1
J:= 0,1
I:= 0 (1) (N1-1)/2
4.
K:=NSTART+N1+I*(J+1)+J*(VZONE+1)
+(VZONE-(N1-1)/2)*2
+(ZNS-(ENTIER(ZNS/VZONE)*VZONE)-1)*(J-1)
XCOD[K]:=X1+COS(A+(A1*I*2))*R[J+3]
YCOD[K]:=Y1+SIN(A+(A1*I*2))*R[J+3]
VARN[K]:= 1
A1*((N1-1)/2)
=
PYE*2
NO
YES
5.
I:=VZONE*2-1
XCOD[NSTART+N1-1]:=XCOD[NSTART]
YCOD[NSTART+N1-1]:=YCOD[NSTART]
XCOD[NSTART+(N1-1)/2+I]:=XCOD[NSTART+I]
YCOD[NSTART+(N1-1)/2+I]:=YCOD[NSTART+I]
XCOD[NSTART+(3*I-1)/2+N1]:=XCOD[NSTART+(3*I-1)/2+1]
YCOD[NSTART+(3*I-1)/2+N1]:=YCOD[NSTART+(3*I-1)/2+1]

6.


VZ:=(N1-1)/2
NZ:=NZ+VZ
ZDF:=VZONE-VZ
I:= 1 (1) VZ-1
ZNS:=I+ZNS
MN[ZN]:= 1
DIVX[ZN]:= 1
DIVY[ZN]:= DY
END

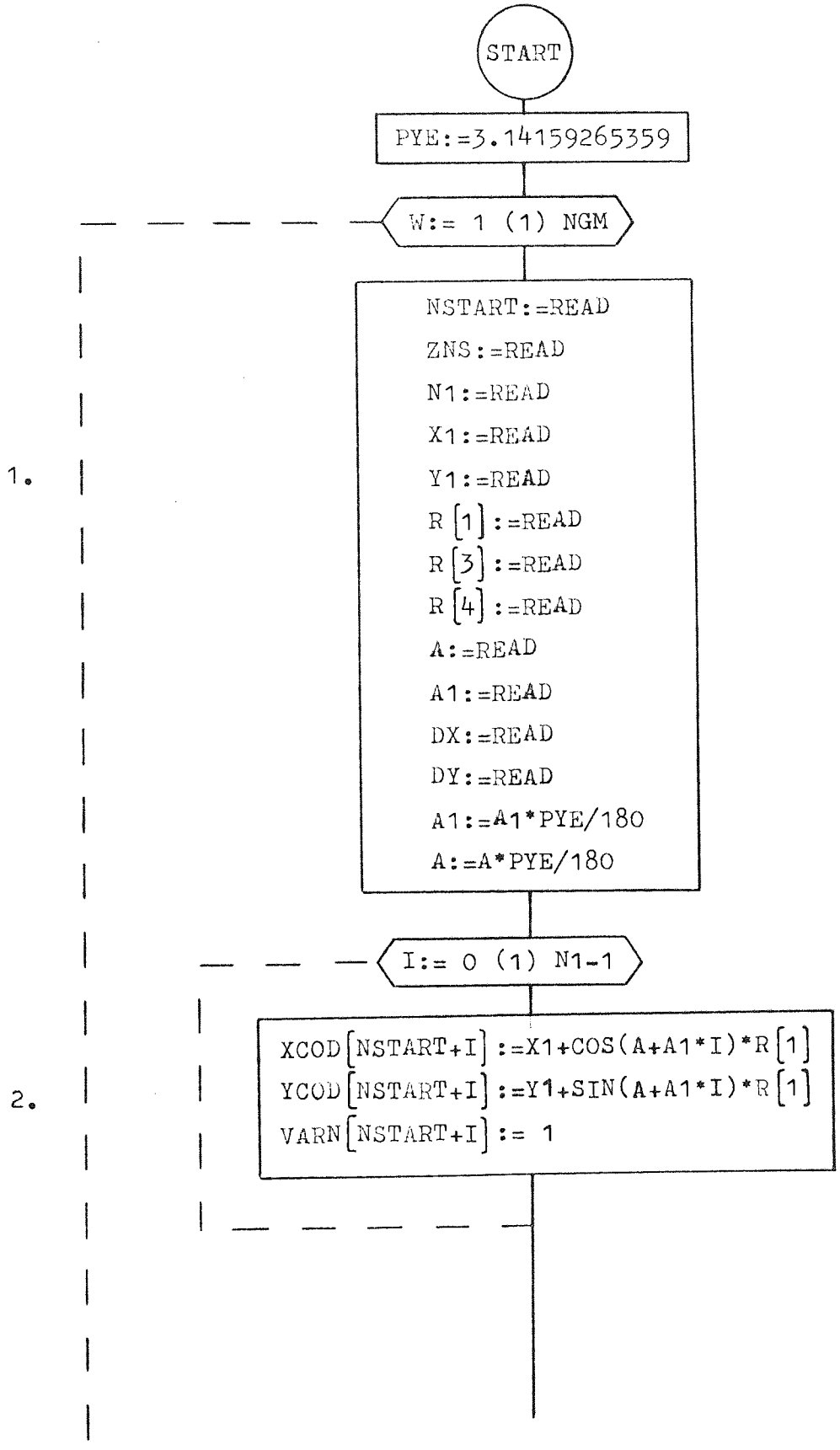
```
'PROCEDURE' COREGEN(XCOD,YCOD,VARN,NTIP,
VZONE,MN,DIVX,DIVY,NZ);
     'VALUE'VZONE;
'INTEGER'NTIP,VZONE,NZ;
'INTEGER''ARRAY'VARN,MN,DIVX,DIVY;
      'REAL''ARRAY'XCOD,YCOD;
'BEGIN'
      'INTEGER'N1,NSTART,I,J,W,K,ZN,ZNS,ZDF,VZ;
      'REAL'A,A1,X1,Y1,PYE,DY;  'REAL''ARRAY'R[1:4];
PYE:=3.14159265359;
'IF' CODE = 3 'THEN' PRINT(NTIP,3,0);
                    'FOR'W:=1'STEP'1'UNTIL'NTIP'DO''BEGIN'
    NSTART:=READ;ZNS:=READ; N1:=READ; X1:=READ; Y1:=READ;
    R[1]:=READ;   R[4]:=3.0*R[1];
R[3]:=R[1]+((R[4]-R[1])*((1-R[1])+(R[1]↑2+2*R[1])↑0.5))/4;
      A:=READ; A[W]:=A+180.0;
        A1:=READ; A:=A*PYE/180; A1:=A1*PYE/180;
      DY:=READ; NS[W]:=READ;
     RO[W]:=R[1];  NI[W]:=(N1-1)*DY+1;
 'FOR'I:=0'STEP'1'UNTIL'N1-1'DO''BEGIN'
     XCOD[NSTART+I]:=X1+COS(A+A1*I)*R[1];
     YCOD[NSTART+I]:=Y1+SIN(A+A1*I)*R[1];
    VARN[NSTART+I]:=1;
     'END';
'FOR'J:=0,1'DO''BEGIN'
    'FOR'I:=0'STEP'1'UNTIL'(N1-1)/2'DO''BEGIN'
      K:=NSTART+N1+I*(J+1)+J*(VZONE+1)+(VZONE-(N1-1)/2)*2
          +(ZNS-(ENTIER(ZNS/VZONE)*VZONE)-1)*(J-1);
       XCOD[K]:=X1+COS(A+(A1*I*2))*R[J+3];
        YCOD[K]:=Y1+SIN(A+(A1*I*2))*R[J+3];
        VARN[K]:=1;
             'END';
               'END' OF J LOOP;
'IF' A1*((N1-1)/2) = PYE*2 'THEN'
'BEGIN'
   I:=VZONE*2+1;
   XCOD[NSTART+N1-1]:=XCOD[NSTART];
   YCOD[NSTART+N1-1]:=YCOD[NSTART];
   XCOD[NSTART+(N1-1)/2+I]:=XCOD[NSTART+I];
   YCOD[NSTART+(N1-1)/2+I]:=YCOD[NSTART+I];
   XCOD[NSTART+(3*I-1)/2+N1]:=XCOD[NSTART+(3*I-1)/2+1];
   YCOD[NSTART+(3*I-1)/2+N1]:=YCOD[NSTART+(3*I-1)/2+1];
'END';
     VZ:=(N1-1)/2;NZ:=NZ+VZ;  ZDF:=VZONE-VZ;
  'FOR'I:=0'STEP'1'UNTIL'VZ-1'DO''BEGIN'
       ZN:=I+ZNS;       MN[ZN]:=1;  DIVX[ZN]:=1;  DIVY[ZN]:=DY;
   'END' OF I LOOP;
                                   'END';
                                        'END' OF PROCEDURE COREGEN
```

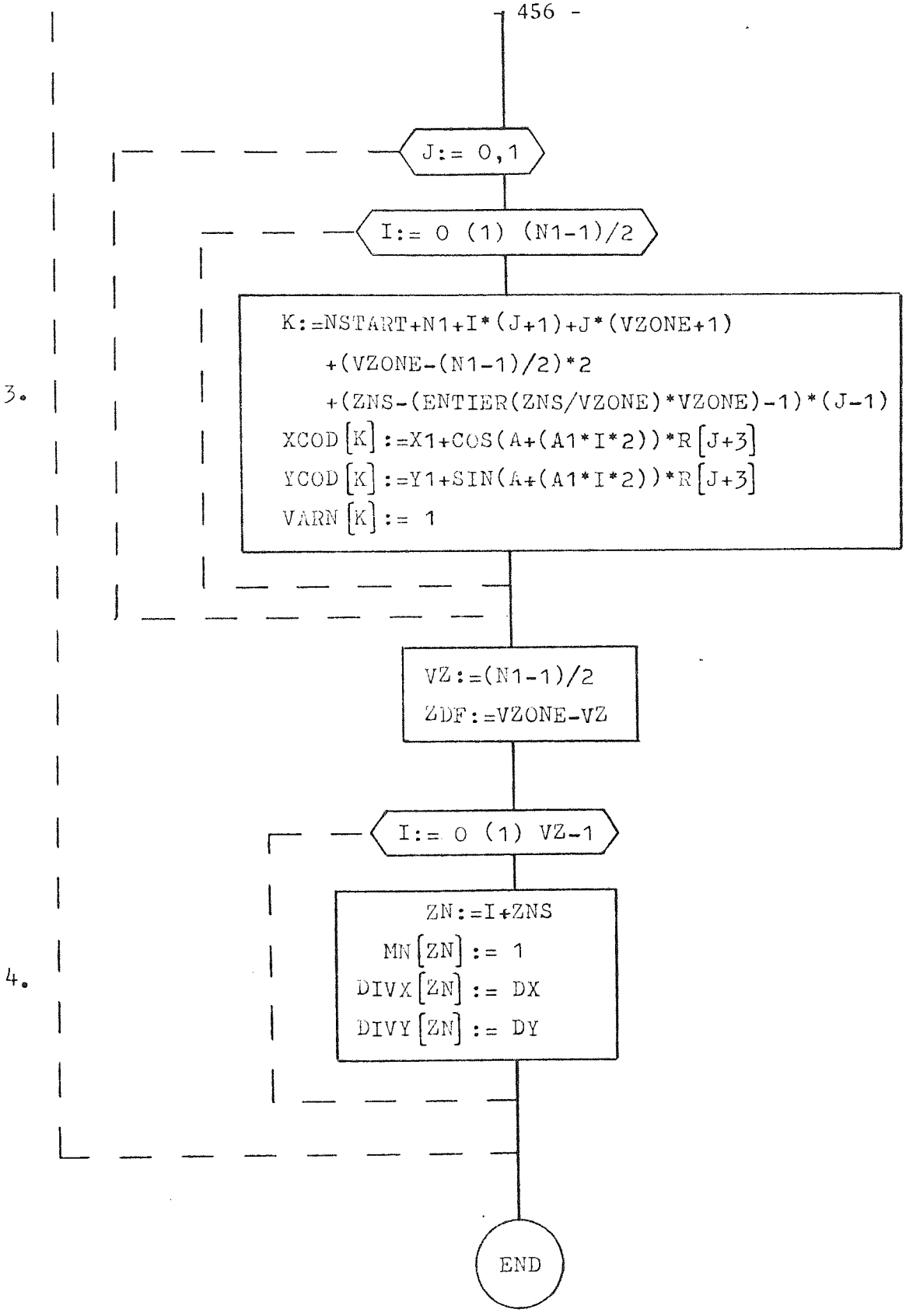
PROCEDURE STDGEN:

In cases where the mesh is uniformly distributed around a hole or circular region, a standard data generation procedure can be employed. The region under consideration can be completely defined using a string of parameters which are used to generate the normal input data. The use of the procedure represents a data input reduction and can be viewed as a labour saving device. The method of use is illustrated by example (2) of Section 5.5. The procedure generates the data for a column of zones, computing the coordinates of all the corner nodes and the mid-side nodes along the curved boundary and normal faces. This allows the internal mesh to be graded in a radial sense. Any of the super-node coordinates can be overridden by re-declaring them in the usual fashion, this allows the generated zones to be merged into the surrounding mesh. The following steps indicate how this procedure functions and correspond with the flowchart:-

1. The string of defining parameters are read into the computer, as given in the users guide (5.5.1), for each generated region.
2. The coordinates of the super-nodes along the curved boundary are designated.
3. The coordinates of the mid-side nodes on the normal zone faces and the corner nodes are determined summing on counters J and I.
4. Within the loop I, the material number and zone sub-division are specified.

Flowchart for Procedure STDGEN :-




J:= 0,1
I:= 0 (1) (N1-1)/2
K:=NSTART+N1+I*(J+1)+J*(VZONE+1)
+(VZONE-(N1-1)/2)*2
+(ZNS-(ENTIER(ZNS/VZONE)*VZONE)-1)*(J-1)
XCOD[K]:=X1+COS(A+(A1*I*2))*R[J+3]
YCOD[K]:=Y1+SIN(A+(A1*I*2))*R[J+3]
VARN[K]:= 1
3.
VZ:=(N1-1)/2
ZDF:=VZONE-VZ
I:= 0 (1) VZ-1
ZN:=I+ZNS
MN[ZN]:= 1
DIVX[ZN]:= DX
DIVY[ZN]:= DY
4.
END

```
'PROCEDURE' STDGEN(XCOD,YCOD,VARN,
VZONE,MN,DIVX,DIVY);
     'VALUE'VZONE;
'INTEGER'VZONE;
'INTEGER''ARRAY'VARN,MN,DIVX,DIVY;
       'REAL''ARRAY'XCOD,YCOD;
'BEGIN'
      'INTEGER'N1,NSTART,I,J,M,K,ZN,ZNS,ZDF,VZ;
      'REAL'A,A1,X1,Y1,PYE,DX,DY;  'REAL''ARRAY'R[1:4];
PYE:=3.14159265350;
                    'FOR'M:=1'STEP'1'UNTIL'NGH'DO''BEGIN'
    NSTART:=READ;ZNS:=READ; N1:=READ; X1:=READ; Y1:=READ;
    R[1]:=READ;          R[3]:=READ;
    R[4]:=READ;  A:=READ;
         A1:=READ; A:=A*PYE/180; A1:=A1*PYE/180;
      DX:=READ; DY:=READ;
 'FOR'I:=0'STEP'1'UNTIL'N1-1'DO''BEGIN'
     XCOD[NSTART+I]:=X1+COS(A+A1*I)*R[1];
     YCOD[NSTART+I]:=Y1+SIN(A+A1*I)*R[1];
    VARN[NSTART+I]:=1;
     'END';
'FOR'J:=0,1'DO''BEGIN'
    'FOR'I:=0'STEP'1'UNTIL'(N1-1)/2'DO''BEGIN'
      K:=NSTART+N1+I*(J+1)+J*(VZONE+1)+(VZONE-(N1-1)/2)*2
         +(ZNS-(ENTIER(ZNS/VZONE)*VZONE)-1)*(J-1);
       XCOD[K]:=X1+COS(A+(A1*I*2))*R[J+3];
        YCOD[K]:=Y1+SIN(A+(A1*I*2))*R[J+3];
        VARN[K]:=1;
             'END';
               'END' OF J LOOP;
   VZ:=(N1-1)/2;  ZDF:=VZONE-VZ;
  'FOR'I:=0'STEP'1'UNTIL'VZ=1'DO''BEGIN'
       ZN:=I+ZNS;       MN[ZN]:=1;  DIVX[ZN]:=DX;  DIVY[ZN]:=DY;
   'END' OF I LOOP;
                                    'END';
'END' OF PROCEDURE STDGEN;
```

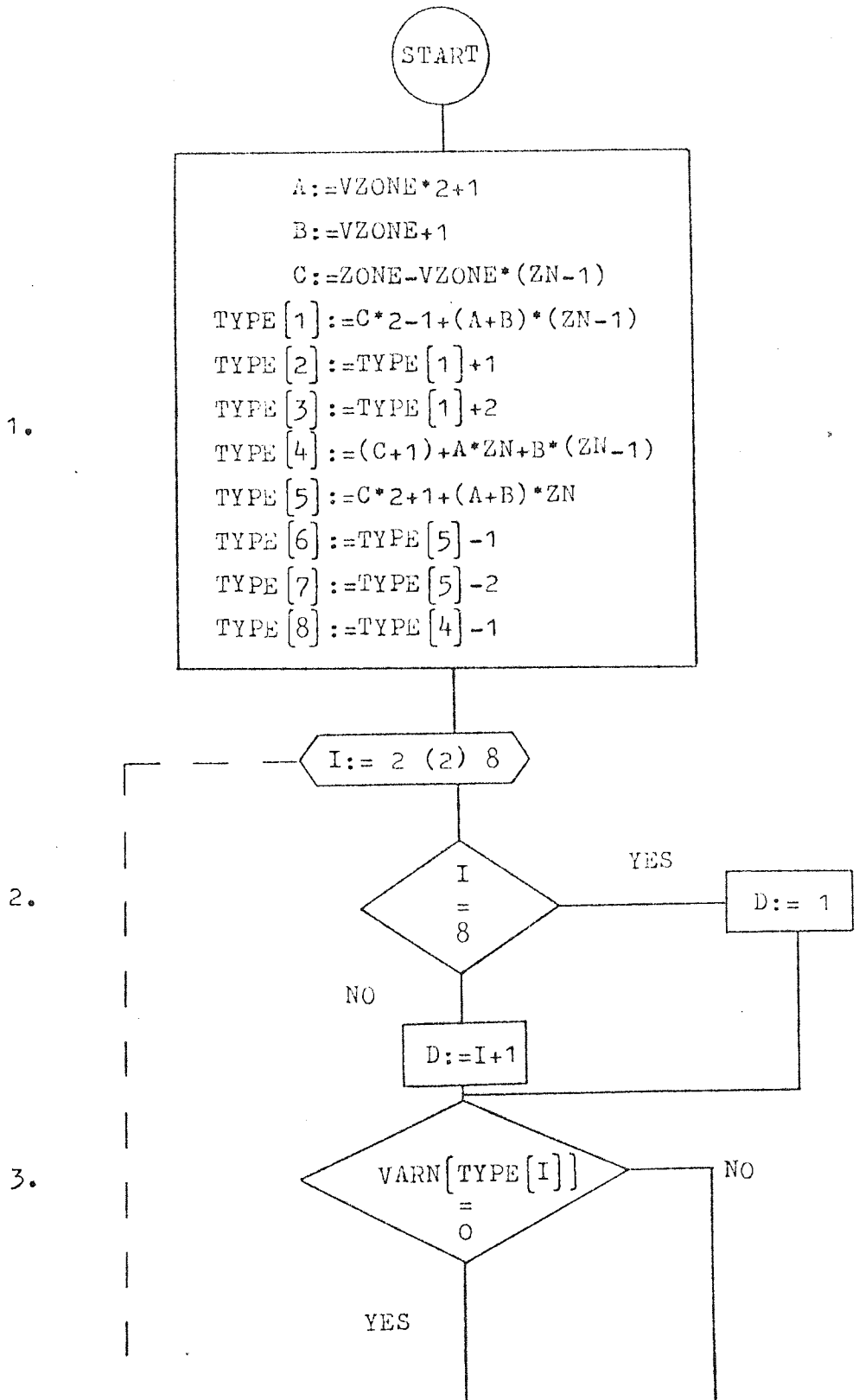
PROCEDURE ZONEXY:

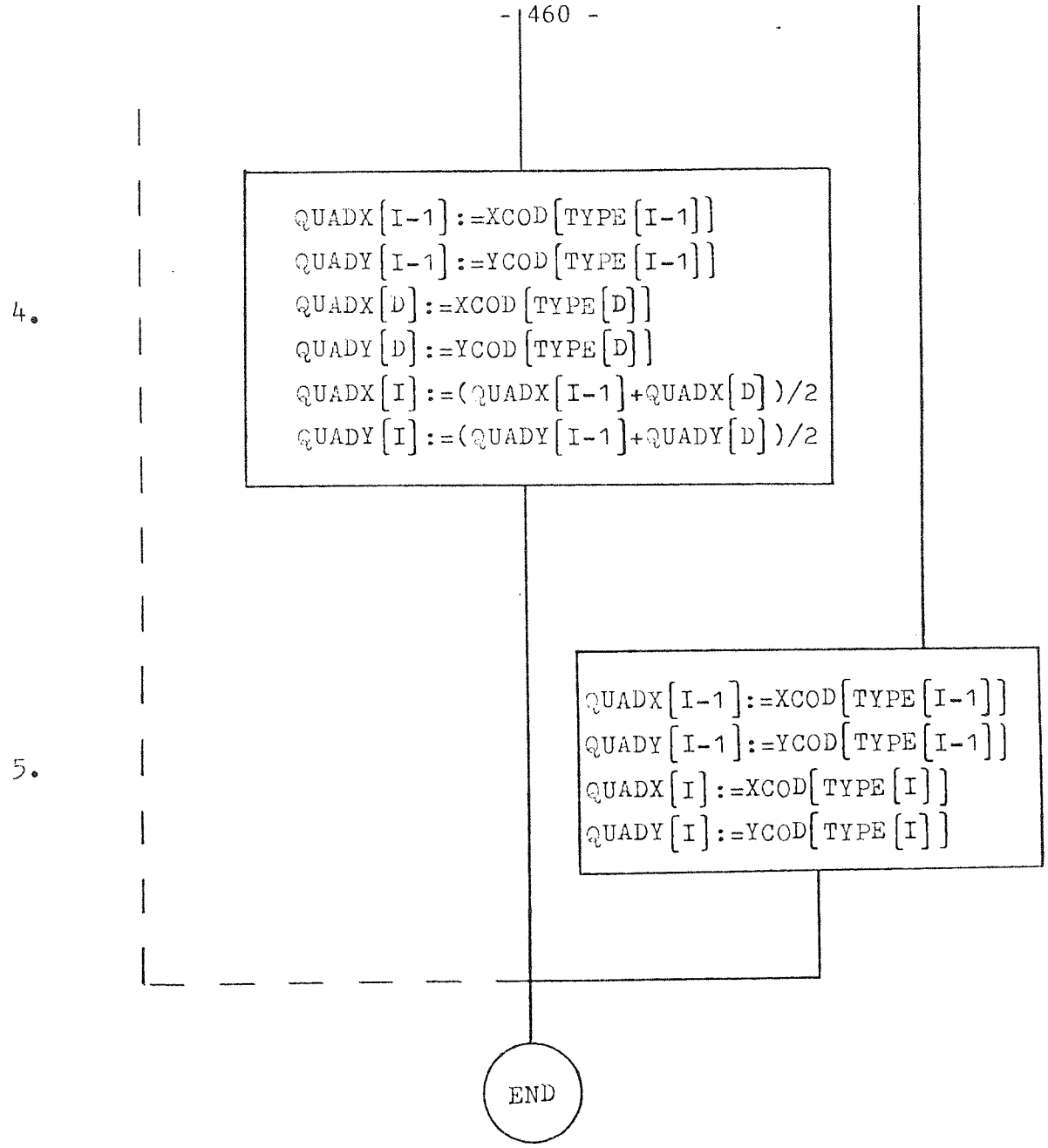
The object of this procedure is to determine the coordinates of the missing super-nodes by interpolation. Each zone is defined by eight super-nodes. In the construction of a finite element mesh the zone mid-side nodes can be used to grade the internal mesh or model curved boundaries. Hence all corner nodes must be specified, but in the case of straight sided zone faces the mid-side node can be omitted. As the super-node coordinates are read into the computer a flag is set, this takes the form of array VARN, which assumes the value of unity for each node read. Each zone can be pre-defined by a set of super-nodes using the zone grid dimensions, therefore, it is a simple matter to test whether the zone's mid-side nodes have been declared. This is achieved by examining its VARN value, if this is zero then interpolation is required. An array TYPE is employed to store the zone's nodal connections, which can be defined from a simple expression. This method of zone definition eliminates the need to enter equivalent data and therefore represents a reduction in labour. The procedure steps are as follows and correspond to the flowchart:-

1. The zone under examination is defined by eight super-node numbers which are stored in array TYPE.
2. The loop I is constructed so that the four zone faces can be tested. Variable D is set to unity if the last zone is under consideration.
3. The mid-side node is checked to determine if it has been specified in the input data, using the array VARN.
4. If the mid-side node has not been declared then its co-ordinates are interpolated from the adjoining corner nodes.

5. If the mid-side node has been entered then its coordinates together with the previous node's coordinates are recorded in arrays QUADX and QUADY for later use with procedures TR/QETETXY.

Flowchart for Procedure Zonexy :-




4.
QUADX[I-1]:=XCOD[TYPE[I-1]]
QUADY[I-1]:=YCOD[TYPE[I-1]]
QUADX[D]:=XCOD[TYPE[D]]
QUADY[D]:=YCOD[TYPE[D]]
QUADX[I]:=(QUADX[I-1]+QUADX[D])/2
QUADY[I]:=(QUADY[I-1]+QUADY[D])/2
5.
QUADX[I-1]:=XCOD[TYPE[I-1]]
QUADY[I-1]:=YCOD[TYPE[I-1]]
QUADX[I]:=XCOD[TYPE[I]]
QUADY[I]:=YCOD[TYPE[I]]
END

CHAPTER 10

REFERENCES

1. ZIENKIEWICZ, O.C., and CHEUNG, Y.K. 'The Finite Element Method in Structural and Continuum Mechanics', McGraw-Hill, New York, 1967.
2. MARTIN, C.H. and CAREY, F.C. 'Introduction to Finite Element Analysis', McGraw-Hill, New York, 1973.
3. COOK, R.D. 'Concepts and Applications of Finite Element Analysis', John Wiley & Sons, New York, 1974.
4. DESAI, C.S. and ABEL, J.F. 'Introduction to the Finite Element Method', Van Nostrand Reinhold, 1972.
5. THOMAS, K. 'Effects of Geometric Distortion on the Accuracy of Plane Quadratic Isoparametric Finite Elements', Guidelines for Finite Element Idealization, pp.161-203 of Meeting Preprint 2504, ASCE, 1975.
6. FRIED, I. 'Accuracy of Complex Finite Elements',AIAA J.,Vol.10, pp.347-349, 1972.
7. HENSHELL, R.D., WALTERS, D., and WARBURTON, G.B. 'On Possible Loss of Accuracy in Curved Finite Elements', J.Sound Vibration, Vol.23, pp.510-513, 1972.
8. HIBBETT, H.D. 'Some Properties of Singular Isoparametric Elements', Int.J.Num.Meth.Engrg., Vol.II, pp.180-184, 1977.
9. ANDERSON, E.W., 'An Engineer views Brittle Fracture History', Boeing Report, 1969.
10. GRIFFITH, A.A., 'The Phenomenon of Rupture and Flow in Solids', Phil.Trans.Roy.Soc.(London), Series A, Vol.221, pp.163-197, 1920.

11. WESTERGAARD, H.M. 'Bearing Pressures and Cracks', J.Appl.Mechs., Vol.6, p.49, 1939 .

12. IRWIN, G.R., 'Analysis of Stresses and Strains Near the End of a Crack Traversing a Plate', J.Appl.Mechs.,Vol.24, p.361, 1957.

13. IRWIN, G.R. 'Relation of Stresses Near a Crack to the Crack Extension Force', 9th Int.Congr.Appl.Mech.Brussels, 1957.

14. IRWIN, G.R. 'Fracture Dynamics', Fracture of Metals, ASME,pub., pp.147-166, 1948.

15. OROWAN, E. 'Energy Criteria of Fracture', Welding Resh.Supplement, Vol.20, pp.107S-160S

16. SIH, G.C., 'A Special Theory of Crack Propagation', Methods of Analysis and Solution to Crack Problems, (ed.by G.C.Sih), p.21, Noordhoff Int., 1973.

17. SIH, G.C. 'Some Basic Problems in Fracture Mechanics and New Concepts', Eng.Frac.Mech., Vol.5, pp.365-377, 1973.

18. IRWIN, G.C. and KIES, J.A. 'Critical Energy Rate Analysis of Fracture Strength', Weld.J., Vol.33, pp.193S-198S, 1954.

19. STRAWLEY, J.E., JONES, M.H. and GROSS, B. 'Experimental Determination of the Dependence of Crack Extension Force on Crack Length for a Single Notch Tension Specimen', NASA TND-2396, 1964.

20. HARDY, A.K. 'Determination of Stress Intensity Factors: an Evaluation of Two Methods', M.Sc.Thesis, Univ. of Bath, 1972.

21. EMERY, A.F., BARRETT, C.F. and KOBAYASHI, A.S. 'Temperature Distributions and Thermal Stresses in a Partially Filled Annulus', Expl.Mech., Vol.6, p.602, 1966.

22. SMITH, D.G. and SMITH C.W. 'A Photoelastic Evaluation of the Influence of Closure and other Effects upon the Local Bending Stresses in Cracked Plates, Int.J.Frac.Mech., Vol.6, pp.305-318,1970.

23. SMITH, D.G. and SMITH C.W. 'Determination of Mixed Mode Stress Intensity Factors', Eng.Frac.Mech., Vol.4, pp.357-366, 1972.

24. CARTWRIGHT,D.J. and ROOKE, D.P. 'Methods of Determining Stress Intensity Factors', R.A.E.Tech.Report, 73031, May 1973.

25. MUSKHELISHVILI, N.I. 'Some Basic Problems of the Mathematical Theory of Elasticity', Trans. by J.R.M. Radok, Noordhoff, 1963.

26. SIH, G.C. 'Application of Muskhelishvili's Method to Fracture Mechanics', Trans.Chinese Ass.for Advanced Studies, Vol.3, p.25,1962.

27. CARTWRIGHT, D.J. and ROOKE, D.P. 'Evaluation of stress Intensity Factors', J. of S.Anal., Vol.10, pp.217-224, 1975.

28. ROBERTSON, A.W. 'On the Stress Analysis of Cracked Bodies by means of Finite Elements', Ph.D., Univ. of A ston, 1976.

29. ALSHARQI, I.A. 'Finite Element Analysis of Axisymmetric Cracked Solids', Ph.D., Univ.of Aston, 1977.

30. CHAN, S.K., TUBA, I.S. and WILSON, W.K. 'On the Finite Element Method in Linear Fracture Mechanics', Int.J.Frac.Mech., Vol.7, pp.1-17, 1971.

31. KOBAYASHI, A., MAIDEN, D., SIMON, B. and IIDA, S. 'Application of the Method of Finite Element Analysis to Two-Dimensional Problems in Fracture Mechanics', Paper 69 WA-PVP-12, ASME, Nov.1969.

32. FRIED, I. and YANG, S.K. 'Best Finite Element Distribution Around a Singularity', AIAA.J., Vol.10, No.9, 1972.

33. TONG, P. and PIAN, T.H.H. 'On the Convergence of the Finite Element Method for Problems with Singularity', Int.J.Sol.Struct., Vol.9, pp.313-321, 1972.

34. BARSOUM, R.S. 'On the Use of Isoparametric Finite Elements in Linear Fracture Mechanics', Int.J.Num.Meth.in Engrg., Vol.10, No.1, pp.25-37, 1976.

35. BROCKHOVEN, M. 'Computation of Stress Intensity Factors for Nozzle Corner Cracks by Various Finite Element Procedures', 3rd. SMIRT Conf., London, Paper 94/6, 1975.

36. MURAKAMI, Y. 'A Simple Procedure for the Accurate Determination of Stress Intensity Factors by Finite Element Method', Engrg.Frac.Mech., Vol.8, No.4, pp.643-656, 1976.

37. PARKS, D.M. 'A Stiffness Derivative Finite Element Technique for Determination of Elastic Crack Tip Stress Intensity Factors', Int.J.Frac., Vol.10, No.4, pp.487-502, 1974.

38. HELLEN, T.K. 'On the Method of Virtual Crack Extensions', Int.J.for Num.Meth.in Engrg., Vol.9, No.1, pp.187-208, 1975.

39. FUKUDA, S., MIYAMOTO, H., KUJIRAI, Y., and SUMIKAWA, K. 'Prediction of Fatigue Crack Path by Finite Element Method', Trans. of J.W.R.I., Vol.6, pp.47-54, 1977.

40. RYBICKI, E.F. and KANNINEN, M. 'A Finite Element Calculation of Stress Intensity Factors by a Modified Crack Closure Integral', AIAAJ., Vol.11, 1977.

41. IRWIN, G.A. 'Fracture' in Handbook der Physik, Springer-Verlog, Vol.6, pp.551-590, 1958.

42. WESTERGAAD, H.M. 'Stresses at a Crack, Size of the Crack and the Bending of Reinforced Concrete', Proc. ACI, Vol.30, pp.93-102, 1963.

43. RICE, J.R. 'A Path Independent Integral and the Approximate Analysis of Strain Cpncentration by Notches and Cracks', Trans. ASME, J.Appl.Mech., Vol.35, pp.379-386, 1968.

44. ANDERSON, G.P., RUGGLES, V.L. and STIBOR, F. 'Use of Finite Element Computer Programs in Fracture Mechanics', Int.J.Frac.Mech., Vol.7, pp.63-76, 1971.

45. STERN, M., BECKER, E. and DUNHAM, R.S. 'A Contour Integral Computation of Mixed Mode Stress Intensity Factors', Int.J.Frac., Vol.12, pp.359-368, 1976.

46. STERN, M. and SONI, M. 'The Calculations of Stress Intensity Factors in Anistropic Materials by a Contour Integral Method in Computational Fracture Mechanics', E.Rybicki and S.Benzley, Eds., ASME Spec.Publ., 1975.

47. BYSKOV, E. 'The Calculations of Stress Intensity Factors Using the Finite Element Method with Cracked Elements', Int.J.Frac.Mech., Vol.6, pp.159-167, 1970.

48. FAWKES, A.J. 'An eight-noded element for K_I and K_{II} Determination', Proc., Int.Conf. on Num.Methods in Fracture Mech., Swansea, Wales, January 1978.

49. WILLIAMS, M.L. 'On the Stress Distribution at the Base of a Stationary Crack', J.Appl.Mech., pp.109-114, 1957.

50. JONES, R. and CALLINAN, R.J. 'On the Use of Special Crack Tip Elements in Cracked Elastic Sheets', Int.J.Frac., Vol.13, pp.51-64, 1977.

51. HILTON, P.D. and HUTCHINSON, J.W. 'Plastic Stress Intensity Factors for Cracked Plates', Harvard Univ. Rept., SM-43, May 1969.

52. WILSON, W.K. 'Some Crack Tip Finite Elements for Plate Elasticity in Fracture Toughness', Proc.of Fifth Nat.Symp. on Frac.Mech., ASTM STP514, 1971.

53. TRACEY, D.M. 'Finite Elements for Determination of Crack Tip Elastic Stress Intensity Factors', Engrg.Frac.Mech., Vol.3, pp.255-266, 1971.

54. BLACKBURN, W.S. 'Calculation of Stress Intensity Factors at Crack Tips Using Special Finite Elements in the Mathematics of Finite Elements', J.R.Whiteman, Ed., Academic Press, N.Y.,pp.327-336,1973.

55. HENSHELL, R.D. and SHAW, K.G. 'Crack Tip Elements are Unnecessary', Int.J.for Num.Meth.in Engrg., Vol.9, pp.495-509, 1975.

56. BARSOUM, R.S. 'On the Use of Isoparametric Finite Elements in Linear Fracture Mechanics', Int.J.for Num.Meth. in Engrg., Vo.10, pp.25-37, 1976.

57. BARSOUM, R.S. 'Triangular Quarter Point Elements as Elastic and Perfectly Plastic Crack Tip Elements', Int.J.Num.Meth. in Engrg., Vol.11, pp.85-98, 1977.

58. LYNN, P. and INGRAFFEA, A.R. 'Transition Elements to be used with Quarter Point Crack Tip Elements', Int.J.Num.Meth.in Engrg., Vol.12, pp.1031-1036, 1978.

59. HELLEN, T.K. and BLACKBURN, W.S. 'The Calculation of Stress Intensity Factors in Two-and Three-Dimensions Using Finite Elements in Computational Fracture Mechanics', E.Rybicki and S.Benzley, Eds., ASME Spec.Publ., 1975.

60. PIAN, T.H.H., TONG, P. and LUK, C. 'Elastic Crack Analysis by a Finite Element Hybrid Method', Proc.Third Air Force Conf. on Matrix Methods in Struct.Mech., Dayton, Ohio, 1971.

61. RICHARDS, T.H. and ROBERTSON, A.W. 'The Determination of Single and Mixed Mode Stress Intensity Factors for Engineering Components of Practical Interest', Fracture Mech. in Engrg.Practice, P.Stanley (Ed), App.Sci.Publishers, 1977.

62. WELLS, A.A. 'The Status of COD in Fracture Mechanics', Canadian Congress of App.Mech., 1971.

63. INGHAM, T.et al. 'The Effect of Geometry on the Inter-penetration of COD Test Data', Proc.of Conf. on Practical App. of Frac.Mech. to Pressure Vessel Technology, I.M.E.London, May 1971.

64. BROEK, D. 'Elementary Engineering Fracture Mechanics', Noordhoff Int., 1974.

65. IRWIN, G.R. 'Structural Mechanics', pp.557-594, Pergamon, New York, 1960.

66. TADA, H. 'Westergaard Stress Functions for Several Periodic Crack Problems', Engrg.Frac.Mech., Vol.2, pp.177-180, 1970.

67. NEALE, B.K. 'Finite Element Crack Tip Modelling when Evaluating the J-Integral', Int.J.Frac., Vol.11, No.1, pp.177-178, 1975.

68. KOBAYASHI, A., CHIU, S. and BEEUWKES, R. 'A Numerical and Experimental Investigation of the Use of the J-Integral', Engrg.Frac.Mech., Vol.5, pp.293-305, 1973.

69. JENNINGS, A. and TUFF, A.D. 'A Direct Method for the Solution of Large Sparse Symmetric Simultaneous Equations', Int.Conf.of Large Sparse Sets of Linear Equations, Oxford, 1970.

70. JENNINGS, A. and TUFF, A.D. 'Solution of Variable Bandwidth Positive Definite Simultaneous Equations', The Computer Journal, pp.446, 1972.

71. TIMOSHENKO, S.P. and GOODIER, J.N. 'Theory of Elasticity', 3rd. ed., McGraw-Hill, 1970.

72. AKYUZ, A.F. 'An Automatic Input Data Generation Scheme for a Finite-Element Method', Nuclear Engrg.& Design, Vol.11, pp.195-207, 1970.

73. REID, J.K. and TURNER, A.B. 'Fortran Subroutines for the Solution of Laplace's equation over a General Region in Two Dimensions', Atomic Energy Research Est't, Harwell, T.P.422, 1970.

74. ZIENKIEWICZ, O.C. and PHILLIPS, D.V. 'An Automatic Mesh Generation Scheme for Plane and Curved Surfaces by 'Isoparametric' Coordinates'. Int.J.Num.Meth.Engrg., Vol.3, pp.519-528, 1971.

75. SIH, G.C. and LIEBOWITZ, H. 'Mathematical Theory of Brittle Fracture', Chap.2 , Fracture: An Advanced Treatise, Vol.II, Ed., Liebowitz, H., Academic Press, 1968.

76. RICHARDS, T.H. 'On Using the Finite Element Method and Sub-region Singular Solutions in Fracture Mechanics', Department of Mechanical Engineering Report, University of Aston, November 1974.

77. RICHARDS, T.H. 'On Using the Finite Element Method in Conjunction with Sub-region Singular Solutions in Fracture Mechanics', Proc., Int.Conf.on Num.Methods in Fracture Mech., Swansea, Wales, pp.374-384, 1978.

78. RICE, J.R. 'Mathematical Analysis in Mechanics of Fracture', Chp.3, Fracture: An Advanced Treatise, Vol.2, Ed. H.Liebowitz, Academic Press, 1968.

79. CARTWRIGHT, D.J. and ROOKE, D.P. 'Compendium of Stress Intensity Factors', HMSO., 1975.

80. BUSINGER, P. and GOLUB, G.H. 'Linear Least Square Solutions by Householder Transformations', Numerische Mathematik, Vol.7, pp.269-276, 1965.

81. PETERS, G. and WILKINSON, J.H. 'The Least Squares Problem and Pseudo-Inverses', Computer Journal, Vol.13, pp.309-316, 1970.

82. RICHARDS, T.H. 'Energy Methods in Stress Analysis'- Pub. Ellis Horwood, 1977.

83. IVERSEN, P.A. 'Some Aspects of the Finite Element Method in Two-Dimensional Analysis', in Finite Element Methods in Stress Analysis, eds. I.Holland and K.Bell, Tapir Press, Trondheim, pp.93-114, 1969.

84. BOWIE, O.L. and FREESE, C.E. 'On the 'Overlapping' Problem in Crack Analysis', Engrg.Frac.Mech., Vol.8, pp.373-379, 1976.

85. INGRAFFEA, A.R. 'On Discrete Fracture Propagation in Rock Loaded in Compression', Proc., Int.Conf. on Num.Methods in Frac.Mech., Swansea, Wales, eds.A.R.Luxmore and D.R.J.Owen, pp.235-248, Jan.1978.

86. INGRAFFEA, A.R. 'Nodal Grafting for Crack Propagation Studies', Int.J.Num.Meth.in Engrg., Vol.11, No.2, 1977.

87. FUKUDA, S., MIYAMOTO, H., KUJIRAI, Y. and SUMKAWA, K. 'Prediction of Fatigue Path by Finite Element Method', Trans.of JWRI, June.1977.

88. STEINMUELLER, G. 'Restrictions in the Application of Auto-Mesh Generation by 'Isoparametric' Coordinates', I.J.N.M., Vol.8, pp.289-294, 1974.

89. HOEK, E. and BIENIAWSKI, Z.T. 'Fracture Propagation Mechanics in Hard Rocks', Tech.Rept., Rock Mech. Div., South African Council for Sc. and Indl. Reah., 1965.