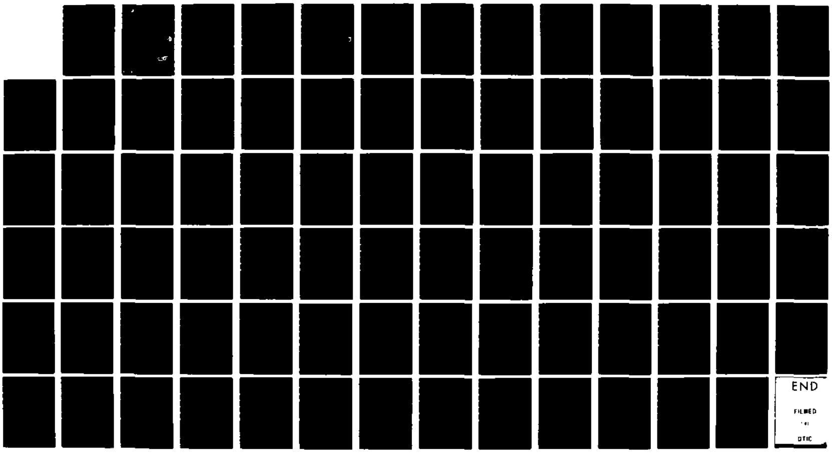
AD-A125 047 FEARS USER'S MANUAL FOR UNIVAC 1100(U) MARYLAND UNIV COLLEGE PARK LAB FOR NUMERICAL ANALYSIS C MESZTENYI ET AL. OCT 82 BN-991 N00014-77-C-0623 1/1

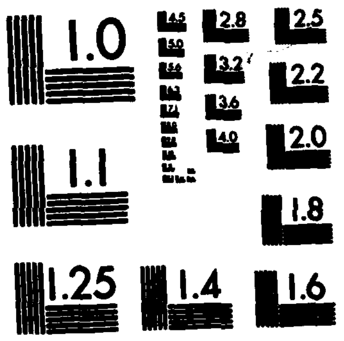
UNCLASSIFIED F/G 9/2 NL



END

FILMED

DTIC



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

INSTITUTE FOR PHYSICAL SCIENCE AND TECHNOLOGY

70

Laboratory for Numerical Analysis

Technical Note No. BN-991

FEARS USER'S MANUAL for UNIVAC 1100

by

C. Mesztenyi, W. Szymczak

October 1982

UNIVERSITY OF MARYLAND

Encl (1)

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER Technical Note BN-991	2. GOVT ACCESSION NO. AD-A125 047	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) FEARS USER'S MANUAL FOR UNIVAC 1100		5. TYPE OF REPORT & PERIOD COVERED final-life of the contract
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) C. Mesztenyi and W. Sztmczak		8. CONTRACT OR GRANT NUMBER(s) ONR N00014-77-C-0623
9. PERFORMING ORGANIZATION NAME AND ADDRESS Institute for Physical Science & Technology University of Maryland College Park, MD 20742		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Department of the Navy Office of Naval Research Arlington, VA 22217		12. REPORT DATE October 1982
		13. NUMBER OF PAGES 74
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report)
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release: distribution unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

The FEARS program normally takes the problem's descriptive data from the run-stream. This input data is composed of the Geometry (Chapter 1) and of the Bilinear/Error matrices (Chapter 2). Alternatively, this data can be read from a "DUMP-file", which has been generated by a previous execution of FEARS (section 3.13). Once the initial data is read, FEARS enters into a command-mode. In the command-mode, the user must input a command (with some parameters) instructing the program about the next step to be taken. The available commands are given in Chapter 3. This mode allows the user to have complete control over the interatio

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-LF-014-6601

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered) process, step by step, reflecting the research nature of the program. On the other hand, the ITERATE command allows the iterations to be automatic. Once a STOP command is given, the run will be terminated.

Printed output generated by FEARS consists of summary data after each iteration step. The user also has the option to reprint initial input and/or detailed information about the solution process during the iteration. The formats of these printouts are described in Chapter 4. In addition, the DUMP command generates a dump-file of the data structure which can be used as input, either for FEARS itself, or for various post-processors. The format of this dump-file is described in Section 4.7.

Chapter 5 gives the computer dependent control statements necessary to run FEARS on the UNIVAC 1100 computer. The program is supplied with a dummy subroutine giving zero values for the bilinear matrices E and D, (see Section 2.7). If these matrices are coordinate dependent, the user must supply the appropriate subroutine replacing the supplied dummy routine. This preparation must be performed prior to the use of FEARS.

FEARS

USER'S MANUAL

for UNIVAC 1100

by C. Mesztenyi, W. Szymczak

Technical Note No. BN-991

The work was partially supported by ONR Contract N00014-77-C-0623. The computation was partially supported by the Computer Science Center of the University of Maryland.

Table of Contents - FEARS User's Manual

0. Introduction

The FEARS program normally takes the problem's descriptive data from the run-stream. This input data is composed of the Geometry ~~(Chapter 1)~~ and of the Bilinear/Error matrices ~~(Chapter 2)~~. Alternatively, this data can be read from a "DUMP-file", which has been generated by a previous execution of FEARS. ~~(Section 3.13)~~. Once the initial data is read, FEARS enters into a command-mode. In the command-mode, the user must input a command (with some parameters) instructing the program about the next step to be taken. The available commands are given in Chapter 3. This mode allows the user to have complete control over the iteration process, step by step, reflecting the research nature of the program. On the other hand, the ITERATE command allows the iterations to be automatic. Once a STOP command is given, the run will be terminated.

Printed output generated by FEARS consists of summary data after each iteration step. The user also has the option to reprint initial input and/or detailed information about the solution process during the iteration. The formats of these printouts are described in Chapter 4. In addition, the DUMP command generates a dump-file of the data structure which can be used as input, either for FEARS itself, or for various post-processors. The format of this dump-file is described in Section 4.7.

Chapter 5 gives the computer dependent control statements necessary to run FEARS on the UNIVAC 1100 computer. The program is supplied with a dummy subroutine giving zero values for the bilinear matrices E and D. ~~(see Section 2.7).~~ If these matrices are coordinate dependent, the user must supply the appropriate subroutine replacing the supplied dummy routine. This preparation must be performed prior to the use of FEARS.

Chapter I. Geometry Input

1.1 Introduction -- 0-D, 1-D, and 2-D domains

The domain $\mathcal{D}$, on which the problem is to be solved, must be initially broken up into subdomains. Each subdomain is a generalized quadrilateral, having 4 corner points and 4 sides, with each side being either a straight line or an arc of a circle. Since a unit quare will be mapped onto each of these subdomains, they should not be "degenerate" or "singular". For example, the angles formed at the vertices should not be too near 0° or 180°, the domains should not be nearly triangular, and no overlapping of the sides is allowed. Thus, it is good practice to avoid subdomains as shown in Figure 1.1.

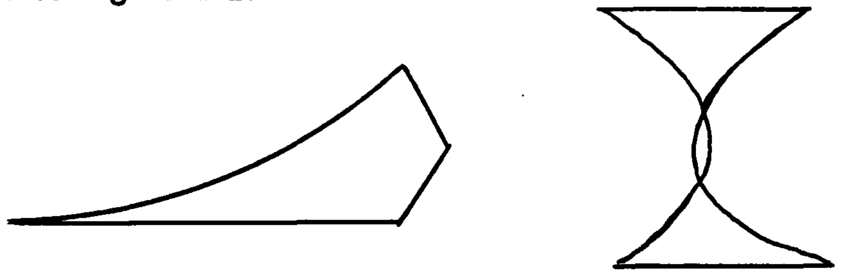


Figure 1.1. Subdomains to be avoided.

Figure 1.2 illustrates how a disk, and triangular and pentagonal domains may be substructured into generalized quadrilaterals.

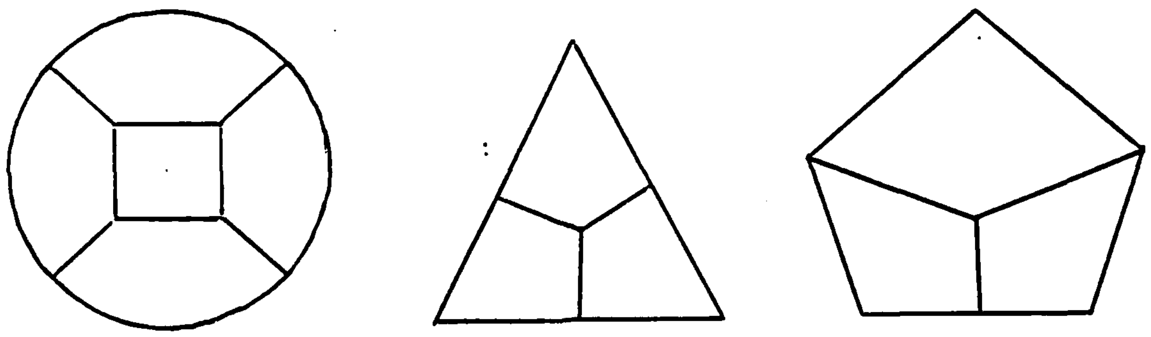


Figure 1.2. Subdomain structuring for a disk, triangular domain, and pentagonal domain.

Because FEARS uses blending functions to map the unit square onto each generalized quadrilateral (see FEARS Mathematical Description, Chapter I), domains having boundaries composed of circular arcs are represented exactly and are not approximated by isoparametric elements. Therefore, the approximations of the geometry and the exact solutions are not mixed, when using FEARS.

The corner points of the subdomains are simply called points or 0-D domains (zero dimensional domains). These are denoted by

$$\mathcal{D}_k^0 \qquad k=1,2,\ldots NO.$$

The open line segments joining the points are called lines or 1-D domains, and are denoted

$$\mathcal{D}_k^1 \qquad k=1,2,\ldots N1.$$

Finally, the open subdomains, each with 4 lines and 4 points forming its boundary, are called 2-D domains, and are denoted

$$\mathcal{D}_k^2 \qquad k=1,2,\ldots N2.$$

For example, the disk can be structured and labeled as in Figure 1.3,

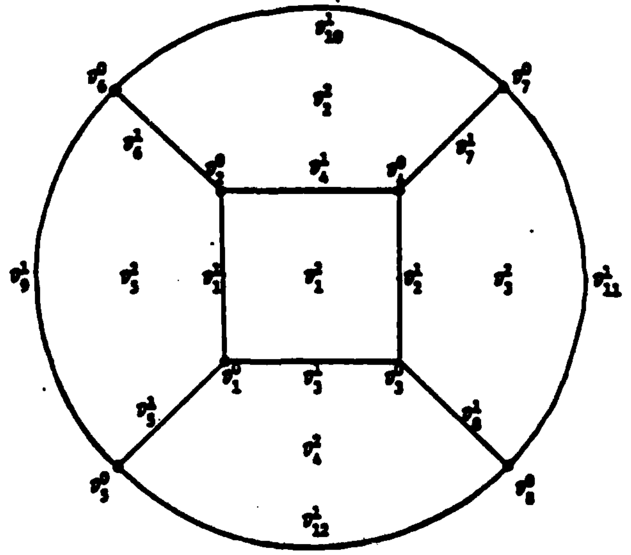


Figure 1.3. The substructuring and labeli*r*• of ˛ Disk.

In the geometry input the following information must be supplied:

(i) the number of points, lines, and 2-D domains,

(ii) the global coordinates of the points,

(iii) the end point indices for each line,

(iv) the radius of each line,

(v) the boundary conditions for each line and point,

and (vi) the cornerpoint indices of each 2-D domain.

1.2 Format

The format for the geometry input is as follows:

$$\left.\begin{array}{l} \qquad\quad NO \\ 1,\ x_1,\ y_1,\ b_1,\ v_1,\ w_1 \\ 2,\ x_2,\ y_2,\ b_2,\ v_2,\ w_2 \\ \cdot \\ \cdot \\ \cdot \\ NO,\ x_{NO},\ y_{NO},\ b_{NO},\ v_{NO},\ w_{NO} \end{array}\right\} \quad \text{points}$$

$$\left.\begin{array}{l} \qquad\quad N1 \\ 1,\ p_1,\ q_1,\ \beta_1,\ \rho_1 \\ 2,\ p_2,\ q_2,\ \beta_2,\ \rho_2 \\ \cdot \\ \cdot \\ \cdot \\ N1,\ p_{N1},\ q_{N1},\ \beta_{N1},\ \rho_{N1} \end{array}\right\} \quad \text{lines}$$

$$\left.\begin{array}{l} \qquad\quad \pm N2 \\ 1,\ \pi_1^1\ \ \pi_1^2\ \ \pi_1^3,\ \pi_1^4 \\ 2,\ \pi_2^1,\ \pi_2^2,\ \pi_2^3,\ \pi_2^4 \\ \cdot \\ \cdot \\ \cdot \\ N2,\ \pi_{N2}^1,\ \pi_{N2}^2,\ \pi_{N2}^3,\ \pi_{N2}^4 \end{array}\right\} \quad \text{2-D domains}$$

1.3. Description of the Format

N0 is the number of points (0-D's). The data, $i, x_i, y_i, b_i, v_i, w_i$, describe the points as follows:

i = the index of $\mathcal{D}_i^0$, $i=1,\ldots,N0$,

(x_i,y_i) = the x-y global coordinates of $\mathcal{D}_i^0$,

b_i = the boundary condition at $\mathcal{D}_i^0$, where

$$b_i = \left\{\begin{array}{l} 0 \Rightarrow u_1 \text{ free, } u_2 \text{ free,} \\ 1 \Rightarrow u_1 \text{ fixed, } u_2 \text{ free,} \\ 2 \Rightarrow u_1 \text{ free, } u_2 \text{ fixed,} \\ 3 \Rightarrow u_1 \text{ fixed, } u_2 \text{ fixed} \end{array}\right\}, \text{ and}$$

(v_i,w_i) = the solution value (u_1,u_2) at $\mathcal{D}_i^0$. These must be given even if the boundary condition is free, in which case any value for v_i,w_i may be given.

N1 is the number of lines (1-D's). The data j, p_j, q_j, β_j, ρ_j describe the lines as follows:

j = the index of $\mathcal{D}_j^1$, $j = 1,\ldots,N1$,

(p_j,q_j) = the index numbers of the endpoints of the line $\mathcal{D}_{p_j}^0$, $\mathcal{D}_{q_j}^0$,

β_j = the boundary conditions of line $\mathcal{D}_j^1$, where

$$\beta_j = \left\{\begin{array}{l} 0 \Rightarrow u_1 \text{ free, } u_2 \text{ free,} \\ 1 \Rightarrow u_1 \text{ fixed, } u_2 \text{ free,} \\ 2 \Rightarrow u_1 \text{ free, } u_2 \text{ fixed,} \\ 3 \Rightarrow u_1 \text{ fixed, } u_2 \text{ fixed,} \\ 5 \Rightarrow u_1 \text{ linear, } u_2 \text{ free} \\ 6 \Rightarrow u_1 \text{ free, } u_2 \text{ linear} \\ 7 \Rightarrow u_1 \text{ linear, } u_2 \text{ linear.} \end{array}\right\}$$

If a component u_1, or u_2 is fixed or linear, then the same component must be fixed at the endpoints $\mathcal{D}^o_{q_j}$, and $\mathcal{D}^o_{p_j}$. Furthermore, if it is fixed then the two values given at these endpoints must be identical.

$\rho_j = \pm 1/R_j$ is the signed reciprocal of the radius of the arc segment $\mathcal{D}^1_j$. The orientation determines the sign of the radius as shown in Figure 1.4.

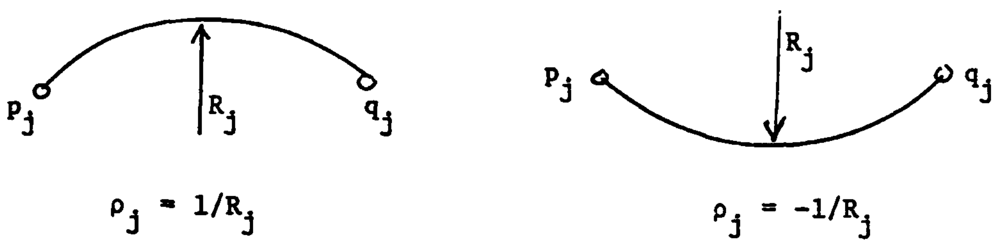


Figure 1.4. Orientation of the arcs.

N2 is the number of 2-D domains. The 2-D domains are prescribed by the 4 corner point indices if N2 is not preceded by a minus sign, and by the 4 boundary line indices if N2 is preceded by a minus sign.

The first input in each line after N2 is k = the index of $\mathcal{D}^2_k$, k=1,...,N2. When no minus sign precedes N2, π^1_k, π^2_k, π^3_k, π^4_k = the index numbers of the 4 cornerpoints of $\mathcal{D}^2_k$, in the order shown in Figure 1.5.

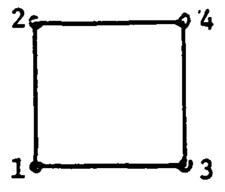


Figure 1.5. Order of the 4 cornerpoint indices for a 2-D domain.

The user should be careful not to change this orientation. For example, the indices cannot be input in the order as displayed in Figure 1.6.

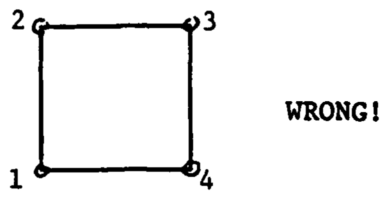


Figure 1.6. Illegal ordering of the 4 cornerpoint indices for a 2-D domain.

When a minus sign precedes N2, π_k^1, π_k^2, π_k^3, π_k^4 = the index numbers of the 4 boundary lines of $\mathcal{D}_k^2$ in the order shown in Figure 1.7.

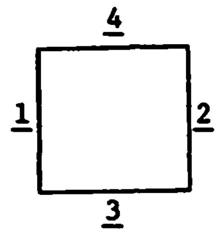


Figure 1.7. Order of the 4 boundary line indices for a 2-D domain.

Chapter II. Bilinear and Error Matrices Input

2.1. Introduction

In the previous section, it was seen that FEARS works on a substructured domain, which allows a simple input and a more accurate representation of the geometry. The substructuring has another advantage, namely, it also allows for a different set of material properties (bilinear matrices) to be described on each 2-D domain.

2.2. Variational Form

The variational problem, solvable by FEARS, can be put in the general form: find U in M_1, such that

$$\sum_{i=1}^{N2} \int_{\mathcal{D}_i^2} \left\{ \left[\frac{\partial V}{\partial Z}\right]^T A_i \left[\frac{\partial U}{\partial Z}\right] + V^T B_i \left[\frac{\partial U}{\partial Z}\right] + \left[\frac{\partial V}{\partial Z}\right]^T B_i^T U + V C_i U \right\} dx\,dy$$

$$+ \sum_{j=1}^{N1} \int_{\mathcal{D}_j^1} V^T \gamma_j U \, ds = \sum_{i=1}^{N2} \int\!\!\int_{\mathcal{D}_i^2} \left\{ \left[\frac{\partial V}{\partial Z}\right]^T D(x,y) + V^T E(x,y) \right\} dx\,dy + \sum_{j=1}^{N1} \int_{\mathcal{D}_j^1} \varepsilon_i(s) V \, ds \; ,$$

for all V in M_2 , where M_1 and M_2 are the appropriate finite dimensional trial and test spaces (see FEARS Mathematical Description).

Here, we have used the notation

$$U = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}, \quad \frac{\partial U}{\partial Z} = \begin{bmatrix} \frac{\partial u_1}{\partial x} \\ \frac{\partial u_2}{\partial x} \\ \frac{\partial u_1}{\partial y} \\ \frac{\partial u_2}{\partial y} \end{bmatrix},$$ and analogously for $\frac{\partial V}{\partial Z}$ and V .

Also,

A_i is a 4 x 4 symmetric matrix,

B_i is a 2 x 4 matrix,

C_i is a 2 x 2 symmetric matrix

$D_i(x,y)$ is a 4 x 1 vector valued function, and

$E_i(x,y)$ is a 2 x 1 vector valued function, for $1 \leq i \leq N2$.

γ_j is a 2 x 2 symmetric matrix, and

$\varepsilon_j(s)$ is a 1 x 2 vector valued function for $1 \leq j \leq N1$.

2.3 Error Norms

The error is approximated in the norm $|||\cdot|||_{2p}$, where*

$$|||U|||_{2p} = \left[\sum_{j=1}^{N2} \int_{\mathcal{D}_j^2} \left\{ \left[\frac{\partial U}{\partial Z}\right]^T \left(A_E\right)_j \left[\frac{\partial U}{\partial Z}\right] \right\}^{p_j} dx\, dy \right]^{1/2p} \tag{2.1}$$

The value p is a global value which is input before we input the geometry and bilinear forms. The values p_j are to be specified for each 2-D domain. Normally $p_1 = p_2 = \quad = p_{N2} = p$, which will correspond to a global L_{2p} estimate. The corresponding L_∞ norm (2.2), is used when $p_i = 0$.

$$|||U|||_{L_\infty\left(\mathcal{D}_i^2\right)} = \sup_{x,y \in \mathcal{D}_j^2} \sqrt{\left|\left[\frac{\partial U}{\partial Z}\right]^T \left(A_E\right)_j \frac{\partial U}{\partial Z}\right|} \tag{2.2}$$

$\left(A_E\right)_j$ is a 4 x 4 matrix, and usually, $(A_E)_j = (A)_j$.

2.4 General Format

The format for the input of the bilinear, error, and output matrices is as follows:

*That $|||\cdot|||_{2p}$ is a norm, and not a seminorm for the error e , follows from the fact that if $|||e|||_{2p} = 0$, the e must be constant. Since the approximate solution is bilinear, the exact solution must be bilinear as well. Because of quasi-optimality of the finite element solution, $e=0$.

NB

$$\left\{\begin{array}{ll} j,\ n_j,\ k_1,\ k_2,\ldots,\ k_{n_j} & \\ \alpha_j,\ \beta_j,\ \gamma_j,\ \delta_j,\ \varepsilon_j & \\ \quad (A)_j & (\text{if } \alpha_j \neq 0) \\ \quad (B)_j & (\text{if } \beta_j \neq 0) \\ \quad (C)_j & (\text{if } \gamma_j \neq 0) \\ \quad (D)_j & (\text{if } \delta_j \neq 0) \\ \quad (E)_j & (\text{if } \varepsilon_j \neq 0) \\ \quad \left({}^{A}{}_{E}\right)_j & \\ \quad \left({}^{N}{}_{c}\right)_j & \\ \quad (S)_j & \end{array}\right\}$$

(Bilinear and Error Matrices)

(This packet is repeated NB times, j=1,...,NB.)

(Output Matrix)

NL

$$\left\{\begin{array}{l} i,\ ni,\ \ell_1,\ \ell_2,\ldots,\ell_{n_i} \\ \quad g_i,\ e_i \\ \quad (\gamma)_i \\ \quad (\varepsilon)_i \end{array}\right\}$$

(Line Integration Matrices)

(This packet is repeated NL times, i=1,...,NL.)

2.5 Description of Bilinear and Error Matrices

NB is the number of different packages of bilinear and error matrices. Each different package must then be listed. In each package

j = the index number of the package,

n_j = the number of 2-D domains for which the package of matrices applies,

$k_1,\ldots,k_{n_j}$ = the indices of the 2-D domains for which the package applies.

α_j, β_j, γ_j, δ_j, ϵ_j, are integers which indicate whether or not the matrices $(A)_j$,$(B)_j$,$(C)_j$,$(D)_j$,$(E)_j$ are zero, in which case its corresponding input line is not present.

$$\alpha_j = \begin{cases} 0 \Rightarrow (A)_j \text{ is zero, no } (A)_j \text{ input line,} \\ 1 \Rightarrow (A)_j \neq 0 \text{ , } (A)_j \text{ line is present,} \end{cases}$$

$$\beta_j = \begin{cases} 0 \Rightarrow (B)_j = 0 \text{ , no } (B)_j \text{ input line,} \\ 1 \Rightarrow (B)_j \neq 0 \text{ , } (B)_j \text{ line is present,} \end{cases}$$

$$\gamma_j = \begin{cases} 0 \Rightarrow (C)_j = 0 \text{ , no } (C)_j \text{ input line,} \\ 1 \Rightarrow (C)_j \neq 0 \text{ , } (D)_j \text{ line is present,} \end{cases}$$

$$\delta_j = \begin{cases} 0 \Rightarrow (D)_j = 0 \text{ , no } (C)_j \text{ input line,} \\ 1 \Rightarrow (D)_j \neq 0 \text{ , D is constant and is defined in the } (D)_j \text{ line,} \\ -1 \Rightarrow (D)_j \neq 0 \text{ , D is coordinate dependent and must be defined} \\ \qquad \text{in a subroutine; dummy values must be supplied} \\ \qquad \text{in the } (D)_j \text{ line,} \end{cases}$$

$$\epsilon_j = \begin{cases} 0 \Rightarrow (E)_j = 0 \text{ , no } (E)_j \text{ input line,} \\ 1 \Rightarrow (E)_j \neq 0 \text{ , E is constant and is defined in the } (E)_j \text{ line,} \\ -1 \Rightarrow (E)_j \neq 0 \text{ , E is coordinate dependent, and must be defined} \\ \qquad \text{in a subroutine and some dummy values must} \\ \qquad \text{be input in the } (E)_j \text{ line,} \end{cases}$$

Note: If $\delta_j = -1$ or $\epsilon_j = -1$ a subroutine must be defined. This is described in Section 2.7.

The $(A)_j$ line should contain the coefficient values of the 4 x 4 matrix $(A)_j$ in the following order,

$$\left[\begin{array}{cc|cc} 1 & 3 & 9 & 11 \\ 2 & 4 & 10 & 12 \\ \hline 5 & 7 & 13 & 15 \\ 6 & 8 & 14 & 16 \end{array}\right]$$

in free format. The appropriate matrix A, when using plane stress or plane strain elasticity, is given in the Appendix (A.2).

The $(B)_j$ line should contain the matrices B and B^T, where

$$B = \left[B_1, \; B_2\right] \quad \text{and} \quad B_j = \begin{bmatrix} b_{11}^{(j)} & b_{12}^{(j)} \\ b_{21}^{(j)} & b_{22}^{(j)} \end{bmatrix} , \; j = 1,2 .$$

The matrices should be input in the form

$$\left[\begin{array}{c|c} B_1 & B_1^T \\ \hline B_2 & B_2^T \end{array}\right] ,$$ the components

of this matrix being input in the same order as with $(A)_j$.

The $(C)_j$ line should contain the coefficient values of the 2×2 matrix $(C)_j$ in the order

$$\begin{bmatrix} 1 & 3 \\ 2 & 4 \end{bmatrix} .$$

The $(D)_j$ line should contain the coefficients of the 2×2 matrix $(D)_j$

The $(E)_j$ line should contain the entries of the 1×2 vector $(E)_j$ in the order

$$e_1, \; e_2 \quad .$$

The $(A_E)_j$ line should contain the entries for the 4×4 error matrix $(A_E)_j$ in the same order as the entires for the matrix $(A)_j$.

The $(N_c)_j$ line is a line where we input 4 parameters for the run,

$$p_j, r_j, w_j, x_j \; .$$

p_j = the p norm for the domains (see 2.1).

If $\frac{1}{2} \leq p_j < \infty$ then our indicators will be based on an L_{2p_j} estimate over these domains. The value $p_j=0$, corresponds to an L_∞ based error estimate.

r_j = the weight for the residual part of the error indicator which is computed through an integration instead of a derivative jump. See FEARS Mathematical description for more details on this. For most cases set $r=1$.

w_j = a parameter which was formerly used in the residue computation. Always set $w_j=1$.

x_j = a free parameter. Input anything here. This value is overwritten in the program.

At the end of each packet we must input the matrix for output generation $(S)_j$. $(S)_j$ is a 5 x 6 matrix and should be input in the order

$$\begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 7 & 8 & 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 & 17 & 18 \\ 19 & 20 & 21 & 22 & 23 & 24 \\ 25 & 26 & 27 & 28 & 29 & 30 \end{bmatrix}$$

In the output FEARS will compute the product

$$[S]_j \begin{bmatrix} \frac{\partial U}{\partial Z} \\ U \end{bmatrix} \text{ for 2-D domain } j \text{ .}$$

The appropriate formulas S_j , needed to yield the five values U_1, U_2, σ_{xx}, σ_{yy}, σ_{xy}, for the plane strain and plane stress assumptions of elasticity are given in the Appendix (A.2).

2.6 Line Integration Boundary Conditions

NL is the number of different packages of line integral matrices. These line integrals arise from boundary conditions in which traction or pressure forces are present. These forces can be either globally defined, if the force is in a fixed global direction, or locally defined, if for example, the force is normal to the boundary as with a hydrostatic force. The vector ε should give the magnitude of the force in the x and y directions, if the force is global, and in the tangential and normal directions, if the force is locally defined. The 2 x 2 matrix γ is present whenever the force depends on the displacement (eg. a spring force).

Mathematically the boundary conditions allowed are of the form

$$\left[A\frac{\partial U}{\partial n}\right] + \gamma U = \varepsilon \text{ , where}$$

$$A\frac{\partial U}{\partial n} = \begin{bmatrix} n_x & 0 & n_y & 0 \\ 0 & n_x & 0 & n_y \end{bmatrix} \left[A\frac{\partial U}{\partial Z}\right] \quad \text{, where}$$

(n_x,n_y) is the outward normal to the boundary and $\left[\frac{\partial U}{\partial Z}\right]$ is defined in Section 2.2.

In terms of stresses the boundary conditions are of the form

$$T + \gamma U = \varepsilon \text{ ,}$$

where

$$T = \begin{bmatrix} T_x \\ T_y \end{bmatrix} = \begin{bmatrix} \sigma_{xx} M_x + \sigma_{xy} M_y \\ \sigma_{xy} M_x + \sigma_{yy} M_y \end{bmatrix}$$

$$= \begin{bmatrix} \text{force in } x \text{ direction} \\ \text{force in } y \text{ direction} \end{bmatrix} .$$

When the forces are specified locally they are transformed into global forces, which change in direction and possibly magnitude, with respect to the arc length of the boundary line.

After NL is specified, each of the NL packets of line integrals must be specified. In the line following NL, we must input

i = the index number of the package,

n_i = the number of 1-D domains to which this package applies,

$\ell_1,\ldots,\ell_{ni}$ = the indices of the 1-D domains to which the package applies.

The next line contains indicators g_i, e_i where

$$g_i = \begin{cases} 0 \text{ , } (\gamma)_i = 0 \text{ but dummy values in the data line for } (\gamma)_i \\ \quad \text{must be input,} \\ 1 \text{ , } (\gamma)_i \neq 0 \text{ ,} \end{cases}$$

$$e_i = \begin{cases} -1 \text{ , } (\varepsilon)_i \neq 0 \text{ and the values of } (\varepsilon)_i \text{ describe a local force} \\ \quad \text{on the boundary,} \\ 0 \text{ , } (\varepsilon)_i = 0 \text{ but dummy values for the data line } (\varepsilon)_i \text{ must} \\ \quad \text{be input,} \\ 1 \text{ , } (\varepsilon)_i \neq 0 \text{ and the values of } (\varepsilon)_i \text{ describe a global force} \\ \quad \text{on the boundary.} \end{cases}$$

The $(\gamma)_i$ line should contain the entries of the 2 x 2 matrix γ in the order

$$\begin{pmatrix} 1 & 3 \\ 2 & 4 \end{pmatrix}$$

The $(\varepsilon)_i$ line should contain the four entries $\varepsilon_1(p)$, $\varepsilon_2(p)$, $\varepsilon_1(q)$, $\varepsilon_2(q)$, where p and q are the endpoints of the boundary line. If the force is described globally then ε_1 is the force in the x direction and ε_2 is the force in the y direction. If the force is described locally then ε_1 and ε_2 denote the forces in the tangential and normal directions respectively, to the boundary.

For example, suppose we wish to prescribe a force of magnitude M which makes an angle of α with the tangent. Then the appropriate values for ε

should be

$$M \cos \alpha, M \sin \alpha, M \cos \alpha, M \sin \alpha.$$

Furthermore, the angle and magnitude can be changed linearly (with respect to the arc length) from α and β and M_0 to M_1, respectively by inputting $M_0 \cos \alpha, M_0 \sin \alpha, M_1 \cos \beta, M_1 \sin \beta$ for ε.

2.7 Subroutines for the functions E(x,y), D(x,y).

If δ_j or ε_j, the indicators for the vectors D and E, are equal to -1, we must supply a fortran subroutine to define these functions.

The subroutines will read in the values (x(1),x(2)), the global (x,y) coordinatesof a point and then compute and return the values

S(1) = the exact solution u(1) (if known),

S(2) = the exact solution u(2) (if known),

$$\left.\begin{matrix} D(1) \\ D(2) \\ D(3) \\ D(4) \end{matrix}\right\} = \text{the 4 scalar functions comprising the components of the vector } D,$$

$$\left.\begin{matrix} E(1) \\ E(2) \end{matrix}\right\} = \text{the 2 scalar functions comprising the components of the vector } E,$$

$$\left.\begin{matrix} W(1) = \frac{\partial}{\partial x}D(1) + \frac{\partial}{\partial y}D(3) \\ W(2) = \frac{\partial}{\partial x}D(2) + \frac{\partial}{\partial y}D(4) \end{matrix}\right\} \text{the derivatives needed for the residue computation.}$$

The subroutine actually has 4 entry points, returning the values S, D, E, and W, respectively. The calling sequence is

```
SUBROUTINE ZPMTRU(X,S)
DIMENSION X(2), S(2)
```

(returns the exact solution S(1), and S(2),

```
COMMON/FUPAR/NSUP,FUP(12)
```

(Common block of function parameters. NSUP=the number of parameters, and FUP(12)=the function parameters--as dimensioned NSUP $\leq$ 12. These parameters are input at the onset of the program (see Section 5.2.)

```
ENTRY DMATX(X,D)
DIMENSION D(4)
```

(returns the components D(1), D(2), D(3), D(4)),

```
ENTRY EMATX(X,E)
DIMENSION E(2)
```

(returns the values E(1), and E(2))

and

```
ENTRY DMATXD(X,W)
DIMENSION W(2)
```

(returns the values W(1), and W(2)) .

This subroutine must be compiled and mapped with the main program. This is briefly discussed in Chapter 5.

Chapter III. Commands and Strategies

3.1 Introduction--Overview of Strategies

Before describing in detail the various options and commands available to the user, we give a brief overview of how FEARS operates. We hope this will help clarify what each instruction actually does.

Let us assume that the geometry and bilinear matrices have been input. An initial subdivision is then performed by the program, and a solution is obtained on this initial mesh. Error indicators are then computed from the solution values. A status message or REPORT is then printed out for this initial solution.

After this initial step, the user has many options for subsequent subdividing (refining) and resolving. In FEARS this iterative process of subdividing and resolving is continued until either the solution obtained is within a specified tolerance of the true solution, or the user runs out of computer resources (money, time, or storage). Each REPORT message contains the approximate relative error as part of its output.

Ideally we would like to employ some optimal strategy which will obtain for us the desired accuracy with the least computer expense. On one hand, an "optimal mesh" is always desired, that is, a mesh which will yield the smallest error in the solution for a fixed number of degrees of freedom. On the other hand,we would not like to spend too much money in order to maintain an optimal mesh at each level. For example, it would be very expensive and wasteful if we subdivided only one or two elements at a time and then recomputed the entire solution after each subdivision. Thus, even though we may get a better mesh by subdividing only one element at a time, it would be a better strategy to refine a larger number of elements, even though the mesh obtained may be only "nearly optimal". It has been shown *(1) that

*(1)Babuska, I., and Rheinboldt, W. Analysis of finite-element meshes in $\mathbb{R}^1$. Math Comput. 33 (1979), 435-463).

(at least for one dimensional problems) a mesh which deviates slightly from the optimal mesh is nearly optimal in the sense that the solution with this mesh is nearly as accurate as the solution on the optimal mesh.

Furthermore, one may ask if it is necessary to resolve the problem globally after each subdivision. For example, there may be circumstances where only one or two elements will get subdivided, or all the elements to be subdivided are concentrated in one region. Perhaps it would be acceptable to resolve the problem locally--either within each element, or only within a 2-D domain where some subdivision has occurred. The FEARS program allows us these options.

For example, we can specify that when some element gets subdivided, all previously obtained solution values will remain fixed, and a new solution will be obtained only for the node created at the center of the element by the subdivision. This is referred to as a SHORT path solution. Error indicators are recomputed for only the 4 new elements. SHORT path solutions are fast and cheap and are recommended when only a few elements are to be subdivided.

Likewise we may specify some set of 2-D domains, for example, only those where subdivision has occurred, and then obtain a new solution values only for those 2-D domains. The boundary conditions on the boundary of the subdomain are taken to be fixed, with displacement values determined from the previous solution. This type of solution path is in between a SHORT path and full solution in both expense and accuracy.

Although the user can control the refinement procedure by specifying which elements are to be subdivided before each solution path FEARS also has a built in recommended refinement strategy. This strategy is enacted through the command AUTO (short for automatic). Each time this command is given, all those elements with error indicators larger than some computed threshold value (see Appendix A3) are subdivided and a new solution is computed. The type of solution path performed must be supplied by the user. For example, (AUTO/1) will refine and recompute the solution globally, and (AUTO/4) will refine and then perform a SHORT path solution on each refined element only.

For many problems, particularly those in which there are no singularities, a sequence of (AUTO/1) commands is the best strategy for obtaining an accurate solution cheaply. However, when solving problems with singularities, it is often the case that an AUTO command will refine only one or two elements. If the mesh already has a large number of elements, producing a new global solution in this case is not only costly, but also unnecessary. In this case we would prefer performing a SHORT path solution with an (AUTO/4) instead of a new global solution with (AUTO/1). If the program is being run interactively, then the user can decide which type of solution should be performed, since after each REPORT the approximate number of elements to be subdivided by the next AUTO command is printed.

Unfortunately, if the program is being run as a batch job there is no a-priori way to determine when an (AUTO/1) or (AUTO/4) should be performed. FEARS also has the ability to make this choice automatically with the ITER (ITERATE) command. The ITER command performs n solution paths composed of (AUTO/1) or (AUTO/4) commands, the choice depending on whether the number of elements in the new mesh is a certain percentage over the number in the old mesh (after the last (AUTO/1)). This cut-off percentage must be supplied by the user, but a 30% increase is recommended.

The user also has control over what information is printed out, after a solution path is performed. For example, with the PRINT command, information about the solution at the nodes and a list of elements with their a-posteriori error indicators may be printed. The OUTPUT command will give a list of stresses and solution values at the center of each element.

Two other useful commands are the DUMP command and the CHANGE command. The DUMP command will cause all information about the problem, eg. geometry, bilinear matrices, solution values, and data structures for the mesh, to be saved on a permanent file. The user may then restart the program where he/she left off at any future time.

The CHANGE command causes small changes in the initial problem. This is useful if one is interested in the effect of perturbing either the geometry or material properties of the prolem. In this case, the refined mesh for the original problem will be almost topologically equivalent to the refined mesh for the next problem. Therefore, instead of using a lot of computer time by iteratively subdividing and resolving for the new problem, the refined mesh for the original problem could be used and a final solution obtained immediately.

Now that the general format has been presented, we describe the user commands in detail. The computer will always acknowledge that it is ready to receive a user command by printing the line

```
**** COMMAND:
```

3.2 PRINT Command

The PRINT command is designed to print out information about the

points - D_j^0 , $j=1,\ldots,N0$, the

lines - D_j^1 , $j=1,\ldots,N1$, and the

2-D domains - D_j^2 , $j=1,\ldots,N2$.

The format for the PRINT command is

```
PRINT
a,b,c
```

The value, a, determines the dimension of the domains to be printed.

$$a = \begin{cases} 0 \Rightarrow \text{print about } D_j^0\text{'s} \\ 1 \Rightarrow \text{print about } D_j^1\text{'s} \\ 2 \Rightarrow \text{print about } D_j^2\text{'s} \\ -1 \Rightarrow \text{print about } D_j^0\text{'s, } D_j^1\text{'s and } D_j^2\text{'s .} \end{cases}$$

The value, b, determines which index k of $D_k^a (a \neq -1)$ is to be printed.

$$b = \begin{cases} \geq 1 = k \text{ , print about } D_k^a \text{ only.} \\ -1 \quad \text{print all } D_k^a\text{'s .} \end{cases}$$

The value, c, determines how much information is to be printed.

$$c = \begin{cases} 0 \quad \text{print only headings} \\ 1 \quad \text{only print data about the nodal points of the 2-D domain(s).} \\ 2 \quad \text{only print data about the elements of the 2-D domain(s).} \\ 3 \quad \text{print all information} \end{cases}$$

For example, the command

```
PRINT
-1,-1,3
```

will cause all information about each 0-D, 1-D and 2-D domain to be printed. The format of the printed data and its interpretation are discussed in detail in Chapter 4.

After the execution of a PRINT command the computer will return to the command mode and print **** COMMAND .

3.3 REPORT Command

The REPORT command accomplishes two things. First of all, it computes statistical information about the error indicators and sorts the elements in the order of decreasing indicators. Thus, a REPORT should be performed before each PRINT command to ensure that the elements are listed in order. Secondly, it prints out a status report on the full domain giving information on the number of elements, total energy, error estimator, percentage of error, number of elements recommended for subdivision, etc. This message is described in detail in Chapter 4-Output.

3.4 SUBDIVIDE Command

The SUBDIVIDE command is used to subdivide elements. It has the following format:

```
computer    **** COMMAND:
user        SUBDIVIDE
computer    SUBDIVISION, 2-D PROCESS INDEX:
user         J1
computer    0.=ALL, GT.0.=CUT OFF VALUE,
            LT.0. = GIVE ELEMENT LIST
user        X1
computer  ┌ ELT LVL R LOCAL COORD ERR.IND PREV ERRIND ┐
          │  .   .  .   .            .            .   │
          │  .   .  .   .            .            .   │
          │  .   .  .   .            .            .   │
          │  .   .  .   .            .            .   │
          │ ELEMENT TO BE SUBDIVIDED:                 ├ IF X1 < 0
user      │ e1                                        │
computer  │ ELEMENT TO BE SUBDIVIDED:                 │
          └                                           ┘
```

```
user       ⎧ e_N                                  ⎫
computer   ⎨ ELEMENT TO BE SUBDIVIDED:            ⎬ IF X_1 < 0
user       ⎩ 0        (zero)                      ⎭
computer     SUBDIVISION 2-D PROCESS INDEX:
user         J_2
computer     0 = ALL, GT.O. = CUT OFF VALUE,
             LT.O. = GIVE ELEMENT LIST.
user         X_2
             .
             .
             .
             .
computer     SUBDIVISION, 2-D PROCESS INDEX:
user         0        (zero)
computer     **** COMMAND:
```

The values $J_1, J_2, \ldots,$ give the indices of the 2-D domains in which the subdivision will occur.

The values $X_1, X_2, \ldots,$ determine which elements get subdivided.

$$X_k = \begin{cases} \geq 0 \Rightarrow & \text{subdivide all elements in 2-D domain } J_k \text{ with an error indicator } \geq X_k. \\ < 0 \Rightarrow & \text{give an element list and ask for an index of an element to be subdivided. The program will continue to ask for an element until the user returns 0 (zero).} \end{cases}$$

The user may get out of the SUBDIVIDE loop with a 0 (zero).

3.5 DOMAIN Command

The DOMAIN command is used to set up the subdomain on which a solution path is to be performed. The subdomain defined may be any subset of the D_j^2's .

The format is either

```
(i)  DOMAIN
     Ø
   or
(ii) DOMAIN
     N
     k1,...,kN .
```

In case (i) the subdomain defined is the full domain, and in case (ii) the subdomain consists of the N 2-D domains $k_1, k_2, \ldots, k_N$. After the command DOMAIN is returned to the computer, the message

```
SUBSET OF 2-D'S:  NUMBER OF 2-D'S (Ø = ALL)
```

will be printed by the program.

3.6 LONG Command

The LONG command will obtain new solution values for each node in the subdomain specified in the DOMAIN command. If the subdomain defined with DOMAIN is a proper subdomain of the full domain then any points (D_j^1's) which are external to this subdomain, but are internal in the full domain are considered as fixed with the values prescribed by a previous solution path. Also, error indicators of elements which are not in the subdomain are not recalculated.

The format for this command is simply

```
LONG .
```

3.7 SHORT Command

The SHORT command performs a subdivision and a short path solution on each element specified. When an element e_j is specified, it is subdivided into 4 sub-elements, and solution values are obtained for only the new node formed in the middle of e_j . This solution is computed by solving the problem on the 4 sub-elements with linear boundary conditions on the boundary of e_j determined by the 4 cornerpoint solution values previously determined on e_j .

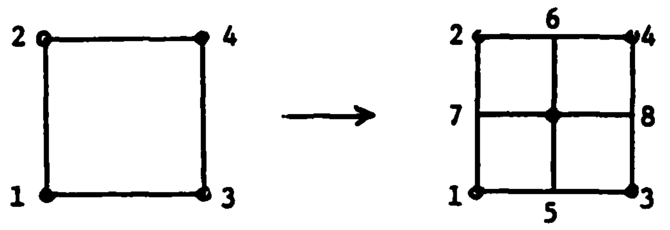


Figure 3.1. Short path subdivision of element e_j.

In the figure we solve for the circled node in the center having the previously obtained solution values at 1, 2, 3, and 4 and the values at 5, 6, 7, and 8 are obtained through interpolation. The new error indicators for the four new elements created are computed in the following way.

Let F denote the father element (to be subdivided) and S_1, S_2, S_3, and S_4 the four sons generated from subdividing F. Let E(F) denote the error indicator for the father element, and P(F) the predicted error indicator for the four sons of the father (see Figure 3.2). Appendix A.3 describes how this predicted error indicator is computed.

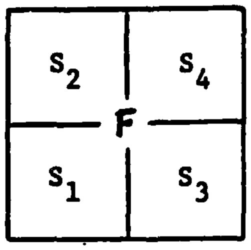


Figure 3.2. A father element subdivided into four sons.

We then compute the error for each son $E(S_i)$ by the formula

$$E(S_i) = \min(\alpha E(F)\ ,\ P(F)),$$ where

$\alpha = .55$ by default, but can be changed by the user with the SHEF command (see Section 3.16), or at the start of the program (see Chapter 5).

The format for the SHORT command is similar to the SUBDIVIDE command.

```
computer    **** COMMAND

user        SHORT

computer    2-D DOMAIN PROCESS INDEX:

user        $J_1$

computer    ELT LVL R  LOCAL COORD ERR.IND PREV.ERR.IND.

            $e_1$  $ℓ_1$  $r_1$  $x_1$    $y_1$  $I_1$       $P_1$
            .   .   .   .     .   .         .
            .   .   .   .     .   .         .
            .   .   .   .     .   .         .
computer    ELEMENT TO BE SUBDIVIDED AND SOLVED

user        $e_1$
.           .
.           .
.           .
computer    ELEMENT TO BE SUBDIVIDED AND SOLVED

user        @EOF

computer    2-D DOMAIN PROCESS INDEX:

user        $J_2$
.           .
.           .
.           .
.           .
.           .
user        @EOF

computer    2-D DOMAIN PROCESS INDEX:

user        @EOF

computer    *** COMMAND
```

3.8 AUTO Command

The AUTO command, short for automatic, is a powerful and useful user command. It performs the subdivision, sets up the subdomain, performs a solution path, and prints a new report.

The format for the AUTO command is

```
AUTO
J
X  (if J < 0)
```

If $J > 0$ each element having an error indicator which is greater then or equal to a computed threshold value is subdivided.

If $J < 0$ another value X must be input and all elements having an error indicator greater than or equal to X will be subdivided.

The value of J determines which type of solution path to take.

$J = \pm 1$ => Obtain a new solution for the full domain.

$J = \pm 2$ => Obtain a new solution for the subdomain composed of those 2-D domains where subdivision occurred.

$J = \pm 3$ => Obtain a new solution for each 2-D domain individually where subdivision occurred.

$J = \pm 4$ => Perform a short path solution for each element subdivided.

Thus, if we wish to subdivide all elements of our full domain and resolve the problem globally we could use the command

```
AUTO
 -1
  0.
```

The same effect could be accomplished with the sequence of commands:

```
SUBDIVIDE
    1
    0.
    2
    0.
    .
    .
    N2
```

```
          0.
        @EOF
        DOMAIN
          0
         LONG
        REPORT
```

When using the AUTO command, it is important to understand how the threshold value is computed. Appendix A.3 contains a detailed explanation. Actually we compute a threshold for each 2-D domain and then use a global threshold being the maximum of all the 2-D domain thresholds. All elements above the global threshold are then subdivided in the AUTO command (if J > 0).

3.9 ERROR Command

The ERROR command recalculates all error indicators using the solutions last obtained and generates a new report. It should be used after repeated short path solutions in order to obtain more accurate error indicators.

3.10 DEBUG Command

The DEBUG command was initially used as a debugging aid, and this option is still available. However, the user may find it more important in its ability to print a list of subdivided elements and/or its control of the output.

The format for the DEBUG command is

computer	**** COMMAND:
user	DEBUG
user	IPR(1) , IPR(2),...,IPR(8)

Where IPR(1),...,IPR(8) deal with the following:

IPR(1) - subdivision element list, short path debug,

IPR(2) - debug error calculation,

IPR(3) - debug matrix assembly,

IPR(4) - debug matrix decomposition,

IPR(5) - debug matrix solution,

IPR(6) - debug back substitution,

IPR(7) - echo input,

IPR(8) - automatic elemental output control.

The values IPR(1)...IPR(8) should be specified as follows:

$$\text{IPR(1)} = \begin{cases} 0 & \text{do nothing,} \\ 1 & \text{print a list of the elements which were subdivided} \\ & \text{before each solution path,} \\ 2 & \text{print out short path solution debugging information} \\ -k & \text{record the subdivision element list on file with FORTRAN} \\ & \text{unit number } k \text{ . File } k \text{ must have been properly} \\ & \text{assigned to the run.} \end{cases}$$

$$\text{IPR(j)} = \left.\begin{cases} 0 & \text{do not print,} \\ 1 & \text{print only summary,} \\ 2 & \text{print all information,} \end{cases}\right\} \quad \text{(For j = 2,...,6.)}$$

$$\text{IPR(7)} = \begin{cases} 0 & \text{do not echo input,} \\ 1 & \text{echo input,} \end{cases}$$

$$\text{IPR(8)} = \begin{cases} 0 & \text{no print, no file write,} \\ 1 & \text{print, no file write,} \\ 2 & \text{no print, file write,} \\ 3 & \text{print, file write.} \end{cases}$$

The parameter IPR(8) causes the listing of the OUTPUT information (see Section 4.6) to be, either printed, or written on FORTRAN unit number 17, after each AUTO command.

The program also asks for the values IPR(1),...,IPR(8) at the beginning of each run, and the DEBUG command offers a way to change the initial values of these parameters.

3.11 OUTPUT Command

The OUTPUT command will have the effect of temporarily changing the parameter IPR(8) and immediately performing the file write and/or print as desired. The format is

```
computer    **** COMMAND:
user         OUTPUT
user         n                  (n=1, 2 or 3)
```

This effect is only temporary. If a new AUTO command is performed the print and/or file write will be done as specified by either the last DEBUG command, or by the initital value given IPR(8) at the start of the program.

3.12 DUMP Command

The DUMP command allows us to save the present data structures, mesh, solution values, etc. in some file so that the problem can be restarted from where we left off at some later time. The format is

```
DUMP
F
```

where F is an integer indicating some fortran unit number. Since units 10-17 are already used in the program, units 18 and up may be used here. These files should all be initially assigned before the program run, and this is described in Chapter 5.

3.13 RESET Command

The RESET command will restart the problem from the time just before the last DUMP was executed. The format is

```
                RESET
                F
```

where F is the same unit number used with the DUMP command. If you were presently running another problem the RESET command will destroy your present run unless you DUMP it onto some other unit number.

3.14 ITERATE Command

The ITERATE command causes the program to take iteration steps, composed of AUTO/J command, automatically, with built in termination. These steps will be either LONG solutions (J=1) or SHORT solutions (J=4).

The format for the ITERATE command is:

computer	**** COMMAND:
user	ITERATE
user	n, β, t, a

where n(>0) is integer valued, β, t, a (>0) are real valued inputs. The value n determines the maximum number of AUTO/J commands to be performed in the sequence.

The value of β is used in the decision strategy to determine the value of J(=1 or 4). The decision value works in the following way. Let EL be the number of elements after the last LONG solution (AUTO/1). Let ES be the estimated number of elements in the mesh when the next refinement occurs. Then if $ES < (1+\beta)\ EL$, the AUTO/4 command (SHORT) is performed, without changing the value of EL. Otherwise, if the estimated increase in elements is β% or more, an AUTO/1 (LONG) is performed which also changes EL to the new number of elements. Exception to this rule is at the last step of the iteration steps, when the program always generates an AUTO/1 (LONG solution).

The value <u>t</u> is the maximum allowed time in seconds. The program assumes that the time needed for an AUTO/1 (LONG) step is linearly dependent on the number of elements E ;

$$T1 = C \cdot E ,$$

where the factor C is derived from a previous AUTO/1 step. If $T1$ and the previously accumulated time of the iteration process is equal or greater than the given t value, then a last step, AUTO/1, is performed to assure a LONG solution before returning to the user's next command.

The value $\underline{a}$ is the required relative accuracy. Again, if $\underline{a}$ has been achieved in the sequence of AUTO/J steps, the program assures that the last step has been a LONG solution before returning to the user's next command.

During the sequence of AUTO/J steps, it may happen that the available data storage area is exhausted due to the increasing number of elements. The required storage areas are also estimated by assuming linear dependency on the number elements, although this may not be very accurate.

Thus, an ITERATE command causes a sequence of AUTO/J steps, where the number of steps is determined by one of the four factors:

(i) n = given maximum number of steps (no message is printed)

(ii) time allowance (t) is exhausted

(iii) accuracy (a) achieved

(iv) storage is exhausted

If one of the last 3 cases occurs, the appropriate message is printed.

3.15 CHANGE Command

The CHANGE command allows the user to make modifications in either the bilinear form, geometry, or function parameters, while keeping the mesh generated from solving the original problem. This command is useful, if for example, the problem of optimal design is of interest.

A problem is solved with some initial geometry using the FEARS program and an optimal mesh is created for this problem. It may be of interest, for example, to determine how the maximum stress is altered by making a perturbance of the geometry. With the CHANGE command, this perturbed problem can be solved with just one solution pass, using the final mesh of the original problem.

Suppose that the final mesh of the original problem was saved on file k by using the DUMP command. After calling the FEARS program (see Chapter 5) you will first be asked to supply the 8 DEBUG parameters IPR(1)-IPR(8). After supplying these values the computer will respond:

computer PROGRAM INPUT DATA IS ON

FILE NUMBER =

to which you should input

user k

k is the file number on which the old mesh was stored. You will then be in the command mode as the computer will respond

```
**** COMMAND: .
```

To use the CHANGE command the user's response is

user CHANGE

The program will then shift back into the input mode and ask for the function parameters, ID Number of the problem, P-NORM for the full domain, and then the geometry and bilinear forms. Here the data should be prepared with the appropriate changes, and input. The program will then immediately obtain a solution for the new problem using the previously saved mesh. A REPORT will be given and the program will return to the command mode.

3.16 SHEF Command

SHEF stands for short path error factor. This command allows the user to prescribe the value of α which is used in the short path error calculations (see Section 3.7). The format for the SHEF command is

```
computer    *** COMMAND
user        SHEF
user        α
```

where $\alpha > 0$ is real. If this command is not used the default value for α is 0.55.

3.17 STOP Command

The STOP command causes the termination of the program execution, thus it should be the last given command:

```
computer    **** COMMAND
user        STOP
```

3.18 ERIT Command

The ERIT command changes the way one of the error terms is calculated for elements adjacent to internal 1-D domains. This error term is based on the difference of derivatives of the solution across the interfacing 1-D domain. Initially in the normal (ON) position, the difference of derivatives is computed form the two adjacent 2-D domains; while in the OFF position, the difference of derivatives are estimated internally in the appropriate 2-D domain. The format of the ERIT Command is

```
computer    **** COMMAND
user        ERIT
```

which causes the change of position, from ON to OFF, or OFF to ON. This command is particularly useful when the problem contains an interface separating two 2-D domains with vastly different material properties.

3.19 MESH Command

The MESH command is similar in purpose to the CHANGE command, i.e., to get solutions for a slightly changed (in geometry or material constants) problem. The use of the MESH command requires that the subdivision element list of the original problem has been saved by the DEBUG command

```
DEBUG
-k,...
```

during the iteration process on file k , where k is a Fortran unit number properly assigned to the run. If this has been done, one can use the MESH command after the initial input (geometry, bilinear forms) to recreate the saved mesh structure for the modified problem by

computer	**** COMMAND
user	MESH
user	k

Note, that the MESH command only recreates the mesh according to the data stored on file k and does not obtain a solution on it. Then it should be followed by a LONG command if the user wishes to obtain solution on the recreated mesh.

3.20 INIT Command

The INIT command reinitializes the program and expects initial input as a new start without asking for the output option.

CHAPTER IV. THE OUTPUT

4.1 Introduction-Global and Local Coordinates

Recall that FEARS takes a unit square and maps it onto each 2-D domain. All computations and mesh refinements are actually computed on this unit square under the appropriate transformation. The coordinates (ξ,η) on the unit square are called the local coordinates and the corresponding values $(x,y)=(x(\xi,\eta),\ y(\xi,\eta))$ are the global coordinates.

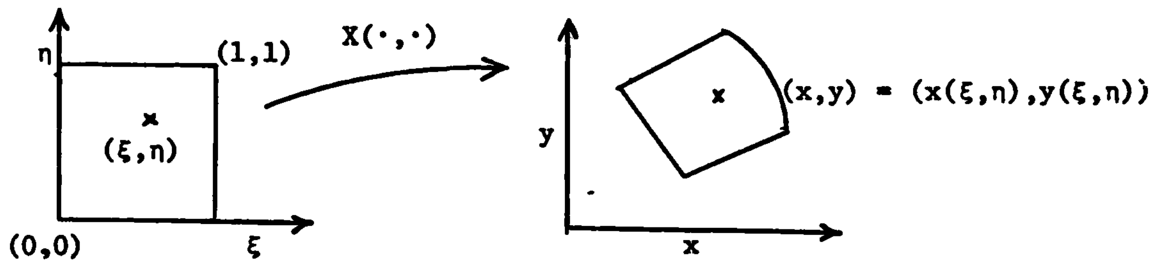


Figure 4.1. Local and Global coordinates for a 2-D domain. Inside this unit square all refinements, element ordering, etc., takes place.

4.2 Numbering the Mesh of a 2-D Domain

In order to help read the printout, a description of how the elements and nodes are indexed is given. This will be a great aid in reconstructing the mesh and hence in locating elements and their neighbors.

The initial subdivision is numbered in the following way:

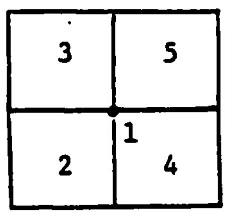


Figure 4.2. Numbering for initial subdivision.

The nodes and elements are numbered together--the index 1 corresponding to the node in the center and the indices 2, 3, 4, and 5 number the 4 elements.

Suppose that element 2 gets subdivided. Then index 2 will correspond to the node formed at the center and the next available indices 6, 7, 8, and 9 will be used to number the 4 new elements created (see Figure 4.3a). Note that the two circled points are not numbered. These points are irregular points without degrees of freedom, whose solution values are obtained through interpolation. Generally, regular points (nodes) are those which lie at the corner of 4 elements and are always numbered. Irregular points lie at the corner of only 2 elements and on the side of some other element and are not numbered.

Suppose next we subdivide element 3. Then index 3 is a point and the next 4 indices 10, 11, 12, 13 number the 4 new elements created (see Figure 4.3b). However, note that the point with local coordinates (.25, .5) is now a regular point and hence gets the next available index 14.

Finally we present the numbering after elements 4 and 5 get subdivided in Figure 4.3c.

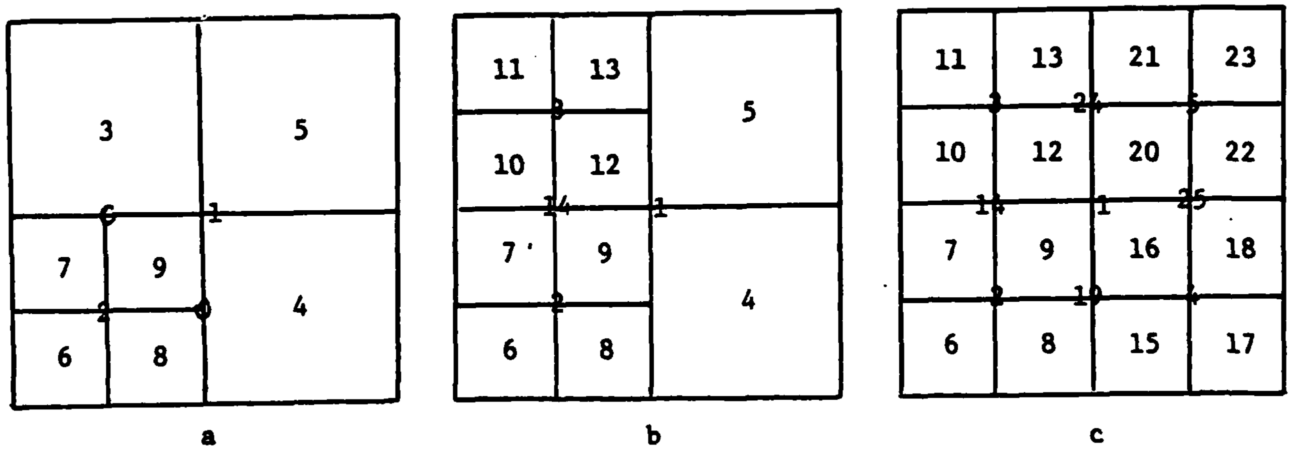


Figure 4.3. Numbering of refined meshes.

A listing of the elements subdivided at each level will be output if the parameter IPR(1) is set to 1 with the DEBUG command (see Section 3.10).

4.3 The PRINT output

The PRINT command will cause data to be printed about the points (D^0_j's), lines (D^1_j's), and 2-D domains (D^2_j's) .

4.3.1. Printing the Points (0-D domains)

The format for the data about the 0-D domains of the geometry is as follows:

<<< 0-D	COORDINATES	SOLUTION	ERROR	BDRY	EXT
1	x_1 y_1	u_1 v_1	eu_1 ev_1	b_1	ex_1
2	x_2 y_2	u_2 v_2	eu_2 ev_2	b_2	ex_2
.	.	.	.	.	.
.	.	.	.	.	.
.	.	.	.	.	.
N_0	$x_{NO}y_{NO}$	$u_{NO}v_{NO}$	$eu_{NO}ev_{NO}$	b_{NO}	ex_{NO}

The information under the heading

0-D	- gives the index number of the point,
COORDINATES	- gives the global (x,y) coordinates of the point as specified by the geometry input,
SOLUTION	- gives the computed solution values of the point,
ERROR	- gives the error between the computed and exact solution if the exact solution is known and supplied in the subroutine ZMPTRU,
BDRY	- gives the boundary conditions of the point as specified by the geometry input, and
EXT	- indicates if the point is internal (0) or external (1) to the full domain.

4.3.2 Printing the lines (1-D domains)

The format of the information about the line with index J is

```
<<<     1-D INDEX:  J  B:  b,  E:  ex,  1/R:  ρ_J,
              FROM    a_J TO  b_J    PTS:  p, R-PTS:  rp


        p POINTS:
PT   R   LOCAL   GLOBAL   COOR   SOLUTION   ERROR
1    r_1  ρ_1      x_1      y_1    u_1  v_1   eu_1  ev_1
.
.
P    r_P  ρ_P      x_P      y_P    u_P  v_P   eu_P  ev_P
```

On the top line the value after

B:	gives the boundary condition of the line as specified by the geometry input,
E:	indicates if the line is internal (0) or external (1)
1/R:	gives the signed reciprocal of the radius as specified by the geometry input,
FROM_TO_:	gives the two indices of the 0-D domains forming the endpoints of the line,
PTS:	gives the number of points on the interior of the line arising from the mesh, and
R-PTS:	gives the number of regular points.

Next, a list of data about the points lying in the interior of the line arising from the mesh is printed. The data under

PT	- gives an index of the point in the order it was formed,
R	- indicates if the point is regular (3) or not (0, 1 or 2),
LOCAL	- gives the local coordinate of the point on the line--the 0-D a_J has a local coordinate 0 and b_J has a local coordinate 1.

GLOBAL COORDINATE - gives the global (x,y) coordinate of the point,

SOLUTION - gives the computed solution values at the point, and

ERROR - gives the error between the computed and exact solutions if the exact solution is known.

4.3.3 The 2-D Domain Printout

The format for the printout of 2-D domain J is

HEAD-ING

<<<2-D INDEX:$\underline{J}$ CORNERS:$\underline{C_1C_2C_3C_4}$ NUMBER OF POINTS:$\underline{NP}$ ELEMENTS:$\underline{NE}$ ENERGY:_

ELT.SIZES:$\underline{S_1S_2S_3S_4S_5S_6S_7S_8}$, SMALLER SIZES:$\underline{S_9}$,MAX.ADJ.RATIO:___

ERROR INDICATORS,TOTAL:___, MAX:___, MEAN:___, THRESHOLD:____

DISTR. BELOW MEAN (_): _ _ _ _ _ _ _ _ _ _

DISTR. ABOVE MEAN (_): _ _ _ _ _ _ _ _ _ _

DISTR. ABOVE PRED (_): _ _ _ _ _ _ _ _ _ _

TIME, ASM& DEC:_____, BCK:_____, THR:_____, TOTAL:_____

STORAGE, MEMORY:__________, AUXILIARY:________

PNT	LOCAL	COORD	GLOBAL	COORD	SOLUTION		ERROR	
P_1	ξ_1	η_1	x_1	y_1	u_1	v_1	eu_1	ev_1
.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.
P_{NP}	ξ_{NP}	η_{NP}	x_{NP}	y_{NP}	u_{NP}	v_{NP}	eu_{NP}	ev_{NP}

ELT	H	R	LOCAL	COORD	ERROR IND.	PREV.ERROR.IND.
E_1	H_1	R_1	ξ_1	η_1	ERR_1	$PERR_1$
.	.	.	.	.	.	.
.	.	.	.	.	.	.
.	.	.	.	.	.	.
E_{NE}	H_{NE}	R_{NE}	ξ_{NE}	η_{NE}	ERR_{NE}	$PERR_{NE}$

In the top line the value(s) after

CORNERS: gives the 4 cornerpoint indices as specified by the geometry input,

NUMBER OF POINTS:	gives the number of regular points (NP) of the mesh in the 2-D domain J,
ELEMENTS:	gives the number of elements (NE) in the mesh, and
ENERGY:	gives the energy of 2-D domain J.

In the next line

ELT. SIZES:	gives the number of elements in the domain having a side of length 2^{-1}, 2^{-2},...,2^{-8} respectively,
SMALLER SIZES:	gives the number of elements smaller than 2^{-8}, and
MAX.ADJ.RATIO:	gives (in power of 2) the maximum of ratios of sizes of two adjacent elements in the 2-D domain.

The next line gives the sum, maximum, mean, and threshold value for the error indicators of the elements in the 2-D domain.

The following three lines give statistical information on the error indicators. The value in parenthesis gives the total, and the following 8 numbers give the number of elements in 8 standard deviations from the mean or maximum predicted value.

The line which gives the TIME breakdown is discussed in Section 4.4.

After the storage requirements are printed, a list of the nodal points of the mesh in the 2-D domain are given. Included are the index of the point as determined from the numbering order described earlier, as well as the point's local coordinates, global coordinates, computed solution values, and the error of the computed solution (if the exact solution is known and defined in ZMPTRU). Only regular points are listed.

After the nodal points are listed, the elements are listed in order of decreasing error indicators. The indices of the elements are determined through the same numbering system as the nodal points and this numbering order was described earlier. Also included are

H - indicating the size of the element (2^{-H+1}),

R - indicates the regularity of the cornerpoints of the element, e.g. R=3 corresponds to all regular cornerpoints (see Section 4.2).

LOCAL COORD - the local coordinates of the center of the element,

ERROR IND. - the error indicator for the element, and

PREV.ERR.IND. - the previous error indicator. See Appendix A.3 for further details on the previous error indicator.

4.4 The TIME: breakdown

FEARS uses substructured solving, obtaining solutions first for those nodes on the 1-D and 0-D domains and then backsolving to obtain solutions for the nodes in the interior of each 2-D domain. In order to better understand the TIME breakdown, write the assembled global stiffness matrix in the form:

$$\begin{bmatrix} A_1 & & & & & B_1 \\ & A_2 & & & & B_2 \\ & & A_3 & & & B_3 \\ & & & \ddots & & \vdots \\ & & & & A_{N2} & B_{N2} \\ B_1^T & B_2^T & & & B_{N2}^T & C \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \\ X_3 \\ \vdots \\ X_{N2} \\ X_{BD} \end{bmatrix} = \begin{bmatrix} Y_1 \\ Y_2 \\ Y_3 \\ \vdots \\ Y_{N2} \\ Y_{BD} \end{bmatrix}$$

where X_i correspond to the unknowns in the interior of 2-D domain i, and X_{BD} the unknowns for the 0-D and 1-D domains.

With the 2-D domain print command, the line starting with TIME lists

ASM & DEC:	the time for the assembly and LU decomposition of A_i
BCK:	the time for the backsubstitution for the unknowns in 2-D domain i.
THR:	the time for computing the error indicators, and
TOTAL:	the sum of the above three times.

The REPORT print lists

EXECUTION TIME:

SUBDIVISION: the time of subdivision for the full domain,

2-D MATRIX SOLUTION: the sum of the ASM & DEC times described above for each 2-D domain,

BDRY MATRIX SOLUTION: the time for solving $C \cdot X_{BD} = Y_{BD}$ where C , Y_{BD} are C , Y_{BD} modified by the 2-D partial decompositions.

2-D ERROR CALCULATION: The sum of the THR times.

At the end of an AUTO command the ***AUTO TIME gives the sum of the 4 execution times plus overhead.

4.5 The REPORT printout

The REPORT has the format

```
***** FULL DOMAIN *****

NUMBER OF POINTS: NP     NUMBER OF ELEMENTS: NE

ENERGY NORM: NM     ENERGY: ENG

ERROR ESTIMATOR: EST     RELATIVE ERROR: REL

MAX.ERROR INDICATOR: MAX     BY 2-D INDEX ___     THRESHOLD: TH

APPROXIMATE NUMBER OF ELEMENTS TO BE SUBDIVIDED:

2-D      NO. OF ELEMENTS

1             S1

2             S2
.             .
.             .
N2            SN2

TOTAL         T

STORAGE SIZES;    MAX,CORE=_____, TOTAL=_____, NO RECORDS=_____

BDRY MATRIX = _____

EXECUTION TIME:, SUBDIVISION:_____,   2-D MATRIX SOLUTION:_____
```

BRDY MATRIX SOLUTION:_____,   2-D ERROR CALCULATION:_____

Here

NUMBER OF POINTS: gives the number of nodes in the 0, 1 and 2-D domains have at least one degree of freedom,

NUMBER OF ELEMENTS: gives the total number of elements in the full domain,

ENERGY NORM: $NM = \sqrt{ENG}$ where

ENERGY: ENG

ERROR ESTIMATOR: gives the error estimate for the full domain,

RELATIVE ERROR: REL = NM/EST.

The next line gives the maximum error indicator, the 2-D domain it is in, and the threshold value for automatic refinement.

Next is a list of the number of elements in each 2-D domain which will be subdivided if the threshold value is used for subdivision.

Storage size information is then given and finally a breakdown of the times as discussed in the previous section.

4.6 The OUTPUT printout

In this section we describe the data either printed or file written, with either the OUTPUT command or by previously setting IPR(8) in the DEBUG command or at the program start. The data printed will be a list with the headings.

```
2-D ELT H LOCAL COORD  GLOBAL COORD  ERR.IND  OUTP(1) . . . OUTP(5)
```

The column headed with

2-D - gives the index of the 2-D domain that the element is in,

ELT - gives the index of the element,

H - indicates the element size ($=2^{-H+1}$),

LOCAL COORD GLOBAL COORD - gives the local and global coordinates at the center of the element, ERRIND - gives the error indicator of the element,

$$\left.\begin{array}{l}\text{OUTP(1)}\\ \text{OUTP(2)}\\ \text{OUTP(3)}\\ \text{OUTP(4)}\\ \text{OUTP(5)}\end{array}\right\}$$ will give the stresses σ_{xx}, σ_{yy}, σ_{xy}, and solutions u_1, u_2 at center of the element if S was given for the elasticity problem.

These last 5 values are determined by the output matrix S given in the bilinear matrix input. The 5 values printed are determined by

$$\underset{5\times 6}{S}\ \underset{6\times 1}{\begin{bmatrix}\frac{\partial U}{\partial Z}\\ U\end{bmatrix}} = \underset{5\times 1}{\begin{bmatrix}\text{OUTP(1)}\\ \cdots\\ \text{OUTP(5)}\end{bmatrix}}$$

The appropriate S for elasticity problems is given in the Appendix.

4.7 DUMP-File output

A sequential binary DUMP-file is generated by the DUMP command. This file can be used as input for FEARS (see Sections 3.13 and 3.20), or as input for various postprocessors. The order of the binary records on the file are as follows:

1. Summary record of the problem (16 words)
2. Last long path history record (40 words)
3. Function parameter record (N+1 words where N is the number of parameters)
4. 0-D summary/records (8 words/records, N_0 records)
5. 1-D summary records (18 words/records, N_1 records)
6. 2-D Summary records (25 words/records, N_2 records)
7. For each 1-D domain with non-fixed boundary condition:
   7.1 1-D data record (8k words, where k is the number of points on the 1-D)

7.2 If γ, ε matrices were defined for the 1-D then γ, ε data record (13 words)

8. For each 2-D domain

8.1 2-D data record (4k words, where k is the number of points and elements in the 2-D)

8.2 Bilinear form data record (125 words)

Details on the fields of records can be found in the Program Documentation.

CHAPTER V. STARTING THE FEARS PROGRAM

This chapter describes the UNIVAC 1100 EXEC control cards necessary to run the FEARS program. There are two elements and one file which are of concern:

Absolute element:	CKM*FA.A
Relocatable file:	CKM*RE.
Map element:	CKM*SE.MAP

The names of the above elements (file) may change by the actual installation. The absolute element contains the absolute program with dummy, zero valued, function routine for E(X,Y),D(S,Y) which can be run if that routine is satisfactory by

```
@XQT,F     CKM*FA.A
```

Otherwise, *section* 5.2 describes how to set up an absolute program with the help of the Relocatabl: file and Map element to incorporate the user's supplied function routine.

Section 5.1 explains the necessary file assignments to be included prior to the run of the program. Section 5.3 describes the necessary initial inputs for FEARS before the geometry, bilinear matrices and command inputs are given as described in chapters 1, 2 and 3, respectively.

5.1 File Assignment

FEARS uses six temporary files (11 to 16) with all runs which should be assigned by

```
@ASG,T     11.
@ASG,T     12.
@ASG,T     13.
@ASG,T     14.
@ASG,T     15.,///512
@ASG,T     16.
```

Four other files may be used depending on the actual run and commands given for the run. These files should be catalogued files with user's given names. These names should be associated with the fortran unit number by @USE:

```
@ASG, A     file name.
@USE        n., file name.
```

where n is the FORTRAN unit number. The four files are as follows:

(i) Output file - If the OUTPUT command is used, see 3.11, or IPR(8) was given as 2 or 3 in either the DEBUG command (see 3.10) or initially (see 5.3), the file should be assigned with n=17 . At the termination of the run, the file will contain the stresses as described 4.6.

(ii) Mesh file - This file is generated as output file if IPR(1) = -n as given for the DEBUG command (3.10), it is used as input file for MESH command (3.19) It is recommended to use n = 18.

(iii) Dump file - This file is generated by the DUMP command (3.12). Since present and future postprocessors use this file as input file, most of the use of FEARS is anticipated to use this file. Recommended n = 20.

(iv) Reset file - This input file is assumed to be generated by a DUMP command in previous run of FEARS. The file is used either initially (5.3) or by the RESET command (3.13). Recommended n = 21.

5.2 Preparation for Execution

When the user has to incorporate his/her function routine (see 2.7), then a new absolute program of FEARS must be generated. The recommended steps to be taken are as follows:

(i) Write the function routine according to the speficication given in 2.7. Compile the symbolics using the ASCII FTN compiler to generate the relocatable element. For further on, it is assumed that F.FXY is element name of this relocatable element.

(ii) Using the editor,

```
                @ED   CKM*SE.MAP,M
```

change the Map element into M

```
user            *N
computer        @MAP,IN   ,CKM*FA.A
user            *C   /CKM*FA/F/
computer        @MAP,IN      ,F.A
user            *N 2
computer        IN   CKM*RE.FUNC
user            *C   /CKM*RE.FUNC/F.FXY/
computer        IN   F.FXY
user            *@
computer        editor signs off
user            @ADD   M
computer        MAP .... .
                END MAP.   ERROR = 0 ...
```

The above sequence places the new absolute element F.A in the users file.

(iii) After the appropriate assign statements, see 5.1, the run can be initiated by

```
        @XQT,F   F.A
```

Note:

The geometry and matrix inputs can be prepared separately in file elements, e.g. in INP.G1 and INP.MX1 , respectively. In this case, the user simply applies the UNIVAC commands

```
@ADD   INP.G1
@ADD   INP.MX1
```

in the above input stream as input for the geometry and bilinear, error and output matrices. This technique is especially useful when FEARS is used interactively.

5.3 Execution of FEARS

After the program execution has been initiated by @XQT,F, the computer will respond

```
***** F E A R S *****
2-D FINITE ELEMENT PROGRAM
UNIVERSITY OF MARYLAND. 1981.

MAXIMUM ALLOWED SIZES
NUMBER OF 0-D, 1-D, 2-D: 34 49 16
NUMBER OF POINTS ON A 1-D: 31
NUMBER OF POINTS AND ELEMENTS IN ONE 2-D: 482
MATRIX STACK SIZE FOR ONE 2-D:   8800
BOUNDARY MATRIX SIZE:   14000

SHORT PATH ERROR-FACTOR =    .55000
PRINT CONTROL INTEGERS (8):
IPR(1) - PRINTS DURING SHORT PATH, SUBDIV.
    NEG. K - RECORD SUBD. ELEMENTS ON FILE K
IPR(2) - PRINTS DURING ERROR CALCULATION
IPR(3) - PRINTS DURING ASSEMBLY
IPR(4) - PRINTS DURING DECOMPOSITION
IPR(5) - PRINTS DURING MATRIX SOLUTION
IPR(6) - PRINTS DURING BACKSUBSTITUTION
IPR(7) - REPRINTS INPUT
IPR(8) - AUTOMATIC ELEMENTAL OUTPUT CONTROL
     IPR(J),J=1,...,8:
```

The user now should input the 8 integer values IPR(1) - IPR(8) as discussed in the DEBUG command, e.g.

```
0,0,0,0,0,0,1,0
```

Next the computer will respond

```
PROBLEM INPUT DATA IS ON

FILE NUMBER :
```

The user should input zero:

```
0
```

if this is a new problem, or the Reset file number, e.g.

```
21
```

on which all the data at some stage of a problem was saved by the DUMP command. In this case, the computer will respond

```
          **** COMMAND
```

and the program will be in the command mode.

Otherwise (in case of a new problem), the computer will respond

```
          FUNCTION PARAMETERS:  N,P1,...,PN(N>0):
```

and the user should input an integer N>0 , and N parameters for use in the user defined functions routine (see section 2.7). If no parameters are required, the dummy values 1,0,may be input.

The computer will then respond

```
          ID - NUMBER OF THE PROBLEM:
```

and any integer response from 1 to 999999 will suffice.

The next computer response is

```
          P-NORM FOR THE FULL DOMAIN:
```

The P requested here is the same one as in (2.1). If an L_{2p} norm is of interest in measuring the errors then the value P>.5 should be input. An L_∞ norm will be used if 0. is input. Usually P=1 which corresponds to the standard L_2 energy norm.

The computer will then respond

```
     PROBLEM ID: ________
     DATE:       ________
     P-NORM:     ________
     GEOMETRY:   _______
```

At this point the geometry should be input in the format described in chapter 1, i.e., starting with number of 0-D domains and ending with the last 2-D domain description line.

If no errors are detected the computer will respond

```
     GEOMETRY ACCEPTED
     BILINEAR, ERROR, AND OUTPUT MATRICES:
```

At this point, these matrices should be input in the format described in Chapter 2.

The computer should then respond

```
MATRICES ACCEPTED
INITIAL SUBDIVISION, 1 OR 2:
```

The value 1 will cause each 2-D domain to have 4 elements and the value 2 will cause each 2-D domain to have 16 elements for the initial mesh.

Finally the computer will respond

```
INITIAL SUBDIVISION ( ) FOR 1-D PERFORMED
INITIAL SUBDIVISION ( ) FOR 2-D PERFORMED.
```

The program will then obtain an initial full solution path on the entire domain, print a REPORT and respond

```
**** COMMAND
```

signifying that it is ready for user commands.

Once in command mode, further actions are governed by the user's commands as described in Chapter 3.

APPENDIX A

A.1 Sample Geometry

In this appendix sample domains and their FEARS geometry input are presented.

Example 1: Quarter Ring - 1 - subdomain

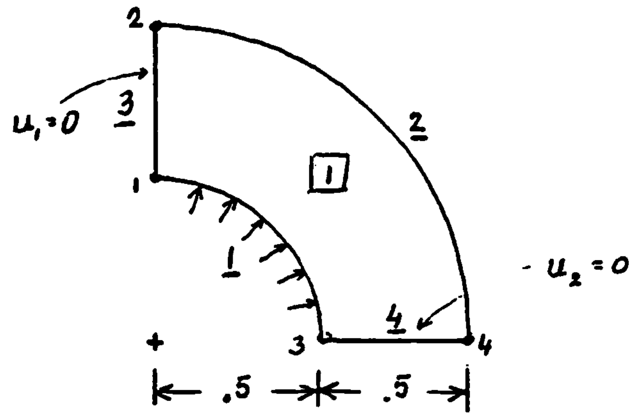


Figure A.1. Quarter Ring

The geometry input for Figure A.1 is:

```
 4
1,0.,.5,1,0.,0.
2,0.,1.,1,0.,0.
3,.5,0.,2.0.,0.
4,1.,.0.,2,0.,0.
 4
1,1,3,0,2.
2,2,4,0,1.
3,1,2,1,0.
4,3,4,2,0.
 1
1,1,2,3,4
```

Note that the boundary condition on line index $\underline{1}$ is specified as free. The force present (indicated by arrows) is specified in the bilinear matrix input.

Example 2: Unit Square - 4 - sub domains

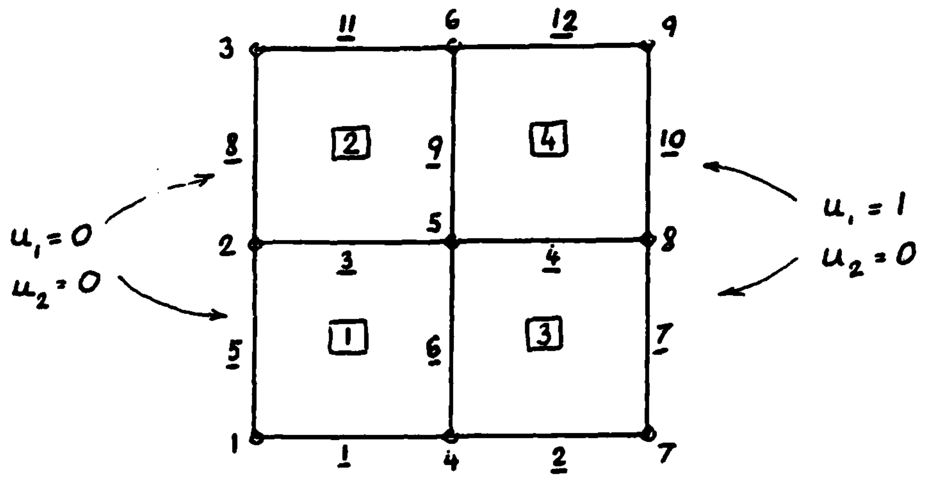


Figure A.2. Unit Square

The geometry input for Figure A.2 is:

```
9
1,0.,0.,3,0.,0.
2,0.,.5,3,0.,0.
3,0.,1.,3,0.,0.
4,.5,0.,0,0.,0.
5,.5,.5,0,0.,0.
6,.5,1.,0,0.,0.
7,1.,0.,3,1.,0.
8,1.,.5,3;1.,0.
9,1.,1.,3,1.,0.
12
1,1,4,0,0.
2,4,7,0,0.
3,2,5,0,0.
4,5,8,0,0.
5,1,2,3,0.
6,4,5,0,0.
7,7,8,3,0.
8,2,3,3,0.
9,5,6,0,0.
10,8,9,3,0.
11,3,6,0,0.
12,6,9,0,0.
4
1,1,2,4,5
2,2,3,5,6
3,4,5,7,8
4,5,6,8,9
```

Example 3: Disk

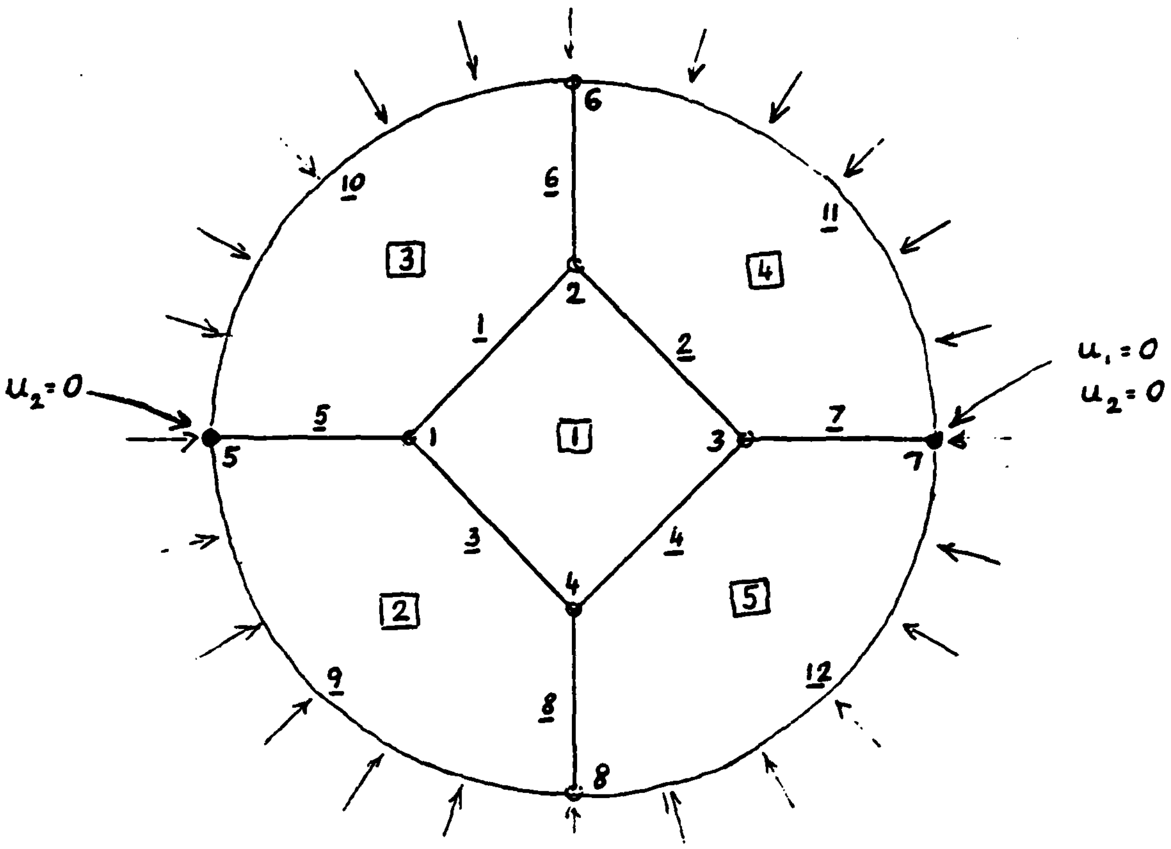


Figure A.3. Disk with hydrostatic force.

```
8
1,-.4,0. ,0,0.,0.
2,0.,.4,0,0.,0.
3,.4,0.,0,0.,0.
4,0.,-.4,0,0.,0.
5.-1.,0.,2,0.,0.
6,0.,1.,0,0.,0.
7,1.,0.,3,0.,0.
8,0.,-1.,0,0.,0.
  12
1,1,2,0,0.
2,2,3,0,0.
3,1,4,0,0.
4,3,4,0,0.
```

```
 5,5,1,0,0.
 6,6,2,0,0.
 7,3,7,0,0.
 8,4,8,0,0.
 9,8,5,0,1.
10,5,6,0,1.
11,6,7,0,1.
12,7,8,0,1.
  5
1,1,2,3,4
2,5,1,8,4
3,5,6,1,2
4,2,6,3,7
5,8,4,7,3
```

Notice the boundary conditions at points 5 and 7. For this problem, additional boundary conditions are necessary to ensure uniqueness of a solution by eliminating rotations and translations of the solution. Point 7 was picked arbitrarily as the stationary point ($u_1=u_2=0$). To prevent rotations the vertical displacement, u_2, was set to zero at point 5. It would be equivalent to fixing the vertical displacement at either point 1 or point 3 as well.

Example 4: Cracked Square

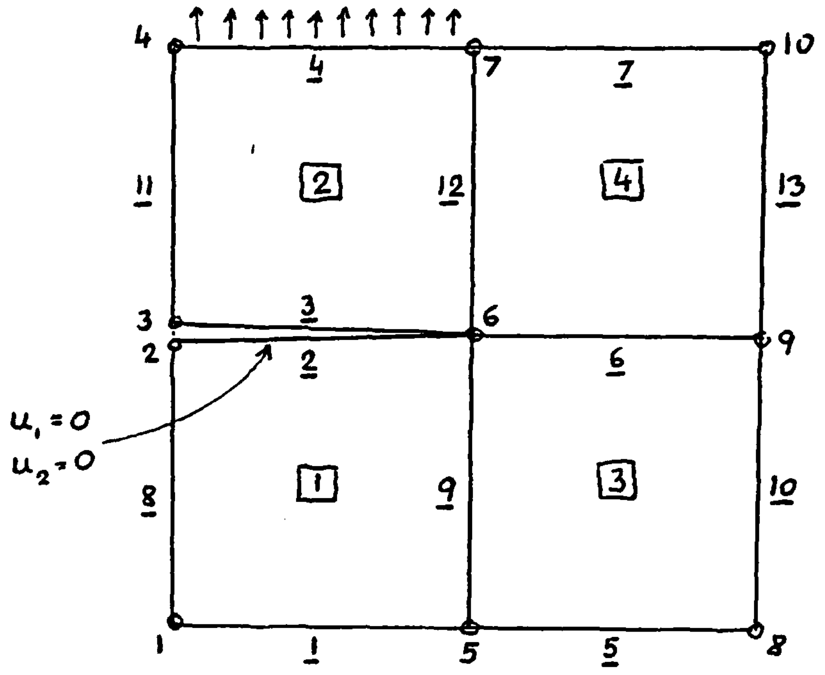


Figure A.4. Cracked Square.

```
  10                            3,3,6,0,0.
 1,0.,0.,0,0.,0.                4,4,7,0,0.
 2,0.,.5,3,0.,0.                5,5,8,0,0.
 3,0.,.5,0,0.,0.                6,6,9,0,0.
 4,0.,1.,0,0.,0.                7,7,10,0,0.
 5,.5,0.,0,0.,0.                8,1,2,0,0.
 6,.5,.5,3,0.,0.                9,5,6,0,0.
 7,.5,1.,0,0.,0.               10,8,9,0,0.
 8,1.,0.,0,0.,0.               11,3,4,0,0.
 9,1.,.5,0,0.,0.               12,6,7,0,0.
10,1.,1.,0,0.,0.               13,9,10,0,0.
  13                              4           or      -4
1,1,5,0,0.                     1,1,2,5,6             1,1,2,8,9
2,2,6,3,0.                     2,3,4,6,7             2,3,4,11,12
                               3,5,6,8,9             3,5,6,9,10
                               4,6,7,9,10            4,6,7,12,13
```

Notice that points 2 and 3 have the same coordinates and thus lines 2 and 3 lie on top of each other. This represents the two edges of a crack.

A.2 Bilinear and Error Matrices

A.2.1. The Matrix A Arising From Elasticity

The principle of virtual work for the elasticity equations yields an integral of the form $\int_D (\delta\underset{\sim}{\varepsilon})^T C\underset{\sim}{\varepsilon}$, where

$$\underset{\sim}{\varepsilon} = \begin{bmatrix} \varepsilon_{xx} \\ \varepsilon_{yy} \\ \varepsilon_{xy} \end{bmatrix} = \begin{bmatrix} \frac{\partial u_1}{\partial x} \\ \frac{\partial u_2}{\partial y} \\ \frac{\partial u_1}{\partial y} + \frac{\partial u_2}{\partial x} \end{bmatrix} , \qquad \text{and} \quad C$$

is the 3 x 3 stress-strain matirx such that

$$\underset{\sim}{\sigma} = \begin{bmatrix} \sigma_{xx} \\ \sigma_{yy} \\ \sigma_{xy} \end{bmatrix} = \underset{\sim}{\varepsilon} \ .$$

For the plane strain assumption the matrix C is given by

$$C = \frac{E}{(1+\nu)(1-\nu)} \begin{bmatrix} 1-\nu & \nu & 0 \\ \nu & 1-\nu & 0 \\ 0 & 0 & \frac{1-2\nu}{2} \end{bmatrix} ,$$

and for the plane stress assumption

$$C = \frac{E}{1-\nu^2} \begin{bmatrix} 1 & \nu & 0 \\ \nu & 1 & 0 \\ 0 & 0 & \frac{1-\nu}{2} \end{bmatrix} . ,$$

where E is Young's molulus and ν is the Poisson ratio.

The corresponding integrand in the FEARS formulation is

$$\left[\frac{\partial V}{\partial Z}\right]^T A \frac{\partial U}{\partial Z} \quad .$$

Since $\frac{\partial U}{\partial Z} = \begin{bmatrix} \frac{\partial U_1}{\partial X} \\ \frac{\partial U_2}{\partial X} \\ \frac{\partial U_1}{\partial Y} \\ \frac{\partial U_2}{\partial Y} \end{bmatrix}$, we may

write $\underset{\sim}{\varepsilon} = D\frac{\partial U}{\partial Z}$, where

$$D = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix} . \qquad (A.1)$$

Also, $\delta\underset{\sim}{\varepsilon} = D\frac{\partial V}{\partial Z}$. Therefore,

$(\delta\underset{\sim}{\varepsilon})^T C \, \underset{\sim}{\varepsilon} = \left[\frac{\partial V}{\partial Z}\right]^T D^T \, C \, D\left[\frac{\partial U}{\partial Z}\right]$, from which we have the relationship

$$A = D^T \, C \, D \; .$$

For the plane strain assumption

$$A = \frac{E}{(1+\nu)(1-2\nu)} \begin{bmatrix} 1-\nu & 0 & 0 & \nu \\ 0 & \frac{1-2\nu}{2} & \frac{1-2\nu}{2} & 0 \\ 0 & \frac{1-2\nu}{2} & \frac{1-2\nu}{2} & 0 \\ \nu & 0 & 0 & 1-\nu \end{bmatrix} ,$$

and for the plane stress assumption

$$A = \frac{E}{1-\nu^2} \begin{bmatrix} 1 & 0 & 0 & \nu \\ 0 & \frac{1-2\nu}{2} & \frac{1-2\nu}{2} & 0 \\ 0 & \frac{1-\nu}{2} & \frac{1-\nu}{2} & 0 \\ \nu & 0 & 0 & 1 \end{bmatrix} .$$

A.2.2 The Error Matrix A_E.

For elasticity problems the matrices $(C)_j$ described in Section 2.3 are zero, and so, from (2.1) the error is approximated in the norm

$$|||U|||_{2p} = \left[\sum_{j=1}^{N2} \int_{\mathcal{D}_j^2} \left\{ \left[\frac{\partial U}{\partial Z}\right]^T (A_E)_j \left[\frac{\partial U}{\partial Z}\right] \right\}^{p_j} \right]^{\frac{1}{2}p_j} .$$

To obtain error estimates for the energy norm, simply take $(A_E)_j = (A)_j$.

Suppose, however, that our interest is concentrated on the error in one of the stresses, say σ_{xx} . Since $\sigma_{xx} = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \underset{\sim}{\sigma}$,

$$\sigma_{xx}^2 = \underset{\sim}{\sigma}^T \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \underset{\sim}{\sigma} = \underset{\sim}{\sigma}^T \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \underset{\sim}{\sigma}$$

$$= \underset{\sim}{\varepsilon}^T C^T \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} C \underset{\sim}{\varepsilon}$$

$$= \left[\frac{\partial U}{\partial Z}\right]^T D^T C^T \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} C D \left[\frac{\partial U}{\partial Z}\right] ,$$

where C is the 3 x 3 stress strain matrix, and D is defined by (A.1).

Let $Q = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$. Then, if the error σ_{xx} is our main concern, we should take

$$A_E = D^T C^T Q C D .$$

For example, using the plane strain assumption with an error in σ_{xx},

$$A_E = \left[\frac{E}{(1+\nu)(1-\nu)}\right]^2 \begin{bmatrix} (1-\nu)^2 & 0 & 0 & \nu(1-\nu) \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ \nu(1-\nu) & 0 & 0 & \nu^2 \end{bmatrix} .$$

For the error in σ_{yy} take Q in (A.2) to be

$$Q = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \text{ and}$$

for σ_{xy}, take Q to be

$$Q = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

A.2.3 The Output Matrix S

For the OUTPUT printout (see Sections 3.11 and 4.6) the vector printed is obtained from the multiplication

$S \begin{bmatrix} \frac{\partial U}{\partial Z} \\ U \end{bmatrix}$ where $\frac{\partial U}{\partial Z}$ and U are

evaluated at the center of each element. In order for a printout of σ_{xx}, σ_{yy}, σ_{xy}, u_1, u_2 the matrix S should be

$$S = \begin{bmatrix} \frac{E}{1-\nu^2} \begin{bmatrix} 1 & 0 & 0 & \nu \\ \nu & 0 & 0 & 1 \\ 0 & \frac{1-\nu}{2} & \frac{1-\nu}{2} & 0 \end{bmatrix} & \begin{matrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{matrix} \\ \begin{matrix} \quad\quad\quad 0 & 0 & 0 & 0 \\ \quad\quad\quad 0 & 0 & 0 & 0 \end{matrix} & \begin{matrix} 1 & 0 \\ 0 & 1 \end{matrix} \end{bmatrix}$$

under the plane stress assumption, and

$$S = \begin{bmatrix} \frac{E}{(1+\nu)(1-2\nu)} \begin{bmatrix} 1-\nu & 0 & 0 & \nu \\ \nu & 0 & 0 & 1-\nu \\ 0 & \frac{1-2\nu}{2} & \frac{1-2\nu}{2} & 0 \end{bmatrix} & \begin{matrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{matrix} \\ \begin{matrix} \quad\quad\quad 0 & 0 & 0 & 0 \\ \quad\quad\quad 0 & 0 & 0 & 0 \end{matrix} & \begin{matrix} 1 & 0 \\ 0 & 1 \end{matrix} \end{bmatrix}$$

under the plane strain assumption.

A.2.4 Sample Inputs

Example 1: A-Matrix: Laplacian

Error norm: H^1 seminorm.

Packages: One package for one 2-D domain.

Output Matrix: $\frac{\partial u_1}{\partial x}, \frac{\partial u_2}{\partial y}, .5\frac{\partial u_2}{\partial x} + .5\frac{\partial u_1}{\partial y}, u_1, u_2$.

Line Integrations: None.

```
  1
1, 1, 1
1, 0, 0, 0, 0
1., 0.,0.,1., 0., 0., 0., 0., 0., 0., 0., 0., 1., 0., 0., 1.
1., 0., 0., 1., 0., 0., 0., 0., 0., 0., 0., 0.,1., 0., 0., 1.
1., 1., 1., 1.
1., 0., 0., 0., 0., 0.
0., 0., 0., 1., 0., 0.
0., .5, .5, 0., 0., 0.
0., 0., 0., 0., 1., 0.
0., 0., 0., 0., 0., 1.
  0
```

Example 2: A Matrix: Plane Strain Elasticity

Error Norm: L_2 Energy

Packages: Two packages for 4 2-D domains.

Package 1: 2-D domains 1 and 2, $\nu = 0.0$, $E = 1.0$

Package 2: 2-D Domains 3 and 4, $\nu = 0.3$, $E = 1.0$.

Output Matrix: σ_{xx}, σ_{yy}, σ_{xy}, u_2, u_2.

Line Integrations: Local normal force on line 4.

```
  2
1, 2, 1, 2, 0., 1, 0, 0, 0, 0
1., 0., 0., .5, 0., 0., .5, 0., 0., .5, 0., 0., .5, 0., 0., 1.
1., 0., 0., .5, 0., 0., .5, 0., 0., .5, 0., 0., .5, 0., 0., 1.
1., 1., 1., 1.
1., 0., 0., 0., 0., 0.
0., 0., 0., 1., 0., 0.
0., .5, .5, 0., 0., 0.
0., 0., 0., 0., 1., 0.
0., 0., 0., 0., 0., 1.
2, 2, 3, 4
1, 0, 0, 0, 0
1.3461538, 0., 0., .38461538,0.,.57692308, .38461538, 0.
   0., .38461538, .57692308, 0., .38461538, 0., 0., 1.3461538
1.3461538, 0., 0., .38461538, 0., .57692308, .38461538,0.
   0., .38461538, .57692308, 0., .38461538, 0., 0., 1.3461538
   1., 1., 1., 1.
1.3461538, 0., 0., .57692308, 0., 0.
.57692308, 0., 0., 1.3461538, 0., 0.
  0., .38461538, .38461538, 0., 0., 0.
  0., 0., 0., 0., 1., 0.
```

```
0., 0., 0., 0., 0., 1.
  1
1, 1, 4
0, -1
0., 0., 0., 0.
0., 1., 0., 1.
```

Example 3: A-Matrix: Plane strain

Error Norm: L_∞ for σ_{yy}.

Packages: One package for one 2-D domain, $\nu = 0.0$, $E = 1.0$.

Output Matrix: σ_{xx}, σ_{yy}, σ_{xy}, u_1,u_2

Line Integration: None

```
  1
1, 1, 1
1, 0, 0, 0, 0
1., 0., 0., .5, 0., 0., .5, 0., 0., .5, 0., 0., .5, 0., 0., 1.
0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 1.
0., 1., 1., 1.
1., 0., 0., 0., 0., 0.
0., 0., 0., 1., 0., 0.
0., .5, .5, 0., 0., 0.
0., 0., 0., 0., 1., 0.
0., 0., 0., 0., 0., 1.
  0
```

A.3 Computation of the Threshold Value

When using the AUTO command or the ITERATE command (the ITERATE command causes a sequence of AUTO commands to be performed), it is important to understand how the threshold value is computed. Recall that with the AUTO command, all elements having error indicators above a certain threshold are subdivided. Because of the remarks made in Section 3.1, we seek a mesh in which all error indicators are nearly equal.

The most naive way to equalize the indicators would be to simply subdivide the element with the largest error indicator before each solution path. Suppose that the elements in the initial mesh are labeled

$$1,\ 2,\ldots,N$$

in order of decreasing error indicators. Following our "naive" strategy, we sould subdivide element 1 into four subelements--1.1, 1.2, 1.3, and 1.4. After resolving and recomputing the error indicator our new list might be

$$2,\ 3,\ldots J_1,\ 1.1,\ J_1+1,\ldots,J_2,\ 1.2,\ J_2+1,\ldots,N\ .$$

Next, element 2 gets subdivided into 2.1, 2.2, 2.3, 2.4, etc. Suppose element 1.1 has the largest indicator of the first $4J_1$ subdivided elements. That is,

$$E(1.1) \geq E(i.j) \quad \text{for } i = 2,\ldots,J_1, \text{ and } j = 1,\ldots,4\ .$$

Then our strategy would have subdivided the first J_1 elements one at a time, obtaining a new solution between each subdivision. Clearly, it would be more efficient to subdivide all J_1 elements at once before obtaining a new solution. The cufoff at J_1 could easily be determined if we know what the maximum error indicator would be for the next level of subdivided elements (in this case $E(1.1)$). The following procedure is used to approximate this cutoff value.

Suppose that a father element F is to be subdivided into its four sons--F.1, F.2, F.3, and F.4. (see Figure A.5)

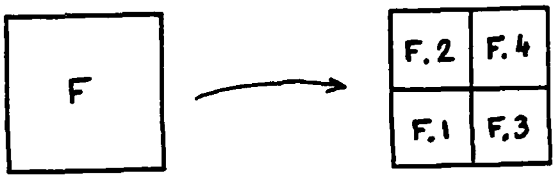


Figure A.5.

The error indicator for the father element, E(F) is then saved as the previous error indicators for the four sons. It is then assumed that the errors arising from subdividing the sons F.1, F.2, F.3, and F.4 will decrease by the same ratio as from the previous subdivision. That is, we use the recipe: $\text{PREDICTION} = \frac{\text{PRESENT}}{\text{PREVIOUS}} \times \text{PRESENT}$. Thus, the predicted error indicator for each of the four sons is computed by the formula

$$P(F.i) = E(F.i)^2/E(F) \ , \quad \text{for} \quad i = 1, 2, 3, 4 \ .$$

In order to ensure that the predicted error is smaller then the present error we add the condition--

$$P(F.i) = \min\left(P(F.i), \begin{cases} (.9)^{2p}E(F.i) & \text{if} \quad p \geq .5 \\ (.9)E(F.i) & \text{if } p = 0 \end{cases}\right) \ ,$$

where p is the p-norm for the 2-D domain as input in the $(N_c)_j$ line of bilinear and error matrix input.

If an element is not a son of some father element, that is, the element also belongs to the initial mesh, then no previous error indicator is available. In this case, we make a prediction for the error indicators upon subdivision of this element F , by the formula

$$P(F) = \begin{cases} (\sqrt{2}/2)^{2p}E(F) & \text{if} \quad p \geq .5 \\ (\sqrt{2}/2)E(F) & \text{if} \quad p = 0 \ . \end{cases}$$

The threshold T is then calculated by the formula

$$T = \max_{\tau} P(\tau) ,$$

as τ ranges over all elements in the mesh.

APPENDIX B

FEARS Element Postprocessor

The purpose of the Postprocessor is to allow flexible computations of various functionals on the approximate solution. The functionals can have various structures and can be used for the effective computation of the stress intensity factors in fracture mechanics etc. At the present the functionals are on the level of 2-D domains, but will be extended in the future to deal with functionals on the 1-D domains also.

A FEARS Postprocessor frame has been set up in the file CKM*ELTOUT. Both symbolic and relocatable elements for an Element Postprocessor has been established in the file. To produce an executable absolute element, the user must write some subroutines and combine it with the above file.

The FEARS Element Postprocessor reads the RESET file generated by the FEARS program using the DUMP command. The Postprocessor requires a user's written subroutine (with 5 entries). The purpose of the Postprocessor frame is to relieve the user of the cumbersome task of setting up the data structure from the RESET file and setting the individual element informations needed for his/her calculations. The frame allows it to process all elements, or only those which are in certain 2-D domains designated as active 2-D domains by the user.

Frame algorithm:

The main program of the Postprocessor frame has the following algorithm:

1. Read the RESET file and set up internal data structure.
2. Clear activity tags MD1TMP(j) and MD2TMP(k) of all 1-D and 2-D domains, repectively.
3. Call ELTO(MSARY) calls user's initialization routine.
4. DO I=1 to MSP2 (number of 2-D).
5. IF MD2TMP(I)=0 THEN GOTO Next I

```
6.  Set up core sto~age for 2-D index I.
7.  CALL ELT1(MD2ARY).
        call user's 2-D initialization routine
8.  DO J=1,M2SXE (number of elements in the 2-D)
9.  CALL ELTX(K,N,X,C)
        call user's processing routine
10. Next J
11. CALL ELT1E(MD2ARY)
        call user's 2-D summary routine
12. Next I
13. CALL ELTO(MSARY)
        call user's final summary routine
14. STOP
```

User's supplied subroutine(s):

The user must supply the following five subroutines, or subroutine(s) with the appropriate entries. Note that the procedures, ZMS, ZMDO, ZMD1 and ZMD2 may be used in these routines. These procedures define data areas, summary records of the problem. The user may also use procedures ZP2T, ZP2OS, ZP21S and ZP2B, which give the data area (restored) for a given 2-D domain. For proper use of these data, the user should consult FEARS program documentation.

```
1.        SUBROUTINE ELTO(MSARY)
          INCLUDE   ZMS
          INCLUDE   ZMD1
          INCLUDE ZMD2
          ......
```

This routine is called as an initialization. The user must set the active 2-D domains by setting MD2TMP(j) to non-zero with j being the active 2-D domain indices. These indices may be obtained by user's defined input, or in case of all 2-D's, through a simple loop on j=1,MSP2, where MSP2 is defined in procedure ZMS (summary array of the full domain) and MD2ARY is defined in procedure ZMD2 (summary records of 2-D domains). user may also mark the 1-D domains, if needed in later calculations. This can be done by storing integer values in MD1TMP(k) where k is the index of the 1-D, and MD1ARY 1-D summary records are defined in procedure ZMD1. On subsequent processing, these integer values may be tested.

The user may define further storage areas needed for subsequent calculations, also perform any input needed, outputs such as headings, etc.

2.
```
      SUBROUTINE ELT1(MA2)
      DIMENSION MA2(26)
        ........
```

This routine is called in the beginning of all active 2-D domains where the array MA2 is the summary array of the 2-D domain with MA2(1) containing the index of the 2-D, MA2(26) the activity non-zero tag set by the user.

3.
```
      SUBROUTINE ELTX(K,N,X,C)
      DIMENSION X(2),C(2,4)
        .........
```

This routine is called for each element in the active 2-D domains where the user must perform his own calculations. The arguments are input arguments as follows:

K = index of the element

N = level number of the element in the transfered unit square,
side length h=2**(-N-1)

X(1),X(2) = local (unit-square) coordinate values of the middle
point of the element

C(1,j),C(2,j) = solution values at the j'th cornerpoint of the element where j is defined in order:

```
2---4
.   .
.   .
1---3
```

Furthermore, the following function/subroutines are also available for use:

H = FUTH(N)

side length of the element

CALL ZP2XY(X,Z)

gives the global coordinates Z(1),Z(2) corresponding to the local coordinates X(1),X(2)

CALL ZP2TRX(X,T,D)

gives transformation matrix T and its determinant D at the local coordinates X(1),X(2):

```
T(1) = d(Z(1))/d(X(1))
T(2) = d(Z(1))/d(X(2))
T(3) = d(Z(2))/d(X(1))
T(4) = d(Z(2))/d(X(2))
```

4.

```
      SUBROUTINE ELT1E(MA2)
      DIMENSION MA2(26)
       ........
```

This routine is called after all elements of an active 2-D domains have been processed by the ELTX routine. This entry allows the user to print any summary for the 2-D domain, MA2 array is the same as for the subroutine ELT1.

5.

```
      SUBROUTINE ELTO(MSARY)
      DIMENSION MSARY(16)
       ........
```

This routine is called after all elements of all active 2-D's have been processed to allow the user to print any summary of his/her calculations.

$$\begin{bmatrix} \nu & 0 & 0 & 1 \end{bmatrix}.$$