

7284

**DEVELOPMENT OF A PREPROCESSOR AND MODIFICATION OF
A FINITE ELEMENT PROCEDURE FOR THE ANALYSIS OF
METAL FORMING PROCESSES**

A MASTER'S THESIS

in

Engineering Sciences

Middle East Technical University

By

Ezgi Günay

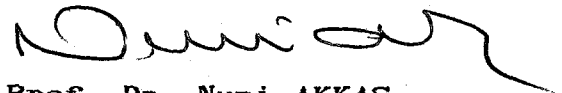
December 1989

**T. C.
Yükseköğretim Kurulu
Dokümantasyon Merkezi**


Approval of the Graduate School of Natural and Applied Sciences


Prof. Dr. Alpay ANKARA
Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science in Engineering Sciences.


Prof. Dr. Nuri AKKAS
Chairman of the Department

We certify that we have read this thesis and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science in Engineering Sciences.


Assoc. Prof. Dr. A. Erman TEKKAYA
Supervisor

Examining Committee in Charge





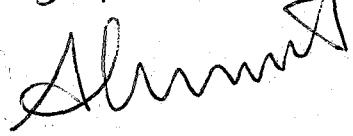
Prof. Dr. Metin GER

Assoc. Prof. Dr. A. Erman TEKKAYA (Supervisor)

Assoc. Prof. Dr. Ömer Gündüz BİLİR

Assoc. Prof. Dr. Turgut TOKDEMİR

Asst. Prof. Dr. Ahmet ERASLAN

DEVELOPMENT OF A PREPROCESSOR AND MODIFICATION OF
A FINITE ELEMENT PROCEDURE FOR THE ANALYSIS OF
METAL FORMING PROCESSES

GÜNAY, Ezgi

Faculty of Engineering

Department of Engineering Sciences , M.S. Thesis

Supervisor : Assoc. Prof. Dr. A.Erman Tekkaya

255 pages, December 1989

ABSTRACT

In this study, the modification of the finite element program EPDAN is performed for the analysis of metal forming processes. Dynamic array dimensioning is used for the modification procedure. In order to use dynamic dimensioning, main program is modified and parameters in subroutine call statements are replaced by COMMON blocks. All modifications are performed for IBM microcomputers.

As a second step, an interactive preprocessor is developed for the metal forming analysis program EPDAN with Quick BASIC language on an IBM-AT microcomputer. The preprocessor DATOR consists of six main modules. These modules generate the required data and prepare the input file interactively for EPDAN. Modules can generate the input files in a short time, correctly and in different formats. DATOR includes its special automatic and interactive mesh generator programs.

Key Words : Metal forming, finite element method, preprocessor, mesh generator .

Science Code : Engineering Sciences (620)

620.01.00. Applied MECHANICS 620.01.01. Mechanics
Statics

ÖN-İŞLEMCİNİN GELİSTİRİLMESİ VE METAL ŞEKİLLENDİRME
İŞLEMİNİN ANALİZİ İÇİN SONLU ELEMAN YÖNTEMİNİN
UYARLANMASI

GÜNAY, Ezgi

Mühendislik Fakültesi

Mühendislik Bilimleri Bölümü, Y.Lisans Tezi

Tez Yöneticisi : Doç. Dr. A. Erman Tekkaya

255 sayfa, Aralık 1989

ÖZET

Bu çalışmada, metal şekillendirme işleminin analizi için sonlu eleman yönteminin modifikasyonu, EPDAN programı üzerinde gerçekleştirilmiştir. Dinamik dizme işlemi uyarlanma sırasında kullanılmıştır. Dinamik dizme işlemini kullanabilmek için, ana program yeniden düzenlenmiş ve alt programları çağırma komutlarındaki parametreler COMMON bloklara yerleştirilmiştir. Butun düzenlemeler, IBM mikro bilgisayarları için yapılmıştır.

İkinci basamak olarak, bir ön-işlemci programı metal şekillendirme analizi programı EPDAN için Quick

BASIC bilgisayar dili ile IBM-AT mikro bilgisayarında gerçekleştirildi. Ön-işlemci programı DATOR, altı alt programdan oluşmuştur. Alt programlar data dosyalarını en kısa zamanda ve değişik formatlarda hazırlarlar. DATOR programı, özel olarak hazırlanmış, otomatik ve etkin olarak çalışan, data hazırlayıcı alt programlarını da içermektedir.

Anahtar Sözcükler : Metal şekillendirme, sonlu eleman yöntemi, ön-işlemci, eleman ağı üreticisi .

Bilim Kodu : Mühendislik Bilimleri (620)

620.01.00. Uygulamalı MEKANİK 620.01.01. Mekanik

Mukavemet

A C K N O W L E D G E M E N T S

I would like to express my earnest gratitude to my supervisor Assoc. Prof. Dr. A. Erman Tekkaya , from the Mechanical Engineering Department , for his invaluable advice, constructive criticism and guidance , helpfull encouragement and interest and also for his continuous support throughout the study.

I wish to thank Utku Tamer from the Department of Basic English for her assistance in the preparation of the text part of the thesis.

I also wish to thank to the Deutsche Gesellschaft fur Technische Zusammenarbeit (GTZ) GmbH, which sponsored on behalf of the Government of the Federal Republic of Germany two IBM/PC AT computer systems as well as to the Stiftung Volkswagenwerk/Hannover , which sponsored the MicroVax/GPX II System to the department of Mechanical Engineering of the Middle East Technical University on which the present study has been carried on.

TABLE OF CONTENTS

| | Page |
|--|------|
| ABSTRACT. | iii |
| ÖZET | v |
| ACKNOWLEDGEMENTS | vii |
| TABLE OF CONTENTS | viii |
| LIST OF TABLES | xii |
| LIST OF FIGURES | xiii |
| NOMENCLATURE | xvi |
| 1. INTRODUCTION | 1 |
| 1.1. Objective of the Thesis | 1 |
| 1.2. The Present Study | 7 |
| 2. LITERATURE SURVEY | 10 |
| 2.1. Finite Element Formulations for Problems of Large Elastic - Plastic Deformation .. | 10 |
| 2.2. Interactive Computer Programs..... | 23 |
| 2.3. Finite Element Programs With Interactive Preprocessor | 27 |
| 2.4. Mesh Generation Procedures | 32 |
| 3. THEORETICAL FUNDAMENTALS OF THE FINITE ELEMENT CODE EPDAN | 39 |

| | Page |
|---|------|
| 3.1. General Concepts | 39 |
| 3.2. Derivation of the Basic Equations for Finite Deformation Plasticity | 42 |
| 4. DESCRIPTION OF THE MODIFICATIONS ON THE FINITE ELEMENT CODE - EPDAN | 60 |
| 4.1. Application of the Dynamic Dimensioning ... | 60 |
| 4.2. Description of the Restructured Code | 67 |
| 4.3. Modification of EPDAN for IBM Microcomputers | 77 |
| 5. THE INTERACTIVE PREPROCESSOR FOR THE ... ELASTIC - PLASTIC CODE EPDAN | 83 |
| 5.1. Description of the Interactive Data Generator Program DATOR | 84 |
| 5.2. Module MAIN | 89 |
| 5.3. Module FIRST | 91 |
| 5.4. Module MODIFY | 92 |
| 5.5. Module ORGANIZE | 99 |
| 5.6. Module MESH | 101 |
| 5.7. Module ISOMESH | 103 |
| 5.8. Description of the Interactive Steps for the | |

| | Page |
|---|------|
| Preparation of the Files | 110 |
| 5.8.1. Step 1 : Preparation of the Input and Output File for DATOR | 111 |
| 5.8.2. Step 2 : Preparation of the Input File for EPDAN | 115 |
| 6. APPLICATIONS | 118 |
| 6.1. Application With Data Files of DATOR | 118 |
| 6.2. Test Run for Simulation of Metal Forming Processes | 133 |
| 6.3. Test Run for Isoparametric Mesh Generation | 138 |
| 7. CONCLUSIONS AND SUMMARY | 151 |
| 7.1. Discussion | 151 |
| 7.2. Conclusions..... | 158 |
| REFERENCES | 162 |
| APPENDICES | |
| Appendix A : INPUT DATA CARDS AND TEST DATA FOR EPDAN | 170 |
| Appendix B : FINITE ELEMENT METHOD FORMULATION | 176 |
| Appendix C : FINITE ELEMENT CODES FOR | |

| | Page |
|--|------|
| MICROCOMPUTERS | 181 |
| Appendix D : MENU LIST OF DATOR | 200 |
| Appendix E : SOURCE LISTING OF THE MODIFIED MAIN PROGRAM OF EPDAN | 226 |
| Appendix F : SOURCE LISTING OF DATOR | 229 |
| Appendix G : ISOPARAMETRIC MESH GENERATION SCREEN OUTPUTS | 253 |



LIST OF TABLES

| | Page |
|---|------|
| Table 1. Pointers of the dynamic array EACDYN) and JACDYN) | 65 |
| Table 2. Comparison of the elapsed times on an CPU time in seconds on IBM - AT microcomputer running at 8 MHz with mathcoprocessor 80836 and 80837 | 66 |



TABLE OF FIGURES

| | Page |
|--|------|
| Figure 1. Kinematic description of the deformation of the body | 43 |
| Figure 2. Mid point stiffness matrix method | 57 |
| Figure 3. Self correcting Euler method | 58 |
| Figure 4. Blank COMMON array JAC(DYN) | 62 |
| Figure 5. Blank COMMON array EAC(DYN) | 63 |
| Figure 6. Flow-chart of the initial form of EPDAN | 68 |
| Figure 7. Flow-chart of the restructured main program of EPDAN | 74 |
| Figure 8. Chained modules and man-machine interaction ... | 90 |
| Figure 9. Function - Key Menu in module MAIN | 92 |
| Figure 10. Material and Function - Key Menu | 93 |
| Figure 11. Master nodes on the isoparametric mesh element | 104 |
| Figure 12. Generation of a mesh for the workpiece | 107 |
| Figure 13. Temporary data file DATOR.DAT | 113 |
| Figure 14. Flow - chart of DATOR for the preparation of the resulting data file | 114 |

| | Page |
|--|------|
| Figure 15. Input processes by | |
| (a) ORGANIZATION-MENU | 119 |
| (b) MESH-MENU | 119 |
| (c) screen oriented case - MENU | 119 |
| Figure 16. Input file with | |
| (a) formatted and without mesh data | 120 |
| (b) formatted and with mesh data | 120 |
| Figure 17. Order of the data names for screen - oriented | |
| file organization | 127 |
| Figure 18. Test data file | |
| (a) of EPDAN for unformatted case | 128 |
| (b) for unformatted case | 128 |
| (c) for formatted case | 128 |
| Figure 19. Order of data names for the sample file | 130 |
| Figure 20. Isoparametric master element and their | |
| coordinates | 132 |
| Figure 21. Extrusion process by EPDAN | 132 |
| Figure 22. Force versus displacement plot | 134 |
| Figure 23. Element node numbering on the workpiece | |
| for test data of EPDAN | 135 |

| | Page |
|---|------|
| Figure 24. Radial force versus nodes plot | 139 |
| Figure 25. Axial force versus nodes plot | 140 |
| Figure 26. Axial stresses versus axial distance plot | 141 |
| Figure 27. Radial stresses versus axial distance plot ... | 142 |
| Figure 28. Tangential stresses versus axial distance plot | 143 |
| Figure 29. Comparison of two types of mesh data with radial stresses | |
| a) for the elements which are touching the die | 144 |
| b) for the elements which are located on the core | 145 |
| Figure 30. Comparison of two different types of mesh data with axial stresses | 146 |
| Figure 31. Comparison of two different types of mesh data with tangential stresses | 147 |
| Figure 32. Radial difference versus axial distance plot | 148 |

NOMENCLATURE

| | |
|-----------------------|--|
| A | area of the body |
| {a} | velocity vector of the nodal points |
| <u>b</u> | vector for body force per unit volume |
| {b} | displacement vector |
| {Δb} | displacement difference vector |
| [B] | element type matrix |
| [C] | rate independent incremental constitutive matrix |
| [D] | elasticity matrix |
| <u>D</u> | rate of deformation tensor |
| <u>D</u> ^E | elastic part of the rate of deformation tensor |
| <u>D</u> ^P | plastic part of the rate of deformation tensor |
| <u>E</u> | Green-Lagrangian strain tensor |
| <u>F</u> | rate of change of the deformation gradient tensor |
| {f ^o } | nodal force rates |
| {ΔF} | force difference vector |
| <u>G</u> | Kirchhoff stress tensor |
| * <u>G</u> | Jaumann rate of Kirchhoff stress tensor |

| | |
|---|---|
| G | shear modulus |
| $\underline{\underline{H}}$ | Second Piola Kirchhoff stress tensor |
| $\underline{\underline{I}}$ | identity tensor |
| J | Jacobian determinant |
| $[K_T]$ | element stiffness matrix |
| $[k_s]$ | initial stress stiffness matrix |
| k | Bulk modulus |
| k_f | flow stress in uniaxial tension |
| $\underline{\underline{L}}$ | spacial gradient tensor |
| $\underline{\underline{L}}$ | elasto - plastic material tensor |
| \underline{n} | unit normal vector |
| $\underline{\underline{S}}$ | First Piola Kirchhoff stress tensor |
| t | time |
| \underline{t} | force vector |
| $\underline{\underline{T}}$ | Cauchy stress tensor |
| $\overset{\wedge}{\underline{\underline{T}}}$ | Truesdell rate of Cauchy stress tensor |
| $\overset{*}{\underline{\underline{T}}}$ | Jaumann rate of Cauchy stress tensor |
| $\dot{\underline{\underline{T}}}$ | time rate of change of the Cauchy stress tensor |

| | |
|-------------------------------------|---|
| $\dot{\underline{\underline{T}}}$ | deviatoric part of the time rate of change of Cauchy stress tensor |
| \underline{u} | displacement vector |
| \underline{v} | velocity vector |
| V | volume of the body |
| \underline{w} | spin tensor |
| α | Poissons' ratio |
| β | constant |
| δ_{ij} | Kronecker delta |
| $\delta \underline{u}$ | virtual displacement vector |
| $\delta \underline{v}$ | arbitrary virtual velocity variation |
| δW | virtual work |
| δw_{ij} | virtual rotation |
| $\dot{\underline{E}}$ | infinitesimal strain rate |
| η | coordinate of the quadrilateral element |
| ξ | coordinate of the quadrilateral element |
| ρ | density |
| $\dot{\underline{\underline{T}}}^*$ | Jaumann rate of Kirchhoff stress tensor |

- v test function
- ϕ_j approximate basis functions for the domains
- φ logarithmic strain in uniaxial tension
- $\dot{\underline{\Psi}}$ vector of rates of nodal degrees of freedom



CHAPTER 1

INTRODUCTION

The finite element solution of metal-forming processes occupies an important place in the nonlinear elastic-plastic deformation analysis. Factors which are included in the solution of elastic-plastic deformation are explained in the first part of this chapter. The user of finite element method meets some difficulties during the preparation of the input files. In order to decrease and/or overcome these difficulties preprocessors are used. The special purpose preprocessor and its features are handled in this chapter as an introductory part.

1.1. Objective of the Thesis

The theory of plasticity can be examined in two ranges. These are metal - forming processes and elastoplastic problems. Metal forming processes such as forging, extrusion, drawing, rolling involve large strains and deformations [1] . Elastoplastic problems

have plastic strains and they are of the same order of magnitude as elastic strains.

Metal - forming is an operation in order to form the shape of the workpiece according to the dimensions of the die. These processes are done under the high pressure conditions. Products which are produced by metal - forming processes have high quality and these processes provide material and energy saving.

For the solution of metal - forming process problems incremental and piecewise linear finite element theory has been developed [2]. Recent progress in the development of non - linear large deformation finite element methods made it possible to formulate the simulation of metal forming processes with complicated die geometries [3]. The nonlinearities arise from three distinct reasons [4,5,6].

i) Geometric nonlinearities : They arise due to large displacements. Geometric nonlinearities have been included in the incremental geometric stiffness (initial stress stiffness matrix). Then , Lagrangian or Green's strain tensor is used in this matrix. At

last by using some additional terms, incremental solution is determined for an updated local coordinate system [2].

ii) Material (or physical) nonlinearities :

Stresses in a deforming body depend on the strains, nonlinearly. For the large strain analysis case, commonly used nonlinear material is the elastic - plastic material and for this material, the linearity of the incremental stress - strain law forms the basis of the equations [2]. In the large plastic problems, Prandtl - Reuss equations are used because these equations [1,2,3,4] include both elastic and plastic parts of the total rate of deformation tensor.

iii) Nonlinear boundary conditions : As a result of changing natural boundary conditions with time and as a function of deformation, nonlinearity occurs.

The additional factors which are used in non - linear finite deformation and finite element formulations are as follows :

(1) The elastic - plastic constitutive equations lead to non - symmetric stiffness matrices [4,5,7,8]. Hill's [9] equations lead to a symmetric stiffness matrix. So Hill's equations are used commonly [10]. The constitutive law is in the rate form. This special feature is applied firstly by McMeeking and Rice. Before this usage, constitutive equation is in the incremental form.

(2) Formulation of rate equilibrium is given in the virtual work equation by Hill [9]. This relation forms the starting relation for the formulation of finite deformation.

(3) Elastic constants which are located in the constitutive equations are assumed to depend on the current stress state. Then material isotropy and isotropic strain hardening are used for the large strain cases.

(4) Elastic - plastic behaviour of metal structures are solved incrementally because of the strain history of structure. So all stiffness equations are solved incrementally.

(5) Elastic constants or material tensor is solved by the help of objective stress rate tensor.

(6) Lagrangian, updated Lagrangian or Eulerian descriptions are used in order to formulate the finite deformation problem.

One of the recent elastic - plastic finite element treatment code EPDAN (acronym : Elastic Plastic Deformation ANalysis) is prepared by Tekkaya [4,5,6]. This program is used for the analysis of residual stresses in extrusion process, the simulation of the Sachs boring - out method, simulation of instabilities and deep - drawing.

The first purpose of this thesis is to modify the code EPDAN for extended numerical experiments. The modification of the code is done to minimize the execution time of the runs, to use the dimensions of the arrays and matrices efficiently and in order to determine a simple structure in the code. These approaches bring several advantages to the user of EPDAN. Minimization of the execution time is obtained in two ways. These are as follows

(1) Application of dynamic array dimensioning.

(2) Application of COMMON blocks.

Generally, finite element programs consist of three main parts. These are called as preprocessor, simulation engine and postprocessor. Simulation engine constitutes the theoretical parts of the finite element code. Output files and plotting options are included by postprocessor. By the help of preprocessors, mesh generation and data generation (i.e. data input, editing, checking, updating) capabilities are performed.

The second purpose of this thesis is to prepare an interactive preprocessor DATOR (acronym : Data GeneraTOR) for the users of EPDAN. Interactive software DATOR has the following fields mainly :

(1) Screen - oriented menu driven data input, editing, updating facilities.

(2) Mesh generation processes.

(3) Data file organizations according to the users' requirements.

DATOR gives the users of EPDAN the following

opportunities :

(1) saving of time with the automatically and quickly prepared input files in a shortest time,

(2) easy input facilities besides educational treatment,

(3) correctly prepared data files for each experimental runs of EPDAN.

Preprocessor DATOR can also be used with the other finite element programs by changing the declarations of data types in the software.

1.2. The Present Study

Preprocessors of the finite element codes are prepared for the users in order to reduce their tedious work while preparing the required input data files. By the help of preprocessors, the amount of time spent is reduced to a minimum while the numerical experiments reach the maximum.

The purpose of this thesis is to prepare a preprocessor and modificate the FE code EPDAN for

extended numerical experiments of metal forming processes.

This thesis has seven chapters. Chapter 1 is the introductory unit. In Chapter 2, literature surveys of the analysis of metal forming process, interactive preprocessors and mesh generation procedures are explained.

The third Chapter mainly deals with the theoretical fundamentals of the finite element code EPDAN which is prepared by Tekkaya [6].

In the fourth Chapter, the writer handles the description of the modifications on the code EPDAN for microcomputers. This section begins with the modification of the code by dynamic array dimensioning, then goes on with the description of the restructured code and finally comes to an end with applied rules for the IBM microcomputers.

In the fifth Chapter, the writer comes up with the necessary explanations about the prepared preprocessor DATOR. In this chapter, explanations and preparation of the modules are given. The writer try to

prepare a preprocessor which has an easy usage and automatically working extended modules for the users of EPDAN.

The sixth Chapter deals with the explanation of the preparation of the input files. Numerical examples also are given.

In the seventh Chapter, a general discussion on the implemented preprocessor, modified code and conclusions are given.

CHAPTER 2

LITERATURE SURVEY

Researches related with large elastic-plastic deformation have an importance in metal-forming industry. Since, high quality, minimum cost and less waste of material are required in the production of metals, researchers are widely encouraged to deal with such work like finding solutions to finite deformation by handling finite element programs. Preprocessors and mesh generation programs form the first main part of the finite element programs. Researches related with these subjects are explained in the following sections.

2.1 Finite Element Formulations for Problems of Large Elastic-Plastic Deformation

For the analysis of large elastic - plastic deformation, finite element formulations are used. In metal forming processes, the geometry of the body, the material properties and boundary conditions change during the deformation. Because of these changing

values the stiffness relationship is nonlinear ; consequently the analysis is performed incrementally. The increment sizes have been found to be 1 or 2% in the finite element analysis [5,6,7].

Finite element programs for a large deformation analysis have used Lagrangian and Eulerian descriptions. The FE - programs which are formulated with Lagrangian description use a fixed mesh w.r.t. the material of the elements. These meshes represent the initial configuration of the elements and they show the fixed reference state for the strain [8,9]. McMeeking and Rice [10] have used the Eulerian description which is based on a mesh, representing the current configuration of the deformation. They started their formulations from Hills' [9] rate equilibrium of the virtual work equation.

$$\int_{V_0} \dot{t}_{ij} \delta (\partial v_i / \partial X_j) dv^0 = \int_{V_0} \dot{b}_i \delta v_i dv^0 + \int_{S_0} \dot{f}_i \delta v_i ds^0 \quad (2.1)$$

where $\underline{f}, \underline{b}, X, \delta v$ have been defined as [10] force vector , body force per unit reference volume, position vector of a material point in that reference state

respectively and t_{ij} , n_i^z have been defined as non-symmetric nominal stress and the reference unit normal.

$$f_j = n_i^z t_{ij} \quad (2.2)$$

They used the Eulerian description in their formulations. McMeeking and Rice denoted that Lagrangian formulation and Eulerian formulation give the similar relations as in the following ,

$$\underline{v} = [N] \dot{\underline{\psi}} \quad \underline{\dot{\epsilon}} = [B] \dot{\underline{\psi}} \quad (2.3)$$

$$\underline{v} = [N] \dot{\underline{\psi}} \quad \underline{D} = [B] \dot{\underline{\psi}} \quad (2.4)$$

where

$$[B_{ij}] = 1/2 [N_i]_{,j} + 1/2 [N_j]_{,i} \quad (2.5)$$

and $\dot{\underline{\psi}}$, \underline{v} , $\underline{\dot{\epsilon}}$, \underline{D} have been defined [10] as the vectors of rates of nodal degrees of freedom, components of velocity, the infinitesimal strain rate at a point in an element, deformation rate at all points in an element. [B], [N] have been defined as element type [10] and shape function matrices. Note

that the equation (2.3) for the Lagrangian description and equation (2.4) for the Eulerian description are to be used respectively. In Euler description (2.4), $[N]$, $[B]$ are related with \underline{v} , \underline{D} , $\dot{\underline{\psi}}$. However in Lagrangian description, the same matrices are related with \underline{v} , $\underline{\underline{e}}$, $\dot{\underline{\underline{e}}}$ of the undeformed mesh. Consequently, the infinitesimal strain rate $\dot{\underline{s}}$ at a point in an element is replaced by the deformation rate \underline{D} at all points in an element. This is the difference between the two relations.

McMeeking and Rice [10] have defined two stiffness terms. The first stiffness equation leads to the relation $\int_{V_0} \tau_{ij}^* \delta D_{ij} dV$ and the second is an initial stress stiffness $[k_s]$ where τ_{ij}^* is the co-rotational rate of Kirchhoff stress or Jaumann stress. But there is no initial strain stiffness as in the formulation of Zienkiewicz [11]. McMeeking and Rice have determined the complete finite element equation for Eulerian description as

$$\left(\int_V [B]^T [C] [B] dV + [k_s] \right) \dot{\underline{u}} = \int_V [N]^T \dot{\underline{b}} dV + \int_S [N]^T \dot{\underline{f}} dS \quad (2.6)$$

where [C] is the rate independent incremental constitutive matrix. Constitutive relation is in the form of

$$\underline{\tau}^* = [C] [D] \quad (2.7)$$

where $\underline{\tau}^*$ is the vector components of $\underline{\tau}^*$. McMeeking and Rice have used the Eulerian description for the treatment of finite rotations and necking analysis. They obtained the results which are close to the exact solution. However, they have observed that smaller elements may cause the beginning of natural loading to take place earlier.

Hibbit, Marcal and Rice [2,10] derived their stiffness terms from Lagrangian description without making any assumptions about the magnitude of the strains and displacements. So they didnot ignore the initial load stiffness matrix. Their stiffness matrix consists of four terms. These are named small strain stiffness, initial load stiffness, initial strain

stiffness and initial stress stiffness. They have considered two types of non-linearities : geometric and material. Body forces are included into the relations. For metal behavior, the constitutive relation is commonly expressed in terms of true stress and strain (e.g. the Prandtl - Reuss equations) rather than as a relation between Kirchhoff stress and Green strain tensors. So they introduced these terms in their constitutive relations by considering the increments .

$$\Delta S_{ij} = D_{ijkl} \Delta E_{kl} \quad (2.8)$$

$$\Delta \sigma_{ij} = C_{ijkl} \Delta e_{kl} \quad (2.9)$$

where ΔS_{ij} , ΔE_{kl} are increments of Kirchhoff stress and Green strain tensors. $\Delta \sigma_{ij}$, Δe_{kl} are increments of stress and strain. C_{ijkl} , D_{ijkl} are the elastic constants . Consequently, equation (2.8) is replaced by the equation (2.9). As in Hill's discussion, they have assumed that the equation (2.9) describes the true stress verses logarithmic strain relation in simple tension . Considering the objectivity and

isotropic hardening, they have derived the material constants .

Needleman [8] has a similar Lagrangian formulation. He has arrived at his equations by using Hills' variational principle [9]. Body forces are neglected . He used the constitutive relation as follows

$$\dot{q}_{ij} = L_{ijkl} \dot{\gamma}_{kl} \quad (2.10)$$

where stress rates \dot{q}_{ij} are related to the strain rates $\dot{\gamma}_{kl}$. His stiffness matrix consists of two terms, one corresponding to plastic loading and the other elastic unloading. If the stress rate of an element is on its current yield surface, then the plastic stiffness matrix is taken . If effective stress rate turns out to be negative, elastic unloading takes place in the next increment.

Fellippa and Sharifi [12] have described another Lagrangian formulation. They didnot put any limitation concerning the size of an increment of deformation. Because of this reason , higher order terms had been

used in their descriptions. In spite of the fact that the increment of the deformation must be small, in order to satisfy the unchanging constitutive rate moduli, their terms are not in tangent modulus approach from one increment to the next [10].

Yagmai and Popov [13] have used the Eulerian formulation in order to make use of a mesh which represents the current configuration of deformation. They have used the variational principle of Fillippa and Sharifi [10,12].

Sharifi and Popov [14] have modified the Yagmais and Popovs' method to investigate the nonlinear bending and buckling behavior of axisymmetric plates and shells. They have included the finite rotations in equations. Their stiffness matrix is the following

$$[K + K_1] \{r\} = \{R\} \quad (2.11)$$

where $[K]$ and $\{R\}$ are the structural stiffness matrix and load vector respectively. $[K_1]$ is the additional indefinite stiffness matrix for postbuckling analysis.

Geometric and material nonlinearities are considered. Their element equilibrium equations contain a load correction term which improves the accuracy and convergency. Their constitutive relation as follows

$$\bar{s}_{ij} = C_{ijmn} \bar{\varepsilon}_{mn} \quad (2.12)$$

where \bar{s}_{ij} and $\bar{\varepsilon}_{mn}$ are increments of Piola stress and strain tensors respectively. C_{ijmn} is the material constants.

Pillinger, Hartley, Sturgess, Rowe [7] have derived the stiffness equations incrementally from Lagrangian formulation. They have assumed that stress and strain are constant during an increment. Stress and strain can be re-evaluated at the end of each increment step. They have identified finite deformation formulations with three stiffness matrix terms which are named the deformation stiffness matrix, the stress increment correction matrix, the constant dilatation matrix. Their incremental technique is based on the Prandtl - Reuss flow rule and Von Mises' yield criterion. They have implemented

the 3D finite element treatment for industrial metal forming problems. For a finite deformation elastic - plastic analysis, the element stiffness matrix K_{ImJn} is considered to be the sum of three terms. Their element stiffness matrices are the following :

$$K_{ImJn} = K_{ImJn}^{(e)} + K_{ImJn}^{(\sigma)} + K_{ImJn}^{(\phi)} \quad (2.13)$$

For a given increment and element , the element stiffness equations are written as

$$\Delta f_{Im} = K_{ImJn} \Delta d_{Jn} \quad (2.14)$$

where Δf_{Ii} is the i th Cartesian component of the increment of force at the I th node of the element , and Δd_{Ii} is the i th component of the incremental displacement of that node from its reference position.

Deformation stiffness matrix :

$$K_{ImJn}^{(e)} = \int (B_{Imi}^T D_{ijkl} B_{Jnkl}) dV \quad (2.15)$$

where the matrix B_{Jnkl} which is a function of position relates the increment of strain at a point to the values of the incremental nodal displacements

$$\Delta \varepsilon_{ij} = B_{Imij} \Delta d_{Im} \quad (2.16)$$

Stress increment correction matrix :

$$K_{ImJn}^{(\sigma)} = \int (\delta_{mn} (\partial N_i / \partial x_i) \sigma_{ik} (\partial N_j / \partial x_k) - B_{Imij} (\delta_{jl} \sigma_{ik} + \delta_{il} \sigma_{jk}) B_{jnkl}) dV \quad (2.17)$$

where δ_{mn} is the Kronecker delta, σ_{ik} is the Cauchy stress tensor, N_i is the element interpolation or shape functions of position . The change in nodal force causes the change of stress in the element.

Stress change occurs due to the

- (1) element deforming (Jaumann increment of Cauchy stress) or
- (2) the original stress alters as the geometry and orientation of the element change .

Constant dilatation correction matrix :

$$K_{ImJn}^{(\phi)} = k [1/V (\int B_{Imii} dV) (B_{Jnjj} dV) - \int B_{Imii} B_{Jnjj} dV] \quad (2.18)$$

where k , B , D , N , V are Bulk modulus, element type matrix, elasticity matrix , shape function matrix and volume respectively . Constant - dilatation method

which leads to the total volume of the element is required to be constant. Their constitutive relation is as follows :

$$\Delta\sigma_{ij} = D_{ijkl}\Delta\varepsilon_{kl} \quad (2.19)$$

where $\Delta\sigma_{ij}$ is the Jaumann increment of Cauchy stress tensor, D_{ijkl} is the material constitutive matrix and $\Delta\varepsilon_{kl}$ is the increment of strain tensor. They have considered the complex and changing boundary conditions and they have applied a frictional restraint to any surface of the workpiece.

Tekkaya [4,5,6] have implemented a finite element treatment for the study of two dimensional metal forming processes. The updated Lagrangian description is used utilizing a generalized Prandtl - Reuss flow rule together with the Von Mises yield condition [4].

Finite element equations have been derived incrementally using the variational principle and the principle of virtual work. Different stress definitions are used. Stresses \underline{T} , \underline{S} and \underline{H} are called as Cauchy stress tensor, First Piola Kirchhoff stress

tensor and Second Piola Kirchhoff stress tensor (see section 3.2).

Tekkaya [5,6] has used the theory which is identical with McMeeking and Rice's as well as Hills' rate equilibrium of the virtual work equation. Tekkaya has identified three stiffness terms.

$$[K]^e = [K_1]^e + [K_2]^e + [K_3]^e \quad (2.20)$$

$$[K_1]^e = \int_{V^e} [B]^e{}^T [\mathcal{L}]^{-1} [B]^e dV^e \quad (2.21)$$

$$[K_2]^e = -2 \int_{V^e} [B]^e{}^T [\sigma_1]^e [B]^e dV^e \quad (2.22)$$

$$[K_3]^e = \int_{V^e} [C]^e{}^T [\sigma_1]^e [C]^e dV^e \quad (2.23)$$

where $[K_1]^e$ is the elastic-plastic stiffness matrix due to geometrically linear materials and summation of $[K_2]^e$ and $[K_3]^e$ gives the stiffness matrix from initial stresses.

In conclusion, he has considered all possible nonlinearities (i.e. geometric, material, nonlinear boundary conditions). Thermal effects and body forces are neglected. The first usage of hypoelastic constitutive equation has been used by McMeeking and

Rice [10]. Hypoelasticity can be defined as a stress increment arising in response to the rate of strain from the immediately preceding state. Tekkaya has used the hypoelastic material law in his finite element procedure (see section 3.2) [4].

2.2 Interactive Computer Programs

The input-output subsystem of a computer provides communication between the central system and user. Programs and data are entered into the computer memory for processing. Results which are obtained from computations are recorded in magnetic tapes and disks. The input-output organization of a computer depends on the size of the computer and devices connected to it. The difference between the large and small systems lies in the amount of hardware that a computer has. Communication depends on the type of peripheral units used. The decreasing prices of the hardwares lead to an increase in the usage of microcomputers [16]. This development causes the

preparation of new softwares for microcomputers. Moreover, interactive programs lead to interactive data transfer according to the users' requirements.

Data transfer between the computer and user is handled in one of the following ways :

- (1) Data transfer under program control.
- (2) Interrupt-initiated I/O operations.
- (3) Direct memory access (DMA) transfer.

In the first choice I/O instructions are written in the programs. For each data transfer process, a new instruction is required. In the program-controlled transfer, the processor stays in a program loop until I/O indicates that it is ready. This process leads to time consumption. Because up to the second instruction, the processor is kept busy needlessly.

The second and third possibilities can be activated with special interrupt facilities and special commands [16].

Interactive data transfer is also the program controlled transfer. It causes loss of time during the

data transfer process and simultaneous interaction.

In spite of this disadvantage, using interactive programming offers the following advantages for long term surveys.:

- (1) Saving of time for surveys.
- (2) Saving of cost for surveys.
- (3) Reduction in data errors.
- (4) Educational data input facilities.
- (5) Ease of checking input files.
- (6) Less tasks for the generation of necessary files.

Data processing operations are performed under the control of CPU of the computer. The CPU contains the hardware components for processing instructions and data. Devices that are under the control of processor are said to be connected on-line [16]. As the command is given by the user, CPU reads and processes it simultaneously. According to the given command, response command is given by CPU to the user. This brief explanation is the description of the interactive process [16].

Today, a large number of finite element programs is prepared for microcomputers [17,18,19,20]. Microcomputers have both advantages and disadvantages.

The advantages are:

- (1) man-machine interaction facilities ,
- (2) inexpensive graphics,
- (3) a user-friendly environment.

As for the disadvantages, they are :

- (1) computing speed,
- (2) limited capacity of hard disks.

In spite of these disadvantages, using [17,18] microcomputers is more popular than the mainframes.

Finite element programs which are prepared for microcomputers are composed of the following sections:

- (1) Interactive preprocessor.
- (2) Simulation engine .
- (3) Interactive postprocessor.

Pre/postprocessors are composed of the following sections :

- (1) Data generation.
- (2) Type of elements generated.

(3) Data input and editing.

(4) Graphics.

Interactive computer programs provide inexperienced users with on-screen prompts. This opportunity gives the user, the ease of preparation of necessary files [20].

In conclusion, the interactive preprocessor DATOR (acronym: Data Generator) is implemented for the finite element code EPDAN (see Chapter 5). It offers the following opportunities:

- (1) Screen - oriented menu driven input process.
- (2) Data generation.
- (3) Mesh generation and design.

2.3. Finite Element Programs With Interactive Preprocessor

Intellectual work needs more and more information. Besides this fact, the volume of the scientific and technical literature is increasing every day. In order to save manpower and time, large information or database systems are developed.

MAKEBASE database is one of the developed databases by Jaroslav Mackerle [17,18,20] which summarizes the finite element codes and their pre/postprocessors.

Noor [21] has performed a survey of computer programs for solution of nonlinear structural and solid mechanics problems. He has summarized the capabilities of thirty-six computer programs in his paper. By the help of the references [17,18,20,21] finite element programs are summarized in the Appendix C.

Interactive data management programs are written with BASIC, PASCAL and FORTRAN computer languages .

Preprocessors consist of several files and modules . Files give the users an opportunity to organize, manipulate, select and use the data, according to their requirements [18].

The interactive functions performed in programs are :

- (A) data definition functions (i.e. decision for a record format/structure),

(B) data manipulation functions (i.e. addition of new records, updating of existing records, listing of specified records, selection of records),

(C) data reduction according to the user-required form (i.e. sorting and formatting functions for the output).

The finite element code CAEFAME was prepared with menu driven interactive preprocessor in U.S.A. It is a special purpose program for three dimensional analysis of frame structures with plates , pipes and equipment supports and plate shell structures [18] . Its preprocessor is composed of the following :

- (1) Screen-oriented menu driven input.
- (2) Mesh generation.

The second example is the CAEPIPE. It was generated in USA for linear static and dynamic 3D analysis of the process/power piping systems. The preprocessor is completely interactive [17]. It offers the following opportunities :

- (1) Screen-oriented menu driven input.

(2) Mesh and load generation.

(3) Automatic generation of a series of runs
and rigid member specifications.

(4) Error checking.

(5) Model generation for instant feed-back.

The third example is the IBA (Interactive Building Analysis). It was generated in Italy for the interactive analysis of framed buildings subjected to vertical and horizontal actions (wind and earthquake). The program is written in BASIC . The preprocessor has the following sections [17] :

(1) Model generation, checking.

(2) Data modifications, corrections.

(3) Numbering of nodes and elements.

(4) Load generation.

Interactive Graphics Nonlinear Optimization Program IGNOP is another interactive program. The data is read initially from the disk file [22]. The program itself is interactive. Interactive programming enables the design engineer to make instantaneous use of his engineering judgment and experience. Code IGNOP is

used by menu items. One of its menus contains the following items :

- (a) problem definition,
- (b) constraint values,
- (c) design variables,
- (d) initial cost,
- (e) penalty function evaluation,
- (f) response factor.

The function keys are interrupt, resume, examine, help, review, reset and exit.

Other special purpose finite element programs with interactive preprocessor are given in Appendix C.

In conclusion , the interactive preprocessor DATOR (acronym: **D**ata **G**enera**TOR**) is implemented for the finite element code EPDAN (see Chapter 5). It offers the following opportunities :

- (1) Screen - oriented menu driven input.
- (2) Data generation.
- (3) Mesh generation and mesh design.

2.4. Mesh Generation Procedures

Man-machine interaction is necessary for the mesh generation processes. Interactive programs give the user the opportunity to control the method being used.

For many practical finite element problems, hundreds of elements and nodes are required. The task of preparing data becomes long and cause tedious work. Besides these disadvantages, the user errors may be seen during the preparation of hundreds of data cards. As a result of these incorrect data cards, the running program gives incorrect results. If the user finds where the error is, it will be corrected. If the user cannot detect where the error comes from, he will need more extra time and do extra work. In order to eliminate these types of difficulties, automatic data and mesh generators are used in the finite element programs. The development of microcomputers allow the implementation of finite element analysis on

microcomputers [23].

The necessary properties of an improved automatic mesh generator are as follows :

- (1) Less tasks for generating and modifying the mesh system.
- (2) Ease of checking the fitting of the result.
- (3) Reliability of the method .
- (4) Ease of handling for the user.

Moreover, the automatic mesh generator determines nodal numbers and their coordinates together with element numbers and their definitions automatically. During the preparation of the input data of the mesh generator, mesh divisions and description of the geometry are declared by the minimum amount of data.

Most of the existing finite element programs include mesh generators with nodal point coordinates and element connectivity. There are approximately 250 finite element programs listed in the survey references by Fredriksson , Mackerle and Noor

[17,18,20,21] . Names of these finite element programs which include the interactive preprocessors with an automatic mesh generator are given in Appendix C.

MICROPUS is one of the finite element programs which is used for the applications of extrusion of polymers [20]. It was prepared by Institut für Kunststoffverarbeitung in West Germany. MICROPUS also permits the simulation of steady state, two dimensional isothermal flow processes of Newtonian and Non-Newtonian liquids. The program can also simulate heat transfer processes considering the temperature dependent material properties. The program has been developed to solve problems appearing in extrusion of polymers. For mesh generation process , axisymmetric isoparametric elements were used. Interactive graphics and mesh generator program MORDOR are included in the preprocessor.

Ghassemi [24] has published a FORTRAN program based on two different methods of automatic mesh generation . Methods are the area system and the

isoparametric coordinate system. Depending on geometrical and material variations, part of the region to be handled is divided into triangular areas. The program is given to generate meshes of angular elements with three or six nodes automatically. For this purpose quadratic shape functions are used. Mesh generator program MESH has the arrays with the dynamic array dimensioning.

Durocher and Gasper [25] have developed an algorithm which automatically generates the meshes for two dimensional structures with bandwidth reduction. Their algorithm is based on the isoparametric coordinate system developed by Zienkiewicz [26]. In their paper, algorithms of the mesh generator program are given.

Suhara and Fukuda [27] have developed an algorithm which automatically generates the meshes for two dimensional structures. In their paper, algorithms of the mesh generator programs for triangular and isoparametric elements are given.

Zienkiewicz and Phillips [26] have developed

an automatic mesh generator program for triangular elements and curved surfaces by isoparametric coordinates. According to their theory, if the whole region in which the mesh is to be generated is described by a quadrilateral of the shape, all of the mesh requirements are generated by specifying

- (1) coordinates of the eight nodal points,
- (2) the number of required subdivisions in ξ and η directions.

Zienkiewicz and Phillips have generated complex shapes.

Cheng [3] has developed an automatic remeshing program to get more accurate results from finite element simulation processes. An important aspect in large deformation metal forming processes is the constantly changing configuration of the deformed body. His idea is to rediscrretize the deformed body at a number of steps, with all the history - dependent quantities being properly transformed from the previous finite element model to the new one. He described his scheme by giving several examples, such as a heading process, an

extrusion process, a one-blow impression-die forging process with flash.

Stefanou [28] published a FORTRAN program which can only be used for two dimensional triangular elements. The whole region must be in a quadrilateral region. In their paper, mesh program is given.

Field and Hatton [29] have described the code SPIFFE for the pre/postprocessing finite element analysis of the three dimensional structures.

Buell and Bush [30] have described the mesh generation methods for two dimensional elements. Their survey consists of two parts: nodal point generation and element generation. The reviewed basic node generation methods are straight line interpolation, sides and parts electro - mechanical devices, simplified finite difference, the equipotential method and the natural coordinate system method.

Taniguchi [19,23] has developed an interactive mesh generator program for generating triangular and quadrilateral areas. The program can generate the element node relation by using the blocking method.

The program was written in BASIC for microcomputers. His model is insufficient for the transient areas. In spite of its insufficiency, the method decreases the amount of input data required for mesh generation and his program is interactive.

Murti and Valliappen [31] have developed an algorithm for numerical inverse isoparametric mapping in remeshing and nodal quantity contouring. They have improved the iterative technique of order N instead of N^2 in a two dimensional mesh. They have used the iterative technique in order to avoid solving a system of nonlinear equations. In their paper the concept of stress contouring is discussed and FORTRAN subroutines are given.

In conclusion, the mesh generation program ISOMESH is implemented interactively in DATQR. Algorithm of the ISOMESH is based on the isoparametric coordinate system denoted by Zienkiewicz [26]. The program can plot the generated mesh on the screen simultaneously.

CHAPTER 3

THEORETICAL FUNDAMENTALS OF THE FINITE ELEMENT CODE EPDAN

As mentioned in Chapter 1 and 2, large elastic - plastic deformation formulations are derived according to the Lagrangian, updated Lagrangian or Eulerian descriptions. Equations are solved incrementally because of the strain history of the structure. Theoretical fundamentals of elastic - plastic deformation analysis program EPDAN which are handled in detail right below, are based on updated Lagrangian description.

3.1. General Concepts

In general the derivations of the governing equations of the engineering problems are not difficult to form. But to determine the exact solutions requires difficult and tedious work. One example is the metal forming process problems. For this type of nonlinear problems, necessary steps and

calculations need considerable amount of work and require not only the complicated mathematical calculations but also the numerical ones.

In such cases the approximate methods of analysis are the finite element method and variational methods. In the variational solution of differential equations, the differential equation is put into an equivalent variational form. The approximate solution is in the form of $\sum c_j \phi_j$ where ϕ_j and c_j are defined as approximate functions and Ritz parameters. Then the approximate solution is the combination of approximate functions of the domains. The parameters c_j are determined by the variational form. The disadvantage of this method is the determination of the approximate functions for different domains [32]. On the other hand, the finite element method provides a solution for this difficulty. Because it gives a systematic procedure for the derivation of the approximate functions. The finite element method has two important special features :

- (1) A geometrically complex domain of the

problem is represented as a collection of geometrically simple elements e.g. triangular and quadrilateral.

(2) For each element, the approximation functions are derived.

The second special feature is based on the idea that any continuous function can be represented by a linear combination of polynomials.

In general, the methods which are used in the code EPDAN [4,5,6] are :

- (1) Variational method .
- (2) Finite element method (see Appendix B).
- (3) Numerical methods for the solution of nonlinear systems of equations. These methods are as follows :

- (i) Self Correcting Euler Method,
- (ii) Mid - Point Stiffness Matrix Method.

3.2. Derivation of the Basic Equations for Finite Deformation Plasticity

Derivation of the basic equations of finite deformation plasticity is based on both Lagrangian and Eulerian formulations. It is assumed that all positions and geometry of the deforming body are known between the initial and final times t^0 and t respectively Fig.1 . In the figure, A is the area of the body, V volume of the body, \underline{t} is the force vector which is acting on the unit area A_t and A_u is the area where displacements are prescribed. Tekkaya [4,5,6] used the deformation and finite strain tensors which are based on the theory of McMeeking and Hill in order to determine the geometry and position of the body at current configuration.

Using the variational methods, the system of governing equations are derived [33,34,35]. Moreover, the variational displacements and functional π which is equal to zero determine the finite element formulation (see Appendix B).

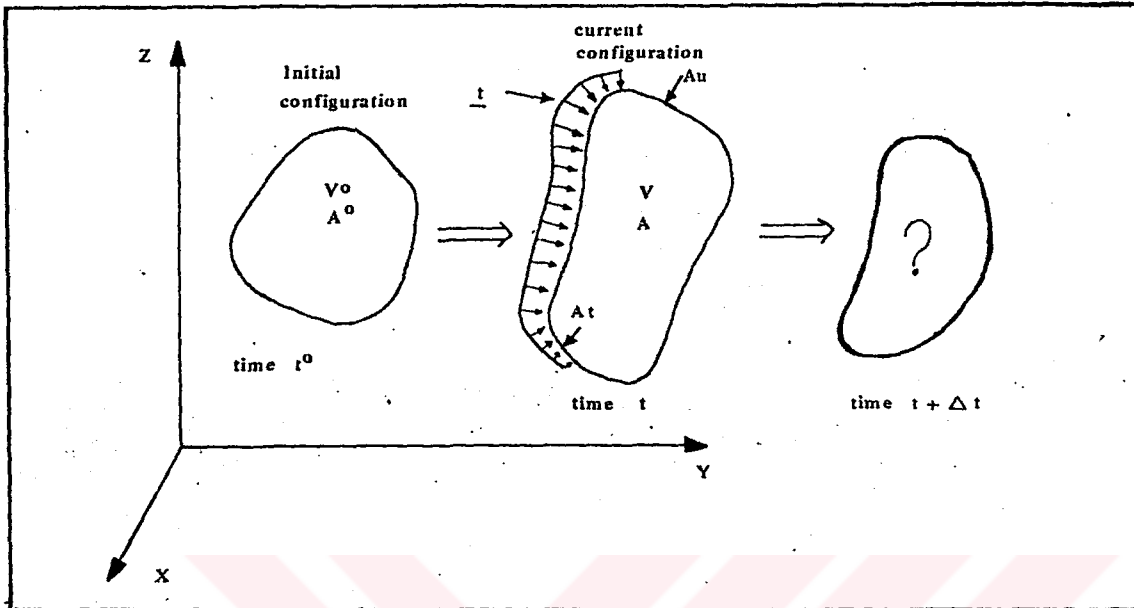


Figure 1. Kinematic definition of the deformation of the body

Virtual displacements satisfy displacement boundary conditions where prescribed. The virtual work δW_{ext} remains unchanged during the virtual displacement and satisfies the equality (3.2) ,

$$\delta \underline{u} = \delta \underline{u}(\underline{x}, t) \quad (3.1)$$

$$\delta W_{ext} = \delta W_{int} \quad (3.2)$$

$$\delta W_{ext} = \int_A \underline{t} \cdot \delta \underline{u} \, dA + \int_V \underline{b} \cdot \delta \underline{u} \, \rho \, dV \quad (3.3)$$

The virtual work W_{ext} is computed with a set of kinematically admissible displacements and statically admissible forces and stresses. The external surface tractions \underline{t} , are also constant during the virtual displacements $\delta \underline{u}$. Because a virtual displacement is defined as an additional displacement from equilibrium displacement and it is equal to zero where actual displacement is prescribed by the boundary conditions [15]. Body forces \underline{b} are neglected during the derivation of the basic equations for finite deformation plasticity.

Kinematically admissible displacement field makes the following possible :

- (a) all prescribed boundary conditions,
- (b) continuous first partial derivatives $\delta \underline{u} = 0$.

Statically admissible stress distribution satisfies the

- (a) equilibrium partial differential equations in the interior of the body,
- (b) traction boundary conditions.

$$\underline{n} \cdot \underline{T} = \underline{t} \quad (3.4)$$

Boundary conditions are defined as

$$\underline{n} \cdot \underline{T} = \underline{t} \quad \text{or} \quad \delta \underline{u} = 0 \quad (3.5)$$

where \underline{n} is the unit vector, \underline{T} is the symmetric Cauchy stress tensor and \underline{t} is defined as

$$\underline{t} = \underline{T} \cdot \underline{n} \quad (3.6)$$

(I) The first step in order to determine the unknowns of the body in current configuration is the derivation of the principle of linear momentum or principle of virtual displacements in the initial configuration. But virtual displacements are defined on the current configuration. Since they are infinitesimal, hypothetical strains and rotations are produced by $\delta \underline{u}$

$$\delta (\partial u_i / \partial x_j) = \delta a_{ij} + \delta \omega_{ij} \quad (3.7)$$

The internal work is given as [33]

$$\delta W_{\text{int}} = \int_V T_{ij} \delta a_{ij} dV = \int_V \underline{T} : \delta \underline{g} dV \quad (3.8)$$

The reverse must also be true for the conservation of mechanical energy. If equation (3.8) satisfies every

kinematically admissible virtual displacement field ,
then the stress field is statically admissible. Upon
using the relations (3.2), (3.3), (3.8) the principle
of virtual displacements is found to be

$$\int_A \underline{t} \cdot \delta u \, dA + \int_V \underline{b} \cdot \delta u \, \rho \, dV = \int_V \underline{T} : \delta \underline{d} \quad (3.9)$$

The determination of the principle of virtual
displacements in the reference configuration [15,33]
requires the following stress tensors and force
vector :

- (A) Cauchy stress tensor \underline{T} is defined on the
current configuration.
- (B) First Piola Kirchhoff stress tensor \underline{S} ,
transforms the force vector on the
undeformed area. \underline{S} is a nonsymmetric tensor.
- (C) Second Piola Kirchhoff stress tensor \underline{H} is
the symmetric one and it transforms the
force on the deformed area element to the
undeformed area element. It is the
hypothetical linear transformation.
- (D) $d\underline{f}$ is the force vector acting on the current

configuration.

Transformation equations can be summarized as

$$d\underline{f} = \underline{I} \cdot (\underline{n} \cdot dA) \quad d\underline{f} = (\underline{n}^0 \cdot dA^0) \cdot \underline{S} \quad \text{and } t=t \quad (3.10)$$

$$d\underline{h} = \underline{F}^{-1} \cdot d\underline{f} \quad d\underline{h} = \underline{H} \cdot (\underline{n}^0 \cdot dA^0) \quad \text{and } t=t \quad (3.11)$$

$$d\underline{g} = \underline{G} \cdot (\underline{n} \cdot dA) \quad d\underline{g} = J \cdot d\underline{f} \quad J = (\det \underline{F}) \quad (3.12)$$

where $d\underline{f}$ is the real force vector acting on the deformed area ($\underline{n}dA$) at $t=t$, $d\underline{h}$ is an hypothetical force acting on the undeformed area at $t=t$, $d\underline{g}$ is the Cauchy stress tensor or Kirchhoff - stress tensor which is acting on the deformed area.

$$\underline{n} \cdot dA = (\rho^0 / \rho) \underline{n}^0 \cdot \underline{F}^{-1} dA^0 \quad (3.13)$$

where $J = (\det \underline{F})$ is the Jacobian determinant of the deformation, \underline{G} is the Kirchhoff stress tensor [33], \underline{F} is the rate of change of the deformation gradient, ρ^0 is the initial density. By using the definitions of $d\underline{f}$, \underline{S} , \underline{H} as given in the equations (3.10), (3.11), (3.12), three relations are determined.

The first one is the First Piola Kirchhoff

stress tensor and in terms of the Cauchy stress tensor, this is

$$\underline{\underline{S}} = \rho^0 / \rho \underline{\underline{F}}^{-1} \cdot \underline{\underline{T}} \quad (3.14)$$

Second relations (3.15) and (3.16) are the expressions for Second Piola Kirchhoff stress tensor in terms of Cauchy stress tensor and First Piola Kirchhoff stress tensor in terms of Second Piola Kirchhoff stress tensor respectively.

$$\underline{\underline{H}} = (\rho^0 / \rho) \underline{\underline{F}}^{-1} \cdot \underline{\underline{T}} \cdot (\underline{\underline{F}}^{-1})^T \quad (3.15)$$

$$\underline{\underline{H}} = \underline{\underline{S}} \cdot (\underline{\underline{F}}^{-1})^T \quad (3.16)$$

Third relation (3.17) is the First Piola Kirchhoff stress tensor in terms of Second Piola Kirchhoff stress tensor.

$$\underline{\underline{S}} = \underline{\underline{H}} \cdot \underline{\underline{F}}^T \quad (3.17)$$

Principle of virtual displacements refers to the current configuration with relation (3.9). However, in finite (geometrically nonlinear) elasticity, this principle refers to the undeformed configuration. So following relations are considered [4,5,6,33]

$$\underline{u} = \underline{x} - \underline{x}^0, \quad \delta \underline{u} = \delta \underline{x}, \quad dV = J dV^0, \quad (3.18)$$

$$\partial(\delta x_i) / \partial(\delta x_j^0) = \delta F_{ij}, \quad \underline{t} dA = \underline{t}^0 dA^0 \quad (3.19)$$

$$\rho^0 dV^0 = \rho dV$$

where $\underline{t}^0(x^0, t^0)$ is the Pseudo traction .

When these relations (3.9), (3.14), (3.18), (3.19) are used the principle of virtual displacements refers to the reference configuration. It is in terms of First Piola Kirchhoff stress tensor \underline{S} [33].

$$\int_{A^0} \underline{t}^0 \cdot \delta \underline{x} dA^0 + \int_{V^0} \underline{b}(x^0, t) \cdot \delta x_i^0 \rho^0 dV^0 = \int_{V^0} \underline{S} : \delta \underline{F}^T dV^0 \quad (3.20)$$

Equation (3.20) may be written in terms of Second Kirchhoff stress tensor \underline{H} , by inserting the relation (3.17).

$$\int_{A^0} \underline{t}^0 \cdot \delta \underline{x} dA^0 + \int_{V^0} \underline{b}(x^0, t) \cdot \delta x_i^0 \rho^0 dV^0 = \int_{V^0} \underline{H} : (\delta \underline{F}^T \cdot \underline{F}) \quad (3.21)$$

Equation (3.21) may be rewritten by using the following relations

$$2\tilde{E}_{ij}(\delta P) = \tilde{F}_{im}^T \tilde{F}_{mj}^{-1} \quad (3.22)$$

$$2\delta \underline{\underline{E}} = \underline{\underline{E}}^T \cdot \delta \underline{\underline{E}} + \delta \underline{\underline{E}}^T \cdot \underline{\underline{E}} \quad (3.23)$$

$$(3.24)$$

$$\int_{V^0} \underline{\underline{H}} : \delta \underline{\underline{E}} \, dV^0 = \int_{A^0} \underline{\underline{t}}^0 \cdot \delta \underline{\underline{x}} \, dA^0 + \int_{V^0} \underline{\underline{b}}(x^0, t) \cdot \delta x \rho^0 \, dV^0$$

where $\underline{\underline{E}}$ is the Green Lagrangian strain at δP . The equation (3.24) is the basic equation for finite deformation geometrically nonlinear elasticity problems [4,5,6,33].

(II) The second step is the determination of the rate form of the principle of virtual displacement or velocities. For finite deformation the constitutive equations must be in the rate form in order to describe the tangential behaviour of the material. By taking the time derivative of equation (3.21)

$$\int_{V^0} \frac{d}{dt} [\underline{\underline{H}} : (\delta \underline{\underline{E}}^T \cdot \underline{\underline{E}})] \, dV^0 = \int_{A^0} \dot{\underline{\underline{t}}}^0 \cdot \delta \underline{\underline{x}} \, dA^0 + \int_{V^0} \dot{\underline{\underline{b}}}(x^0, t) \cdot \delta x \rho^0 \, dV^0 \quad (3.25)$$

the resulting equation is obtained in the following form

$$\int_V \underline{\underline{J}} \left(\hat{\underline{\underline{T}}} : \delta \underline{\underline{L}} + \underline{\underline{T}} : (\delta \underline{\underline{L}}^T \cdot \underline{\underline{L}}) \right) \, dV = \int_{A^0} \dot{\underline{\underline{t}}}^0 \cdot \delta \underline{\underline{v}} \, dA^0 + \int_V \dot{\underline{\underline{b}}} \cdot \delta \underline{\underline{v}} \, dV \quad (3.26)$$

where $\hat{\underline{\underline{T}}}$ is the Truesdell rate of Cauchy stress tensor

$$\hat{\underline{\underline{T}}} = \dot{\underline{\underline{T}}} + \underline{\underline{T}} \cdot (\underline{\underline{I}} : \underline{\underline{L}}) - \underline{\underline{L}} \cdot \underline{\underline{T}} - \underline{\underline{T}} \cdot \underline{\underline{L}}^T \quad (3.27)$$

and

$$\dot{\underline{\underline{H}}} = J \underline{\underline{F}}^{-1} \cdot \hat{\underline{\underline{T}}} \cdot (\underline{\underline{F}}^{-1})^T \quad (3.28)$$

where $\underline{\underline{I}}$ is the identity tensor, $\underline{\underline{L}}$ is spatial gradient tensor.

Truesdell rate of Cauchy stress tensor $\hat{\underline{\underline{T}}}$ in the equation (3.23) is nonlinear term. After the application of the variational methods, the stiffness matrix is unsymmetric. The initial and final times (t^0, t) are assumed to be equal so that

$$x^0 = x, J = 1, A^0 = A, V^0 = V, t^0 = t \quad (3.29)$$

but $\dot{t}^0 \neq \dot{t}$.

After this procedure, the equation (3.26) is modified [4,33].

$$\int_V [\overset{*}{\underline{\underline{G}}} : \delta \underline{\underline{D}} - 2(\underline{\underline{D}} \cdot \underline{\underline{T}}) : \delta \underline{\underline{D}} + \underline{\underline{T}} : (\delta \underline{\underline{L}} \cdot \underline{\underline{L}})] dV = \int_A \dot{t}^0 \cdot \delta \underline{\underline{V}} dA \quad (3.30)$$

In the equation (3.30) $\overset{*}{\underline{\underline{G}}}$ is the Jaumann time change

of Kirchhoff stress tensor, $\underline{\underline{D}}$ is the rate of deformation tensor.

(III) Last step is the determination of the suitable material law. This suitable new material tensor $\underline{\underline{L}}$ must satisfy the following relation for the current configuration [5].

$$\underline{\underline{G}}^* = \underline{\underline{L}}^{-1} : \underline{\underline{D}} \quad (3.31)$$

Then by considering the boundary conditions, the current configuration (geometry and position) of the body can be found. $\underline{\underline{G}}^*$ can be defined in terms of the Jaumann time change of Cauchy stress tensor $\underline{\underline{T}}^*$.

$$\underline{\underline{G}}^* = J \underline{\underline{T}}^* + \dot{J} \underline{\underline{T}} \quad (3.32)$$

$$\underline{\underline{G}}^* = \dot{\underline{\underline{G}}} - \underline{\underline{W}} \cdot \underline{\underline{G}} + \underline{\underline{G}} \cdot \underline{\underline{W}} \quad (3.33)$$

After the determination of the material law, the velocity $\underline{\underline{v}}$ is separated from the equation (3.30) and then it is integrated within the time limit Δt .

Prandtl - Reuss Law is formed as in the linear formulation of Hookes' Law and Levy - Mises equations.

$$\underline{\underline{D}} = \underline{\underline{D}}^E + \underline{\underline{D}}^F \quad (3.34)$$

where $\underline{\underline{D}}^E$, $\underline{\underline{D}}^F$ are the elastic and plastic parts of the rate of deformation tensor $\underline{\underline{D}}$ respectively. Equation (3.34) satisfied only for these conditions

$$\underline{\underline{D}}^E \ll 1 \quad \text{and} \quad \underline{\underline{W}}^E \ll 1 \quad (3.35)$$

$\underline{\underline{D}}^E$ is given in Hookes' Law by the Prandtl-Reuss equation in the following relation

$$\underline{\underline{D}}^E = (\underline{\underline{\dot{T}}})' / (2G) + (\underline{\underline{I}} : \underline{\underline{\dot{T}}}) \underline{\underline{I}} / (9k) \quad (3.36)$$

where $(\underline{\underline{\dot{T}}})'$ is the deviatoric part of the time rate of change of the Cauchy stress tensor, G is the shear modulus, k is the bulk modulus. $\underline{\underline{D}}^F$ is given by the Levy-Mises Law as [5]

$$\underline{\underline{D}}^F = \frac{3}{2} \beta \frac{\dot{k}_f}{k_f (dk_f/d\varphi)} \underline{\underline{T}} \quad (3.37)$$

$$\beta = \begin{cases} 0 & \bar{\sigma}(\underline{\underline{T}}) = k_f(\varphi) \text{ and } \dot{\bar{\sigma}}(\underline{\underline{T}}) < 0 \text{ for} \\ & \text{or, } \bar{\sigma}(\underline{\underline{T}}) < k_f(\varphi) \text{ for} \\ 1 & \bar{\sigma}(\underline{\underline{T}}) = k_f(\varphi) \text{ and } \dot{\bar{\sigma}}(\varphi) \geq 0 \end{cases} \quad (3.38)$$

where k_f is the flow stress in uniaxial tension, φ is

the logarithmic strain in uniaxial tension.

The equations (3.36) and (3.37) are substituted into the equation (3.34) and after the application of inverse process [4,5,33] $\dot{\underline{\underline{T}}}$ is eliminated as

$$\dot{\underline{\underline{T}}} = 2G \left[\underline{\underline{T}} + \frac{\nu}{1-2\nu} \underline{\underline{I}} \underline{\underline{I}} - \beta \frac{\underline{\underline{T}}' \underline{\underline{T}}'}{(2/3)k_f^2 [1 + (1/3G)k_f'(\varphi)]} \right] : \underline{\underline{D}} \quad (3.39)$$

where ν is the Poisson's ratio. This equation is valid for small deformation cases so in order to determine a general formula, the axiom of objectivity is used [4,5,6,15]. Neither $\dot{\underline{\underline{T}}}$ nor Prandtl-Reuss equations are objective, whereas $\dot{\underline{\underline{T}}}^*$ is an objective stress rate.

$$\dot{\underline{\underline{T}}}^* = 2G \left[\underline{\underline{T}} + \frac{\nu}{1-2\nu} \underline{\underline{I}} \underline{\underline{I}} - \beta \frac{\underline{\underline{T}}' \underline{\underline{T}}'}{(2/3)k_f^2 [1 + (1/3G)k_f'(\varphi)]} \right] : \underline{\underline{D}} \quad (3.40)$$

Here $\dot{\underline{\underline{T}}}^*$ is the Jaumann time change of Cauchy stress tensor. Therefore, Prandtl-Reuss equations may be objective when $\dot{\underline{\underline{T}}}^*$ replaces $\dot{\underline{\underline{T}}}$. Moreover, as a result of the following assumptions $\dot{\underline{\underline{G}}}^*$ is found to be equal to $\dot{\underline{\underline{T}}}^*$.

- (1) small elastic deformation,

$$(2) J \approx 1 \quad J \ll 1,$$

$$(3) \underline{\underline{G}}^* = J \underline{\underline{I}}^* + \dot{J} \underline{\underline{I}} \quad \text{and} \quad \dot{J} = 0$$

$$\underline{\underline{G}}^* = \underline{\underline{I}}^* \quad (3.41)$$

From the equations (3.31), (3.40), (3.41) the new material tensor is obtained [5].

$$\underline{\underline{L}}^{-1} = 2G \left[\underline{\underline{I}} + \frac{\nu}{1-2\nu} \underline{\underline{I}}\underline{\underline{I}} - \beta \frac{\dot{\underline{\underline{I}}} \dot{\underline{\underline{I}}}}{(2/3)k_p^2 [1+(1/3G) k_p(\varphi)]} \right] \quad (3.42)$$

(IV) As a result of these steps, a system of equations is obtained. For the equations (3.25), (3.31), (3.38), (3.42) there is not any analytical solution. In order to determine the exact solution, two numerical methods are used (see Section 3.1).

The relation (3.30) is formed as

$$[K_T] \{a\} = \{\dot{f}^0\} \quad (3.43)$$

by using the Ritz method [4,5,34]. In this equation $[K_T]$ is the element stiffness matrix, $\{a\}$ is the velocity vector for the nodal points, $\{\dot{f}^0\}$ is the velocity of the force vector at the nodal points. Equation (3.43) is a linear one for the velocities at the nodal points but within the time range Δt , the

equation becomes nonlinear in terms of change of displacement differences

$$[K]^* \{\Delta b\} = \{\Delta f^0\} \quad (3.44)$$

and

$$[K_T]^* \{\Delta b\} = \int_{\Delta t} [K_T] \{a\} dt \quad (3.45)$$

The equations (3.43) and (3.44) depend on each other as mentioned in relation (3.45). This relation depends on $\{\Delta b\}$ in a complex way because of the [4,5,6]

(1) material response,

(2) geometrical changes.

These two effects cause nonlinear terms in relation (3.45). A nonlinear equation is solved by using the two numerical methods. Nonlinear terms which are caused by the material response are solved by the "Mid point stiffness matrix method". For the first step $[K]^*$ is substituted for $[K_T]$ in the equation and for the given force vector $\{\Delta f^0\}$, a change of displacement vector $\{\Delta b_1\}$ is found. By using the values of $\{\Delta b_1\}$ stress differences are obtained ($\Delta \underline{T}$ and Δk_f). And then using these values see Fig.2, relations (3.46),

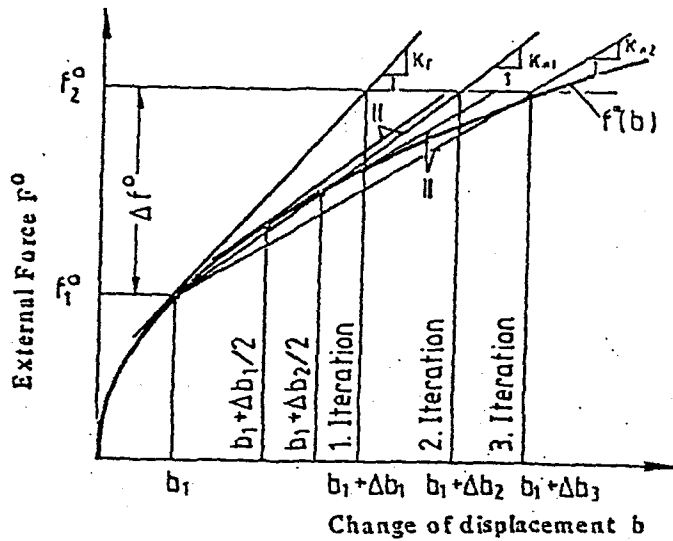


Figure 2. Mid point stiffness matrix method

$$\{ \underline{T} \} = \{ \underline{T}_1 \} + 0.5 \{ \Delta \underline{T} \} \quad (3.46)$$

$$\{ k_f \} = \{ k_{f1} \} + 0.5 \{ \Delta k_{f1} \} \quad (3.47)$$

(3.47) are found. Furthermore, stiffness matrix is calculated $[K_m]$,

$$[K_m] \{ \Delta b \} = \{ \Delta f^0 \} \quad (3.48)$$

The second method "Self correcting Euler method" is used at this point in order to calculate the nonlinear terms caused by the geometrical changes. From equation (3.48), $\{ \Delta b \}$ changes of displacements are calculated. The new position of the body is calculated by the equation [4,5]

$$\{ b_2 \} = \{ b_1 \} + \{ \Delta b \} \quad (3.49)$$

For this new position of the body

$$\langle \Delta F \rangle = \langle f^0 \rangle - \int_V [B]^T \langle \underline{T} \rangle dV \quad (3.50)$$

force difference vector $\langle \Delta F \rangle$ is calculated. The last step for the first iteration is formed in relation to the next time increment

$$[K_m] \langle \Delta b \rangle = \langle \Delta f_o^2 \rangle + \langle \Delta F \rangle \quad (3.51)$$

For the stiffness matrix $[K_m]$, the same steps are redone for the new changes of displacement vector $\langle \Delta b_2 \rangle$. Here the limits of the iterations are considered to be 3 or 4 (see Fig.3).

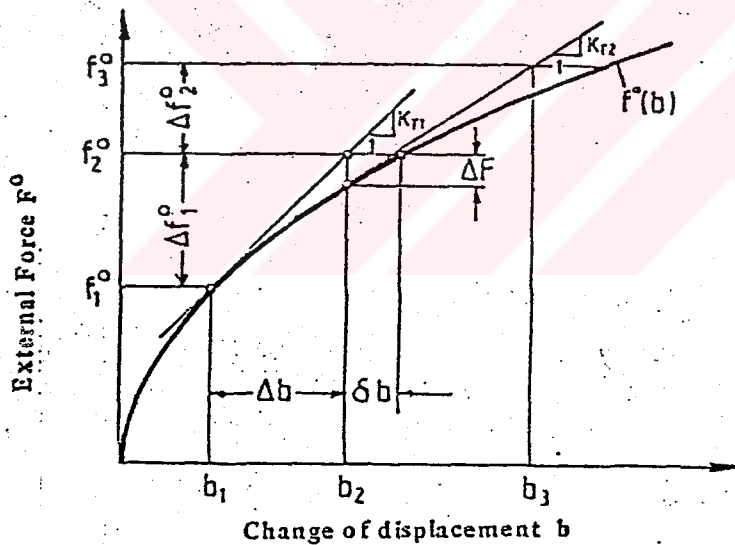


Figure 3. Self correcting Euler method

So two methods are used in the same iteration loop. The error δb is very small, because the time

increment is small.

For the numerical solution of elastic-plastic geometric changes, two reasons for using the small time increments are to be considered. These are [5]

(1) the material law which is defined in terms of the time increments, so that it shows the tangential response of the material with respect to time,

(2) the boundary conditions in metal forming processes, which depend on time, so that the solution can be obtained by using the small time increments for the true solution.

CHAPTER 4

DESCRIPTION OF THE MODIFICATIONS ON THE FINITE ELEMENT CODE - E P D A N

The computer code EPDAN has been implemented by using the FORTRAN 77 language. EPDAN is based on the finite element analysis for two dimensional metal forming operations given in Chapter 3. Modifications of EPDAN play an important role in the application of dynamic array dimensioning. The steps involved in the modification should be follow some certain criteria. First one of them is the reconstruction of the main program and the second one is the application of COMMON blocks. During the reconstruction process, the rules which are particular to an IBM microcomputer should be used and these are handled with in detail in the following sections.

4.1 Application of the Dynamic Dimensioning

Bathe [35] has implemented his finite element code STAP " In-core solution static analysis program " with the dynamic array dimensioning.

Chassemi and Haringer [24] have prepared their

code for two or three dimensional triangular curved surface "MESH" by using the dynamic array dimensioning concept.

Finite element code FEMFAM [17] has been implemented by the language HP-BASIC. All parts of the code has been implemented by the dynamic array dimensioning (see Appendix C).

The dynamic dimensioning is applied to the code EPDAN

- (1) in order to determine the maximum high speed storage allocation in the all globally used arrays and matrices,
- (2) in order to supply the ease of the change of dimensions of all arrays in one step,
- (3) in order to save the storage area from arrays.

To make this possible, the structure of the main program is completely changed.

First, a large real array EA(DYN) is opened in order to include thirty-one arrays. Because

- (a) if dynamic array is opened as an integer type e.g. JA(DYN) ; to this large array can only

integer type data correctly be applied , (see Fig.4 and Table 1.),

(b) if dynamic array is opened as a real type e.g. EA(DYN) ; to this array , both real and integer type arrays can be applied correctly , (see Fig.5 and Table 1.).

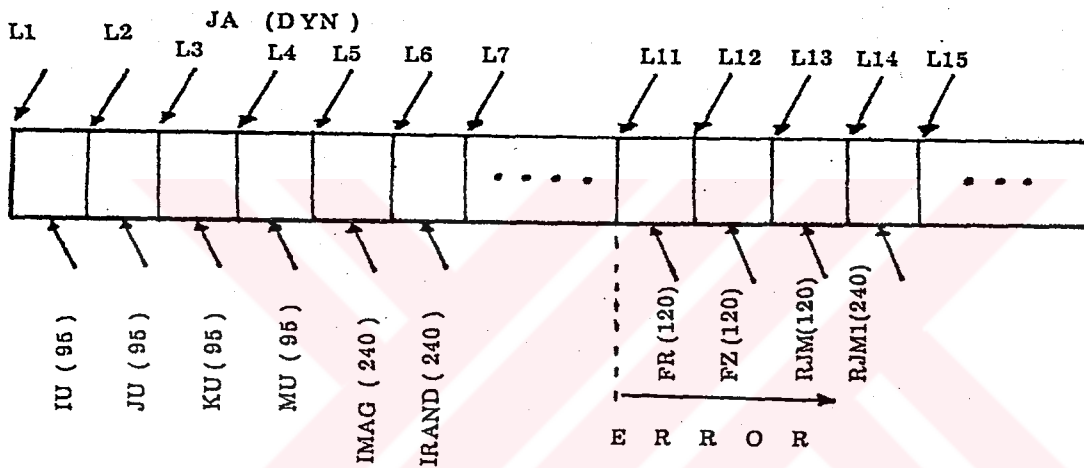


Figure 4. Blank COMMON array JA(DYN)

As a result of the dynamic dimensioning, the reduced main program uses only one large array EA(DYN) instead of these arrays . And subprograms use the required arrays from the array EA(DYN) , (see Fig.5 and see section 4.2 for the construction of the subroutines) .

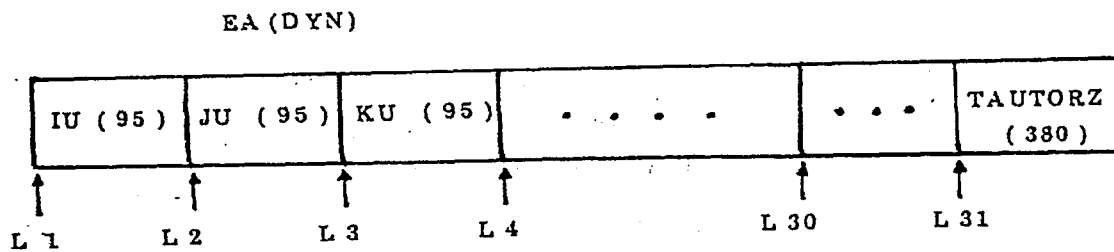


Figure 5. Blank COMMON array EA(DYN)

Second, for each array a starting pointer is asserted by calculating the spaces of the arrays. In the code EPDAN, all arithmetic expressions are declared in a single statement at the beginning of the program as explained in the references [36,37].

```
IMPLICIT REAL*8(A-H,O-Z),INTEGER*4(I-N) (4.1)
```

Data types are represented and stored internally with their byte specifications [38,39]. In order to determine an equality between real and integer operands in dynamic dimensioning, two ways can be referred to

- (i) representing the real data in terms of the integer type of data by dividing the total number of calculations of array or matrice by

two,

- (ii) representing the integer type data in terms of real type of data by multiplying the total number of locations by two.

These conversions are required for the dynamic dimensioning because of the relation (4.1). In code, the second case is used because the dimensions with the odd numbers cause difficulty during the calculations of pointers.

In order to declare the starting location numbers for using arrays, the pointers are calculated in the first lines of the main program (see Table.1 , Fig.5 and Appendix E). The pointers from L1 to L31 are determined in order to declare the starting locations of the thirty-one arrays from the Blank COMMON array EA(DYN) . The pointer L32 is used to determine the maximum number of the dimensions of the common array.

Third, subroutine CALL parameters of the main program are replaced by the common array EA(DYN) with

=====

REAL AND INTEGER TYPE DYNAMIC ARRAYS

=====

| Integer*4 | POINTERS | Real Type array EACDYN | Integer Type array JACDYN |
|------------|--|---------------------------|------------------------------|
| IU(95) | L1=1 | EA(L1) | JA(L1) |
| JU(95) | L2=KEG+L1 | EA(L2) | JA(L2) |
| KU(95) | L3=KEG1+L2 | EA(L3) | JA(L3) |
| MU(95) | L4=KEG2+L3 | EA(L4) | JA(L4) |
| IMAG(240) | L5=KEG2+L4 | EA(L5) | JA(L5) |
| IRAND(240) | L6=KKG4+L5 | EA(L6) | JA(L6) |
| INUM(241) | L7=KKG4+L6 | EA(L7) | JA(L7) |
| IZUST(380) | L8=KNEG2+L7 | EA(L8) | JA(L8) |
| IZIT(380) | L9=KIG+L8 | EA(L9) | JA(L9) |
| IPHKF(380) | L10=KIG+L9 | EA(L10) | JA(L10) |
| | L11: STARTING POINTER FOR THE REAL ARRAYS | | |

Real*8 =====

| | | | |
|--------------|-------------|---------|---------------|
| FR(120) | L11=KIG+L10 | EA(L11) | JA(L11) ERROR |
| FZ(120) | L12=KIG+L11 | EA(L12) | JA(L12) ERROR |
| RJM(120) | L13=KKG+L12 | EA(L13) | JA(L13) ERROR |
| | | | |
| . | . | . | . |
| . | . | . | . |
| . | . | . | . |
| . | . | . | . |
| TAUTORZ(380) | L31=KIG+L30 | EA(L31) | JA(L31) ERROR |

=====

Table 1. Pointers of the dynamic arrays EA(DYN) and JA(DYN)

their pointers. For example, in the main program

subroutine ELACON is defined as

```
CALL ELACON( EA(L8),EA(L13),EA(L14),EA(L21),EA(L23),  
RESTART )
```

and the subroutine itself is defined as

```
SUBROUTINE ELACON(IZUST,RJM,RJM1,TKF,DENS,RESTART).
```

Parameters IZUST(380), RJM(240), RJM1(240), TKF(380), DENS(380)(380), are the arrays and they are declared in the FORTRAN file CCOM1.FOR. All initializations of the variables are declared in the other FORTRAN file CCOM2.FOR.

| NUMBER OF INCREMENTS | TOTAL TIME ELAPSED | |
|----------------------|--------------------------------|------------------------------|
| | Initial form of the code EPDAN | Final form of the code EPDAN |
| 3 | 8. min 11.20 sec | 8.min 1.92 sec |
| 50 | 93.min 8.50 sec | 90.min 49.10 sec |
| 100 | 184.min. 57.47 sec | 181.min 30.79 sec |

Table 2. Comparison of the elapsed times on an CPU time in seconds on IBM - AT microcomputer running at 8 MHz with mathcoprocessor 80836 and 80837

As a result of dynamic array dimensioning implementation, the author has found that the total execution time is reduced three minutes for the fifty and one hundred increments (see Table.2).

4.2. Description of the Restructured Code

The FE - code EPDAN is modified for the IBM microcomputers. The first version of the main program has 590 source lines and it has 36 subroutine call statements (see Fig.6).

In order to modify the code for dynamic array dimensioning, the procedures given below are as follows.

- (1) The main program is rearranged with new subprograms.
- (2) Declaration statements are modified for IBM microcomputers.
- (3) Arguments in the CALL statements are rearranged in the COMMON block statements.
- (4) Code EPDAN is modified for dynamic

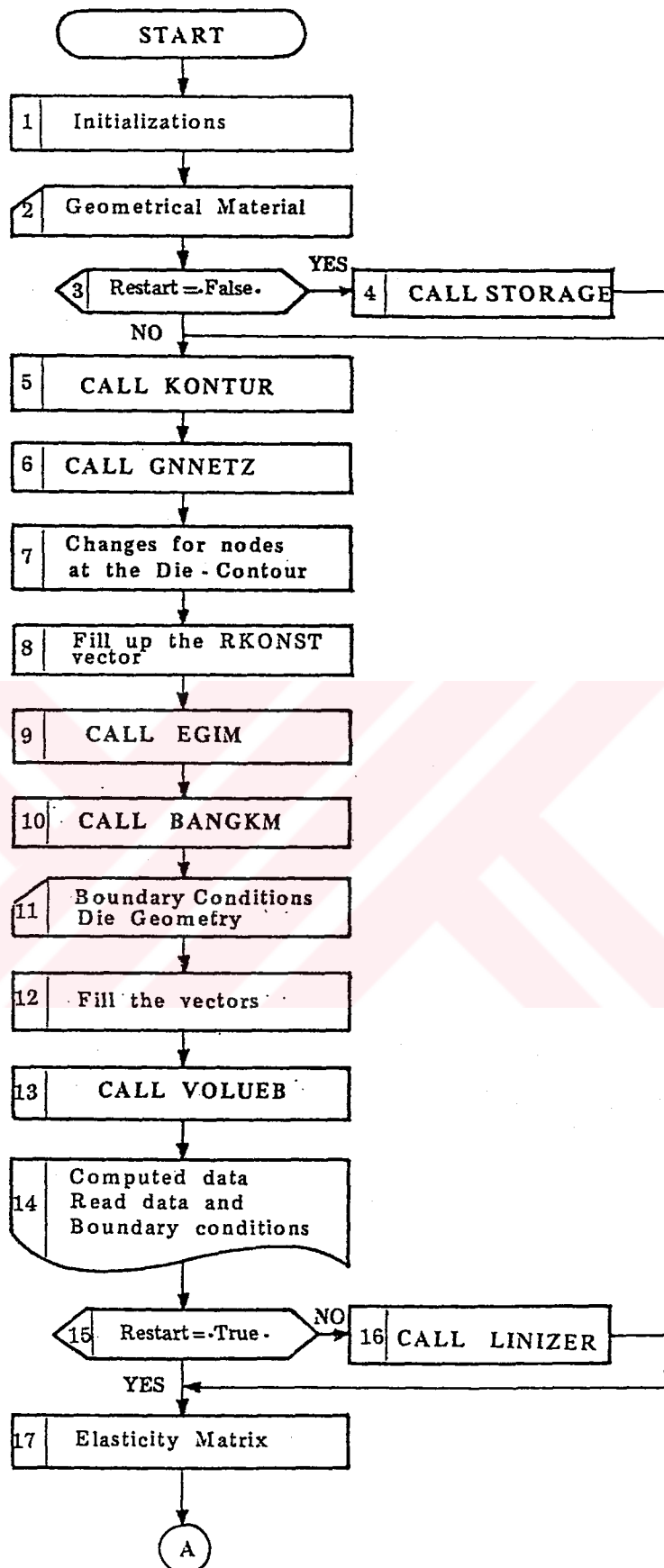
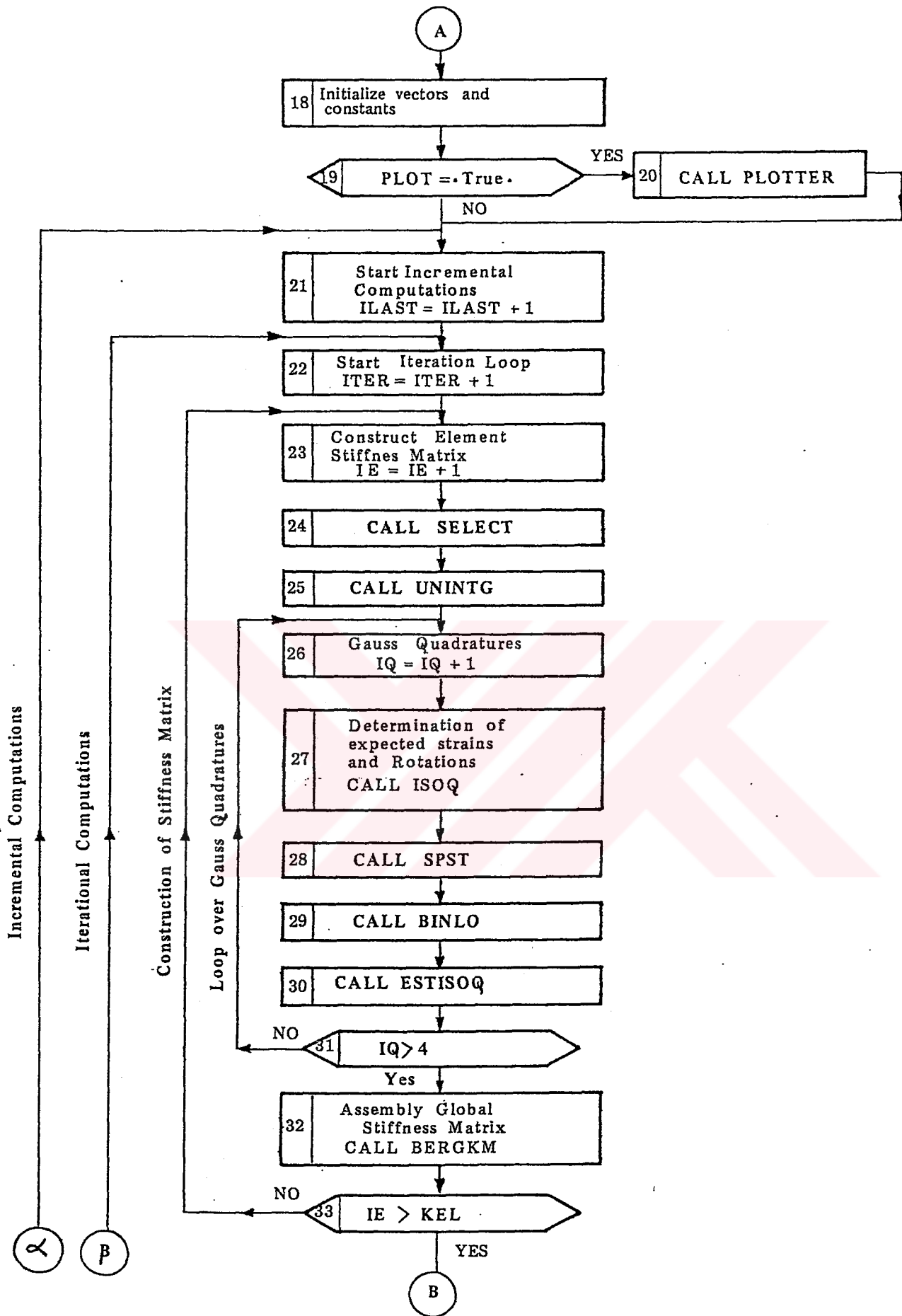
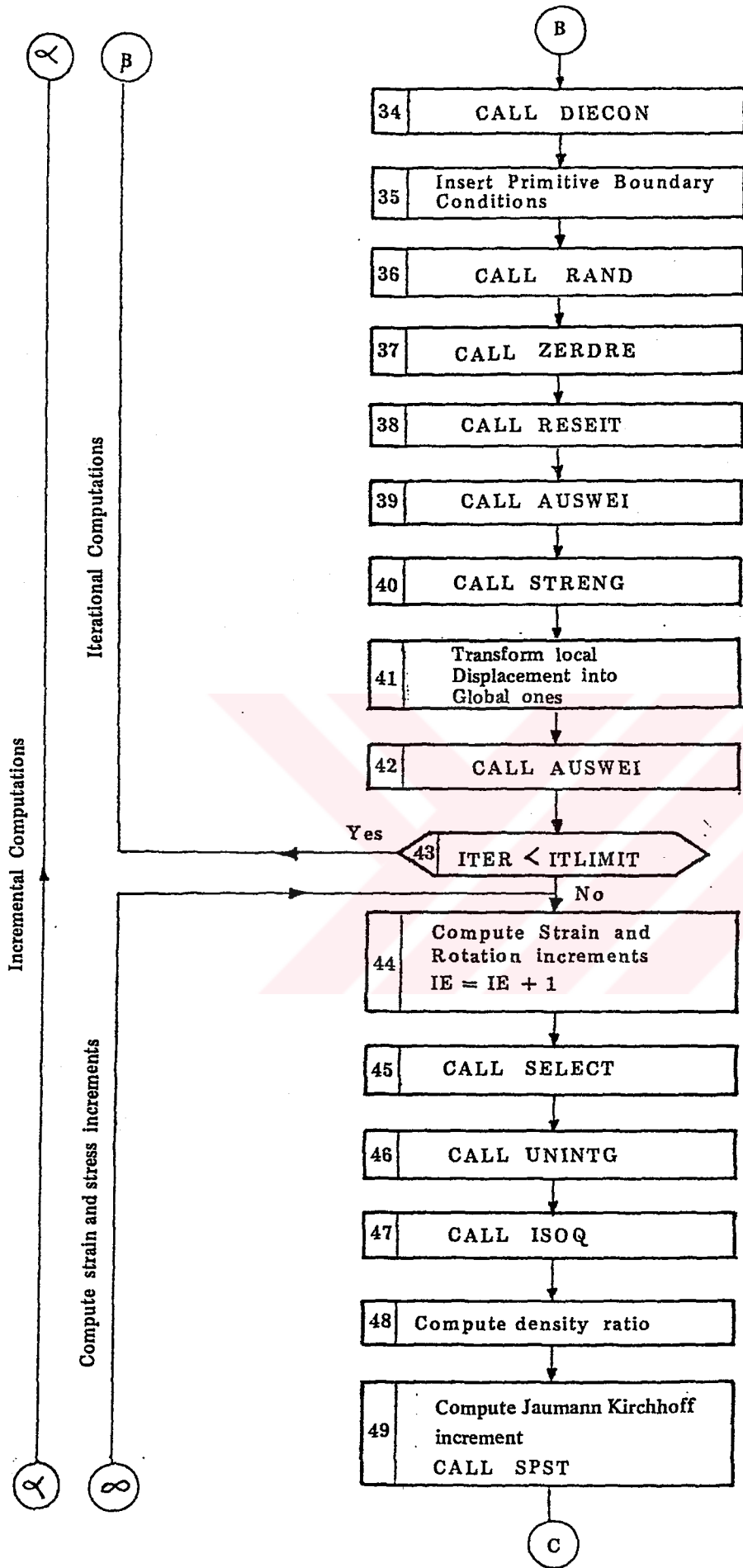
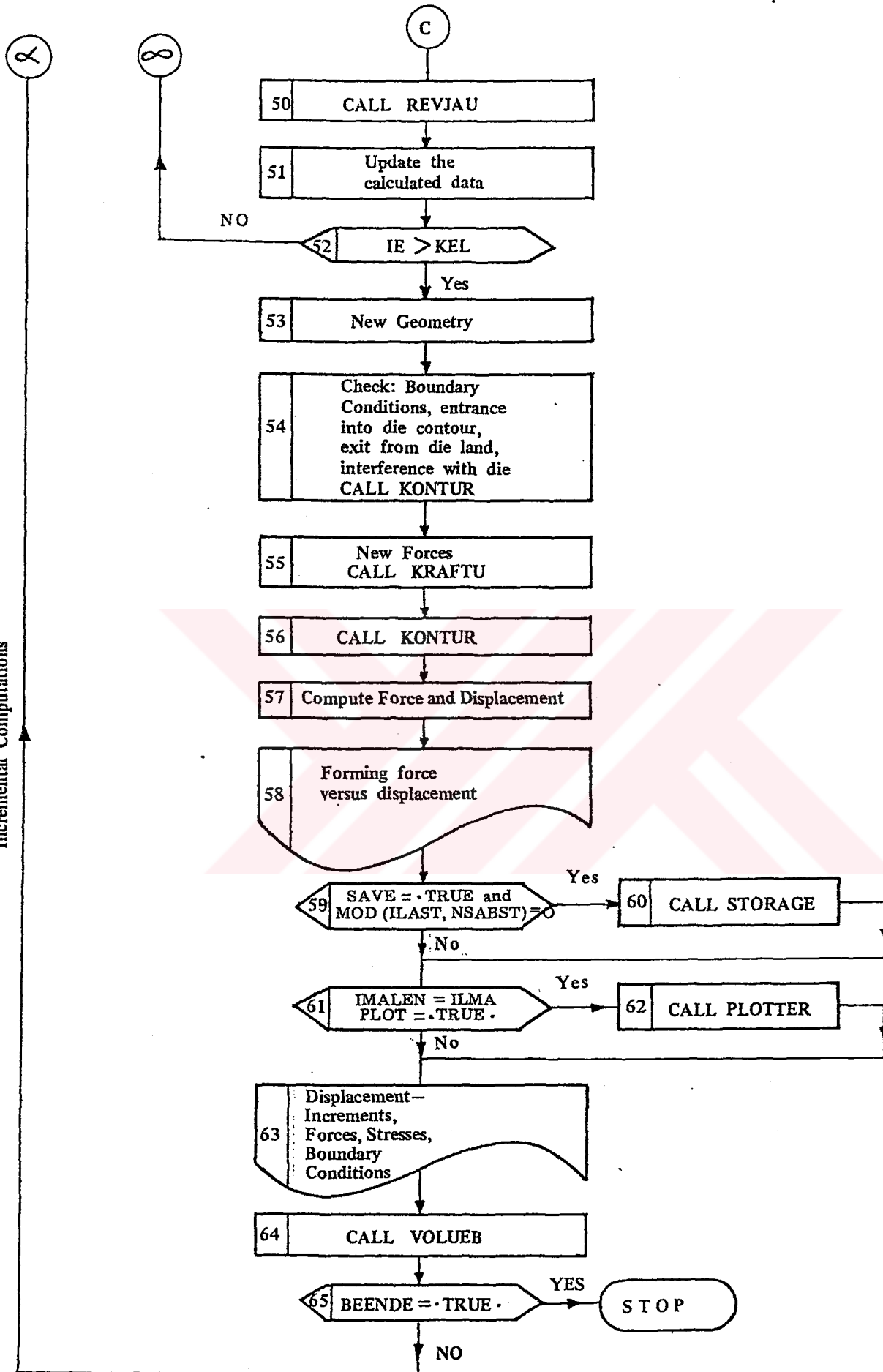


Figure 6. Flow-chart of the initial form of EPDAN







dimensioning with the new implemented form of the main program and subroutines.

First, the main program is reorganized by modifying new subroutines. Thereby, the new main program consists of only calling subroutine names. The general flow diagram of the modified code is given in Fig.7. As it is shown in the diagram, 590 source lines are reduced to 26 subroutine calls. The initialization part of the program which is shown by the box number two is included in the program by the INCLUDE command using the file CCOM2.FOR. COMMON statements are substituted into main program and all other subroutines by the file CCOM1.FOR. Likewise the file CCOM.FOR which is opened for the DIMENSION statements is included in all the subprograms (see Appendix E).

The order of the items in the COMMON statements is important [38] (see Section 4.3). Another important fact is that the same COMMON block with all its statements should be used in all the subroutines whereas this is not required at all in the FORTRAN

language references . Only one blank COMMON block is permitted in the FORTRAN programs, such that

```
COMMON/ARRY1/GKM(4080)

COMMON//EA(20000)
```

The real array EA(DYN) is defined for the dynamic dimensioning in the blank COMMON.

The subroutine OPECLO is called twice during the execution. In the first call, the necessary files are opened as unformatted and sequential. During the second call, all files are closed and the program completes its execution with the stop command.

The subroutine DYNCHK is called after the pointers of the dynamic array are defined. The program gives error messages to the screen and the output file VVFP.OUT if the dimension of the array EA(DYN) exceeds the permitted number (see Fig.7 , Box Number 3).

The subroutine DYNCHK has not any subroutine parameters and dimension declaration . Because if the maximum permitted dimension range is exceeded the program overflows and enters into an infinite loop .

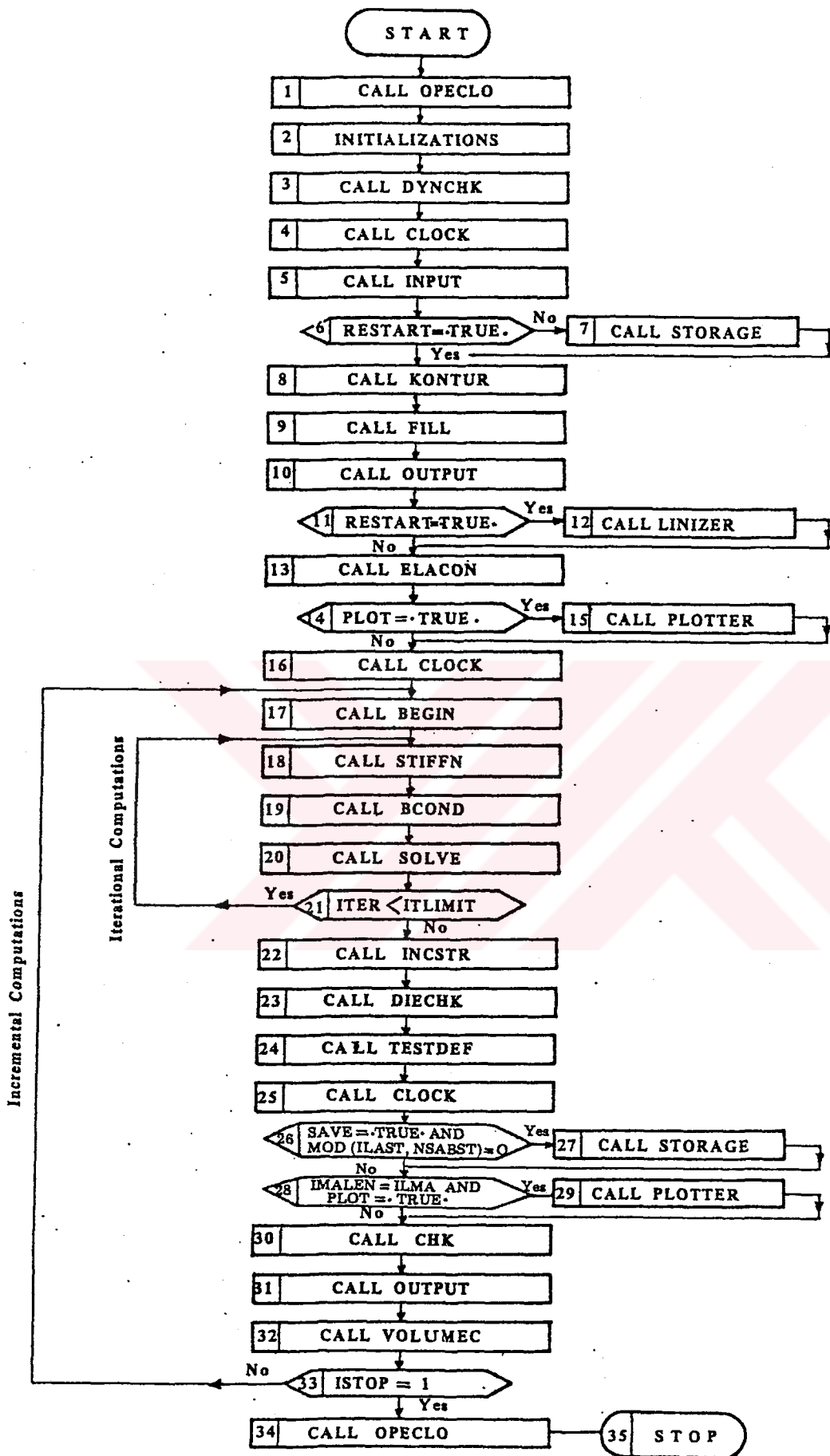


Figure 7. Flow-chart of the restructured main program of EPDAN

The subroutine INPUT is called only once in order to read the data file DATOR.FOR which is prepared by the DATOR. Mesh generation data calculated in the DATOR are also read from the DATOR.FOR (see Fig. 7 , Box Number 5).

The subroutine OUTPUT is called on where the printing procedure is necessary in the main program (see Fig. 7 , Boxes 10, 31).

The main program contains seven IF statements in order to control the flow of incremental and iterational loops. They are defined with the Box Numbers 6, 11, 14, 21, 26, 28, 33 in Fig.7 .

Second, arguments in the CALL statements are reduced to the minimum value in order to simplify and modify the subprograms for dynamic dimensioning case. For example, the flow curve subroutine CALL has the following forms. The first one is the initial form of the subroutine CALL (see Fig.7 , Box Number 12).

```
CALL LINIZER( DPHI, PHIMAX, NPHKF, IPHKF, PHKF, KIN )  
CALL LINIZER( EA(L10), EA(L15) )
```

In the main program each subroutine which is called with the dynamic array EA(DYN) has different numbers of subroutine calls. For example, the call statement for STIFFN with the number 18 corresponds to the Boxes with numbers 23 to 33 in the Fig.7. These call statements refer to the other subroutines which are compiled with the files EPD2, EPD3, EPD4, EPD5, EPD6, EPD7, and CLOCK. Subroutine STIFFN constructs the stiffness matrix.

The subroutine INCSTR is used for the calculation of strain and rotation increments (see Fig.7, Boxes from 44 to 52).

The total number of passing arguments from the subroutine CALL to the COMMON areas are 94 (see Appendix E).

Modular programming gives the user more advantages. If the user is interested in definite subroutines, it is not necessary to compile the whole program. Subroutines which have less source lines gives opportunity for the determination of errors easily. Declarations and the argument orientations of

easily. Declarations and argument orientations of EPDAN are all performed so as to be suitable for the IBM microcomputers: (For detailed information see Section 4.3).

4.3. Modification of E P D A N for IBM Microcomputers

Code EPDAN is implemented with the high speed computer CRAY2 [4]. The modification of the code is done by IBM - AT microcomputer. The name of the compiler is used is RM-FORT [40] which is the fastest one of all the compilers available on the market.

During the modification of the code, some important rules for IBM microcomputers are applied. The steps for the modification of the program are related these rules [36,37,38,39].

- (1) Initialization of all variables before using.
- (2) Compilation of the subprograms by the /B option.
- (3) Organization of the COMMON blocks.

(4) Modification of subroutine CALL statements.

The first step is the initialization of the variables before using them in CALL statements. The error appears when the array names are used as an argument in CALL statements without initialization. The message given by the computer for this type of initialization error is not a normal one. It appears in message " 1025 INCORRECT DSQRT ARGUMENT " [34]. Dummy arguments, names of constants, function and statement functions may not be assigned to initial values. All variables are initialized in file CCOM2.FOR for code EPDAN as mentioned in Section 4.1.

The second rule is used in the compilation of the program. Arrays which are used in the code have a data space greater than 64KB. So subroutines are compiled with the /B option which stands for " big arrays".

The third step consists of the rules for the organization of the COMMON blocks. They are used to pass the variables real and integer. These are mentioned below :

(1) Logical statements are used in the argument list of the CALL statements. Incorrect results are obtained when they pass in the COMMON statements. These results can be explained by the description as mentioned in the IBM Professional FORTRAN references [36,37]. The logical type data declared as a double precision is not supported. Because the location addresses of logical statements .TRUE. and .FALSE. are mixed up. In order to prevent these types of errors which come from the logical type of data in COMMON blocks, one of the following can be used :

- (A) logicals can be declared as LOGICAL*1,
- (B) logicals can be passed in an argument list of the CALL statements,
- (C) instead of logicals, character type declaration can be used [41] .

(2) Blank COMMON block may have different lengths in different program units. Because arguments of blank COMMON may be omitted in any subroutine if it is necessary [37]. An example for this case is given. In subroutine SUB1 for the blank COMMON

the last item may be omitted if it is not needed there.

Subroutine SUB2 requires the array A(100), the parameter R and instead of array B(50) the two different blocks D(25) , U(25) as seen in the following program segment.

```
MAIN PROGRAM          SUBROUTINE SUB1
COMMON//A(100),B(50),P  COMMON//SS(100),BL(50)
.                      .
.                      .
.                      RETURN
CALL SUB1
CALL SUB2
.                      SUBROUTINE SUB2
.                      COMMON//A(100),D(25),U(25),R
.                      .
STOP                  .
END                   RETURN
```

In the subroutine SUB2, U(1) is equivalent to B(26) . So dynamic array EA(DYN) is declared in the blank COMMON block.

Although the variables and arrays in a blank COMMON cannot be assigned to initial values , the dynamic array is initialized before the first usage. When it is not initialized, an error with the number 1025 appears at the subroutine call statements

[36,37].

(3) There is an important rule to observe in the COMMON statements for the IBM computers. It is that one type of data, real or integer, is not allowed to mix with the other. Note that the order may be achieved as in one of the following statements.

```
COMMON/BL1/REAL,REAL,.....,INTEGER,INTEGER,.....,INTEGER
COMMON/BL2/INTEGER,INTEGER,.....,INTEGER
COMMON/BL3/REAL,REAL,.....,REAL
COMMON/BL4/INTEGER,INTEGER,....,REAL,REAL,.....,REAL
```

These rules are explained in the references [38,39] widely. They are applied in the code for the following areas :

- (A) In the argument list of CALL statements.
- (B) In COMMON blocks.
- (C) In the calculation of the pointers for the dynamic array EA(DYN).

(4) The fourth step is the modification of subroutine calls. In addition to the rule, as mentioned above, in the third step, the Hollerith data can only appear in the argument list of the CALL

statements. The error is seen when Hollerith or character type argument is used in the COMMON statements. The remaining arguments in the subroutine calls are : dynamic array EA(DYN) with its pointers ; dummy arguments ; logical arguments ; character type data ; Hollerith data.



CHAPTER 5

THE INTERACTIVE PREPROCESSOR FOR THE ELASTIC - PLASTIC CODE E P D A N

Most of the tedious work for the users of the finite element programs is involved in the preparation of the input files. The preprocessor helps to the user by giving information. The simulation engine or analysis part of the finite element program works in accordance with the input data which are generated at the preprocessor. This fact indicates that a better preprocessor cannot only save the cost for the preparation of the input data but also yields better results by giving opportunity to the user for the modification of the data file interactively.

The interactive preprocessor code DATOR is implemented in modular programming. It has mainly six modules which are called as MAIN, FIRST, MODIFY, ORGANIZE, MESH, ISOMESH. Each module can be used separately with the help of module MAIN. Modular programming gives to the user following advantages :

- (1) Each module has a specific task. This makes

the program handy.

(2) Modules are developed as independent units which can be used in many different applications.

(3) Each module can be modified and rewritten without affecting the other parts of the code.

The interactive preprocessor DATOR has all these advantages. Properties of the code DATOR are handled in detail in the following sections.

5.1. Description of the Interactive Data Generator Code DATOR

The preprocessor DATOR is implemented in Quick BASIC language in order to generate the input file of EPDAN in a easy method, correctly and in a shortest time.

BASIC has two main memories in order to run a program [42]. The first one is the instruction area and the second is the data area. Although it permits 64 K bytes for each storage area , this amount is insufficient for large programs in the microcomputers.

DATOR can reach 153 K bytes of memory when it is compiled. Consequently, for large programs which require more than 64 K bytes of memory, modular programming must be used.

The program modules of DATOR are implemented with the instruction storage area with less than 64 K bytes. Because of this usage, the whole program is prepared and compiled in six different modules.

There are two methods to combine the modules.

These are

(1) Subprograms which are called by CALL statements.

(2) Program modules which are called by CHAIN commands.

Each module is implemented with the help of subroutine calls. The number of subroutine CALL statements are very large. As a result of this fact, passing of parameters in an argument list of CALL statements can cause difficulties. Instead of subroutine CALL statements, GOSUB (statement number of the subroutine) commands are used. This type of

subroutines works as a local part of the main program. Hence, each parameter is used only for one type of process in each module.

For the implementation of the code, CHAIN commands are used in one of the following forms, in order to call the required modules.

```
CHAIN "C:MAIN.EXE"  
CHAIN "C:FIRST.EXE"  
CHAIN "C:MODIFY.EXE"  
CHAIN "C:ORGANIZE.EXE"  
CHAIN "C:MESH.EXE"  
CHAIN "C:ISOMESH.EXE"
```

All parameters are passed by COMMON blocks with the help of dynamic array declaration statements. Dynamic arrays are declared in REDIM statements. COMMON block array parameters which are declared in DIM statements create an error and all values of the passing arguments are erased. For this reason, all arrays are declared in REDIM statements before the COMMON blocks and these COMMON blocks are declared before each CHAIN command. DIM statements are only used for the local parameters.

The Quick BASIC compiler does not accept the CHAIN command with the returning line number as mentioned in the references [42,43]. Hence, COMMON statements are used for the declaration of the returning line number of the program, otherwise chained program starts from the beginning.

The general execution diagram of the DATOR is shown in Fig.8. It shows the order of connection or the chained modules during the execution of the program.

After the compilation process, a linking library is chosen as BRUN10.LIB instead of BCOM10.LIB. It provides the following advantages [42].

(1) Linking time is shorter.

(2) COMMON and CHAIN commands can be used to support a system of programs with shared data. But with the BCOM10.LIB, COMMON is not supported and CHAIN is equivalent to RUN command.

(3) Compiler - generated code, using BRUN10.LIB can be 20 percent smaller than the code generated with BCOM10.LIB.

(4) The BRUN10.EXE run - time module stays in memory. Therefore, it does not need to be reloaded for each program in a system of chained programs.

Furthermore, when programs are compiled with BRUN10.LIB, all files in use are left open during the chaining processes. This facility brings more advantages for the organization of the data files in the modules ORGANIZE, MESH and ISOMESH.

Interactive procedures and computer graphics can easily be implemented with BASIC language. On the other hand, the same program can be written by PL/I, FORTRAN, PASCAL or C language. But in order to write an interactive menu - driven preprocessor with FORTRAN, commands of Assembler language can be activated by the help of byte addresses on the screen.

The graphic card of the microcomputer is not necessary for the menu - drawing cases. Because all the tables are designed with ASCII codes.

In the following sections, the purpose of each of the modules is explained.

5.2. Module MAIN

Module MAIN generates the MAIN-MENU of the code DATOR to satisfy the chain procedure between the other five modules (see Appendix D and Fig.8). It is the main program of the code. MAIN-MENU has four choices. By the help of first two choices , two modules FIRST and MODIFY are chained to the module MAIN . These two modules satisfy the interactive data input process. The creation of the input data file is started by the third choice. The last choice in this menu is used for stopping the program execution.

The execution of the program starts upon writing its name on the screen. The program asks the user the names of the two files. The first one is the input data file which is organized by the user with the help of module ORGANIZE for EPDAN and the second one is the temporary file which is used by DATOR. Both of them have default names as DATOR.INP and DATOR.DAT respectively.

Main program is exactly 170 source lines and it

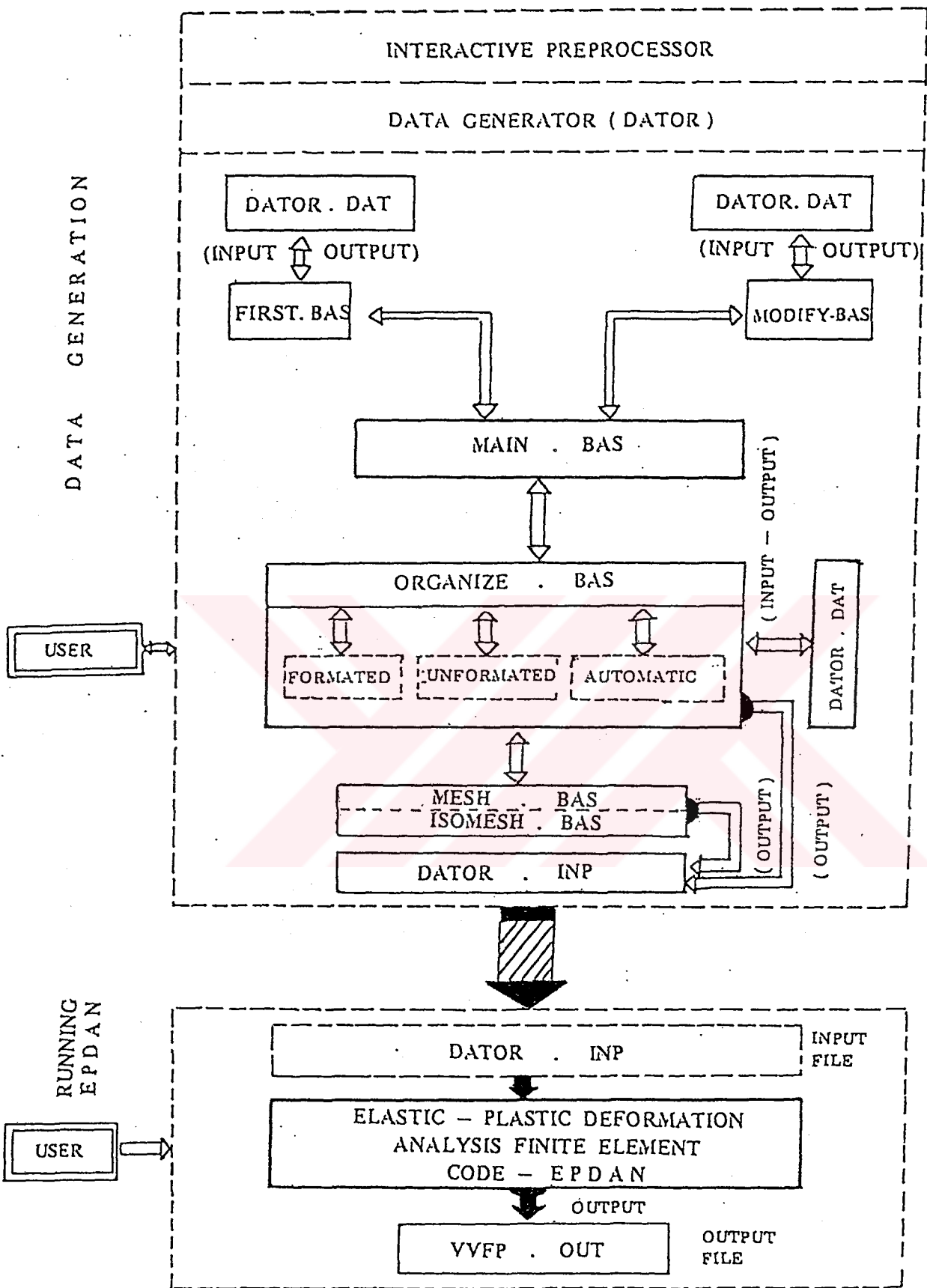


Figure 8. Chained modules and man-machine interaction

needs 7 K bytes in hard disk (see Appendix F).

5.3. Module FIRST

Module FIRST for the first creation of the temporary data file DATOR.DAT is implemented in order to read all the data in a constant order. So the user does not need the main menus in order to record the whole data.

Temporary data file DATOR.DAT is prepared with eight arrays. All of the data-names and their corresponding values are stored there. This input and output file is a common one for the whole menus and it is formatted only once in the module FIRST for each different file creation process.

The function-Key Menu is prepared in order to constitute and use the temporary file easily. The purpose of this menu is as follows :

- (a) to repeat the last executed data,
- (b) to save all previously read data and exit from the module,
- (c) to start the input process in this module

again (see Fig.9).

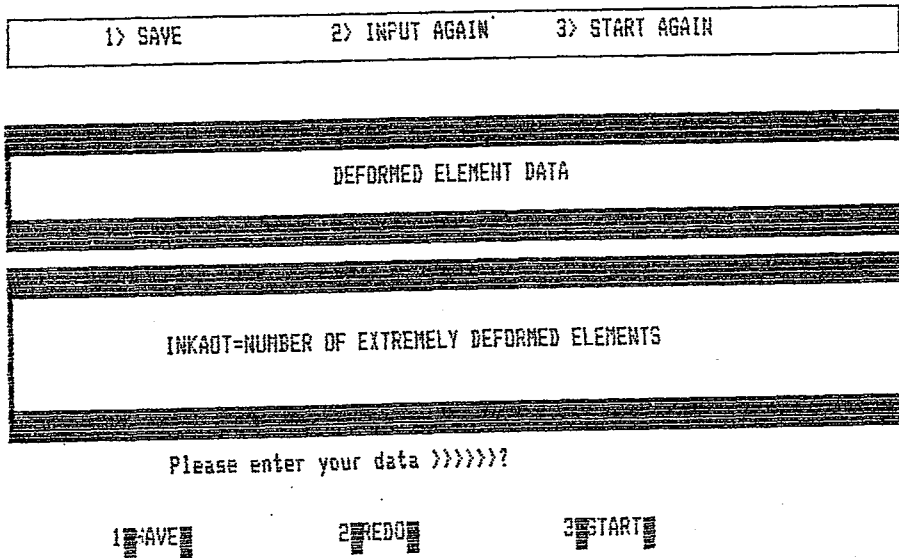


Figure 9. Function-Key Menu in the module FIRST

If the input process is stopped by the help of choice SAVE at any step of the execution, all the data names are recorded into the arrays with their corresponding values.

Module FIRST has 1091 source lines and it needs 47 K bytes in hard disk.

5.4. Module MODIFY

Input data required at analysis or simulation engine is classified into six kinds ; WORKPIECE-MATERIAL, DATA-BASE, ITERATION BOUNDS, DIE,

MATERIAL CONSTANTS

- A > SKFNUL = Initial flow stress
- B > EM = Young's Modulus
- C > PO = Poisson's Ratio
- D > SK = Coefficient of Friction
- E > Return to the WORKPIECE - MENU

(A)

Parallel
passing
between
submenus

Please enter your choice >>>> ?

Remark : Hit the return key, if you don't use FUNC - KEYS for second prompt

| | | | | | | | | | | | | | | | |
|---|------|---|------|---|-------|---|-----|---|-------|---|--------|---|------|---|------|
| 1 | WORK | 2 | DATB | 3 | ITERB | 4 | DIE | 5 | PUNCH | 6 | F.E.M. | 7 | MAIN | 8 | MODT |
|---|------|---|------|---|-------|---|-----|---|-------|---|--------|---|------|---|------|

(B)

Choices of the MODIFICATION - MENU
FUNCTION - KEY MENU for cross passing
between submenus.

Figure 10. Material and Function - Key Menu

PUNCH and F.E.M. (see Appendix D). The main menu in the module MODIFY contains these classifications. This module is implemented in order to modify the previously prepared temporary file DATOR.DAT according to these classifications.

Module MODIFY performs the following jobs:

- (1) Drawing of the classification main-menus.
- (2) Facilities of the screen-oriented menu-driven input.
- (3) Controlling the program between six main-menus.
- (4) Reorganization of the previously stored data in the temporary file.
- (5) Returning to the main module MAIN.

The main menu in this module is the MODIFICATION-MENU which contains seven alternative choices (see Appendix D).

The first choice is WORKPIECE-MATERIAL Menu which has two alternatives. These are defined as elastic and plastic type data. Plastic type data is a keyword which is called IFLITY. If this keyword is

equal to initially assigned value (i.e. ILUDW), two new data are read. These are the constants of the flow curve relation which are named CEE and ENN.

$$\bar{\sigma} = C \bar{\epsilon}^n \quad (5.1)$$

where the constants C and n are the corresponding values of the CEE and ENN. $\bar{\sigma}$ is the flow stress, $\bar{\epsilon}$ is the strain.

Elastic type data is read under the control of MATERIAL-CONSTANTS Menu (see Fig.10 and Appendices A and D). These are called SKFNULL, EM, PO, SK which are initial flow stress, Young's modulus, Poisson's ratio and coefficient of friction data respectively.

The second choice in the MODIFICATION-MENU gives the DATA-BASE Menu . The first two choices "RESTORE-I" and "RESTART-J" gives the flag-type data. The user must give the values of one for I and zero for J in order to continue the execution of the program if electricity is goes out. The values of I and J may have the values zero and one ; in this case , the code starts its execution from the beginning.

The remaining choices provide input of the increments of the force, boundary conditions, displacements and stresses. These are called IDINKF, IDINKV, IDINKR, IDINKS respectively (see Appendices A and D). Another one is ILMA . It is the number of increments after a plot-file is saved. IWRITEG is the number of increments for the output generation.

The third choice in the MODIFICATION-MANU starts the ITER-Menu. With this menu, seven types of data are entered . They are the numbers of increments and iterations which are called LIMVER, ILAST1, LIMVER1, ILAST2, LIMVER2, ITANL, IHALT (see Appendices A and D).

The fourth choice is the DIE Menu . It consists of four alternatives. In the first choice , R and Z , coordinates of the die contour points are entered. In the second and third choices, the inlet and outlet radiuses of the die are read as associated with the names RADA and RADE. The last data is INANL . It is the number of nodes which may hit the die.

The fifth choice is the PUNCH-Menu. It consists of four types of input processes. These are ZEND, IECK, IECK2, ANF. Starting from the first, they are defined as punch displacement at which computation will cease, number of nodes at the end of the punch, number of degree of freedom for the given punch velocity and punch increment respectively (see Appendices A and D).

The last data input process in the MODIFICATION-Menu is the F.E.M.- Menu . The first choice in this menu is the CONTROLLING-Menu. Flag-type data is processed in this menu. These are IHEPSI force flag and IDENGE self-correcting flag . If the value of IHEPSI is given as zero, all forces will be computed. If the value of IDENGE is given as one , self-correction will be done.

The second choice in F.E.M. - Menu gives another menu which is called as NUM-NODES . With the help of this menu ,the number of nodes in R and Z direction are read.

The third alternative is the element type

keyword which is called as ITOP.

The fourth one is the INKAOT. It is the deformed element number.

The fifth choice constitutes the coordinates of the nodes with NODES-Menu (see Appendices A and D).

Controlling of the program between the main menus is satisfied by the Function-Keys Menu which is defined on the below part of the screen in Fig.10 (see Part B). These eight choices refer to the subroutines which call their corresponding submenus according to the user's commands and requirements. This menu is implemented in order to satisfy the change of the program control from one main menu to another. At the same time all the input data is saved in the temporary file.

Module MODIFY is implemented according to the dynamic array dimensioning. Arrays which are used for this purpose satisfy the requirements of the dynamic dimensioning as in the following.

(1) One common array is used.

(2) Each data name of the arrays is defined by

its pointer.

(3) Insertion and extraction of the arrays are considered according to their changing dimensions. For each data processing in the arrays, dimensions are searched and if necessary, common array is updated.

Module MODIFY has 1300 source lines. For this module the required hard disk capacity is 54 K bytes (see Appendix F).

5.5. Module ORGANIZE

Module ORGANIZE prepares the input data file of code EPDAN after a series of processes with the help of previously prepared temporary file DATOR.DAT (see Fig.8).

Module has one main menu which is called ORGANIZE-Menu (see Appendix D). All the data organization processes are controlled from this menu.

There are three types of data files which can be prepared by the user. They are as in the following :

(A) User oriented data file.

(1) Formatted file.

(2) Unformatted file.

(B) Automatically produced data file.

(1) Formatted file.

(C) Mesh generation data file.

(1) Formatted and user oriented file based on isoparametric elements.

(2) Formatted and automatically generated file based on the quadrilateral elements.

Each data file is prepared by its corresponding special subroutine or module.

Formatted and automatic data generation subroutines are prepared in order to produce the data file in a few minutes. Data file is formatted according to the predefined formats of the data in EPDAN. Each data is arranged in the input file DATOR.INP by matching its value and corresponding format in the temporary file DATOR.DAT.

User oriented data files for unformatted and formatted cases are designed according to the users' different necessities. These subroutines are implemented in order to create different types of

input files interactively (see Appendix D).

Module ORGANIZE has 17 K bytes in hard disk and it has 800 source lines exactly.

5.6. Module MESH

EPDAN has two subroutines for the mesh generation process . These are known as GNNETZ and KONTUR . Subroutine KONTUR [4,5,6] is implemented in order to compute the R-coordinates of the die for a given Z-coordinates on the die contour. Subroutine GNNETZ [4,5,6] uses the data which are generated by the subroutine KONTUR. In this subroutine , the nodal coordinates and element - node correspondence are calculated.

The transition region of the workpiece is defined with the mesh elements which hit the die. For this region , the R-coordinates of these elements are taken as smaller than the others.

The weld surface of the die are considered by the two points which are called inlet and outlet radiuses.

Subroutines KONTUR and GNNETZ generate the nodes, elements and coordinates in the y - direction. Mesh elements are quadrilateral. In order to define boundary nodes of the workpiece in the three different regions of the die , four arrays are used. The first one is used for the punch which is touching the node numbers of the workpiece along the y - direction. Node numbers which are in contact with the three different regions of the die are stored in the remaining three arrays.

These two subroutines are translated from FORTRAN 77 to the Quick BASIC according to the file conditions of DATOR and it is called module MESH. In order to create the mesh data automatically by this module, the temporary data file DATOR.DAT is used.

The required input data for this module is in the following

(1) number of nodes in the R and Z directions for the workpiece,

(2) R and Z coordinates of the nodes which are touching the punch and die,

(3) R and Z coordinates of the four die contour points,

(4) inlet and outlet radiuses of the weld surface.

Module MESH has 325 source lines and it needs 8 K bytes in hard disk.

5.7. Module ISOMESH

Zienkiewicz and Phillips [26] have described the theory and techniques which are related with the mesh generator based on isoparametric coordinate system. They have used the curvilinear mapping of quadrilaterals for triangular elements. Because the whole region in which the mesh is to be generated can be described adequately by a quadrilateral of the shape [25,26,43] with eight master nodes (see Fig.11). They have described the criteria for general mesh generation scheme in order to minimize the input data. This criteria is explained as in the following.

(1) An adequate boundary description.

(2) A facility for describing regions of

different materials wherever these occur.

(3) A facility for generating the mesh with required accuracy of idealization.

(4) An adequate mesh element description such as triangular shapes do not lead to numerical ill-conditioning [44].

(5) A numerical system which leads to the best computational efficiency.

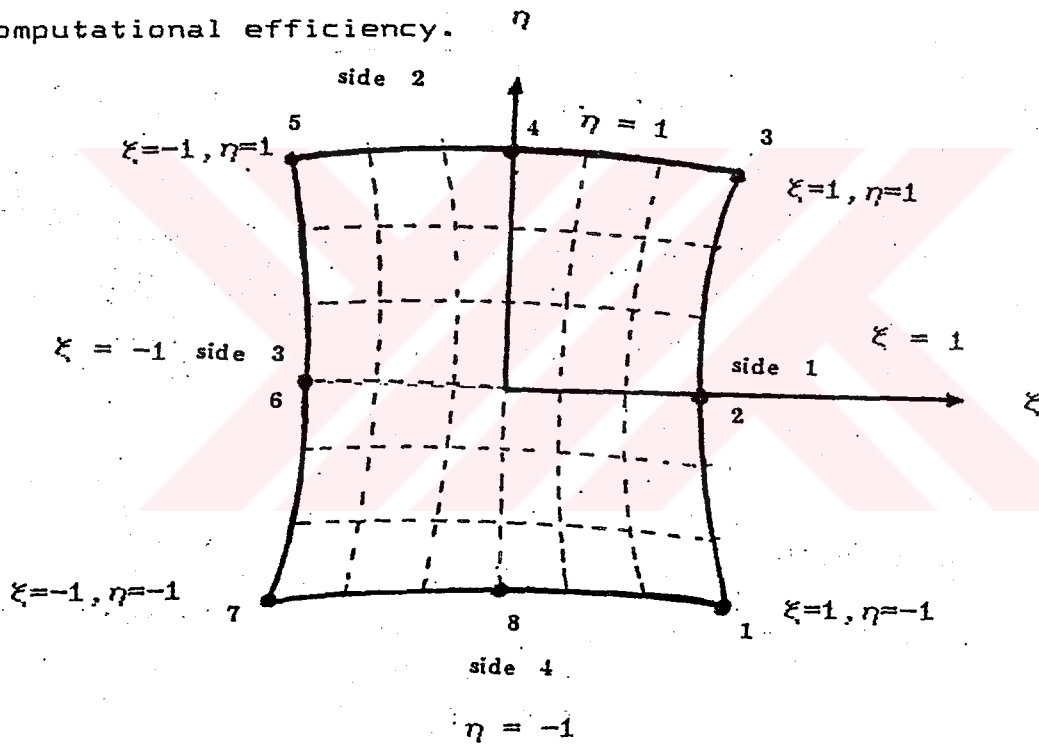


Figure 11. Master nodes on the isoparametric mesh element

The use of isoparametric curvilinear mapping of quadrilateral is introduced by the derivation of

special element forms [25,26,43]. This allows a unique coordinate mapping of curvilinear and Cartesian coordinates. Considering the parabolic quadrilateral (see Fig.11), the coordinates of any point j are calculated by the following formulas [43,45]

$$x_j = \sum_{i=1}^8 N_i(\xi_j, \eta_j) x_i \quad (5.2)$$

$$y_j = \sum_{i=1}^8 N_i(\xi_j, \eta_j) y_i \quad (5.3)$$

The general equation for the shape functions at all corner nodes is

$$N_i = 1/4 (1+\xi\xi_i)(1+\eta\eta_i)(\xi\xi_i+\eta\eta_i-1) \quad (5.4)$$

In general, for midside nodes with $\xi_i=0$, it is

$$N_i = 1/2 (1-\xi^2)(1+\eta\eta_i) \quad (5.5)$$

and for midside nodes with $\eta_i=0$, it is

$$N_i = 1/2 (1+\xi\xi_i)(1-\eta^2) \quad (5.6)$$

The purpose of module ISOMESH is to generate the mesh data of EPDAN with isoparametric coordinates in the preprocessor code DATOR. This module is implemented according to the theory of Zienkiewicz and

Phillips [24].

Special features of the code are as follows :

(a) Data for the mesh generation is prepared interactively.

(b) Mesh is designed according to the specified number of element subdivisions which are given by the user.

(c) Mesh elements are isoparametric.

(d) Element and node numbering direction can be chosen by the user along the ξ and η directions, in order to reduce the size of half band - width. In the module, the node numbering starts from $\xi=-1$ and $\eta=-1$ and proceeds along the axes of constant ξ and η according to user's specification.

(e) Generated mesh is plotted on the graphic screen in order to check the created mesh.

All the input data is given by the user. The number of the types of data is only four. The first one is "NUMEL" which is the number of superelement subdivisions and it is used in order to create the complete superelement mesh. The second one is "NNX"

and "NNY" which are the numbers of the elements along ξ and η directions. The third is the ξ and η coordinates of the eight nodes of the superelement subdivisions. And the fourth one is "ECO" which, according to the isoparametric mesh generation, is the starting node number from the left side of the superelement to combine all superelement subdivisions (see Fig.12). The last three types of data are entered for each superelement subdivisions or each different types of element.

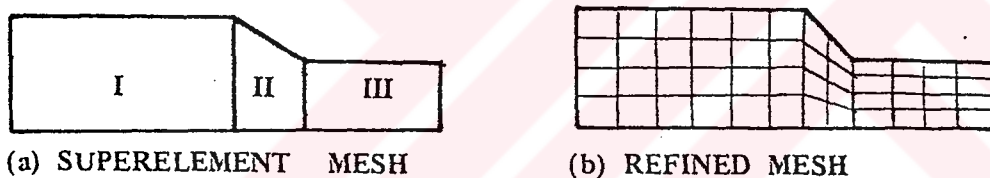


Figure 12. Generation of a mesh for the workpiece

Module ISOMESH generates the final mesh by combining all the given different superelement subdivisions.

In order to prevent the incorrect results, the

user must follow the following steps :

(1) In order to combine the superelements along the ξ and η direction, the last chosen node numbering direction must maintain compatibility with the previous subdivisions because the program generates all nodal and element definition information associated with the previous element nodal definition direction. The same node numbers for the combined side are merged and then deleted.

(2) The user must check the generated mesh on the screen.

(3) The total number of generated mesh nodes must not exceed 500 because of the limited capacity of data area in BASIC. This limited number can be increased if the number of arrays which are declared in REDIM statements and in COMMON blocks are decreased.

(4) The first starting node number must be 1 .

(5) The ξ and η coordinates on the isoparametric element must be given in the order shown on the figure which is drawn on the screen.

(6) The nodes with the numbers 2,4,6,8 must be

on the midsides of the isoparametric element (see Fig.11).

In the module ISOMESH node numbering direction associates with the same direction of element numbering.

The user should use the shortest direction along the sides in order to reduce the half band - width [45] .

$$B=(d.o.f \text{ for each node})[\max |i-j| + 1] \quad (5.7)$$

where B is the half band - width.

The mesh plotting subroutine consists of three main parts. The first part plots the superelement mesh. The second section plots the refined mesh on the superelement. The third is the user oriented mesh plotting created by changing the dimensions of the mesh by the command WINDOW [42]. Meanwile, the user can observe some special parts of the mesh easily.

The last subroutine modifies the generated isoparametric mesh data for EPDAN. Each eight node isoparametric element is translated to the four node quadrilateral mesh element with isoparametric data.

Element and node numbering system is put in order according to EPDAN.

Module ISOMESH has 700 source lines and it needs 20 K bytes in hard disk.

5.8. Description of the Interactive Steps for the Preparation of the Files

Modules MAIN, MESH, FIRST, ORGANIZE, MODIFY and ISOMESH are saved with the "A" option such as [SAVE "MESH.BAS",A]. Because this option saves the programs in ASCII format. Otherwise, BASIC saves the file in a compressed binary format. Although ASCII files take up more space, they give two opportunities to the user.

(1) Programs which are saved in ASCII format can be merged with the other files.

(2) Programs saved in ASCII format can be read as data files.

The preparation of the data files can be examined in two groups. The first file is used as a temporary file "DATOR.DAT" which is used by code DATOR. The second one is the resultant

organized file "DATOR.INP" for EPDAN. The following sections describe the preparation of these files.

5.8.1. Preparation of the Input and Output File for DATOR

The temporary file which is used by the code DATOR is created specifically in order to perform all the data input and output processes in the same file, including the dynamic array dimensioning.

In order to protect the format of the temporary file DATOR.DAT during the input and output processes [42,46], the WRITE command is used instead of PRINT command. DATOR uses the temporary file as an input and output file at the same time. Once the file is used as an input file, in order to use the same file as an output file, the input file is closed and then the same file is opened as an output file as seen in Appendix F. This procedure is followed continuously [47].

WRITE command has special features [42,46]. They

are as follows :

(1) WRITE inserts commas between the numeric and string arrays (see Fig.13).

(2) WRITE inserts quotation marks for the character type data (see Fig.13).

Temporary file is destroyed when PRINT command is used in place of WRITE and after this step , the next INPUT process is cancelled. In module FIRST, the commas and quotation marks are prepared automatically according to the order of the arrays (see Fig.13). So module FIRST must be the first executed module in the code DATOR.

Each data has a specific address in code DATOR. These addresses show the location number of the input data in the temporary file at the corresponding numeric or character arrays. Eight numerical and character type arrays are used as seen in the Fig.13 . Each address is defined as

$$W = N_i \quad i = 1,2,\dots n \quad (5.8)$$

where W is the dummy argument for the address, N_i is

the pointer of the last input data from the screen.

```
"NKAOT",1,0,0,0,"INKAOT","IFLITY","LUDW"  
"HANI",2,0,2,0,"INANI","ITOP","VIER"  
"RK",3,1,4,0,"ITANL",,""  
"ZK",4,7,6,1,"I",,""  
"0,27,8,0,"J",,""  
"0,0,8.964,1,"NSABST",,""  
"0,0,210.4,.025,"DPHI",,""  
"0,0,209.8,2,"PHIMAX",,""  
"0,0,209.2,1,"IDINKF",,""  
"0,0,208.6,1,"IDINKV",,""  
"0,0,208,1,"IDINKR",,""  
"0,0,207.4,1,"IDINKS",,""  
"0,0,206.8,240,"SKFNUL",,""  
"0,0,206.2,210000,"EN",,""  
"0,0,205.6,.3,"PO",,""  
"0,0,205,.05,"SK",,""  
"0,0,204.4,704,"CEE",,""  
"0,0,203.8,.235,"ENN",,""  
"0,0,203.2,1,"IHA",,""  
"0,0,202.6,50,"ILMA",,""  
"0,0,202,50,"IHALT",,""  
"0,0,201.4,1,"IHEPST",,""  
"0,0,200.8,1,"IDENGE",,""  
"0,0,200.2,50,"IWRITEG",,""  
"0,0,199.6,50,"ZEND",,""  
"0,0,199,6,"IX",,""  
"0,0,0,20,"IY",,""  
"0,0,0,8.964,"RB",,""  
"0,0,0,199.6,"ZD",,""  
"0,0,0,7.5,"RC",,""  
"0,0,0,194.136,"ZC",,""  
"0,0,0,7.5,"RD",,""  
"0,0,0,192.136,"ZD",,""  
"0,0,0,8.964,"RA",,""  
"0,0,0,230,"ZA",,""  
"0,0,0,3,"RADE",,""  
"0,0,0,3,"RADA",,""  
"0,0,0,6,"IRN",,""  
"0,0,0,20,"IZN",,""  
"0,0,0,2,"LINVER",,""  
"0,0,0,2,"ILAST1",,""  
"0,0,0,1,"LINVER1",,""  
"0,0,0,3,"ILAST2",,""  
"0,0,0,1,"LINVER2",,""  
"0,0,0,6,"IECK",,""  
"0,0,0,12,"IECK2",,""  
"0,0,0,-.04,"ANF",,""  
"0,0,0,0",,""  
"0,0,0,0",,""  
"0,0,0,0",,""  
"0,0,0,0",,""  
"0,0,0,0",,""  
"0,0,0,0",,""
```

Figure 13. Temporary data file DATOR.DAT

So using the W values, required file organization processes are activated by the user interactively. The

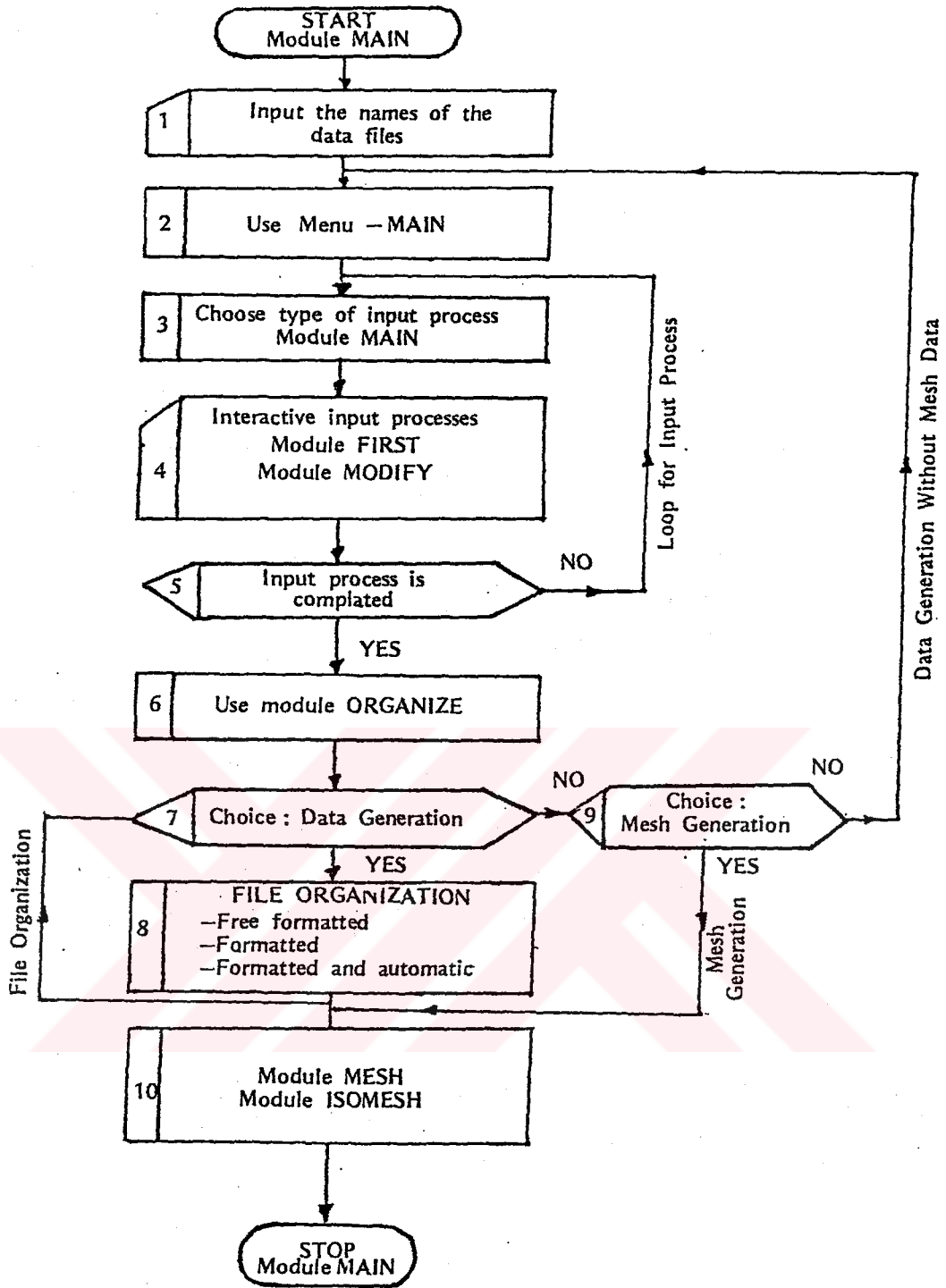


Figure 14. Flow - chart of DATOR for the preparation of the resulting data file

last four arrays as seen in the Fig.13 are arranged by this method. The first four arrays are filled up by the dynamic allocation during the process.

5.8.2. Preparation of the Input File for EPDAN

The resulting data file which is prepared by DATOR is the input file for EPDAN. It can be prepared by one of the following first three methods.

(1) Interactively generated free - formatted data file.

(2) Interactively generated formatted file.

(3) Automatically generated formatted file.

(4) Automatically generated formatted mesh data.

(5) Interactively generated formatted mesh data.

These five types of data can be activated in module ORGANIZE (see Appendices A and F). The last two methods generate the mesh data . Mesh data can also be added to the end of one of the first three data files by the help of menus to form the resulting input file (see Appendix D).

For the first two methods, two special

subprograms are implemented for all kinds of files.

The first subroutine performs the following

interactively :

(1) For each new input data name which is given from the screen, the corresponding data value is found from the temporary file (see Appendix D).

(2) Numerical and character type data are arranged into their numerical and character type matrices.

(3) The sizes of the processing arrays are calculated and their values are stored in the corresponding locations of another matrix. This matrix can be calculated as an address matrix.

(4) Locations of the oriented data are entered with the indexes I and J by the user. Because values must come, in a particular order in the data file Resultant input file is produced line - by - line , in the row order. This process is repeated for each data.

After these procedures , the second subroutine is called. This subroutine mainly performs the two operations.

(a) Data which are located in the matrices are determined according to their type. At the same time, the indexes which are the pointers of the data are found.

(b) Then, according to the determined indexes and data, corresponding format is found by the searching process.

During these procedures, file formatting commands are used [42,46,47]. And then the data is printed on the opened file DATOR.INP (see Fig.8).

Up to this step only one location data is processed from matrices. For the next data, the second column in the first row is read. If it is not empty, the same procedures are repeated. These processes continue up to the last index of the matrices.

Automatically generated and formatted data file is organized by another special purpose subprogram. In this subroutine the formats of each data is assigned directly (see Appendix F). All the preceding steps which are followed are performed automatically in this case.

CHAPTER 6

APPLICATIONS

As the creation of the files requires some basic steps to follow, the user of EPDAN first prepares the necessary input file when using the interactive code DATOR. Then related data suitable for the extrusion process is executed by EPDAN. Later, results are plotted. In order to compare the two different types of mesh data, the writer runs the isoparametric mesh generator program. Finally EPDAN is executed for the resulting input file.

6.1. Application With Data Files of DATOR

Resulting input file is prepared after basic steps. These main steps are summarized in Fig.14.

Module MAIN starts the execution of the program DATOR (see Appendix D). In this module two types of instructions are performed. They are defined with the box numbers 1 and 2 in Fig.14. According to the users' desire one of the modules FIRST or MODIFY is executed for the interactive input process. These are

ORGANIZATION-MENU

- A) AUTOMATIC DATA PREPARATION
- B) SCREEN ORIENTED DATA PREPARATION
- C) AUTOMATIC MESH DATA
- D) ISOPARAMETRIC MESH DATA
- E) RETURN TO MAIN-MODULE

Please enter your choice >>>>>> ?

MESH MENU

- A) ADD MESH GENERATION DATA TO DATOR.INP AUTOMATICALLY.
- B) RETURN TO ORGANIZATION-MENU

ENTER YOUR CHOICE >>>?

PLEASE ENTER 1) THE NAME OF THE NEW DATA
2) LOCATION NUMBERS IN THE DATA FILE

1 NAME OF THE DATA >>>>? ANF

I, J
↓ ↓
? I, ? I

1) SAVE 2) INPUT AGAIN 3) START AGAIN

?

1) SAVE

2) EDG

3) START

Figure 15. Input processes by
(a) ORGANIZATION-MENU
(b) MESH-MENU
(c) screen oriented case - MENU

```

TEST DATA FOR DIELESS EXTRUSION 5 x 19 ELEMENTS 29/07/1989
0
0 0
1 0 1
0.02500 2.00000
1 1 1 1
240.000 210000.0 0.3000 0.0500
LUDW
704.000 0.235
1 50 50 1 1
VIER 50 50.00000
6 20
8.964 199.600 7.500 194.136 7.500 192.136 8.964 230.000
3.0000 3.0000
0.0000
2.0000
4.0000
6.0000
8.0000
8.9640
210.4000
209.8000
209.2000
208.6000
208.0000
207.4000
206.8000
206.2000
205.6000
205.0000
204.4000
203.8000
203.2000
202.6000
202.0000
201.4000
200.8000
200.2000
199.6000
199.0000
2 2 1 3 1
6 12 -0.04000

```

.Figure 16. Input file with
(a) formatted and without mesh data

```

TEST DATA FOR DIELESS EXTRUSION 5 x 19 ELEMENTS 29/07/1989
0
0 0
1 0 1
0.02500 2.00000
1 1 1 1
240.000 210000.0 0.3000 0.0500
LUDW
704.000 0.235
1 50 50 1 1
VIER 50 50.00000
5 20
8.964 199.600 7.500 194.136 7.500 192.136 8.964 230.000
3.0000 3.0000
0.0000

```

2.0000
 4.0000 RADIAL COORDINATES
 6.0000
 8.0000
 8.9640
 210.4000
 209.8000
 209.2000
 208.6000
 208.0000
 207.4000
 206.8000
 206.2000 AXIAL COORDINATES
 205.6000
 205.0000
 204.4000
 203.8000
 203.2000
 202.6000
 202.0000
 201.4000
 200.8000
 200.2000
 199.6000
 199.0000

2 2 1 3 1
 6 12 -0.04000

210.4 0
 210.4 2
 210.4 4
 210.4 6
 210.4 8
 210.4 8.964
 209.8 0
 209.8 2
 209.8 4
 209.8 6
 209.8 8
 209.8 8.964
 209.2 0
 209.2 2
 209.2 4
 209.2 6
 209.2 8
 209.2 8.964
 208.6 0
 208.6 2
 208.6 4
 208.6 6
 208.6 8
 208.6 8.964
 208 0
 208 2
 208 4
 208 6
 208 8
 208 8.964
 207.4 0
 207.4 2
 207.4 4
 207.4 6
 207.4 8
 207.4 8.964
 206.8 0
 206.8 2
 206.8 4
 206.8 6
 206.8 8
 206.8 8.964
 206.2 0

COORDINATES OF THE NODES

206.2 2
 206.2 4
 206.2 6
 206.2 8
 206.2 8.964
 205.6 0
 205.6 2
 205.6 4
 205.6 6
 205.6 8
 205.6 8.964
 205 0
 205 2
 205 4
 205 6
 205 8
 205 8.964
 204.4 0
 204.4 2
 204.4 4
 204.4 6
 204.4 8
 204.4 8.964
 203.8 0
 203.8 2
 203.8 4
 203.8 6
 203.8 8
 203.8 8.964
 203.2 0
 203.2 2
 203.2 4
 203.2 6
 203.2 8
 203.2 8.964
 202.6 0
 202.6 2
 202.6 4
 202.6 6
 202.6 8
 202.6 8.964
 202 0
 202 2
 202 4
 202 6
 202 8
 202 8.964
 201.4 0
 201.4 2
 201.4 4
 201.4 6
 201.4 8
 201.4 8.964
 200.8 0
 200.8 2
 200.8 4
 200.8 6
 200.8 8
 200.8 8.964
 200.2 0
 200.2 2
 200.2 4
 200.2 6
 200.2 8
 200.2 8.964
 199.6 0
 199.6 1.994174
 199.6 3.988349
 199.6 5.982523
 199.6 7.976697
 199.6 8.937889
 199 0
 199 1.964132
 199 3.928263
 199 5.892395
 199 7.856527
 199 8.803238
 1 7 8 2
 2 8 9 3
 3 9 10 4
 4 10 11 5
 5 11 12 6
 7 13 14 8
 8 14 15 9
 9 15 16 10

ELEMENT NODE CORRESPONDANCE

| | | | |
|-----|-----|-----|-----|
| 10 | 16 | 17 | 11 |
| 11 | 17 | 18 | 12 |
| 13 | 19 | 20 | 14 |
| 14 | 20 | 21 | 15 |
| 15 | 21 | 22 | 16 |
| 16 | 22 | 23 | 17 |
| 17 | 23 | 24 | 18 |
| 19 | 25 | 26 | 20 |
| 20 | 26 | 27 | 21 |
| 21 | 27 | 28 | 22 |
| 22 | 28 | 29 | 23 |
| 23 | 29 | 30 | 24 |
| 25 | 31 | 32 | 26 |
| 26 | 32 | 33 | 27 |
| 27 | 33 | 34 | 28 |
| 28 | 34 | 35 | 29 |
| 29 | 35 | 36 | 30 |
| 31 | 37 | 38 | 32 |
| 32 | 38 | 39 | 33 |
| 33 | 39 | 40 | 34 |
| 34 | 40 | 41 | 35 |
| 35 | 41 | 42 | 36 |
| 37 | 43 | 44 | 38 |
| 38 | 44 | 45 | 39 |
| 39 | 45 | 46 | 40 |
| 40 | 46 | 47 | 41 |
| 41 | 47 | 48 | 42 |
| 43 | 49 | 50 | 44 |
| 44 | 50 | 51 | 45 |
| 45 | 51 | 52 | 46 |
| 46 | 52 | 53 | 47 |
| 47 | 53 | 54 | 48 |
| 49 | 55 | 56 | 50 |
| 50 | 56 | 57 | 51 |
| 51 | 57 | 58 | 52 |
| 52 | 58 | 59 | 53 |
| 53 | 59 | 60 | 54 |
| 55 | 61 | 62 | 56 |
| 56 | 62 | 63 | 57 |
| 57 | 63 | 64 | 58 |
| 58 | 64 | 65 | 59 |
| 59 | 65 | 66 | 60 |
| 61 | 67 | 68 | 62 |
| 62 | 68 | 69 | 63 |
| 63 | 69 | 70 | 64 |
| 64 | 70 | 71 | 65 |
| 65 | 71 | 72 | 66 |
| 67 | 73 | 74 | 68 |
| 68 | 74 | 75 | 69 |
| 69 | 75 | 76 | 70 |
| 70 | 76 | 77 | 71 |
| 71 | 77 | 78 | 72 |
| 73 | 79 | 80 | 74 |
| 74 | 80 | 81 | 75 |
| 75 | 81 | 82 | 76 |
| 76 | 82 | 83 | 77 |
| 77 | 83 | 84 | 78 |
| 79 | 85 | 86 | 80 |
| 80 | 86 | 87 | 81 |
| 81 | 87 | 88 | 82 |
| 82 | 88 | 89 | 83 |
| 83 | 89 | 90 | 84 |
| 85 | 91 | 92 | 86 |
| 86 | 92 | 93 | 87 |
| 87 | 93 | 94 | 88 |
| 88 | 94 | 95 | 89 |
| 89 | 95 | 96 | 90 |
| 91 | 97 | 98 | 92 |
| 92 | 98 | 99 | 93 |
| 93 | 99 | 100 | 94 |
| 94 | 100 | 101 | 95 |
| 95 | 101 | 102 | 96 |
| 97 | 103 | 104 | 98 |
| 98 | 104 | 105 | 99 |
| 99 | 105 | 106 | 100 |
| 100 | 106 | 107 | 101 |
| 101 | 107 | 108 | 102 |
| 103 | 109 | 110 | 104 |
| 104 | 110 | 111 | 105 |
| 105 | 111 | 112 | 106 |
| 106 | 112 | 113 | 107 |
| 107 | 113 | 114 | 108 |
| 109 | 115 | 116 | 110 |
| 110 | 116 | 117 | 111 |

summarized in Fig.14 with the Box Numbers 3, 4, 5. After the input data process is completed, file organization part of the module ORGANIZE is started (see Fig.14 and Boxes 6, 7, 8, 9, 10).

Module ORGANIZE generally performs two types of processes . These are data and mesh generation processes (Boxes 7, 9). All the choices of this module are included in ORGANIZATION-MENU as seen in Fig.15 (a) .

In order to perform the first three alternative choices in this module, DATOR uses the temporary file (see Fig.13). For the last case of organization process (choice " D ") , the user gives all the required data from the screen instantaneously during the running of the module ISOMESH.

Using the test data which is declared in Appendix A, the first alternative of the menu (see Fig.15 (a)) prepares the data file automatically in a few seconds. This data file is the first main part of the resulting input file of EPDAN as seen in Fig.16 . In order to complete the input file , the

third choice " C " must be used. This process is summarized in Fig.15 (b) . After this process , the resulting file is generated (see Fig.16 (b)) in the free format .

Screen oriented data preparation is started by the choice " B " in this menu (see Fig.15 (a)). For this case the necessary input steps are as follows :

(1) Input the title of the data file (see Appendix D).

(2) Enter the number of data type to put in order.

(3) Process the screen commands to put in order the data in the file interactively (see Fig.15 (c)).

For the third step, the user must follow the three types of input process for each data. These are explained as in the following :

(a) Enter the name of the data

(b) Define the location of the data in the input file . This is performed by the location pointers which are declared with the indexes I and J. The user can arrange his data file by using the indexes of a

J →

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|----|---------|---------|---------|--------|---------|----|----|----|
| 1 | INKAOT | | | | | | | |
| 2 | INANL | ITANL | | | | | | |
| 3 | I | J | NSABST | | | | | |
| 4 | DPHI | PHIMAX | | | | | | |
| 5 | IDINKF | IDINKV | IDINKR | IDINKS | | | | |
| 6 | SKFNULL | EM | PO | SK | | | | |
| 7 | IFLITY | | | | | | | |
| 8 | CEE | ENN | | | | | | |
| 9 | IMA | ILMA | IHALT | IHEPSI | IDENGE | | | |
| 10 | ITOP | IWRITEG | ZEND | | | | | |
| 11 | IX | IY | | | | | | |
| 12 | RB | ZB | RC | ZC | RD | ZD | RA | ZA |
| 13 | RADE | RADA | | | | | | |
| 14 | RK | | | | | | | |
| 15 | ZK | | | | | | | |
| 16 | LIMVER | ILAST1 | LIMVER1 | ILAST2 | LIMVER2 | | | |
| 17 | IECK | IECK2 | ANF | | | | | |

Figure 17. Order of the data names for screen - oriented file organization

```

TEST DATA FOR DIELESS EXTRUSION 5 X 9 ELEMENTS 29/10/1989
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
1 0 1 0 0 0 0 0 0 0
.025 2 0 0 0 0 0 0 0 0
1 1 1 1 0 0 0 0 0 0
240 210000 .3 .05 0 0 0 0 0 0
LUDW 0 0 0 0 0 0 0 0 0 0
704 .235 0 0 0 0 0 0 0 0
1 50 50 1 1 0 0 0 0 0
VIER 50 50 0 0 0 0 0 0 0 0
6 20 0 0 0 0 0 0 0 0
8.964 199.6 7.5 194.136 7.5 192.136 8.964 230 0 0
3 3 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
2 0 0 0 0 0 0 0 0 0
4 0 0 0 0 0 0 0 0 0
6 0 0 0 0 0 0 0 0 0
8 0 0 0 0 0 0 0 0 0
8.964 0 0 0 0 0 0 0 0 0
210.4 0 0 0 0 0 0 0 0 0
209.8 0 0 0 0 0 0 0 0 0
209.2 0 0 0 0 0 0 0 0 0
208.6 0 0 0 0 0 0 0 0 0
208 0 0 0 0 0 0 0 0 0
207.4 0 0 0 0 0 0 0 0 0
206.8 0 0 0 0 0 0 0 0 0
206.2 0 0 0 0 0 0 0 0 0
205.6 0 0 0 0 0 0 0 0 0
205 0 0 0 0 0 0 0 0 0
204.4 0 0 0 0 0 0 0 0 0
203.8 0 0 0 0 0 0 0 0 0
203.2 0 0 0 0 0 0 0 0 0
202.6 0 0 0 0 0 0 0 0 0
202 0 0 0 0 0 0 0 0 0
201.4 0 0 0 0 0 0 0 0 0
200.8 0 0 0 0 0 0 0 0 0
200.2 0 0 0 0 0 0 0 0 0
199.6 0 0 0 0 0 0 0 0 0
199 0 0 0 0 0 0 0 0 0
2 2 1 3 1 0 0 0 0 0
6 12 -.04 0 0 0 0 0 0 0

TEST DATA 29/10/1989 UNFORMATTED AND SCREEN ORIENTED CASE
0 1 0 1 0 0 0 0 0 0
.025 2 0 0 0 0 0 0 0 0
1 1 240 210000 .3 0 0 0 0 0
.05 1 1 6 -.04 0 0 0 0 0
704 .235 0 0 0 0 0 0 0 0
VIER 0 0 0 0 0 0 0 0 0 0
LUDW 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
2 0 0 0 0 0 0 0 0 0
4 0 0 0 0 0 0 0 0 0
6 0 0 0 0 0 0 0 0 0
8 0 0 0 0 0 0 0 0 0
8.964 0 0 0 0 0 0 0 0 0

TEST DATA 29/10/1989 FORMATTED AND SCREEN ORIENTED CASE
0 1 0 1
0.02500 2.00000
1 1 240.000 210000.0 0.3000
0.0500 1 1 6 -.04000
704.000 0.235
VIER
LUDW
0.0000
2.0000
4.0000
6.0000
8.0000
8.9640

```

Figure 18. Test data file
(a) of EPDAN for unformatted case
(b) for unformatted case
(c) for formatted case

matrix. For example the test data of EPDAN (see Fig.(a)) has the data name order as shown in Fig.17 .

(c) Use the function-Menu . Each screen process can be repeated from the beginning or for the same data by using this menu. The third possibility is the stopping of the process at the end of each screen process. The user must use the enter key in order to continue its' process.

EPDAN has fifty-three types of data names. Some of them are arranged in the matrix form as shown in Fig.17 for the test data which is given in Appendix A.

FORMAT-Menu is used after the completion of data arranging process (see Fig.15 (d)). If all the data of EPDAN is entered from the screen the formatted case prepares the file as shown in Fig.16 (a) . Unformatted file organization choice prepares the file as shown in Fig.18 (a). By changing the orientation of the names of the data , different types of data files can be created. For an example , 21 types of data of EPDAN are arranged in a different orientation

as seen in Fig.18 (b) for unformatted case and formatted case as in Fig.18 (c) . For this example , the order of the data is given in Figure 19.

| | | | | | | |
|-----|---|--------|--------|---------|--------|-----|
| | | J → | | | | |
| | | 1 | 2 | 3 | 4 | 5 |
| I ↓ | 1 | INKAOT | I | J | NSABST | |
| | 2 | DPHI | PHIMAX | | | |
| | 3 | IDINKF | IDINKV | SKFNULL | EM | PO |
| | 4 | SK | IDINKR | IDINKS | IECK | ANF |
| | 5 | CEE | ENN | | | |
| | 6 | ITOP | | | | |
| | 7 | IFLITY | | | | |
| | 8 | RK | | | | |

Figure 19. Order of data names for the sample data

Isoparametric mesh data preparation has the following commands on the screen :

- (1) Start the program ISOMESH.
- (2) Enter the number of superelement domains in the whole domain.

(3) For each superelement mesh generation procedure user must follow the following commands orderly .

(a) enter the number of elements in the X-direction ,

(b) enter the number of elements in the Y-direction,

(c) enter master nodal coordinates KSI and ETA with respect to the X and Y coordinate system of the whole domain (see Fig.20),

(d) input the first starting node number of this domain with respect to eight nodes isoparametric element.

(4) Change the ranges of the screen to plot the mesh for different dimensions.

The third case is repeated according to the number of superelements in the whole domain. The first starting node number must be determined correctly. If one of these numbers is given incorrectly, the user can notice this situation on the plotted mesh. Several types of isoparametric mesh data for EPDAN and their

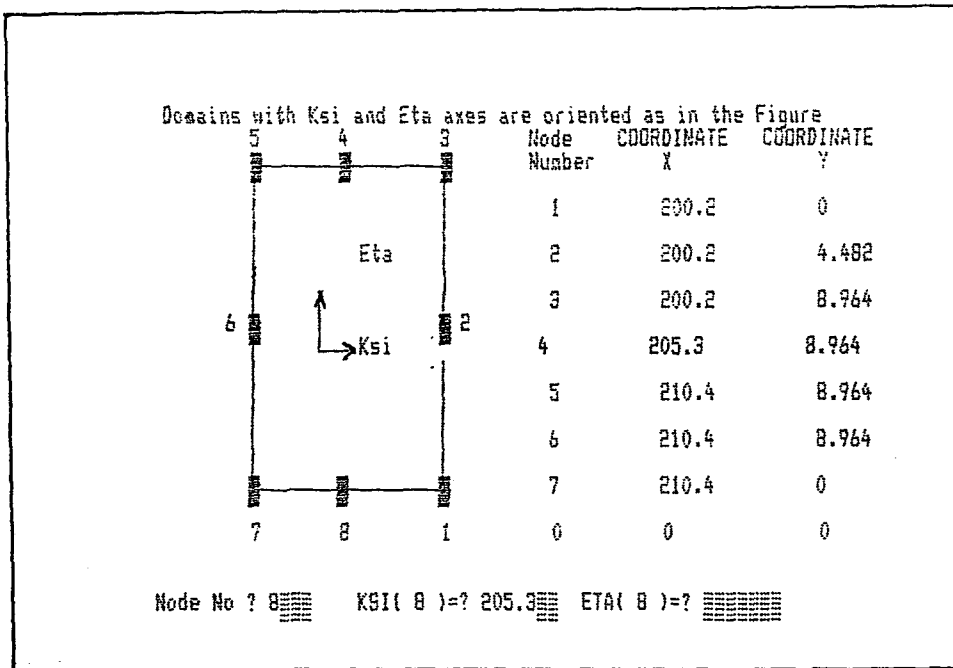


Figure 20. Isoparametric master element and their coordinates

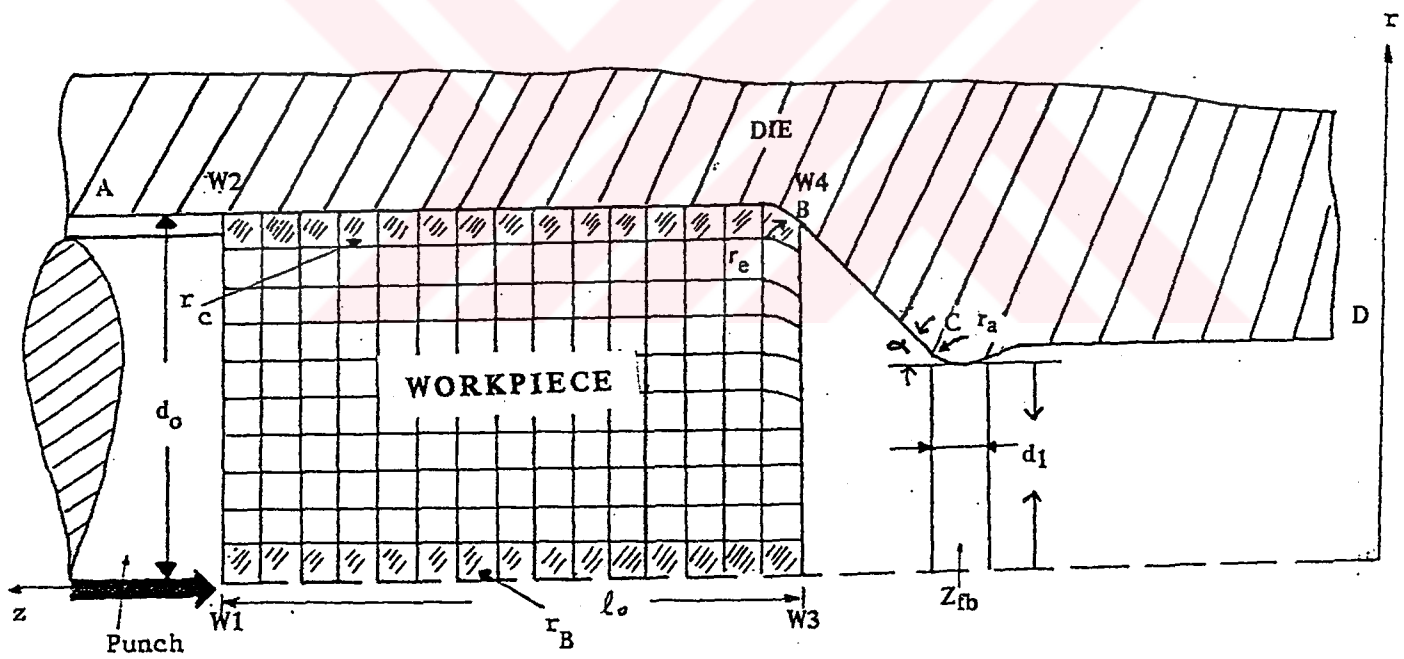


Figure 21. Extrusion process by EPDAN

corresponding figures are given in Appendix G.

6.2. Test Run for the Simulation of Metal Forming Processes

Extrusion is one of the most important metal forming process. For fifty increments, the results which are obtained by EPDAN are plotted (see Fig.21). The test data is given in Appendices A.

For the given data, the initial force value is found as 5.1 KN. and after fifty increments are completed, the punch is pushed 2 mm. Then the value of the force reaches to 29.6 KN (see Fig.22). The slope of the force displacement graphic depends on the internal pressure [5,6]. Coefficient of friction is taken as constant for the simulation of EPDAN. Results which are obtained by the finite element method have the numerical vibrations as seen in Fig.22. The range of the vibration can be explained by the number of mesh elements which are used. Because of the using method in use, seperation of the volume into a number of elements causes the seperation in the boundary

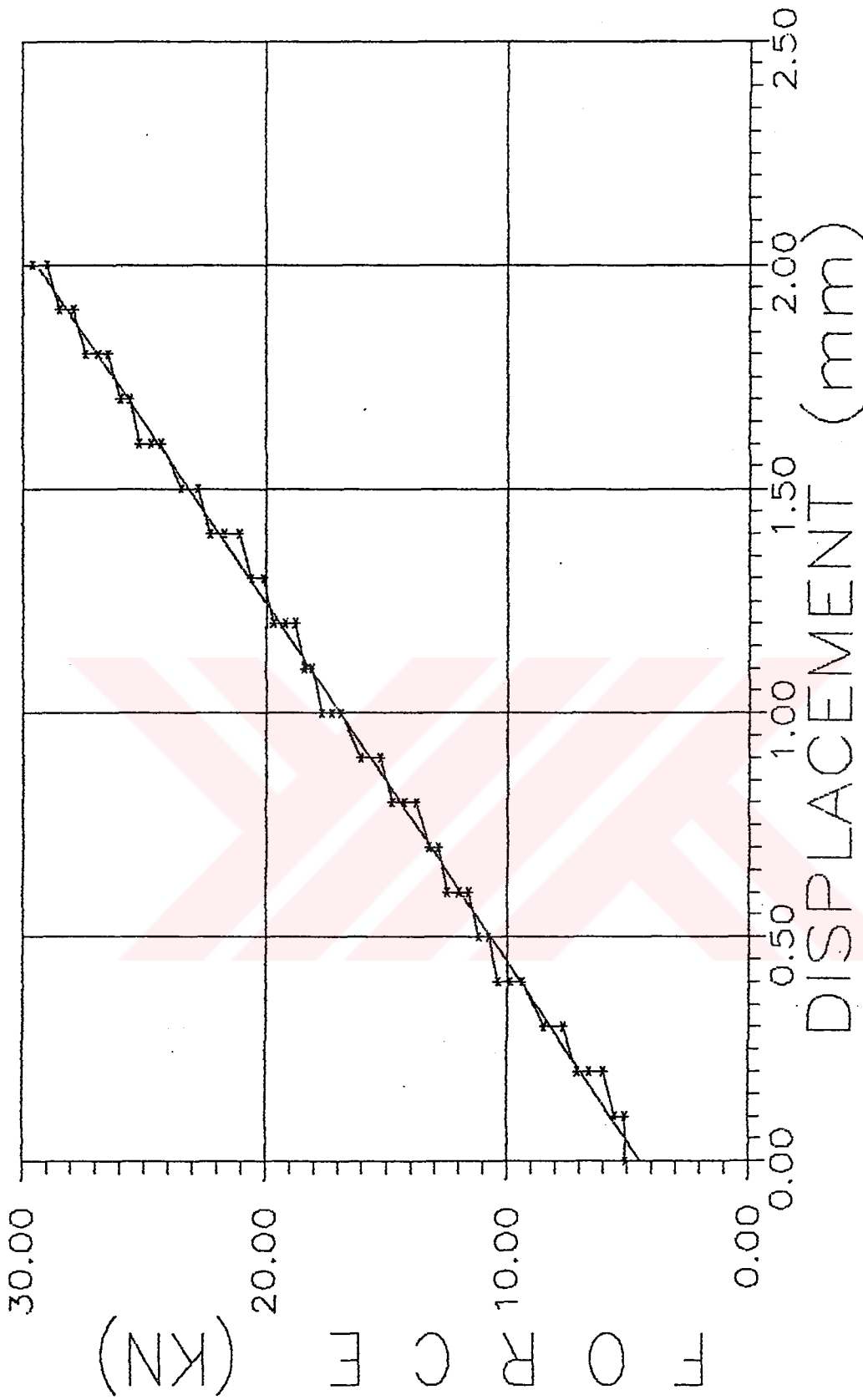


Figure 22. Force versus displacement plot

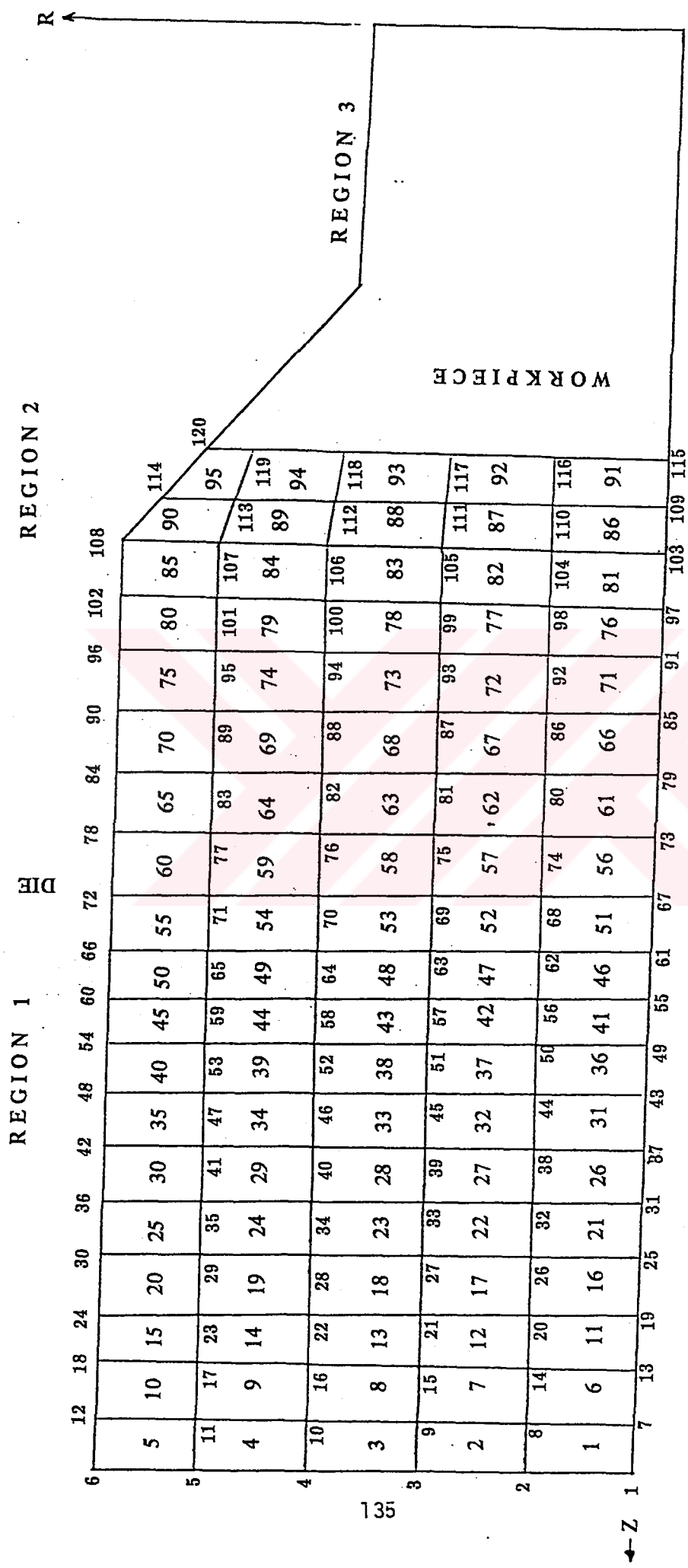


Figure 23. Element node numbering on the workpiece for test data of EPDAN

conditions. So the period of the numerical vibrations corresponds to the wideness of the mesh elements. Consequently, vibrations can be decreased by increasing the number of elements in the mesh [5,6].

The computed radial and axial force values are plotted with respect to the node numbers (see Fig. 23, Fig.24, Fig.25). Maximum axial force along the Z-direction is located on the node numbers one to six. Node number four carries the maximum axial force with the value -8426.23 N. Considering the radial force verses node numbers graphic Fig.24, the radial forces have zero values at some nodes. Because these nodes are located on the edge of the workpiece where the punch is pushing.

Nodes which are contacted with the die have the axial and radial force values as seen in Fig.23, Fig.24, Fig.25. Other nodes have zero axial and radial forces. Nodes 102, 108 and 114 which are located on the transition region are loaded with the second maximum axial and radial force region.

Stress distributions, which are shown in

Fig.26, Fig.27. and Fig.28) are plotted for the axial distance of the workpiece. High magnitudes of axial compressive stresses occur in the core of the workpiece (see Fig.26) . Surface layers of the workpiece which touch the die have tensile radial stresses. In the same region in the core, compressive stresses are formed . This region is located between the distances 200.4 mm and 202.0 mm (see Fig.27). In Fig.28 tangential stresses in the surface of the workpiece are tensile while they are compressive in the core between the distances 200.3 mm and 201.5 mm in Fig. 28. Radial and tangential stresses for the core and surface regions have the same values at the end of the workpiece.

Cold extrusion processes are used by Tekkaya [4,5,6] in order to control the quality of products by the determination of residual stresses and these stresses have values much larger than the yield strength of the material.

6.3. Test Run for Isoparametric Mesh Generation Program

Application results of the module ISOMESH for the solution of EPDAN are plotted and they are shown in Figures 29, 30, 31 and 32. For fifty incremental solution, test data which is given in Appendix A is used. For the test of isoparametric mesh program, instead of quadrilateral mesh data, isoparametric mesh data is used.

Isoparametric mesh data is generated with DATOR according to the given coordinates of the die for two regions (see Fig.23). For the first region 85 and for the second region of the workpiece 10 mesh elements are generated separately. Element numbering direction is chosen as R and node numbering direction is specified in the counterclockwise direction. These are the basic characteristics of the mesh data which are prepared automatically by the module MESH. For test data R and Z coordinates of the nodes are generated with equally prepared spaces. But in the original test data of EPDAN, one transition region is

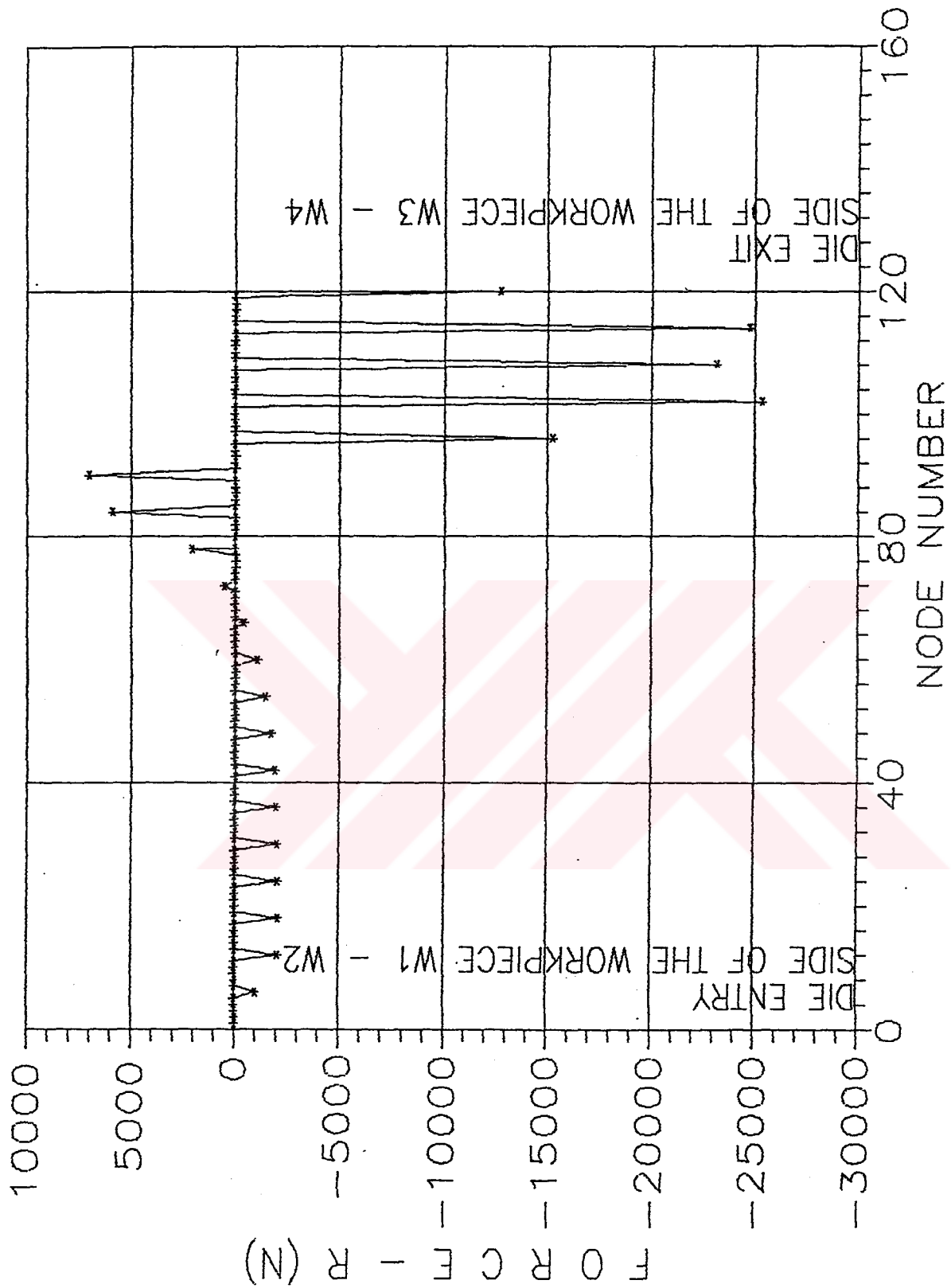


Figure 24. Radial force versus nodes plot

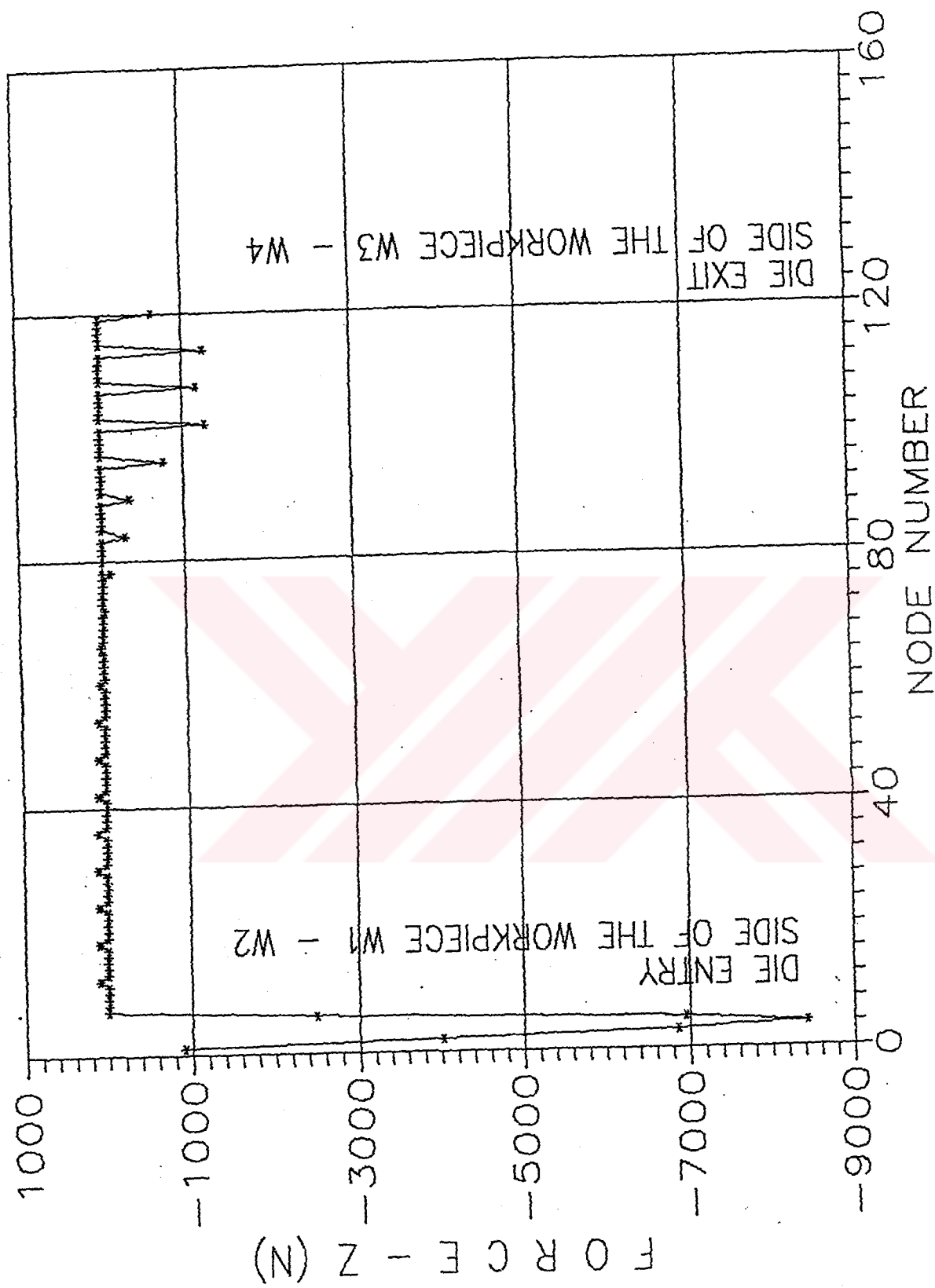


Figure 25. Axial force versus nodes plot

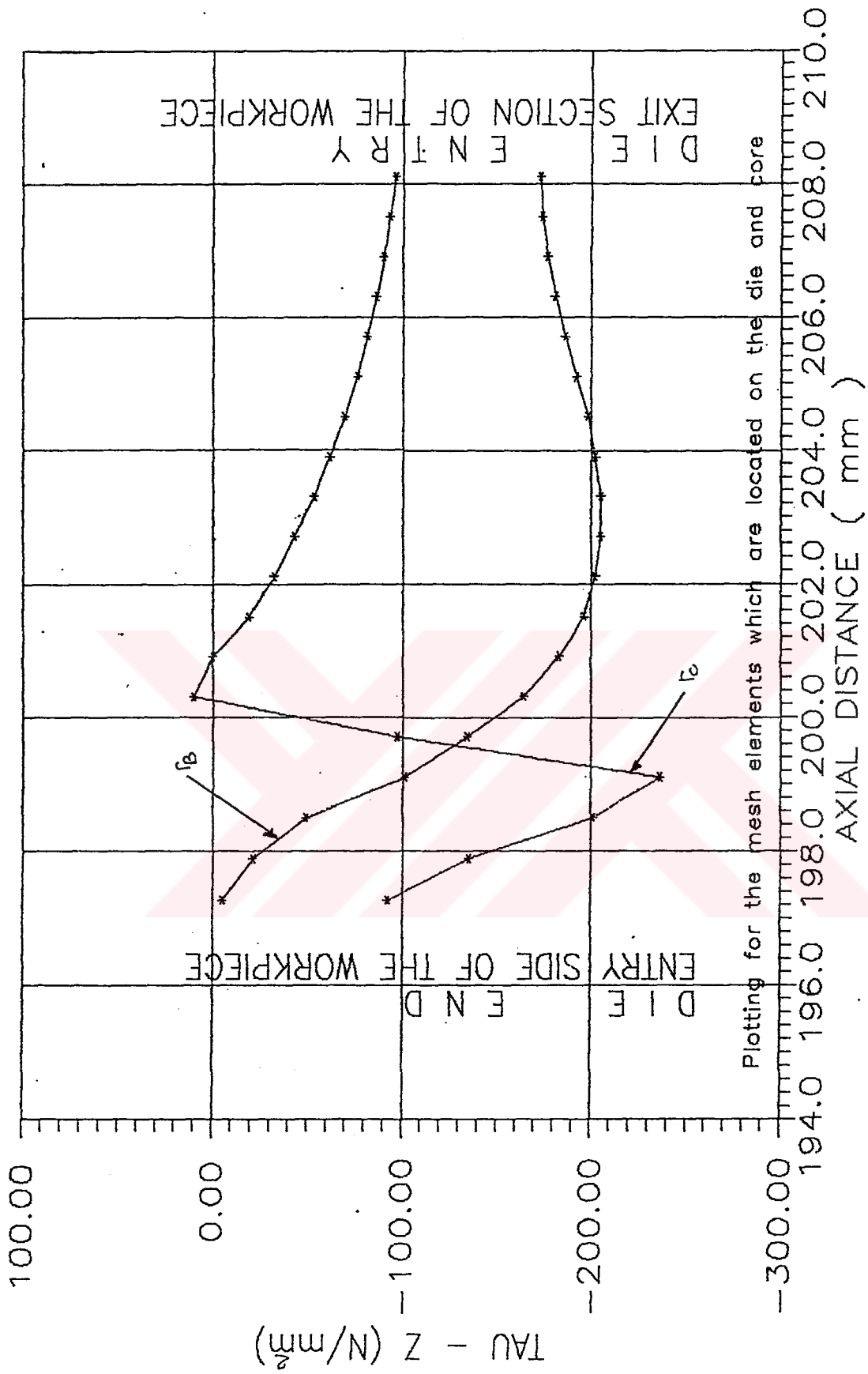


Figure 26. Axial stresses versus axial distance plot

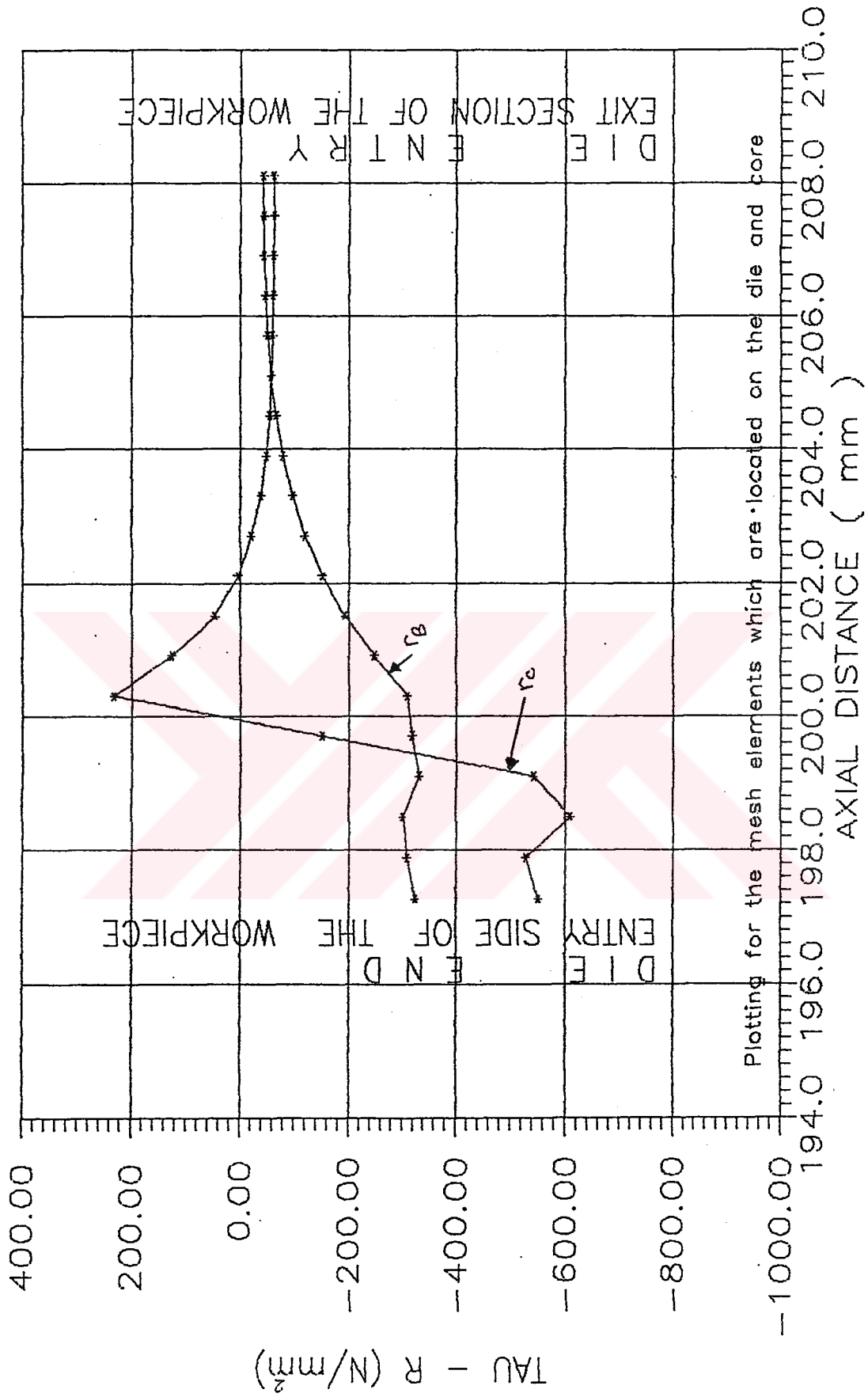


Figure 27. Radial stresses versus axial distance plot

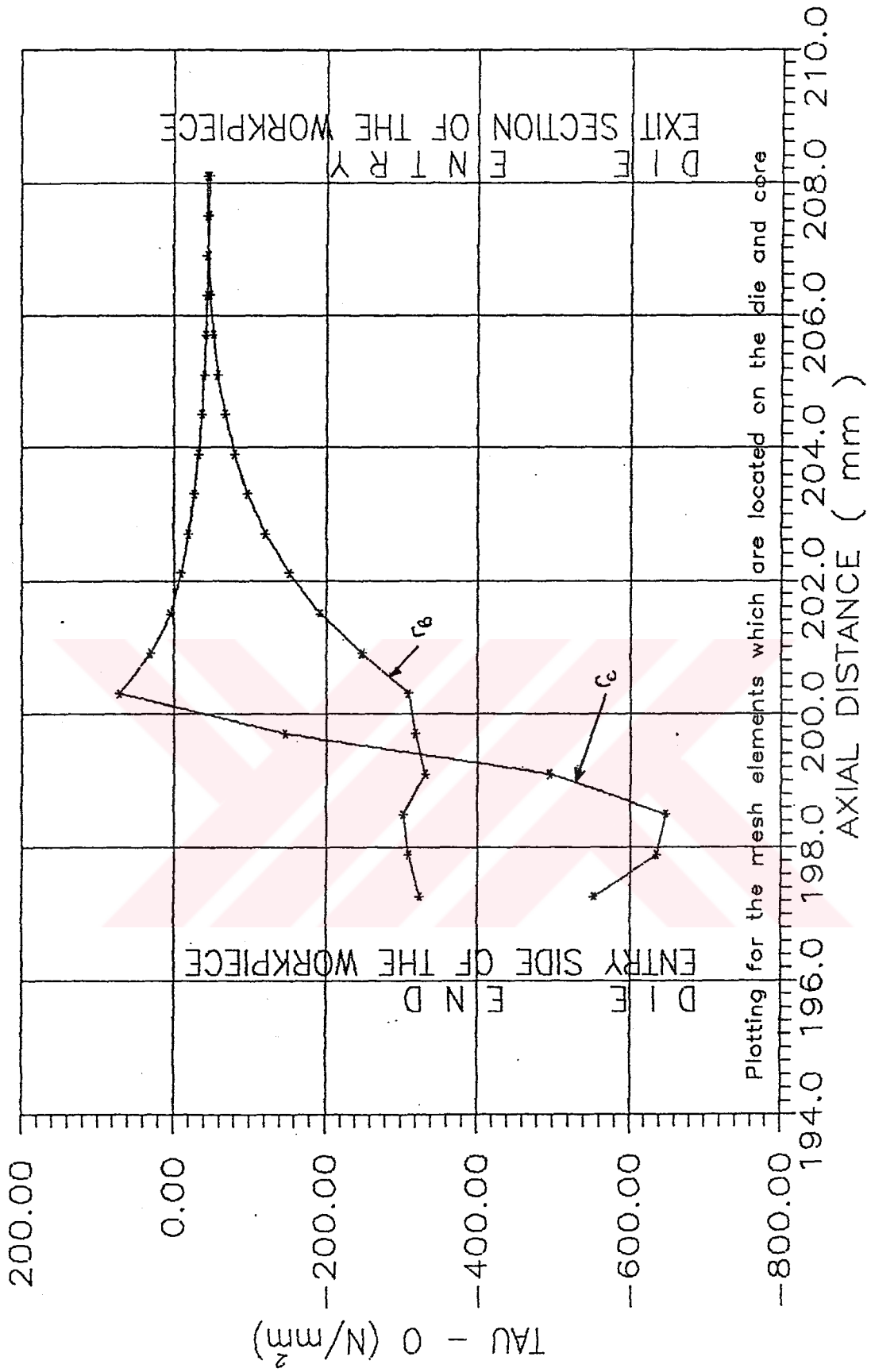


Figure 28. Tangential stresses versus axial distance plot

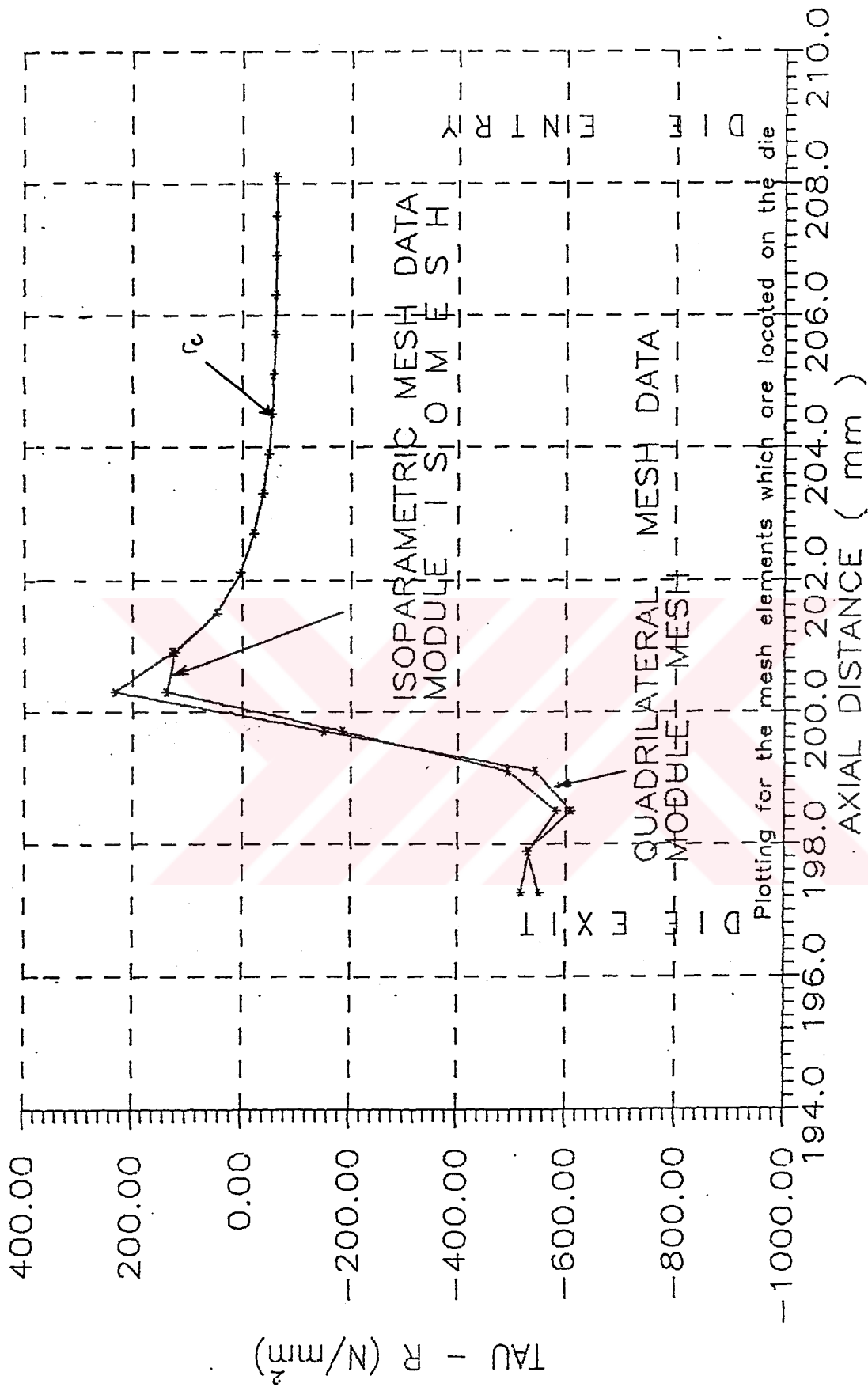


Figure 29. Comparison of two types of mesh data with radial stresses

a) for the elements which are touching the die

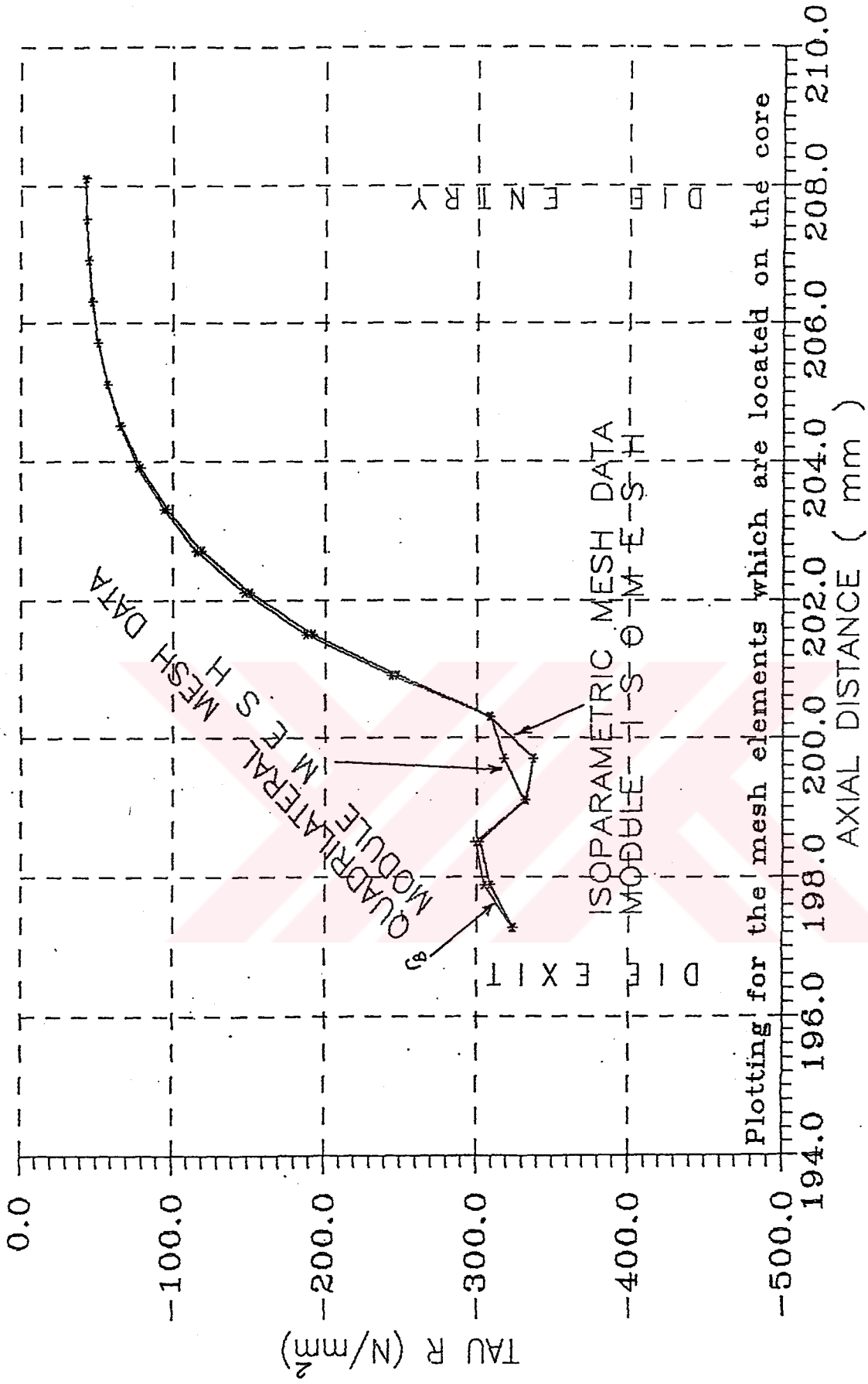


Figure 29. Comparison of two types of mesh data with radial stresses.

b) for the elements which are located on the core

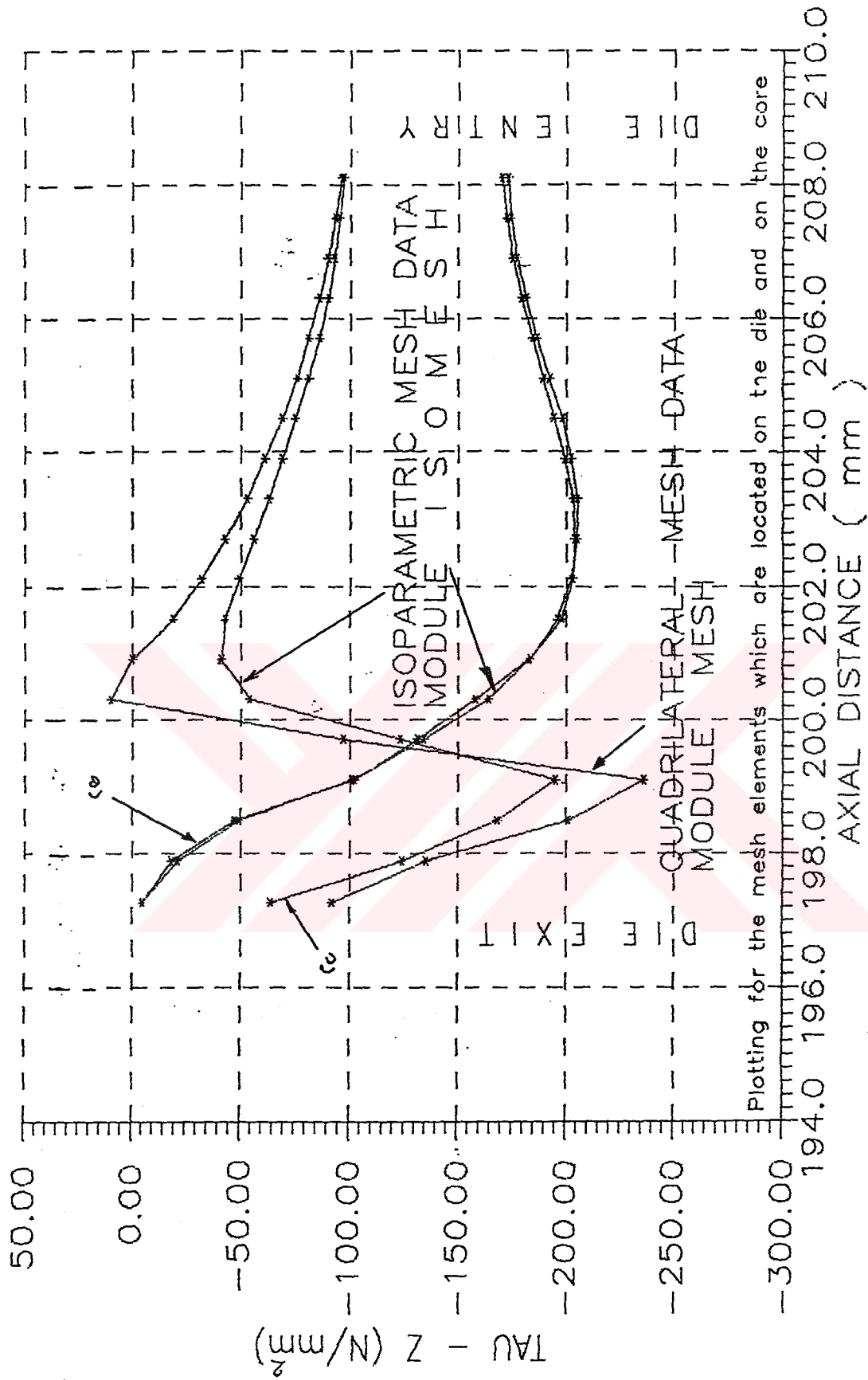


Figure 30. Comparison of two different types of mesh data with axial stresses

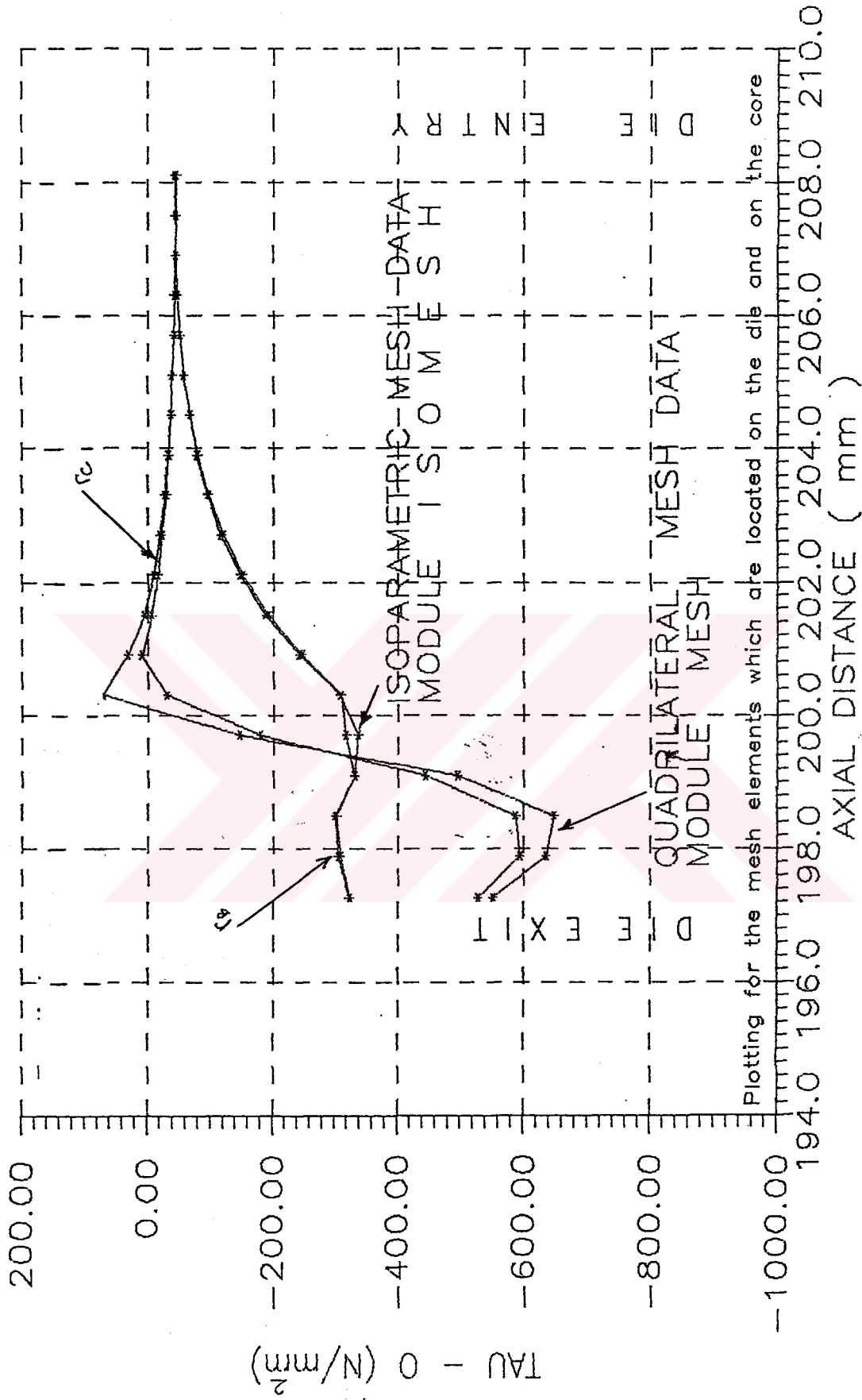


Figure 31. Comparison of two different types of mesh data with tangential stresses.

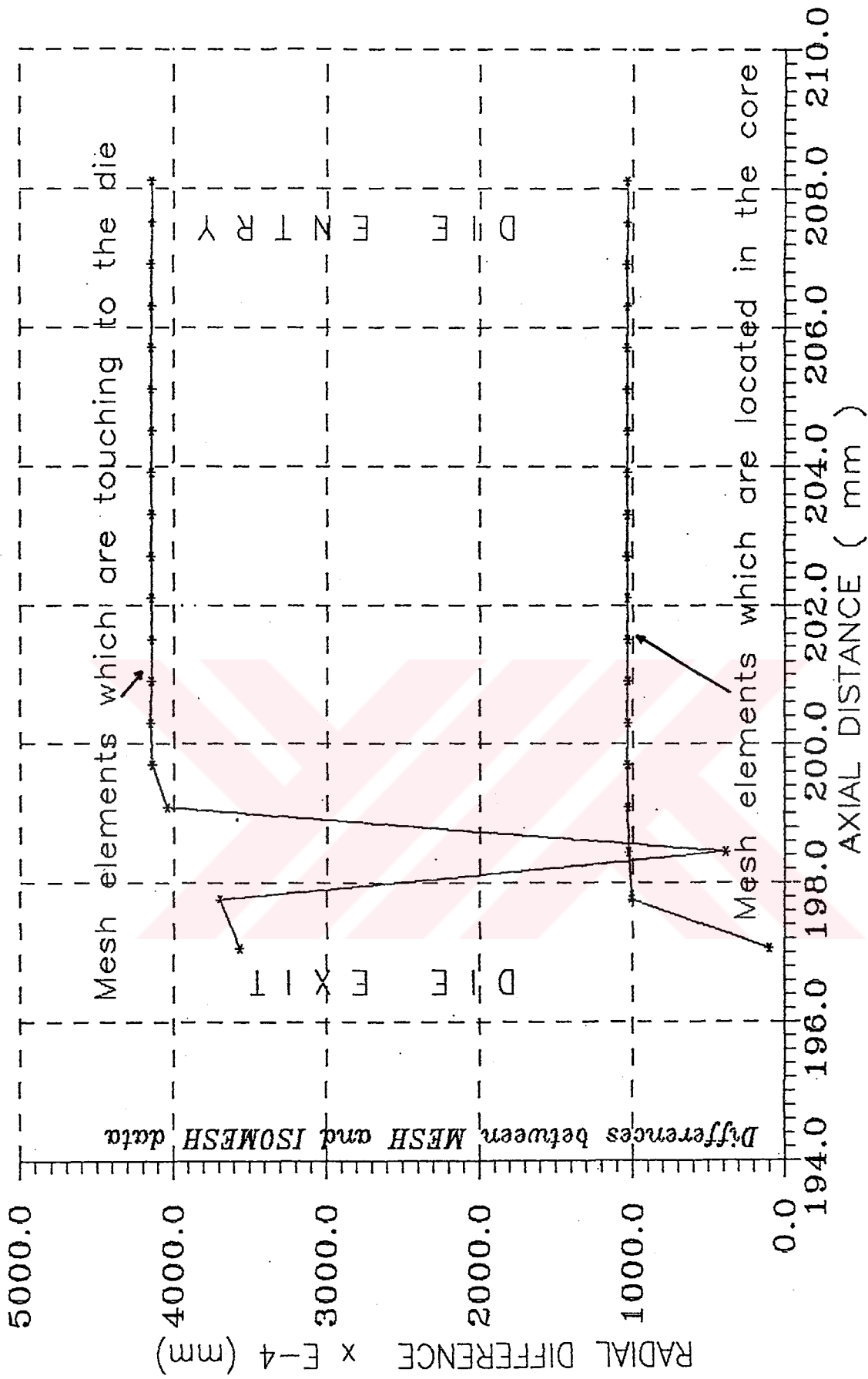


Figure 32. Radial difference versus axial distance plot

specified for the elements which are touching the die. In this region, R coordinates of the center lines of the elements are 0.41 mm smaller than the isoparametric mesh data of EPDAN. Differences in R coordinates for the elements touching to the die and in the core are plotted as shown in Figure 32. Maximum differences in the core are appeared in the elements with numbers 1, 6, 11, 16, 21, 26, 31, 36, 41 and 51. The determined difference is 0.1036 mm. For the elements 56, 61, 66, 71 the difference is 0.1035. Minimum difference is seen in the element with number 91. According to these differences stress distributions show discriminations. Maximum difference in the r_e elements which are touching the die occurs in the element 70. So for this element radial stress is equal to 232 N/mm^2 and 137 N/mm^2 respectively for the MESH and ISOMESH data (see Fig.29). As seen in Figure 30 mesh elements in the core have the same axial stress values because of the small magnitudes of the differences in the radial direction.

Consequently except the radial difference effects , the same stress distributions are determined for the mesh elements of the workpiece.



CHAPTER 7

CONCLUSIONS AND SUMMARY

7.1. Discussion

Finite element programs with preprocessors are summarized in Appendix C. Interactive programs and and interactive preprocessors together with the finite element method are explained in Chapter 2. The interactive computer code DATOR is implemented by utilizing the knowledge and procedures given in Chapter 2 and Appendix C. The description of the preprocessor and data processing facilities is explained in Chapters 5 and 6. As mentioned before, modular programming is the best method to implement the large programs. Modules of the interactive preprocessor are explained in Figure 8. In the same way, menus and their procedures are explained in Chapter 5 with Figures 8, 15 and with the menu list of DATOR in Appendix D. There are two ways in order to combine the modules. These are the usage of subroutine CALL statements and CHAIN commands. In this code ,

CHAIN commands are used in order to determine the required fastest execution and usage of the parameters globally. In each module, using parameters has only one kind of process and this gives the user the easiest control for the execution of the program. All function and sub menus are prepared by this method. The preparation of data files is explained in Chapters 5 and 6. The user of EPDAN can prepare the input file in 15 seconds automatically if the temporary file is prepared initially. This temporary file (see Figure 13) can be changed any time by using the menus . After this step , resulting formatted input file is prepared correctly . Besides this facility , different types of formatted and unformatted files can be prepared according to the user's needs . These files are explained in Chapter 6 with Figures 15, 16, 17, 18, 19. Figure 14 expresses the order to prepare an input file completely.

Interactive preprocessor is written in BASIC. It has some disadvantages in usage. Finite element code EPDAN has been implemented with FORTRAN 77 language.

But its' preprocessor has a different computer language. So there is not any continuous flow of execution between these two programs. In order to start EPDAN after the preparation of input file a different job is required. Programs which are written in Quick BASIC have some difficulties. One of them is the limited capacity of the data and instruction memory areas. These areas are limited with 64 K bytes. This amount is insufficient for large programs such as DATOR. Because of this limited capacity of the memory areas, dimensions of the arrays and number of using parameters creates undesirable effects. Dimensions which are opened as dynamic, cover more space than static dimensions. As a result of this limitation, in module ISOMESH, locally using arrays and matrices is opened in the static form (see Appendix F). The third disadvantage appears in the assignment statements. For example, if the parameter E is assigned to the different numbers in the same module, parameters are mixed and incorrect results appears at the end of the execution. The fourth

disadvantage is seen in the preparation of data file because of the types of the numbers. If real numbers with one digit and zero decimal point are input, for the printed case the integer number is determined. Such as the real number 1.0 is printed as 1 in the data file. Real type READ statements of EPDAN cannot read the integer type data. There are only two methods to overcome this difficulty. The first one is the replacement of the all integer and real type input data commands by character type data inputs. As a result of this process, data values can be printed to the data file with their correct decimal points. But in this case the preparation of different types of formatted files is impossible. The second method is to use free format input statements for the real type of data in FORTRAN program. The fifth disadvantage appears during the loading of the code into the memory space of GW-BASIC. All modules with BAS extension cannot be loaded into the memory at the same time. In order to make this possible, modules which have smaller than 64 K bytes memory are loaded. After the

usage and compilation steps, the module is removed and another one is copied from the disk. Besides these disadvantages, BASIC is the more powerful language to write an interactive program with graphic capabilities.

Mesh generation procedures are explained in Chapter 2. Isoparametric mesh generation program ISOMESH is based on the isoparametric coordinate system denoted by Zienkiewicz [26]. The whole region is divided into superelements and for each superelement four different types of data are entered. These are number of elements in X and Y directions, element and node numbering direction, eight coordinates of the superelement and starting node number. These are explained in Figures 11, 12 and 20. At the end of the process, module ISOMESH converts the isoparametric element and node correspondance to the quadrilateral mesh elements for the three regions of workpiece. The program has some limitations. The first one is the limited conversion region which is declared as three for the workpiece of EPDAN. The second

limitation can be seen because of the limited capacity of data space of BASIC. This program is written in BASIC because it is included into the interactive BASIC code DATOR. The program can generate 500 mesh elements. As mentioned before, this number can be increased by decreasing the dynamic arrays in the software (see Appendix G). The last one can be seen when the shape of the workpiece has discontinuities.

Methods in order to determine an increase for the speed of EPDAN throughout the modification are explained in Chapter 4. Modifications are explained with the flow-chart in Figures 6 and 7. Increase in speed is obtained by using the COMMON blocks and dynamic array dimensioning in the whole program. As a result of modifications three minutes are gained from the total execution time (see Table 1 and 2).

Results of EPDAN are plotted in Chapter 6. Discretization property of finite element method causes the vibrations in the force - displacement plot (see Fig.22). EPDAN does not use a constant coefficient of friction and slope of this graphic

depends on the internal pressure. All these properties affect the results of the graphics. Element node connection on the workpiece for the test data is denoted in Figure 23. According to the Figures 24, 25 nodes with numbers 84 and 90 have the maximum tensile forces and nodes 96, 102, 108, 114 and 120 touching to the die have the maximum compressive radial forces. Maximum tensile radial , tangential and axial stresses are located on the elements with numbers 65 and 70 as shown in Fig.26, Fig.27, Fig.28 . Maximum compressive stresses are established on the elements with numbers 66, 71, 76 . These elements are located in the core of the workpiece (see Fig. 26, 27, 28).

Isoparametric mesh generation data results are compared with the quadrilateral mesh data in Chapter 6 with Figures 29, 30, 31 and 32. Original mesh generation code MESH has a transition region between the die and workpiece for the elements with numbers 5,10 .. to 95 (see Fig.23). Module ISOMESH data does not have any transition region . Radial coordinate

differences between the two mesh data are 0.41 mm and 0.1036 mm for the r_c and r_B mesh elements respectively. Maximum coordinate difference is seen on the element with number 70. As a result of this radial difference, stress distributions on the element 70 reaches to maximum as seen in Figures 29, 30, 31. Radial differences of the mesh elements for two different types of mesh data are plotted in Figure 32.

7.2. Conclusions

In this study, interactive preprocessor computer code is implemented for the finite element code EPDAN. In addition to this study, modification of this code is performed in order to obtain an increase in the speed of the program by using dynamic dimensioning and COMMON blocks for IBM microcomputers.

The preprocessor DATOR consists of six main modules. All modules are interactive. This opportunity gives the user a large flexibility in order to prepare special and different types of programs. Modules are

compiled and linking between them is satisfied by CHAIN commands in the shortest time. Each module is developed separately so that each one is developed independently from each other. Interactive data input facilities are performed with the help of menus. Error messages are given for the special cases in order to stimulate the user. For each type of data, a separate menu is prepared. The main program or module MAIN contains MAIN MENU and this menu combine all the other five modules here. All arrays in the modules are organized according to the dynamic dimensioning concept. As a result of this application, dimensions can be changed interactively in an automatic procedure. Organization of the input files and generation of mesh data for quadrilateral or isoparametric cases are controlled in the module ORGANIZE. Modules FIRST and MODIFY are prepared according to the needs of the user of EPDAN. These modules include the data types of EPDAN. Besides, the same code can be used with the other finite element programs by changing the

declaration statements and data names in the software.

Plot results of the isoparametric mesh generator program ISOMESH show that it is a more practical one. Number of elements, node numbering direction, shape of the workpiece, coordinates of the superelements can be changed interactively. The program sketches the generated mesh on the graphic screen in order to check the results. Original mesh program MESH generates the mesh data for quadrilateral elements automatically. Many finite element programs have not their special interactive preprocessors as seen in Appendix C. Code DATOR covers the interactive preprocessor deficiency of the finite element program EPDAN.

Restructured code of EPDAN permits the users for further improvements on the code easily. Dynamic dimensioning reduces the execution time and gives the user an automatic control of the dimensions of the arrays. In order to increase the speed of the code, ninety-five subroutine CALL parameters are arranged in a single COMMON block. Main program is formed in terms of subroutine CALL statements. Moreover,

dynamic array dimensioning is used in the code.

Consequently, gaining time is three minutes .



LIST OF REFERENCES

1. Mendelson, A., 1968, Plasticity : Theory and Applications , The Macmillan Company , New York.
2. Hibbit, H.D., Marcal P.V. , Rice, J.R. , 1970 , " A Finite Element Formulation for Problems of Large Strain and Large Displacement " , Int. J. Solids Struc. , Vol.6, pp.1069-1086 .
3. Cheng, J., 1988, "Automatic Adaptive Remeshing for Finite Element Simulation of Forming Process " , International Journal For Numerical Methods In Engineering, Vol.26, pp.1-18 .
4. Tekkaya , A.E. , Oct. 1987, "Finite Element Simulation of Metal Forming Processes " , AGARD Conference Proceedings, No.CP-426 on Aerospace Materials Process Modelling , Pub. AGARD-NATO, Paper No.15, pp.15-1/15-13 .
5. Tekkaya ,A.E. , 1986, "Metal Şekillendirmenin Sonlu Elaman Yöntemiyle Sayısal Benzetimi" , Makina Tasarım ve İmalat Dergisi, Vol.1, No.1, pp.14-24.
6. Tekkaya , A.E, 1986, Ermittlung von Eigenspannungen in der Kaltmassivumformung, Springer-Verlag,

Berlin .

7. Hartley , P.P. , Sturgess, C.E.N. , Rowe, G.W. , 1985, "An Elastic-Plastic Three Dimensional Finite Element Analysis of the Upsetting of Rectangular Blocks and Experimental Comparison", Int. J. Mach. Tool. Des. Res. , Vol.25, pp.229-243 .
8. Needleman, A. , 1972, "A Numerical Study of Necking in Circular Cylinder Bars", J. Mech. Phys. Solids , Vol.20, pp.111-127 .
9. Hill , R. , 1959, " Some Basic Principles in the Mechanics of Solids Without a Nature Time ", J. Mech. Phys. Solids, Vol.7 , pp.209 .
10. McMeeking , R.M. , Rice , J.R. , 1975 , "Finite Element Formulations for Problems of Large Elastic-Plastic Deformation ", Int. J. Solids Structures, pp.601-616 .
11. Zienkiewicz , O.C. , 1977 , The Finite Element Method , 3rd Ed. , McGraw-Hill , London .
12. Fellippa , G.A. , Sharifi , P. , 1973 , "Computer Implementation of Nonlinear Finite Element Analysis in Numerical Solution of Nonlinear Structural

- Problems " , ASME , pp.31 .
13. Yagmai , S. , Popov , E.P. , 1971 , "Incremental Analysis of Large Deflection of Shells of Revolution " , Int. J. Solids Struc. , Vol.7 , pp.1375-1393 .
14. Sharifi , P. , Popov , E.P. , 1973 , "Nonlinear Finite Element Analysis of Sandwich Shells of Revolution" , AIAA Journal , Vol.11 , pp.715-722 .
15. Malvern , L.E. , 1969 , Introduction to the Mechanics of A Continuous Medium , Prentice-Hall. , New Jersey .
16. Mano , M.M. , 1982 , Computer System Architecture , Prentice Hall . Inc. , Canada .
17. Mackerle , J. , 1986 , "Finite Element Codes for Microcomputers - A Review " , Computers & Structures , Vol.24 , No.4. , pp.657-682 .
18. Mackerle , J. , 1986 , "Makebase , An On-Line Information Retrieval System for Structural Mechanics " , Computers & Structures , Vol.24. No.6, pp.977-983 .
19. Taniguchi , T. , 1988 , "An Interactive Automatic

- Mesh Generator for the Microcomputers" , Computers & Structures , Vol.30. No.3. , pp.715-722 .
- 20.Mackerle , J. , 1988 , "Finite Element Codes for Microcomputers - An Addendum", Computers & Structures , Vol.28. No.6 , pp.797-814 .
- 21.Noor , A.K. , 1981 , "Survey of Computer Programs for Solution of Nonlinear Structural and Solid Mechanics Problems", Computers & Structures , Vol.13. , pp.425-465 .
- 22.Saouma , V.E. , Sikiotis , E.S. , 1985 , "Interactive Graphics Nonlinear Constrained Optimization", Computers & Structures , Vol.21 , No.4 , pp.759-769 .
- 23.Taniguchi , T. , 1987 , "Flexible Mesh Generator for Triangular and Quadrilateral Areas", Adv. Eng. Software , Vol.9. No.3 , pp.142-149 .
- 24.Ghassemi , F. , 1982 , "Automatic Mesh Generation Scheme for a Two or Three Dimensional Triangular Curved Surface", Computers & Structures, Vol.15, pp.613-626 .
- 25.Durocher , L.L. , 1979 , "A Versatile

- Two-Dimensional Mesh Generator With Automatic Bandwidth Reduction", Computers & Structures , Vol.10. pp.561-575 .
26. Zienkiewicz , O.C. , Phillips , D.V. , 1971 , "An Automatic Mesh Generation Scheme for Plane and Curved Surfaces by Isoparametric Coordinates", International Journal for Numerical Methods in Engineering, Vol.3 , pp.519-528 .
27. Suhara , J. , Fukuda , J. 1977 , Automatic Mesh Generation for Finite Element Analysis . In Advanced in Computational Methods in Structural Mechanics and Design . , UAH Press , Alabama , pp.607-627 .
28. Stefanu , G.D. , 1980 , "Automatic Triangular Mesh Generation in Flat Plates for Finite Elements", Computers & Structures, Vol.11 , pp.439-464 .
29. Field , D.A. , Matton , M.B. , 1985 , "Processing Three Dimensional Finite Element Assemblies and Solutions", Computers & Structures, Vol.21, No.4, pp.815-823 .
30. Buell , W.R. , Bush , B.A. , 1973 , "Mesh

Generation - A Survey ", Journal of Engineering for Industry , Tractions of the American Society of Mechanical Engineers , Series B , Vol.95 , No.1 , pp.332-338 .

31.Vallipan , S. , Murti , V. , 1986 , "Numerical Inverse Isoparametric Mapping in Remeshing and Nodal Contouring", Computers & Structures , Vol.22., No.6 , pp.1011-1021 .

32.Reddy , J.N. , 1984 , An Introduction to the Finite Element Method , McGraw-Hill , New York .

33.Tekkaya , A.E. , 1986 , Lecture Notes for the Introduction to the Continuum Mechanics .

34.Gallagher , R.H. , 1975 , Finite Element Analysis Fundamentals , Prentice-Hall , Englawood Cliffs , New Jersey .

35.Bathe , 1982 , Finite Element Procedures in Engineering Analysis , Prentice-Hall , Inc. , New Jersey .

36.IBM , 1984 , Professional Fortran Reference , Ryan-McFarland Corporation . .

37.IBM , 1984 , Professional Fortran Installation and

- Use , McFarland Corporation .
38. Organic , E.I. , 1966 , A FORTRAN IV Primer ,
Addison-Wesley Publishing Company , Canada .
39. Schonbeck , R.G. , 1968 , FORTRAN IV for Multi
Programming , Addison-Wesley Publishing Company ,
Canada .
40. RM/FORTRAN , 1986 , Quick Reference Card for DOS
Systems , RYAN-McFarland Corporation .
41. Brainerd , W. , 1978 , "FORTRAN 77" ,
Communications of the ACM , PP.806-820 .
42. IBM , Quick Basic Reference , Ryan-McFarland
Corporation .
43. Cheung , Y.K. , Yeo , M.F. , 1979 , A Practical
Introduction to Finite Element Analysis , PITMAN
Publishing Limited , London .
44. James, M.L., Smith, G.M., Wolford, J.C., 1977,
Applied Numerical Methods for Digital Computation
with FORTRAN and CSMP , Harper and Row Publishers ,
Inc. , New York .
45. Tekkaya , A.E. , 1986 , Lecture Notes for the
Finite Element Method .

46.Keskinel , F. , 1987 , BASIC , 2nd Ed. , Birsen
Yayınevi , İstanbul .

47.Yalçıner , U. , 1985 , FORTRAN IV , BASIC , Teori
Yayınevi , Ankara .



Appendix A

INPUT DATA CARDS AND TEST DATA FOR
EPDAN

1. IA(72) : Title of the process.

("TEST DATA FOR DIELESS EXTRUSION 5X19 ELEMENTS
20/10/1989")
2. INKAOT : Number of the extremely deformed elements.

INKAOT = 0
3. NKAOT(N): If number of extremely deformed

elements is not equal to zero, number

of the elements.

NKAOT is omitted.
4. INANL : Number of the nodes which may hit the die

INANL = 0
5. ITANL : Number of additional iteration for

the increment after the hit.

ITANL = 0
6. NANL(N) : If number of nodes which may hit the die

is not equal to zero, node numbers .

NANL(N) is omitted.
7. I : Restore-flag (Restore = 1).

I = 1

8. J : Restart-flag (Restart = 1).

J = 0

9. NSABST : Increment for restoring.

NSABST = 1

10. DPHI : Increment of strain.

DPHI = 0.025

11. PHIMAX : Maximum strain.

PHIMAX = 2.0

12. IDINKF : Force increment.

IDINKF : 1

13. IDINKV : Increment for the boundary conditions.

IDINKV = 1

14. IDINKR : Increment for the displacements.

IDINKR = 1

15. IDINKS : Increment for the stresses.

IDINKS = 1

16. SKFNULL : Initial flow stress.

SKFNULL = 240. N/mm²

17. EM : Young's modulus.

EM = 210000 N/mm²

18.PO : Poisson's ratio.

PO = 0.3

19.SK : Coefficient of friction.

SK = 0.05

20.IFLITY : Keyword for the flow curve type,
(s = C*E**N).

IFLITY = "LUDW"

21.CEE : Constant (If IFLITY is equal to ILUDW).

CEE = 704

22.ENN : Constant (If IFLITY is equal to ILUDW).

ENN = 0.235

23.IMA : Plot-flag (=1 plot file will be saved).

IMA = 1

24.ILMA : Number of increments after which a
plot-file will be generated.

ILMA = 50

25.IHALT : Number of increments computed.

IHALT = 50

26.IHEPSI : Flag for computing all external force
values (=0 all forces will be computed).

IHEPSI = 1

27.IDENGE : Self-correcting flag (=1 self correction
will be done).

IDENGE = 1

28.ITOP : Keyword for element type.

ITOP = "VIER"

29.IWRITEG : Number of increments for output
generation.

IWRITEG = 50

30.ZEND : Punch displacement at which computation
should cease.

ZEND = 50.0 mm

31.IX : Number of nodes in R-direction.

IX = 6

32.IY : Number of nodes in Z-direction.

IY = 20

33. RB,ZB,RC,ZC,RD,ZD,RA,ZA : R and Z coordinates of
the die contour points
A ; B ; C ; D ;

RB = 8.964 mm , ZB = 199.6 mm , RC = 7.5 mm ,

ZC = 194.136 mm , RD = 7.5 mm , ZD = 192.136 mm,

RA = 8.964 mm , ZA = 230 mm.

34.RADE : Inner radius.

RADE = 3.0 mm

35.RADA : Outside radius.

RADA = 3.0 mm

36.RK(N) : Radius of the node rows.

RK(N) = (0.0, 2.0, 4.0, 6.0, 8.0, 8.964) mm

37.ZK(N) : Axial coordinates of the node columns.

ZK(N) = (210.4, 209.8, 209.2, 208.6, 208.0, 207.4,
206.8, 206.2, 205.6, 205.0, 204.4, 203.8,
203.2, 202.6, 202.0, 201.4, 200.8, 200.2,
199.6, 199.0) mm

38.LIMVER : Basic iterations per increment.

LIMVER = 2

39.ILAST1 : Number of increments with extra

iterations at the beginning.

ILAST1 = 2

40.LIMVER1 : Number of extra iterations.

LIMVER1 = 1

41.ILAST2 : Number of second increment with extra

iterations.

ILAST2 = 3

42.LIMVER2 : Second extra iterations.

LIMVER2 = 1

43.IECK : Number of the node at the end of the
punch.

IECK = 1

44.IECK2 : Number of degree of freedom of the
given punch velocity.

IECK2 = 12

45.ANF : Punch increment.

ANF = -0.04

Appendix B

FINITE ELEMENT METHOD FORMULATION

A finite element analysis of a deformation leads to an approximate solution [32]. In two dimensional problems two types of approximations can be seen :

(1) approximate solution to a given partial differential equation,

(2) approximation of the domain by a suitable finite element mesh .

Finite element method considers the displacement of only a finite number of points or nodes of the domain. In metal forming processes workpiece is partitioned into finite number of elements by using the mesh generation programs. The required values may be found by the interpolation between the nodal values of the function. Finally, stress continuity and force equilibrium equations are used to construct a system of equations in order to describe the behaviour of all nodes.

Considering the nodal problem in the given

domain,

$$-\nabla^2 u = f(x,y) \quad (B1)$$

$$\Gamma = \Gamma_1 + \Gamma_2 \quad (B2)$$

where Γ_1 is the boundary for stresses are described, Γ_2 is the boundary for displacements are prescribed, u is the displacement.

$$\partial u / \partial n = \underline{n} \cdot \underline{\nabla} u \quad (B3)$$

The first step is to develop the variational form over an element. Then using the relation (B1) and variational formulation the following equation is found

$$0 = \int_{A^e} v [-\nabla^2 u - f] dx dy \quad (B4)$$

where v is the test function, A^e is the area of the element. By using the divergence theorem, the variational statement of the model is found as

$$\int_{A^e} \underline{\nabla} v \cdot \underline{\nabla} u dx dy = \int_{A^e} v f dx dy + \int_{\Gamma^e} v \partial u / \partial n ds \quad (B5)$$

$$B(v,u) = \int_{A^e} \underline{\nabla} v \cdot \underline{\nabla} u dx dy \quad , \quad \underline{n} \cdot \underline{\nabla} u = \partial u / \partial n \quad (B6)$$

Equation (B6) is symmetric bilinear form. It is both

linear in u and v .

$$L(v) = \int_{A^e} v f \, dx dy + \int (\partial u / \partial n) \, ds \quad (B7)$$

The equation (B7) is a linear form, $\partial u / \partial n$ is the flux in the normal direction. U is approximated over an element by using the shape functions such that

$$u = \sum_{j=1}^n u_j N_j(x,y) \quad (B8)$$

$$v = N_i \quad i = 1, 2, 3, \dots, n \quad (B9)$$

$$N_j(x,y) \quad j = 1, 2, 3, \dots, n \quad (B10)$$

Equation (B10) is called shape functions or interpolation functions. For the triangular elements, shape function must satisfy the following property

$$N_i(x_j, y_j) = \delta_{ij} \quad \delta_{ij} = 1 \quad \text{if } i = j \quad (B11)$$

$$\delta_{ij} = 0 \quad \text{if } i \neq j$$

and in general

$$N_i(x,y) = 1/2A_e (a_i + b_i x + c_i y) \quad i = 1, 2, 3, \dots, n \quad (B12)$$

$$u(x,y) = a_1 + a_2 x + a_3 y \quad (B13)$$

where δ_{ij} is the kronecker delta, $a_i, b_i, c_i, \alpha_1, \alpha_2, \dots$ are constants. For quadrilateral element shape function is defined as

$$N_i(\xi, \eta) = 1/4 (1 + \xi\xi_i) (1 + \eta\eta_i) \quad (B14)$$

where ξ_i and η_i are the coordinates of the 4 node quadrilateral master element.

Variational statement equation (B5) is modified by substituting the equations (B8) and (B9) into it.

$$\sum_{i=1}^n \int_{A^e} [(\partial N_i / \partial x_i) (\partial N_j / \partial x) + (\partial N_i / \partial y) (\partial N_j / \partial y)] dx dy u_j = \int_{A^e} f N_i dx dy + \int_{\Gamma^e} q_n^{(e)} N_i dA \quad (B15)$$

$$\text{or} \quad \sum_{j=1}^n K_{ij}^{(e)} u_j = F_i^{(e)} \quad (B16)$$

$$[K] \{a\} = \{f\} \quad (B17)$$

Equations (B15) and (B17) are called the finite element approximation formulas. After the application of stress - strain relationship to the equation (B15), the element stiffness matrix for triangular elements and elastic linear case is given in the

following form

$$[K] = \int_V [B]^T [D] [B] dV \quad (B18)$$

where $[D]$ is the elasticity matrix, $[B]$ is the derivatives of the shape function matrix, V is the volume of the element.

Element stiffness matrix of the quadrilateral element and for elastic linear case is defined as

$$[K]^{(e)} = \int_A [G(\xi, \eta)] d\xi d\eta \quad (B19)$$

where $[G] = h [B]^T [D] [B] (\det [J])$. Because of the complexity of the $[G(\xi, \eta)]$ terms, numerical integration is necessary which calls for the Gauss quadrature.

Appendix C

FINITE ELEMENT CODES FOR MICROCOMPUTERS

Searching for Interactive Preprocessors With Menues

| CODE | INTERACTIVE PREPROCESSOR and TYPE of ANALYSIS | | |
|---|---|-----------|--|
| | DATA GEN. | MESH GEN. | |
| 1 ACES (Australia) | (+) | (-) | <p>QIKDRAW system may be used to generate program input including structure definition and load application.</p> <p>Linear elastic static analysis of 2D and 3D arbitrary structures.</p> |
| 2 ADINA (U.S.A. Cambridge) | (-) | (-) | <p>(FORTRAN IV)</p> <p>Program does not include any preprocessor. Mesh generator program FEMGEN is included.</p> <p>A general purpose linear and nonlinear static and dynamic 3D program. The nonlinearities may be due to large displacements, large strains and nonlinear material behavior.</p> |
| 3 AEG-PC (Sweden Vastra Frölunda) | (+) | (-) | <p>No user instructions are needed for these programs. Since input are easily entered through the user's responses to prompting questions on the CRT screen.</p> <p>Linear static analysis.</p> |
| 4 AFENA (Australia) | (-) | (+) | <p>Preprocessor contains a limited automatic mesh generation. CHKFE and CONTUR are interactive graphics programs.</p> <p>Linear and nonlinear static analysis, heat transfer analysis.</p> |

| CODE | INTERACTIVE PREPROCESSOR and TYPE of ANALYSIS | | |
|--|---|-----------|--|
| | DATA GEN. | MESH GEN. | |
| 5 AIT (Thailand Sangkok) | (-) | (+) | <p style="text-align: right;">(BASIC)</p> <p>In preprocessor mesh generation is possible (2-D meshes).</p> <p>Linear static and dynamic analysis of general structures. The program is organized in modules.</p> |
| 6 ANSR-I and ANSR-II (U.S.A. Berkeley) | (-) | (-) | <p style="text-align: right;">(FORTRAN IV)</p> <p>Not known.</p> <p>Nonlinear general purpose codes for static and dynamic response considering both large displacement and inelastic effects.</p> |
| 7 ANSYS-PC/ LINEAR (U.S.A. Houston) | (+) | (-) | <p style="text-align: right;">(FORTRAN)</p> <p>Batch or interactive mode of operation creates data which is requested for the analysis. (Preprocessor-PREP7).</p> <p>Linear static analysis.</p> |
| 8 APOLLO (U.S.A. Boston) | (-) | (+) | <p style="text-align: right;">(FORTRAN)</p> <p>Preprocessor includes mesh generator.</p> <p>3D design of reinforced concrete buildings, linear static and dynamic analysis.</p> |
| 9 ASAS-NL (U.K. Surrey) | (-) | (-) | <p style="text-align: right;">(FORTRAN IV)</p> <p>Mesh generation program MESHMOD is included.</p> <p>Nonlinear static problems due to material or geometric nonlinearity can be handled.</p> <p>Program provides different capabilities for plasticity, creep and swelling, large deflections and buckling.</p> |
| 10 ASE (Switzerland Zurich) | (-) | (+) | <p style="text-align: right;">(FORTRAN 77)</p> <p>Preprocessor include module for mesh generation.</p> <p>Linear static and dynamic analysis of spatial structures like shells or solids.</p> |

| CODE | INTERACTIVE PREPROCESSOR and TYPE of ANALYSIS | | |
|--|---|-----------|---|
| | DATA GEN. | MESH GEN. | Data or mesh generator program exists (+) or not exists (-) in the preprocessor |
| 11 ASKA (West Germany Stuttgart) | (-) | (-) | (FORTRAN IV) Standard ASKA does not include pre-and postprocessors. For this reason separate pre and postprocessors are used in connection with ASKA, i.e., INGA, FEMGEN, GIFTS. Linear static and dynamic analysis. Some selected nonlinear applications are included, mainly elasto-plastic analysis and bifurcation buckling analysis. Isotropic, kinematic and mixed hardening are covered. |
| 12 Elastoplastic Analysis of Axisymmetric Sheels. (Portugal Codex) | (-) | (+) | (BASIC+FORTRAN) Not known. Analysis of elastoplastic axisymmetric sheels. |
| 13 B (Australia Kensington) | (+) | (-) | (PASCAL) BDATA is a preprocessor for input file preparation. Free-format input. Linear stability analysis of 2D frames. |
| 14 BASIS (Italy Genova) | (+) | (+) | (MS+BASIC) Preprocessor includes interactive graphic input/output operations, mesh generation, load generation, boundary condition generation. Linear special purpose program for planning and design of buildings. |
| 15 BEAM (U.S.A. Bethlehem) | (-) | (-) | (FORTRAN IV) Program BEAM does not have a preprocessor. Inelastic analysis of reinforced and prestressed concrete beams and beam columns. Modular programming is used. |

| CODE | INTERACTIVE PREPROCESSOR and TYPE of ANALYSIS | | |
|---|---|-----------|--|
| | DATA GEN. | MESH GEN. | |
| 16 BOSOR4 (U.S.A. Palo Alto) | (-) | (-) | (FORTRAN IV) Free format preprocessor-B4PRE is included. Certain commonly occurring geometries, such as cylinders, spheres, cones, etc. are generated with use of integer pointers and end-point coordinates. Stress, buckling, vibration of branched stiffened, elastic sheels of revolution. |
| 17 BOSOR5 (U.S.A. Palo Alto) | (-) | (-) | (FORTRAN IV) Input data is fixed format in the preprocessor. Buckling of elastic-plastic complex sheels of revolution including large deflections and creep. |
| 18 BOVA and BOVAC (U.S.A. Bethlehem) | (-) | (-) | (FORTRAN EXTENDED) Program contains a minimal preprocessing capability. Bridge overload analysis. |
| 19 BRICK (U.S.A. Pennsylvania) | (-) | (-) | (FORTRAN IV) Program contains minimal amount of preprocessing capability in terms of prediscrretization. Inelastic analysis of Masonry Panel wall-beam column structures. |
| 20 CAEFRAME (U.S.A. Sunnyvale) | (+) | (+) | The program is completely interactive. Screen-oriented, menu-driven input; mesh generation; generation of load; generation of boundary conditions. Linear static analysis of 3D frame structures. |
| 21 CAEPIPE (U.S.A. Sunnyvale) | (+) | (+) | Completely interactive preprocessor and postprocessor are included screen-oriented, menu-driven input; mesh and load generation. Linear static and dynamic 3D analysis of process/power piping systems. |

| CODE | INTERACTIVE PREPROCESSOR and TYPE of ANALYSIS | | Data or mesh generator program exists (+) or not exists (-) in the preprocessor |
|--|---|--------------|---|
| | DATA GEN. | MESH GEN. | |
| 22 CASTOR-MICRO (France Senlis) | (+) | (-) | (BASIC) Preprocessor consists of extensive data generation-free format input. Static elastic and thermal analysis of 2D structures, static and dynamic analysis of 3D frames, static and dynamic analysis, of 3D beam and shell structure. |
| 23 CEDRUS (Switzerland Zurich) | (-) | (-) | (PASCAL) Free-format input. The input file can be checked for syntax errors. Linear static analysis and design of plates and slabs. |
| 24 COSMOS/F (France Pares) | (-) | (+) | (BASIC+FORTRAN) Interactive graphics: pre and post-processors, mesh and load generators are included in the package. Linear static and dynamic analysis, of 3D structures. |
| 25 COSMOS/USA (U.S.A. Santa Monica) | (+) | (+) | Interactive mesh generator, load generator, and generation of boundary conditions are included in the program CINEMA. Linear static and dynamic analysis of general structures, heat transfer analysis, fluid flow, electrical and a coustical field problems. |
| 26 DANUTA (U.S.A. York) | (-) | (-) | (FORTRAN IV) Not known. Static and dynamic analysis of structures encountered in the mechanical and civil engineering fields. |
| 27 DIAL (U.S.A. Sunnyvale) | (-) | (-) | (FORTRAN) Free format input. 2D and 3D automatic mesh generation program is included. Static and transient nonlinear analysis, linear stress analysis, vibration and buckling. |

| CODE | INTERACTIVE PREPROCESSOR and TYPE of ANALYSIS | | |
|--|---|-----------|---|
| | DATA GEN. | MESH GEN. | Data or mesh generator program exists (+) or not exists (-) in the preprocessor |
| 28 DIANA (The Netherlands Delft) | (+) | (+) | (FORTRAN IV) Dynamic memory allocation is handled by FILOS. Communication between user and computer is by means of table-oriented (free format). Preprocessor can be used for mesh generation. Linear and nonlinear static and dynamic analysis of general structures, heat transfer analysis. |
| 29 DRAIN-2D, DRAIN-TABS (U.S.A. Berkeley) | (-) | (-) | (FORTRAN IV) There is no preprocessor. Seismic response analysis of inelastic 2D structures and 3D buildings. |
| 30 PLANS, DYCAST (U.S.A. Bethpage) | (-) | (-) | (FORTRAN IV) Preprocessor includes data checking bandwidth optimization and plotting the undeformed structure. Static nonlinear analysis of structures. Plastic and large deflection analysis of structures, dynamic crash analysis of structures. |
| 31 ELAS55 (U.S.A. Durham) | (-) | (-) | (FORTRAN IV) There is no preprocessor. Nonlinear equilibrium problems of thermo-visco-elastic-plastic solids and structures. |
| 32 FE2000 (U.S.A. Southfield) | (+) | (-) | (FORTRAN 77) Free format input. Interactive graphics preprocessor for data generator. Linear and nonlinear static analysis, linear dynamic analysis, heat transfer analysis. |
| 33 ESA (Belgium Herk-de-Stad) | (+) | (+) | (BASIC) Interactive, menu driven preprocessor facilities data input. Linear and nonlinear static and dynamic analysis of general structures, heat transfer analysis. |

| CODE | INTERACTIVE PREPROCESSOR and TYPE of ANALYSIS | | |
|--|---|-----------|---|
| | DATA GEN. | MESH GEN. | |
| 34 FEAP (U.S.A. Berkeley) | (-) | (+) | Data or mesh generator program exists (+) or not exists (-) in the preprocessor Preprocessor includes mesh generation. Linear and nonlinear static and dynamic problems, heat transfer analysis. |
| 35 FEB-SOFEC (West Germany Kerpen) | (+) | (-) | Data generation, interactive graphics included in preprocessor. Linear static and dynamic analysis of general structures. |
| 36 FEDBA INFEDBA INFELB... (Australia Kensington) | (-) | (-) | No programs available (Pre and for postprocessor). Linear elastic and inelastic buckling. Buckling analysis of beams, plates and plate assemblies. |
| 37 FEMFAM (West Germany Aachen) | (+) | (+) | (HP-BASIC) GENFAM-interactive preprocessor program (free format). Mesh and load generators are included in the package. All parts with dynamic array dimensioning. Linear static analysis, heat transfer analysis. |
| 38 FENAP (U.S.A. Lexington) | (+) | (-) | APREP is an interactive preprocessor to generate input data files for FENAP. Nonlinear axisymmetric static analysis, heat transfer analysis. |
| 39 FESDEC (Canada Ontario) | (+) | (+) | (HP-BASIC) Preprocessor includes plotter graphics, mesh generation, generation of boundary conditions and loading. Linear and nonlinear static analysis, linear dynamic analysis, heat transfer. |

| CODE | INTERACTIVE PREPROCESSOR and TYPE of ANALYSIS | | |
|------------------------------------|---|-----------|--|
| | DATA GEN. | MESH GEN. | Data or mesh generator program exists (+) or not exists (-) in the preprocessor |
| 40 | | | No programs are available. |
| FEODP (U.S.A. Lexington) | (-) | (-) | Nonlinear layered slab or cylinder heat conduction/stress analysis program. |
| 41 | | | (FORTRAN 77) |
| FINESTRA (West Germany Köln) | (-) | (+) | Preprocessor includes mesh generation program. Linear static and eigenmode analysis. |
| 42 | | | |
| FINITE/GP (U.S.A. Houston) | (+) | (+) | Disk-based data processing. All input data are stored in the same allocated memory area. Interactive mode of operation. Mesh generation and generation of boundary conditions are included. Linear static analysis, heat transfer analysis, fluid a flow. |
| 43 | | | (FORTRAN 77) |
| FLASH (Switzerland Zurich) | (-) | (+) | The input is described by syntax diagrams. Preprocessor includes generation of meshes and boundary conditions. Linear elastic static and dynamic analysis. |
| 44 | | | (FORTRAN IV) |
| FLOSYS (France Malmaison) | (-) | (-) | Free-format input. Nonlinear static and dynamic analysis, fluid-structure interaction problems. |
| 45 | | | (FORTRAN 77) |
| FLOWERS (Swiss Zuerich) | (-) | (-) | The input is described by syntax diagrams. Linear, nonlinear static and dynamic analysis. |

| CODE | INTERACTIVE PREPROCESSOR and TYPE of ANALYSIS | | |
|---|---|-----------|---|
| | DATA GEN. | MESH GEN. | |
| 46 FRA/VEGA (Israel Tel Aviv) | (-) | (+) | <p style="text-align: right;">(C-LANGUAGE)</p> <p>Preprocessor includes automatic mesh generator.</p> <p>Computation of stress concentration and stress intensity factors.</p> |
| 47 GIFTS (U.S.A. Greenbelt) | (+) | (-) | <p>Preprocessor includes powerful interactive graphics. Automatic model generation with interactive data editing.</p> <p>Linear static and dynamic analysis, heat transfer analysis.</p> |
| 48 HONDO II (U.S.A. Albuquerque) | (-) | (-) | <p style="text-align: right;">(FORTRAN IV)</p> <p>Preprocessor is not included. Mesh generator program QMESH is included by the code.</p> <p>Nonlinear 2D finite element program for the solution of large deformation finite strain, inelastic transient dynamic response of solids.</p> |
| 49 IBA (Italy Roma) | (+) | (+) | <p style="text-align: right;">(BASIC)</p> <p>Preprocessor includes model generation, numbering of nodes and elements, load generation, model checking. The preprocessor allows for data corrections and modifications.</p> <p>Interactive linear analysis of framed buildings subjected to vertical and horizontal actions.</p> |
| 50 IISS (Italy Genova) | (-) | (-) | <p>No generators are available.</p> <p>Linear static and dynamic analysis of 2D and 3D structures composed of beams or isoparametric plate elements.</p> |
| 51 IMAGES 2D, 3D (U.S.A. Berkeley) | (-) | (+) | <p>IMAGES 2D and 3D mesh generators are available. For 3D case boundary conditions and load generators are available.</p> <p>Linear static and dynamic analysis of 2D and 3D structures.</p> |

| CODE | INTERACTIVE PREPROCESSOR and TYPE of ANALYSIS | | |
|--|---|-----------|---|
| | DATA GEN. | MESH GEN. | |
| 52 JAC (U.S.A. Albuquerque) | (-) | (-) | <p style="text-align: right;">(FORTRAN IV)</p> <p>Preprocessor is not included.</p> <p>A two-dimensional program which uses a nonlinear conjugate gradient technique for the solution of the nonlinear quasistatic response of solids.</p> |
| 53 LARSTRAN 80 (West Germany Stuttgart) | (-) | (-) | <p style="text-align: right;">(FORTRAN IV)</p> <p>Program has a simple data generator capability. Interactive graphics package INGA is used.</p> <p>Large strain nonlinear analysis. LARSTRAN 80 is a program system designed to provide a basis for nonlinear static and dynamic structural computations.</p> |
| 54 LIBRA (U.S.A. Campbell) | (+) | (+) | <p>Interactive model builder, mesh generator, generation of boundary conditions and loads are included in the preprocessor.</p> <p>Linear static analysis of 2D and 3D structures, heat transfer analysis.</p> |
| 55 MARC (U.S.A. Palo Alto) | (+) | (+) | <p style="text-align: right;">(FORTRAN IV)</p> <p>Interactive preprocessor MENTAT includes data generation and 2D, 3D mesh generation programs.</p> <p>Linear and nonlinear analysis of structures in the static and dynamic system. The following nonlinearities are handled by the program: elastic-plastic, large displacements, finite strain, creep, thermally dependent properties.</p> |
| 56 Micro ABAQUS (Canada Ontario) | (-) | (+) | <p>Mesh, loading generation is possible in preprocessor.</p> <p>Linear static analysis.</p> |
| 57 MicroFe (West Germany Kaiserlautern) | (-) | (+) | <p style="text-align: right;">(BASIC+PASCAL+FORTRAN)</p> <p>Interactive graphics, mesh generation, plotting.</p> <p>Linear static and dynamic analysis.</p> |

| CODE | INTERACTIVE PREPROCESSOR and TYPE of ANALYSIS | | |
|--|---|-----------|--|
| | DATA GEN. | MESH GEN. | |
| 58 MICROFEAP (Thailand Bangkok) | (+) | (+) | <p style="text-align: right;">(BASIC)</p> <p>Interactive data preparation. Mesh and load generation can be handled.</p> <p>Program can solve 2D and 3D linear elastic static problems. Modular programming is used. The user controls solution steps by specifying a set of macrocommands.</p> |
| 59 MICROFEA (U.S.A. Bellevue) | (-) | (-) | <p>3D interactive color graphic.</p> <p>Linear static analysis 2D and 3D structures can be analyzed.</p> |
| 60 MICROFIELD (U.K. West Glamorgan) | (-) | (+) | <p>Interactive graphic.</p> <p>A macro element technique is used for the mesh generation. The mesh, boundary conditions and applied loads can be displayed. Linear static stress analysis.</p> |
| 61 MICROPUS (West Germany Aachen) | (-) | (+) | <p>Interactive graphics. Mesh generation-MORDOR.</p> <p>The program is developed to solve problems appearing in extrusion of polymers.</p> |
| 62 MICROSAFE (U.S.A. WA) | (-) | (-) | <p>Interactive color graphics to display the model. Free format input.</p> <p>Linear elastic static analysis.</p> |
| 63 MICRO-SIMS (Greece Thessaloniki) | (+) | (-) | <p style="text-align: right;">(BASIC)</p> <p>Interactive data input and modifications are included by preprocessor.</p> <p>Matrix manipulation, static and dynamic linear elastic structural analysis program.</p> |

| CODE | INTERACTIVE PREPROCESSOR and TYPE of ANALYSIS | | |
|--|---|-----------|--|
| | DATA GEN. | MESH GEN. | Data or mesh generator program exists (+) or not exists (-) in the preprocessor |
| 64 MICRO STRESS (U.K. Southampton) | (+) | (+) | Free-format input. Mesh generator is available. Plotting is possible. Linear static analysis of frames. |
| 65 MINDLIN (Italy Padova) | (+) | (-) | (BASIC+FORTRAN 77) Automatic data generator is included in the preprocessor. Analysis of thick plates on elastic soil. |
| 66 MSC/NASTRAN (U.S.A. NASA) | (-) | (+) | (FORTRAN IV) Program includes batch mesh generation in the MSG MESH module. The basic program processes free and fixed field input data files. Linear and nonlinear general purpose structural analysis program. |
| 67 MSC/pal (U.S.A. Los Angeles) | (+) | (-) | An interactive preprocessor includes free format data creation and modification. Graphical output for result presentation. 2D and 3D linear static and dynamic analysis. Modular programming is used. Interactive model generation-PALPREP. |
| 68 NEPSAP (U.S.A. Sunnyvale) | (-) | (-) | (FORTRAN+ASSEMBLY) Free format input. Nonlinear 3D elastic-plastic structural analysis program capable of large displacement, thermo-elastic-plastic and creep analysis of arbitrary structures. |
| 69 NISA/DISPLAY (U.S.A. Tray) | (-) | (+) | DISPLAY/DIGIT-interactive preprocessor. It includes menu-driven mesh generation/modification. Static and dynamic analysis of 2D and 3D structures, heat transfer analysis, analysis of field problems. Modular programming is used. |

| CODE | INTERACTIVE PREPROCESSOR and TYPE of ANALYSIS | | |
|---|---|-----------|---|
| | DATA GEN. | MESH GEN. | Data or mesh generator program exists (+) or not exists (-) in the preprocessor |
| 70 NOLINA (U.K. Wales) | (-) | (-) | (BASIC) Input data file is created in free-format input. Nonlinear elasto-plastic stress analysis. NOLINA is a program for the elasto-plastic analysis including creep of 2D and axisymmetric structures. Modular concept program design. |
| 71 OLIGINEST (Italy Milano) | (+) | (-) | (ASSEMBLY+BASIC) Input data are prepared using system file editor. Static and dynamic analysis of 2D and 3D structures. Modular concept program design. |
| 72 PAC78 (Egypt Giza) | (-) | (-) | (FORTRAN IV) Program does not include any pre/post-processors. PAC78 (Plastic Analysis of Composites) is a general purpose 3D code. The program may be used for linear elastic analysis or nonlinear elastic-plastic analysis. |
| 73 PC-MEF (Canada Quebec) | (+) | (+) | (PASCAL) PREMEF is an interactive preprocessor which prepares the database for PC-MEF. Automatic generation of meshes. Linear elastic static analysis, potential flows in porous media. |
| 74 PCSAP (Australia Indooroopilly) | (-) | (-) | Not known. Linear static analysis of 3D frames. |
| 75 PEPSY (West Germany Postfach) | (-) | (-) | (BASIC) Free-format input. Linear static analysis of arbitrary large spatial frameworks. |

| CODE | INTERACTIVE PREPROCESSOR and TYPE of ANALYSIS | | |
|--|---|-----------|---|
| | DATA GEN. | MESH GEN. | |
| 76 PERFINE (U.K. Coventry) | (-) | (+) | <p>Data or mesh generator program exists (+) or not exists (-) in the preprocessor</p> <p>Graphics in the preprocessor is exist. Boundary and mesh generation is included.</p> <p>Linear static analysis, potential problems in 2D and axisymmetric geometries. Modular programming is used.</p> |
| 77 PLANEAXI/MICRO (U.K. Swansea) | (+) | (+) | <p>(FORTRAN IV+BASIC)</p> <p>The interactive preprocessor-postprocessor consists of four programs. MESHING-easy input and data editing AUTOGEN-automatic mesh generation using macro elements. MESHOPT-mesh renumbering. PLOTDATA-mesh display at any stage of the mesh preparation.</p> <p>Linear static analysis of 2D structures. Modular programming.</p> |
| 78 PLATE (Austria Graz) | (-) | (+) | <p>Interactive graphics.</p> <p>Linear static analysis of plate structures. Which are subjected to arbitrary mechanical and thermal loads. Program has restart capability.</p> |
| 79 POLO-FINITE. (U.S.A. Lawrence) | (+) | (-) | <p>(POL+ANSI 66+FORTRAN)</p> <p>Preprocessor is prepared with problem oriented language (POL) in batch and interactive modes. The POL includes coordinate and incidence data generation commands that minimize the need for preprocessors.</p> <p>Linear and nonlinear static analysis program.</p> |
| 80 PORTSMOUTH-MICRO (S&CON package) (U.K. Portsmouth) | (-) | (-) | <p>(BASIC+PASCAL)</p> <p>No generators available.</p> <p>Linear and nonlinear static and dynamic analysis, heat transfer analysis, field problem analysis. The software developed is aimed as a teaching tool for problems in structural and continuum mechanics.</p> |

| CODE | INTERACTIVE PREPROCESSOR and TYPE of ANALYSIS | | |
|--|---|-----------|---|
| | DATA GEN. | MESH GEN. | |
| 81 PSTAR (Switzerland Baden) | (-) | (+) | Data or mesh generator program exists (+) or not exists (-) in the preprocessor Mesh, boundary conditions and loading generation included in the preprocessor Graphics are interactive. Linear static and dynamic analysis of piping systems. |
| 82 SAMCEF (Belgium Liege) | (+) | (-) | (FORTRAN IV) Program has pre-and postprocessors as well as graphics displays for the data and results. Linear and nonlinear, static and dynamic, 3D analysis code. |
| 83 SAMSON (U.S.A. Evanston) | (-) | (-) | (FORTRAN IV) Program does not include preprocessor. Dynamic stress analysis of media structure problems with nonlinearities. |
| 84 SAP80 (U.S.A. Berkeley) | (-) | (+) | Free-format input. Linear static and dynamic analysis, analysis of fluid problems. The program segments are designed to operate with a common database system. |
| 85 SAP86 (U.S.A. San Francisco) | (+) | (+) | Fully interactive input and output capabilities. Models are preparing using full-screen data forms. MICROTAB is a general purpose interactive preprocessor. Linear static and dynamic analysis of 2D and 3D structures. SAP86 is an modified version of SAPIV program. |
| 86 SCADA (U.S.A. Los Angles) | (+) | (+) | Preprocessor includes mesh generation, generation of loads, boundary conditions. Linear and nonlinear static and dynamic analysis, heat transfer analysis, fluid flow problems. |

| CODE | INTERACTIVE PREPROCESSOR and TYPE of ANALYSIS | | |
|--------------------------------------|---|-----------|--|
| | DATA GEN. | MESH GEN. | Data or mesh generator program exists (+) or not exists (-) in the preprocessor |
| 87 | | | (ANSI FORTRAN) |
| SESAM-69 (Norway Hovik) | (+) | (-) | Preprocessor includes geometric modelling, specification of loads, boundary conditions, interactive data generation facilities. The input, data generator has checking and visualization facilities. Linear and nonlinear analysis program. The solution algorithm is based on the superelement-technique. The total system is split into program modules of pipes, beams, membranes, shells and solids which may be executed with the super-element program. |
| 88 | | | (PASCAL) |
| SMAP (Australia, N.S.N) | (-) | (-) | Free-format input. User written preprocessor may be included. Linear and nonlinear static analysis of 2D structure. |
| 89 | | | (FORTRAN IV) |
| SMART (West Germany Stuttgart) | (-) | (-) | Pre/postprocessor is included in the module INGA. Linear and nonlinear static analysis. Nonlinear applications are included by SMART I for limit load analysis, viscoelastic creep problems. SMART II contains nonlinear, nonstationary coupled hygrothermal analysis. |
| 90 | | | |
| SNAP/FE (U.S.A. Blacksburg) | (+) | (-) | Menu-driven programs are included by preprocessor. Line type input. Linear static, transient and vibration analysis. |
| 91 | | | |
| SPACEFRAME (Austria Graz) | (-) | (+) | Interactive graphics capabilities. Mesh and load generators are available. Linear static and dynamic analysis of 3D frame structures. |

| CODE | INTERACTIVE PREPROCESSOR and TYPE of ANALYSIS | | |
|---|---|-----------|--|
| | DATA GEN. | MESH GEN. | Data or mesh generator program exists (+) or not exists (-) in the preprocessor |
| 92 STAGSC-1 (U.S.A. Palo Alto) | (-) | (-) | (FORTRAN IV) Software GIFTS includes pre- and postprocessing modules. Automatic data management facilities are included. Linear and nonlinear structural analysis of shell type structures. It contains static stress analysis, bifurcation buckling, vibrations and transient response. |
| 93 sst-micro (West Germany Bochum) | (+) | (-) | Free-format input. Total data input is divided according to objective criteria into submenus with arbitrary calling squence. - mask oriented input. - automatic control of user input. Linear and nonlinear static analysis of structures composed of arbitrary beams. |
| 94 STAPOG (U.S.A. Boston) | (+) | (+) | (FORTRAN+BASIC) Menu-driven input. Interactive mode of operation. The data are storing in an output datafile. Display on the screen. Linear static analysis and optimal desing of 3D truss structures. |
| 95 STAR (Switzerland Zurich) | (-) | (-) | (FORTRAN 77) Graphical display Static analysis of spatial frames. |
| 96 STRAND/AUS (Australia Sydney) | (+) | (+) | (PASCAL) Preprocessor STIN performs interactive menu-driven operations. Mesh generation facilities, generation of boundary conditions and loads, are included by this preprocessor. Linear and nonlinear static analysis, linear dynamic analysis, heat transfer analysis. |
| 97 STRAW (U.S.A. Argonne) | (-) | (-) | (FORTRAN IV) Not known. Short duration transient, nonlinear response of core subassemblies for safety analysis in the Liquid Metal Fast Breeder Reactor program. |

| CODE | INTERACTIVE PREPROCESSOR and TYPE of ANALYSIS | | |
|--|---|-----------|--|
| | DATA GEN. | MESH GEN. | |
| 98 SUPERSAP (U.S.A. Pittsburgh) | (+) | (+) | Data or mesh generator program exists (+) or not exists (-) in the preprocessor Interactive modules for data input, checking and editing, mesh generators are included by the module BEAMBUILDER. Linear and nonlinear static and dynamic analysis, heat transfer analysis, of general structures. |
| 99 TABU (Italy Padova) | (-) | (+) | Preprocessor postprocessor includes mesh generation and plotting. Linear static and dynamic analysis of tall buildings. |
| 100 TEPSA (Canada Manitoba) | (-) | (-) | (FORTRAN IV) Not known. Nonlinear thermal elastic-plastic stress analysis. Code accepts temperature and strain-rate dependent material properties. Isotropic hardening rule is used for load increments and kinematic hardening rule for cyclic thermomechanical loadings. |
| 101 TITOU (France Marseille) | (-) | (-) | (BASIC) Data can be input by using a digitizer. Data modification without restarting the whole process. Linear static analysis, analysis of field problems. |
| 102 TSTAR (Switzerland Baden) | (-) | (+) | Interactive graphics. Generators for mesh, boundary conditions and loading are Generators for mesh, boundary conditions and loading are tee-junction stress analysis and reporting. |
| 103 ULARC (U.S.A. Berkeley) | (-) | (-) | (FORTRAN IV) No pre-or postprocessors. No mesh generation. Ultimate load analysis of small plane frames. |

| CODE | | INTERACTIVE PREPROCESSOR and TYPE of ANALYSIS | |
|------|---------------------------------|---|-----------|
| | | DATA GEN. | MESH GEN. |
| 104 | WECAN (U.S.A. Pittsburgh) | (+) | (+) |
| 105 | WHAMS (U.S.A. Evanston) | (-) | (-) |

INTERACTIVE PREPROCESSOR and TYPE of ANALYSIS

Data or mesh generator program exists
(+) or not exists (-) in the preprocessor

(FORTRAN IV COMPASS)

FIGURES is a collection of interactive preprocessors for WECAN. The preprocessors generate meshes and loads, check isoparametric elements shapes, reduce wave fronts prepare input for general matrix input and for composite materials. WAPPP is a collection of batch pre- and postprocessors for WECAN.

Linear and nonlinear static and dynamic 3D analysis program.

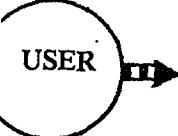
(FORTRAN IV)

No preprocessors. Simple mesh generators are included.

Transient analysis of 2D structures and continua. Large strains may be treated by some of the elements. The following material models are included: elastic, elastic-plastic with isotropic strain hardening, the cap model for soils.

Appendix D

MENU LIST OF D A T O R



```

D A T O R

(D A T A   G E N E R A T O R)
For the Finite Element Code  EPDAN
Version : 17.2.1989
By : Ezgi Gunay

```

```

Reading of the data from the SCREEN and
CREATION of the output-file

(Default : DATOR.INP)

```

Please enter the output file name >>>>?
WARNING : The File Must Exist

```

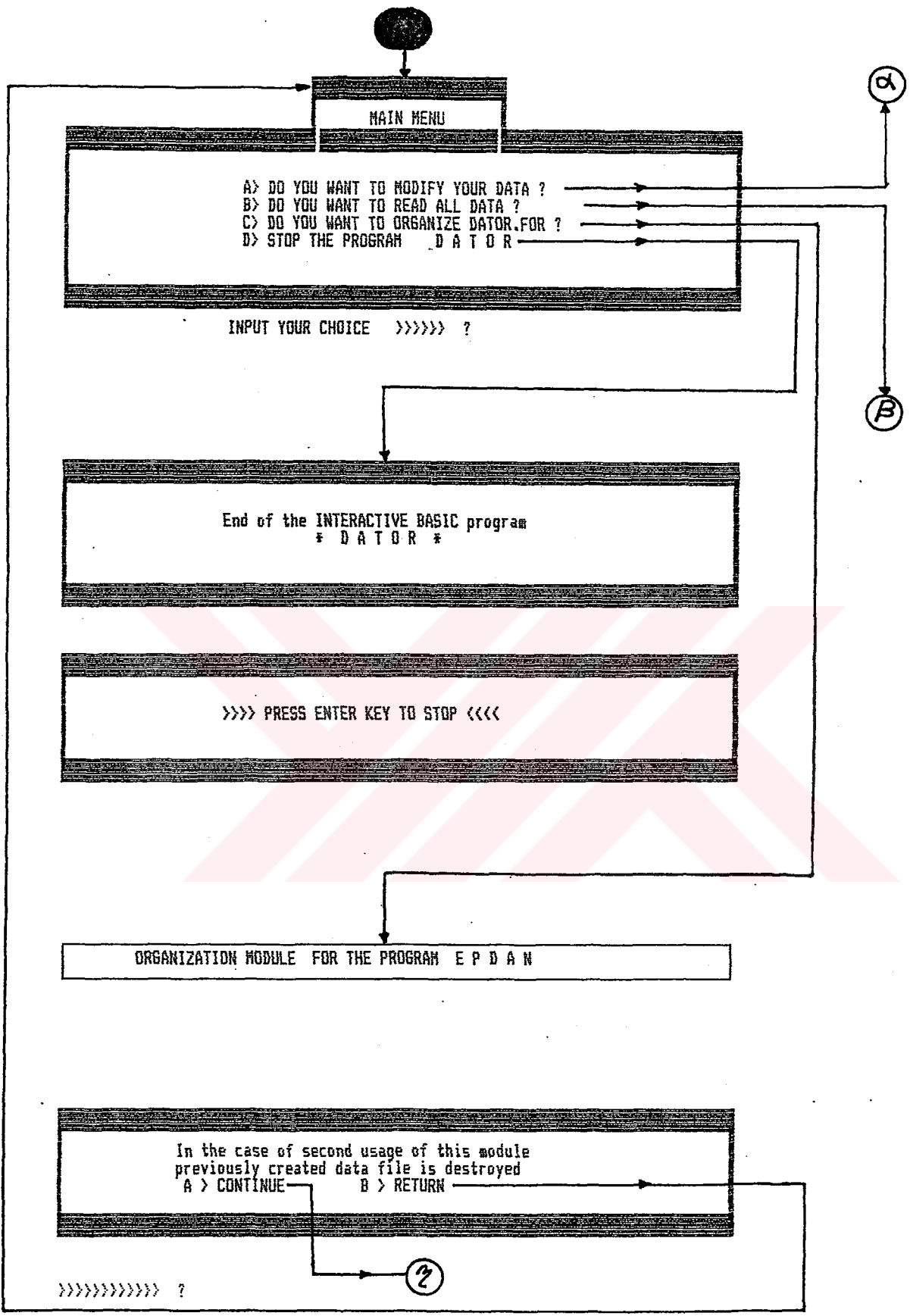
The temporary file containing all
arrays will be created

(Default : DATOR.DAT)

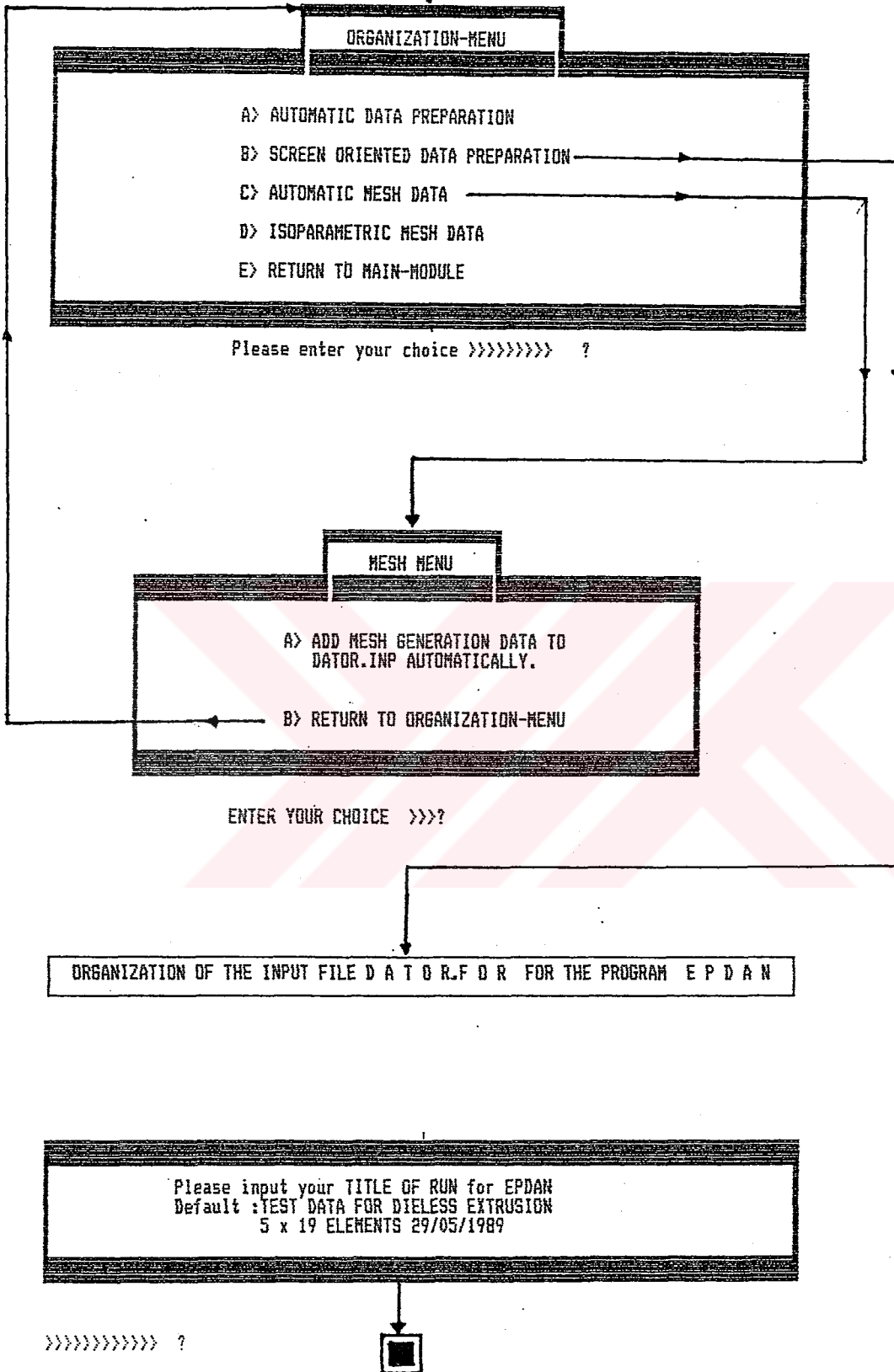
```

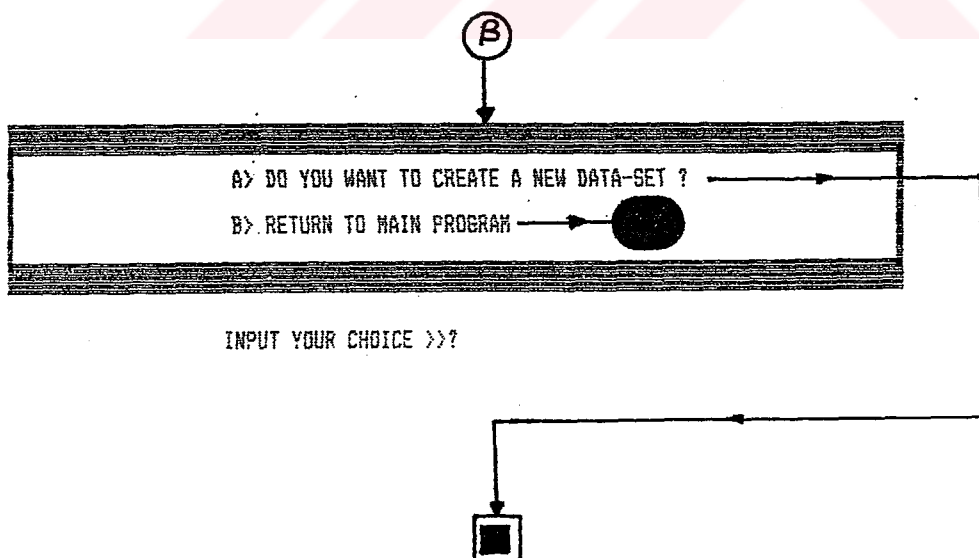
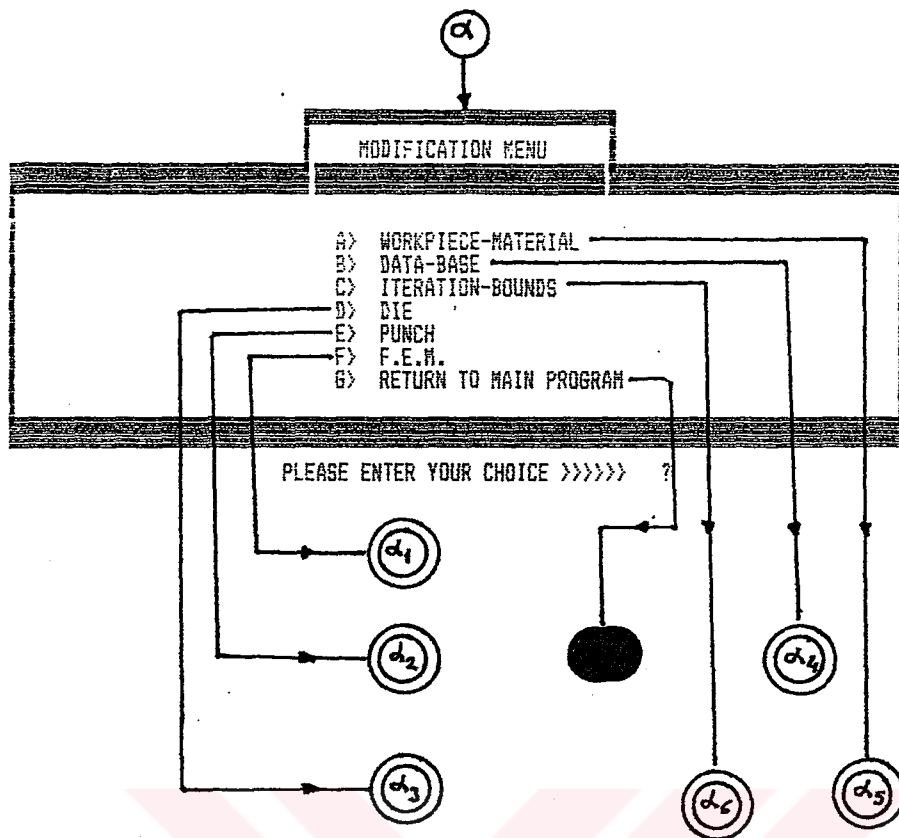
Please enter the file name >>>?
WARNING : The File Must Exist





?





ds

WORKPIECE - MATERIAL DATA

WORKPIECE MENU

- A> ELASTIC TYPE DATA
- B> PLASTIC TYPE DATA
- C> RETURN TO MODIFY MENU

ENTER YOUR CHOICE >>>>> ?
 REMARK: Hit the ENTER key if you don't use FUNCTION-KEYS for the second prompt
 1 WORK 2 DATB 3 ITERB 4 DIE 5 PUNCH 6 F.E.M 7 MAIN 8 MODT

A

MATERIAL CONSTANTS

- A> SKFNUL=Initial flow stress
- B> EM=Young's Modulus
- C> PD=Poisson's Ratio
- D> SK=Coefficient of Friction
- E> RETURN TO THE WORKPIECE-MENU

ENTER YOUR CHOICE >>>>> ?
 REMARK: Hit the ENTER key if you don't use FUNCTION-KEYS for the second prompt
 1 WORK 2 DATB 3 ITERB 4 DIE 5 PUNCH 6 F.E.M 7 MAIN 8 MODT



α5

WORKPIECE - MATERIAL DATA

WORKPIECE MENU

A) ELASTIC TYPE DATA

B) PLASTIC TYPE DATA

C) RETURN TO MODIFY MENU — α

ENTER YOUR CHOICE >>>>>?

REMARK: Hit the ENTER key if you don't use FUNCTION-KEYS for the second prompt

1 WORK 2 DATB 3 ITERB 4 DIE 5 PUNCH 6 F.E.M 7 MAIN 8 MODT

PLASTIC TYPE DATA INPUT

PLASTIC

A) KEYWORD

B) RETURN TO WORKPIECE MENU

ENTER YOUR CHOICE >>>>> ?

REMARK: Hit the ENTER key if you don't use FUNCTION-KEYS for the second prompt

1 WORK 2 DATB 3 ITERB 4 DIE 5 PUNCH 6 F.E.M 7 MAIN 8 MODT

IFLITY - KEYWORD FOR THE FLOW CURVE TYPE

PLEASE INPUT THE KEYWORD

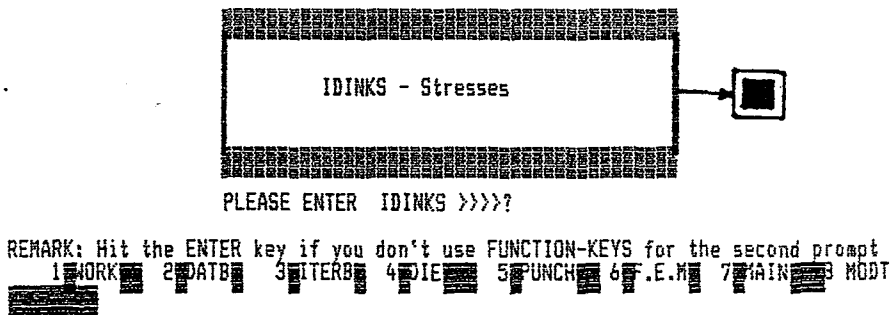
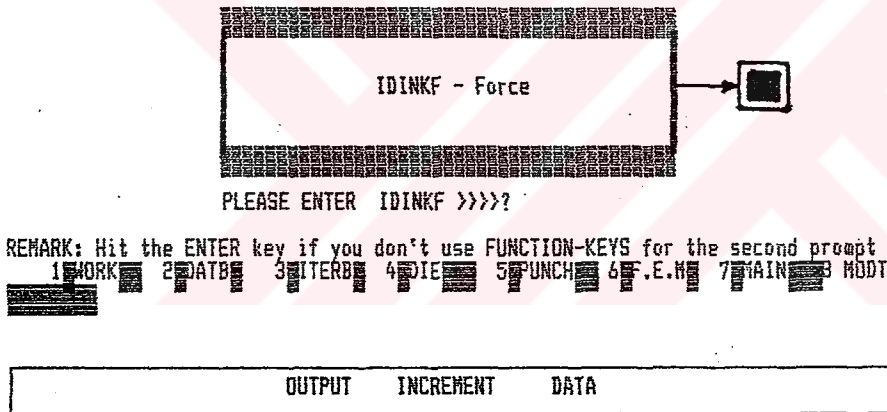
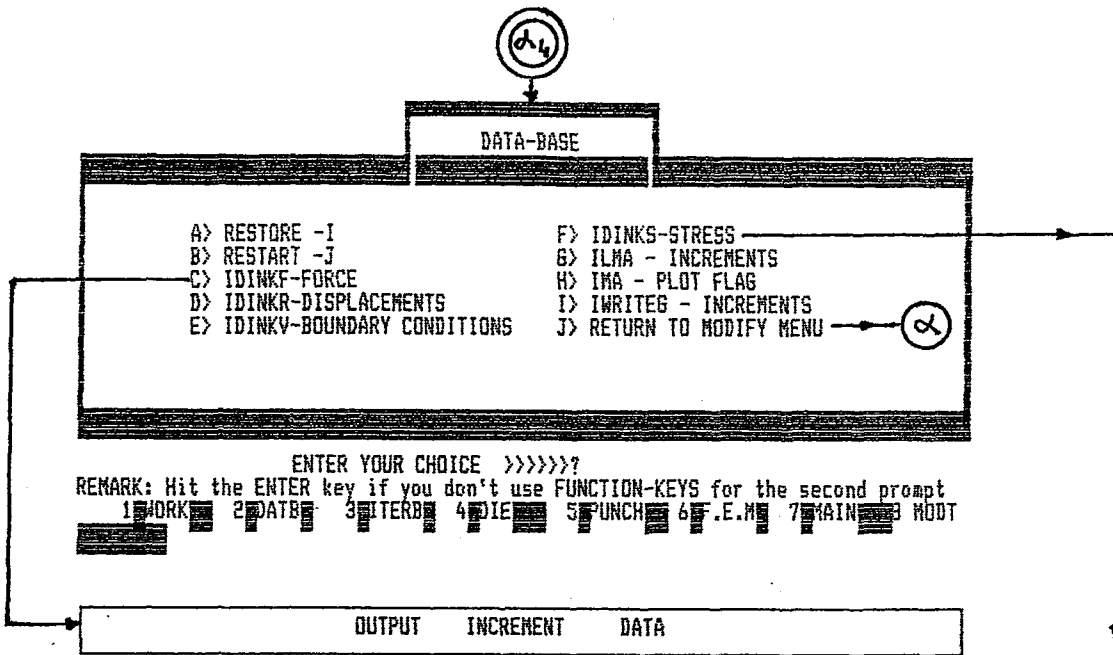
IFLITY

IFLITY >>>...?

REMARK: Hit the ENTER key if you don't use FUNCTION-KEYS for the second prompt

1 WORK 2 DATB 3 ITERB 4 DIE 5 PUNCH 6 F.E.M 7 MAIN 8 MODT

A





DATA-BASE

```

A> RESTORE -I          F> IDINKS-STRESS
B> RESTART -J         G> ILMA - INCREMENTS
C> IDINKF-FORCE       H> IMA - PLOT FLAG
D> IDINKR-DISPLACEMENTS I> IWRITEG - INCREMENTS
E> IDINKV-BOUNDARY CONDITIONS J> RETURN TO MODIFY MENU
  
```



ENTER YOUR CHOICE >>>>?

REMARK: Hit the ENTER key if you don't use FUNCTION-KEYS for the second prompt

1 WORK 2 DATB 3 ITERB 4 DIE 5 PUNCH 6 .E.M 7 IAIN 8 MODT

OUTPUT INCREMENT DATA

IDINKV - Boundary Conditions



PLEASE ENTER YOUR DATA IDINKV >>>>?

REMARK: Hit the ENTER key if you don't use FUNCTION-KEYS for the second prompt

1 WORK 2 DATB 3 ITERB 4 DIE 5 PUNCH 6 .E.M 7 IAIN 8 MODT

OUTPUT INCREMENT DATA

IDINKR - Displacements



PLEASE ENTER YOUR DATA IDINKR >>>>?

REMARK: Hit the ENTER key if you don't use FUNCTION-KEYS for the second prompt

1 WORK 2 DATB 3 ITERB 4 DIE 5 PUNCH 6 .E.M 7 IAIN 8 MODT

(dw)

DATA-BASE

- A> RESTORE -I
- B> RESTART -J
- C> IDINKF-FORCE
- D> IDINKR-DISPLACEMENTS
- E> IDINKV-BOUNDARY CONDITIONS
- F> IDINKS-STRESS
- G> ILMA - INCREMENTS
- H> IMA - PLOT FLAG
- I> IWRITEG - INCREMENTS
- J> RETURN TO MODIFY MENU

(2)

ENTER YOUR CHOICE >>>>>?

REMARK: Hit the ENTER key if you don't use FUNCTION-KEYS for the second prompt

1 WORK 2 DATB 3 ITERB 4 DIE 5 PUNCH 6 F.E.M 7 MAIN 8 MODT

RESTARTING MENU

- A> FLAG FOR RESTARTING
- X> RETURN TO DATA-BASE

ENTER YOUR CHOICE >>>>> ?

REMARK: Hit the ENTER key if you don't use FUNCTION-KEYS for the second prompt

1 WORK 2 DATB 3 ITERB 4 DIE 5 PUNCH 6 F.E.M 7 MAIN 8 MODT

RESTORE MENU

- A> FLAG FOR RESTORING
- X> RETURN TO DATA-BASE

ENTER YOUR CHOICE >>>>> ?

1 WORK 2 DATB 3 ITERB 4 DIE 5 PUNCH 6 F.E.M 7 MAIN 8 MODT



DATA-BASE

```

A> RESTORE -I          F> IDINKS-STRESS
B> RESTART -J         G> ILMA - INCREMENTS
C> IDINKF-FORCE       H> IMA - PLOT FLAG
D> IDINKR-DISPLACEMENTS I> IWRITES - INCREMENTS
E> IDINKV-BOUNDARY CONDITIONS J> RETURN TO MODIFY MENU

```



ENTER YOUR CHOICE >>>>?

REMARK: Hit the ENTER key if you don't use FUNCTION-KEYS for the second prompt
 1 WORK 2 DATB 3 ITERB 4 DIE 5 PUNCH 6 F.E.M 7 MAIN 8 MDDT

```

IWRITES : NUMBER OF INCREMENTS FOR OUTPUT GENERATION

```

PLEASE ENTER YOUR DATA FOR IWRITES >>> ?

REMARK: Hit the ENTER key if you don't use FUNCTION-KEYS for the second prompt
 1 WORK 2 DATB 3 ITERB 4 DIE 5 PUNCH 6 F.E.M 7 MAIN 8 MDDT

```

ILMA - NUMBER OF INCREMENTS AFTER WHICH A PLOT-DATA-SET
WILL BE ENTERED INTO THE DATA-BASE

```

Please enter your data ILMA >>>?

REMARK: Hit the ENTER key if you don't use FUNCTION-KEYS for the second prompt
 1 WORK 2 DATB 3 ITERB 4 DIE 5 PUNCH 6 F.E.M 7 MAIN 8 MDDT

24

DATA-BASE

- A) RESTORE -I
- B) RESTART -J
- C) IDINKF-FORCE
- D) IDINKR-DISPLACEMENTS
- E) IDINKV-BOUNDARY CONDITIONS
- F) IDINKS-STRESS
- G) ILMA - INCREMENTS
- H) IMA - PLOT FLAG
- I) IWRITEG - INCREMENTS
- J) RETURN TO MODIFY MENU

2

ENTER YOUR CHOICE >>>>?

REMARK: Hit the ENTER key if you don't use FUNCTION-KEYS for the second prompt
1 WORK 2 DATB 3 ITERB 4 DIE 5 PUNCH 6 F.E.M 7 MAIN 8 MODT

PLOT FLAG DATA-BASE

- A) WILL BE CREATED
- X) RETURN TO DATA-BASE MENU

Please enter your choice >>>>?

REMARK: Hit the ENTER key if you don't use FUNCTION-KEYS for the second prompt
1 WORK 2 DATB 3 ITERB 4 DIE 5 PUNCH 6 F.E.M 7 MAIN 8 MODT

IMA = PLOT-FLAG(plot data-base will be created)

Please enter your data for IMA >>>> ?

REMARK: Hit the ENTER key if you don't use FUNCTION-KEYS for the second prompt
1 WORK 2 DATB 3 ITERB 4 DIE 5 PUNCH 6 F.E.M 7 MAIN 8 MODT

26

ITERATION BOUNDS DATA

ITER MENU

- A) LIMVER
- B) ILAST1
- C) LIMVER1
- D) ILAST2
- E) LIMVER2
- F) ITANL
- G) IHALT
- H) RETURN TO MODIFY MENU

α

ENTER YOUR CHOICE >>>>?

REMARK: Hit the ENTER key if you don't use FUNCTION-KEYS for the second prompt

1. JORK 2. DATB 3. ITERB 4. JIE 5. PUNCH 6. F.E.M 7. MAIN 8. MDDT



IHALT - NUMBER OF INCREMENTS TO BE COMPUTED



Please enter your data IHALT >>>> ?

REMARK: Hit the ENTER key if you don't use FUNCTION-KEYS for the second prompt

1. JORK 2. DATB 3. ITERB 4. JIE 5. PUNCH 6. F.E.M 7. MAIN 8. MDDT

CONTACT DATA

ITANL=NUMBER OF ADDITIONAL ITERATION FOR THE INCREMENT
AFTER THE CONTACT



Please enter ITANL >>?

REMARK: Hit the ENTER key if you don't use FUNCTION-KEYS for the second prompt

1. JORK 2. DATB 3. ITERB 4. JIE 5. PUNCH 6. F.E.M 7. MAIN 8. MDDT

26

ITERATION BOUNDS DATA

ITER MENU

- A> LIMVER
- B> ILAST1
- C> LIMVER1
- D> ILAST2
- E> LIMVER2
- F> ITANL
- G> IHALT
- H> RETURN TO MODIFY MENU

ENTER YOUR CHOICE >>>>?

REMARK: Hit the ENTER key if you don't use FUNCTION-KEYS for the second prompt
1 WORK 2 DATB 3 ITERB 4 DIE 5 PUNCH 6 F.E.M 7 MAIN 8 MDDT



LIMVER - BASIC ITERATIONS / INCREMENT

Please enter the data LIMVER >>>?

REMARK: Hit the ENTER key if you don't use FUNCTION-KEYS for the second prompt
1 WORK 2 DATB 3 ITERB 4 DIE 5 PUNCH 6 F.E.M 7 MAIN 8 MDDT



LIMVER2 - SECOND EXTRA ITERATIONS

Please enter the data LIMVER2 >>>?

REMARK: Hit the ENTER key if you don't use FUNCTION-KEYS for the second prompt
1 WORK 2 DATB 3 ITERB 4 DIE 5 PUNCH 6 F.E.M 7 MAIN 8 MDDT

(26)

ITERATION BOUNDS DATA

ITER MENU

- A> LIMVER
- B> ILAST1
- C> LIMVER1
- D> ILAST2
- E> LIMVER2
- F> ITANL
- G> IHALT
- H> RETURN TO MODIFY MENU

ENTER YOUR CHOICE >>>>>?

REMARK: Hit the ENTER key if you don't use FUNCTION-KEYS for the second prompt
 1 WORK 2 DATB 3 ITERB 4 JIE 5 PUNCH 6 F.E.M 7 MAIN 8 MODT



ILAST2 - NUMBER OF SECOND INCREMENT WITH EXTRA ITERATIONS

Please enter the data ILAST2 >>>?

REMARK: Hit the ENTER key if you don't use FUNCTION-KEYS for the second prompt
 1 WORK 2 DATB 3 ITERB 4 JIE 5 PUNCH 6 F.E.M 7 MAIN 8 MODT



LIMVER1 - NUMBER OF EXTRA ITERATIONS

Please enter the data LIMVER1 >>>?

REMARK: Hit the ENTER key if you don't use FUNCTION-KEYS for the second prompt
 1 WORK 2 DATB 3 ITERB 4 JIE 5 PUNCH 6 F.E.M 7 MAIN 8 MODT

26

ITERATION BOUNDS DATA

ITER MENU

- A> LINVER
- B> ILAST1
- C> LINVER1
- D> ILAST2
- E> LINVER2
- F> ITANL
- G> IHALT
- H> RETURN TO MODIFY MENU

ENTER YOUR CHOICE >>>>?

REMARK: Hit the ENTER key if you don't use FUNCTION-KEYS for the second prompt
 1 WORK 2 DATB 3 ITERB 4 DIE 5 PUNCH 6 F.E.M 7 MAIN 8 MODT



ILAST1 - NUMBER OF INCREMENTS WITH EXTRA ITERATIONS

Please enter the data ILAST1 >>>?

REMARK: Hit the ENTER key if you don't use FUNCTION-KEYS for the second prompt
 1 WORK 2 DATB 3 ITERB 4 DIE 5 PUNCH 6 F.E.M 7 MAIN 8 MODT

23

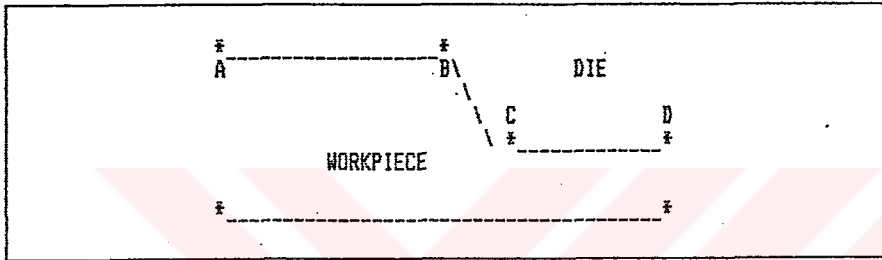
DIE MENU

- A> R and Z COORDINATES OF THE DIE CONTOUR POINTS
- B> RADA - OUTLET RADIUS OF THE DIE
- C> RADE - INLET RADIUS OF THE DIE
- D> INANL - NUMBER OF NODES WHICH MAY HIT THE DIE
- E> RETURN TO MODIFY MENU

ENTER YOUR CHOICE >>>>> ?

REMARK: Hit the ENTER key if you don't use FUNCTION-KEYS for the second prompt

1 WORK 2 DATB 3 ITERB 4 DIE 5 PUNCH 6 F.E.M 7 MAIN 8 MODT



COORD DIE
R & Z COORDINATES OF THE POINTS

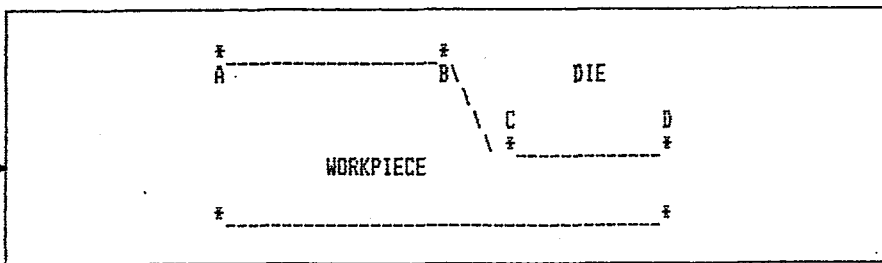
- 1> B
- 2> C
- 3> D
- 4> A

X> RETURN TO THE DIE MENU

Please enter your choice >>>>> ?

REMARK: Hit the ENTER key if you don't use FUNCTION-KEYS for the second prompt

1 WORK 2 DATB 3 ITERB 4 DIE 5 PUNCH 6 F.E.M 7 MAIN 8 MODT



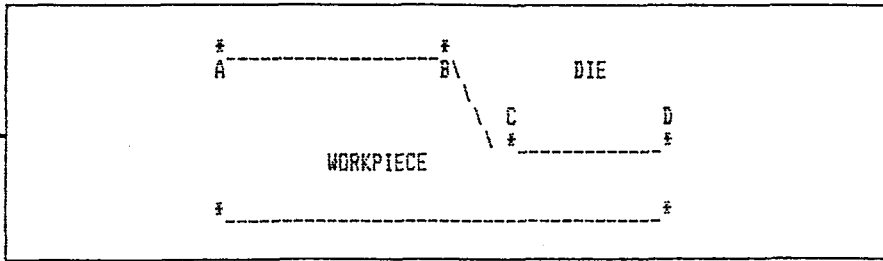
RB : R-COORDINATE OF THE POINT B
ZB : Z-COORDINATE OF THE POINT B

PLEASE ENTER YOUR DATA FOR RB,ZB >> ?

REMARK: Hit the ENTER key if you don't use FUNCTION-KEYS for the second prompt

1 WORK 2 DATB 3 ITERB 4 DIE 5 PUNCH 6 F.E.M 7 MAIN 8 MODT

(d3)



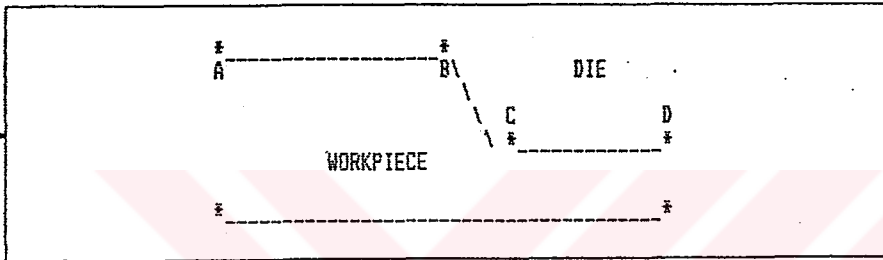
COORD DIE
R & Z COORDINATES OF THE POINTS

- 1> B
- 2> C
- 3> D
- 4> A

X> RETURN TO THE DIE MENU
Please enter your choice >>>>> ?

REMARK: Hit the ENTER key if you don't use FUNCTION-KEYS for the second prompt

1 WORK 2 DATB 3 ITERB 4 DIE 5 PUNCH 6 F.E.M 7 MAIN 8 MODT

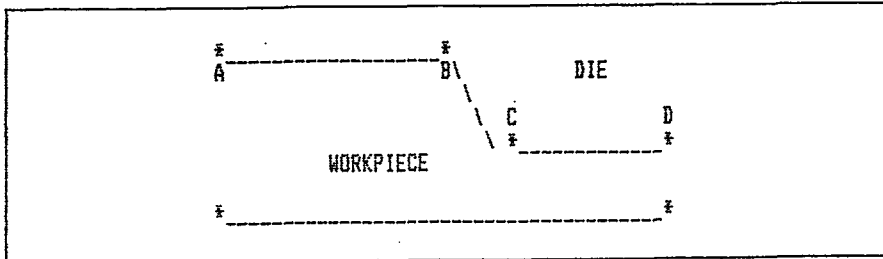


RD : R-COORDINATE OF THE POINT D
ZD : Z-COORDINATE OF THE POINT D

PLEASE ENTER YOUR DATA FOR RD,ZD >> ?

REMARK: Hit the ENTER key if you don't use FUNCTION-KEYS for the second prompt

1 WORK 2 DATB 3 ITERB 4 DIE 5 PUNCH 6 F.E.M 7 MAIN 8 MODT



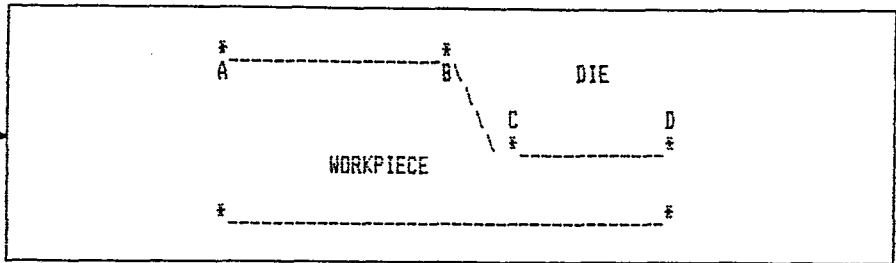
RA : R-COORDINATE OF THE POINT A
ZA : Z-COORDINATE OF THE POINT A

PLEASE ENTER YOUR DATA FOR RA,ZA >> ?

REMARK: Hit the ENTER key if you don't use FUNCTION-KEYS for the second prompt

1 WORK 2 DATB 3 ITERB 4 DIE 5 PUNCH 6 F.E.M 7 MAIN 8 MODT

23



COORD DIE
R & Z COORDINATES OF THE POINTS

- 1> B
- 2> C
- 3> D
- 4> A

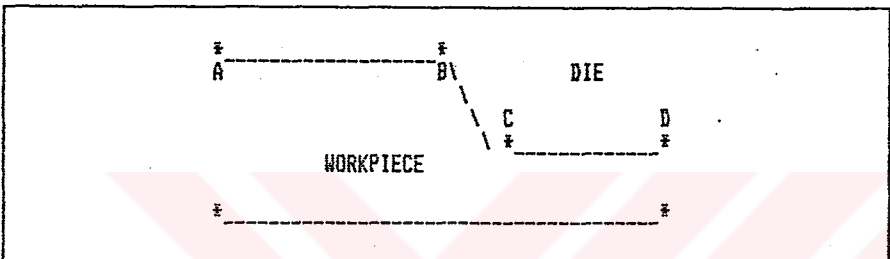
X> RETURN TO THE DIE MENU

Please enter your choice >>>>> ?

REMARK: Hit the ENTER key if you don't use FUNCTION-KEYS for the second prompt

- 1 WORK
- 2 DATB
- 3 ITERB
- 4 DIE
- 5 PUNCH
- 6 F.E.M
- 7 MAIN
- 8 MODT

23



RC : R-COORDINATE OF THE POINT C
ZC : Z-COORDINATE OF THE POINT C

PLEASE ENTER YOUR DATA FOR RC,ZC >> ?

REMARK: Hit the ENTER key if you don't use FUNCTION-KEYS for the second prompt

- 1 WORK
- 2 DATB
- 3 ITERB
- 4 DIE
- 5 PUNCH
- 6 F.E.M
- 7 MAIN
- 8 MODT



DIE MENU

- A> R and Z COORDINATES OF THE DIE CONTOUR POINTS
- B> RADA - OUTLET RADIUS OF THE DIE
- C> RADE - INLET RADIUS OF THE DIE
- D> INANL - NUMBER OF NODES WHICH MAY HIT THE DIE
- E> RETURN TO MODIFY MENU

ENTER YOUR CHOICE >>>>> ?

REMARK: Hit the ENTER key if you don't use FUNCTION-KEYS for the second prompt

1 WORK 2 DATB 3 ITERB 4 DIE 5 PUNCH 6 F.E.M 7 MAIN 8 MODT

DIE CONTACT DATA

INANL=NUMBER OF THE NODES WHICH MAY HIT THE DIE

Please enter INANL >>>>>?

REMARK: Hit the ENTER key if you don't use FUNCTION-KEYS for the second prompt

1 WORK 2 DATB 3 ITERB 4 DIE 5 PUNCH 6 F.E.M 7 MAIN 8 MODT

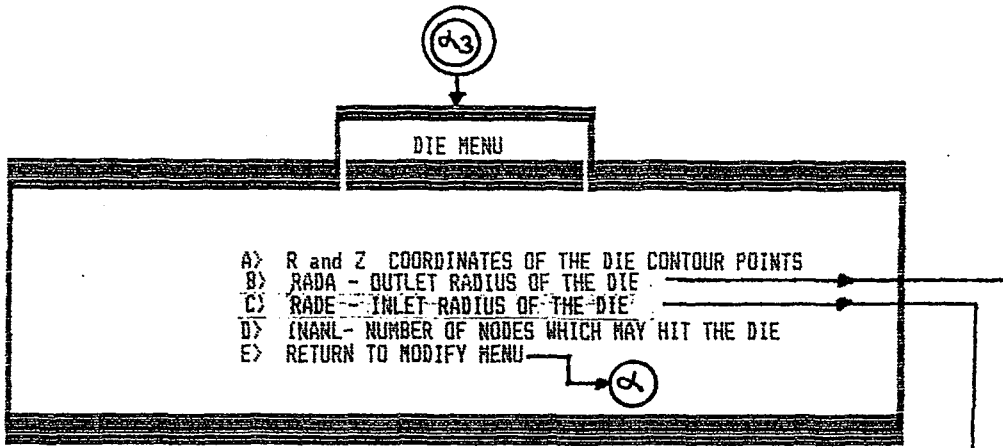
DIE CONTACT DATA

NANL(J) = NODE NUMBERS

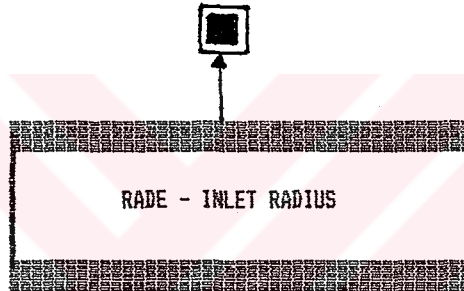
NANL(1)=?

REMARK: Hit the ENTER key if you don't use FUNCTION-KEYS for the second prompt

1 WORK 2 DATB 3 ITERB 4 DIE 5 PUNCH 6 F.E.M 7 MAIN 8 MODT

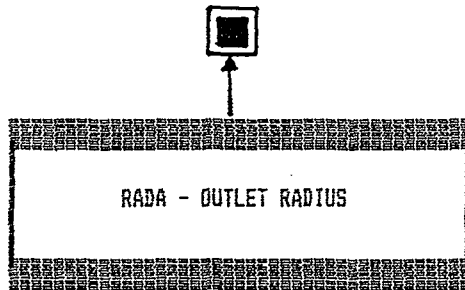


ENTER YOUR CHOICE >>>>> ?
 REMARK: Hit the ENTER key if you don't use FUNCTION-KEYS for the second prompt
 1 WORK 2 DATB 3 ITERB 4 DIE 5 PUNCH 6 F.E.M 7 MAIN 8 MODT



PLEASE ENTER YOUR DATA FOR RADE >>> ?

REMARK: Hit the ENTER key if you don't use FUNCTION-KEYS for the second prompt
 1 WORK 2 DATB 3 ITERB 4 DIE 5 PUNCH 6 F.E.M 7 MAIN 8 MODT



PLEASE ENTER YOUR DATA FOR RADA >>>>?

REMARK: Hit the ENTER key if you don't use FUNCTION-KEYS for the second prompt
 1 WORK 2 DATB 3 ITERB 4 DIE 5 PUNCH 6 F.E.M 7 MAIN 8 MODT

2

PUNCH MENU

- A> ZEND
- B> IECK
- C> IECK2
- D> ANF
- E> RETURN TO MODIFY MENU

ENTER YOUR CHOICE >>>>> ?

REMARK: Hit the ENTER key if you don't use FUNCTION-KEYS for the second prompt
1 WORK 2 DATB 3 ITERB 4 DIE 5 PUNCH 6 F.E.M 7 MAIN 8 MODT



ANF : PUNCH INCREMENT

PLEASE ENTER YOUR DATA FOR ANF >>?

REMARK: Hit the ENTER key if you don't use FUNCTION-KEYS for the second prompt
1 WORK 2 DATB 3 ITERB 4 DIE 5 PUNCH 6 F.E.M 7 MAIN 8 MODT



IECK2 : NUMBER OF D.O.F FOR THE GIVEN PUNCH VELOCITY

PLEASE ENTER YOUR DATA FOR IECK2 >>?

REMARK: Hit the ENTER key if you don't use FUNCTION-KEYS for the second prompt
1 WORK 2 DATB 3 ITERB 4 DIE 5 PUNCH 6 F.E.M 7 MAIN 8 MODT

2

PUNCH MENU

- A> ZEND
- B> IECK
- C> IECK2
- D> ANF
- E> RETURN TO MODIFY MENU

2

ENTER YOUR CHOICE >>>>>> ?

REMARK: Hit the ENTER key if you don't use FUNCTION-KEYS for the second prompt
1 WORK 2 DATB 3 ITERB 4 IE 5 PUNCH 6 F.E.M 7 MAIN 8 MODT

█

IECK : NUMBER OF THE NODE AT THE END OF THE PUNCH

PLEASE ENTER YOUR DATA FOR IECK >>> ?

REMARK: Hit the ENTER key if you don't use FUNCTION-KEYS for the second prompt
1 WORK 2 DATB 3 ITERB 4 IE 5 PUNCH 6 F.E.M 7 MAIN 8 MODT

█

ZEND - PUNCH DISPLACEMENT AT WHICH COMPUTATION WILL CEASE

PLEASE ENTER YOUR DATA FOR ZEND >>> ?

REMARK: Hit the ENTER key if you don't use FUNCTION-KEYS for the second prompt
1 WORK 2 DATB 3 ITERB 4 IE 5 PUNCH 6 F.E.M 7 MAIN 8 MODT



F.E.M.

- A> FLAGS (IDENGE, IHEPSI)
- B> NUMBER OF NODES IN R & Z DIRECTIONS
- C> ITOP - KEYWORD FOR THE ELEMENT TYPE
- D> INKAOT - DEFORMED ELEMENT NUMBER
- E> COORDINATES OF THE NODES
- F> RETURN TO MODIFY MENU



ENTER YOUR CHOICE >>>>>>?

REMARK: Hit the ENTER key if you don't use FUNCTION-KEYS for the second prompt

1 WORK 2 DATB 3 ITERB 4 DIE 5 PUNCH 6 F.E.M 7 MAIN 8 MODT

NODES

- A> IRN - NUMBER OF NODES IN R-DIRECTION
- B> IZN - NUMBER OF NODES IN Z-DIRECTION
- C> RETURN TO THE F.E.M. MENU

ENTER YOUR CHOICE >>>>>> ?

REMARK: Hit the ENTER key if you don't use FUNCTION-KEYS for the second prompt

1 WORK 2 DATB 3 ITERB 4 DIE 5 PUNCH 6 F.E.M 7 MAIN 8 MODT

NUM NODES

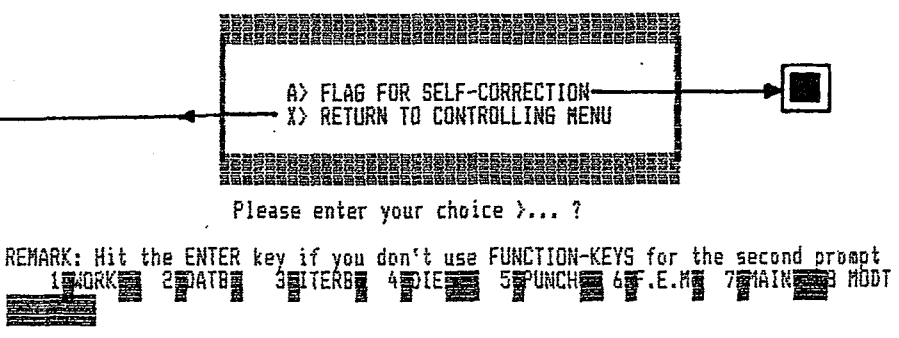
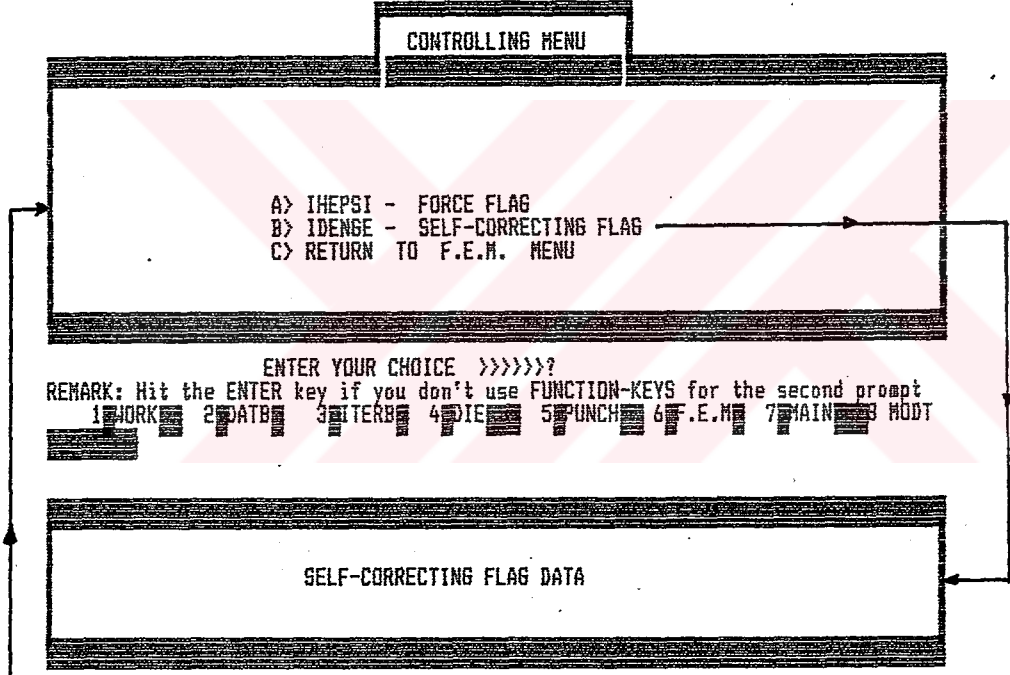
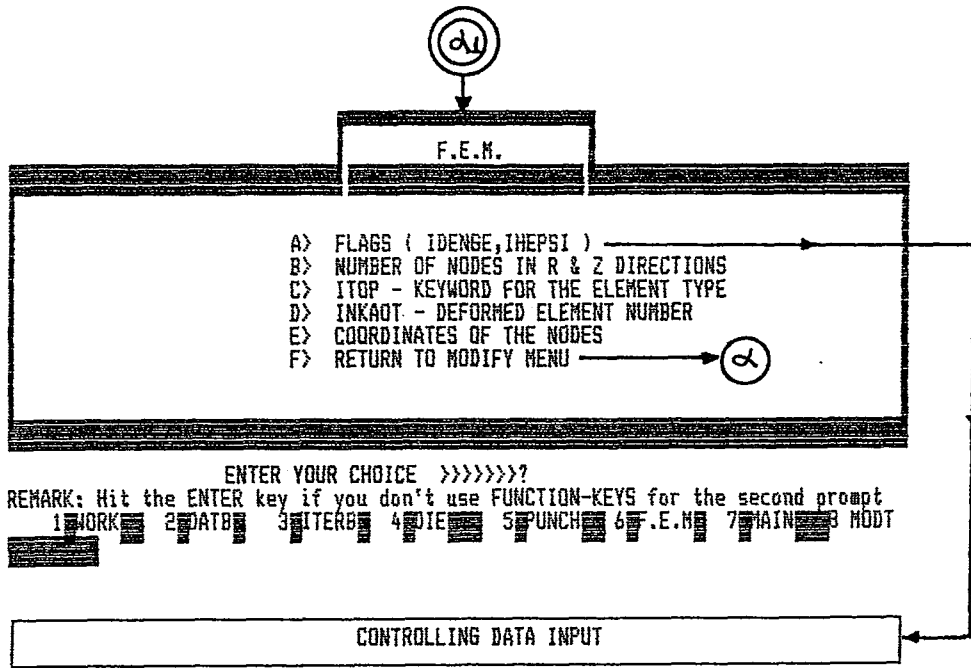
- A> IX
- B> IY
- C> RETURN TO THE F.E.M. MENU



ENTER YOUR CHOICE >>>>>> ?

REMARK: Hit the ENTER key if you don't use FUNCTION-KEYS for the second prompt

1 WORK 2 DATB 3 ITERB 4 DIE 5 PUNCH 6 F.E.M 7 MAIN 8 MODT



(d)

F.E.M.

- A) FLAGS (IDENGE, IHEPSI)
- B) NUMBER OF NODES IN R & Z DIRECTIONS
- C) ITOP - KEYWORD FOR THE ELEMENT TYPE
- D) INKAOT - DEFORMED ELEMENT NUMBER
- E) COORDINATES OF THE NODES
- F) RETURN TO MODIFY MENU

(d)

ENTER YOUR CHOICE >>>>>?

REMARK: Hit the ENTER key if you don't use FUNCTION-KEYS for the second prompt
1 WORK 2 DATB 3 ITERB 4 DIE 5 PUNCH 6 F.E.M 7 MAIN 8 MODT

DEFORMED ELEMENT DATA

INKAOT=NUMBER OF EXTREMELY DEFORMED ELEMENTS

Please enter your data ????

REMARK: Hit the ENTER key if you don't use FUNCTION-KEYS for the second prompt
1 WORK 2 DATB 3 ITERB 4 DIE 5 PUNCH 6 F.E.M 7 MAIN 8 MODT

DEFORMED ELEMENT DATA

NKAOT(J)=if INKAOT is not equal to zero number
of the extremely deformed elements

NKAOT(2)= ?

REMARK: Hit the ENTER key if you don't use FUNCTION-KEYS for the second prompt
1 WORK 2 DATB 3 ITERB 4 DIE 5 PUNCH 6 F.E.M 7 MAIN 8 MODT

Ⓛ

F.E.M.

- A> FLAGS (IDENGE, INEPSI)
- B> NUMBER OF NODES IN R & Z DIRECTIONS
- C> ITOP - KEYWORD FOR THE ELEMENT TYPE
- D> INKAOT - DEFORMED ELEMENT NUMBER
- E> COORDINATES OF THE NODES
- F> RETURN TO MODIFY MENU

Ⓛ

ENTER YOUR CHOICE >>>>>>?

REMARK: Hit the ENTER key if you don't use FUNCTION-KEYS for the second prompt
1 WORK 2 DATB 3 ITERB 4 DIE 5 PUNCH 6 F.E.M 7 MAIN 8 MODT

■

ITOP : KEYWORD FOR ELEMENT TYPE

PLEASE ENTER YOUR DATA FOR ITOP >>> ?

REMARK: Hit the ENTER key if you don't use FUNCTION-KEYS for the second prompt
1 WORK 2 DATB 3 ITERB 4 DIE 5 PUNCH 6 F.E.M 7 MAIN 8 MODT

Appendix E

SOURCE LISTING OF THE MODIFIED MAIN PROGRAM OF EPDAN

```

C*****
C*****
C*   ELASTIC-PLASTIC DEFORMATION   *
C*   ANALYSIS PROGRAM EPDAN       *
C*****
C*****
C
C
C   PROGRAM EPDAN
C   IMPLICIT REAL*8 (A-H,O-Z), INTEGER*4 (I-N)
C   DIMENSION IA(72),IRA(6)
C   INCLUDE 'CCOM1.FOR'
C   COMMON//EA(10000)
C
C
C   OPEN ALL THE FILES
C   =====
C   CALL DPECL0
C
C   INCLUDE 'CCOM2.FOR'
C
C   KEG2=KEG*2
C   KKG4=KKG*2
C   KNEG2=KNEG*2
C   L1=1
C   L2=KEG2+L1
C   L3=KEG2+L2
C   L4=KEG2+L3
C   L5=KEG2+L4
C   L6=KKG4+L5
C   L7=KKG4+L6
C   L8=KNEG2+L7
C   L9=KIG+L8
C   L10=KIG+L9
C   L11=KIG+L10
C   L12=KKG+L11
C   L13=KKG+L12
C   L14=KKG2+L13
C   L15=KKG2+L14
C   L16=KKG2+L15
C   L17=KKG2+L16
C   L18=KKG2+L17
C   L19=KKG2+L18
C   L20=KKG2+L19
C   L21=KIG+L20
C   L22=KIG+L21
C   L23=KIG+L22
C   L24=KIG+L23
C   L25=KIG+L24
C   L26=KIG+L25
C   L27=KIG+L26
C   L28=KIG+L27
C   L29=KIG+L28
C   L30=KIG+L29
C   L31=KIG+L30
C   L32=KIG+L31
C
C   CALL CHKDYN
C   KC=1
C   CALL CLOCK(KC)
C   CALL INPUT(EA(L1),EA(L2),EA(L3),EA(L4),EA(L11),EA(L12),
C   1  FLITY,ITOP,IDOL,ILUDW,ITOPVA,IKEINR,RESTART,MALLE,
C   2  SAVE,PLOT,FLITYP,DENGE)
C   IF(.NOT.RESTART) GO TO 30
C   IR=1
C   CALL STORAGE(EA(L1),EA(L2),EA(L3),EA(L4),
C   a  EA(L5),EA(L6),EA(L7),EA(L8),EA(L9),EA(L10),EA(L11),

```

```

1 EA(L12),EA(L13),EA(L14),EA(L15),EA(L18),
2 EA(L17),EA(L20),EA(L21),EA(L22),EA(L23),
3 EA(L24),EA(L25),EA(L26),EA(L27),EA(L28),
4 EA(L29),EA(L30),EA(L31),KA,FREE)
630 CONTINUE
   IBER=FALSE.
   CALL KONTUR(IDEK,ZZ,RR,IBER)

   CALL FILL(EA(L1),EA(L2),EA(L3),EA(L4),EA(L6),EA(L11),
1 EA(L12),EA(L17),RGAUSS,IDEK,IBER)
   KO=1
   CALL OUTPUT(EA(L1),EA(L2),EA(L3),EA(L4),EA(L6),EA(L8),
1 EA(L7),EA(L11),EA(L12),EA(L14),EA(L16),EA(L18),EA(L19),
2 EA(L20),EA(L21),EA(L22),EA(L23),EA(L28),EA(L29),EA(L30),
3 EA(L31),KO,IRA,IKEINR,IXU,IYU,IX86,IY86,ISCHL,IXUG,
4 IY86,FLITYP)

C
C
C**** LINEARIZE FLOW-CURVE (stress-strain curve)
C=====
C      IF(.NOT.RESTART) CALL LINIZER(EA(L10),EA(L15))
C
C**** COMPUTE ELASTICITY MATRIX
C=====
C      CALL ELACDN(EA(L6),EA(L13),EA(L14),EA(L21),EA(L23),RESTART)
C
C      IF(PLOT) CALL PLOTTER(EA(L1),EA(L2),EA(L3),EA(L4),
1 EA(L8),EA(L11),EA(L12),EA(L14),EA(L22),EA(L23),
2 EA(L24),EA(L25),EA(L26),EA(L27),EA(L28),EA(L29),
3 EA(L30),EA(L31))
C      KC=2
C      CALL CLOCK(KC)

C
C *****
C *
C *      START INCREMENTAL COMPUTATIONS
C *
C *****
C
C *****
C *
C *      START ITERATION LOOP
C *
C *****
C
C
723 CALL BEGIN(EA(L8),EA(L9),LOGANL)
C
705 ITER=ITER+1
DO 704 I=1,IBS
704 BKN(I)=0.
C**
C
C1* CONSTRUCT STIFFNESS MATRIX
C=====
C      CALL STIFFN(EA(L1),EA(L2),EA(L3),EA(L4),EA(L8),EA(L9),EA(L10),
1 EA(L11),EA(L12),EA(L14),EA(L15),EA(L20),EA(L21),EA(L22),EA(L23),
2 EA(L24),EA(L25),EA(L26),EA(L27),EA(L28),EA(L29),EA(L30),EA(L31),
3 RGAUSS,LKAGT,ISSTEIF)
C**
C**
C
C2* BOUNDARY CONDITIONS
C=====
C**
C      CALL BCND(EA(L6),EA(L7),EA(L11),EA(L12),EA(L13),EA(L14),
1 EA(L17),EA(L18),EA(L19),EA(L24),EA(L25),
2 ENTER,IDEK,IBER,ANFANG)
C
C**
C
C      CALL SOLVE(EA(L5),EA(L6),EA(L7),EA(L13),EA(L14),EA(L16),EA(L17),
1 EA(L18),EA(L24),EA(L25),NIPODE,LOCAL,FREE,MALLE,RGAUSS)
IF(ITER.LT.ITLIMIT) GO TO 705

C
C *****
C *
C *      END OF ITERATION LOOP
C *
C *****
C
C
C

```

```

C *****
C *
C * SOLUTION IS SATISFACTORY : COMPUTE THE NEW STATE *
C *
C *****
C
C
C**
C
C1* COMPUTE STRAIN AND ROTATION - INCREMENTS
C=====
C**
    CALL INCSTR(EA(L1),EA(L2),EA(L3),EA(L4),EA(L8),EA(L9),
    1 EA(L10),EA(L11),EA(L12),EA(L14),EA(L15),EA(L20),EA(L21),
    2 EA(L22),EA(L23),EA(L24),EA(L25),EA(L26),EA(L27),EA(L28),
    3 EA(L29),EA(L30),EA(L31),LKAOT,ISSTEIF)
C
C    CHECK BOUNDARY CONDITIONS
C=====
C    CHECK ENTRANCE INTO DIE CONTOUR
C=====
C    CALL DIECHK(EA(L6),EA(L11),EA(L12),EA(L18),EA(L19),
    1 ZPOP,RDUMNY,RNCU,NOTFALL,IDEC,IBER,ENTER,FREE,BEENDE,LOGANL)
C
C    CALL TESTDEF(EA(L1),EA(L2),EA(L3),EA(L4),EA(L6),
    1 EA(L11),EA(L12),EA(L13),EA(L14),EA(L16),EA(L18),
    2 EA(L19),EA(L28),EA(L29),EA(L30),EA(L31),ZPOP,
    3 RDUMNY,LKAOT,IBER,IDEC,FREE,DENGE)
C
C
C *****
C *
C * END OF INCREMENTAL COMPUTATIONS *
C *
C *****
C
C    DECISION FOR WRITING THE PLOTTER - FILE
C=====
C    KC=4
C    CALL CLOCK(KC)
C    KA=2
C    IF(SAVE.AND.MOD(ILAST,NSABST).EQ.0)
    a CALL STORAGE(EA(L1),EA(L2),EA(L3),EA(L4),EA(L6),EA(L8),
    1 EA(L9),EA(L10),EA(L11),EA(L12),EA(L13),EA(L14),EA(L15),
    2 EA(L18),EA(L19),EA(L20),EA(L21),EA(L22),EA(L23),EA(L24),
    3 EA(L25),EA(L26),EA(L27),EA(L28),EA(L29),EA(L30),EA(L31),
    4 KA,FREE)
C
C    IF(IMALEN.EQ.TLMA.AND.PLOT) CALL PLOTTER(EA(L1),EA(L2),
    1 EA(L3),EA(L4),EA(L8),EA(L11),EA(L12),EA(L14),EA(L22),
    2 EA(L23),EA(L24),EA(L25),EA(L26),EA(L27),EA(L28),EA(L29),
    3 EA(L30),EA(L31))
C
C    CALL CHK(EA(L12),ZPOP,IRDUM,BEENDE,NOTFALL)
    IF(IRDUM.EQ.1) GOTO 723
C
C 724    KO=2
    CALL OUTPUT(EA(L1),EA(L2),EA(L3),EA(L4),EA(L6),
    1 EA(L8),EA(L7),EA(L11),EA(L12),EA(L14),EA(L16),
    2 EA(L18),EA(L19),EA(L20),EA(L21),EA(L22),EA(L23),
    3 EA(L28),EA(L29),EA(L30),EA(L31),
    4 KO,IRA,IKEINR,IXU,IYU,IXBG,IYBG,
    5 ISCHL,IXUG,IYUG,FLITYP)
C
C    CALL VOLUMEC(EA(L1),EA(L2),EA(L3),EA(L4),EA(L11),EA(L12),
    1 ZPOP,BEENDE,NOTFALL)
C
C    IF(ISTOP.EQ.1) GOTO 444
    GO TO 723
C
C    CLOSE THE FILES
C=====
C 444    CALL OPECLO
    END
C
CC
CC
CC

```

Appendix F

SOURCE LISTING OF D A T O R

```

2 REM =====
3 REM =====
4 REM =
5 REM =   DATOR : DATA GENERATOR PROGRAM   =
6 REM =
7 REM =====
8 REM =====
9 REM *****
10 REM *****
11 REM *****
12 REM * P R O G - M A I N *
13 REM *****
14 REM *****
15 REM
22 REM *****
23 REM =====
24 REM *****
26 REM * MAIN-MENU COMMUNICATE WITH THE GORG.BAS,GALL.BAS,GORG.BAS,GMESH.BAS **
27 REM * FOR THE FOLLOWINGS : **
28 REM * 1> USER - MACHINE INTERACTION FOR MODIFICATION OF THE DATA FILE **
30 REM * 2> USER - MACHINE INTERACTION FOR READING ALL DATA (First case) **
32 REM * 3> USER - MACHINE INTERACTION FOR FILE ORGANIZATION **
33 REM * 4> MESH GENERATION. **
34 REM *****
35 REM =====
45 REM
46 COMMON NKAOT(1),NANL(1),RK(1),ZK(1),ADR(1),NAW$(1)
47 COMMON VV(1),UV$(1),IV(1),EV(1),WA$(1),WA1$(1),AA(2),BB$(2)
48 COMMON PAR,R1$,R2$,X3,IX1,IX2,X1,X2
49 COMMON MMA$(2),NMB$(2),DUM(1),SE(2)
50 COMMON RET,X4
51 COMMON FZ(1),FR(1),NPOS(1),NACT(1),NAPR(1),IU(1),JU(1),KU(1),MU(1),NSTEM(1)
52 COMMON ISO
60 CLS:KEY OFF:X3=100:X1=100:X2=10:X4=150
62 IF PAR=7 THEN GOTO 135
63 PRINT CHR$(7);:COLOR 7,0:CLS:GOSUB 2092:COLOR 0,7
64 FOR I=4 TO 17
65 LOCATE I,3:PRINT STRING$(76,178);
75 NEXT I
80 REM FIRST APPEARANCE ON THE SCREEN
82 LOCATE 11,22:PRINT"
83 LOCATE 14,22:PRINT"
85 LOCATE 15,22:PRINT" ( D A T A   G E N E R A T O R )
86 LOCATE 16,22:PRINT" For the Finite Element Code   EPDAN
87 LOCATE 11,22:PRINT"
88 LOCATE 17,22:PRINT"      Version : 17.2.1989
89 LOCATE 18,22:PRINT"      By : Ezgi Gunay
95 LOCATE 12,22:PRINT"      D   A   T   O   R
97 LOCATE 13,22:PRINT"
100 COLOR 7,0
110 REM
115 A$=INKEY$:IF A$="" THEN 115
116 REM Output file names DATOR.FOR and DATOR.DAT are introducing
117 CLS:GOSUB 8000
124 CLS:GOSUB 8055
125 CLS:IX2=5:IX1=4
135 COLOR 7,0:LOCATE 8,1:PRINT STRING$(80,219);
145 FOR I=9 TO 16
155 LOCATE I,1:PRINT CHR$(221);:LOCATE I,80:PRINT CHR$(222);
165 NEXT I
175 LOCATE 17,1:PRINT STRING$(80,219);
185 LOCATE 5,30:PRINT STRING$(23,219);
195 LOCATE 7,30:PRINT CHR$(221);:LOCATE 7,52:PRINT CHR$(222);
196 LOCATE 6,30:PRINT CHR$(221);:LOCATE 6,52:PRINT CHR$(222);
205 LOCATE 8,30:PRINT CHR$(221);:LOCATE 8,52:PRINT CHR$(222);
209 COLOR 0,7
210 REM ----- MAIN MENU -----
215 LOCATE 7,37:PRINT"MAIN MENU";

```



```

224 LOCATE 10,18:PRINT"
225 LOCATE 11,18:PRINT"      A> DO YOU WANT TO MODIFY YOUR DATA ?
235 LOCATE 12,18:PRINT"      B> DO YOU WANT TO READ ALL DATA ?
237 LOCATE 13,18:PRINT"      C> DO YOU WANT TO ORGANIZE DATOR.FOR ?
240 LOCATE 14,18:PRINT"      D> STOP THE PROGRAM      D A T O R
241 LOCATE 15,18:PRINT"
245 LOCATE 19,20:PRINT"INPUT YOUR CHOICE ";
246 COLOR 7,0:LOCATE 19,40:COLOR 28,0:PRINT">>>>>";:COLOR 7,0
247 COLOR 0,7:LOCATE 19,48:INPUT A1$
250 COLOR 7,0
253 REDIM BB$(X1,X2),NMA$(X1,X2),NMB$(X1,X2),DUM(X1),SE(X1,X2)
255 REDIM NKAOT(X3),NANL(X3),RK(X4),ZK(X4),ADR(X3),NAN$(X3)
250 REDIM VV(X3),UV$(X3),IV(X3),EV(X3),WA$(X3),WA1$(X3),AA(X1,X2)
262 REDIM FZ(X4),FR(X4),NPOS(X4),NACT(X4),NAPR(X4)
264 REDIM IU(X4),JU(X4),KU(X4),MU(X4),NSTEM(X4)
295 IF A1$="A" OR A1$="a" THEN GOTO 363
305 IF A1$="B" OR A1$="b" THEN GOTO 367
315 IF A1$="C" OR A1$="c" THEN GOTO 368
345 IF A1$="D" OR A1$="d" THEN GOTO 400
355 GOTO 125
356 REM -----
357 REM ----- CONNECTION WITH THE PROGRAMS -----
358 REM ----- 1) MODIFY          2) FIRST -----
359 REM ----- 3) ORGANIZE       4) MESH   5) ISOMESH -----
360 REM -----
363 CHAIN "C:MODIFY.EXE"
365 REM
367 CHAIN "C:FIRST.EXE"
368 RET=1
374 CHAIN "C:ORGANIZE.EXE"
375 REM
400 CLS
405 GOSUB 2000:GOSUB 9000:LOCATE 4,19:PRINT" End of the INTERACTIVE BASIC program";
406 LOCATE 5,30 : COLOR 28,0:PRINT" * D A T O R * "; : COLOR 7,0
408 LOCATE 15,20:COLOR 28,0:PRINT">>>>";:COLOR 7,0 :PRINT " PRESS ENTER KEY TO STOP ";:COLOR 28,0:PRINT"<<<<:"
410 REM
420 LOCATE 20,10:AS$=INKEY$:IF AS$="" THEN 420
425 CLS
450 REM
455 REM ----- END OF THE MAIN PROGRAM G M A I N . B A S -----
460 REM -----
470 REM
480 REM
490 REM STOP
500 END
510 REM -----
511 REM -----
515 REM ----- TABLE DRAWING SUBROUTINES -----
517 REM -----
519 REM -----
1360 REM -----
1365 REM ----- MENU DRAWING (1) -----
1370 REM -----
1375 REM
2000 CLS:LOCATE 12,1:PRINT STRING$(80,219);
2002 FOR I=13 TO 17
2004 LOCATE I,1:PRINT CHR$(221);:LOCATE I,80:PRINT CHR$(222);
2006 NEXT I
2008 LOCATE 18,1:PRINT STRING$(80,219);
2010 RETURN
2015 END
2092 REM -----
2093 REM ----- MENU DRAWING (2) -----
2094 REM -----
2095 REM
2100 CLS
2110 LOCATE 4,1:PRINT STRING$(80,178);
2112 FOR I=5 TO 18
2114 LOCATE I,1:PRINT CHR$(221);:LOCATE I,80:PRINT CHR$(222);
2116 NEXT I
2118 LOCATE 19,1:PRINT STRING$(80,178);
2210 RETURN
6900 REM -----
7000 REM ----- MENU DRAWING (3) -----

```

```

7100 REM -----
8000 CLS:GOSUB 2092
8005 LOCATE 22,18:COLOR 28,0:PRINT"WARNING : The File Must Exist ";:COLOR 7,0
8010 COLOR 0,7
8014 LOCATE 7,18:PRINT STRING$(44,177);
8015 LOCATE 8,18:PRINT"
8016 LOCATE 9,18:PRINT"
8018 LOCATE 10,18:PRINT" Reading of the data from the SCREEN and
8022 LOCATE 11,18:PRINT" CREATION of the output-file
8023 LOCATE 12,18:PRINT"
8024 LOCATE 13,18:PRINT"
8025 LOCATE 14,18:PRINT" (Default : DATOR.INP)
8026 LOCATE 15,18:PRINT"
8036 LOCATE 16,18:PRINT STRING$(44,177);
8037 LOCATE 21,18:PRINT"Please enter the output file name ";:COLOR 7,0
8040 COLOR 28,0 : PRINT">>>>";
8043 COLOR 7,0 : COLOR 0,7
8044 INPUT R1$ : COLOR 7,0
8045 IF R1$ <> "" THEN GOTO 8050
8049 R1$="DATOR.INP"
8050 RETURN
8055 REM -----
8056 REM ----- MENU DRAWING (4) -----
8057 REM -----
8060 CLS:GOSUB 2092:COLOR 0,7
8063 LOCATE 22,18:COLOR 28,0:PRINT"WARNING : The File Must Exist ";
8064 COLOR 7,0:COLOR 0,7
8065 LOCATE 7,18:PRINT STRING$(44,177);
8067 LOCATE 8,18:PRINT"
8068 LOCATE 9,18:PRINT"
8070 LOCATE 10,18:PRINT" The temporary file containing all
8073 LOCATE 11,18:PRINT" arrays will be created
8075 LOCATE 12,18:PRINT"
8076 LOCATE 13,18:PRINT"
8077 LOCATE 14,18:PRINT" (Default : DATOR.DAT)
8078 LOCATE 15,18:PRINT"
8080 LOCATE 16,18:PRINT STRING$(44,177);
8082 LOCATE 21,18:PRINT"Please enter the file name " ;:COLOR 7,0
8083 COLOR 28,0 : PRINT">>>>";
8084 COLOR 7,0 : COLOR 0,7
8085 INPUT R2$ : COLOR 7,0
8087 IF R2$ <> "" THEN GOTO 8108
8089 R2$="DATOR.DAT"
8108 RETURN
9000 LOCATE 1,1:PRINT STRING$(80,219);
9002 FOR I=2 TO 7
9004 LOCATE I,1:PRINT CHR$(221);LOCATE I,80:PRINT CHR$(222);
9006 NEXT I
9008 LOCATE 8,1:PRINT STRING$(80,219);
9010 RETURN
9015 END
1 REM
2 REM
3 REM
4 REM MODULE : O R G A N I Z A T I O N
5 REM
6 REM
7 REM
8 REM
10 REM =====
20 REM *****
30 REM * SUBROUTINES FOR THE ORGANIZATION OF THE FILE *
38 REM * D A T O R . F O R *
40 REM *****
45 REM =====
50 COMMON NKAOT(1),NANL(1),RK(1),ZK(1),ADR(1),NAW$(1)
52 COMMON VV(1),UV$(1),IV(1),EV(1),WA$(1),WA1$(1),AA(2),BB$(2)
54 COMMON PAR,R1$,R2$,X3,IX1,IX2,X1,X2
55 COMMON NMA$(2),NMB$(2),DUM(1),SE(2)
57 COMMON RET,X4
58 COMMON FZ(1),FR(1),NPOS(1),NACT(1),NAPR(1),IU(1),JU(1),KU(1),MU(1),NSTEM(1)
59 COMMON ISO
60 CLS:KEY OFF
65 PAR=7
67 REM
68 REM
69 REM DYNAMIC ARRAY DIMENSIONING IS USED
72 REDIM BB$(X1,X2),NMA$(X1,X2),NMB$(X1,X2),DUM(X1),SE(X1,X2)
74 REDIM NKAOT(X3),NANL(X3),RK(X4),ZK(X4),ADR(X3),NAW$(X3)
76 REDIM VV(X3),UV$(X3),IV(X3),EV(X3),WA$(X3),WA1$(X3),AA(X1,X2)

```

```

77 REDIM FZ(X4),FR(X4),NPOS(X4),NACT(X4),NAPR(X4)
78 REDIM IU(X4),JU(X4),KU(X4),MU(X4),NSTEM(X4)
79 IF RET=5 OR ISO=5 THEN GOTO 84
80 GOSUB 1000 : IF P=2 THEN GOTO 210
81 OPEN R1$ FOR OUTPUT AS 2
82 TITLE$="TEST DATA FOR DIELESS EXTRUSION 5 x 19 ELEMENTS 29/07/1989"
83 REM ----- ORGANIZATION-MENU -----
84 RET=0: ISO=0:CLS:GOSUB 8000
85 IF DUMMY=1 THEN GOTO 90
86 IF DUMMY=2 THEN GOTO 96
87 IF DUMMY=3 THEN GOTO 130
88 IF DUMMY=4 THEN GOTO 200
89 IF DUMMY=5 THEN GOTO 165
90 CLS : GOSUB 8500
91 GOSUB 2500
92 PRINT #2,
94 REM ----- FORMAT-MENU -----
95 CLS : GOTO 84
96 CLS : GOSUB 12000
97 IF FOR1=1 THEN GOTO 105
98 IF FOR2=1 THEN GOTO 119
99 IF FOR3=1 THEN GOTO 150
100 GOTO 82
105 GOSUB 11000
106 PRINT #2,
107 GOTO 130
119 GOSUB 5000
121 REM ----- MESH-MENU -----
130 CLS : GOSUB 7000
131 REM
132 IF UFD1=1 THEN GOTO 135
133 IF UFD2=1 THEN GOTO 140
134 GOTO 84
135 CHAIN "C:MESH.EXE"
136 PRINT #2,
137 PRINT #2,
138 PRINT #2,
140 PRINT #2,
150 GOTO 84
165 REDIM BB$(X1,X2),NNA$(X1,X2),NHB$(X1,X2),DUM(X1),SE(X1,X2)
166 REDIM NKADT(X3),MANL(X3),RK(X4),ZK(X4),ADR(X3),NAW$(X3)
167 REDIM VV(X3),UV$(X3),IV(X3),EV(X3),WA$(X3),WA1$(X3),AA(X1,X2)
168 REDIM FZ(X4),FR(X4),NPOS(X4),NACT(X4),NAPR(X4)
169 REDIM IU(X4),JU(X4),KU(X4),MU(X4),NSTEM(X4)
170 CHAIN "C:ISOMESH.EXE"
200 CLOSE(2)
210 CHAIN "C:MAIN.EXE"
635 REM *****
700 REM SUBROUTINE FOR THE READING OF THE WHOLE ARRAYS
701 REM *****
703 OPEN R2$ FOR INPUT AS 1
704 KEZ=0
710 KEZ=KEZ+1
715 REM
716 INPUT #1,UV$(KEZ), IV(KEZ), EV(KEZ), VV(KEZ), ADR(KEZ), NAW$(KEZ),WA1$(KEZ),WA$(KEZ)
717 IF EOF(1) THEN 719
718 GOTO 710
719 CLOSE(1)
720 REM
721 RETURN
722 REM *****
725 REM SUBROUTINES FOR THE DRAWING OF THE TABLES
730 REM *****
800 LOCATE 2,1:PRINT STRING$(80,219);
810 FOR K=3 TO 7
815 LOCATE K,1:PRINT CHR$(221);:LOCATE K,80:PRINT CHR$(222);
820 NEXT K
825 LOCATE 8,1:PRINT STRING$(80,219);
830 LOCATE 5,8:PRINT"PLEASE ENTER 1) THE NAME OF THE NEW DATA ";
835 LOCATE 6,24:PRINT"2) LOCATION NUMBERS IN THE DATA FILE ";
840 RETURN
900 REM *****
901 REM FUNCTION KEY SUBROUTINE
902 REM TO SAVE THE INPUT DATA
903 REM *****
905 LOCATE 18,1:PRINT CHR$(218)+STRING$(78,196)+CHR$(191);
908 LOCATE 19,1:PRINT CHR$(179);:LOCATE 19,80:PRINT CHR$(179);
910 LOCATE 20,1:PRINT CHR$(192)+STRING$(78,196)+CHR$(217);
914 COLOR 0,7
915 LOCATE 19,10:PRINT"1) SAVE ";:LOCATE 19,30:PRINT"2) INPUT AGAIN";
920 LOCATE 19,50:PRINT"3) START AGAIN";

```



```

2550 ML=M
2555 NEXT M
2560 IF IK>1 THEN GOTO 2705
2565 IF IK=1 THEN GOSUB 2625
2570 IF T$="" THEN GOTO 2580
2575 GOTO 2622
2580 FOR LL=1 TO X2
2585 K=N: KK=LL
2590 A$=NMA$(N,LL)
2592 IF A$="" THEN GOTO 2600
2596 KKK=1
2597 GOTO 3000
2600 NEXT LL
2622 GOTO 4300
2623 PR=1
2625 FOR LL=1 TO ML-1
2630 K=N : KK=LL
2635 A$=NMA$(N,LL):IF A$="" THEN GOTO 2645
2637 KKK=2
2640 GOTO 3000
2642 REM
2645 NEXT LL
2650 K=N : KK=ML
2655 A$=NMB$(N,ML)
2657 KKK=3
2660 GOTO 3000
2663 REM
2665 FOR LL=ML+1 TO X2
2670 K=N : KK=LL :A$=NMA$(N,LL)
2673 KKK=4
2675 GOTO 3000
2676 REM
2680 NEXT LL
2685 REM
2690 GOTO 2575
2695 REM
2700 REM
2705 FOR LL=1 TO X2
2710 K=N : KK=LL
2715 IF DUM(LL) > 0 GOTO 2735
2720 A$=NMA$(N,LL)
2723 KKK=5
2725 GOTO 3000
2726 REM
2730 REM
2735 A$=NMB$(N,LL)
2737 KKK=6
2740 GOTO 3000
2742 REM
2745 NEXT LL
2755 GOTO 2575
3000 REM =====
3005 REM ====          CHECKING OF THE FORMATS          =====
3007 REM =====
3008 Y=K+SUM
3009 REM
3010 IF N=1 AND LL=1 THEN GOTO 3020
3012 IF LL > 1 THEN GOTO 3020
3014 PRINT #2,
3020 IF A$="INKAOT" THEN GOTO 3400
3030 IF A$="INANL" THEN GOTO 3400
3040 IF A$="ITANL" THEN GOTO 3400
3050 IF A$="I" THEN GOTO 3400
3060 IF A$="J" THEN GOTO 3400
3070 IF A$="NSABST" THEN GOTO 3400
3080 IF A$="DPHI" THEN GOTO 3420
3090 IF A$="PHIMAX" THEN GOTO 3420
3100 IF A$="IDINKF" THEN GOTO 3400
3105 IF A$="IDINKV" THEN GOTO 3400
3110 IF A$="IDINKR" THEN GOTO 3400
3115 IF A$="IDINKS" THEN GOTO 3400
3120 IF A$="SKFNUL" THEN GOTO 3440
3125 IF A$="EM" THEN GOTO 3450
3130 IF A$="PD" THEN GOTO 3480
3135 IF A$="SK" THEN GOTO 3480
3140 IF A$="IFLITY" THEN GOTO 3500
3145 IF A$="CEE" THEN GOTO 3440

```

```

3155 IF A$="ENN" THEN GOTO 3440
3160 IF A$="IMA" THEN GOTO 3400
3165 IF A$="ILMA" THEN GOTO 3400
3170 IF A$="IHALT" THEN GOTO 3400
3180 IF A$="IHEPSI" THEN GOTO 3400
3190 IF A$="IDENGE" THEN GOTO 3400
3200 IF A$="ITOP" THEN GOTO 3500
3205 IF A$="IWRITEG" THEN GOTO 3400
3210 IF A$="ZEND" THEN GOTO 3420
3215 IF A$="IX" THEN GOTO 3400
3220 IF A$="IY" THEN GOTO 3400
3225 IF A$="RB" THEN GOTO 3440
3230 IF A$="ZB" THEN GOTO 3440
3235 IF A$="RC" THEN GOTO 3440
3240 IF A$="ZC" THEN GOTO 3440
3245 IF A$="RD" THEN GOTO 3440
3250 IF A$="ZD" THEN GOTO 3440
3255 IF A$="RA" THEN GOTO 3440
3260 IF A$="ZA" THEN GOTO 3440
3265 IF A$="RADE" THEN GOTO 3480
3270 IF A$="RADA" THEN GOTO 3480
3275 IF A$="IRN" THEN GOTO 3400
3280 IF A$="IZN" THEN GOTO 3400
3285 IF A$="LIMVER" THEN GOTO 3400
3290 IF A$="ILAST1" THEN GOTO 3400
3295 IF A$="LIMVER1" THEN GOTO 3400
3297 IF A$="ILAST2" THEN GOTO 3400
3300 IF A$="LIMVER2" THEN GOTO 3400
3305 IF A$="IECK" THEN GOTO 3400
3315 IF A$="IECK2" THEN GOTO 3400
3325 IF A$="ANF" THEN GOTO 3420
3335 IF A$="NKAOT" THEN GOTO 3600
3345 IF A$="NANL" THEN GOTO 3600
3355 IF A$="RK" THEN GOTO 3600
3365 IF A$="ZK" THEN GOTO 3600
3370 GOTO 4200
3400 PRINT #2,USING"#####";AA(Y,KK);
3410 PR1=1;GOTO 4200
3420 PRINT #2,USING"####.####";AA(Y,KK);
3430 PR1=1;GOTO 4200
3440 PRINT #2,USING"#####.###";AA(Y,KK);
3450 PR1=1;GOTO 4200
3460 PRINT #2,USING"#####.#";AA(Y,KK);
3470 PR1=1;GOTO 4200
3480 PRINT #2,USING"#####.###";AA(Y,KK);
3490 PR1=1;GOTO 4200
3495 REM *****
3497 REM CHARACTER FORMAT
3500 REM *****
3501 PRINT #2,USING"&";BB*(K,KK);
3502 GOTO 4200
3503 REM
3510 REM
3600 REM =====
3605 REM ===== DEFINITION OF FORMATS FOR THE ARRAYS =====
3610 REM =====
3611 REM
3613 IF PR=0 AND LL=1 THEN GOTO 3615
3614 PRINT #2,
3615 NS=SE(K,KK)
3616 MS=NS+SUN+K-1
3617 JP=K+SUM
3620 IF A$="NKAOT" OR A$="NANL" THEN GOTO 3630
3625 IF A$="RK" OR A$="ZK" THEN GOTO 3690
3630 FOR P=JP TO MS
3631 IF P=MS THEN GOTO 3655
3650 PRINT #2,USING"#####";AA(P,KK)
3653 GOTO 3663
3655 PRINT #2,USING"#####";AA(P,KK);
3663 NEXT P
3664 PR=1;PR1=1;SUM=SUM+NS-1
3665 GOTO 4200
3690 FOR P=JP TO MS
3691 IF P=MS THEN GOTO 3698
3695 PRINT #2,USING"#####.###";AA(P,KK)
3697 GOTO 3700

```

```

3698 PRINT #2,USING"#####.###";AA(P,KK);
3700 NEXT P
3705 PR=1 ;PR1=0:SUM=SUM+NS-1
3710 GOTO 4200
3790 REM =====
4190 REM CHECKING FOR THE RETURNING STATEMENT
4195 REM =====
4200 IF KKK=1 GOTO 2600
4210 IF KKK=2 GOTO 2642
4215 IF KKK=3 GOTO 2663
4220 IF KKK=4 GOTO 2676
4230 IF KKK=5 GOTO 2726
4235 IF KKK=6 GOTO 2742
4260 REM *****
4300 PR=0
4320 REM
4335 NEXT N
4340 RETURN
4350 REM
4360 REM
4370 REM
5000 REM =====
5010 REM SUBROUTINE FOR THE ORGANIZATION OF THE OUTPUT FILE AS UNFORMATED
5011 REM =====
5015 REM
5017 IF TIT# <> "" THEN GOTO 5024
5019 PRINT #2,TITLE#
5022 GOTO 5030
5024 PRINT #2,TIT#
5027 REM TITLE OF THE RUN IS PRINTED INTO THE FILE DATOR.FOR
5030 FOR N=1 TO RROW
5031 FOR IY=1 TO X2
5032 DUM(IY)=0
5033 NEXT IY
5040 T#="" :IK=0
5050 FOR M=1 TO X2
5060 IF BB*(N,M)="" THEN GOTO 5100
5070 T#="CHK":IK=IK+1
5080 DUM(M)=M:IF M > 1 THEN GOTO 5100
5090 ML=M
5100 NEXT M
5105 IF IK > 1 THEN GOSUB 5330
5110 IF IK = 1 THEN GOSUB 5230
5115 IF T#="" THEN GOTO 5130
5125 GOTO 5170
5130 FOR LL=1 TO X2
5140 PRINT #2,AA(N,LL);
5150 NEXT LL
5152 NN=N
5153 IF NN+1 > RROW GOTO 5170
5160 PRINT #2,
5170 NEXT N
5180 REM CLOSE(2)
5190 RETURN
5200 REM
5205 REM
5207 REM
5209 REM
5210 REM
5230 FOR J=1 TO ML-1
5240 PRINT #2,AA(N,J);
5250 NEXT J
5260 PRINT #2,BB*(N,ML);
5270 FOR JJ=ML+1 TO X2
5280 PRINT #2,AA(N,JJ);
5290 NEXT JJ
5292 IF N+1 > RROW GOTO 5310
5300 PRINT #2,
5310 RETURN
5329 REM
5330 FOR IJ=1 TO X2
5340 IF DUM(IJ) > 0 GOTO 5370
5350 PRINT #2,AA(N,IJ);
5360 GOTO 5400
5370 PRINT #2,BB*(N,IJ);
5382 IF N+1 > RROW GOTO 5400
5390 PRINT #2,
5400 NEXT IJ
5410 RETURN
6000 REM
7000 REM -----

```

```

7010 REM ----- MESH MENU -----
7020 REM -----
7040 UFG1=0 : UFG2=0
7050 COLOR 7,0:CLS:GOSUB 1310:COLOR 0,7
7060 LOCATE 8,34:PRINT " MESH MENU ";
7070 LOCATE 12,25:PRINT " A> ADD MESH GENERATION DATA TO ";
7080 LOCATE 13,25:PRINT " DATOR.INP AUTOMATICALLY. ";
7090 LOCATE 14,25:PRINT " B> RETURN TO ORGANIZATION-MENU ";
7100 LOCATE 21,42:INPUT Q$:COLOR 7,0
7110 IF Q$="A" OR Q$="a" THEN GOTO 7170
7120 IF Q$="B" OR Q$="b" THEN GOTO 7190
7140 CLS:COLOR 28,0
7150 LOCATE 15,10:PRINT"ERROR IN YOUR CHOICE !!! Press the enter key >>>...";
7160 INPUT RR$ : GOTO 7050
7170 UFG1=1
7180 RETURN
7190 UFG2=1
7200 RETURN
7210 END
8000 REM -----
8010 REM          MENU FOR THE ORGANIZATION OF THE
8020 REM          DATA FILE      DATOR.FOR(=R1*)
8030 REM -----
8040 REM
8045 CLS: LOCATE 8,1: PRINT STRING$(80,219);
8047 FOR IK=9 TO 20
8050 LOCATE IK,1: PRINT CHR$(221);:LOCATE IK,80:PRINT CHR$(222);
8060 NEXT IK
8070 LOCATE 21,1 : PRINT STRING$(80,219);
8080 LOCATE 5,27: PRINT STRING$(28,220) ;
8085 LOCATE 6,27: PRINT CHR$(221);: LOCATE 6,54:PRINT CHR$(222);
8090 LOCATE 7,27: PRINT CHR$(221);: LOCATE 7,54:PRINT CHR$(222);
8100 LOCATE 8,27: PRINT CHR$(221);: LOCATE 8,54:PRINT CHR$(222);
8110 COLOR 0,7: LOCATE 7,31: PRINT " ORGANIZATION-MENU " ;
8120 LOCATE 11,20: PRINT " A> AUTOMATIC DATA PREPARATION ";
8130 LOCATE 13,20: PRINT " B> SCREEN ORIENTED DATA PREPARATION ";
8140 LOCATE 15,20: PRINT " C> AUTOMATIC MESH DATA ";
8150 LOCATE 17,20: PRINT " D> ISOPARAMETRIC MESH DATA ";
8153 LOCATE 19,20: PRINT " E> RETURN TO MAIN-MODULE ";
8160 LOCATE 23,20: PRINT "Please enter your choice ";
8170 PRINT ">>>>>>>>";:COLOR 28,0 :LOCATE 23,57:COLOR 7,0
8180 COLOR 0,7: INPUT QQ$
8190 COLOR 7,0
8200 IF QQ$="A" OR QQ$="a" THEN DUMMY=1
8210 IF QQ$="B" OR QQ$="b" THEN DUMMY=2
8220 IF QQ$="C" OR QQ$="c" THEN DUMMY=3
8230 IF QQ$="D" OR QQ$="d" THEN DUMMY=5
8235 IF QQ$="E" OR QQ$="e" THEN DUMMY=4
8240 IF QQ$="" THEN GOTO 8000
8250 RETURN
8500 REM -----
8510 REM          AUTOMATIC DATA PREPARATION FOR EPDAN
8520 REM          --FORMATTED--
8530 REM -----
8540 LLI=0 :LL1=2 :LL2=4
8560 NMA$(1,1)="INKAOT" : NMA$(2,1)="INANL"
8570 NMA$(2,2)="ITANL" : NMA$(3,1)="I"
8580 NMA$(3,2)="J" : NMA$(3,3)="NSABST"
8590 NMA$(4,1)="DPHI" : NMA$(4,2)="PHINAX"
8600 NMA$(5,1)="IDINKF" : NMA$(5,2)="IDINKV"
8610 NMA$(5,3)="IDINKR" : NMA$(5,4)="IDINKS"
8620 NMA$(6,1)="SKFNUL" : NMA$(6,2)="EN"
8630 NMA$(6,3)="PO" : NMA$(6,4)="SK"
8640 NMA$(8,1)="CEE" : NMA$(8,2)="ENN"
8650 NMA$(9,1)="IMA" : NMA$(9,2)="ILMA"
8660 NMA$(9,3)="IHALT" : NMA$(9,4)="IHEPSI"
8670 NMA$(9,5)="IDENGE" : NMA$(10,2)="IWRITEG"
8680 NMA$(10,3)="ZEND" : NMA$(11,1)="IX"
8690 NMA$(11,2)="IY" : NMA$(12,1)="RB"
8700 NMA$(12,2)="ZR" : NMA$(12,3)="RC"
8710 NMA$(12,4)="ZC" : NMA$(12,5)="RD"
8720 NMA$(12,6)="ZD" : NMA$(12,7)="RA"
8730 NMA$(12,8)="ZA" : NMA$(13,1)="RADE"
8740 NMA$(13,2)="RADA" : NMA$(14,1)="RK"
8750 NMA$(15,1)="ZK" : NMA$(16,1)="LIMVER"
8760 NMA$(16,2)="ILASTI" : NMA$(16,3)="LIMVER1"
8770 NMA$(16,4)="ILASTE" : NMA$(16,5)="LIMVER2"
8780 NMA$(17,1)="IECK" : NMA$(17,2)="IECK2"

```



```

8790 NMA$(17,3)="ANF" : NMB$(7,1)="IFLITY"
8800 NMB$(10,1)="ITOP"
8810 PRM=0 : AZ$="INKAOT"
8820 GOSUB 9300
8830 IF PRM=0 THEN GOTO 8850
8835 LLI=LLI+1 : GOSUB 9500
8837 NMA$(LLI,1)="NKAOT"
8840 PRM=0
8850 AZ$="INANL"
8860 GOSUB 9300
8870 IF PRM=0 THEN GOTO 8879
8871 IF LLI ><0 THEN GOTO 8873
8872 LL2=LL2-1
8873 LLI=LL2+1
8875 GOSUB 9500
8877 NMA$(LL2,1)="NANL"
8879 INC=0 : LT=0 : CV$="" : DM=0
8880 FOR IN=1 TO 60
8890 FOR IN=1 TO X2
8891 I=IN : J=IN
8893 IF I=LT THEN CV$="SAME"
8894 LT=I
8900 AE$=NMA$(I,J)
8910 IF AE$ >< "" THEN GOTO 8920
8915 AE$=NMB$(I,J)
8915 IF AE$ >< "" THEN GOTO 8920
8917 GOTO 8922
8920 GOSUB 10000
8922 CV$=""
8930 NEXT IN
8940 NEXT IN
9010 RETURN
9300 REM -----
9310 REM          SEARCHING PROCESS
9320 REM -----
9340 GOSUB 700
9350 FOR JJ=1 TO KEZ
9360 IF AZ$=NAW$(JJ) GOTO 9390
9370 NEXT JJ
9380 GOTO 9410
9390 IF ADR(JJ) >< 0 THEN GOTO 9400
9395 GOTO 9410
9400 PRM=1
9410 RETURN
9500 REM -----
9510 REM          INSERTING PROCESS
9520 REM          FOR THE NON ZERO ARRAYS
9530 REM -----
9570 FOR K=X1 TO LLI STEP -1
9575 JJ=K-1
9580 FOR J=1 TO X2
9600 NMA$(K,J)=NMA$(JJ,J) : NMA$(JJ,J)=""
9620 NEXT J
9630 NEXT K
9635 FOR K=X1 TO 2 STEP -1
9637 JJ=K-1
9639 FOR J=1 TO X2
9645 NMB$(K,J)=NMB$(JJ,J) : NMB$(JJ,J)=""
9647 NEXT J
9650 NEXT K
9660 RETURN
10000 REM -----
10010 FOR II=1 TO KEZ
10020 IF AE$=NAW$(II) THEN GOTO 10110
10030 NEXT II
10040 FOR II=1 TO KEZ
10050 IF AE$=UV$(II) THEN GOTO 10180
10060 NEXT II
10070 FOR II=1 TO KEZ
10080 IF AE$=WA1$(II) THEN GOTO 10340
10090 NEXT II
10095 SA$="NOT" : RETURN
10100 REM

```

```

10110 IF LQ=1 THEN GOTO 10120
10115 IF CV$="SAME" THEN GOTO 10130
10120 I=I+INC:DM=I : LQ=0:GOTO 10140
10130 I=DM
10140 AA(I,J)=ADR(II)
10160 RETURN
10170 REM
10180 III=II+1
10190 EN1=EV(II)
10200 EN2=EV(III)
10210 EN3=EN2-1 :IF EN3=0 THEN GOTO 10330
10220 EN4=EN2-EN1
10225 SE(I,J)=EN4
10230 IF CV$="SAME" THEN GOTO 10250
10240 I=I+INC:DM=I:GOTO 10230
10250 I=DM
10280 FOR IE=EN1 TO EN3
10290 AA(I,J)=VV(IE)
10295 I=I+1
10300 NEXT IE
10310 INC=INC+EN4-1
10330 RETURN
10340 REM IF CV$="SAME" THEN GOTO 10370
10350 DM=I
10370 I=DM:LQ=1
10380 AC$=WA$(II)
10390 BB$(I,J)=AC$
10457 REM
10460 RETURN
11000 REM -----
11010 REM SUBROUTINE FOR THE SCREEN ORIENTED FILE ORGANIZATION
11020 REM -----
11040 IF TIT$ <> "" THEN GOTO 11055
11045 PRINT #2,TITLE$
11050 GOTO 11065
11055 PRINT #2,TIT$
11065 FOR N=1 TO RR0W
11070 FOR IT=1 TO X2
11075 DUM(IT)=0
11080 NEXT IT
11090 PR2=0:JL=0
11095 T$="":IK=0
11100 FOR M=1 TO X2
11110 IF BB$(N,M)="" THEN GOTO 11150
11120 T$="CHK":IK=IK+1
11130 DUM(M)=M:IF M > 1 THEN GOTO 11150
11140 ML=M
11150 NEXT M
11160 IF IK>1 THEN GOTO 11445
11170 IF IK=1 THEN GOSUB 11300
11180 IF T$="" THEN GOTO 11200
11190 GOTO 11280
11200 FOR LL=1 TO X2
11210 K=N: KK=LL
11220 A$=NNA$(N,LL)
11230 IF A$="" THEN GOTO 11260
11240 KKK=1
11250 GOTO 11510
11260 NEXT LL
11280 GOTO 11821
11282 PR=1
11300 FOR LL=1 TO ML-1
11305 K=N : KK=LL: A$=NNA$(N,LL)
11310 KKK=2
11315 GOTO 11510
11330 NEXT LL
11340 K=N : KK=ML
11350 A$=NMB$(N,ML)
11360 KKK=3
11370 GOTO 11510
11390 FOR LL=ML+1 TO X2
11395 K=N : KK=LL :A$=NMA$(N,LL)
11400 KKK=4
11410 GOTO 11510
11420 NEXT LL
11430 GOTO 11190

```

```

11445 FOR LL=1 TO X2
11450 K=N : KK=LL
11455 IF DUM(LL) >< 0 GOTO 11485
11460 A$=NMA$(N,LL)
11465 KKK=5
11470 GOTO 11510
11485 A$=NMB$(N,LL)
11490 KKK=6
11495 GOTO 11510
11497 REM
11499 NEXT LL
11500 GOTO 11190
11510 REM ===== CHECKING OF THE FORMATS =====
11520 REM =====
11525 IF N=1 AND LL=1 THEN GOTO 11540
11530 IF LL >< 1 THEN GOTO 11540
11535 PRINT #2,
11540 IF A$="INKAOT" THEN GOTO 11619
11545 IF A$="INAML" THEN GOTO 11619
11547 IF A$="ITANL" THEN GOTO 11619
11549 IF A$="I" THEN GOTO 11619
11550 IF A$="J" THEN GOTO 11619
11555 IF A$="NSABST" THEN GOTO 11619
11557 IF A$="DPHI" THEN GOTO 11623
11558 IF A$="PHIMAX" THEN GOTO 11623
11559 IF A$="IDINKF" THEN GOTO 11619
11560 IF A$="IDINKV" THEN GOTO 11619
11561 IF A$="IDINKR" THEN GOTO 11619
11562 IF A$="IDINKS" THEN GOTO 11619
11563 IF A$="SKFNUL" THEN GOTO 11627
11566 IF A$="EM" THEN GOTO 11633
11567 IF A$="PO" THEN GOTO 11640
11568 IF A$="SK" THEN GOTO 11640
11569 IF A$="IFLITY" THEN GOTO 11670
11570 IF A$="CEE" THEN GOTO 11627
11571 IF A$="ENN" THEN GOTO 11627
11572 IF A$="INA" THEN GOTO 11619
11573 IF A$="ILMA" THEN GOTO 11619
11574 IF A$="IHALT" THEN GOTO 11619
11575 IF A$="IHEPS1" THEN GOTO 11619
11576 IF A$="IDENGE" THEN GOTO 11619
11577 IF A$="ITOP" THEN GOTO 11670
11578 IF A$="IWRITEG" THEN GOTO 11619
11579 IF A$="ZEND" THEN GOTO 11623
11580 IF A$="IX" THEN GOTO 11619
11581 IF A$="IY" THEN GOTO 11619
11582 IF A$="RB" THEN GOTO 11627
11583 IF A$="ZB" THEN GOTO 11627
11584 IF A$="RC" THEN GOTO 11627
11585 IF A$="ZC" THEN GOTO 11627
11586 IF A$="RD" THEN GOTO 11627
11587 IF A$="ZD" THEN GOTO 11627
11588 IF A$="RA" THEN GOTO 11627
11589 IF A$="ZA" THEN GOTO 11627
11590 IF A$="RADE" THEN GOTO 11640
11591 IF A$="RADA" THEN GOTO 11640
11592 IF A$="IRN" THEN GOTO 11619
11594 IF A$="IZN" THEN GOTO 11619
11596 IF A$="LIMVER" THEN GOTO 11619
11598 IF A$="ILAST1" THEN GOTO 11619
11599 IF A$="LIMVER1" THEN GOTO 11619
11600 IF A$="ILAST2" THEN GOTO 11619
11610 IF A$="LIMVER2" THEN GOTO 11619
11611 IF A$="IECK" THEN GOTO 11619
11612 IF A$="IECK2" THEN GOTO 11619
11613 IF A$="ANF" THEN GOTO 11623
11614 IF A$="NKAOT" THEN GOTO 11680
11615 IF A$="NANL" THEN GOTO 11680
11616 IF A$="RK" THEN GOTO 11680
11617 IF A$="ZK" THEN GOTO 11680
11618 GOTO 11799
11619 PRINT #2,USING"####";AA(K,KK);
11620 PR1=1:GOTO 11799
11623 PRINT #2,USING"####.####";AA(K,KK);
11625 PR1=1:GOTO 11799
11627 PRINT #2,USING"#####.###";AA(K,KK);
11629 PR1=1:GOTO 11799
11633 PRINT #2,USING"#####.##";AA(K,KK);
11636 PR1=1:GOTO 11799
11640 PRINT #2,USING"#####.###";AA(K,KK);
11650 PR1=1:GOTO 11799

```



```

12085 LOCATE 13,15:PRINT "I , J";
12087 COLOR 28,0
12090 LOCATE 14,15:PRINT CHR$(25):LOCATE 14,22 :PRINT CHR$(25);
12095 COLOR 7,0:COLOR 0,7
12099 LOCATE 15,13:INPUT I: LOCATE 15,18:PRINT CHR$(44):LOCATE 15,20:INPUT J
12100 RWN=I
12102 IF I<=0 OR J<=0 THEN GOTO 12069
12105 IF I=LT THEN CV$="SAME"
12107 LOCATE 21,13:INPUT SA$ :COLOR 7,0
12109 IF SA$="" THEN GOTO 12129
12110 IF SA$="1" THEN GOTO 12225
12115 IF SA$="2" THEN GOTO 12066
12120 IF SA$="3" THEN GOTO 12019
12121 CLS
12122 LOCATE 21,45:PRINT" Please reinput your data ! Press any key ...";
12125 Q$=INKEY$:IF Q$="" THEN GOTO 12125
12126 CLS
12128 GOTO 12066
12129 SA$="" : LT=I
12130 GOSUB 12300
12132 CV$=""
12135 IF SA$="NOT" THEN GOTO 12066
12216 NEXT OT
12217 RROW=RWN+UNF4
12218 GOTO 12234
12225 SA$=""
12230 GOSUB 12300
12232 IF SA$="NOT" THEN GOTO 12066
12234 GOSUB 12975
12236 COLOR 7,0
12240 RETURN
12300 REM -----
12305 REM SUBROUTINE FOR THE SEARCHING OF THE ARRAYS
12310 REM -----
12314 GOSUB 700
12315 FOR II=1 TO KEZ
12320 IF AE$=NAW$(II) THEN GOTO 12368
12325 NEXT II
12330 FOR II=1 TO KEZ
12335 IF AE$=UV$(II) THEN GOTO 12400
12340 NEXT II
12345 FOR II=1 TO KEZ
12350 IF AE$=WA1$(II) THEN GOTO 12440
12355 NEXT II
12365 SA$="NOT" : RETURN
12366 REM
12368 IF CV$="SAME" THEN GOTO 12370
12369 I=I+INC:DN=I : GOTO 12374
12370 I=DN
12374 AA(I,J)=ADR(II)
12375 NNA$(I,J)=AE$
12390 RETURN
12395 REM
12400 III=II+1
12405 EN1=EV(II)
12406 EN2=EV(III)
12407 EN3=EN2-1 :IF EN3=0 THEN GOTO 12435
12408 EN4=EN2-EN1
12414 IF CV$="SAME" THEN GOTO 12416
12415 I=I+INC:DN=I:GOTO 12417
12416 I=DN
12417 SE(I,J)=EN4
12418 NNA$(I,J)=AE$
12419 FOR IE=EN1 TO EN3
12420 AA(I,J)=VV(IE)
12422 I=I+1
12425 NEXT IE
12426 I=0:INC=INC+EN4-1
12428 UNF4=UNF4+EN4-1
12435 RETURN
12440 IF CV$="SAME" THEN GOTO 12443
12441 I=I+INC:DN=I.
12442 GOTO 12445
12443 I=DN
12445 AC$=WA$(II)
12450 BB$(I,J)=AC$
12455 NNB$(I,J)=AE$
12557 REM

```

```

12560 RETURN
12800 REM
12950 REM -----
12960 REM MENU FOR THE GENERATION OF THE OUTPUT FILE "DATOR.FOR"
12970 REM -----
12975 FOR1=0 : FOR2=0 : FOR3=0
12980 COLOR 7,0:CLS:GOSUB 1310:COLOR 0,7
12990 LOCATE 8,34:PRINT"FORMAT MENU";
13000 LOCATE 12,25:PRINT" A> FORMATED FILE ORGANIZATION ";
13005 LOCATE 14,25:PRINT" B> UNFORMATED FILE ORGANIZATION ";
13010 LOCATE 16,25:PRINT" C> RETURN TO ORGANIZATION-MENU ";
13020 LOCATE 21,42:INPUT Q$:COLOR 7,0
13030 IF Q$="A" OR Q$="a" THEN GOTO 13300
13040 IF Q$="B" OR Q$="b" THEN GOTO 13320
13050 IF Q$="C" OR Q$="c" THEN GOTO 13340
13060 CLS:COLOR 28,0
13070 LOCATE 15,10:PRINT"ERROR IN YOUR CHOICE ! Press the enter key >>>...";
13080 INPUT RE$: GOTO 12980
13300 FOR1=1
13310 RETURN
13320 FOR2=1
13330 RETURN
13340 FOR3=1
13350 RETURN
13355 END
10 REM *****
15 REM *
20 REM * ISOPARAMETRIC *
30 REM *
40 REM * MESH GENERATION PROGRAM *
42 REM *
43 REM *****
44 COMMON NKAOT(1),NANL(1),RK(1),ZK(1),ADR(1),NAW$(1)
45 COMMON VV(1),UV$(1),IV(1),EV(1),WA$(1),WA1$(1),AA(2),BB$(2)
46 COMMON PAR,R1$,R2$,X3,IX1,IX2,X1,X2
47 COMMON NMA$(2),NMB$(2),DUM(1),SE(2)
48 COMMON RET,X4
49 COMMON FZ(1),FR(1),NPOS(1),NACT(1),NAPR(1),IU(1),JU(1),KU(1),MU(1),NSTEM(1)
55 COMMON ISO
60 DIM ETA(8),KSI(8),NUMN(8)
70 DIM NODN(500),SHPF(8),COORDX(500),COORDY(500),ISOX(8),ISOY(8)
80 DIM IEL(200,8),SW(50),KSET(100,2),COORD(8,2),EAA(8),KII(8)
84 DIM NOD(200),COR(200,2),ELE(150,4),CORB(10,4)
85 DIM NPO(30),NAC(30),NAP(30),NSTE(30)
90 REM
93 ISO=5
95 CLS:KEY OFF
97 GOTO 200
100 GOSUB 2000
120 GOSUB 3100
125 FOR TIME=1 TO NUMEL
127 GOSUB 3200
130 GOSUB 3000
134 GOSUB 3269
135 GOSUB 3800
140 NEXT TIME
145 GOSUB 5400
150 GOSUB 8000
170 GOTO 290
200 CLS:GOSUB 5000
210 LOCATE 14,20:PRINT" A> START THE ISOMESH PROGRAM";
220 LOCATE 16,20:PRINT" B> RETURN TO THE MODULE-ORGANIZATION";
230 LOCATE 18,20:PRINT"Please input your choice >>";:INPUT SW$
240 IF SW$="A" OR SW$="a" THEN GOTO 270
250 IF SW$="B" OR SW$="b" THEN GOTO 290
260 GOTO 200
270 GOTO 100
290 REDIM BB$(X1,X2),NMA$(X1,X2),NMB$(X1,X2),DUM(X1),SE(X1,X2)
295 REDIM NKAOT(X3),NANL(X3),RK(X4),ZK(X4),ADR(X3),NAW$(X3)
300 REDIM VV(X3),UV$(X3),IV(X3),EV(X3),WA$(X3),WA1$(X3),AA(X1,X2)
310 REDIM FZ(X4),FR(X4),NPOS(X4),NACT(X4),NAPR(X4)
320 REDIM IU(X4),JU(X4),KU(X4),MU(X4),NSTEM(X4)
330 CHAIN"C:ORGANIZE.EXE"
500 REM SUBROUTINE INITIALIZE -----
510 FOR I=1 TO 8
520 NUMN(I)=0
530 KSI(I)=0
540 ETA(I)=0

```

```

550 NEXT I
560 RETURN
600 REM SUBROUTINE MOVE THE LOCATIONS OF THE ARRAYS FOR THE SAME DATA
620 FOR II=1 TO SSUM-1
630 DU=NODN(II)
640 FOR I=II+1 TO SSUM
650 IF DU >< NODN(I) THEN GOTO 710
660 FOR J=1 TO SSUM-1
670 NODN(J)=NODN(J+1)
680 COORX(J)=COORX(J+1)
690 COORY(J)=COORY(J+1)
700 NEXT J
710 NEXT I
720 NEXT II
730 RETURN
800 REM SUBROUTINE - SEARCH
805 REG=0
810 FOR K=1 TO NUMEL
820 IF ND >< COMB(K,1) THEN GOTO 860
830 COMB(K,3)=J
835 IF E ><2 GOTO 858
845 COMB(K,4)=COMB(K,3)+IMX
856 GOTO 860
858 COMB(K,4)=COMB(K,3)+IMX
860 NEXT K
870 RETURN
1000 REM TABLE DRAWING (1) -----
1010 CLS:COLOR 28,0
1020 LOCATE 1,1:PRINT STRING$(80,196);
1030 FOR II=1 TO 24
1040 LOCATE II,2:PRINT CHR$(179);:LOCATE II,79:PRINT CHR$(179);
1050 NEXT II
1060 LOCATE 23,1:PRINT STRING$(80,196);
1070 COLOR 7,0
1080 RETURN
2000 REM SUBROUTINE - START THE MESH GENERATION PROCESS -----
2005 GOSUB 1000
2010 LOCATE 13,23:PRINT" M E S H   G E N E R A T I O N ";
2020 LOCATE 14,23:PRINT"   P R O C E S S ";
2030 LOCATE 15,23:PRINT"Based on isoparametric concept";
2040 LOCATE 16,23:PRINT"Press any key to continue ";
2042 COLOR 28,0:PRINT">>>>>";:INPUT A$
2045 COLOR 7,0
2050 RETURN
2500 REM SUBROUTINE - ERROR -----
2510 CLS:COLOR 28,0:GOSUB 5000
2520 LOCATE 15,20:PRINT" ERROR in your data ";
2530 LOCATE 16,20:PRINT"Please input again .....";
2535 COLOR 7,0
2540 LOCATE 22,20:PRINT" Press any key to INPUT   >>>>>>>>>";
2550 LOCATE 22,40:INPUT A$
2560 CLS:RETURN
3000 REM INPUT NUMBER OF ELEMENTS IN THE X AND Y DIRECTIONS -----
3010 CLS
3020 GOSUB 5000
3030 LOCATE 15,10:PRINT"Number of elements in the X direction";
3040 LOCATE 22,10:COLOR 28,0:PRINT" NNX >>>>>>";:INPUT NNX
3050 COLOR 7,0 : CLS
3052 IF NNX <= 0 THEN GOTO 3058
3055 GOTO 3060
3058 GOSUB 2500:GOTO 3010
3060 GOSUB 5000
3070 LOCATE 15,10:PRINT"Number of elements in the Y direction";
3080 LOCATE 22,10:COLOR 28,0:PRINT" NNY >>>>>>";:INPUT NNY
3085 COLOR 7,0 : CLS
3087 IF NNY <= 0 THEN GOTO 3089
3088 GOTO 3090
3089 GOSUB 2500:GOTO 3060
3090 RETURN
3100 REM SUBROUTINE - SUPERELEMENT MESH -----
3110 CLS:ECON=0
3120 GOSUB 5000
3125 LOCATE 15,10:PRINT"NUMEL=Number of divisions for the whole domain";
3130 LOCATE 22,10:COLOR 28,0:PRINT"NUMEL >>>>";:INPUT NUMEL
3135 COLOR 7,0

```

```

3137 IF NUMEL <=0 THEN GOTO 3140
3139 GOTO 3145
3140 GOSUB 2500 :GOTO 3110
3145 SSM=0 : SSM=0 : QU=8 : TTE=0
3150 RETURN
3200 REM SUBROUTINE - COORDINATES -----
3210 CLS
3220 ECOM=ECOM+1
3230 GOSUB 5000
3240 LOCATE 15,10:PRINT"REGION";:PRINT ECOM ;
3250 LOCATE 16,10: PRINT"Master nodal coordinates KSI and ETA with respect";
3260 LOCATE 17,10: PRINT"to the X,Y coordinates of the section";
3265 LOCATE 22,10:COLOR 28,0:PRINT"Press any key to continue";:INPUT A$
3266 COLOR 7,0
3267 RETURN
3268 REM SUBROUTINE - SUPERELEMENT MESH DATA -----
3269 CLS : GOSUB 500 :KKK=1
3270 FOR I=1 TO 8
3271 GOSUB 3400
3272 LOCATE 22,2:PRINT"Node No ";:INPUT NUMN(I)
3273 IF NUMN(I)=0 THEN GOTO 3272
3274 IF NUMN(I)>8 THEN GOTO 3279
3275 DD=NUMN(I) : INXX=I
3276 GOSUB 5100
3277 IF INX>1 THEN GOTO 3271
3278 GOTO 3280
3279 GOSUB 2500 :GOTO 3271
3280 LOCATE 22,20:PRINT"KSI(";:PRINT I;:PRINT ")=";:INPUT KSI(I)
3290 LOCATE 22,40:PRINT"ETA(";:PRINT I;:PRINT ")=";:INPUT ETA(I)
3295 LOCATE 22,28:PRINT STRING$(10,176);:LOCATE 22,48:PRINT STRING$(10,176)
3300 LOCATE 22,10:PRINT STRING$(6,176)
3302 LOCATE 22,24:PRINT STRING$(3,176):LOCATE 22,44:PRINT STRING$(3,176)
3310 KKK=KKK+1
3320 NEXT I
3330 CLS:GOSUB 3400
3340 RETURN
3400 REM
3401 LOCATE 1,2:PRINT"Domains with Ksi and Eta axes are oriented as in the Figure";
3413 LOCATE 16,10:PRINT CHR$(219)
3415 LOCATE 17,11:PRINT STRING$(17,196)
3416 LOCATE 3,11:PRINT STRING$(17,196)
3417 LOCATE 3,27:PRINT CHR$(219)
3418 LOCATE 17,27:PRINT CHR$(219)
3419 LOCATE 17,18:PRINT CHR$(219)
3425 LOCATE 17,10:PRINT CHR$(219)
3427 LOCATE 3,18: PRINT CHR$(219)
3428 LOCATE 3,10: PRINT CHR$(219)
3429 FOR II=4 TO 16
3430 LOCATE II,10:PRINT CHR$(179)
3440 LOCATE II,27:PRINT CHR$(179)
3445 NEXT II
3448 LOCATE 10,27:PRINT CHR$(219)
3449 LOCATE 10,10:PRINT CHR$(219)
3450 LOCATE 11,18:PRINT CHR$(26)
3452 LOCATE 9,16:PRINT CHR$(179):LOCATE 8,16:PRINT CHR$(24)
3460 LOCATE 7,20:PRINT "Eta":LOCATE 10,16:PRINT CHR$(179)
3470 LOCATE 11,21:PRINT "Ksi" :LOCATE 11,17:PRINT CHR$(196)
3475 LOCATE 11,18:PRINT CHR$(196):LOCATE 11,19:PRINT CHR$(26)
3476 LOCATE 11,16:PRINT CHR$(192)
3480 LOCATE 19,27:PRINT CHR$(49)
3490 LOCATE 2,27:PRINT CHR$(51)
3495 LOCATE 10,29:PRINT CHR$(50)
3500 LOCATE 2,18: PRINT CHR$(52)
3510 LOCATE 2,10: PRINT CHR$(53)
3520 LOCATE 10,8: PRINT CHR$(54)
3540 LOCATE 19,10: PRINT CHR$(55)
3550 LOCATE 19,18: PRINT CHR$(56)
3554 IF KKK=9 THEN GOTO 3609
3555 IF KKK=1 THEN GOTO 3780
3556 LOCATE 2,35:PRINT"Node";:LOCATE 2,43:PRINT"COORDINATE";
3557 LOCATE 2,56:PRINT"COORDINATE";:LOCATE 3,35:PRINT"Number";
3558 LOCATE 3,43:PRINT" X ";:LOCATE 3,57:PRINT" Y ";
3559 L=3
3560 FOR I1=1 TO 8
3561 L=L+2
3566 IF KKK ><9 GOTO 3570
3567 REM PRINT #3, NUMN(I1),KSI(I1),ETA(I1)
3570 LOCATE L,36:PRINT NUMN(I1)
3580 LOCATE L,46:PRINT KSI(I1)
3585 LOCATE L,60:PRINT ETA(I1)
3590 NEXT I1
3595 GOTO 3780

```



```

3609 FOR I=1 TO 8
3610 MIN=NUMN(I) : ID=I
3620 FOR J=I+1 TO 8
3630 IF NUMN(J) > MIN THEN GOTO 3650
3640 MIN=NUMN(J) : ID= J
3650 NEXT J
3660 DUM1=NUMN(ID) :DUM2=KSI(ID) :DUM3=ETA(ID)
3670 NUMN(ID)=NUMN(I) :KSI(ID)=KSI(I) :ETA(ID)=ETA(I)
3680 NUMN(I)=DUM1 :KSI(I)=DUM2 : ETA(I)=DUM3
3690 NEXT I
3695 J=0
3700 JO=ECOM*QU
3710 JH=JO-(QU-1)
3720 FOR I=JH TO JO
3730 J=J+1
3740 KSET(I,1)=KSI(J)
3750 KSET(I,2)=ETA(J)
3760 NEXT I
3770 GOTO 3556
3780 RETURN
3800 REM SUBROUTINE - ISOPARAMETRIC MESH GENERATION -----
3810 ISOX(1)=1! : ISOX(2)=1! : ISOX(3)=1! : ISOX(4)=0!
3820 ISOX(5)=-1! : ISOX(6)=-1! : ISOX(7)=-1! : ISOX(8)=0!
3830 ISOY(1)=-1! : ISOY(2)=0! : ISOY(3)=1! : ISOY(4)=1!
3840 ISOY(5)=1! : ISOY(6)=0! : ISOY(7)=-1! : ISOY(8)=-1!
4010 KKK=0: IDA=1! : IDE=-1!:ELN=0
4015 E=0 :QU=8
4020 REM
4042 TIMX=2*NNX+1
4044 TIMY=2*NNY+1
4045 CLS:GOSUB 5000
4046 LOCATE 15,17:PRINT "Input the starting node number for the new domain";
4047 LOCATE 22,17:PRINT"ECO >>>";:INPUT ECO
4048 IF ECO <1 THEN GOTO 4045
4050 EC=ECO
4051 TTE=TTE+1:COMB(TTE,1)=ECO
4052 CLS:LOCATE 15,15:PRINT "Input ELEMENT-NODE NUMBERING DIRECTION";
4054 LOCATE 16,15:PRINT" X) Along the x direction ";
4055 LOCATE 17,15:PRINT" Y) Along the y direction ";
4056 LOCATE 20,15:PRINT" INPUT YOUR CHOICE >>>";:INPUT A#:CLS
4057 IF A#="X" OR A#="x" THEN GOTO 4090
4058 IF A#="Y" OR A#="y" THEN GOTO 4060
4059 GOTO 4052
4060 E=1
4062 DTUMX=DTUMX+TIMX-1 :DUMX=DUMX+NNX
4063 DTUMY=TIMY : DUMY=NNY
4064 EMM=TIMX
4065 ENN=TIMY
4070 GOTO 4100
4090 E=2
4091 DTUMY=DTUMY+TIMY-1 :DUMY=DUMY+NNY
4093 DTUMX=TIMX : DUMX=NNX
4095 EMM=TIMY : ENN=TIMX
4100 NNNY=NNY : NNXX=NNX
4103 MAXEL=NNY*NNX
4105 MAXNOD=(DUMX+1)*(DUMY+1)
4109 SSH=SSH+MAXEL
4110 FOR KK=1 TO EMM
4115 IF E=2 THEN GOTO 4120
4117 COKSI=IDE+2!*(KK-1)/(2*NNXX)
4118 GOTO 4130
4120 COETA=IDE+2!*(KK-1)/(2*NNYY)
4130 ELN=ELN+1
4150 IF ELN=3 THEN ELN=1
4160 FOR LL=1 TO ENN STEP ELN
4162 IF E=1 THEN GOTO 4175
4170 COKSI=IDE+2!*(LL-1)/(2*NNXX)
4173 GOTO 4180
4175 COETA=IDE+2!*(LL-1)/(2*NNYY)
4180 FOR LLL=1 TO QU
4190 ON LLL GOTO 4200,4230,4200,4250,4200,4230,4200,4250
4200 SHP=.25*(IDA+COKSI*ISOX(LLL))*(1! + COETA*ISOY(LLL))
4210 SHPF(LLL)=SHP*(COKSI*ISOX(LLL)+COETA*ISOY(LLL)-1!)
4220 GOTO 4260
4230 SHPF(LLL)=.5*(IDA+COKSI*ISOX(LLL))*(IDA-COETA*COETA)
4240 GOTO 4260
4250 SHPF(LLL)=.5*(IDA+COETA*ISOY(LLL))*(IDA-COKSI*COKSI)
4260 NEXT LLL
4270 SUMX=0!

```

```

4280 SUMY=0!
4290 FOR LP=1 TO QU
4300 SUMX=SUMX+SHPF(LP)*KSI(LP)
4310 SUMY=SUMY+SHPF(LP)*ETA(LP)
4320 NEXT LP
4325 NODN(EC)=EC
4330 COORX(EC)=SUMX
4340 COORY(EC)=SUMY
4350 EC=EC+1
4360 NEXT LL
4370 NEXT KK
4380 MAX=EC-1
4390 ECC=NODN(ECO)
4395 GOSUB 600
4400 IF ECC >> NUNEL THEN GOTO 4860
4405 REM END OF SUPERELEMENT MESH GENERATION -----
4410 REM FOR ND=1 TO MAX
4420 REM PRINT #2,NODN(ND),COORX(ND),COORY(ND)
4430 REM NEXT ND
4440 CLS
4460 REM NODE AND ELEMENT GENERATION -----
4480 IF E=2 THEN GOTO 4510
4490 IF E=1 THEN GOTO 4500
4495 GOTO 4440
4500 TTMX=DTUMY
4502 IMX=DUMY:IMY=DUMX
4505 GOTO 4520
4510 TTMX=DTUMX
4512 IMX=DUMX:IMY=DUMY
4520 FOR LL=1 TO 50 STEP 2
4521 SW(LL)=1
4522 NEXT LL
4525 J=0 : JJ=0 : JJJ=0 : IK=0 : SUM=0! : IJK=0
4527 DTMX=TTMX
4530 IK=0
4531 IK=IK+1
4532 J=J+1
4534 JJ=JJ+1
4536 IJK=IJK+1
4540 IF IK=1 THEN GOTO 4570
4544 IF JJ=IMX+1 THEN GOTO 4549
4545 IF IJK=3 THEN GOTO 4572
4547 I=IK+1
4548 GOTO 4580
4549 JJ=1 : SUM=0
4550 IF SW(IMX)=1 THEN GOTO 4553
4551 IK=IK+IMX+4
4552 GOTO 4554
4553 IK=IK+IMX+3
4554 DTMX=TTMX
4555 JJJ=JJJ+1
4556 IF JJJ=IMY+1 THEN GOTO 4780
4557 I=IK
4558 JJ=1
4559 IJK=1
4560 GOTO 4580
4570 I=IK
4571 GOTO 4580
4572 IJK=1
4574 IK=IK+2 : I=IK
4576 DTMX=DTMX-2
4578 SUM=SUM+2
4580 A1=NODN(I)
4590 A2=NODN(I+1)
4595 A3=NODN(I+2)
4600 T1=DTMX+IK
4610 T2=DTMX+IK+1
4620 A5=NODN(T1)
4630 A4=NODN(T2)
4640 TT1=DTMX+IMX+I+3+SUM
4642 TT2=TT1-1 : TT3=TT1-2
4650 A8=NODN(TT1)
4660 A7=NODN(TT2)
4670 A6=NODN(TT3)
4680 IEL(J,1)=A1
4690 IEL(J,2)=A2
4700 IEL(J,3)=A3
4710 IEL(J,4)=A4
4720 IEL(J,5)=A6
4730 IEL(J,6)=A7

```

```

4740 IEL(J,7)=A6
4750 IEL(J,8)=A5
4760 GOTO 4531
4780 REM FOR I=1 TO SSM
4790 REM PRINT #2, I;
4800 REM FOR J=1 TO QU
4810 REM PRINT #2, IEL(I,J);
4820 REM NEXT J
4830 REM PRINT #2,
4840 REM NEXT I
4860 RETURN
5000 REM TABLE DRAWING (3) -----
5010 CLS:LOCATE 11,1:PRINT STRING$(80,219);
5020 FOR II=12 TO 18
5030 LOCATE II,1:PRINT CHR$(221);
5040 LOCATE II,80:PRINT CHR$(222);
5050 NEXT II
5060 LOCATE 19,1:PRINT STRING$(80,219);
5070 RETURN
5100 REM SUBROUTINE :SEARCH THE PREVIOUS DATA -----
5110 REM
5120 INX=0
5130 FOR P=1 TO INXX
5140 IF DD >< NUMN(P) THEN GOTO 5170
5150 INX=INX+1
5170 NEXT P
5180 RETURN
5190 REM
5400 REM SUBROUTINE:MESH PLOTTING ON THE SCREEN -----
5410 REM
5420 CLS:GOSUB 5000
5425 LOCATE 15,15:PRINT" Do you want to see the last generated MESH ";
5430 LOCATE 16,15:PRINT" Remark: Using microcomputer must have the ";
5435 LOCATE 17,15:PRINT"          graphic card.          ";
5440 LOCATE 18,15:PRINT"          (Y/N)          ";
5445 LOCATE 21,15:PRINT"> >";:INPUT SS$
5450 IF SS$="Y" OR SS$="y" THEN GOTO 5500
5460 IF SS$="N" OR SS$="n" THEN GOTO 6300
5465 GOTO 5420
5500 SC=0:CLS
5501 GOSUB 5000:LOCATE 2,25:PRINT" SCREEN ORIENTATION MENU ";
5502 LOCATE 14,20:PRINT"A) CHANGE THE VALUES      "
5503 LOCATE 16,20:PRINT"B) CONTINUE TO THE PROCESS      "
5504 LOCATE 18,20:PRINT"C) SCREEN OUTPUT      "
5505 LOCATE 20,20:PRINT"Please input your choice      "
5506 LOCATE 20,43:INPUT SS$
5507 IF SS$="A" OR SS$="a" THEN GOTO 5515
5508 IF SS$="B" OR SS$="b" THEN GOTO 5740
5509 IF SS$="C" OR SS$="c" THEN GOTO 5520
5510 GOTO 5500
5515 GOSUB 7000
5517 CLS:SC=1
5520 SCREEN 2
5525 IF SC=1 THEN GOTO 5550
5530 CLS:X=0:Y=0:XX=350:YY=200
5550 II=0
5560 WINDOW (X,Y)-(XX,YY)
5570 PR=1
5580 FOR Z=1 TO NUMEL
5590 FOR I=1 TO QU
5592 II=II+1
5600 KII(I)=KSET(II,1)
5610 EAA(I)=KSET(II,2)
5620 NEXT I
5630 FOR J=1 TO QU-1
5640 IF J >< 1 THEN GOTO 5660
5650 W=KII(J)*PR ;WW=EAA(J)*PR
5660 I=J+1
5670 A=KII(J)*PR ;B=EAA(J)*PR
5680 C=KII(I)*PR ;D=EAA(I)*PR
5690 LINE (A,B)-(C,D)
5700 NEXT J
5710 LINE (C,D)-(W,WW)
5720 NEXT Z
5730 GOSUB 6000
5735 LOCATE 1,1 :PRINT"Press any key to continue";:INPUT A$
5738 SCREEN 0 :GOTO 5500
5740 RETURN
5800 REM
6000 REM -----
6010 REM

```

```

6020 FOR K=1 TO SSM
6030 FOR JJ=1 TO QU
6040 DUM=IEL(K,JJ)
6050 FOR KK=1 TO MAX
6060 IF DUM >< NODN(KK) THEN GOTO 6100
6070 COOR(JJ,1)=COORX(KK)
6080 COOR(JJ,2)=COORY(KK)
6090 GOTO 6110
6100 NEXT KK
6110 NEXT JJ
6120 FOR N=1 TO QU-1
6130 IF N >< 1 THEN GOTO 6160
6140 Z1=COOR(N,1)*PR
6150 Z2=COOR(N,2)*PR
6160 C1=COOR(N,1)*PR
6170 C2=COOR(N,2)*PR
6180 NN=N+1
6190 CC1=COOR(NN,1)*PR
6200 CC2=COOR(NN,2)*PR
6210 LINE (C1,C2)-(CC1,CC2)
6220 NEXT N
6230 LINE (CC1,CC2)-(Z1,Z2)
6240 NEXT K
6300 RETURN
7000 REM SUBROUTINE : CHANGE THE RANGES OF THE MESH ON THE SCREEN ----
7010 GOSUB 5000
7015 LOCATE 5,10:PRINT"Default : X0=0 Y0=0 XX=350 YY=100";
7020 LOCATE 12,10:PRINT" MAXIMUM XX=639 X0=ORIGIN ";
7030 LOCATE 12,45:PRINT" MAXIMUM YY=320 Y0=ORIGIN ";
7040 LOCATE 14,20:PRINT" 1> XX 2> YY ";
7050 LOCATE 16,20:PRINT" 3> X0 4> Y0 ";
7060 LOCATE 18,20:PRINT" 5> SAME 6> CONTINUE ";
7070 LOCATE 18,70:INPUT SS$
7080 IF SS$="1" THEN GOTO 7100
7082 IF SS$="6" THEN GOTO 7500
7090 IF SS$="2" THEN GOTO 7150
7095 IF SS$="3" THEN GOTO 7200
7097 IF SS$="4" THEN GOTO 7300
7098 IF SS$="5" THEN GOTO 7500
7099 GOTO 7010
7100 CLS:LOCATE 10,10:PRINT" Please input the value :XX >>>";
7105 INPUT XX
7110 GOTO 7010
7150 CLS
7152 LOCATE 10,10:PRINT" Please input the value :YY >>>";
7153 INPUT YY
7155 GOTO 7010
7200 CLS
7205 LOCATE 10,10:PRINT" Please input the value :X >>>";
7207 INPUT X
7210 GOTO 7010
7300 CLS
7302 LOCATE 10,10:PRINT" Please input the value :Y >>>";
7305 INPUT Y
7310 GOTO 7010
7500 RETURN
8000 REM SUBROUTINE - TRANSFORMATION OF GENERATED ISOPARAMETRIC -----
8001 REM MESH DATA TO THE MESH DATA FOR EPDAN -----
8004 ECOM=1
8005 IF E=2 THEN GOTO 8050
8010 IMX=DUMY
8020 IMY=DUMX
8030 GOTO 8100
8050 IMX=DUMY
8060 IMY=DUMX
8100 J=0
8110 I=1
8120 ND=1:III=1
8130 GOTO 8150
8140 ND=ND+2
8150 J=J+1
8160 IF ND > MAX THEN GOTO 8257
8170 IF III > IMX+1 THEN GOTO 8190
8180 GOTO 8210
8190 III=1
8200 ND=ND+IMX
8210 GOSUB 800
8215 MOD(J)=J
8220 COOR(J,1)=COORX(ND)
8230 COOR(J,2)=COORY(ND)
8240 III=III+1

```

```

8250 GOTO 8140
8265 REM NODE AND ELEMENT NUMBER GENERATION -----
8267 REM
8300 I1=0 :ICK=0 :IJ1=0 :IJ2=0:IJ3=0
8305 IJK=0 :J=0 :IJ4=0 :KE=0
8308 KP=1 :H=0 :HH=0 :LLX=0
8309 REGN$="NO"
8310 J=J+1
8315 IJK=IJK+1
8317 I1=I1+1
8320 GOSUB 9300
8321 ICK=ICK+1
8327 IF ICK > SSM THEN GOTO 8709
8328 IF IJK=IMX AND KP=1 THEN H=1
8329 IF IJK=1 AND E=2 THEN HH=1
8330 IF IJK=IMX+1 THEN GOTO 8350
8335 GOTO 8420
8350 IF E=2 THEN GOTO 8396
8360 IF ECOM=1 THEN GOTO 8366
8362 IF ECOM=2 THEN GOTO 8372
8364 IF ECOM=3 THEN GOTO 8378
8366 IF KP+1 < IMY THEN GOTO 8369
8367 GOSUB 8995
8368 GOTO 8396
8369 IJ1=IJ1+1
8370 NPO(IJ1)=I1
8371 GOTO 8396
8372 IF KP+1 < IMY THEN GOTO 8375
8373 GOSUB 8995
8374 GOTO 8396
8375 IJ2=IJ2+1
8376 NAC(IJ2)=I1
8377 IF REGN$ > "OK" THEN GOTO 8379
8378 REGN$="NO" : ECOM=REGION
8379 GOTO 8396
8380 IF KP+1 < IMY THEN GOTO 8394
8381 IF REGN$ > "OK" THEN GOTO 8383
8382 REGN$="NO" : ECOM=REGION
8383 GOSUB 8995
8384 GOTO 8396
8394 IJ3=IJ3+1
8395 NAP(IJ3)=I1
8396 IF REGN$ > "OK" THEN GOTO 8398
8397 REGN$="NO" : ECOM=REGION
8398 IJK=1:KP=KP+1
8400 I1=I1+1
8410 GOSUB 9300
8420 I2=I1+1
8440 B1=NOD(I1) : B2=NOD(I2)
8450 TQ=IMX+I2 : TQQ=TQ+1
8460 B3=NOD(TQ) : B4=NOD(TQQ)
8560 ELE(J,1)=B1
8570 ELE(J,2)=B3
8580 ELE(J,3)=B4
8590 ELE(J,4)=B2
8592 IF E=2 THEN GOTO 8652
8594 IF H=1 THEN GOTO 8597
8595 IF KP > 1 THEN GOTO 8652
8597 IJ4=IJ4+1
8599 NSTE(IJ4)=B1
8600 IF H=0 GOTO 8652
8604 IF E=2 THEN GOTO 8652
8605 IJ4=IJ4+1
8610 NSTE(IJ4)=B2 :H=0
8652 GOSUB 9100
8708 GOTO 8310
8709 FOR N=1 TO MAXNOD
8710 REM PRINT #2,MOD(N);
8720 FOR NN=1 TO 2
8730 PRINT #2, COR(N,NN);
8740 NEXT NN
8750 PRINT #2,
8760 NEXT N
8770 FOR I=1 TO SSM
8780 REM PRINT #2,I;
8790 FOR J=1 TO 4
8800 PRINT #2,ELE(I,J);
8810 NEXT J
8820 PRINT #2,
8830 NEXT I

```

```

8840 FOR I=1 TO 22
8850 PRINT #2, NPO(I)
8860 NEXT I
8870 FOR I=1 TO 14
8890 PRINT #2, NAC(I)
8900 NEXT I
8910 FOR I=1 TO 9
8920 PRINT #2, NSTE(I)
8930 NEXT I
8940 FOR I=1 TO 5
8950 PRINT #2, NAP(I)
8960 NEXT I
8970 PRINT #2, IJ1,IJ2,IJ3,IJ4
8980 RETURN
8990 REM SUBROUTINE - ARRAY ORGANIZATION (1) -----
8995 IF ECOM=1 THEN GOTO 9020
9000 IF ECOM=2 THEN GOTO 9035
9010 IF ECOM=3 THEN GOTO 9045
9020 IJ1=IJ1+1
9025 NPO(IJ1)=I1
9026 IF REGN# ><"OK" THEN GOTO 9029
9027 IF NUMEL ><2 THEN GOTO 9029
9028 REGN#="NO" :GOTO 9039
9029 IJ1=IJ1+1
9030 NPO(IJ1)=I1+IMX+1
9031 IJ1=IJ1+1
9032 NPO(IJ1)=I1+2*IMX+2
9034 GOTO 9054
9035 IJ2=IJ2+1
9036 NAC(IJ2)=I1
9037 IF REGN# ><"OK" THEN GOTO 9039
9038 REGN#="NO" :GOTO 9049
9039 IJ2=IJ2+1
9040 NAC(IJ2)=I1+IMX+1
9041 IJ2=IJ2+1
9042 NAC(IJ2)=I1+2*IMX+2
9044 GOTO 9054
9045 IJ3=IJ3+1
9047 NAP(IJ3)=I1
9049 IJ3=IJ3+1
9050 NAP(IJ3)=I1+IMX+1
9052 IJ3=IJ3+1
9053 NAP(IJ3)=I1+2*IMX+2
9054 RETURN
9100 REM SUBROUTINE - ARRAY ORGANIZATION (2) -----
9110 REM
9120 IF HH=0 THEN GOTO 9275
9125 IF KP >< IMY THEN GOTO 9263
9130 IF LLX=1 THEN GOTO 9160
9135 IJ4=IJ4+1
9140 NSTE(IJ4)=B1
9145 LLX=1
9150 IJ4=IJ4+1
9155 NSTE(IJ4)=B3
9160 IF ECOM=1 THEN GOTO 9167
9163 IF ECOM=2 THEN GOTO 9190
9165 IF ECOM=3 THEN GOTO 9230
9167 IJ1=IJ1+1
9170 NPO(IJ1)=B3
9173 IF IJK >< IMX THEN GOTO 9275
9175 IJ1=IJ1+1
9180 NPO(IJ1)=B4
9187 GOTO 9260
9190 IJ2=IJ2+1
9195 NAC(IJ2)=B3
9200 IF IJK >< IMX THEN GOTO 9275
9210 IJ2=IJ2+1
9220 NAC(IJ2)=B4
9227 GOTO 9260
9230 IJ3=IJ3+1
9235 NAP(IJ3)=B3
9240 IF IJK >< IMX THEN GOTO 9275
9245 IJ3=IJ3+1
9250 NAP(IJ3)=B4
9260 IF LLX=1 THEN GOTO 9275
9263 IF IJK >< 1 THEN GOTO 9275
9265 IJ4=IJ4+1
9270 NSTE(IJ4)=B1
9275 RETURN
9300 REM SUBROUTINE - SEARCHING PROCESS FOR THE DETERMINATION OF THE -----

```

```
9310 REM REGION NUMBER -----
9320 IF E=2 THEN GOTO 9425
9330 FOR I=1 TO NUMEL
9350 IF I1 >< COMB(I,3) THEN GOTO 9420
9370 IF I >< 1 THEN GOTO 9390
9380 REGION=1:REGN$="OK":GOTO 9490
9390 IF I >< 2 THEN GOTO 9400
9395 REGION=2:REGN$="OK":GOTO 9490
9400 IF I >< 3 THEN GOTO 9420
9410 PRINT AA$:REGION=3:REGN$="OK":GOTO 9490
9420 NEXT I
9425 GOTO 9490
9425 IF KP >< IMY THEN GOTO 9490
9430 FOR I=1 TO NUMEL
9440 IF I1 >< COMB(I,4) THEN GOTO 9480
9455 IF I >< 1 THEN GOTO 9470
9460 ECOM=1 : GOTO 9480
9470 IF I >< 2 THEN GOTO 9477
9475 ECOM=2 :GOTO 9480
9477 IF I >< 3 THEN GOTO 9480
9479 ECOM=3
9480 NEXT I
9490 RETURN
```

Appendix G

ISOPARAMETRIC MESH GENERATION SCREEN OUTPUTS

