INTERACTIVE PROCEDURE FOR FINITE ELEMENT METHOD:

A SOFTWARE ENGINEERING APPROACH.

BY

RAPHAEL OGUGUA ANUMBA

A Dissertation submitted to the Postgraduate School, Ahmadu Bello University, in partial fulfillment of the Requirements for the Degree of Ph.D. Computer Science,

Department of Mathematics,

Faculty of Science,

Ahmadu Bello University, Zaria

July, 1991

DECLARATION

I hereby declare that this Research has been carried out by me, under the supervision of Dr. G.A. Olayi, Dr. S.C. Mathur and Prof. M.A. Rashid. I also wish to state that, the outcome of the research is a true finding of mine, and is in no way part of another researcher's work.

July 1991

CERTIFICATION

This Thesis entitled **Interactive procedure for Finite Element Method: A Software Engineering Approach**, by Raphael Ogugua Anumba meets the regulations governing the award of the degree of Ph.D. Computer Science of Ahmadu Bello University, and is approved for its contribution to Scientific knowledge and Library presentation.

---------- Date: 6/7/94
(Chairman, Supervisory Committee)

---------- Date: 2/6/94
(Member, Supervisory Committee)

---------- Date: 23/9/92
(Head of Department)

---------- Date: 2/2/94
(Dean, Post-Graduate School)

ACKNOWLEDGEMENT

It is with the deepest gratitude and appreciation of many towards the success of this work that I convey my sincere acknowledgement of their contribution.

Firstly, I will thank my Supervisors, Dr. G.A. Olayi and Dr. S.C. Mathur, particularly Dr. Olayi who had continued the supervision of my work from his new station; Dr. Makarfi, the Head of Department; Professor Rashid the Post Graduate Coordinator of the Department of Mathematics who saw me through the problem of supervisors; and all members of the Department of Mathematics who contributed in various ways during the departmental seminars this research.

Secondly, I extend my sincere gratitude to the Vice-Chancellor of Abubakar Tafawa Balewa University, Bauchi, Professor Buba Bajoga, who gave me all the necessary support through out the research; Professor Mshelia, Professor Ale, Dr. E.J.A. Edemenang, all of Abubakar Tafawa Balewa University, Bauchi, whose pieces of advice I will never forget.

Thirdly, to my wife whose love, care and encouragement held me through the long period of this research.

Fourthly, I thank the Abubakar Tafawa Balewa University for being my main sponsor and for providing all the research funding under the direct supervision of research grant committee.

Finally, I am most grateful to the entire staff of the Mathematics Programme and the University Computer Centre of Abubakar Tafawa Balewa University, Bauchi, with special thanks to Mal. Hamidu Bukar for typing the whole thesis.

ABSTRACT

The study, design and implementation of a comprehensive Program aimed at relieving the Finite Element Analyst of the rigors of correctly using the computer to implement his technique, was presented in this thesis.

It was principally meant to remove the constraint on the so much required public literacy in the use of the potentials of Finite Element Method in problem solving and also increase the non-computer professionals desire for Finite Element Method as a tool, that will help to describe in an explicit manner, the steps that will realize his intentions.

The research started from a theoretical standpoint, with the stages of the Finite Element Method being isolated and transformed, for cognition and easy manipulation, to a well structured techniques based on Software Engineering Approach using an empirical variation functional technique.

The resulting interactive Software tool, which makes use of electronic sheets and subsheets has an edge over most conventional methods. It has also the capability of storage conservation and program expansion facilities. Though interactively slow, it is hoped that the design will be very useful in most applications that will involve the use of finite elements.

TABLE OF CONTENTS

Chapters		Page
	Title page	i
	Declaration	ii
	Certification	iii
	Acknowledgement	iv
	Abstract	v
	Table of Contents	vi
	List of Tables	x
	List of Figures	xii
	List of Appendices	xv
	Specification of Flowchart Symbols	xvi
1	Introduction	1
1.1	Finite Element Method	1
1.2	The General Procedure	3
1.3	Difficulties	4
1.4	Requirements	6
1.5	Communication Languages/Problems	6
1.6	Software Development Methodology	8
1.7	Rational of the Research	8
1.8	The Proposed Research	10
1.8.1	Theory	10
1.8.2	Programming	11
1.9	Research Methodology	12

Chapters		Page
1.10	Facilities Available	13
2	The Theory of Finite Element Method	15
2.1	Introduction	15
2.2	General Finite Element Procedure	15
2.2.1	Nodal and Element Characteristics	15
2.2.2	Approximation of Elements Behaviour and Equation	20
2.2.3	Assembly and Solution of Equation	22
2.3	Variational Formulation of the Finite Element Method	24
2.3.1	Problem Definition	25
2.4	Assembly of Element Equations into Global Equation	32
3	Development of Finite Element Method Software..	41
3.1	Computer Implementation of Finite Element Method	41
3.1.1	Program Structures	43
3.2	The PREPROCESSOR	45
3.2.1	Procedure Set-up-Control-Parameter SCON	47
3.2.2	Setting up the Mesh Coordinate, SMESH	49
3.2.3	Procedure to set up the Element Data,SELE	49
3.2.4	Procedure to Read Boundary Conditions SBC	50
3.2.5	Procedure to Read the Element Material Properties SMAT	52
3.2.6	Procedure to Compute the Band width of the Problem SBAND	55

Chapters		Page
3.3	The PROCESSOR	55
3.3.1	The Material property Routine-GETMAT.....	58
3.3.2	Routine to Calculate the Shape Function-SHAPE.	58
3.3.3	Element Routine - ELEMT	60
3.3.4	Assembly Routine - ASSEMB	64
3.3.5	Boundary Conditions Routine - APLYBC	66
3.3.6	Reduction of the Stiffness Matrix to Triangular Form-TRI	69
3.3.7	Back Substitution Routine - BACK	71
3.4	The POSTPROCESSOR - POST	73
3.4.1	The Element and Exact Solution Routine-EVAL ..	75
3.4.2	Calculating of Exact Solution Routine-EXACT..	77
3.5	Programming Environment	77
4.	The Program Implementation	81
4.1	The Software System	81
4.2	The Computational SOLVER	83
4.3	The Sheets	84
4.3.1	The Subsheets	86
4.3.2	The Structure of the Sheets	88
4.3.2.1	The Variable Sheet and Subsheet	90
4.3.2.2	The Rule Sheet	94
4.3.2.3	The Function Sheet and Subsheets	95
4.3.2.4	Unit Sheet	100
4.3.2.5	List Sheet and Subsheet	103
4.3.2.6	Table Sheet and Subsheet	105

Chapters		Page
4.3.2.7	Number Format Sheet and Subsheet	107
4.3.2.8	Plot Sheet and Subsheet	111
4.3.2.9	The Global Sheet	112
4.4	The SOLVER Implementation	112
4.4.1	The Tools of the PREPROCESSOR	113
4.4.2	The Tools for the PROCESSOR	116
4.4.3	The Tools for the POSTPROCESSOR	118
4.5	The Central List Subsheet	120
5.	Experimental Results and Software Evaluation ..	122
5.1	Heat Conduction and Convection in two Dimensional Body	123
5.2	Irrotational Flow of Fluid	131
5.3	One Dimensional Fluid Flow between flat Permeable plates	144
5.4	Finite Element Problem on Hydrostatic pad	148
5.5	Software Evaluation	157
5.6	Experimental Framework	160
5.6.1	Programming Counting Tools	163
5.6.2	Measurement of Program Efficiency.....	173
6.	Summary and Conclusion	182
6.1	Program Development.....	188
6.2	Experimental Results.....	194
6.3	Expandability.....	195
6.4	Drawback and Limitation	196
	REFERENCES.....	197

LIST OF TABLE

TABLE		PAGE
2.1	Commonly used polynomial	21
2.2	Element Numbers and their corresponding global coordinates	34
5.1	Boundary condition for Heat Conduction and Convection in 2D Body	125
5.2	Elements and their Associated Nodes for the Heat Conduction problem	127
5.3	The Nodal Temperature Distribution for 8 element	127
5.4	Prescribed Boundary Condition for irrotational Fluid Flow	135
5.5	The Stream function and velocity potential distribution 13 node of 13 elements	136
5.6	The global nodes for 13 elements for irrotational fluid flow.....	137
5.7	Stream function and velocity potential at various nodes for 300 elements.....	140
5.8a	Solution for 3 elements	145
5.8b	Solution for 49 elements	146
5.9	The pressure distribution for the Hydrostatic pad problem	150
5.10	Pressure distribution for 300 elements.....	153
5.11	Velocity potential of irrotational flow of section 5.2, using Triangular elements for selected nodes..	159

TABLE		PAGE
5.12	Weight assigned to symbolic name elements.....	164
5.13	Weight assigned to constant elements.....	165
5.14	Weight assigned to operator elements.....	165
5.15	Weight assigned to statement type elements.....	166
5.16	Statement types by assigned weights.....	169
5.17	Weight assigned to other program elements.....	170
5.18	Factors for data type of symbolic name element..	171
5.19	Factors for other attributes.....	171
5.20	Weight of program segments.....	180
5.21	The comparition of the Benchmarks with the SOLVER.....	181

LIST OF FIGURES

FIGURES		PAGE
2.1	Assemblage method of four triangular element ..	33
3.1	Main programming Call of PREP, PROC and POST ..	44
3.2	Sub divided procedures of PREP	46
3.3	Procedure to set up control parameters	48
3.4	Procedure to Read in the Boundary Conditions ..	51
3.5	Procedure to Read Element Material Properties ..	53
3.6	Sub divided procedures of PROC	56
3.7	Procedure to calculate the shape function	59
3.8	Procedure that calculate the Element Stiffness Matrix and Load Vector	61
3.9	Procedure that does the Assemblage of the Stiffness matrix and load vector	65
3.10	Procedure for applying the prescribed boundary conditions	67
3.11	Procedure to reduce the stiffness matrix to triangular form-TRI	70
3.12	procedure to do the back substitution for the Nodal solution	72
3.13	Sub divided procedure for the Post processor-POST.	74
3.14	The Element and Exact solution Routine	76
4.1	Relationship between Sheets and Subsheets	85
4.2	The Variable Sheet	91
4.3	Variable Subsheet	91
4.4	The Rule Sheet	93

4.5 Function Sheet 93

FIGURES PAGE

4.6 List Function Subsheet 96
4.7 Rule Function Subsheet 96
4.8 Procedure Function Subsheet 99
4.9 Unit Sheet 102
4.10 List Sheet 102
4.11 List Subsheet 104
4.12 Table Sheet 104
4.13 Table Subsheet 106
4.14 Interactive Table Subsheet 106
4.15 Numerical Format Sheet 106
4.16 Numerical Format Subsheet 108
4.17 Plot Sheet 108
4.18 Line Plot Subsheet 110
4.19 Bar Plot Subsheet 110
4.20 Pie Plot Subsheet 110
4.21 The Tools for PREPROCESSOR Routines 115
4.22 The Tools for the PROCESSOR Routines 117
4.23 The Tools for the POSTPROCESSOR Routines 119
4.24 The Central List Subsheet 121
5.1 Problem Geometry and Boundary Condition 124
5.2 Temperature Distribution for 360 elements 132
5.3 Temperature Distribution for 1440 elements 133
5.4 Geometry of Irrotational flow problem 134

FIGURES		PAGE
5.5	The Distribution of the Stream Function for 400 elements	138
5.6	Plots of Variations Number of Element for Fluid Flow between Permeable Plates	147
5.7	Pad Geometry of the Hydrostatic Pad problem ...	149
5.8	The Pressure Distribution for 500 Elements	151
5.8a	The Pressure Distribution for 1400 Elements ...	152

INTRODUCTION

1.1 *Finite Element Method*

The Finite Element Method we present, is a general technique for constructing solutions to boundary value problems. The method involves dividing the domain of the solution into a finite number of simple subdomains - the finite element, and using variational concepts to construct an approximate solution over the collection of finite elements. Because of the generality and richness of the ideas underlining the method, it has been used with remarkable success in solving a wide range of problems in virtually all areas of engineering and mathematical sciences.

To render the problem amenable to numerical treatment, the infinite degree of freedom of a system are discretized or replaced by a finite number of unknown parameters as indeed is the practice in other processes of numerical approximation.

The method which was developed in the early 1950's in the aerospace industry, was first presented by Turner et al, (1956) as a publication. Their publication stimulated other researches and resulted in several technical articles that discussed the application of the method to structural and solid mechanics. In 1963, Melosh was able to show that the finite element method was really a variation of Raleigh-Ritz procedure.

The application of the Finite Element Method to some physical problem like conduction of heat, started with publication by Zienkiewicz et al (1965); Visser (1965), and Wilson et al (1966). The application enlarged when other investigators like Szab et al (1969) and Zienkiewicz were able to show that the element equation has some relation to structural mechanics, heat transfer and fluid mechanics could also be derived by using a weighted residual procedure such as Galerkin's method or the least-square approach. This knowledge became a very important contribution to the theory of Finite element, because it allowed the application of the method to many different equations. Mitchell et al (1977) concentrated on the techniques for solving partial differential equation. Their work was intended to bridge the gap between the engineering works of Zienkiewicz (1971) and the Mathematical works of Strang et al (1973).

Subsequent development has been rapid, and the techniques are now well established within many scientific and engineering fields. This advancement has been accomplished in a period of 27 years and has been assisted by the development of high-speed digital computers, which provided a rapid means of performing the heavy calculation involved.

1.2 The General Procedure

With the above development, the Finite Element method became an important and practical numerical analysis tool. It has found application in almost all areas of Engineering and applied Mathematics. The Literature on Finite Element Method is extensive and rapidly increasing. Extensive bibliographies are available but are still incomplete. Numerous text presented the theory of various Finite Element procedures and most of them relegated programming consideration only to the background, secondary or lower level Zienkiewicz (1971), Seglerlind (1976) Mitchell et al (1977) and Rao (1982), with exception of few like Norrie et al (1975) Taylor (1977) and Akin (1982).

The relegation of programming to the background must have resulted from the failure in the prospect of a single package which would be useful for all types of science and engineering problems resulting to partial differential equation.

Though some of such packages like:

PDEL by Cardenas et al (1970) LEAN III by Schiesser (1972) and PDELAN by Gary et al (1972)

were proposed and implemented, their general approach were academic and could not stand the test of time. The development

of these packages had been particularly intensive in structural analysis and includes:

ASKA - for Automatic System for Kinematic Analysis

STRUDL (ICES) - A STRUctural Design-Language in the Integrated Civil Engineering System.

1.3 Difficulties

Invariably, in designing a general-purpose programming system, speedy execution of the programme is always sacrificed for programming convenience Berztiss (1975). The more general a package becomes, the more inefficient it is likely to be for specific problems. Frank et al (1983), concluded that a more concerted effort is highly needed in the area of the economics of designing general software, since the software must not consume more than certain amount of computer core memory, nor tie up more than a certain number of peripherals, and it must achieve some specified response time.

For small problems, the loss of efficiency in programming is of minimal importance for an inexperienced programmer to obtain meaningful results. On the other hand, for large problems that arise in Fluid Dynamics, Nuclear reaction Technology, Structural Engineering to mention a few, it is of

vital economic importance (both in cost of computing time and analysis as well as storage consumption) to design an efficient program.

In the former situation, the programming ease is necessarily sacrificed, and therefore, only experienced programmers can tackle the Finite Element problems. Attempts have been made in the past to achieve meaningful engineering results by overcoming the difficulty in learning a Computer oriented language and employing professional programmers, but it has been found that they lack good background of Finite Element Method.

Later, the design of a general-purpose package, that is, a design to facilitate the solving of field problem regardless of application area started giving way to an intensive effort in a number of fields of endeavour to develop packages tailored specifically for users from a specific application discipline, Newmark (1959).

This permitted the inclusion of features and terminologies that would have no place in a general-purpose package and recently, Morley (1981), came up with the production of a specialized language which uses engineering terms as its basic input.

Apart from these improvements, the problem still remains that it is only the computer experts that can write some efficient and economical programmes, if they understand the Finite Element methods, Akin (1982). This is where the difficulty started and had been a big barrier, since the nature of communication language between the Finite Element Analyst (non computer professional) and the Computer, does not cognitively fit into the Analyst problem solving skills.

1.4 Requirements

While it is generally believed that everyone need not be a professional programmer, it is of increasing importance to describe in explicit manner and in a convenient way, to the computer, the steps that are required to achieve one's intentions.

1.5 Communication Languages/Problems

The problem of translating correctly analyzed Finite Element problems into their Computer equivalent is still an imperative sophistication (or complexity), particularly with some complex real-life problems. A programming language choice will not be out of place, if there will be adaptable tools (or methods) that will assist or translate with ease, the well

organized Finite Element Method into its computer equivalent.

If this can be achieved, then, the finite element analyst's problems are solved to some extent, because he is now rest assured that his ideas can correctly and perfectly be translated or transformed into a computer language via the designed tools or methodology. Moreover, he will now have to concentrate on enhancing the application of Finite Element method rather than concentrating on ways of solving a limited number of problems.

Although a number of programming languages are currently available, only some, such as PL/I, FORTRAN AND BASIC are considered suitable for solving problems of Finite Element Method, Akin (1981) and Cardenas et al (1970). Some of these languages subject the user to adapt to the constraints and implicit strategies facilitated by them, almost all of them presently have available efficient facilities onto which structured equivalent of Finite Element Method will be translated through some tools. Generally, people tend to program correctly, easily and perfectly when the language facilitates their preferred strategy Berns(1984). Most recently Packages, which include Rule, Function, Table and Plot windows, had become valuable tools for achieving interactive results.

1.6 Software Development Methodology

Traditionally, preparation of a program meant that one prepares a flow chart, Aho (1974) and write the code while sprinkling the code with an occasional comments. As software development cost are mounting, writing and maintaining program became a serious business, research in the development of tools or areas of programming methodology which will ease the transformation of problems that will reduce these costs started to emerge. One of these approaches is the Software Engineering Techniques, which is principally based on the aim of correctness, clarity, comprehensibility, modifiability and maintainability, Berns (1984).

This approach can lead to a development in terms that match intuitive data type and operations, that is, the concept used to understand a problem originally, and generally the best concept to be used in writing program to solve the problem.

1.7 Rationale of the Research

Since there had been advancement in the theory and use of the generality and potentials of Finite Element Method in solving many scientific and engineering problems, there is need

for correct and satisfactory transformation programming. There is also need for the removal of the constraints placed, as a result of the steps of this method, which normally results to an imperative use of computer in achieving meaningful results.

So it is on the bases of these constraints and their concomitant problems that we have embarked, through a software engineering approach, on the search for some simple, easy and less complex method and procedure to be used by non-computer professionals, which will not be far from the conventional steps of Finite Element Method.

This approach will remove the constraint on the so much required public literacy in the use of the potentials of Finite Element Method in problem solving and will also increase the non-computer Professionals desire for Finite Element Method as a tool, to be able to describe in an explicit manner to the computer the steps that is required to realize his intentions.

1.8 The Proposed Research

1.8.1 Theory

It is proposed to make a general study of the procedure and steps of Finite Element method. These studies will include:

(i) The study of the formulation, generating theory and numerical procedure of Finite Element Method.

(ii) The study of the problems and important computational aspects pertaining to the understanding and implementation of the Finite Element methods.

(iii) To study and utilize the large generality of Finite Element Method Analysis Techniques over other numerical methods.

(iv) To study the potentials by utilizing many elements to variationally approximate any continuum with complex boundaries and loading conditions, to some degree that an accurate analysis can be attained.

1.8.2 Programming

As we intend to design, structure and implement the theoretical studies in a more efficient manner, it is desired that the major programming will concentrate on:

(i) The reduction of cost of analysis, since the increase of the number of elements also increases the means required for relevant data and interpretation of result, and large amount of time.

(ii) To find a way of reducing the much pronounced limiting problem placed by program and computer in the use of large computer storage in the idealization of continuum.

(iii) To develop some programming methodologies to bridge the language communication gap between the Finite Element method and Computer implementation.

1.9 Research methodology

It is intended that concentration will be on problems resulting to partial differential equations, this will be broken down to their equivalent but approximate set of algebraic equations. Variational technique will be used to achieve functionals which may actually represent some physical quantities in the problem to obtain lower-order derivatives, Bathe (1982).

Restrictively, it is always possible to find an equivalent variational formulation to a given differential equation, so the method of derivation of the required equilibrium equation by using a weighted residual method such as Galerkin method, Bathe (1982), which has considerably enlarged the range of application of Finite Element method, will also be considered.

Functionals will be subjected to various trial or shape functions, Allaire (1985), and structured, not only to study their potentials in solving the language communication problems, but also to see to what extent the various approaches will reduce the storage limitation and the enhancement of accuracy.

Programming techniques, based on the method and theory that will allow planning, design development and possibly maintenance of software for modern computers, will concentrate on the following:

(i) Data manipulation of element incident list for linear and isoparametric elements.

(ii) Calculating of element characteristic and equivalent nodal load.

(iii) Assemblage of element characteristic matrix and load vector onto their global equivalent. To ascertain the efficiency of the adopted approach, application will be made to some special and common field problems like Fluid Dynamics, Heat Transfer, Solid and Structural Mechanics and other empirically application fields.

1.10 Facilities Available

The Kashim Ibrahim Library of the Ahmadu Bello University, Zaria and the Abubakar Tafawa Balewa University (ATBU) Library of Science and Technology served as the main source of Literature.

All Computer work was done at the Abubakar Tafawa Balewa University Computer Centre, which is equipped with various Computer Systems, including the interactive VAX 11/780 VMS, with a memory capacity of 4 MegaByte and having FORTRAN, PASCAL and BASIC compilers with debugging facilities, and an online CALCOMP plotting package. The Centre is also equipped with IBM PC MicroComputer.

Chapter 2.

2. The Theory of Finite Element Method

2.1 Introduction

Finite element models of various problems have been formulated from simple physical intuition and from mathematical principles. Historically the use of physical intuition led to several early practical models. However, today there is increased emphasis on the now well established mathematical foundations of the procedure Babuska et al (1972), Debor (1974), Oden et al (1976).

2.2 General Finite Element Procedure

2.2.1 Nodal and Element Characteristics

In the finite element method, the solution domain is subdivided, into a finite number of discrete-sized subregion called finite elements. A discrete number of nodal points are established with the imaginary mesh that divides the region into elements ranging from line (one dimensional) to solid (three dimensional) elements. These nodes are often located at intersecting mesh line but can also be located inside the element.

The nodes are numbered or assigned identifying integer

numbers (node numbers) beginning from unit and ranging to some maximum value. Each element is also given an identification number which also begin with unit and extend to the maximum number of elements. The number of the nodes and element have some significant effect on the accuracy, solution time and storage requirement. Each node has its local and global number.

A number of degree of freedom is usually assigned to each node (usually uniform). The number of degree of freedom in the system, say n_g , is the product of the number of nodes, N and the number of degree of freedom per node n_i

$$n_g = N * n_i$$

Similarly, the number of degree of freedom per element is the product of the number of nodes of the element, M, and the number of degree of freedom per node n_i :

$$n_e = M * n_i$$

A system with N=5 nodes, n_i = 6 degree of freedom and M=2 local nodes will have a system degree of freedom:

n_g = 5*6 = 30 and element degree of freedom

n_e = 2 x 6 = 12

Each of these degrees of freedom will have a numeric number for every node of the element such that the degree of freedom number D_{ij} of the jth parameter at the system nodal number i is defined as:

$$D_{ij} = n_i * (i-j) + j, \quad 1 < i < N, \quad i < j < n_i$$

so that $1 \leq D_{ij} \leq n_g$ Graham et al (1984).

In addition to the above constants the space dimension of the problem can also be calculated and these quantities can be used to calculate the size of the storage requirement.

An important concept is that of the element connectivity, i.e. the list of global node numbers that are attached to an element. The element connectivity defines the topology of the mesh. Thus each element requires (in some consistent order) data for all the node number that are associated with a particular element. This list of node numbers connected to a particular element is the element incident list, Akin (1982).

Several approaches to mesh generation and refinement had been presented. These include that of Jones (1974) and Sloan et al (1983). A study on mesh generation method gave the algorithms for simple meshes in one and two dimension.

(i) A uniform mesh of linear elements on interval (a,b) is generated thus:

1. h = (b-a)/n:

x_1 =a and n=number of elements.

2. For i = 1 to n by 1

x_{i+1} = x_i + h : defines the node location

3. Element i is connected to node i and i+1

(ii) A uniform mesh of linear rectangular element on (a,b) x (c,d) in $\mathbb{R}^2$ is also generated thus

1. h = (b-a)/n, k=(d-c)/m

   x_i =a+(i-1)h for i=1 to n+1

   y_i =c+(j-1)k for j=1 to m+1

   such that m x n = N elements.

2. *For* j = 1 *to* m

   For i = 1 to n

   k=(j-1)n+i at the nodes (i,j)

   ∴ (x_k, y_k) give the node location for node k

3. Element e = (j-1) (n-1)+i, i ≠ n, j ≠ m is

   defined by node k, k+1, k+n and k+n+1.

2.2.2 Approximation of Element Behaviour and Equation

The variables of interest (and its derivatives) are uniquely specified throughout the solution domain by the nodal parameters associated with the nodal points of the system. It is these nodal parameter that are regarded as the unknown parameters of the problem. The parameter at a particular node influences only the values of the quantity of interest within the elements that are connected to that particular node.

Interpolation function is needed to relate the quantity of interest within the element in terms of the values of the nodal parameter at the nodes that are connected to that particular element. (i.e. element incident list). For computerization, it is easier to formulate the finite element equations with polynomial type of interpolation function. It is used specifically as it is easy to differentiate as well as integrate. The order of the polynomial can also be increased in order to improve accuracy, Strang et al (1973), Vemuri et al (1981).

Commonly Used Polynomial are Tabulated in Table 2.1. below:

Table 2.1. Commonly used Polynomial

Mode	1-D	2-D	3=D
n=1 Linear	$T(x) = \alpha_1 + \alpha_2 x$	$T(x,y) = \alpha_1 + \alpha_2 x + \alpha_3 y$	$T(x,y,z) = \alpha_1 + \alpha_2 x + \alpha_3 y + \alpha_4 z$
n=2 Quadratic	$T(x) = \alpha_1 + \alpha_2 x + \alpha_3 x^2$	$T(x,y) = \alpha_1 + \alpha_2 x + \alpha_3 y + \alpha_4 x^2 + \alpha_5 y^2 + \alpha_6 xy$	$T(x,y,z) = \alpha_1 + \alpha_2 x + \alpha_3 y + \alpha_4 z + \alpha_5 x^2 + \alpha_6 y^2 + \alpha_7 z^2 + \alpha_8 xy + \alpha_9 yz + \alpha_{10} xz$
n=3 Cubic	$T(x) = \alpha_1 + \alpha_2 x + \alpha_3 x^2 + \alpha_4 x^3$	$T(x,y) = \alpha_1 + \alpha_2 x + \alpha_3 y + \alpha_4 x^2 + \alpha_5 y^2 + \alpha_6 xy + \alpha_7 x^3 + \alpha_8 y^3 + \alpha_9 x^2 y + \alpha_{10} xy^2$	$T(x,y,z) = \alpha_1 + \alpha_2 x + \alpha_3 y + \alpha_4 z + \alpha_5 x^2 + \alpha_6 y^2 + \alpha_7 z^2 + \alpha_8 xy + \alpha_9 yz + \alpha_{10} xz + \alpha_{11} x^3 + \alpha_{12} y^3 + \alpha_{13} z^3 + \alpha_{14} x^2 y + \alpha_{15} x^2 z + \alpha_{16} xy^2 + \alpha_{17} y^2 z + \alpha_{18} xz^2 + \alpha_{19} yz^2 + \alpha_{20} xyz$

After the element behaviour had been approximated by a chosen polynomial, the next fundamental problem is the establishment of the element matrix and vector. Historically, the matrix and vector are known as the stiffness matrix and load vector respectively. The realization of these matrices involves the substitution of the interpolation function into the governing integral form.

Although these matrices are sometimes developed from intuition, they are usually formulated by the variational minimization of a functional or by the method of weighted residuals (Galerkin's method) Norrie et al (1975).

2.2.3 Assembly and Solution of Equations

Once the element equations have been established, the contribution of each element is added to form the system equations. This equation is usually represented in vectorial form as:

$$S\{T\} = \{V\}$$

where the square matrix S is ($n_g \times n_g$) in size and the vectors T and V contain n_g values each. The vector T will contain the unknown nodal parameters and the matrices S and V are obtained by assembling the element matrices. The element matrices S^e and V^e are ($n_e \times n_e$) and (n_e x 1) in size respectively.

The system matrix S is usually symmetric and banded about the diagonal. Thus, the half band width, including the diagonal is an important quantity in the consideration of any finite element solution. It has been established, Seglerlind (1976), that the half-bandwidth (or simply bandwidth), B, is calculated using

$$B = (R+1) * n_i$$

where R is the absolute value of the maximum difference between the number of the nodes connected to the element and n_i is the nodal degree of freedom.

The minimization of B depends on the minimization of R, which can be partially accomplished by labelling the nodes across the shortest dimension of the body. The quantities B, is one of the important values which govern the storage requirement and solution time of the system equation. The normal practice is to minimize the maximum difference in the node number associated with each element. Several finite element application do not utilize a banded matrix solution technique as they can employ a frontal solution of a sparse matrix solution, Rao (1982) and Bathe (1982).

Before solving for the unknown nodal parameters, the boundary constraint are applied to the system equation after the assembly. The common types of nodal parameter boundary constraints are:

(i) defining explicit values of the unknown on the boundary

(ii) defining constraint equation that are linear combinations of the nodal quantities.

Finally the system equations are solved by means of procedure which accounts for the sparse and symmetric nature of the problem. The commonly used method of solving system equations are:

(i) Gaussian Elimination

(ii) Choleski Factorization

(iii) Gauss-Choleski Factorization

2.3. Variational Formulation of the Finite Element Method

As the technique will attach importance to the area that interfaces the user with the details of the problem area, we will empirically used the variation method on some two-dimensional poisson equation, with a simplex trial function on a triangularly discretized region, Rao (1982).

2.3.1. Problem definition:

We considered a generalized functional

$$J=\iint_A [\frac{K_x}{2}(\frac{\partial T}{\partial x})^2+\frac{K_y}{2}(\frac{\partial T}{\partial y})^2-\frac{PT^2}{2}-QT]\,dA+\int_L [\frac{\alpha T^2}{2}-\beta T]\,dl \quad .....(2.1)$$

which is an equivalence of a general two dimensional boundary value problem of a homogenous material

$$\frac{\partial}{\partial x}(K_x\frac{\partial T}{\partial x})+\frac{\partial}{\partial y}(K_y\frac{\partial T}{\partial y})+P(x,y)\,T+Q(x,y)=0 \quad in \quad A \quad .....(2.2)$$

subject to the boundary condition

$$K_x\frac{\partial T}{\partial x}\eta_x+K_y\frac{\partial T}{\partial y}\eta_y+\alpha(x,y)\,T+\beta(x,y)=0 \quad on \quad L \quad .....(2.3)$$

the coefficient K_x and K_y are constant while η_x and η_y are the direction cosine, P, Q, α, β are all function of x and y with α and β defined on the boundary, L, alone. The continuum is the union of A and L. The domain, A, is assumed divided into E triangular elements, such that the general functional becomes

$$J = J^{(1)} + J^{(2)} + J^{(3)} + ... + J^{(E)} = \sum_{e=1}^{E} J^{(e)} \quad .....(2.4)$$

Vemuri (1981).

We now consider a particular element, e, and assume a simplex two dimensional trial function to approximate the field variable at the various local nodal points i, j, and k, such that if

$$T^{(e)} = \alpha_1 + \alpha_2 x + \alpha_3 y \quad \text{in the region} \quad A^{(e)} ,$$

then we have

$$T_i = \alpha_1 + \alpha_2 x_i + \alpha_3 y_i$$
$$T_j = \alpha_1 + \alpha_2 x_j + \alpha_3 y_j$$
$$T_k = \alpha_1 + \alpha_2 x_k + \alpha_3 y_k$$

Transformation will give

$$T^{(e)} = N_i T_i + N_j T_j + N_k T_K \qquad \text{.....(2.5)}$$

where

$$N_r = \frac{1}{2A^{(e)}} (a_r + b_r x + C_r y) \quad \text{for r=i, j, k.} \qquad \text{.....(2.6)}$$

(x,y) is the spatial location, T_i, T_j *and* T_k are the nodal parameters and

$$a_i = X_j Y_k - X_k Y_j, \quad b_i = Y_j - Y_k, \quad c_i = X_k - X_j$$
$$a_j = X_k Y_i - X_i Y_k, \quad b_j = Y_k - Y_i, \quad c_j = X_i - X_k \qquad \text{.....(2.7)}$$
$$a_k = X_i Y_j - X_j Y_i, \quad b_k = Y_i - Y_j, \quad c_k = X_j - X_i$$

(X ,Y), r = i,j,k are the coordinates of the element incident list. The area of the element is given as

$$A(e) = \frac{1}{2} (b_i c_j - b_j c_i) \qquad \text{.....(2.8)}$$

For optimal solution, it is required that the functional be minimized such that

$$\frac{\partial J}{\partial T} = \frac{\partial}{\partial T}\left(\sum_{e=1}^{E} J^{(e)}\right) = 0 \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (2.9)$$

$$T = \begin{pmatrix} T_i \\ T_j \\ T_k \end{pmatrix}$$

Evaluating equation (2.9) and substituting (2.5) where

$$\frac{\partial T^e}{\partial x} = \frac{\partial N_i}{\partial x} T_i + \frac{\partial N_j}{\partial x} T_j + \frac{\partial N_k}{\partial x} T_l$$

$$\frac{\partial T^e}{\partial y} = \frac{\partial N_i}{\partial y} T_i + \frac{\partial N_j}{\partial y} T_j + \frac{\partial N_k}{\partial y} T_l \quad \text{and}$$

$$\frac{\partial N_r}{\partial x} = \frac{1}{2A} b_r , \qquad \frac{\partial N_r}{\partial y} = \frac{1}{2A} c_r , \quad r=i, j, k.$$

such that

$$\frac{\partial T^e}{\partial x} = \frac{1}{2A}\left[b_i T_i + b_j T_j + b_k T_k\right]$$

$$\frac{\partial T^e}{\partial y} = \frac{1}{2A}\left[c_i T_i + c_j T_j + c_k T_k\right]$$

We now have

$$\frac{\partial J}{\partial T} = \frac{\partial}{\partial T_r}\left[\frac{1}{2}\iint_A \left[\frac{K_x}{4A^2}(T_ib_i+T_jb_j+T_kb_k)^2+\frac{K_y}{4a^2}(T_ic_i+T_jc_j+T_kc_k)^2\right.\right.$$
$$\left.-\frac{1}{2}P(N_iT_i+N_jT_j+N_kT_k)^2-Q(N_iT_i+N_jT_j+N_kT_k)\right]dA$$
$$\left.+\int_L\left(\frac{1}{2}\alpha(N_iT_i+N_jT_j+N_kT_k)^2-\beta(N_iT_i+N_jT_j+N_kT_k)\right)dL\right] = 0$$

.....(2.10)

were $T_r = \begin{pmatrix} T_i \\ T_j \\ T_k \end{pmatrix}$

i.e. $\frac{\partial J}{\partial T} = \begin{pmatrix} \frac{\partial J}{\partial T_i} \\ \frac{\partial J}{\partial T_j} \\ \frac{\partial J}{\partial T_k} \end{pmatrix} = 0$ .....(2.11)

such that

$$\frac{1}{2}\iint_A\left[\frac{b_i}{2A^2}(T_ib_i+T_jb_j+T_kb_k)+\frac{c_i}{2A^2}(T_ic_i+T_jc_j+T_kc_k)\right.$$
$$-N_iP(N_iT_i+N_jT_j+N_kT_k)-QN_i]\,dA$$
$$\left.+\int_L(\alpha N_i(N_iT_i+N_jT_j+N_kT_k)-\beta N_i)\,dL\right] = 0$$

$$\frac{1}{2}\iint_A\left[\frac{b_j}{2A^2}(T_ib_i+T_jb_j+T_kb_k)+\frac{c_j}{2A^2}(T_ic_i+T_jc_j+T_kc_k)\right.$$
$$-N_jP(N_iT_i+N_jT_j+N_kT_k)-QN_j]\,dA \qquad .....(2.12)$$
$$\left.+\int_L(\alpha N_j(N_iT_i+N_jT_j+N_kT_k)-\beta N_j)\,dL\right] = 0$$

$$\frac{1}{2}\iint_A\left[\frac{b_k}{2A^2}(T_ib_i+T_jb_j+T_kb_k)+\frac{c_k}{2A^2}(T_ic_i+T_jc_j+T_kc_k)\right.$$
$$-N_kP(N_iT_i+N_jT_j+N_kT_k)-QN_k]\,dA$$
$$\left.+\int_L(\alpha N_k(N_iT_i+N_jT_j+N_kT_k)-\beta N_k)\,dL\right] = 0$$

Integrating (2.12) with respect to A and L and taking

$$\iint_A b_i b_i dA = b_i b_i, \quad \iint_A b_j b_i dA = b_j b_i$$
$$\iint_A N_i^2 dA = \tfrac{1}{6}A, \quad \iint_A N_i N_i dA = \tfrac{1}{12}A$$

$$\int_L N_i dL = \int_L N_k dL = \frac{1}{2}, \quad \int_L N_i^2 dL = \frac{L}{3}, \quad \int_L N_i N_j dL = \frac{L}{6}$$

we obtain

$$\frac{K_x b_i}{4A}(T_i b_i + T_j b_j + T_k b_k) + \frac{K_y c_i}{4A}(T_i c_i + T_j c_j + T_k c_k) - \frac{PA}{12}(2T_i + T_j + T_k) - \frac{QA}{3} + \frac{\alpha L}{6}(2T_i + T_j + T_k) + \frac{\beta L}{2} = 0 \quad \text{.....(2.13)}$$

$$\frac{K_x b_j}{4A}(T_i b_i + T_j b_j + T_k b_k) + \frac{K_y c_j}{4A}(T_i c_i + T_j c_j + T_k c_k) - \frac{PA}{12}(T_i + 2T_j + T_k) - \frac{QA}{3} + \frac{\alpha L}{6}(T_i + 2T_j + T_k) + \frac{\beta L}{2} = 0 \quad \text{.....(2.14)}$$

$$\frac{K_x b_k}{4A}(T_i b_i + T_j b_j + T_k b_k) + \frac{K_y c_k}{4A}(T_i c_i + T_j c_j + T_k c_k) - \frac{PA}{12}(T_i + T_j + 2T_k) - \frac{QA}{3} + \frac{\alpha L}{6}(T_i + T_j + 2T_k) + \frac{\beta L}{2} = 0 \quad \text{.....(2.15)}$$

When written in matrix form, the element continuum equation becomes:

$$\frac{K_x}{4A}\begin{pmatrix} b_ib_i & b_ib_j & b_ib_k \\ b_jb_i & b_jb_j & b_jb_k \\ b_kb_i & b_kb_j & b_kb_k \end{pmatrix} + \frac{K_y}{4A}\begin{pmatrix} c_ic_i & c_ic_j & c_ic_k \\ c_jc_i & c_jc_j & c_jc_k \\ c_kc_i & c_kc_j & c_kc_k \end{pmatrix} \qquad \text{.....(2.16)}$$

$$-\frac{PA}{12}\begin{pmatrix} 2 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 2 \end{pmatrix} + \frac{\alpha L}{6}\begin{pmatrix} 2 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 2 \end{pmatrix}\begin{pmatrix} T_i \\ T_j \\ T_k \end{pmatrix} - \frac{QA}{3}\begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} - \frac{\beta L}{2}\begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$$

which is the same as

$$S\{T\} = \{V\} \qquad \text{.....(2.17)}$$

S is the element stiffness matrix and $\{V\}$ the element load Vector.

A more compact equation of (2.16) and (2.17) becomes

$$\begin{pmatrix} S_{ii} & S_{ij} & S_{ik} \\ S_{ji} & S_{jj} & S_{jk} \\ S_{ki} & S_{kj} & S_{kk} \end{pmatrix}\begin{pmatrix} T_i \\ T_j \\ T_k \end{pmatrix} - \begin{pmatrix} V_i \\ V_j \\ V_k \end{pmatrix} \qquad \text{.....(2.18)}$$

Noting that L is specifying all the boundaries of the element, L_{ij}, L_{jk} and L_{jk} under consideration, and that a maximum of two sides can be evaluated depending on the location of the element under consideration, we define a "natural" alternative form of equation (2.18), the stiffness matrix Allaire (1985).

$$S_{ii} = \frac{K_x b_i b_i}{4A} + \frac{K_y c_i c_i}{4A} - \frac{AP}{6} + \frac{\alpha L_{ij}}{3} + \frac{\alpha L_{ki}}{3}$$

$$S_{ij} = \frac{K_x b_i b_j}{4A} + \frac{K_y c_i c_j}{4A} - \frac{AP}{12} + \frac{\alpha L_{ij}}{3}$$

$$S_{ik} = \frac{K_x b_i b_k}{4A} + \frac{K_y c_i c_j}{4A} - \frac{AP}{12} + \frac{\alpha L_{ij}}{3}$$

$$S_{ji} = \frac{K_x b_j b_i}{4A} + \frac{K_y c_j c_i}{4A} - \frac{AP}{12} + \frac{\alpha L_{ji}}{3}$$

$$S_{jj} = \frac{K_x b_j b_j}{4A} + \frac{K_y c_j c_j}{4A} - \frac{AP}{6} + \frac{\alpha L_{ij}}{3} + \frac{\alpha L_{jk}}{3} \quad \text{.....(2.19)}$$

$$S_{jk} = \frac{K_x b_j b_k}{4A} + \frac{K_y c_j c_k}{4A} - \frac{AP}{12} + \frac{\alpha L_{ji}}{3}$$

$$S_{ki} = \frac{K_x b_k b_i}{4A} + \frac{K_y c_k c_i}{4A} - \frac{AP}{12} + \frac{\alpha L_{ik}}{3}$$

$$S_{kj} = \frac{K_x b_k b_j}{4A} + \frac{K_y c_k c_j}{4A} - \frac{AP}{12} + \frac{\alpha L_{jk}}{3}$$

$$S_{kk} = \frac{K_x b_k b_k}{4A} + \frac{K_y c_k c_k}{4A} - \frac{AP}{6} + \frac{\alpha L_{kj}}{3} + \frac{\alpha L_{ki}}{3}$$

It could be noticed that S_{ji}-S_{ij}, S_{ki}-S_{ik} *and* S_{kj}-S_{jk} .

The load vector can also be written in the form

$$V_i = \frac{AQ}{3} - \frac{\beta L_{ij}}{3} - \frac{\beta L_{ki}}{2}$$

$$V_j = \frac{AQ}{3} - \frac{\beta L_{jk}}{3} - \frac{\beta L_{ij}}{2} \quad \text{.....(2.20)}$$

$$V_k = \frac{AQ}{3} - \frac{\beta L_{ik}}{3} - \frac{\beta L_{jk}}{2}$$

where

$$L_{ij} = \sqrt{(X_j-X_i)^2 + (Y_j-Y_i)^2}$$

$$L_{jk} = \sqrt{(X_k-X_j)^2 + (Y_k-Y_j)^2} \quad \text{.....(2.21)}$$

$$L_{ki} = \sqrt{(X_i-X_k)^2 + (Y_i-Y_K)^2}$$

2.4 Assembly of Element Equations into Global Equation

Once the element stiffness matrix (equation 2.19) and the load vector (equation 2.20) are found in their common global coordinates system, the next step is to construct the global stiffness matrix and the global load vector.

Each element is considered in turn and the element nodal numbers matched to the proper location in the global matrices. We considered a triangular element where the element

coefficient matrix is 3 x 3 (equation 2.18) and the element column is of order 3, such that if K is the global stiffness matrix and {P} is the global Load vector, then K can be obtained by algebraic addition as

$$K = \sum_{e=1}^{N} S^e$$

and $$\{P\} = \sum_{e=1}^{N} \{V\}^e \quad \text{.....(2.22)}$$

where S^e can be regard as the expanded element stiffness matrix and $\{V\}^e$ is the expanded element load vector. The expansion is to the size of the global stiffness matrix and load vector respectively. The procedure of assemblage can be illustrated with reference to the assemblage of four triangular element each with one degree of freedom.

i - Element number.
j - Global Nodal number.

Fig 2.1. Assemblage Method for Triangular Elements.

Each node is numbered locally in anticlockwise direction, i,j,k, such that the following table will give the global nodal number in relation with their local nodal number.

Table 2.2. Element numbers and their corresponding global coordinates.

Element Number:	i	j	k	<---- local nodal number
1	1	2	3	
2	2	5	3	Global nodal number
3	2	4	5	
4	1	4	2	

If the equation represented by (2.18) is that of any element in Fig 1. then the expanded form of element 1, will be:

$$\begin{pmatrix} S_{ii} & S_{ij} & S_{ik} & 0 & 0 \\ S_{ji} & S_{jj} & S_{jk} & 0 & 0 \\ S_{ki} & S_{kj} & S_{kk} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}^{(1)} \begin{pmatrix} T_1 \\ T_2 \\ T_3 \\ T_4 \\ T_5 \end{pmatrix} - \begin{pmatrix} V_i \\ V_j \\ V_k \\ 0 \\ 0 \end{pmatrix}^{(1)} \quad \text{.....(2.23)}$$

for the 2nd element the expanded global equation is thus:

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & S_{ii} & S_{ik} & 0 & S_{ij} \\ 0 & S_{ki} & S_{kk} & 0 & S_{kj} \\ 0 & 0 & 0 & 0 & 0 \\ 0 & S_{ji} & S_{jk} & 0 & S_{jj} \end{pmatrix}^{(2)} \begin{pmatrix} T_1 \\ T_2 \\ T_3 \\ T_4 \\ T_5 \end{pmatrix} - \begin{pmatrix} 0 \\ V_i \\ V_k \\ 0 \\ V_j \end{pmatrix}^{(2)} \quad \text{.....(2.24)}$$

The expanded global equation for the 3rd element will result to:

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & S_{ii} & 0 & S_{ij} & S_{ik} \\ 0 & 0 & 0 & 0 & 0 \\ 0 & S_{ji} & 0 & S_{jj} & S_{jk} \\ 0 & S_{ki} & 0 & S_{kj} & S_{kk} \end{pmatrix}^{(3)} \begin{pmatrix} T_1 \\ T_2 \\ T_3 \\ T_4 \\ T_5 \end{pmatrix} - \begin{pmatrix} 0 \\ V_i \\ 0 \\ V_j \\ V_k \end{pmatrix}^{(3)} \quad \text{.....(2.25)}$$

and the 4th expanded global equation follows:

$$\begin{pmatrix} S_{ii} & S_{ik} & 0 & S_{ij} & 0 \\ S_{ki} & S_{kk} & 0 & S_{kj} & 0 \\ 0 & 0 & 0 & 0 & 0 \\ S_{ji} & S_{jk} & 0 & S_{jj} & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}^{(4)} \begin{pmatrix} T_1 \\ T_2 \\ T_3 \\ T_4 \\ T_5 \end{pmatrix} - \begin{pmatrix} V_i \\ V_k \\ 0 \\ V_j \\ 0 \end{pmatrix}^{(4)} \quad \text{.....(2.26)}$$

The global stiffness, K matrix can now be obtained

from the summation of equation (2.23) to (2.26). The same can be done for the global load vector {P} by applying equation - (2.22), such that:

$$K\{T\} = \{P\} \qquad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (2.27)$$

where

$$K = \begin{pmatrix} S_{ii}^{(1)}+S_{ii}^{(4)} & S_{ij}^{(1)}+S_{ik}^{(4)} & S_{ik}^{(1)} & S_{ij}^{(4)} & 0 \\ S_{ji}^{(1)}+S_{kj}^{(4)} & S_{jj}^{(1)}+S_{ii}^{(2)}+S_{ii}^{(3)}+S_{kk}^{(4)} & S_{jk}^{(1)}+S_{ik}^{(2)} & S_{ij}^{(3)}+S_{kj}^{(4)} & S_{ij}^{(2)}+S_{ik}^{(3)} \\ S_{ki}^{(1)} & S_{kj}^{(1)}+S_{ki}^{(2)} & S_{kk}^{(1)}+S_{kk}^{(2)} & 0 & S_{kj}^{(2)} \\ S_{ji}^{(4)} & S_{ji}^{(3)}+S_{jk}^{(4)} & 0 & S_{jj}^{(3)}+S_{jj}^{(4)} & S_{jk}^{(3)} \\ 0 & S_{ji}^{(2)}+S_{ki}^{(3)} & S_{jk}^{(2)} & S_{kj}^{(3)} & S_{jj}^{(2)}+S_{kk}^{(3)} \end{pmatrix} \qquad \ldots (2.28)$$

$$\{P\} = \begin{pmatrix} V_i^{(1)}+V_i^{(4)} \\ V_j^{(1)}+V_i^{(2)}+V_i^{(3)}+V_k^{(4)} \\ V_k^{(1)}+V_k^{(2)} \\ V_i^{(3)}+V_i^{(4)} \\ V_j^{(2)}+V_k^{(3)} \end{pmatrix} \quad \text{and} \quad \{T\} = \begin{pmatrix} T_1 \\ T_2 \\ T_3 \\ T_4 \\ T_5 \end{pmatrix}$$

As S is symetric according to (2.19), for a particular element e, $S_{ij}^{(e)} = S_{ji}^{(e)}$, $S_{ik}^{(e)} = S_{ki}^{(e)}$ and $S_{jk}^{(e)} = S_{kj}^{(e)}$ and then the global matrix K is a symmetric matrix such that:

$$K_{12} = S_{ij}^{(1)} + S_{ik}^{(4)} = K_{21} = S_{ji}^{(1)} + S_{ki}^{(4)}$$

$$K_{13} = S_{ik}^{(1)} = K_{31} = S_{ki}^{(1)}$$

$$K_{14} = S_{ij}^{(4)} = K_{41} = S_{ji}^{(4)}$$

$$K_{15} = 0 = K_{51} = 0$$

$$K_{23} = S_{jk}^{(1)} + S_{ik}^{(2)} = K_{32} = S_{kj}^{(1)} + S_{ki}^{(2)}$$

$$K_{24} = S_{ij}^{(3)} + S_{kj}^{(4)} = K_{42} = S_{ji}^{(3)} + S_{jk}^{(4)}$$

$$K_{25} = S_{ij}^{(2)} + S_{ik}^{(3)} = K_{52} = S_{ji}^{(2)} + S_{ki}^{(3)}$$

$$K_{34} = 0 = K_{43} = 0$$

$$K_{35} = S_{kj}^{(2)} = K_{53} = S_{jk}^{(2)}$$

$$K_{45} = S_{jk}^{(3)} = K_{54} = S_{kj}^{(3)}$$

The global load vector will then be:

$$\{P_1\} = V_i^{(1)} + V_i^{(4)}$$

$$\{P_2\} = V_j^{(1)} + V_i^{(2)} + V_i^{(3)} + V_k^{(4)}$$

$$\{P_3\} = V_k^{(1)} + V_k^{(2)}$$

$$\{P_4\} = V_i^{(3)} + V_j^{(4)}$$

$$\{P_5\} = V_j^{(2)} + V_k^{(3)}$$

The expansion of the element stiffness matrix and load vector to the global equivalent is just an illustration of the algebraic summation of the elements (equation 2.27), the local nodal degree of freedom numbers are translated to fit into the global ones.

Some of the stiffness coefficient and their load components are not yet of the general form described in (2.19) and (2.20) because they do not contain the boundary contributions. The derivative boundary conditions are actually included in the element matrices in (2.17) and during assembly in (2.28).

The prescribed or the geometric boundary conditions can be effected by modifying equation (2.27) at the nodal points to include the prescribed nodal values.

If equation (2.27) is considered a general banded matrix just assembled, and $T_i - \theta_i$ is a prescribed nodal value, then:

(1) all coefficient K_{ij} in row i are set to zero, $j \neq i$ and P_i is set to $K_{ij}\theta_i$

(2) Subtract $K_{ji}\theta_i$ from P_j and set K_{ij} to zero, $j \neq i$.

The resulting matrix will remain symmetric and banded.

Consider the global equation:

$$\begin{pmatrix} K_{11} & K_{12} & K_{13} & 0 & 0 \\ K_{21} & K_{22} & K_{23} & K_{24} & 0 \\ K_{31} & K_{32} & K_{33} & K_{34} & K_{35} \\ 0 & K_{42} & K_{43} & K_{44} & K_{45} \\ 0 & 0 & K_{53} & K_{54} & K_{55} \end{pmatrix} \begin{pmatrix} T_1 \\ T_2 \\ T_3 \\ T_4 \\ T_5 \end{pmatrix} - \begin{pmatrix} P_1 \\ P_2 \\ P_3 \\ P_4 \\ P_5 \end{pmatrix}$$

in which a geometric boundary condition is prescribed on the nodal value $T_2 - \theta_2$

The first transformation will be:

$$\begin{pmatrix} K_{11} & K_{12} & K_{13} & 0 & 0 \\ 0 & K_{22} & 0 & 0 & 0 \\ K_{31} & K_{32} & K_{33} & K_{34} & K_{35} \\ 0 & K_{42} & K_{43} & K_{44} & K_{45} \\ 0 & 0 & K_{53} & K_{54} & K_{55} \end{pmatrix} \begin{pmatrix} T_1 \\ T_2 \\ T_3 \\ T_4 \\ T_5 \end{pmatrix} - \begin{pmatrix} P_1 \\ K_{22}\theta_2 \\ P_3 \\ P_4 \\ P_5 \end{pmatrix}$$

The second transformation follows:

$$\begin{pmatrix} K_{11} & 0 & K_{13} & 0 & 0 \\ 0 & K_{22} & 0 & 0 & 0 \\ K_{31} & 0 & K_{33} & K_{34} & K_{35} \\ 0 & 0 & K_{43} & K_{44} & K_{45} \\ 0 & 0 & K_{53} & K_{54} & K_{55} \end{pmatrix} \begin{pmatrix} T_1 \\ T_2 \\ T_3 \\ T_4 \\ T_5 \end{pmatrix} = \begin{pmatrix} P_1 - K_{12}\theta_2 \\ K_{22}\theta_2 \\ P_3 - K_{32}\theta_2 \\ P_4 - K_{42}\theta_2 \\ P_5 - K_{52}\theta_2 \end{pmatrix}$$

The resulting system of linear equations can be solved directly by Gaussian elimination or the Choleski method; or iteratively by the Gauss-Seidel method.

Chapter 3.

DEVELOPMENT OF FINITE ELEMENT METHOD SOFTWARE

3.1 Computer Implementation of Finite Element Method

The success of the finite element method is due, in no small degree, to the ease with which the method can be implemented in the form of a general purpose computer programs. This also requires the use of a Computer and a sizeable amount of programming effort, in that it will be impossible for a potential user of Finite Element Method to program the entire code from scratch. The user will also be faced with a long, difficult and often impracticable task. It is therefore essential that the programming effort and knowledge of previous workers be utilized as much as possible.

A number of large finite element computer codes and packages had been developed, Marek et al(1980). Several of the early codes were developed at a great cost in both money, labour and time, yet they were highly specialized, machine dependent and difficult to modify, thus they rapidly became obsolete, Madsen et at (1976). The more desirable types of general purpose Finite Element Method program are those that are designed for comprehension, modification and possible updating by the user. It is also desirable for the user to be familiar with the internal function of the computer program to

be able to take full advantage of the basic potential and to avoid its inappropriate application. These desirable objectives are most easily met if the program consist of several building blocks or modules.

In this chapter, we discuss the development of a Finite Element Method program for the solution of some general class of boundary value problems. Some fundamental considerations of the code design are presented and the overall structure of the code outlined.

The details of the code are given, in form of flow charts, so that for a given application it is possible to assemble the appropriate procedures in addition to any written program to meet some special needs. Of course, such a System of Finite Element code generally requires a simple main programme whose function is to call the various required procedures in the appropriate order.

As will be viewed by the user, the finite element code consist of three major parts and controlled by a main program.

3.1.1. Program Structure

A digital computer performs three basic functions:

a.) It receives data defining the specifics of a problem;

b.) It processes the data;

c.) It produces the results of the processing.

Most of Finite element method codes can be broken down into three corresponding units:

a. PREPROCESSOR;

b. PROCESSOR;

c. POSTPROCESSOR.

The main program consists of only a few program statements which establish data blocks and call various procedures in the system. Here the main program calls the three program units: PREP, the Preprocessor, PROC, the PROCESSOR unit and POST, the Postprocessor Fig. 3.1.

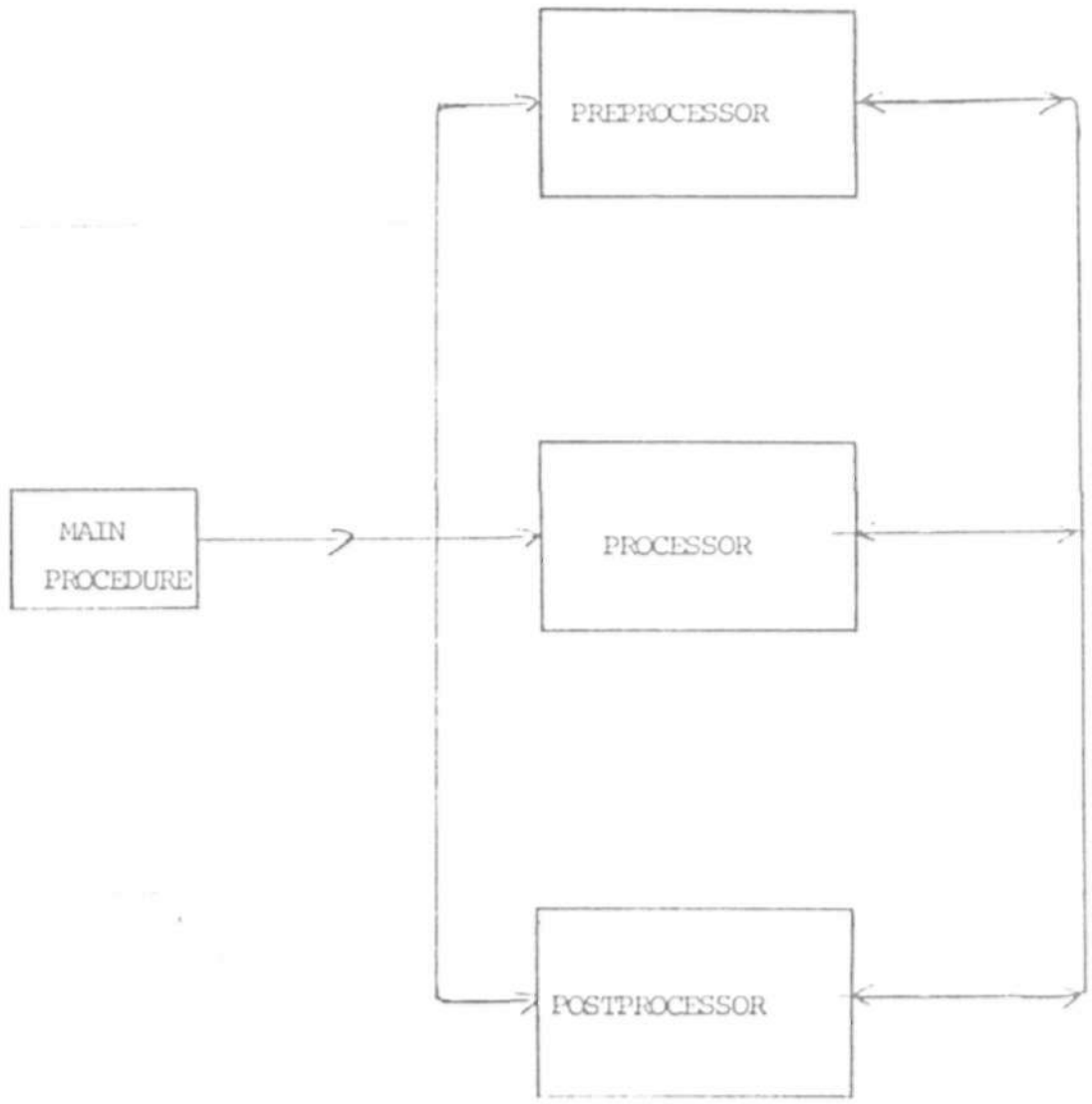


Fig 3.1.: Main Programming Call of PREP, PROC, and POST

For trends towards more flexibility, more reliable and more fully automated codes, the idea is that the program objective may not necessarily be fulfilled by merely the presentation of results of a specific series of calculation. Test can be done during postprocessing to assess automatically the quality of the solution which can signal the need for changing the parameters in the problem statement and thereby cause a return to either the preprocessor or processor.

3.2. The PREPROCESSOR.

The unit PREP (Preprocessor), Fig. 3.2., consist of various procedures that sets up the input data; generate problem data; and define problem-dependent parameter and data array. These procedures are:

(a) SCON - sets the control parameters;

(b) SMESH - sets up the mesh coordinates;

(c) SELEM - sets up the element data;

(d) SBAND - sets up maximum band width;

(e) SMAT - sets up the material property;

(f) SBC - set up boundary condition.

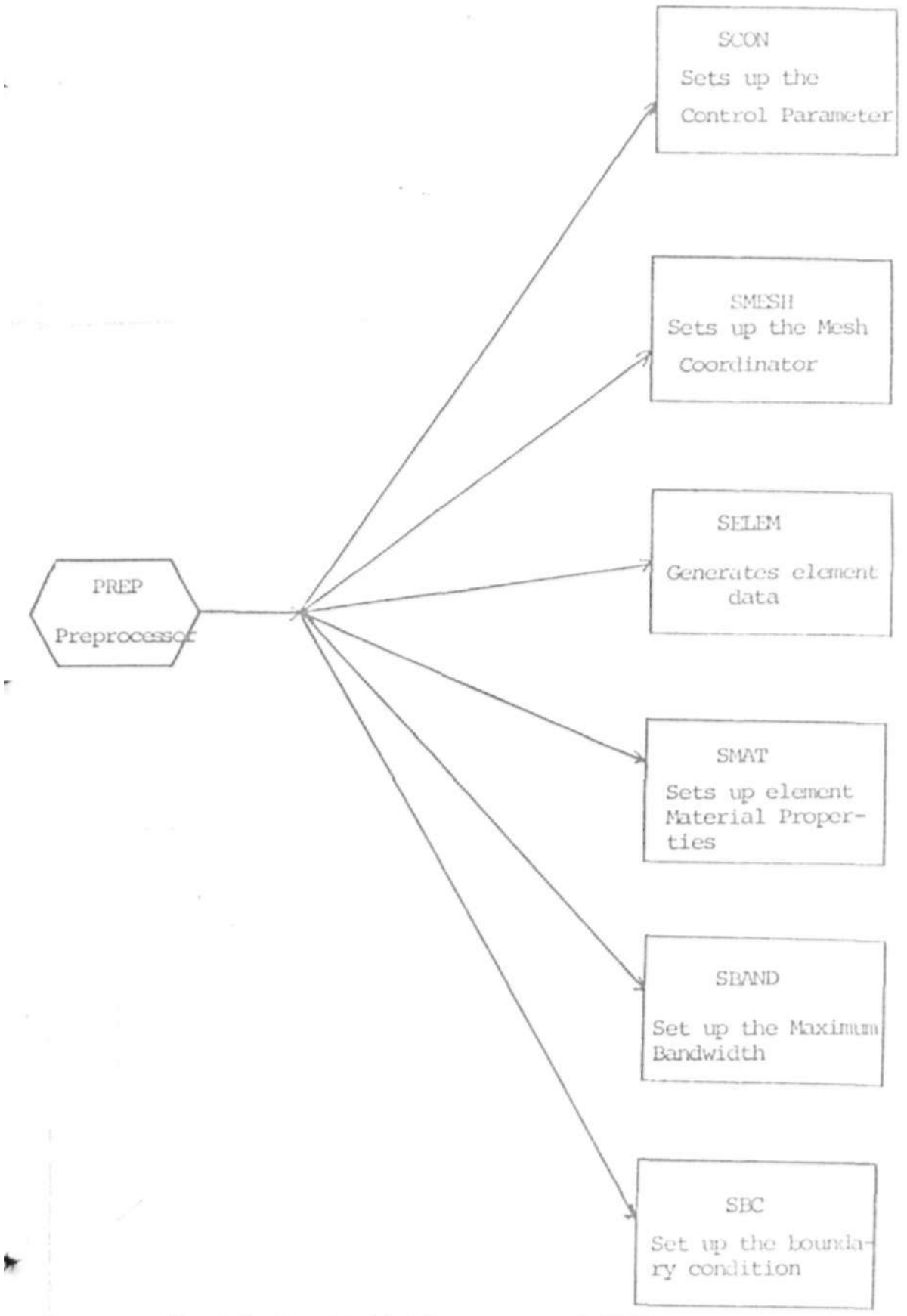


Fig 3.2: Sub Divided Procedures of PREP

The SMESH procedures which is supposed to do labelling, numbering and counting of elements and nodes together with the listing of coordinates of nodes is an extremely time-consuming and arduous task, expensive, likely to lead to data errors and delays project completion. An alternative method is automatic mesh generation Smith (1980).

3.2.1. Procedure Set-up-Control-Parameter SCON.

This procedure first reads in the number of global nodal number, the number of local nodal number, the number of elements and the number of coordinates. The algorithm to perform this operation is shown in Fig. 3.3. and the variables are :

NGNODE - Number of global variable collected by counting the number of nodes in the entire problem required;

NLNOD - Number of nodes associated with an element;

NELE - Number of elements in the entire problem region;

NCODS - The number of coordinates values in a node, one dimensional problem has one coordinate, two dimensional has two coordinates while three dimension has three coordinates;

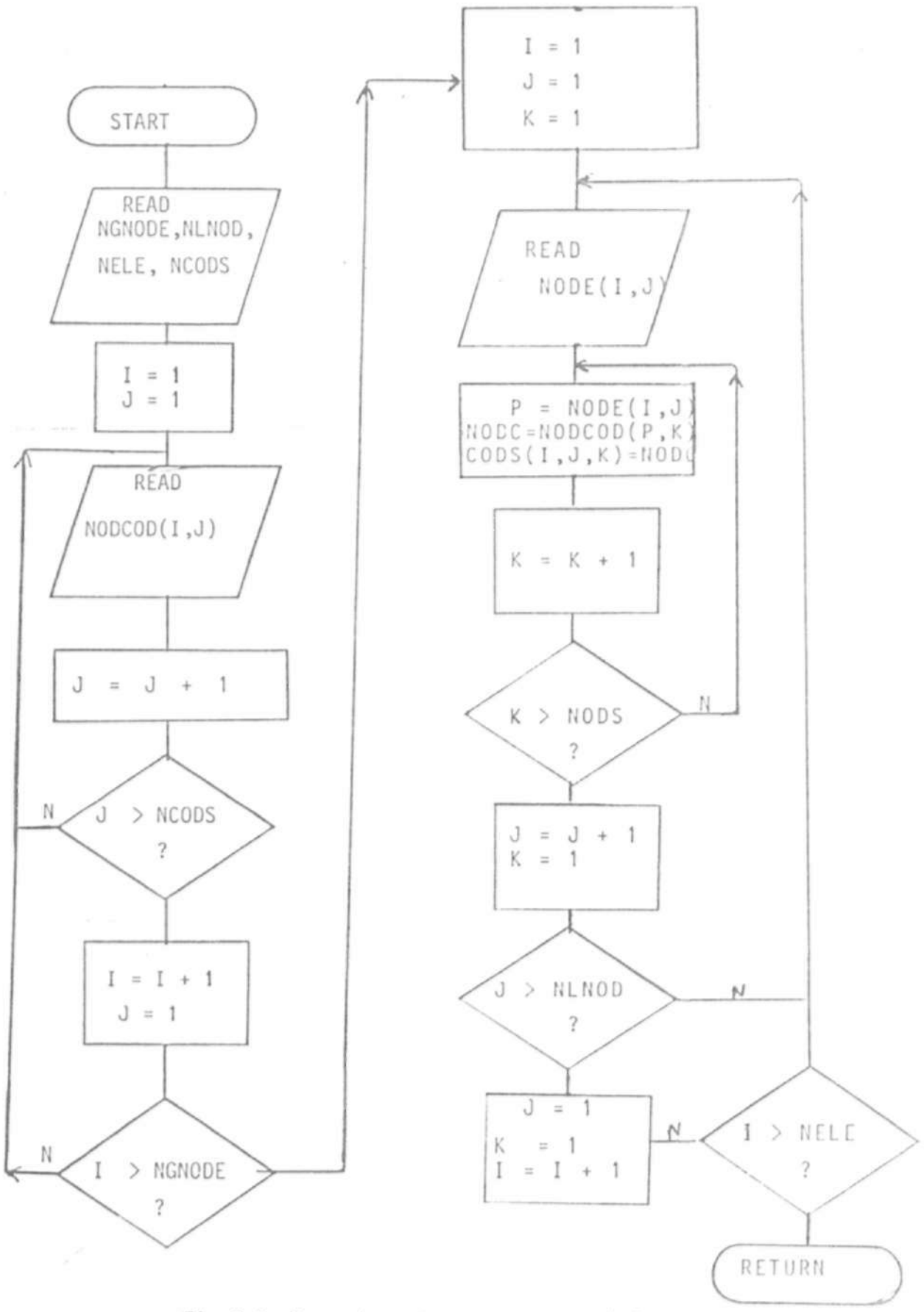


Fig 3.3: Procedure to set up control Parameters.

NMAT - The number of material properties to be read;

NDGRIVE - Number of the derivative boundary conditions;

NPBOUND - The number of node with prescribed boundary conditions.

3.2.2. Setting up the Mesh Coordinate, SMESH.

This procedure reads a data that define the x, y and/or z coordinate of all the nodal points of a problem, depending on the value of NCODS in procedure SCON. This procedure is still a part of algorithm in fig. 3.3. The variable that keeps account of these coordinates is:

NODECOD(I,J) - The Ith node of the Jth coordinates

$1 \leq I \leq$ NGNODE and $1 \leq J \leq$ NCODS.

3.2.3. Procedure to Set up the Element Data, SELE.

Based on the data gathered from SMESH the elements can now be idealized with respect to the subsequent input variable. These variables are:

NODE(I,J) - the Jth node of Ith element

1<J<NLNOD and 1<I<NELE;

CODS(I,J,K) - the Kth coordinates of the Jth node of Ith element, such that 1<K<NCODS.

The nodes that compose an element are read in a set of NLNOD times, and the associated nodal coordinates are allocated to CODS from previously read NODDECOD, based on the presently read node - NODE. The later part of Fig. 3.3 shows this procedure.

3.2.4 Procedure to Read the Boundary Conditions SBC

The function of the procedure is to read in the prescribed boundary conditions together with the node on which it is prescribed based on the number of the prescribed boundary conditions which was read by procedure SCON. The element number of the derivative boundary condition, along with the side of the element that contain it, are read. The algorithm is shown in Fig. 3.4 with the variables:

GBNDNO(I) - is the node number that has a prescribed boundary value;

PBOUND(I) - is the prescribed boundary value for the GBNDNO(I) node 1<I<NPBOUN;

DELENO(I) - is the element number that contains the derivative boundary value;

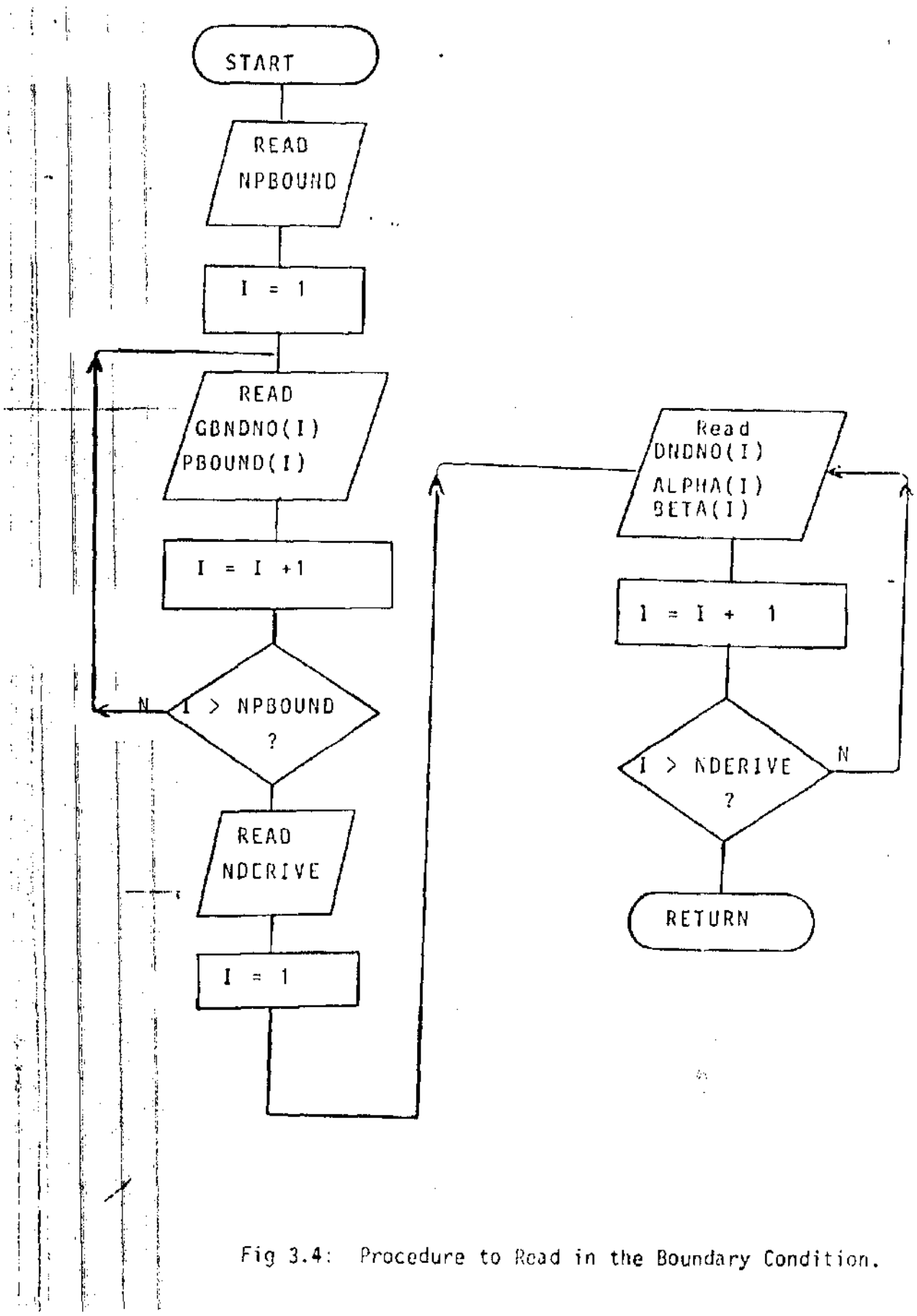


Fig 3.4: Procedure to Read in the Boundary Condition.

NSIDE(I) - is the side of DELENO(I) element that contain the boundary value;

ALPHA(I) and BETA(I) - are the value of α and β that are to be considered on the 1<I<NDRIVE.

3.2.5 Procedure to Read the Element Material Properties-SMAT

The element material property is read by this procedure. In many cases this properties is the same for all the elements. In the cases where they vary, they usually vary with respect to their geometric position. These are calculated subsequently as shown in Fig. 3.5. The variables in the algorithm are:

KX(I)
KY(I) the coefficient of the equation for the
P(I) Ith element.
Q(I)

Because these coefficients vary from problem to problem it is difficult to write a special subroutine that will compute the various values when one of them is not a constant. The one- and two-dimensional problems can be read, based on the inputs made in SCON.

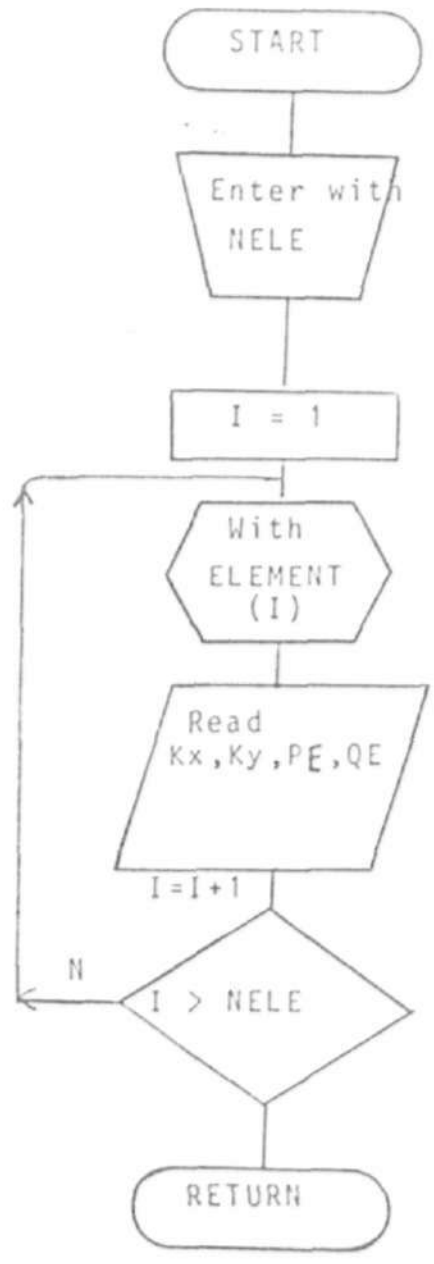


Fig 3.5: Procedure to Read Element Material Properties.

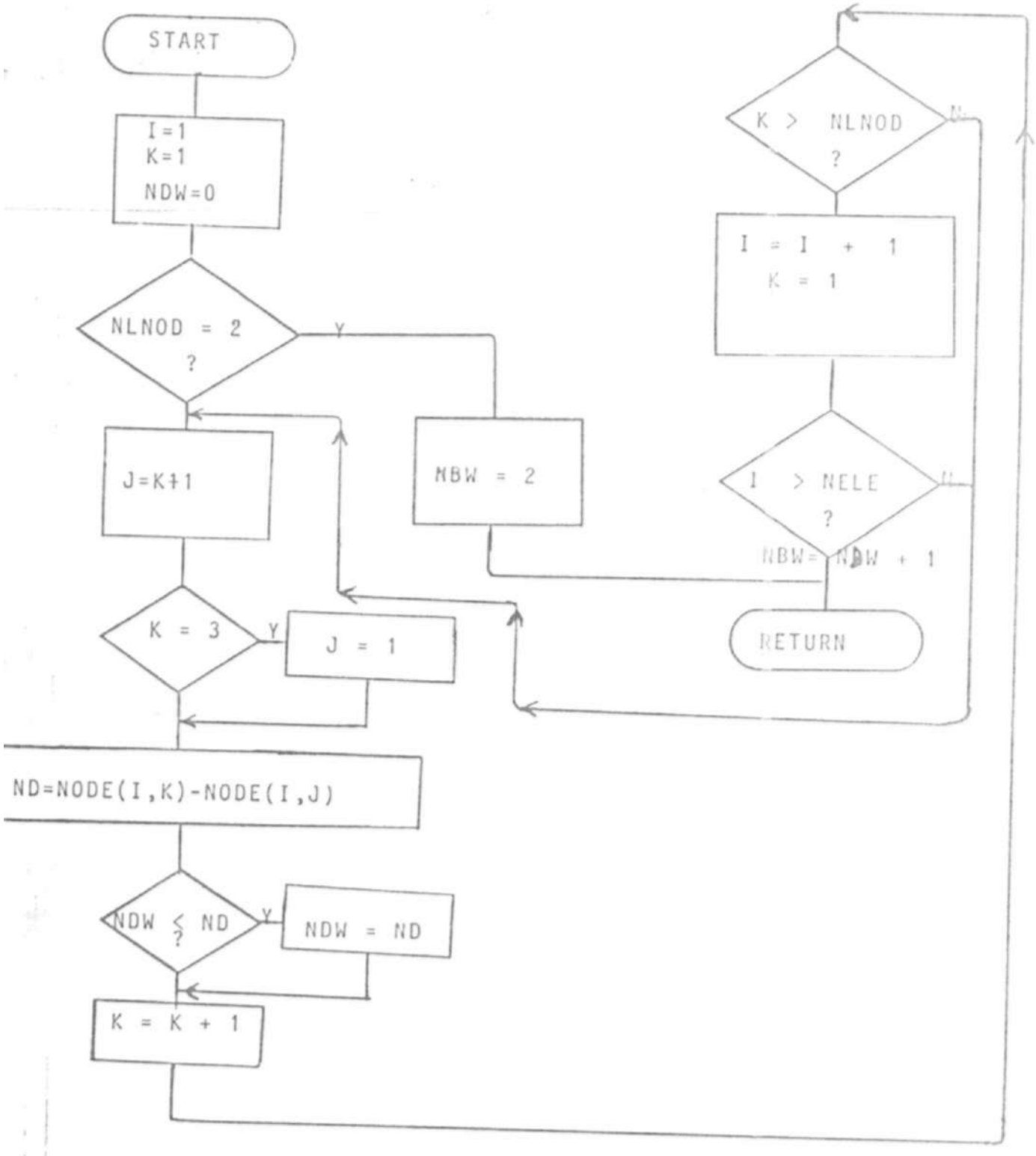


Fig 3.5b: Continuation of Fig 3.5

3.2.6 Procedure to Compute the Band width of the problem SBAND.

The labeling of the nodes (assigning a number) would be a trivial operation if the node number did not influence the computational efficiency associated with obtaining a solution. A reduction in the bandwidth produces a reduction in the required computer memory space plus a reduction in the computation time. Algorithm SBAND, Fig. 3.5b, goes through the whole number in pairs to find the maximum bandwidth, by using:

NODE(I,J) - the Jth node of the Ith element;

ND - a temporary variable to hold a particular difference in node numbers;

NBW - the maximum bandwith.

3.3. The PROCESSOR.

The Processor, Fig. 3.6, is the most important unit of the entire package. Here, we typically evaluate the element shape functions; calculate the element stiffness matrix, and load vector; assemble the global matrices; apply the

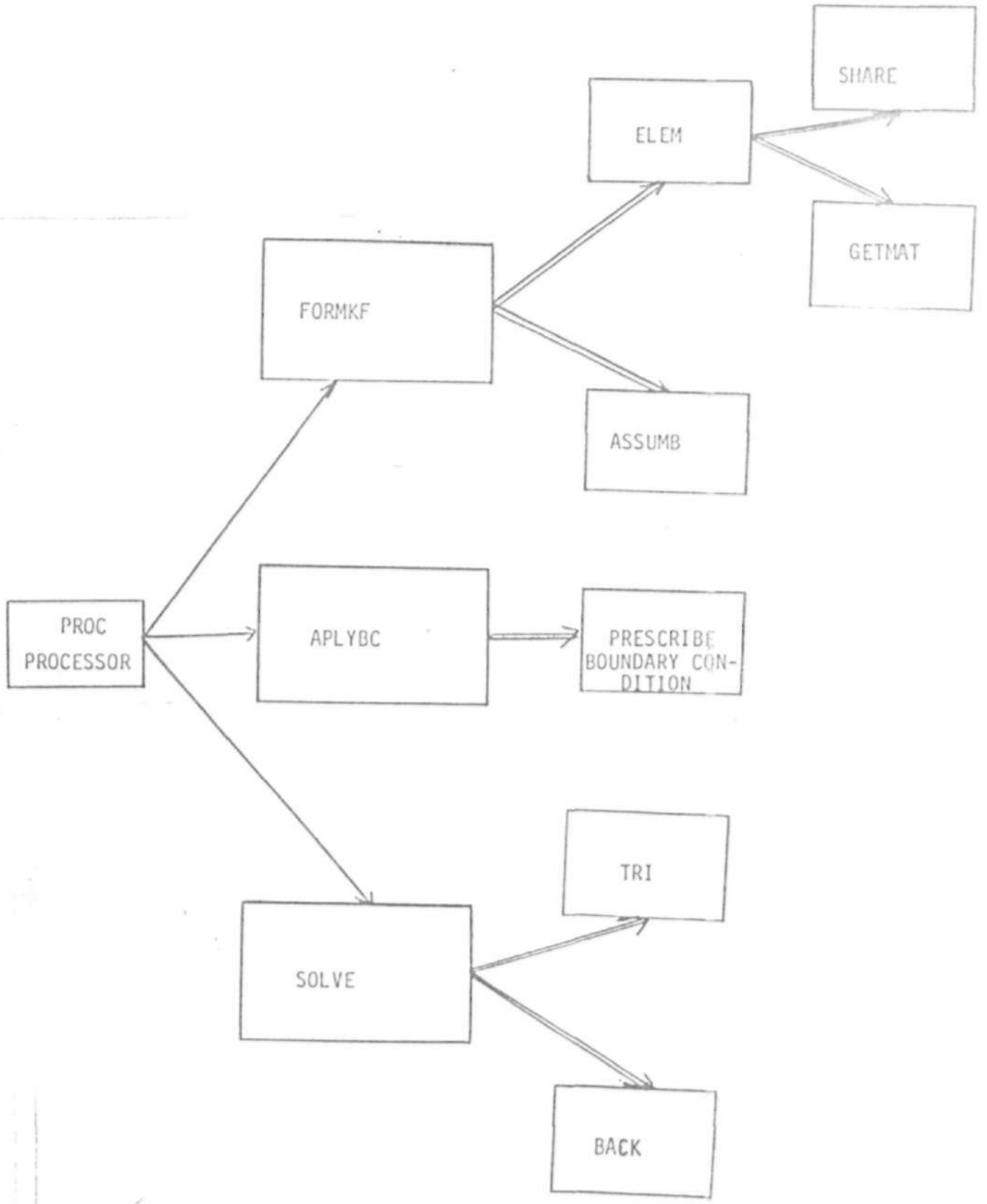


Fig 3.6: Sub Divided Procedures of PROC.

boundary conditions; and solve the resulting system of algebraic or differential equations. The routine PROC (Processor), is broken down into three major subroutines which in turn has subsequent subroutine. The major subroutines are:

a) *FORMKF* - *formulation of the stiffness matrix* and load vector;

b) APLYBC - applying the boundary conditions to the global stiffness matrix and load vector;

c) SOLVE - solves the linear equations for the nodal point value.

The FORMKF procedure will call four other subroutines:

* to calculate the material properties of the region under consideration by using subroutine SMAT;

* to calculate the shape function by using subroutine SHAPE;

* to calculate the element stiffness matrix and load vector by using subroutine ELEM; and

* to assemble the element stiffness matrix and load vector by using subroutine ASSMB.

The SOLVE procedure consists of two other routines that will:

* performs the gaussian elimination by using subroutine TRI; and

* perform the back substitution by using subroutine BACK.

3.3.1. The Material property Routine - GETMAT.

This routine calculates the material properties (coefficients in the differential equation) at a given point in a particular material. The value K_x, K_y, P, or Q, are returned to the calling program ELEM based on the location of the property.

The routine operates in conjunction with SMAT which reads and stores the data used for the calculation.

3.3.2. Routine to calculate the Shape function - SHAPE.

This routine is called principally by the procedure ELEM, to return some coefficients of the shape function together with the length, area or volume of the particular element under consideration, depending on the dimension of the element. Fig. 3.7. is the algorithm for this routine, and variables in the routine are:

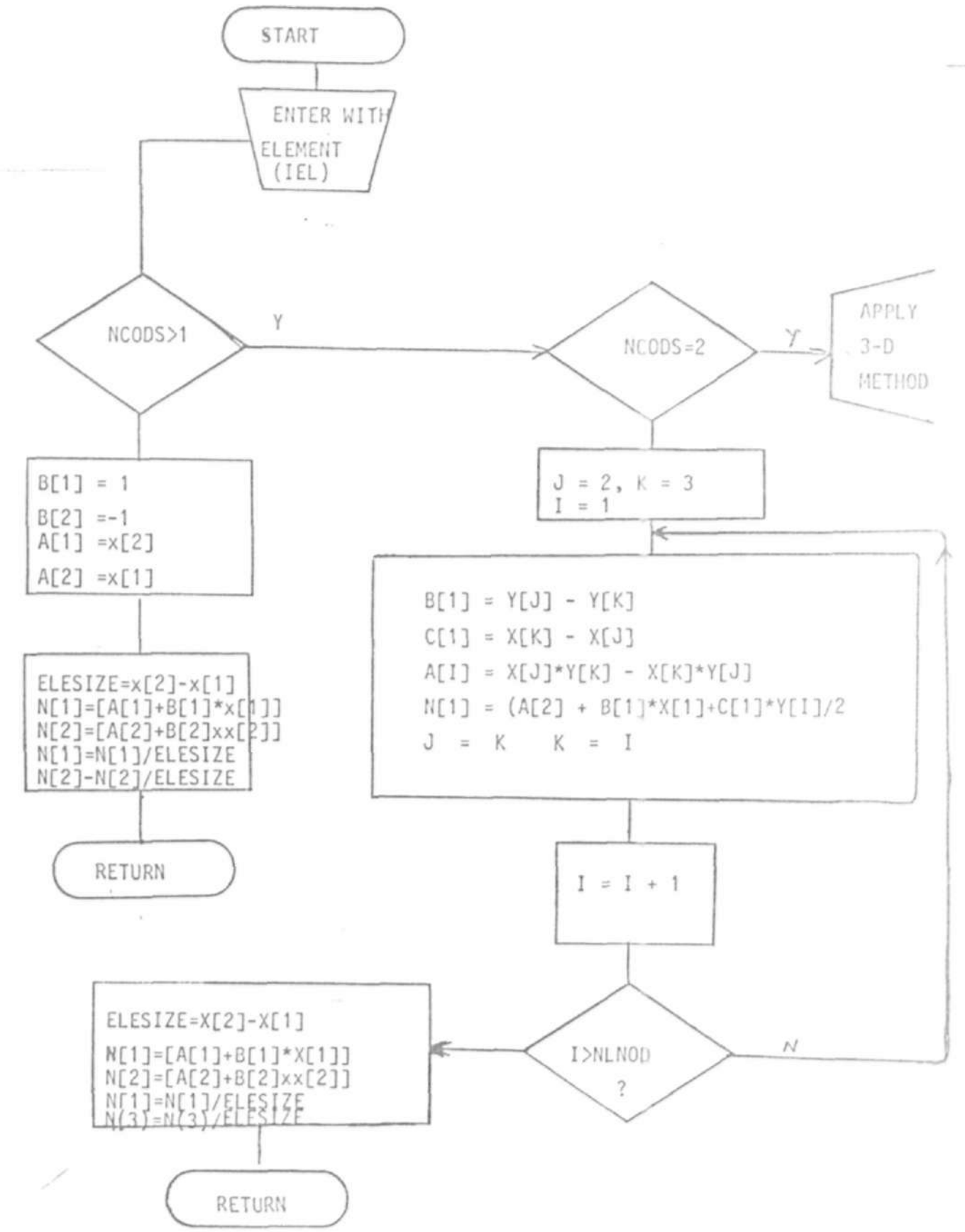


Fig 3.7: Procedure to Calculate the Shape Function.

ELESIZE - return either the length, area or volume depending on the dimension of the problem.

A(I),B(I),C(I)- returns the coefficient of the shape function at the Ith node for $1 \leq I \leq$ NLNOD.

N(I) - Returns the value of the shape function at Ith node number.

3.3.3. Element routine - ELEMT.

The main function of this routine is to calculate the values of the element stiffness matrix and load vector. This calculation will involve the calling of the routine SHAPE and GETMAT to pass the necessary parameters to be used in the calculation. The operation is performed for a particular element ILE as called by the calling routine FORMKF.

The algorithm for ELEMT is shown in Fig. 3.8. and variable in the algorithm are as follows:

ELEM(I,J) - is the value of the Jth column of the Ith row of the element matrix equation;

ELEV(I) - is the Ith value of the element vector equation. $1 \leq I,J \leq$ NLNOD.

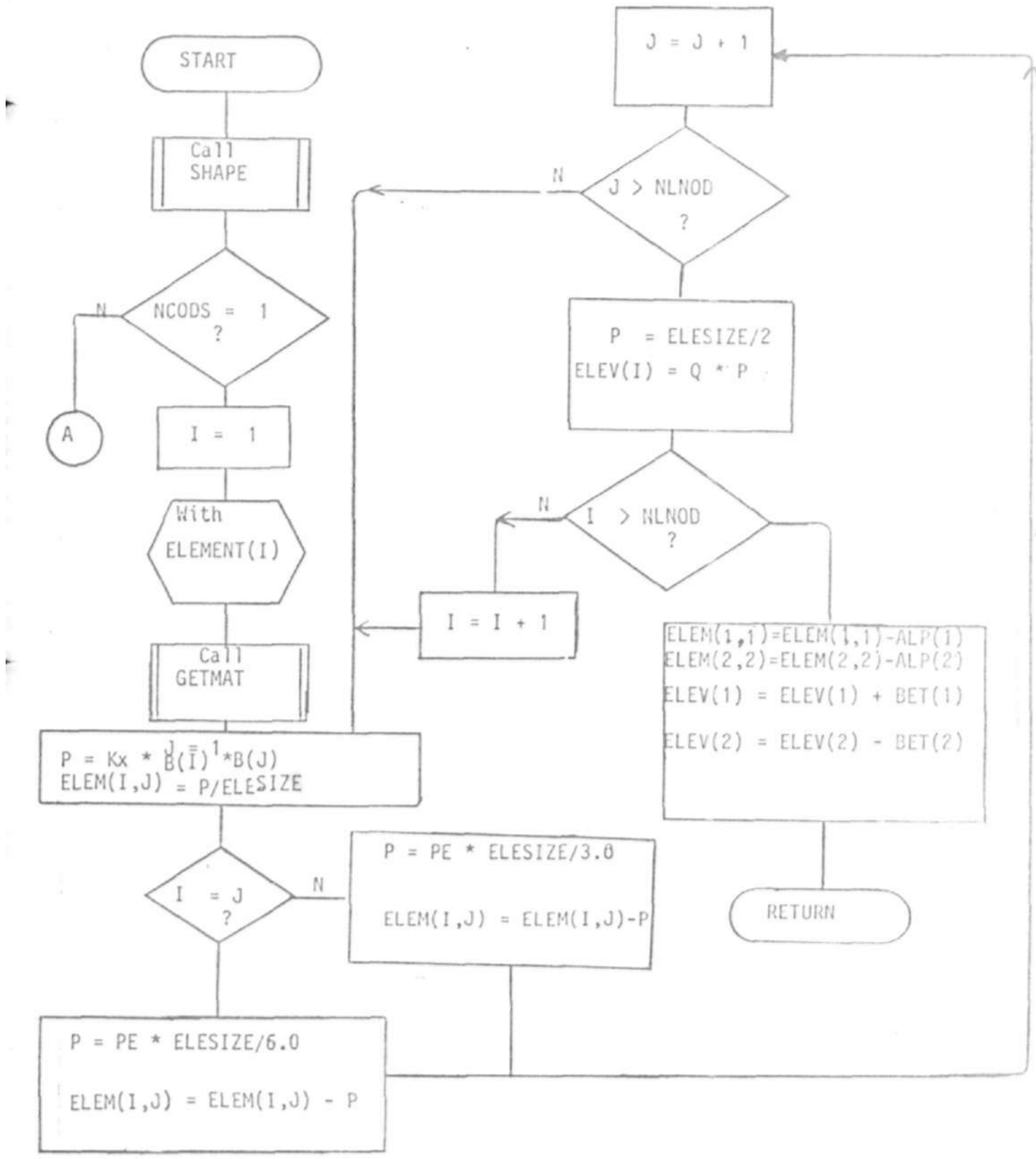


Fig 3.8: Procedure that Calculate the Element Stiffness Matrix and Load Vector.

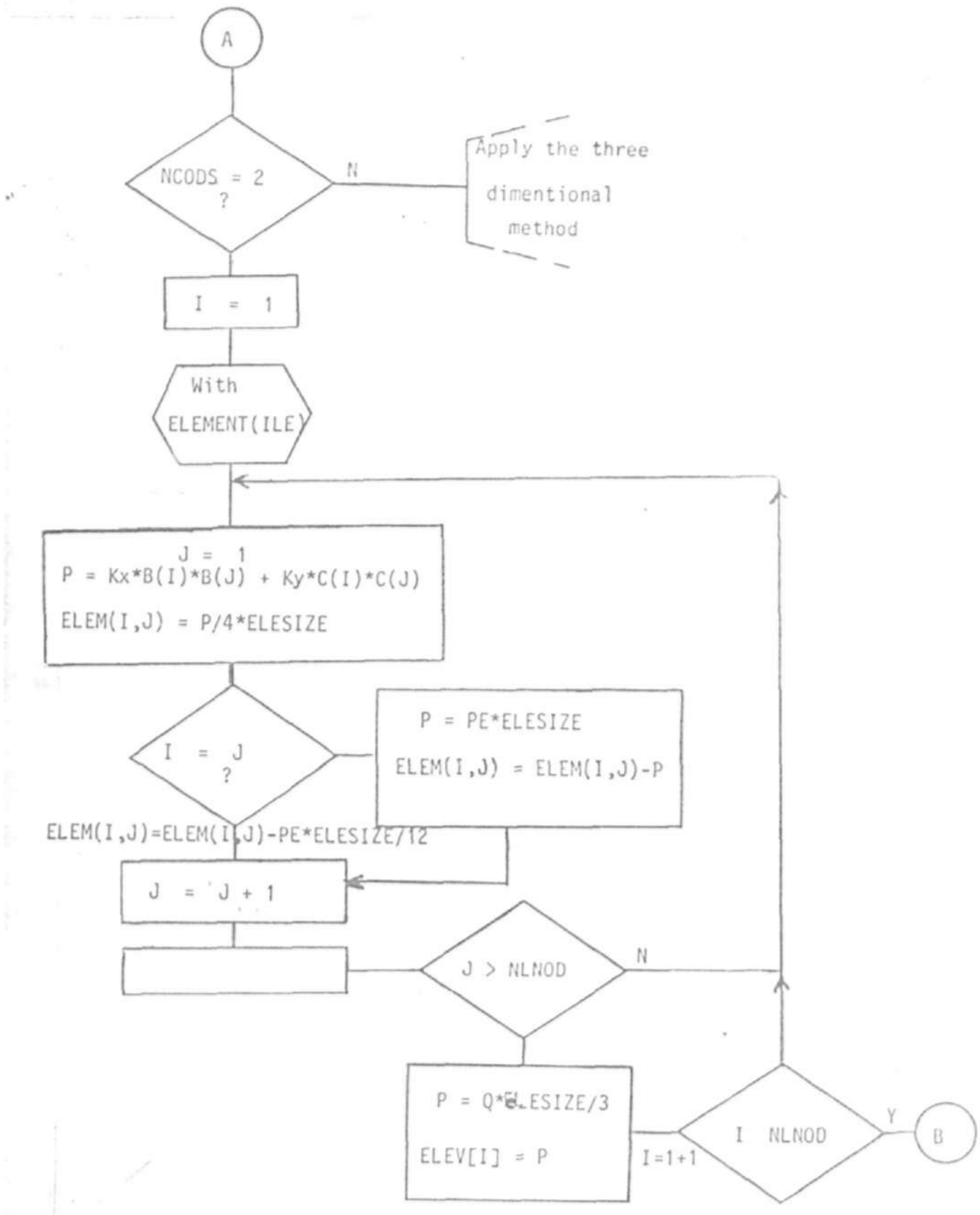


Fig 3.8b: Continuation of Fig 3.8.

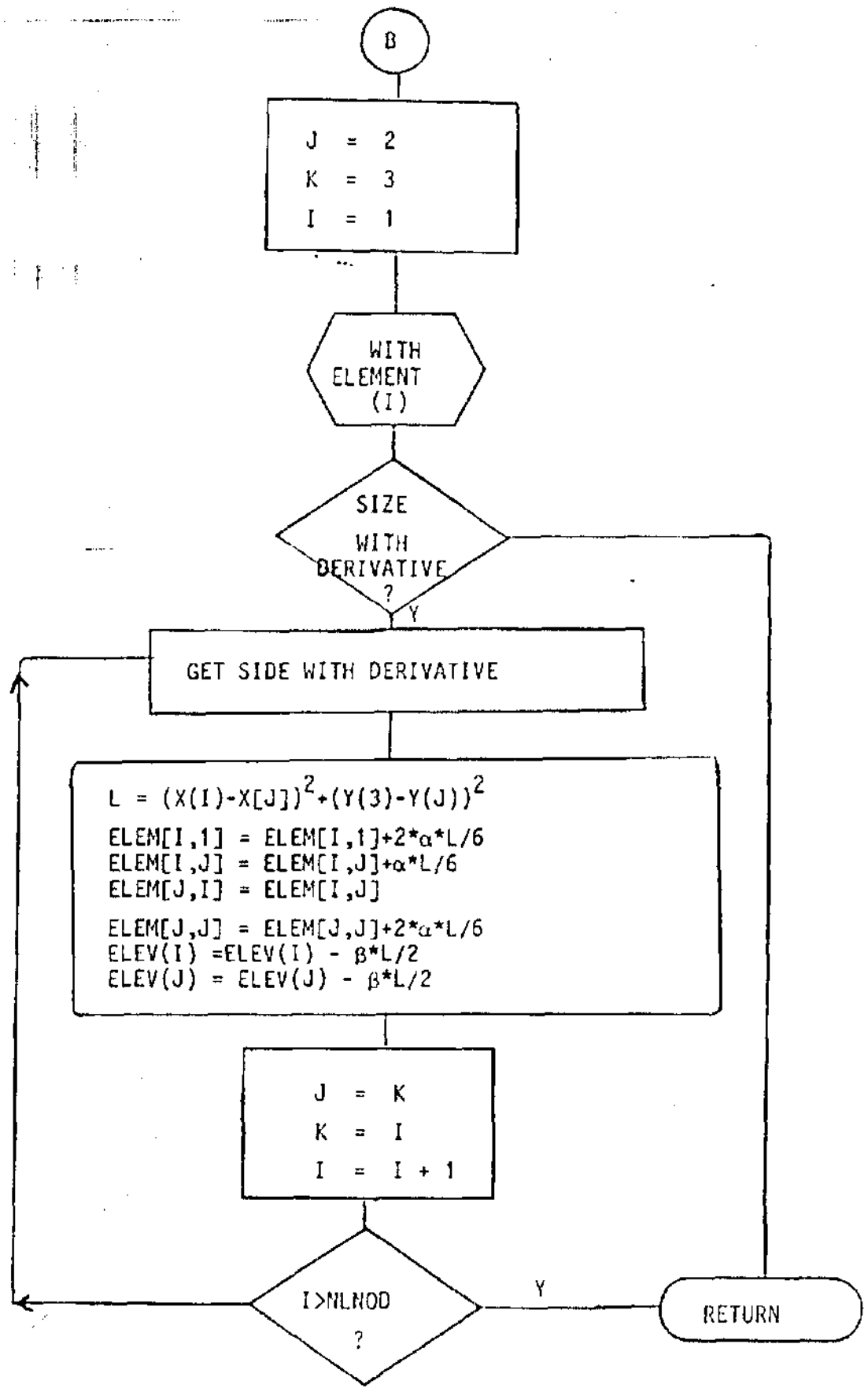


Fig 3.8c: Continuation of Fig 3.8b.

Other variables passed as parameters from SHAPE are: B(I), ELESIZE, X(I), and Y(I). Those passed by GETMAT are: K_x, K_y, P, Q, α and β.

The algorithm calculates the element matrix and vector based on the dimension of the problem- one- two- or three-dimensional problem.

3.3.4. Assembly routine - ASSEMB.

The last subroutine to be called by the routine FORMKF is the routine ASSEMB. It adds up the individual element's contribution into the global matrix and global load vector. Before the assemble is done, the global arrays are set to zero. After ASSEMB had been called for each element, the global arrays are complete. The assemblage in this routine is made in such a way that the NxN matrix is not used, rather, the matrix with dimension (NBW x N) is used, where N is the number of global node NGNOD.

The routine is shown in Fig. 3.9. with variables in the equation as follows:

ROW	- pointer to the row under consideration;
COL	- pointer to the column under consideration;
GLOBM(I,J)	- the value of the Jth column of the Ith row of the global matrix;
GLOBV(I)	- the Ith entry of the global vector.

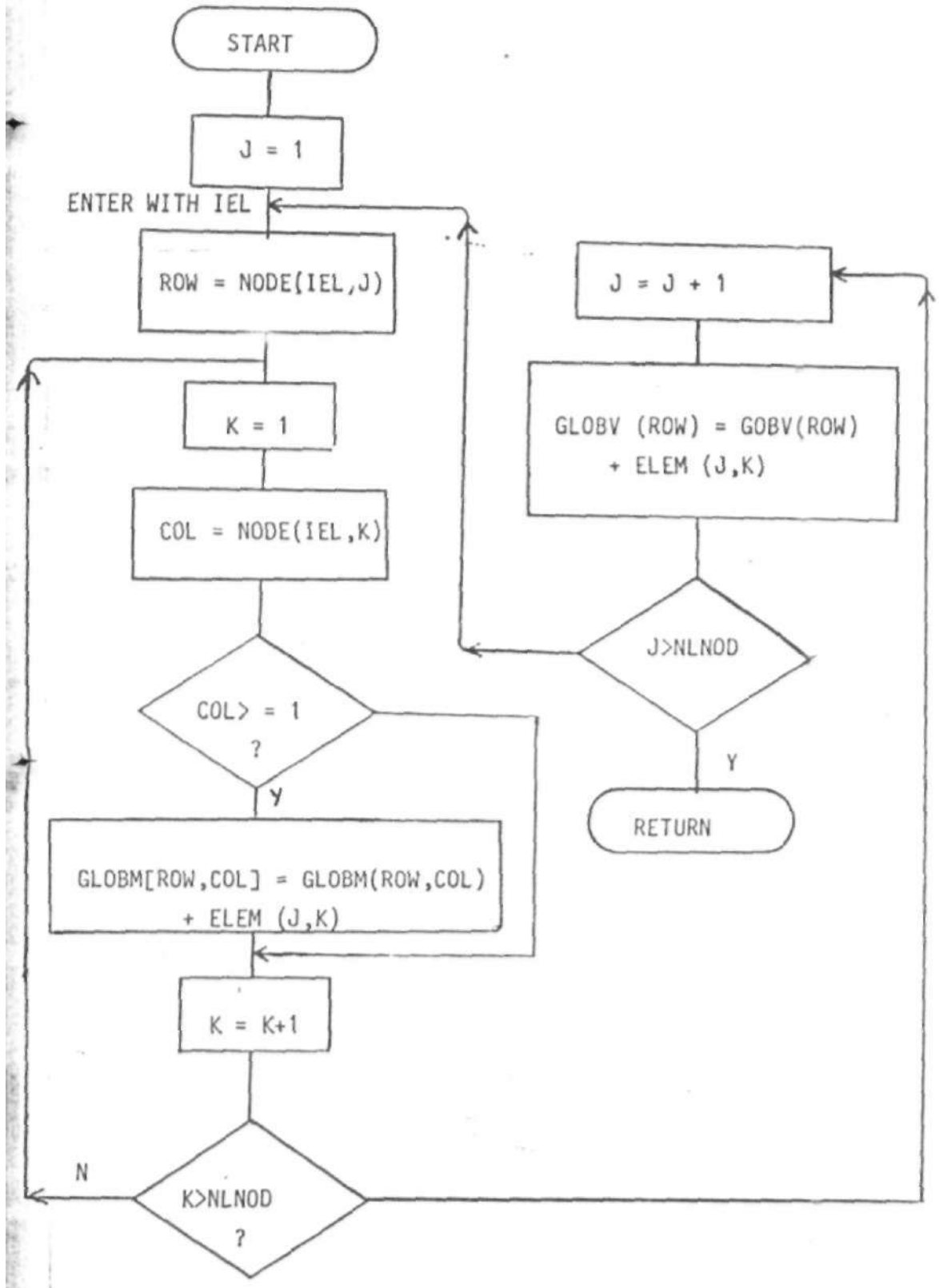


Fig 3.9: Procedure that does the Assemblage of the Stiffness Matrix and Load Vector.

3.3.5. Boundary conditions routine - APLYBC.

The routine APLYBC makes the modifications to the global matrix and load vectors required to impose the boundary condition at the boundary of the domain.

The required variables are:

GBNDNO(I) - the global node number on which the boundary conditions are prescribed, where I is for a particular node $1 \leq I \leq$ NPBOUND;

NBW - the maximum bandwith;

PBOUND[I] - the value of the prescribed boundary condition at the ith node 1<I< NPBOUND.

The algorithm for the application of boundary condition is shown in Fig. 3.10. The row and column on which there are prescribed boundary conditions are computed and the modification are made on GLOBV while the corresponding GLOBM are set to zero. Other elements of the Global matrix are reorganized to reflect the modification due to the prescribed boundary conditions.

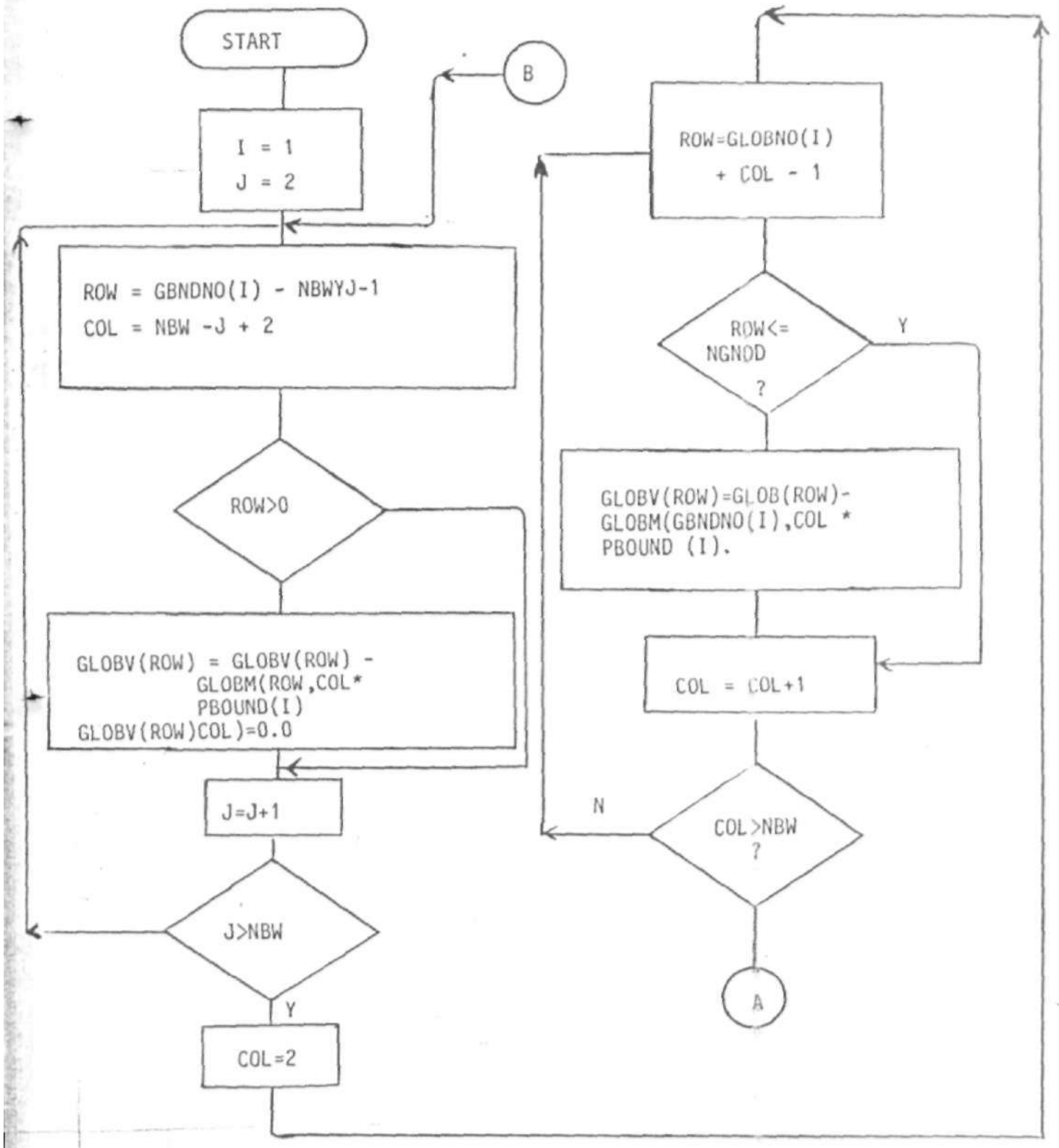


Fig 3.10: Procedure for Applying the Prescribed Boundary Conditions.

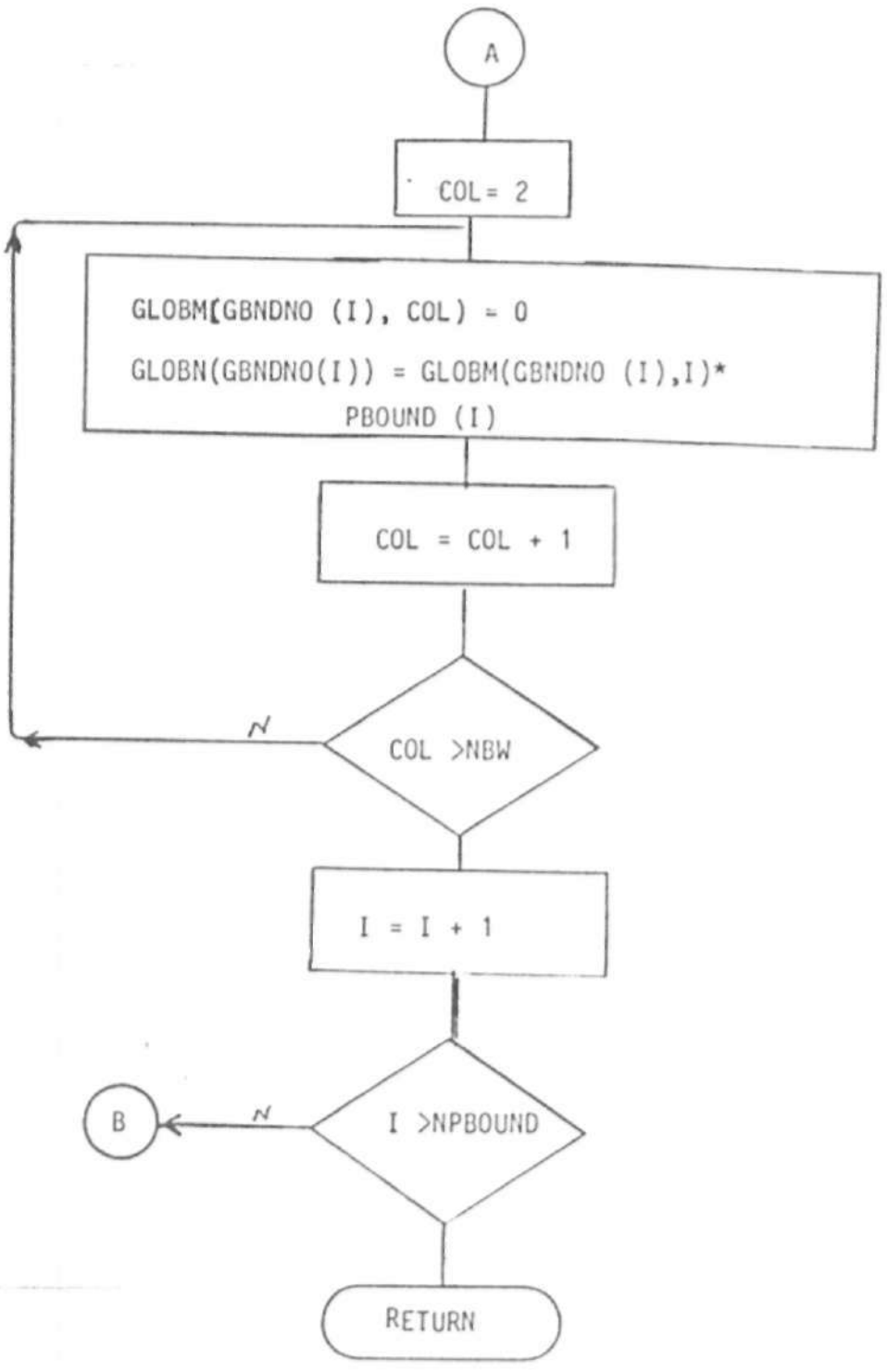


Fig 3.10b: Continuation

3.3.6 Reduction of the Stiffness Matrix to Triangular Form: TRI.

This routine TRI performs standard Gaussian elimination (row operations) on the Global stiffness matrix. No pivoting (searching for the large entries to divide by) is done. The elimination is not done on the entire matrix, rather it is done on the transformed global matrix (skypole matrix). The variables in TRI are all variables declared in ASSEMB. The algorithm is shown in Fig. 3.11.

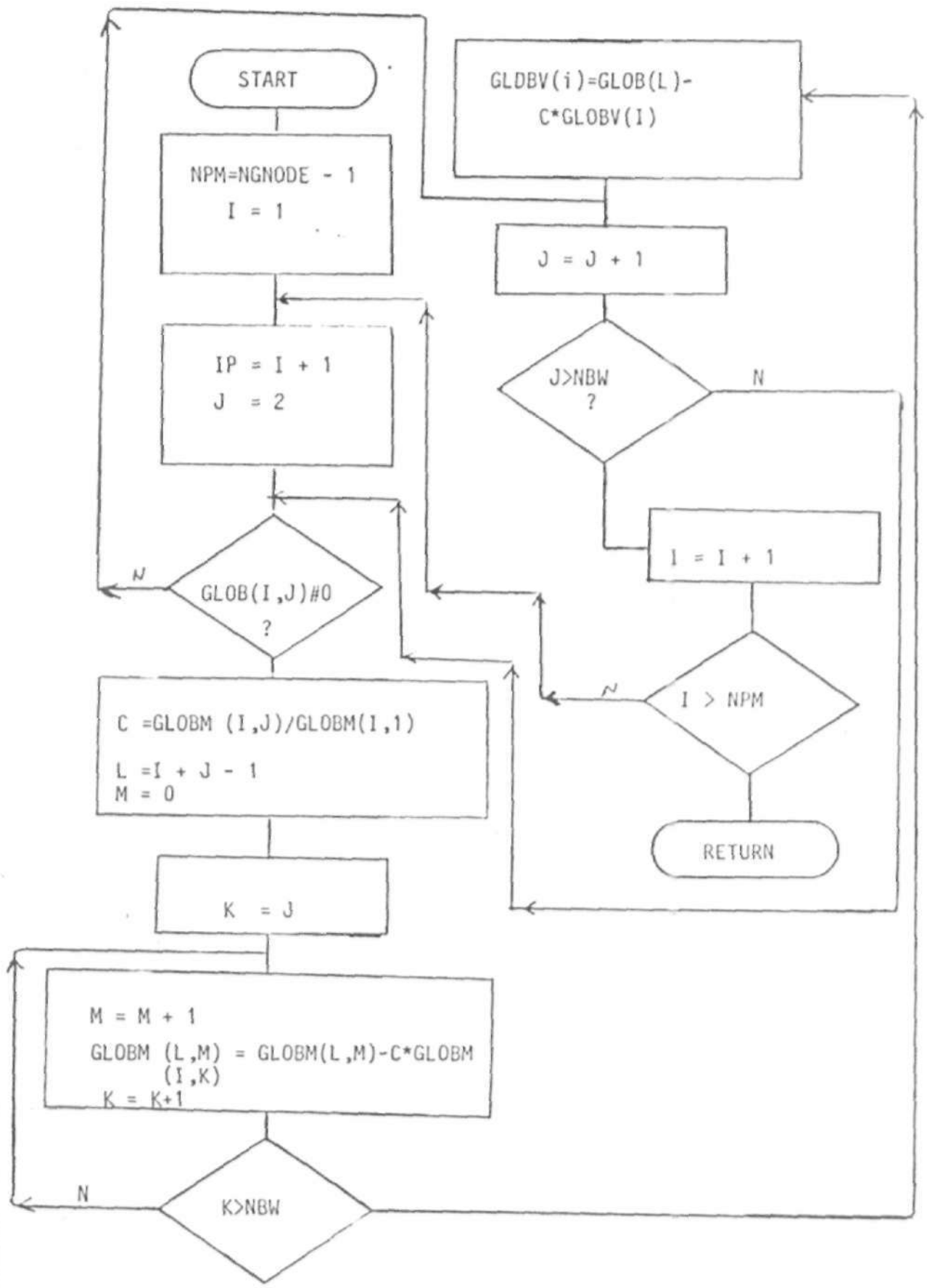


Fig 3.11: Procedure to reduce the Stiffness Matrix to Triangular form - TRI.

3.3.7. Back Substitution Routine - BACK

Routine BACK, does the backwards substitution on the right-hand-side vector, repeatedly to calculate the unknown. This routine requires addition vector of the same capacity as the Global load vector. This is to avoid the destruction of the values of the load vector which are first transferred into a temporary vector TEMPV. The unknown will finally be in the temporary vector TEMPV. Other variables remain as already defined in the ASSEMB, APLYBC and TRI. The complete algorithm is defined by Fig. 3.12.

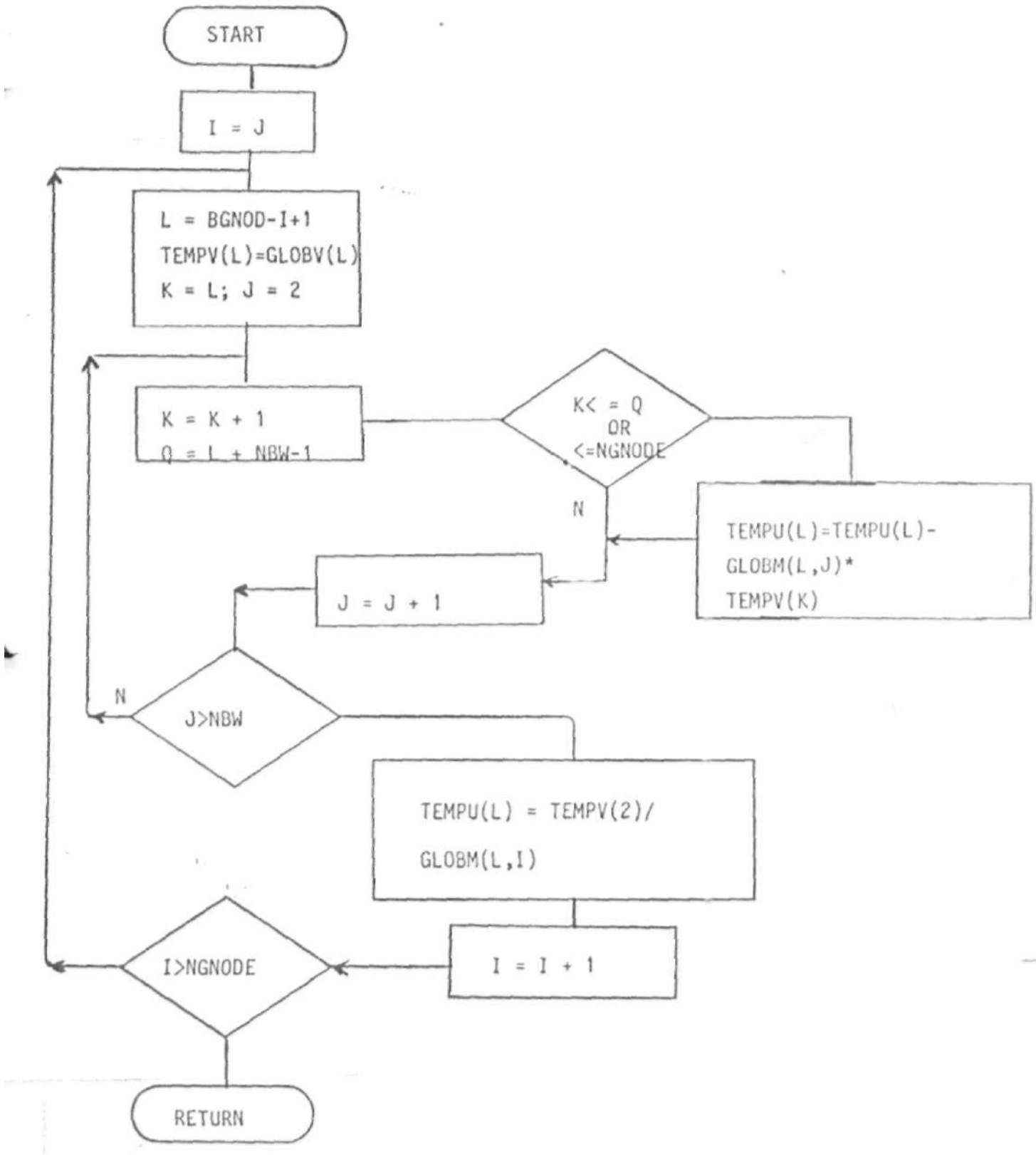


Fig 3.12: Procedure to do the Back Substitution for the Nodal Solution.

3.4. The POSTPROCESSOR -POST.

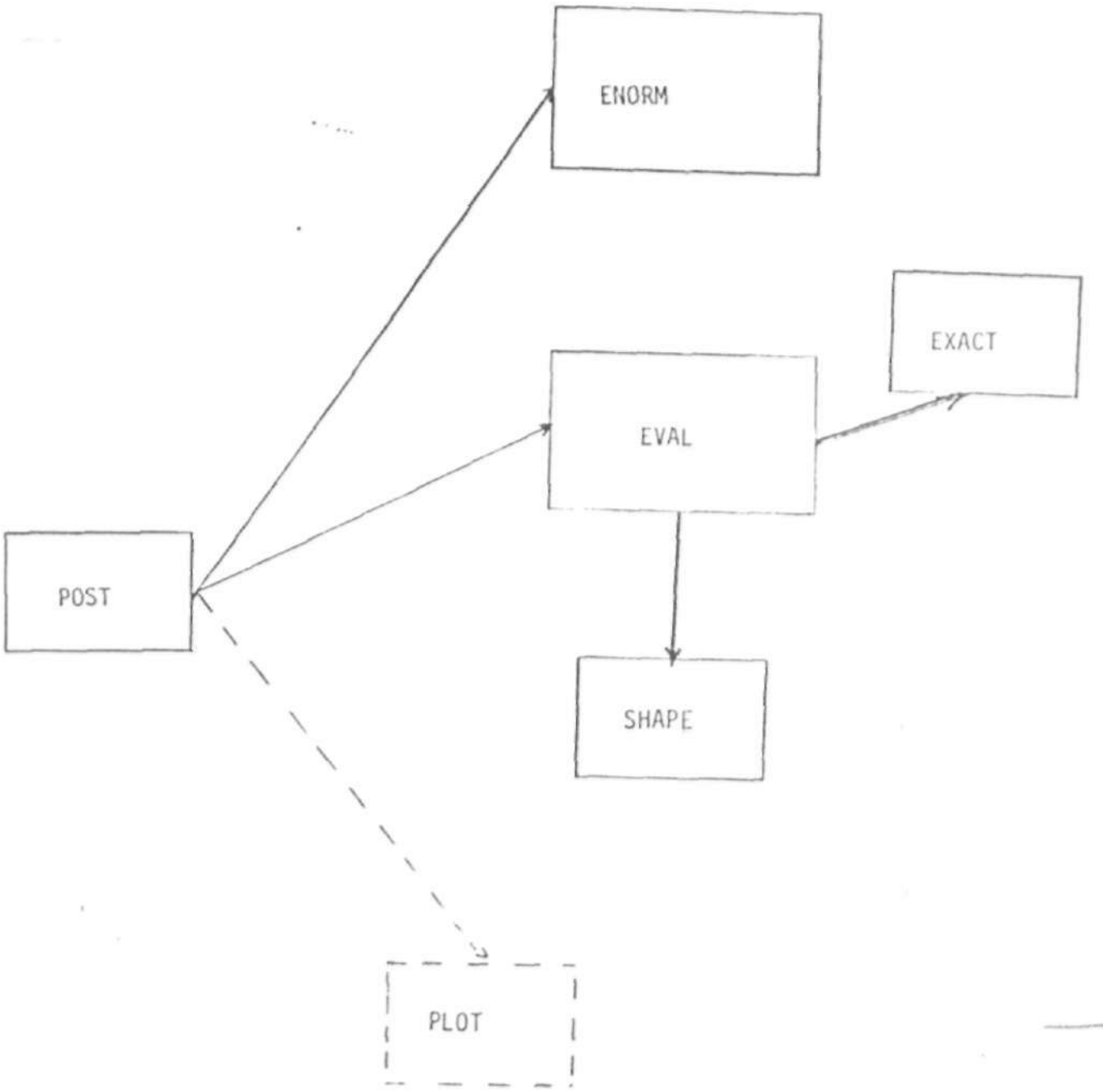
The nature of Finite Element Method always result to voluminous output. This is associated with the printing of the nodal point value, computing and printing the derivatives. In the case of problems with exact equation, errors can be printed. Other results can be transformed in a suitable form for tabulation or plotting.

The procedure will complete it processes by calling three other routines, Fig. 3.13.

(i) EVAL - evaluate and prints finite element and exact solutions.

(ii) ENORM - evaluate and prints the error if the exact solution exists.

(iii) EXACT - calculate the exact solutions and conditions.


ENORM
EXACT
EVAL
POST
SHAPE
PLOT

3.4.1. The Finite Element and Exact Solution Routine - EVAL.

This routine calculates the values of the finite element solutions and derivatives at a given point in a given element. The element number IEL and element coordinates at which the solution is to be calculated are transmitted to EVAL through the calling sequence. A routine, which can be coded for particular problem will be called as routine EXACT to obtain the exact solutions at the same element points. Algorithm for the routine is shown in Fig. 3.14., Variable required by this procedure are:

X,Y - the coordinate of the point at which the unknown value is to be computed. X alone, for one-dimension, problem and (X,Y) for two dimensional problem.

T - the unknown parameter to be computed at either X or (X,Y) coordinate point;

DTx, DTy - the derivatives with respect to X and Y to be computed at the point X, or (X,Y) as the case may be.

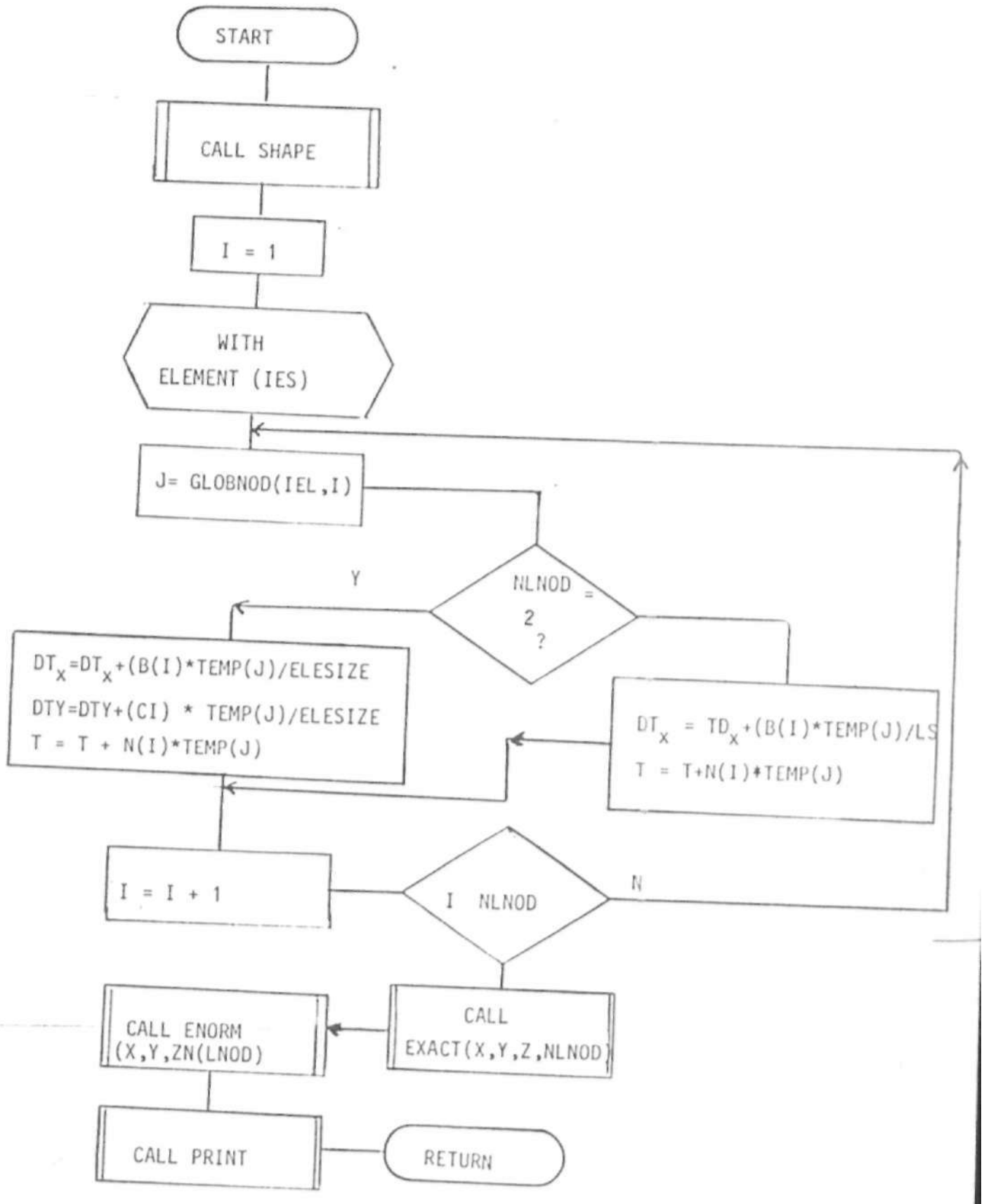


Fig 3.14: The Element and Exact Solution Routine.

3.4.2. Calculate Exact Solution Routine - EXACT

This routine is subject to modification as problems with exact solution vary for a given value of X and Y, as transmitted to EXACT through the calling sequence, the exact value of the solution T, and the derivatives, DTx and DTy, are calculated.

3.5. Programming Environment

The programming environment was designed such that it will reflect the major objectives of the project. To this end we have considered the use of one and two dimension problem and provisions made for enhancement to three dimensional problem. Type of element considered are simplex linear and simplex triangular elements.

The type of problem can be known from the number of co-ordinates NCODS as read by procedure SCON. As simplex element are used, the number of nodes NLNOD will be two for linear element and three for triangular element.

Discritization of a problem region can be done automatically or manually. In this procedure, it is assumed that the problem region had been discritized and the nodal coordinate read from a file.

Automatic mesh generation programme can be called in place of reading from a file. With the mesh generation procedure there can be mesh refinement routine, Carey(1980), this will reduce the maximum band width NBW in routine SBAND of Fig. 3.5b. With the use of automatic element generation Smith (1976), the process of reading the global nodal points and their conditions first, and later reading local nodes of an element and then associating them with their various coordinate as in SMESH and SELE respectively, will be removed.

With the prescribed boundary condition for one-dimension element problem it will be either on one end or on both ends. The same is applicable if it were derivatives boundary conditions.

For two dimensional element (in this case triangular element), the value of the prescribed boundary condition are entered for specified elements together with their respective global node number. The same method was adapted for the derivative boundary condition. This brought about the reduction of storage wasted for unprescribed nodes as well as nodes without derivative boundary condition. Moreover the computational time for the entire elements had been reduced from $O(N^2)$ to $O(K^2)$, where $k<<N$, K is the number of either prescribed boundary conditions or derivative boundary conditions while N is the number of global nodes. This can

be seen in Fig. 3.4., where NSIDE, ALPHA, BETA are array of very long dimension which will not be more than NDRIVE for any run.

With the problem of bulkiness of Finite element software and its implementation on PC's in mind, we came up with a finite element solver, which will be more of a tool for solving some standard routines of finite element methods on sheets.

To this end, the finite element solver can be a modelling package comprising of sheets and subsheets. Such that the use of these sheets and subsheets have:

(i) Trimmed off the excess and redundant use of array;

(ii) Centralized the element/nodal data;

(iii) Reduced the number of programs statement to handle;

(iv) The ability to handle interpolating function internally;

Other areas that we considered very useful in the achievement of the objective of this project were with regards to:

(i) avoidance of superfluous items in a program (Economy);

(ii) breaking up lengthy or complex passages into subprograms (Modularity);

(iii) Writing and using simple or straight forward method (simplicity);

(iv) Writing an orderly or well organized program (structure);

(v) Efficient use of comments (documentation)

(vi) Efficient use of space (layout).

These objectives are principally: clear understanding, reduction of storage consumption and reduction of time taken by finite element procedure. The interactive facilities calls for automatic modification of data, both internally and on the screen.

CHAPTER 4

THE PROGRAMME IMPLEMENTATION

4.1 The Software System.

The design of Finite Element Method programs and packages had undergone a sequence of transformation in the quest of improving the performance of the application of the Method. During the evolution of these packages some essential features like size (large number of variables), flexibility (choice of operation), integration, algorithms, modularity, interactiveness, diagnostic, robustness, and trackrecord maintenance were considered. Some desirable features were also considered. These include variable step-size library and input/output routines, translator, and run-time efficiency, documentation, service bureau availability and maintenance cost. In the same manner, methods of achieving essential and desirable feature were sought by improving on the software.

With a brief survey of numerical software, it can be seen that these are in four basic forms.

(a) Individual Programs: (perhaps with a few subprograms) are the most common. They are designed to carry out special computation eg. ACM algorithm, the Computer Journal Algorithms.

(b) Software Package: is a set of programs for a practical problem area and is usually narrowly focused. Like packages for linear equation, Initial boundary value problem in Ordinary differential equation and elementary functions.

(c) Numerical Software Library: with much larger set of programs to support general numerical computations. This require well organized documentation to enable good selection which will provide maximum compatibility, user convenience and reliability. The usage still require some programs efforts eg. IMSL, NAG, BMD.

(d) Software System: is a package which contains specialized user interface. The user writes (or interacts) in a special language. They serve as problem solving tools or programming tools. These include ELLPACK (ELLIPTIC PDE), Statistical Package for Social Scientist.(SPSS).

Other software systems not necessary for numerical computation had been developed to facilitate problem solving. These software systems require little or no programming to obtain meaningful results. These include Data Base Systems, Spreadsheets and some Word Processors.

It is based on the idea of this type of software system that the Computational SOLVER was designed to facilitate the majority of the function performed by the Finite Element User. This can be viewed as a modelling package comprising of Sheets and Subsheets, instead of files records and arrays.

4.2. The Computational SOLVER

The idea of interactive software system where the designer has the options to modify the Preprocessing procedure or the Processor procedure in other to produce a desired solution or output was implemented principally in Pascal programming language. Being interactive, it has the facilities for on-the screen modification that will effect changes in the entire model.

In view of the major objectives of this work which are: economy, modularity, simplicity, structured programming, clarity and space conservation, the entire software system had been structured to fall into five major components. These are:

(a) Discritization of the solution region into elements.

(b) Selection of interpolation function.

(c) Derivation of element stiffness matrix and load vector.

(d) Assemblage of element stiffness matrix and load vector into the global stiffness matrix and load vectors.

(e) Solution of the global systems of linear equations for the nodal values.

4.3 The Sheets.

These are highly structured and modular procedures in which there are nine sheets, six of which can have one or more subsheets. These sheets are automatically updated if there are changes in the input, operational or output values at any time and from any where in the interactive sheets. The relationship between sheets and subsheets are shown in Fig. 4.1.

The modularity of the SOLVER is based on the ability to:

(i) Define the variables that are to be used in the model or problem on the VARIABLE sheet.

(ii) Give the rules to operate on the defined variables on the RULE sheet.

(iii) Define the function that can be called from the rule sheet on the FUNCTION sheet.

(iv) Converts from one unit to another via the UNIT sheet.

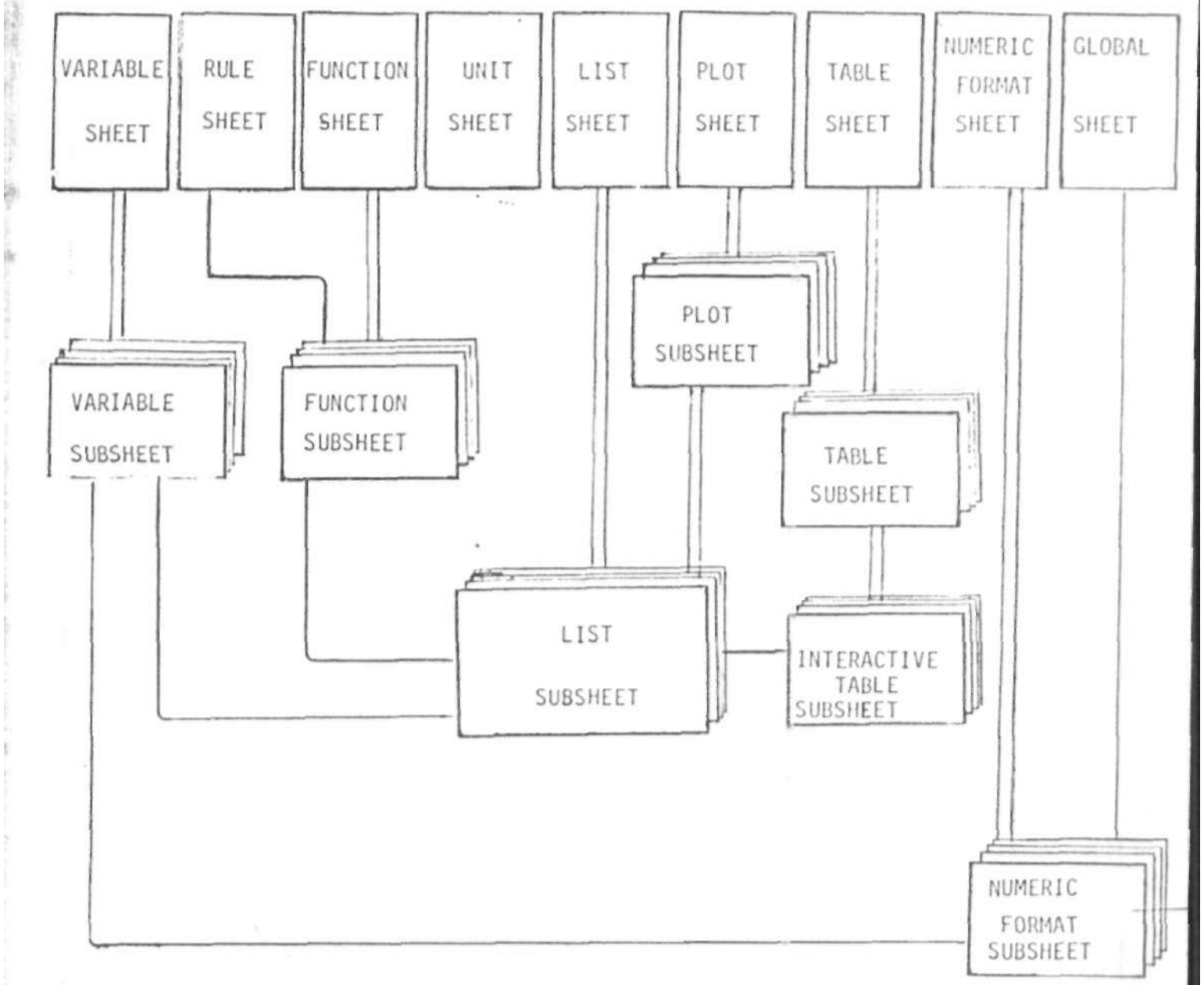


Fig 4.1: Relationship between Sheets and Subsheets.

(v) Define list of values, both numeric and literal that are used for calculation in other sheets and subsheets in the LIST sheet.

(vi) Do quick matrix plotting, using the PLOTsheet, directly on the screen to approximate the nature of chosen list of values from the list sheet.

(vii) To display selected output for concurrent study from the list subsheet and variable sheet. This is done by the TABLEsheet.

(viii) Control the format of numerical values on the other sheets and subsheets, on the NUMERICAL FORMAT sheet.

(ix) Control the handling of the entire settings and defaults in the whole SOLVER or model. This is done on the GLOBAL SHEET.

4.3.1 The Subsheets.

Out of the nine sheets enumerated above, six of them have subsheets. These subsheets are subordinates to the sheets and contain more detailed information about items specified on the sheets.

The subsheets are as follows:

(i) VARIABLE Subsheet:- each variable defined in the variable sheet has its own subsheet, where more information about the variable are defined.

(ii) FUNCTION Subsheet:- each function which is defined in the function sheet has a subsheet in which the necessary operations, to be performed by the function, are stated in form of rules or statements.

(iii) LIST Subsheet:- each list defined on the list sheet has its own subsheet in which the elements are enumerated together with their associated values which can be numerical or symbolic.

(iv) PLOT Subsheet:- each plot named in the plot sheet must have plot subsheet, where the type (pie, bar or line) and values to be plotted will be associated with the appropriate list from the list sheet.

(v) TABLE Subsheets:- each table-name given in the table sheet must have a table subsheet in which the display format will be specified. Each subsheet in

turn has an interactive subsheet similar to the convention electronic spreadsheet.

(vi) NUMERIC FORMAT Subsheets:- each Format name defined in the Numerical format sheet has a subsheet associated with it. In these subsheets the numeric notation, type of decimal point, digit grouping and significant digits are defined for each name.

The rule, unit and global sheet have not got any sub-sheets. But one can get into some other subsheets through the variable, function or list name, specified in any of the sheets or subsheets as the case may be.

4.3.2 The Structure of the Sheets.

The various sheets and subsheets are viewed through window on the screen. It may consist of one window, displaying a single sheet or two windows simultaneously, displaying two different information sheets (or different parts of the same sheet).

Generally, the solver has facility for:

(i) Navigation within a sheet. These are possible with the arrow keys just like in some text editors and word

processors. The use of Goto command to move to a particular line and column;

(ii) Navigation between sheets and subsheets by means of Dive; Return; Select; and switch commands.

(a) The Dive command is used to go into a subsheet of a selected name of Variable, List, Table, Plot, Numeric Format, or Functions.

(b) The Return command returns along the same route taken by one or more Dive commands.

(c) The Select command is used to replace the correct displayed sheet with the specified sheet.

(d) The Switch command is used to move between two displayed windows of different sheets (or the same sheet).

(iii) In field editor within sheets and subsheets, the editing commands include: Blank field; copy field; delete rows, insert rows; delete character.

Individual sheets have their specific structures and layout, such that the same subsheets of a particular

sheet will contain similar parameters and variables. The structure and function of the various sheets and subsheets are described in the next section.

4.3.2.1 The Variable Sheet and Subsheet.

In the variable sheet each variable is named with its major attributes defined. Each variable can have all or some of the following attributes:

(a) Status of the variable at any time before execution, is given in the Status field against the variable in question. A variable can have a status of a guess value, a list, input, output or blank.

(b) Input attribute: is where the input value is inserted.

(c) Name attribute: the actual name of the variable is stated.

(d) Output attribute: is where the output value is expected if the variable is an output variable.

(e) Unit attribute: specifies the unit to be used on the variable.

(f) Comment attribute: for the user specification of further information or functions of the variable.

```
========================== VARIABLE SHEET ==============================
STATUS----INPUT-----NAME-----OUTPUT----UNIT----COMMENT-------------------
========================================================================
          410      NGNODE                SC       ;Number of Global Nodes
            3      NLNOD                 SC       ;Number of Local Nodes
          400      NELE                           ;Number of Elements
            2      NCODS                 SC       ;Number of Co-ordinates
            3      NMAT                           ;Number of Material Properties
           50      NBAND                          ;Number of Bandwidth
```

Fig. 4.2: The Sample of a Variable Sheet

```
========================== VARIABLE: NBAND =============================
STATUS:                          L
FIRST GUESS:                     0
ASSOCIATED LIST:                 Bandlist
NUMERIC FORMAT:                  -
DISPLAY UNIT:                    SC
CALCULATION UNIT:                SC
```

Fig. 4.3: The Sample of a Variable Subsheet

The fields associated with these attributes for a particular variable can be left blank if it is not an input nor output field, see Fig 4.2. Each variable has subsheet in which further attribute are stated, as shown in Fig 4.3. These attributes include: the name of the associated list; the first guess value; the numeric format name (as stated in the numeric sheet); the display unit and calculation unit names, (as stated in the unit sheet).

```
============================== RULE SHEET ================================

Status--Rule-------------------------------------------------------------
=========================================================================

  ^       call BAND(NBAND)
          if NBAND <> 0 then call PROCESSOR(NBAND)
          call FORMKF

            -
            -
            -
            -
```

Fig. 4.4: The Sample of a Rule Sheet

```
============================== FUNCTION SHEET ============================

NAME--------TYPE-------ARGUMENTS--COMMENTS-------------------------------
=========================================================================

BAND         Procedure   (0:1)      A function that computes the Band wid

PROCESSOR    Rule        (1:0)      The function that gives the rules of
                                    Processor

Findval      List        (1:1)      A list function to find a mode in a r.
                                    of elements

  -
  -
  -
  -
```

Fig. 4.5: The Sample of Function Sheet

4.3.2.2 The Rule Sheet.

The rule sheet, unlike the variable sheet, has only two fields: the status and the rule field, Fig. 4.4. The rule sheet contains the rules for a problem. The rules can be in form of equations, function call and conditions.

The status field can contain an asterisk, which is automatically generated whenever a rule is entered or modified. A "greater than" sign (>) appear on the status field to indicate that an error occurred in the corresponding rule during execution. A "C" can be in the status field if corresponding rule has been cancelled. A blank status field indicates that a rule has been satisfied. Comments can be added to the rule sheet by placing them on the right of a quotation mark (").

Variable names used in the rules, equation and condition, in the rule sheet, are automatically added to the variable sheet. The rule sheet has no subsheet, but it is possible to go into any function subsheet if the function is selected. The list subsheet can also be accessed if a list function was accessed through the rule sheet.

4.3.2.3 The FUNCTION Sheet and Subsheets.

The function sheet is structured such that all the names of the user-defined functions in a problem are entered in the name field of the function sheet. Each stated function name requires to be given a type; the number of argument into and out of the function; and an optionally comment field for further information about the function. Fig. 4.5 gives a sample of a Function Sheet. There are three possible types of functions that can be defined in the function sheet, they are:

(a) The list function - to give the relationship between two sets of values, the domain (function argument) and the range (function value). It is mostly used for interpolation and mapping.

(b) The rule function contains a group of rules and accepts one or more input values and returns zero or more result values.

(c) The procedure function - contains a group of PASCAL-like statements which accepts one or more input values and returns zero or more result values.

```
============================== LIST FUNCTION: Findval ==================
Comment: A list function to find a value at a given node
Domain list: NODE
Mapping: Table
Range list: Value
Element-------Domain---------Range-------------------------------------
 1              6              2.3
 2              7              7.6
 3              8              9.3
 4              9              8.1
 5             10              5.7
 6             11              7.8
 -
 -
```

Fig. 4.6: The List Fuction Subsheet

```
============================== RULE  FUNCTION: Processor================
Comment:              The function that gives the rule of the processor
Parameter variables: -
Argument Variables:  NBAND
Result Variables:    -
Status---Rule-------------------------------------------------------------
*         call SHAPE(x)
          if x<>0 call GETMAT(y)
          call ELEM(ELE,NODE)
           -
           -
```

Fig. 4.7: Rule Function Subsheet

Each function defined in the function sheet has its own function subsheets. In these subsheets the exact values for the domain and range are given for a List function and, the statement are given for a procedure function.

The list function subsheet, Fig 4.6, requires the selection of a mapping or interpolation method for finding the function value when an argument is given. One of the following mapping or interpolation can be selected:

(a) The Table mapping - This is one-to-one mapping type in which one element of the domain list has a corresponding element in the Range list. An argument must be in the domain list, for the mapping to be executed.

(b) The Linear interpolation - This will search for the top of the domain list to locate the first pair of consecutive elements whose interval contain the value of the argument. The resulting function value is found by a linear interpolation using the position of the argument value within that interval to calculate a value in the interval between the corresponding pair of elements in the range list.

(c) The Cubic interpolation:- This is similar to the linear interpolation, except that the calculation of intermediate values between a pair of elements is based on cubic polynomial approximation.

The rule function subsheet, Fig. 4.7, will give additional information about the named rule function. The additional information are:

(a) Parameter variables:- the variables that will be regarded as global variables. This concept provides a direct link between the variable sheet and the rule function subsheet.

(b) Argument variables:- these are variables which the rule function subsheet will use to represent variables from either the rule sheet or other functions. An argument variable cannot be used as a name in the parameter variable as they are locally defined with the rule function subsheet and has no meaning outside the rule function subsheet.

```
============================== PROCEDURE FUCTION: BAND ================
Comment:                A function that computes the band width
Parameter Variables: NELE,NLNOD
Input Variables:     -
Output Variables:    NBW
Status---Rule-------------------------------------------------------------
         NDW:=0
         if NLNOD = 2 the NBW:=2 else goto cont
         return
         cont:
             for i:=1 to NELE
               for k:=1 to NLNOD
                  j:=k+1
                  if k=3 then j:=1
                ND:=NODE(i,k) - NODE(i,j)
                if NDW < ND the NDW:=ND
               next k
             next i
```

Fig. 4.8: Procedure Function Subsheet

(c) Result Variables:- These are variables whose values are returned by the rule function. The result variable name cannot be used as a name in parameter variables or argument variables.

(d) The Status field:- This field is used to indicate the status of the executed rule. The indicator is similar to that on the rule sheet. Following the status field is the rule field where the required rules are stated by row. Each rule has its own status field and when ever an error occurs in a statement it is indicated on the status field.

All the rules defined here will define the rule function.

The procedure function subsheet, Fig. 4.8, is essentially the same as the rule function subsheet except that the argument and parameter variables are replaced by the input and output parameter respectively and, the rule field is replaced by the statement field. The statements are sequential actions in a PASCAL-like programming language which has the capability of branching and looping. Unlike the rule function, procedure function can be used recursively.

4.3.2.4 Unit Sheet

The unit sheet is used to define conversions between units of measurements used in a problem. In the unit sheet there are four fields for defining conversion: From-field, To-field, Multiply-by-field and Add-offset-field. The name of the unit from which conversion is intended is entered in the From-field while the name of the unit to which the conversion is made is entered in the to-field. The value of the multiply-by, which the From-field will be converted to the to-field is given in the multiply-by field. If there is an offset which will make the conversion complete, its value is entered in add-offset-field. A good example, Fig. 4.9, is the conversion of temperature from celsius to fahrenheit - multiply the celsius by 1.8 and add 32 to give the fahrenheit. The 1.8 is the multiply-by value while 32 is the add-offset value. The unit function has no subsheet and diving into it from another sheet or diving into another sheet from it is not possible. Unlike the rule sheet, which does not have a subsheet but diving into a function subsheet and even a list subsheet through list function subsheet is possible.

UNIT SHEET

From	To	Multiply by	Add offset
cels	fah	1.8	32.0
in	cm	2.5	0.0

Fig. 4.9: Unit Function Sheet

LIST SHEET

Name	Element	Unit	Comment
PBOUND	30	cm	Prescribed boundary values
GBNDNO	500	cm	Global node number

4.3.2.5 List Sheet and Subsheet.

A list is a set of numeric or symbolic values called elements. The list sheet, Fig. 4.10, is used to define list name, their scope, the required unit and comment (not executable) about the list.

Each list name defined in the list sheet has its own list subsheet which will contain the list elements and additional information. In addition to the attribute given in the list sheet, the numeric format of the list values, display unit of the list values, and calculation unit of the list values can be defined in the list subsheet, as can be seen in Fig. 4.11, as SC, CM, and CM respectively. The element number are automatically enumerated in the element field as the actual values are entered in the value sheet. The value can be numeric, symbolic or any expression that will evaluate into numeric or symbolic value.

The list subsheet can be dived into from various subsheets like the variable subsheet, plot subsheet, table subsheet and interactive table subsheet, and list function subsheet. Diving can be made into only the numerical format subsheet from the list subsheet.

```
============================== LIST: PBOUND ============================
Comment:              Prescribed boundary values
Numeric Format:       sc
Display Unit:         cm
Calculation Unit:     cm
Element-----Value-------------------------------------------------------
1                 5
2                 7
3                10
4                15
6                27
-
-
-
-
```

Fig. 4.11: List Subsheet

```
==============================TABLE SHEET ============================
Name---------Title-----------------------------------------------------
=======================================================================
ENORM         The display of error values
EVAL          Evaluation of the nodal parameter
EXACT         Calculates the exact values
-
-
-
-
```

Fig. 4.12: Table Sheet

4.3.2.6 Table Sheet and Subsheet.

The table sheet can be used to define the names of various tables which will be used to show the relationship between values of corresponding elements in different list by displaying the values in columns and rows. The table sheet, Fig. 4.12, contain only two fields: the name field for defining the name of tables; and the title field for the table title.

Each table name given in the table sheet has a table subsheet where detailed information is given about the table, Fig. 4.13. In the table subsheet information like:

(a) Vertical or Horizontal display lists can be stated.

(b) Row and column separator - which will be characters for separation.

(c) First and last elements to be displayed in a situation where the entire list is not required.

(d) The name of the lists to be related are enumerated under the list field.

(e) Numerical format of display values can be selected for any list and entered under the numerical format field of the subsheet.

```
========================== TABLE: EVAL ============================
Screen or Printer:  Screen
Vertical or Horizontal:          Vertical
Row Separator:       -
Column Separator     ¦
First Element        0
Last Element         500
List-----Numeric Format---Width---Heading--------------------------
DTx       sc                  8     The Derivatives with respect to x
DTy       sc                  8     The Derivatives with respect to y
T         eng                 7     The resulting nodal values
Ele                           7     The element numbers
-
-
-
-
-
```

Fig. 14.13 Table Sheet

```
======================= INTERACTIVE TABLE: EVAL=====================
Title:
List------1------2------3------4------5------6------7------8------9-
DTx      0.22     0.23     0.24     0.25     0.26     0.27     0.28     0.29     0.30
DTy      1.56     1.57     1.45     1.68     1.88     1.99     2.11     2.50     2.75
T           1        2        3        4        5        7        8        9       10
-
-
-
-
```

Fig. 4.15 Interactive Table Sheet

```
======================= NUMERICAL FORMAT SHEET ======================
Name-----------Comment-------------------------------------------------
=================================================================
sc          Format for scientific models
eng         Format for engineering models
cm          General format
-
-
-
```

Fig 4.15: Numerical Format Sheet

(f) The width of the display element of a list can be given under the width field.

(g) The heading for each list can be given correspondingly under the heading field.

(h) The display table can be directed to the screen, or printer and screen. This can be selected by using the screen or printer option.

A dive into the interactive table subsheet, Fig. 4.14, will show a row and column display of the table defined in the table sheet based on the detailed information given in the table subsheet. Any change made on the interactive table subsheet will be effected in both the table subsheet as well as corresponding list in the list subsheet.

A dive is possible from the table sheet through the table subsheet and interactive table subsheet down to the list subsheet, when the desired list is selected in the table subsheet.

4.3.2.7 Numeric Format Sheet and Subsheets.

The numeric format sheet, Fig. 4.15, holds the names of the numeric formats used to control the appearance of numeric values displayed on the screen and in printed output.

NUMERIC FORMAT: sc

Numeric Notaion:	Scientific format
Significant digits:	15
Decimal places:	2
Padding:	None
Zero Representation:	0
+/- Notation:	- only
Justification:	Right

Fig. 4.16: Numerical Format Subsheet

PLOT SHEET

Name	Plot Type	Title
Error-plot	Line	Error variation with no. of elemen
Exact-plot	Line	Plot of exact values
Aprox-plot	Line	Plot of approximate values
Mixed-plot	Bar	Comparition of space consumption

Fig. 4.17: Plot Sheet

A numeric format defined in the numeric format sheet can be used: in the variable sheet/subsheet, in the list subsheet, in the table subsheet, and in the Global sheet.

The numeric format defined in the Global sheet is the format that will be used for values through out the problem, except another format name is defined in: variable, list or table subsheet.

Each numerical format name on the numeric format sheet has a corresponding numeric format subsheet, Fig. 4.16, which contains the actual specification controlling the appearance of numeric values.

A numeric format subsheet contains the following specification:

(a) Type of numeric notation, either scientific or decimal.

(b) Number of desired significant digits.

(c) Number of desired decimal places.

(d) Whether values after the decimal point be padded with zeros or blanks.

(e) Type of zero representation either zero blank or any character.

(f) Type of representation of negative value (-, + or 0).

```
========================= LINE CHART Exact-plot ==================
Title:              Error variation with number of elements
X-Axis Label:       Point Values (selected)
Y-Axis Label:       Nodal values
X-Axis Min, Max:    0,
Y-Axis Min, Max:    0,
X-Axis List:        px
Y-Axis----First-----Last-----Style-------Character-----------------
 Ty         0                 Line
 Dy         0                 Symbol          *
-
-
-
-
-
```

Fig. 4.18: Line Plot Subsheet

```
========================== BAR CHART: Mixed ======================
Title:              Comparition of memory space by number of elements
X-Axis Label:       Memory space
Y-Axis Label:       Number of elements
X-Axis List:        ELENO
Y-Axis list(s):     MEMSIZE
```

Fig. 4.19: Bar Plot Subsheet

```
========================== PIE CHART: Memory =====================
Title:        % memory space used by each procedure
Data List:    MEMDATA
Label List:   PRONAME
```

Fig. 4.20: Pie Plot Subsheet

(g) The alignment of the value which includes:

Justification, left margin and right margin, and suffix and prefix.

Diving into the numerical format subsheet is possible from the numeric sheet and global subsheet.

4.3.2.8 Plot Sheet and Subsheet.

The plot sheet and subsheet can be used to give a rough graphical depiction of corresponding elements in different lists. The plot sheet, Fig. 4.17, summarizes all plots that can take place in a problem by name, type and title. The type of graph can be line, bar or pie chart, and can be obtained by using Fig. 4.18, Fig. 4.19 and Fig. 4.20 respectively. The desired option are specified in the appropriate field.

Each plot name in the line plot sheet has a plot subsheet in which more information about the desired plotting will be given. For the plot sheet the required information include: X- and Y-Axis; Maximum and Minimum Value; Title; X- and Y-Axis label; the X-axis list name; the various Y-axis list name; and their corresponding first and last values, style and plot characters.

4.3.2.9 The Global Sheet.

The global sheet controls settings and defaults in effect throughout the problem. In this sheet specifications like maximum iteration count; global numeric format, whether to stop on list error or to continue, printer setting and automatic iteration are specified. The settings are subject to changes whenever problem changes. It is the first sheet to be loaded to initialize the entire SOLVER.

4.4 The SOLVER Implementation.

As discussed during the design stage of Finite Element programming in section 3.1.1, the main program is structured into three modules: PREPROCESSOR, PROCESSOR, and POSTPROCESSOR.

Using the capability of SOLVER, the three modules can be called from the RULE Sheet. They are sequentially arranged in the rule sheet in order of execution.

Each module in the rule sheet will use other tools in the SOLVER to implement the function associated with it.

4.4.1 The Tools for the PREPROCESSOR.

The tools required from the SOLVER, for the implementation of the PREPROCESSOR are:

(a) The Variable Sheet: This will hold all singly defined variables, and control variables. Some of these variables are automatically entered into the variable sheet when they are used in the rule sheet.

(b) The Variable Subsheet: Here the required array is entered by name and associated with a variable name.

(c) Function Subsheet: Where variables to hold the nodal, element, boundary and material property values are specified so that they will have a direct link with the list subsheet.

(d) List Subsheet: The actual values are entered into the appropriate list subsheet, the numeric value is automatically inserted in the list sheet.

This can be seen in Fig. 4.21.

The programming effort required for reading input values has been removed as the single values and arrays can be

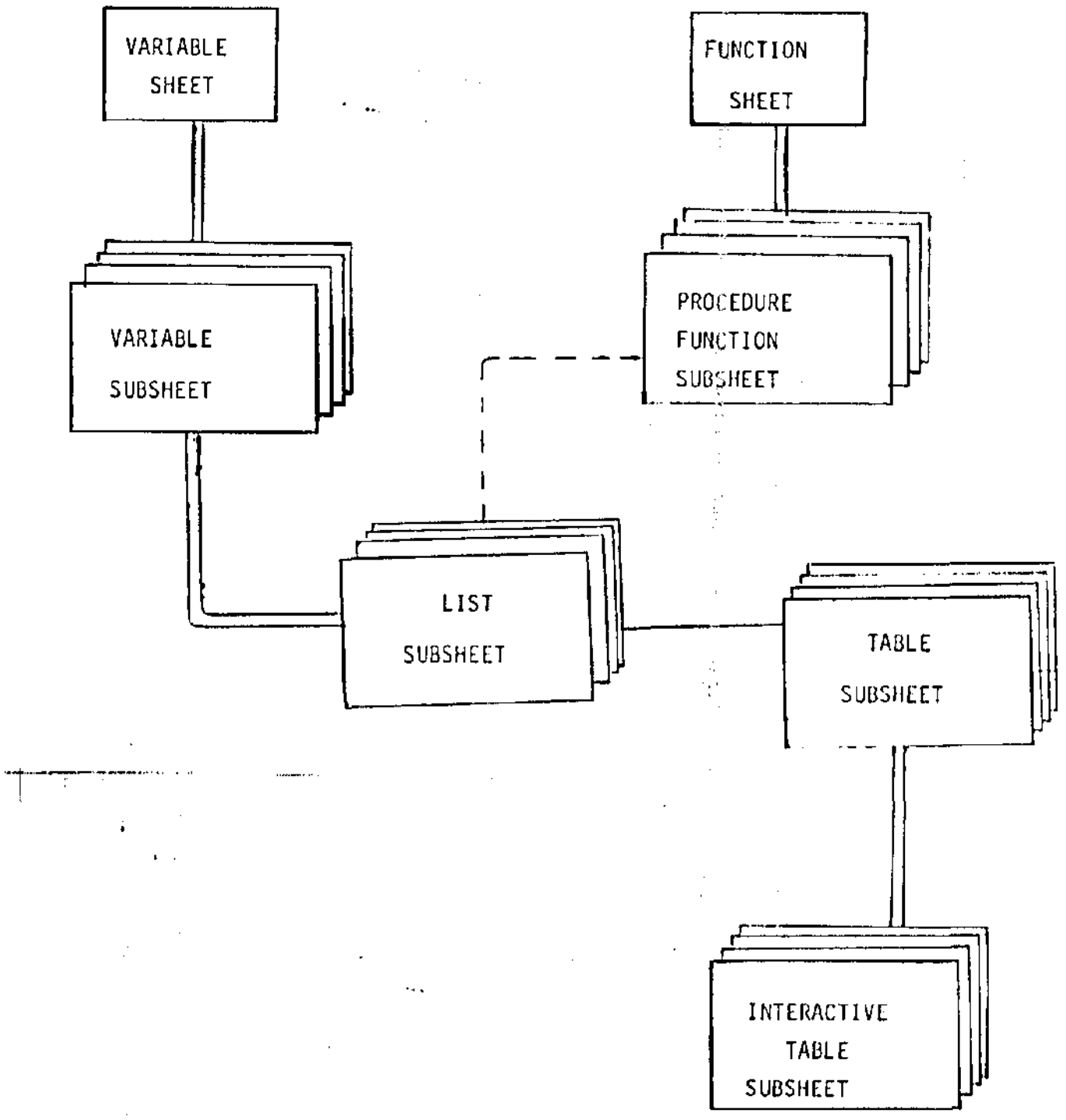


Fig 4.21: The Tools for the PREPROCESSOR Routine.

directly entered either through the variable sheet or on the list subsheet. In addition to this there is the flexibility in direct modification of variable and array value without executing the program again. The interactive table subsheet is a more flexible means of data entry for the preprocessing activity.

4.4.2 The Tools for the PROCESSOR.

The PROCESSOR is made up of several procedures which were all defined in the function sheet. Principally, three procedures: FORMKF, APLBC, and SOLVE are called from the PROC Function subsheet shown in Fig. 4.22.

(i) FORMKF - Formulate the element and Global stiffness matrix together with the element and global load vector.

(ii) ASSMB - Assembles the element stiffness matrix and load vector into their global component.

(iii) APLYBC - Perform the function of applying the boundary condition to the assembled global matrix and load vector.

(iv) SOLVE - will solve the resulting systems of linear equation.

Each of these procedures in turn invokes subsidiary procedures from which more information will facilitate the desired result. The PROC is invoked from the rule sheet with

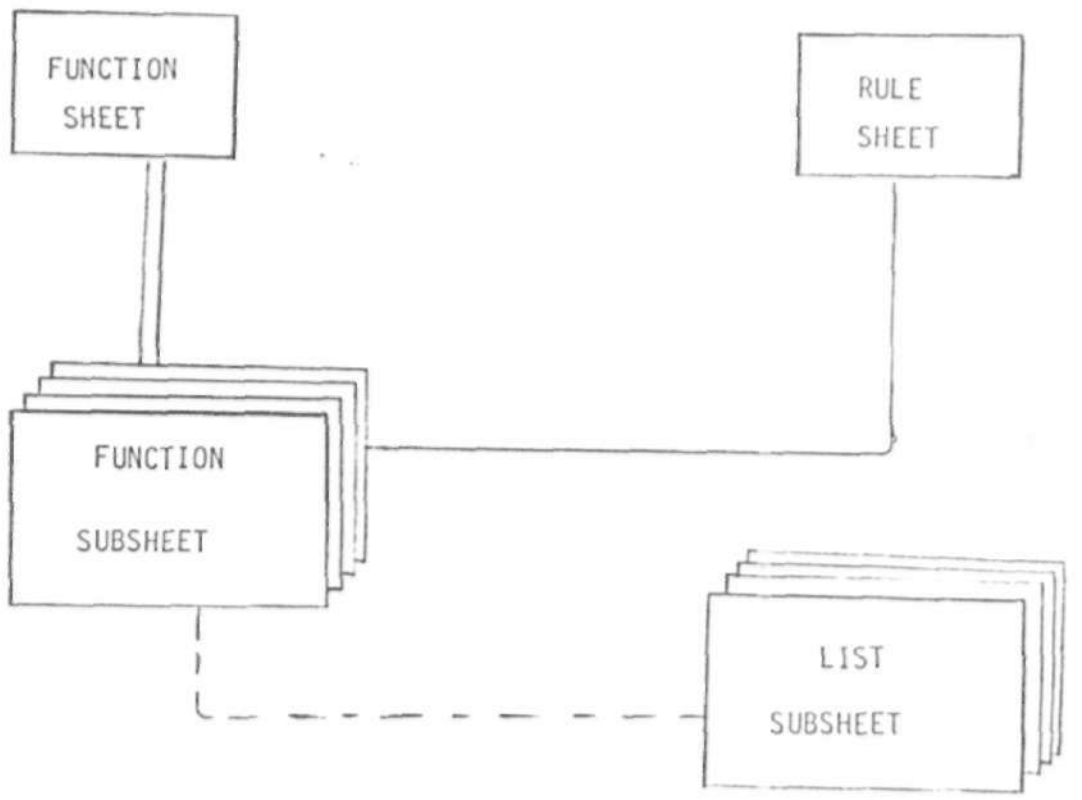


Fig 4.22: The Tools for the PROCESSOR.

the appropriate arguments which are transmitted to the subsequent procedures in their various function subsheet. Values resulting from the execution of these procedures and functions will go straight to the list subsheet, from where they can be subsequently used by other procedures.

4.4.3 The Tools For POSTPROCESSOR.

With the help of the Table sheet, numerical format sheet and subsheet, values of the outcome of the PROCESSOR from the function rule and variable sheet, can be displayed on the Table Subsheet. This can be translated for interactive modifications on the interactive Table subsheet, Fig. 4.23.

Sketches of relationship between the exact and approximate values can be produced by using the plot sheet and subsheet to specify the required relationships. Selection of a desired plot from the plot sheet is done automatically.

The use of numerical format from the numerical format sheet/subsheet will affect the values of the desired displays on the connected list or Table-sheet and even on the plot sheets. If no particular format is specified for specific lists or variables, the format specified on the global sheet serves as the default format.

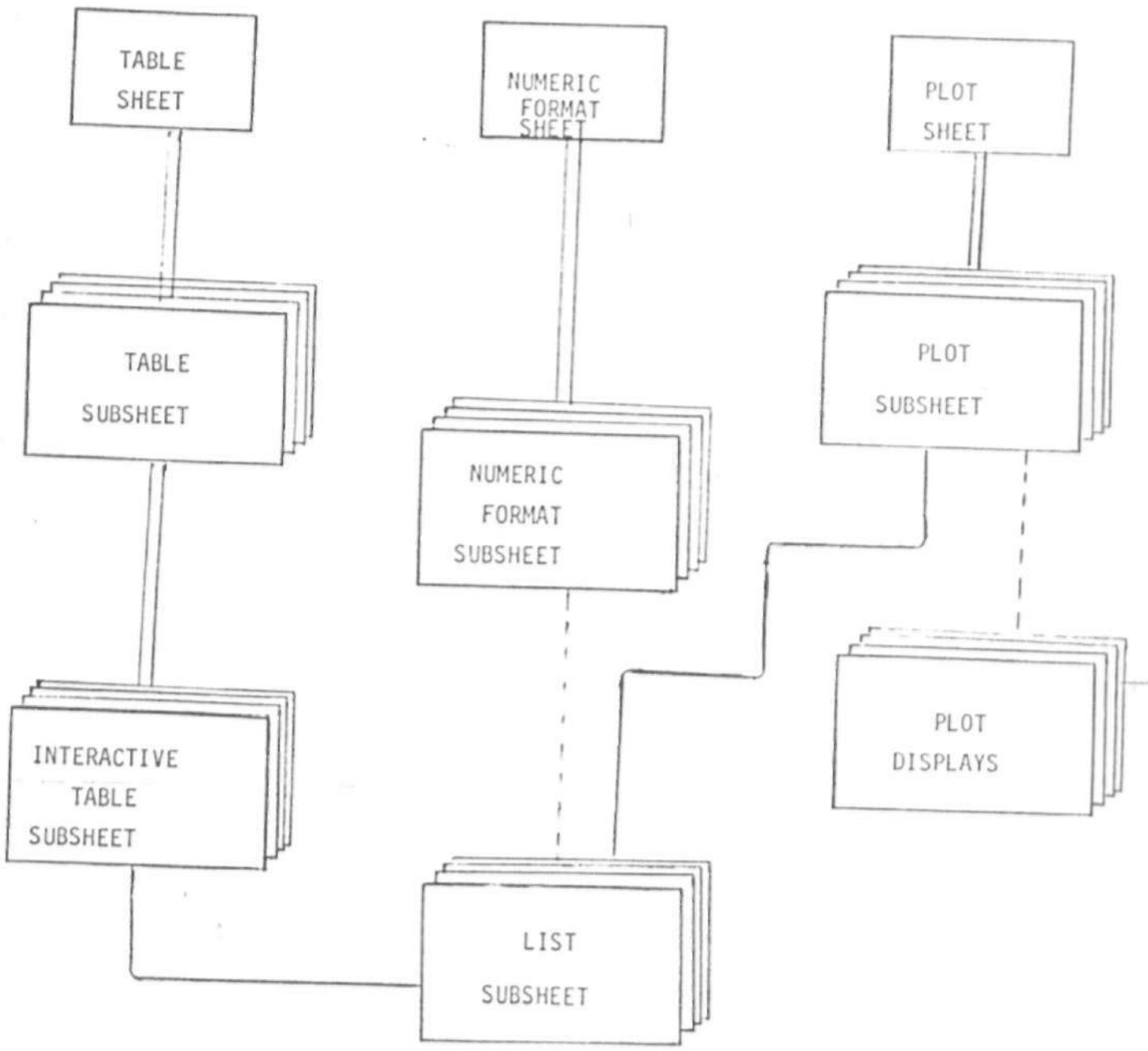


Fig 4.23: The Tools for POSTPROCESSOR.

4.5 The Central List Subsheets.

All the sections of the major modules has got the list subsheet as the data repertoire, Fig. 4.24. The list subsheet can be accessed from anywhere in the problem model except from the unit, numerical format and global sheets. The Central List subsheet can be accessed as follows:

(i) Assigning a list name to a variable in the variable sheet will take the variables as an array.

(ii) Assign a list name to the domain or range of the list function subsheet will be used as values for interpolating polynomial.

(iii) Values to be plotted are assigned the X-and Y-axis field of the plot subsheets by List name only.

(iv) Tabulated values in the interactive Table subsheet are assigned to list field of the Table Subsheet by giving a list name.

In all these cases the values need not be transferred to the various named locations rather, during execution these names are used for direct access to the centralized list subsheet.

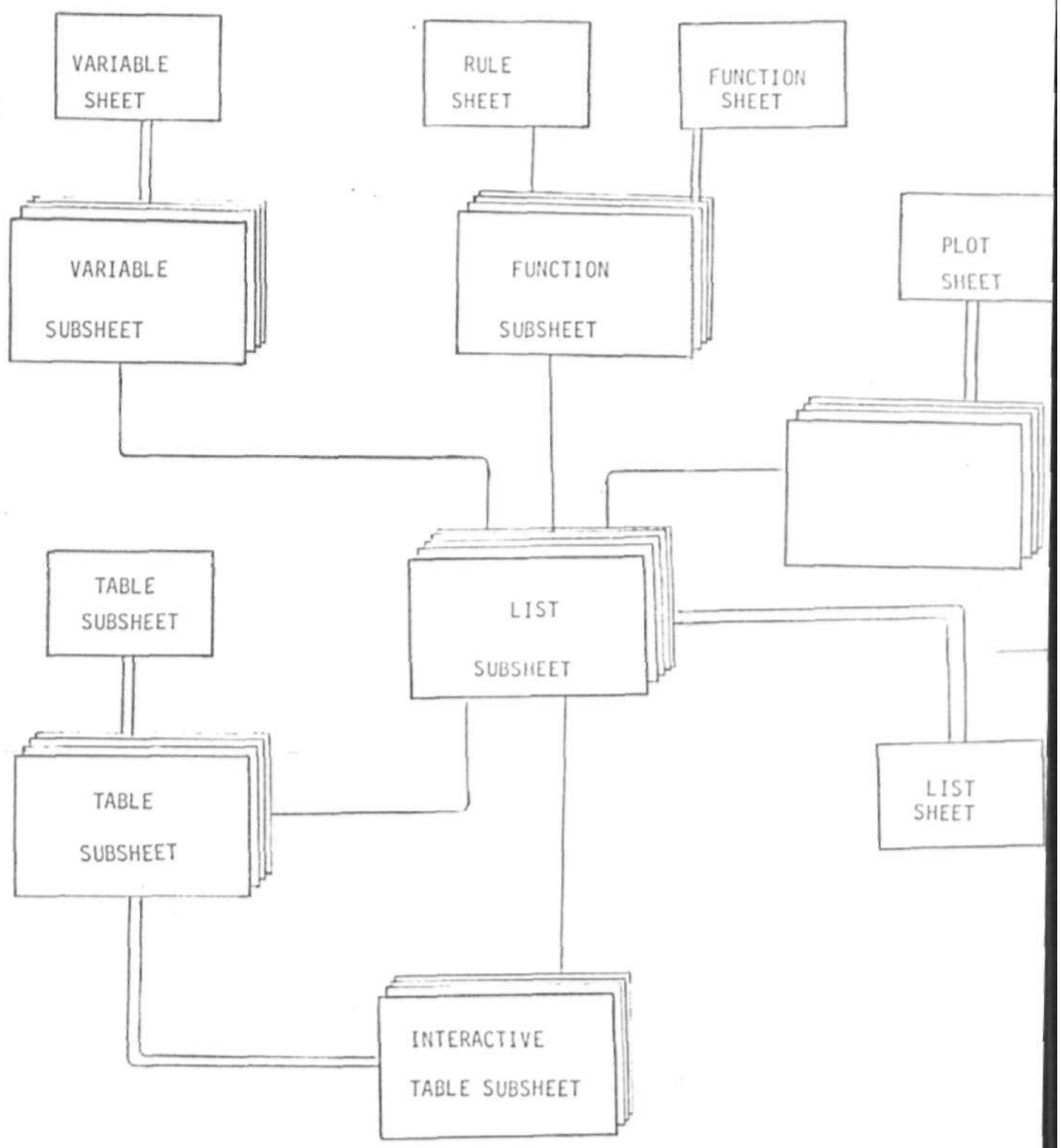


Fig 4.24: The Centralized List Subsheet.

Chapter 5.

EXPERIMENTAL RESULTS AND SOFTWARE EVALUATION

We now apply the approach discussed in Chapter 2 to implemente the SOLVER package of Chapter 4.

Four problems are solved using the variational formulation of Finite Element Method. Since the solvable problem are not generalized and the aim is the understanding and minimization of computer resources, we have limited the experimental problems to four. Moreover the selected problems were such that other experimental packages can be applied (and in fact the problems were the ones that met the requirements of these experimental packages).

We consider the following problems:

(a) Fluid flow between flat permeable plates

(b) Calculation of pressure on Hydrostatic pad

(c) Heat conduction and convection in two dimensional body.

(d) Irrotational Fluid flow.

5.1 Heat Conduction and Convection in two dimensional Body.

We now discuss heat conduction in the interior of a very long body with various boundary conditions, including convection around the body. The emphasis in the experiment is to apply the discussed software approach which demonstrate the elimination of the imperative sophistication (or complexity), particularly with some complex real-life problem.

We assume that the convection loss from the internal region of the fin is negligible. The heat condition has a conductive coefficient k. The boundary conditions on various sides (Fig. 5.1) are:

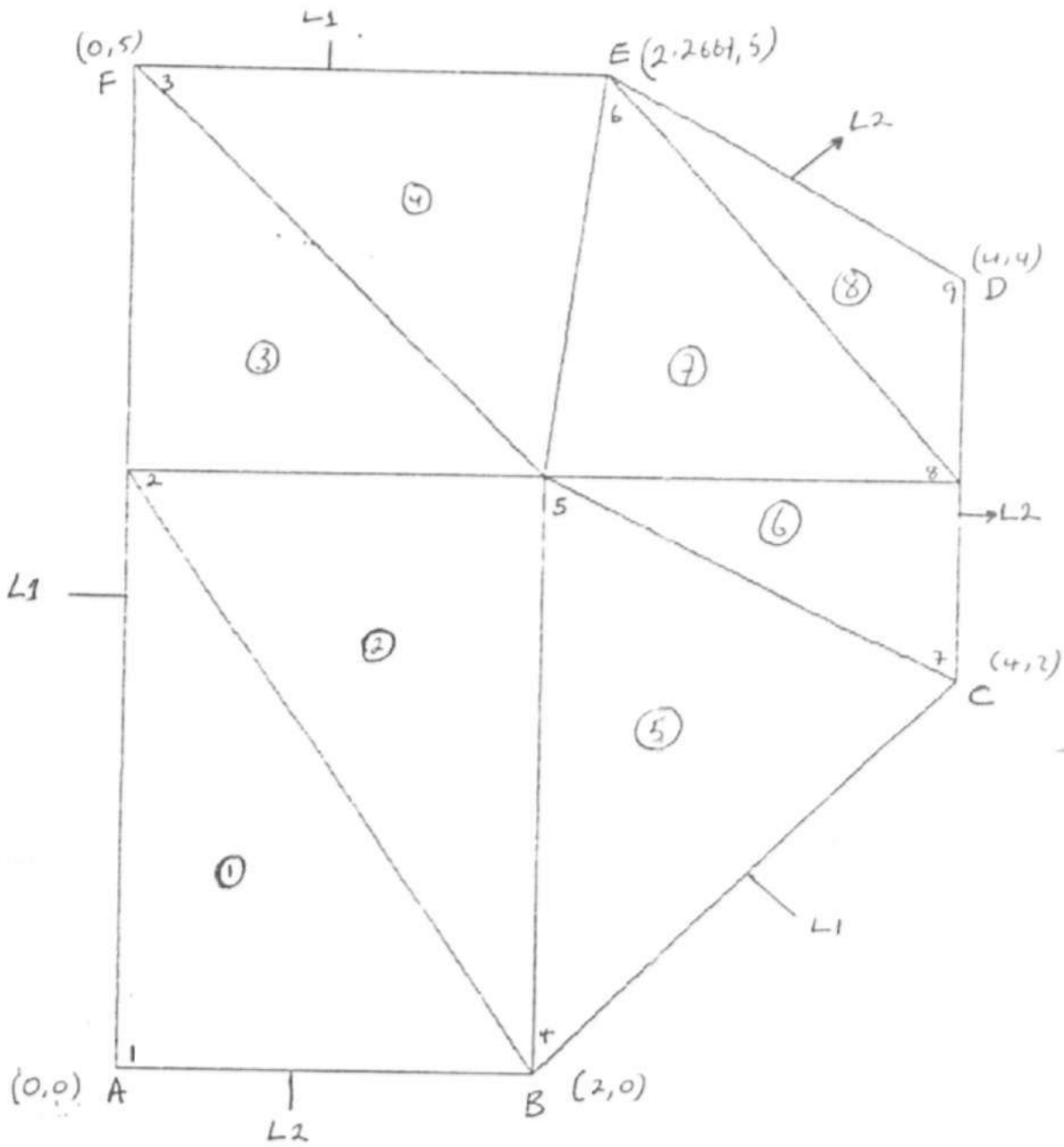


Fig.5.1: Problem Geometry and Boundary Condition

Table 5.1. The Boundary Condition for Heat Conduction and Convection in a 2D Body.

SIDE	DESCRIPTION	BODY CONDITION	SIDE TYPE
AB	Insulated	Derivative boundary condition: $K_t\frac{\partial T}{\partial x}\eta_x + K_t\frac{\partial T}{\partial y}\eta_y = 0$	L2
BC	Linearly Distributor	From 200 oC (at B) to 400oC (at C)	L1
CD	Convection heat flow	Derivative boundary condition: $K_t\frac{\partial T}{\partial x}\eta_x + K_t\frac{\partial T}{\partial y}\eta_y + h_t(T-T_\infty) = 0$	L2
DE	Convection heat flow	Derivative boundary condition: $K_t\frac{\partial T}{\partial x}\eta_x + K_t\frac{\partial T}{\partial y}\eta_y + h_t(T-T_\infty) = 0$	L2
EF	Prescribed boundary condition (constant)	All nodes at the boundary has 400 oC	L1
FA	Prescribed boundary condition (constant)	All nodes at the boundary has 400 oC	L1

The heat conduction in the 2D body is governed by the differential equation:

$$\frac{\partial}{\partial x}(K_t\frac{\partial T}{\partial x}) + \frac{\partial}{\partial y}(K_t\frac{\partial T}{\partial y}) = 0 \quad \text{.....(5.1)}$$

with constant conduction coefficient, K. The thickness t, of the body is assumed to be 10 mm such that

$$k = 0.25 \text{ w/mm } ^\circ\text{C}, \quad h = 0.001 \text{ w/mm } ^\circ\text{C}, \quad T_\infty = 50 \ ^\circ\text{C}.$$

The boundary conditions are tabulated in Table 5.1.

Equation 5.1, when compared with equation 2.2, i.e.

$$\frac{\partial}{\partial x}(K_x\frac{\partial T}{\partial x}) + \frac{\partial}{\partial y}(K_y\frac{\partial T}{\partial y}) + P(x,y)T + Q(x,y) = 0$$

will give $K_x = K_y = K_t = 0.25$ w/cc, $P(x,y) = 0$ and $Q(x,y) = 0$.

The boundary condition on the side AB when compared with the general boundary condition, equation 2.3, i.e.

$$K_x\frac{\partial T}{\partial x}\eta_x + K_y\frac{\partial T}{\partial y}\eta_y + \alpha(x,y)T + \beta(x,y) = 0$$

will give $\alpha(x,y) = 0 = \beta(x,y)$ meaning that the side AB is insulated. Sides CD and DE, when compared with the same equation 2.3 will imply $\beta(x,y)$ to equal to -ht T_∞ which is 0.050 w/mm.

Table 5.2 Elements and their associated Nodes for the Heat Conduction problem.

Element	Nodes
1	1 4 2
2	5 2 4
3	2 5 3
4	6 3 5
5	4 7 5
6	8 5 7
7	5 8 6
8	9 6 8

Table 5.3a The nodal temperature distribution for 8 elements

Node	Nodal Value
1	400.0000
2	400.0000
3	400.0000
4	200.0000
5	302.1307
6	400.0000
7	100.0000
8	203.8213
9	259.1417

Table 5.3a 5.3b and 5.3c had shown the temperature

Table 3.5b: Nodal Temperature Distribution for 30 Elements

Coordinates X	Coordinates Y	Element Number	Temperature	Nodal Number
0.0000	0.0000	10	400.0000	1
0.0000	3.0000	16	400.0000	2
0.0000	5.0000	3	400.0000	3
2.0000	0.0000	11	200.0000	4
2.0000	3.0000	13	302.1400	5
2.2667	5.0000	32	400.0000	6
4.0000	2.0000	23	100.0000	7
4.0000	3.0000	8	203.0000	8
4.0000	4.0000	31	259.1400	9
1.0000	0.0000	10	300.0000	10
1.0000	1.5000	2	300.0000	11
0.0000	1.5000	10	400.0000	12
1.0000	3.0000	16	351.0700	13
2.0000	1.5000	13	251.0700	14
1.0000	4.0000	15	351.0700	15
0.0000	4.0000	3	400.0000	16
1.1334	5.0000	19	399.9999	17
2.1334	4.0000	7	351.0700	18
3.0000	1.0000	21	150.0000	19
3.0000	2.5000	23	201.0700	20
3.0000	3.0000	27	252.5700	21
4.0000	2.5000	6	151.5000	22
3.1334	4.0000	27	301.5000	23
3.1334	4.5000	31	329.5701	24
4.0000	3.5000	31	231.0700	25

Table 5.3c: Nodal Temperature Distribution for 128 Elements

Coordinates X	Coordinates Y	Element Number	Temperature	Nodal Number
0.0000	0.0000	43	[illegible]	1
0.0000	3.0000	1	100.0000	2
0.0000	5.0000	3	400.0000	3
2.0000	0.0000	2	200.0000	4
2.0000	3.0000	5	[illegible]	5
2.2667	5.0000	76	400.0000	6
4.0000	2.0000	6	100.0000	7
4.0000	3.0000	8	200.0000	8
4.0000	4.0000	124	[illegible]	9
1.0000	0.0000	41	300.0000	10
1.0000	1.5000	40	300.0000	11
0.0000	1.5000	9	400.0000	12
1.0000	3.0000	65	[illegible]	13
2.0000	1.5000	46	251.0000	14
1.0000	4.0000	62	[illegible]	15
0.0000	4.0000	15	400.0000	16
1.1334	5.0000	75	[illegible]	17
2.1334	4.0000	74	251.0000	18
3.0000	1.0000	35	150.0000	19
3.0000	2.5000	33	201.0000	20
3.0000	3.0000	97	252.5000	21
4.0000	2.5000	24	151.5000	22
3.1334	4.0000	122	[illegible]	23
3.1334	4.5000	121	[illegible]	24
4.0000	3.5000	31	251.0700	25
0.5000	1.5000	35	350.0000	26
0.5000	2.2500	56	350.0000	27
0.0000	2.2500	1	400.0000	28
1.0000	0.7500	36	300.0000	29
0.5000	0.7500	37	350.0000	30
0.5000	0.0000	41	350.0000	31
0.0000	0.7500	40	400.0000	32
1.5000	0.0000	42	250.0000	33
1.5000	0.7500	11	350.0000	34
1.5000	1.5000	45	275.5350	35
2.0000	0.7500	33	225.5350	36
1.0000	2.2500	50	325.5350	37
1.5000	2.2500	43	301.0700	38
1.5000	3.0000	52	[illegible]	39
2.0000	2.2500	53	276.0000	40

Table 5.3c: Continued

Coordinates X	Coordinates Y	Element Number	Temperature	Nodal Number
0.5000	3.0000	54	375.4350	41
0.5000	4.0000	59	400.0000	42
0.5000	4.5000	80	375.5349	44
0.0000	4.5000	57	400.0000	44
1.0000	3.5000	67	351.6700	45
0.5000	3.5000	15	375.4350	46
0.0000	3.5000	63	400.0000	47
1.5000	3.5000	17	326.6050	48
1.5667	4.0000	71	351.0693	49
2.0667	3.5000	111	326.6050	50
1.0667	4.5000	78	375.5348	51
1.6334	4.5000	73	375.5352	52
1.7000	5.0000	77	399.9999	53
2.2000	4.5000	7	375.4350	54
0.5667	5.0000	80	400.0001	55
2.5000	2.0000	83	226.6700	56
2.5000	2.7500	104	251.6650	57
3.0000	1.7500	86	176.5350	58
2.5000	1.2500	22	200.0350	59
2.5000	0.5000	89	175.0000	60
3.5000	1.5000	91	151.9900	61
3.5000	2.2500	23	150.0350	62
3.5000	2.5000	96	175.0350	63
4.0000	2.2500	6	125.7500	64
3.0000	2.7500	102	226.8200	65
3.5000	2.7500	24	202.0350	66
3.5000	3.0000	116	227.7850	67
4.0000	2.7500	100	177.2500	68
2.5000	3.0000	113	277.2550	69
2.6334	4.0000	27	326.0352	70
2.7000	4.5000	128	350.2501	71
3.0667	3.5000	110	277.6352	72
2.5667	3.5000	113	301.6200	73
3.5667	3.5000	117	252.3505	74
3.5667	3.7500	30	264.8350	75
4.0000	3.2500	8	247.6349	76
3.1334	4.2500	127	315.4350	77
3.5667	4.0000	121	289.8200	78
3.5667	4.2500	123	314.3549	79
4.0000	3.7500	31	245.1949	80
2.7000	4.7500	130	364.7851	81

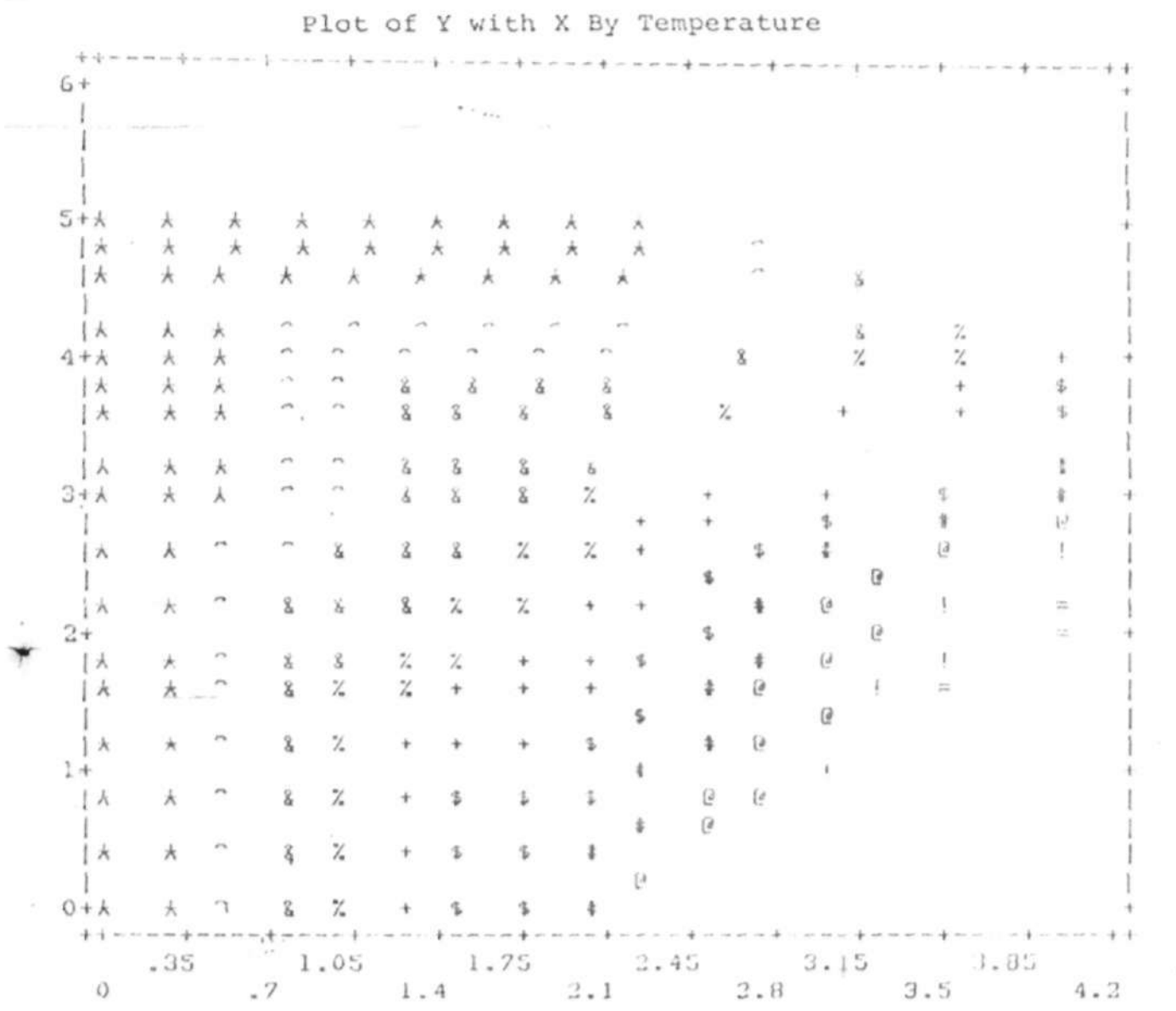
distribution in the region for 8, 32, 128 elements respectively. The experiment continued for elements up to 100,000 elements. The temperature distribution for 360 and 1440 element are shown in Fig. 5.2 and Fig. 5.3.

5.2 Irrotational flow of Fluid

The formulation of the irrotational flow problem in terms of the velocity potential function yields the governing equation.

$$\frac{\partial^2 \mu}{\partial^2 x} + \frac{\partial^2 \mu}{\partial^2 y^2} = 0 \quad \text{.....(5.3)}$$

In this case we considered the flow about a cylinde,r de Vries and Norrie (1971). We determined the streamlines for flow around a symmetry body 40 mm in diameter centered between two parallel walls, which are 80 mm apart. The fluid has a uniform velocity of 40 mm/sec at a location far removed from the cylinder. Due to symmetry only the region ABCDEA is considered Fig. 5.4. Table 5.4 shows the boundary conditions.



The bounds of Temperature are defined as follows:

=: 100.000- 130.000 ': 130.000- 160.000 @: 160.000- 190.000
$: 190.000- 220.000 $: 220.000- 250.000 +: 250.000- 280.000
%: 280.000- 310.000 &: 310.000- 340.000 *: 340.000- 370.001
A: 370.001- 400.001

Fig. 5.2: Temperature Distribution for 36 Elements

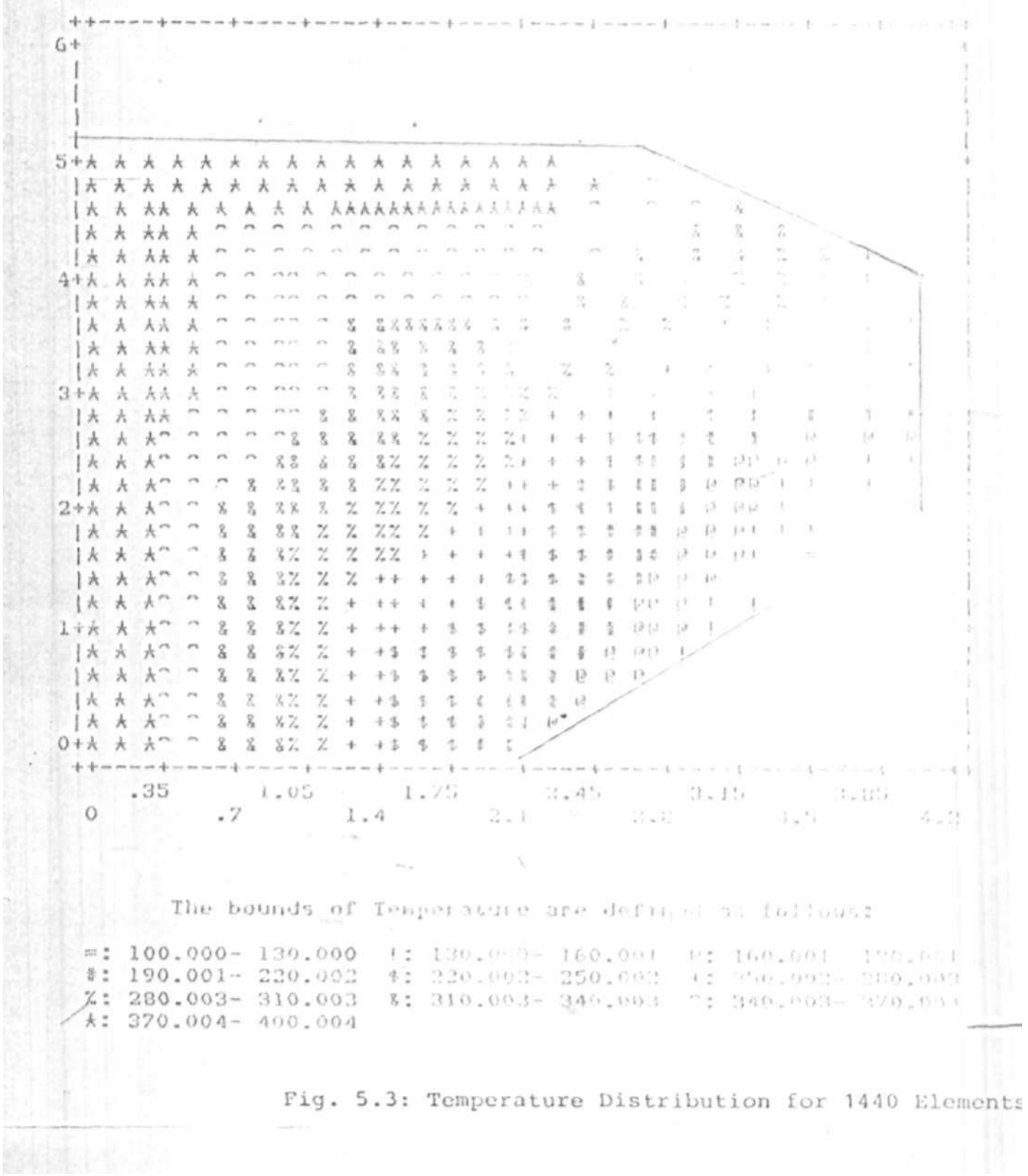


Fig. 5.3: Temperature Distribution for 1440 Elements

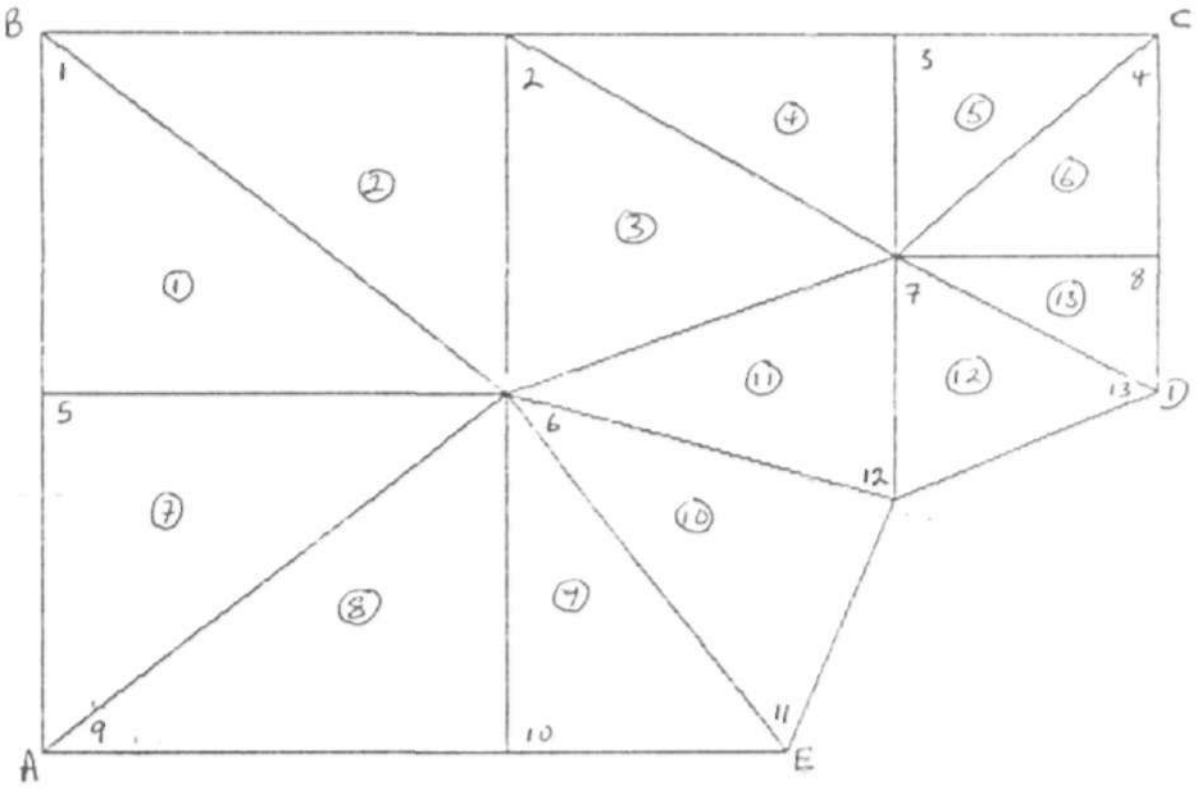


Fig. 5.4: Geometry of Irrotational Flow Problem

Table 5.4 Prescribed Boundary condition for irrotational Fluid Flow.

Side	Description	Boundary condition
AB	$\frac{\partial \mu}{\partial \eta} = 0$	$\frac{\partial \mu}{\partial x} = 0, \frac{\partial \mu}{\partial y} = 0$
BC	Streamlined	0
AED	Streamlined	0
CD	Prescribed	0

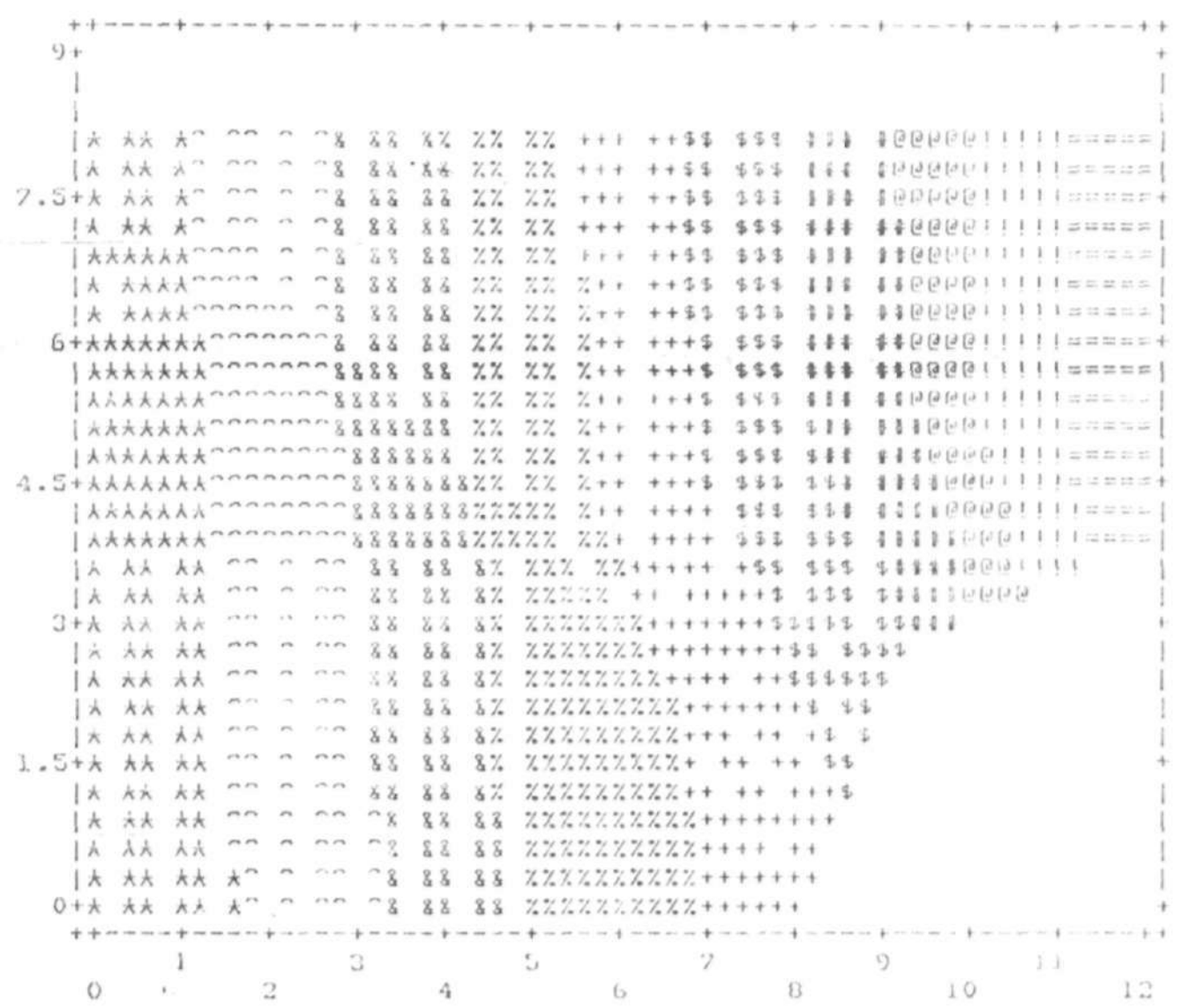
The values of the velocity potential function and stream function are given at their various nodal points of 13 triangular elements.

Table 5.6 The stream function and velocity potential distribution for 13 nodes of 13 elements

Node	Coordinate	Stream Function	Velocity Potential
1	0,8	4	14.9
2	5,8	4	9.675
3	9.17,8	4	4.482
4	12,8	4	0
5	0,4	2	15.044
6	5,4	1.741	10.011
7	9.17,5.5	2.042	4.784
8	12,5.5	1.673	0
9	0,0	0	15.231
10	5,0	0	10.524
11	8,0	0	8.469
12	9.17,2.83	0	6.229
13	12,4	0	0

Table 5.7 The global node for the 13 elements.

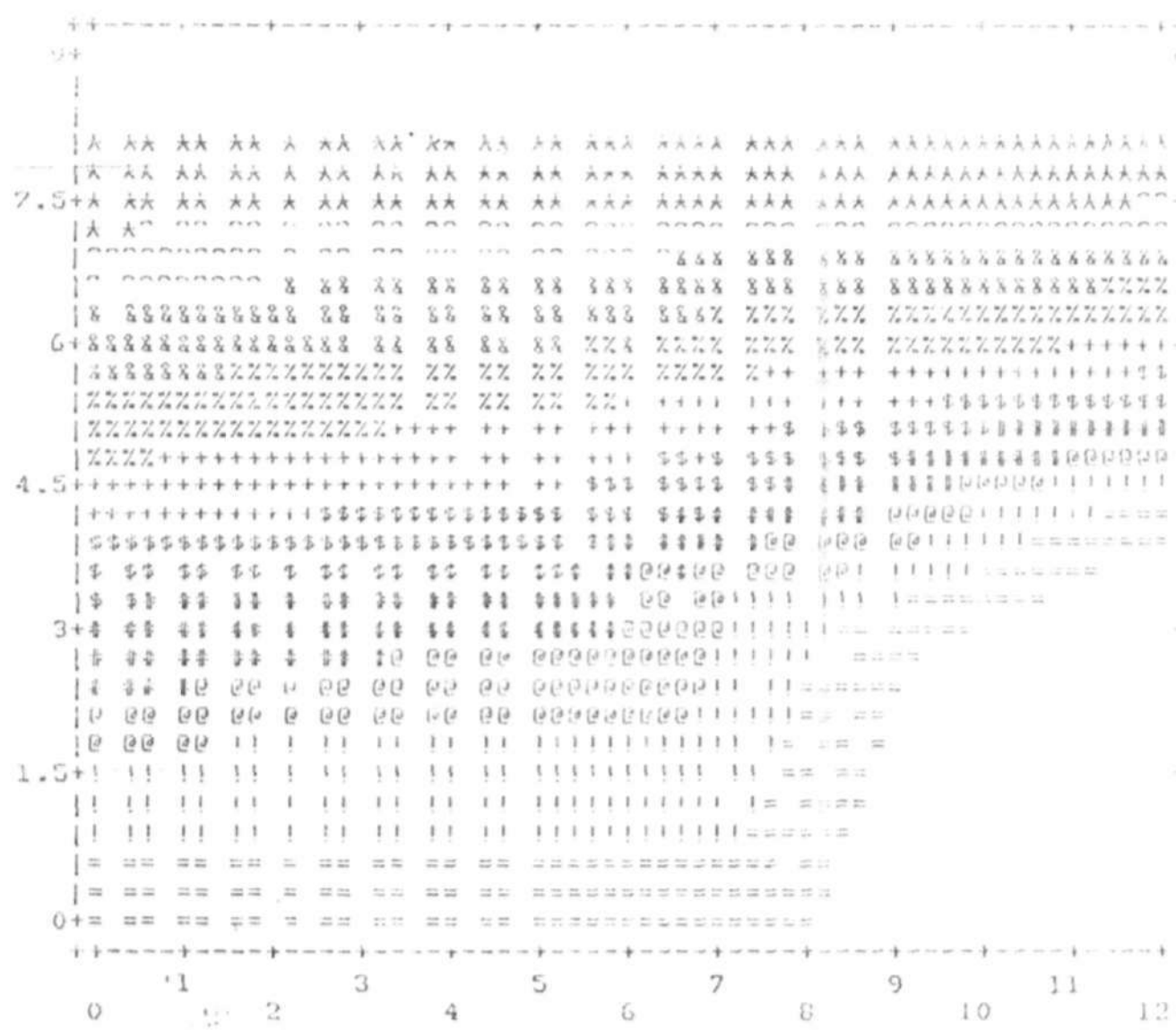
Element	Nodes
1	5 1 6
2	6 1 2
3	6 2 7
4	7 2 3
5	7 3 4
6	7 4 8
7	5 6 9
8	9 6 10
9	10 6 11
10	11 6 12
11	6 7 12
13	7 8 13



The bounds of Stream Function are defined as follows:

-:	.000-	1.523	!:	1.523-	3.046	@:	3.046- 4.569
#:	4.569-	6.092	$:	6.092-	7.615	+:	7.615- 9.139
%:	9.139-	10.662	&:	10.662-	12.185	^:	12.185- 13.708
A:	13.708-	15.231					

Fig.5.5: The Distribution of the Stream Function for 400 Elements



The bounds of Velocity Potential are defined as follows:

=:	.000-	.400	!:	.400-	.800	@:	.800-	1.200
#:	1.200-	1.600	$:	1.600-	2.000	+:	2.000-	2.400
%:	2.400-	2.800	&:	2.800-	3.200	^:	3.200-	3.600
A:	3.600-	4.000						

Fig. 5.5b: The Distribution of the Velocity Potential for 400 Elements.

Table 5.8: Stream Function and Velocity Potential at Various Nodes for 300 Elements

Nodal Number	Coordinate X	Y	Stream Function	Velocity Potential
1	0.000	8.000	4.0000	14.9000
2	5.000	8.000	4.0000	9.6750
3	9.170	8.000	4.0000	4.4530
4	12.000	8.000	4.0000	0.0000
5	0.000	4.000	2.0000	15.0440
6	5.000	4.000	1.7410	10.0110
7	9.170	5.500	2.0420	4.7840
8	12.000	5.500	1.6730	0.0000
9	0.000	0.000	0.0000	15.2310
10	5.000	0.000	0.0000	10.5340
11	8.000	0.000	0.0000	8.4690
12	9.170	2.830	0.0000	6.2290
13	12.000	4.000	0.0000	0.0000
14	0.000	6.000	3.0000	14.9720
15	2.500	6.000	2.8705	13.4555
16	2.500	4.000	1.8705	12.5275
17	2.500	8.000	4.0000	12.2875
18	5.000	6.000	2.8705	9.8430
19	7.085	6.750	3.0210	7.2295
20	7.085	4.750	1.8915	7.3975
21	7.085	8.000	4.0000	7.0785
22	9.170	6.750	3.0210	4.6330
23	10.585	8.000	4.0000	2.2410
24	10.585	6.750	3.0210	2.3920
25	12.000	6.750	2.8365	0.0000
26	10.585	5.500	1.8575	2.3920
27	2.500	2.000	0.8705	13.6210
28	0.000	2.000	1.0000	15.1375
29	5.000	2.000	0.8705	10.2675
30	2.500	0.000	0.0000	12.8775
31	6.500	2.000	0.8705	9.2400
32	6.500	0.000	0.0000	9.4965
33	7.085	3.415	0.8705	8.1200
34	8.585	1.415	0.0000	7.9490
35	9.170	4.165	1.0210	5.5065
36	10.585	3.415	0.0000	3.1145
37	10.585	4.750	1.0210	2.3920
38	12.000	4.750	0.8365	0.0000
39	2.500	5.000	2.3705	12.4915
40	3.750	5.000	2.3058	11.2332
41	3.750	4.000	1.8058	11.2692
42	1.250	6.000	2.9353	13.7137
43	1.250	5.000	2.4353	13.7498
44	0.000	5.000	2.5000	15.0000
45	1.250	4.000	1.9353	13.7858

Table 5.8: Continued

Nodal Number	Coordinate X	Y	Stream Function	Velocity Potential
46	0.000	7.000	3.5000	14.9360
47	1.250	7.000	3.4353	13.6777
48	3.750	7.000	3.4353	11.0653
49	3.750	8.000	4.0000	10.9813
50	5.000	7.000	3.4353	9.7590
51	2.500	7.000	3.4353	12.3715
52	3.750	6.000	2.8705	11.1493
53	5.000	5.000	2.3058	9.9270
54	1.250	8.000	4.0000	13.5938
55	7.085	5.750	2.4563	7.3135
56	8.127	6.125	2.5315	6.0067
57	8.127	5.125	1.9668	6.0908
58	6.043	6.375	2.9457	8.5363
59	6.043	5.375	2.3810	8.6202
60	6.043	4.375	1.8163	8.7043
61	6.043	7.375	3.5105	8.4522
62	8.127	7.375	3.5105	5.8558
63	8.127	8.000	4.0000	5.7803
64	9.170	7.375	3.5105	4.5575
65	7.085	7.375	3.5105	7.1540
66	8.127	6.750	3.0210	5.9312
67	9.170	6.125	2.5315	4.7085
68	6.043	8.000	4.0000	8.3768
69	10.585	7.375	3.5105	2.3165
70	11.292	8.000	4.0000	1.1205
71	11.292	7.375	3.5105	1.1960
72	9.877	7.375	3.5105	3.4370
73	9.877	6.750	3.0210	3.5125
74	9.877	6.125	2.5315	3.5880
75	9.877	8.000	4.0000	3.3615
76	11.292	6.125	2.3470	1.1960
77	12.000	6.125	2.2547	0.0000
78	11.292	5.500	1.7653	1.1960
79	11.292	6.750	2.9287	1.1960
80	10.585	6.125	2.4392	2.3920
81	9.877	5.500	1.9497	3.5880
82	12.000	7.375	3.4183	0.0000
83	1.250	2.000	0.9353	13.8793
84	1.250	1.000	0.4353	13.9260
85	0.000	1.000	0.5000	15.1842
86	2.500	3.000	1.3705	12.5743
87	1.250	3.000	1.4353	13.8325
88	0.000	3.000	1.5000	15.0907
89	3.750	3.000	1.3058	11.3160
90	3.750	1.000	0.4353	11.5725

Table 5.8: Continued

Nodal Number	Coordinate X	Y	Stream Function	Velocity Potential
91	5.000	1.000		
92	3.750	0.000		
93	3.750	3.000		
94	2.500	1.000		
95	1.250	0.000		
96	5.000	3.000		
97	6.500	1.000		
98	7.250	1.000		
99	7.250	0.000		
100	5.750	2.000		
101	5.750	1.000		
102	5.750	0.000		
103	5.750	3.000		
104	7.835	2.415		
105	8.127	3.122		
106	8.877	2.122		
107	6.793	2.707		
108	7.543	1.707		
109	8.292	0.707		
110	6.043	3.707		
111	8.127	3.729		
112	9.170	3.492		
113	8.127	4.457		
114	7.085	4.082		
115	9.170	4.832		
116	10.585	4.082		
117	11.292	3.707		
118	11.292	4.375		
119	9.877	4.750		
120	9.877	3.457		
121	9.877	5.125		
122	9.877	2.122		
123	11.292	2.750		
124	12.000	3.375		
125	11.292	5.135	0.5105	4.3105
126	10.585	5.125	1.0210	3.9495
127	12.000	5.125	1.5315	3.5880
128	3.750	4.500	0.0000	4.6715
129	4.375	4.500	0.9288	1.1960
130	4.375	4.000	0.4182	0.0000
131	3.125	5.000	1.3420	1.1960
132	3.125	4.500		2.3920
133	2.500	4.500		
134	3.125	4.000		
135	2.500	5.500		

Table 5.8: Continued

Node No.	Coordinate X	Coordinate Y	Stream Function	Velocity Potential
136	3.125	5.500	2.5881	11.8444
137	1.875	5.000	2.4029	13.1206
138	1.875	4.500	2.1529	13.3280
139	1.875	5.500	2.6529	13.1226
140	1.250	5.500	2.6058	13.7318
141	0.625	6.000	2.9676	14.3439
142	0.625	5.500	2.7176	14.3609
143	1.875	5.000	2.9029	13.6846
144	1.250	4.500	2.1058	13.7079
145	1.875	4.000	1.9029	13.1586
146	0.625	5.000	2.4676	14.3789
147	0.625	4.500	2.2176	14.3969
148	0.000	4.500	2.2500	15.0203
149	0.625	4.000	1.9676	14.4139
150	0.000	5.500	2.7500	14.9900
151	1.250	6.500	3.1058	13.6058
152	1.875	6.500	3.1529	12.0666
153	0.625	7.000	3.4676	14.3069
154	0.000	6.500	3.2176	14.3349
155	0.000	6.500	3.2500	14.9340
156	0.000	7.500	3.7500	14.9100
157	0.625	7.500	3.7176	14.3969
158	4.375	7.500	3.7176	10.3701
159	4.375	8.000	4.0000	10.5281
160	5.000	7.500	3.7176	9.7170
161	5.250	7.500	3.7176	11.0833
162	4.375	7.000	3.4353	10.4121
163	4.375	6.500	3.1529	10.4041
164	5.000	6.500	3.1529	9.8010
165	5.125	7.500	3.7176	11.6764
166	5.125	8.000	4.0000	11.8844
167	5.750	6.500	3.1529	11.1073
168	4.375	6.000	2.8705	10.4913
169	3.750	7.000	3.4353	11.7184
170	4.375	6.500	3.1529	11.7166
171	3.500	6.500	3.1529	12.4139
172	3.750	6.000	2.8705	11.8054
173	3.500	7.500	3.7176	12.8295
174	4.375	5.500	2.5881	10.5881
175	5.000	5.500	2.5881	9.8850
176	5.250	5.500	2.5881	11.4913
177	4.375	5.000	2.3058	10.8801
178	3.750	4.500	2.0234	9.9690
179	1.875	7.500	3.7176	13.9826
180	1.875	8.000	4.0000	13.9406
181	1.250	7.500	3.7176	13.6857
182	1.875	7.000	3.4353	13.0246

5.3 <u>One Dimensional Fluid Flow between Flat permeable plates</u>

Since the generalized functional in two variables (equation 2.1) can be reduced to one dimensional problem, we also applied fluid flow between flat permeable plates to the Finite Element Method so designed. For the above problem the continuity and momentum equations are:

$$\frac{\partial \mu}{\partial x} + \frac{\partial v}{\partial y} = 0$$

$$P\mu \frac{\partial \mu}{\partial x} + Pv \frac{\partial \mu}{\partial x} = \frac{\partial P}{\partial x} + \mu \left(\frac{\partial^2 \mu}{\partial x^2} + \frac{\partial^2 \mu}{\partial y^2} \right)$$

with boundary conditions

$u = U = 0.1m, \quad u = 0, \quad U = v$

$y = u = 0.03m, \quad y=0;$

pressure gradient

viscosity $\mu = 2.0 \times 10$ Ns/m

density $\beta = 1.0$ g/m

fluid Velocity $U = 0.1$m/s

$$\therefore \quad \frac{\partial \mu}{\partial x} = 0 \qquad \text{continuity equation.}$$

Such that momentum becomes

$$-Pu \frac{\partial \mu}{\partial y} = -\frac{\partial P}{\partial x} + \mu \frac{\partial^2 \mu}{\partial y^2}$$

Table 5.8a: solution for 3 Elements

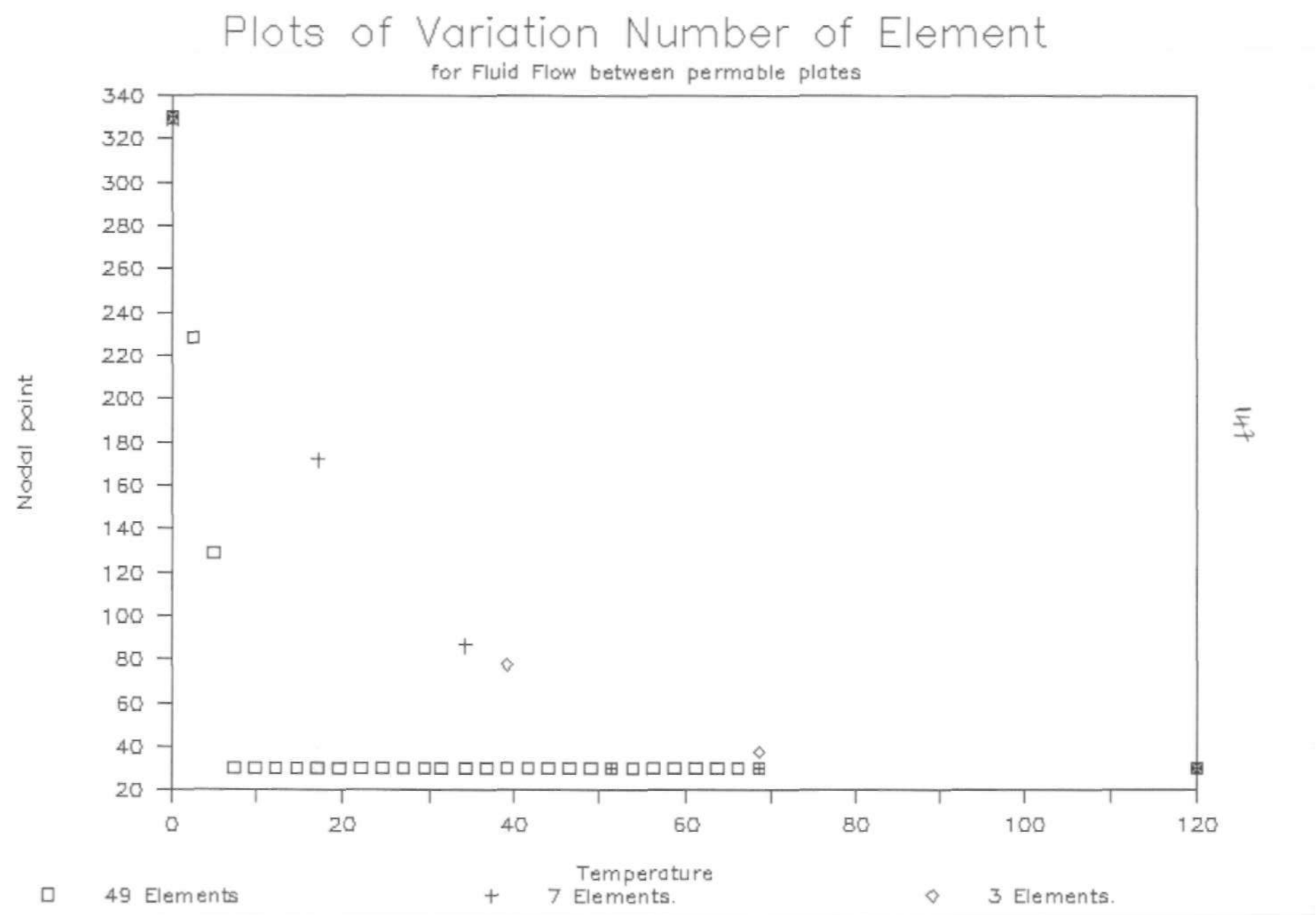
Nodal Number	Nodal Coordinate-X	Nodal Temperature
1	0	330
2	40	77.54057
3	80	37.35135
4	120	30

Solution for 7 Elements

Nodal Number	Nodal Coordinate-X	Nodal Temperature
1	0	330
2	17.14286	172.0583
3	34.28571	86.59275
4	51.42857	30
5	68.57143	30
6	85.71428	30
7	102.8571	30
8	120	30

Table 5.8b: Solution for 49 Elements

Nodal Number	Nodal Coordinate-x	Nodal Temperature
1	0	350
2	2.44898	228.4123
3	4.897959	128.7717
4	7.346939	30
5	9.795919	30.00001
6	12.2449	30.00002
7	14.69388	30.00003
8	17.14286	30.00003
9	19.59184	30.00004
10	22.04082	30.00005
11	24.4898	30.00005
12	26.93878	30.00005
13	29.38776	30.00005
14	31.83674	30.00005
15	34.28572	30.00005
16	36.7347	30.00005
17	39.18368	30.00005
18	41.63265	30.00005
19	44.08163	30.00005
20	46.53061	30.00005
21	48.97959	30.00005
22	51.42857	30.00004
23	53.87755	30.00004
24	56.32653	30.00004
25	58.7755	30.00004
26	61.22448	30.00004
27	63.67346	30.00004
28	66.12244	30.00004
29	68.57142	30.00004


Plots of Variation Number of Element
for Fluid Flow between permable plates
Nodal point
340
320
300
280
260
240
220
200
180
160
140
120
100
80
60
40
20
0
20
40
60
80
100
120
Temperature
49 Elements
7 Elements.
3 Elements.

where

$Kx = \mu$, $P(x) = P\mu$ and $Q(x) = -\frac{\partial P}{\partial x}$

The solution output for various number of element are shown in Table 5.8a and 5.8b, while Fig 5.6 shows the various number of elements for fluid flow between permeable plates.

5.4 Finite Element Problem on Hydrostatic Pad

In this problem a symmetric cross section of the hydrostatic pad was considered. Because of this symmetry only 1/8th of the pad was considered (Fig. 5.7). The area was initially divided into eight element with nine nodes.

The pressure under the pad is governed by the Reynolds equation.

$$\frac{\partial}{\partial x}(h^3\frac{\partial P}{\partial x}) + \frac{\partial}{\partial y}(h^3\frac{\partial P}{\partial y}) = 0$$

Such that when compared with the generalized equation 2.3, $K_x = h^3$, $K_y = h^3$ $P(x,y) = 0$ and $Q(x,y) = 0$. Nodal values are specified along the inner and outer edges such that the pressure under the pad is $P = 20$ on $y = 12.0$ and $P = 0$ on $y = 36$. Other nodes at the boundary are governed by the boundary condition:

$$K_x\frac{\partial P}{\partial x} - K_y\frac{\partial P}{\partial y} + \alpha(x,y)P + \beta(x,y) = 0$$

where $\alpha = \beta = 0$

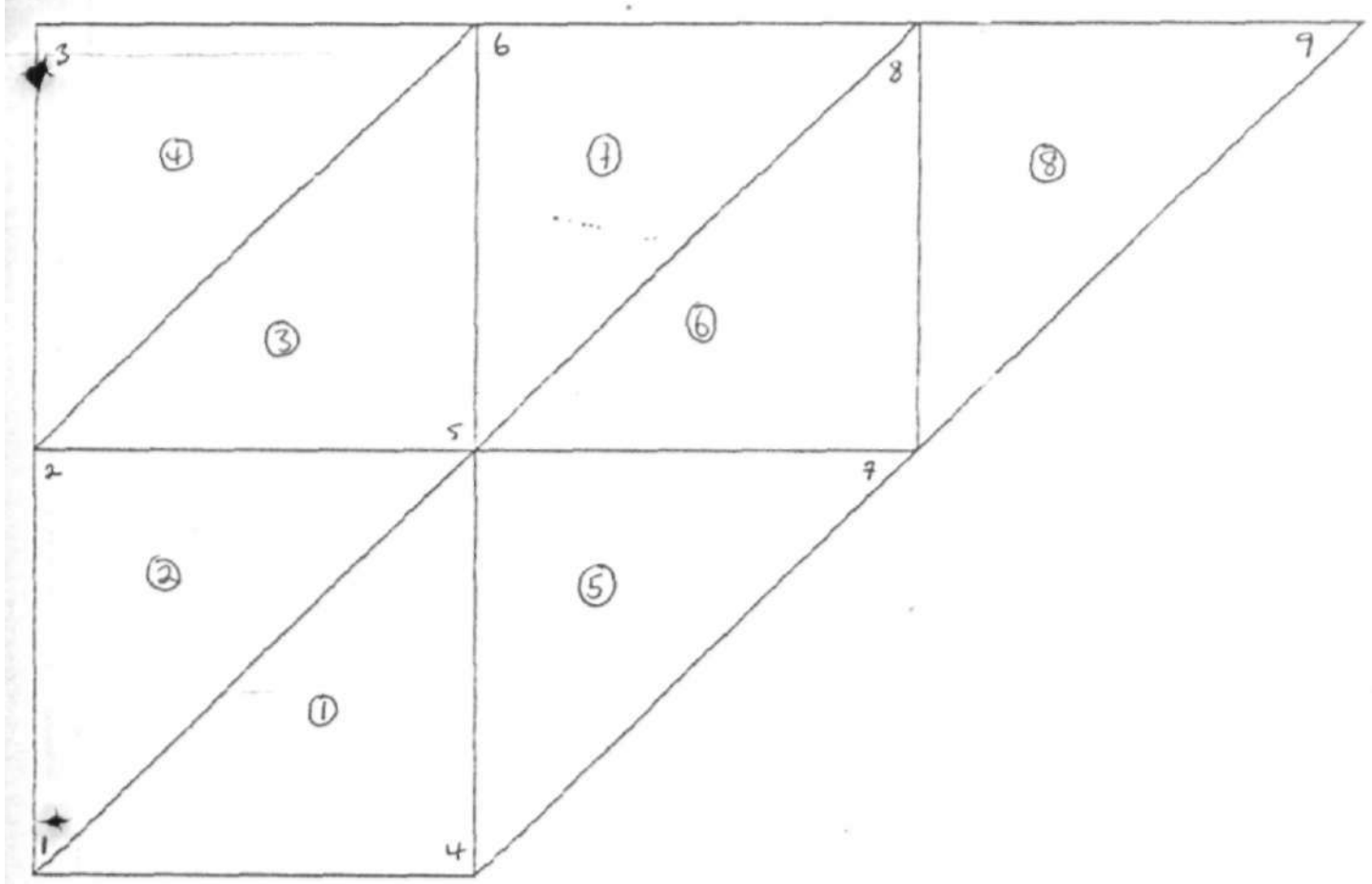
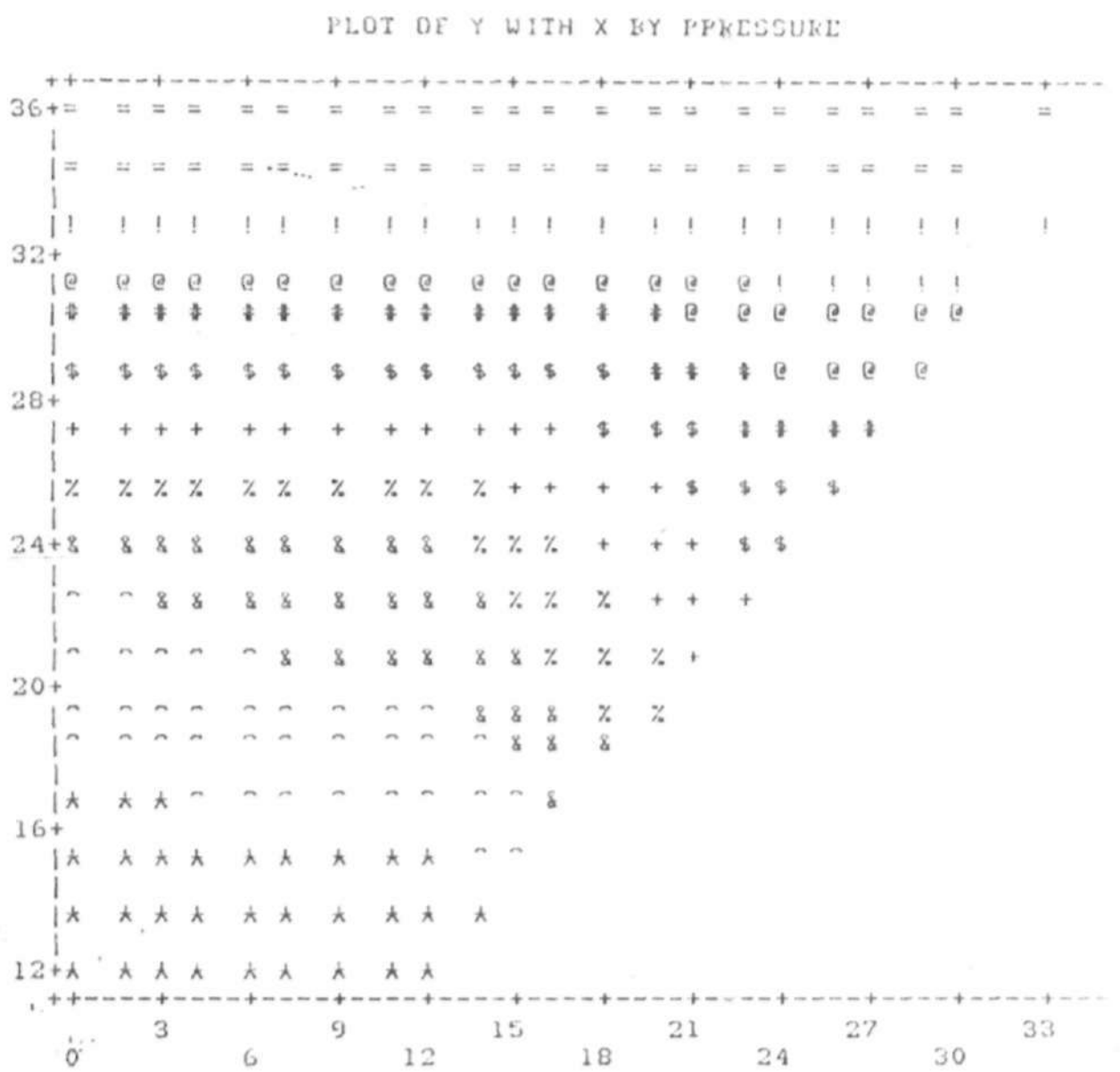


Fig. 5.7: The Geometry of the 1/8th of the Hydroslatic Pad Proble

The result for the nine node are given on Table 5.9, and Fig 5.8 shows the pressure distribution of 500 elements of the hydrostatic pad.

Table 5.9 The pressure distribution for the hydrostatic pad problem.

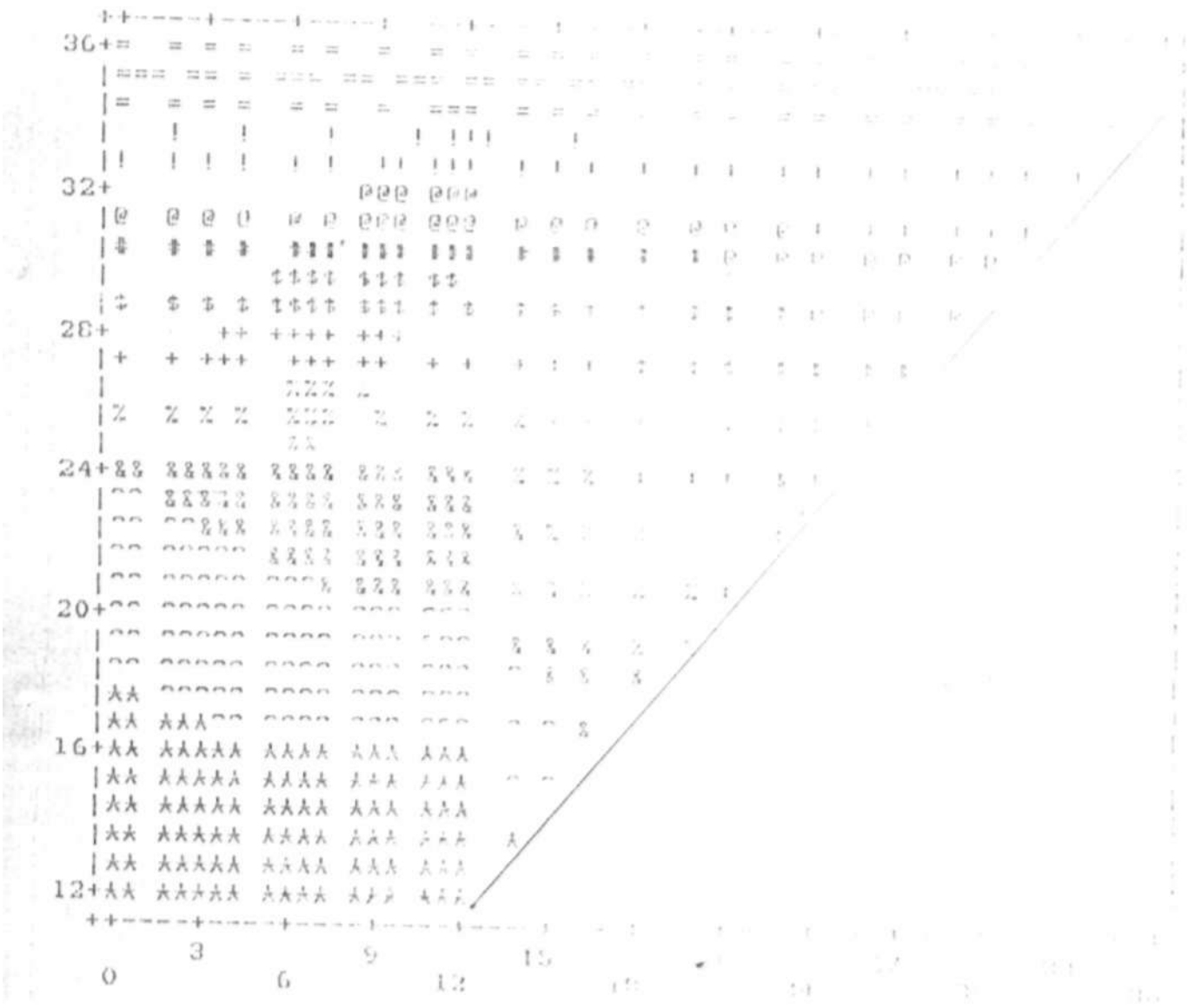
Node	Coordinates	Pressure
1	0,12	20.000
2	0,24	15.855
3	0,36	0.000
4	12,12	20.000
5	12,24	14.331
6	12,36	0.000
7	24,24	9.2762
8	24,36	0.000
9	36,36	0.000



The bounds of Pressure are defined as follows:

=:	.000-	2.000	!:	2.000-	4.000	@:	4.000-	6.000
#:	6.000-	8.000	$:	8.000-	10.000	+:	10.000-	12.000
%:	12.000-	14.000	&:	14.000-	16.000	^:	16.000-	18.000
*:	18.000-	20.000						

Fig. 5.8: The Pressure Distribution for 500 Elemen



The bounds of Pressure are defined as follows:

=:	.000-	2.000	::	[illegible]	[illegible]
#:	6.000 -	8.000	%:	[illegible]	[illegible]
%:	12.000	14.000	&:	[illegible]	[illegible]
*:	18.000-	20.000			

Fig. 5.8b: The Pressure Distribution for 1400 Elements

Table 5.10: Pressure Distribution for 300 Elemen

Nodal Number	Element Number	Coordinates X	Coordinates Y	Pressure
1	208	0.000	12.000	20.0000
2	76	0.000	24.000	15.8550
3	4	0.000	36.000	0.0000
4	88	12.000	12.000	20.0000
5	5	12.000	24.000	14.3310
6	7	12.000	36.000	0.0000
7	104	24.000	24.000	9.2762
8	116	24.000	36.000	0.0000
9	128	36.000	36.000	0.0000
10	148	6.000	12.000	20.0000
11	22	12.000	18.000	17.1655
12	136	6.000	18.000	17.1655
13	200	6.000	24.000	15.0930
14	13	0.000	18.000	17.9275
15	17	12.000	30.000	7.1655
16	292	6.000	30.000	7.9275
17	296	6.000	36.000	0.0000
18	19	0.000	30.000	7.9275
19	85	18.000	18.000	14.6381
20	97	18.000	24.000	11.8036
21	118	24.000	30.000	4.6381
22	94	18.000	30.000	7.1655
23	29	18.000	36.000	0.0000
24	125	30.000	30.000	4.6381
25	122	30.000	36.000	0.0000
26	38	9.000	18.000	17.1655
27	35	12.000	21.000	15.7483
28	221	9.000	21.000	15.7483
29	42	9.000	15.000	18.5828
30	158	6.000	15.000	18.5828
31	161	3.000	12.000	20.0000
32	40	3.000	15.000	18.5828
33	175	9.000	12.000	20.0000
34	87	12.000	15.000	18.5828
35	185	3.000	21.000	16.5103
36	182	3.000	24.000	15.4740
37	45	0.000	21.000	16.8913
38	193	6.000	21.000	16.1293
39	48	3.000	18.000	17.5465
40	52	0.000	15.000	18.9638
41	265	9.000	24.000	14.7130
42	239	9.000	30.000	7.5465
43	7	12.000	33.000	3.5828

Table 5.10: Continued

Nodal Number	Element Number	Coordinates X	Coordinates Y	Pressure
44	226	9.000	33.000	3.9637
45	245	9.000	27.000	11.1293
46	254	6.000	27.000	11.5103
47	63	3.000	27.000	11.8912
48	263	12.000	27.000	10.7483
49	287	3.000	33.000	3.9637
50	275	3.000	36.000	0.0000
51	274	0.000	33.000	3.9637
52	20	6.000	33.000	3.9637
53	77	3.000	30.000	7.9275
54	298	0.000	27.000	11.8912
55	78	9.000	36.000	0.0000
56	82	15.000	21.000	14.4846
57	81	15.000	24.000	13.0673
58	90	18.000	21.000	13.2209
59	87	15.000	18.000	15.9018
60	88	15.000	15.000	17.3190
61	90	21.000	21.000	11.9572
62	92	21.000	24.000	10.5399
63	93	21.000	30.000	5.9018
64	117	24.000	33.000	2.3191
65	116	21.000	33.000	3.5828
66	96	21.000	27.000	8.2208
67	25	18.000	27.000	9.4846
68	99	15.000	27.000	10.7483
69	104	24.000	27.000	6.9572
70	107	15.000	33.000	3.5828
71	7	15.000	36.000	0.0000
72	109	18.000	33.000	3.5828
73	28	15.000	30.000	7.1655
74	29	21.000	36.000	0.0000
75	119	27.000	33.000	2.3191
76	117	27.000	36.000	0.0000
77	127	30.000	33.000	2.3191
78	30	27.000	30.000	4.6381
79	125	27.000	27.000	6.9572
80	128	33.000	33.000	2.3191
81	128	33.000	36.000	0.0000
82	132	10.500	21.000	15.7483
83	131	12.000	22.500	15.0396
84	129	10.500	22.500	15.0396
85	133	10.500	19.500	16.4569
86	137	9.000	19.500	16.4569
87	143	7.500	18.000	17.1655
88	219	7.500	19.500	16.4569
89	140	10.500	18.000	17.1655
90	138	12.000	19.500	16.4569

Table 5.10: Continued

Nodal Number	Element Number	Coordinates X	Coordinates Y	Pressure
91	143	7.500	16.500	17.8741
92	155	6.000	16.500	17.8741
93	144	9.000	16.500	17.8741
94	144	7.500	15.000	18.5828
95	149	7.500	13.500	19.2914
96	41	6.000	13.500	19.2914
97	152	10.500	16.500	17.8741
98	155	4.500	15.000	18.5828
99	154	4.500	16.500	17.8741
100	157	4.500	13.500	19.2914
101	159	3.000	13.500	19.2914
102	159	1.500	12.000	20.0000
103	159	1.500	13.500	19.2914
104	164	4.500	12.000	20.0000
105	170	10.500	15.000	18.5828
106	165	12.000	16.500	17.8741
107	168	10.500	13.500	19.2914
108	42	9.000	13.500	19.2914
109	173	7.500	12.000	20.0000
110	174	10.500	12.000	20.0000
111	176	12.000	13.500	19.2914
112	45	1.500	22.500	16.1826
113	255	1.500	24.000	15.6645
114	2	0.000	22.500	16.3731
115	181	3.000	22.500	15.9921
116	181	1.500	21.000	16.7008
117	189	1.500	19.500	17.2189
118	46	0.000	19.500	17.4094
119	186	4.500	22.500	15.8016
120	258	4.500	24.000	15.2835
121	192	3.000	19.500	17.0284
122	189	1.500	18.000	17.7370
123	199	4.500	21.000	16.3198
124	49	4.500	19.500	16.8379
125	195	6.000	19.500	16.6474
126	49	4.500	18.000	17.3560
127	214	6.000	22.500	15.6111
128	201	1.500	16.500	18.2551
129	202	0.000	16.500	18.4456
130	206	3.000	16.500	18.0646
131	207	1.500	15.000	18.7733
132	52	0.000	13.500	19.4819
133	54	7.500	22.500	15.4206
134	14	7.500	24.000	14.9025
135	216	9.000	22.500	15.2301
136	54	7.500	21.000	15.9388
137	271	10.500	24.000	14.5215

Table 5.10: Continued

Nodal Number	Element Number	Coordinates X	Y	Pressure
138	228	10.500	33.000	3.7733
139	227	12.000	34.500	1.7914
140	3	10.500	34.500	1.9819
141	230	10.500	31.500	5.5646
142	229	9.000	31.500	5.7551
143	237	7.500	30.000	7.7370
144	231	7.500	31.500	5.9456
145	234	10.500	30.000	7.3560
146	234	12.000	31.500	5.3741
147	60	7.500	28.500	9.5284
148	16	6.000	28.500	9.7189
149	241	9.000	28.500	9.3379
150	61	7.500	27.000	11.3198
151	268	7.500	25.500	13.1111
152	244	6.000	25.500	13.3016
153	248	10.500	28.500	9.1474
154	63	4.500	27.000	11.7008
155	249	4.500	30.500	8.2811
156	253	4.500	25.500	13.4921
157	64	3.000	25.500	13.6826
158	255	1.500	25.500	13.8731
159	261	10.500	27.000	10.9388
160	17	12.000	28.500	8.9569
161	271	10.500	25.500	12.7301
162	67	9.000	25.500	12.9206
163	272	12.000	25.500	12.5396
164	273	1.500	34.500	1.9819
165	273	1.500	36.000	0.0000
166	4	0.000	34.500	1.9819
167	283	3.000	34.500	1.9819
168	281	1.500	33.000	3.9637
169	281	1.500	31.500	5.9456
170	279	0.000	31.500	5.9456
171	74	4.500	34.500	1.9819
172	71	4.500	36.000	0.0000
173	286	3.000	31.500	5.9456
174	19	1.500	30.000	7.9275
175	289	4.500	33.000	3.9637
176	72	4.500	31.500	5.9456
177	291	6.000	31.500	5.9456
178	291	4.500	30.000	7.9275
179	294	6.000	34.500	1.9819
180	298	1.500	28.500	9.9094
181	19	0.000	28.500	9.9094
182	302	3.000	28.500	9.9094
183	75	1.500	27.000	11.8912

5.5 Software Evaluation

A software development process will not come to a good conclusion without evaluating the work done in relation with existing ones in order to justify the performance of the new product. In most software development process, which involve numerical computation, attention were mostly paid to the accuracy of the numerical approach, without putting into consideration the programming aspects. Many work have been done in which the accuracy of the work had been extensively analyzed in isolation of the programming effort that brought about the level of accuracy, the programming environment and may be, the complexity of the programming.

In this project we have placed Finite Element Method in the forefront of the study followed by the programming methodologies as applied to some class of partial differential equation. Various programming approaches had been considered, some implementation had been taken as BenchMark Programs to enable a comparative study of the SOLVER approach.

Since the method were coded for different machines in different languages, it will not simply be possible to compare the performance of the various program in terms of time taken and storage consumption in the different environment. Moreover we have not only considered accuracy

as the measure of how good the package is, rather we also covered other quantities like:

(a) Efficiency,

(b) Easy understanding,

(c) Reliability and Robustness.

The accuracy of the Benchmark programs had been tested in various programming methods under consideration and found that they are all at an acceptable range. Table 5.11 shows the accuracy value for various programming methods including that of the SOLVER. Numerical Results by Pesaro et al (1978), Duchsky (1980) and Molus (1982) were considered.

Table 5.11: Velocity potential of Irrotational Flow of section 5.2, using Triangular Elements for selected nodes.

Coordinates	Pesaro (1978)	Duchsky (1980)	Molus (1982)	SOLVER
12,5.5	0.0	0.0	0.0	0.0
0,6	14.852	15.00	14.888	14.843
5,6	9.756	9.975	9.968	9.843
2.5,2	12.456	12.627	12.661	12.621
1.25,6	13.012	13.784	13.571	13.714
1.25,4	12.956	13.964	13.772	13.787
5,5	9.702	10.004	9.873	9.927
2.5,3	12.653	12.696	12.565	12.562
6.043,8	8.502	8.453	8.354	8.377
2.5,5.5	12.326	12.754	12.501	12.474

5.6 Experimental Framework

Since accuracy is not the only measure of the software performance we had looked at the various factors that affect the software performance, and designed a method of measuring them experimentally.

These performance measure or factors where chosen based on the objective of the project and they are as follows:

(i) Efficiency - This will involve the consideration of the use of computer resources by programme (economy).

(ii) Easy usage - Involves the consideration of how natural the input required for computer is, how clearly understandable and complete the output is (simplicity and structured).

(iii) Reliability and Robustness: This involves the consideration of how the program performs the intended function satisfactorily and how the performance degrades as usage moves away from intended domain.

The measure of these complexities are not straight forward, particularly for diverse study. Suitable measure will be possible if the various Benchmarks are brought to common unit. The approach used here was based on the tenet that the measured complexity of Benchmark was a sum of the complexity of its constituent elements, and these element can be quantified by the use of carefully selected weights and factors.

This method was first used by Halstad (1970) at Purdue University. He focused on the total number of operands and operators in a program as the measure of complexity, developed estimator formulas for program complexity and related his result to the number of man-hour and errors. This was based much on his mathematical development which was intuitively motivated.

Laemnel et al (1977) subsequently showed that Halstad's work was related to more basic ideas rooted in statistical communication theory, generally known as Zipf's law of natural language and Shannon's information theory.

Performance evaluation of competive software had been made by Ferrari (1978). In his approach instruction mixes (grouping of sets of instruction in the assembling language level) was devised.

The first work, by Pesaro et al (1978) had used the program count technique to assess software maintainability in attempt to quantify program difficulty. He implemented his ideas by assigning weights and factors to the program elements. As with all weight the value are tuned so that the results correlates with reality or a standard that represent reality.

The other two benchmarks under study, Duchsky (1980) and Molus (1982) adopted the method by Pesaro et al (1978) in their measurement of performance. We have also adopted this method to give the profile and assessment of the various complexities, since the software we are assessing are from different environment (software Hardware, and operating systems).

The BenchMarks are:

FEM 1 - ALGOL programming language Pesaro et al (1978)

FEM 2 - FORTRAN programming language Duchsky (1980)

FEM 3 - FORTRAN programming language Molus (1982)

SOLVER - FORTRAN/PASCAL Assembler programming.

The FEM1, FEM2, FEM3 did not pay attention to programming environment in which the Pascal and Assembler programming

languages were used in the SOLVER. These programming environment includes the tools that made the package self-contained with various sheets that facilitates programming with respect to screen management.

5.6.1. Programming Counting Tools

A program counting tool, Berns (1984), was used to extract the token (symbolic name, constant, operators and statement elements) and given their various frequencies. It will also group symbols according to the data type to which they belong as well as their common attributes. This will help in assigning factors to the elements of these data types and attributes.

We pragmatically rely on the approach by Pesaro (1978) as did by other BenchMarks under consideration for uniformity of the measurement of performance.

In this work element weights were in the range 0 - 30. The values for the various weights and factors are given in Table 5.12 to Table 5.19. Table 5.12 shows the weights that are assigned to the various kinds of symbolic names.

Table 5.12 Weights Assign to Symbolic Name Elements

Symbolic Name Elements	Weights
Parameter	1.00
Variable	0.85
Array	1.50
Function	2.00
Statement Function	1.80
Subroutine	2.00
Common	0.00
Label	1.00

Variables arrays and statement functions are assigned the basic weights of Table 5.12 in every non-specification statement in which they appear.

The same is applicable to every reference to functions and subroutines. Functions that appear in parameter expressions were also weighted. Parameters receive weights in every statement in which they were used. The weights assigned to constant elements are shown in Table 5.13. Constants received their specific weights in every executable statement they appeared. They also received their weights in specification statement when they represent values that initialize symbolic names (eg. DATA statement).

Table 5.13 Weights Assign to Constant Elements

Constant Elements	Weights
Logical	0.00
Integer	0.00
Real (all methods)	0.05
Hexadecimal	0.10
Octal	0.10
Character	0.10
Hollerith	0.10

Operator elements were assigned weights specified in Table 5.14 in every statement in which they appeared. Weights were assigned to a symbol when they were used as operators.

Table 5.14 Weights Assigned to Operator Elements

Operator Element	Weights
.OR;.AND.	0.25
.NOT.	1.00
Relational	0.20
+,-,//	0.20
/	0.30
*	0.25
**	0.50

The weights assigned to statement type are given in table 5.15 and table 5.16, by type and assigned weights respectively.

Table 5.15: Weights Assigned to Statement Type Elements

Statement Type	Weights
ASSIGNMENT	0.0
ACCEPT	3.0
ASSIGN	2.0
BLOCK DATA	1.0
BYTE	0.0
CALL	1.0
CHARACTER	0.0
CLOSE	0.1
COMMON	0.0
COMPLEX	0.0
CONTINUE	0.0
DATA	0.0
DIMENSION	0.0
DO	7.0
DOUBLE PRECISION	0.0
DO WHILE	7.0
ELSE	0.0
ELSE IF	4.0
ENCODE	3.0
END	0.1

Table 5.15: Continued Weights Assigned to Statement Type Elements

Statement Type	Weights
END DO	0.0
END FILE	1.0
END IF	0.0
EQUIVALENCE	0.0
EXTERNAL	0.0
FORMAT	0.1
FUNCTION	4.0
GOTO	10.0
IF	5.0
IMPLICIT	0.0
INCLUDE	0.0
INQUIRE	2.0
INTEGER	0.0
LOGICAL	0.0
OPEN	2.0
PARAMETER	0.0
PAUSE	0.1
PRINT	3.0
PROGRAM	0.0
READ	3.0
REAL	0.0
RETURN	0.1 for first RETURN 12.1 for other RETURNS
REWRITE	3.0

Table 5.15: Continued Weights Assigned to Statement Type Elements

Statement Type	Weights
REWIND	3.0
SAVE	3.0
STATEMENT FUNCTIONS	10.0
STOP	0.0
SUBROUTINE	4.0
TYPE	3.0
UNLOCK	4.0
WRITE	3.0

*The statement types are from American National Standard Programming Language, X3, 9-1978 of American National Standard Institute (ANSI), New York 1978.

Out of all the statements assigned 0 weights only the assignment statement is executable, others are specification and non-executable statements. All other statements are given progressively higher weights, culminating in a weight of 12.10 assigned to each return statement after the first in a program. The relative heavy weights assigned to return statement reflects the impact their use has on the idea of single entry, single-exit structured programming.

Table 5.16 Statement Types by Assigned Weights

Weights	Statement Types
12.10	RETURN (except the first Return statement)
10.0	GOTO, ENTRY, SAVE
7.0	DO, DO WHILE
5.0	IF
4.0	BACKSPACE, ELSEIF, FUNCTION, SUBROUTINE
3.0	ACCEPT, PRINT, READ,REWRITE,TYPE, WRITE
2.0	ASSIGN,INCLUDE OPEN
1.0	BLOCK,DATA,CALL,ENDFILE,REWIND
0.1	CLOSE,END,FORMAT,PAUSE,RETURN(First only),STOP
0.0	ASSIGNMENT, BYTE, CHARACTER, COMMON COMPLEX, CONTINUE, DATA,DIMENSION, DOUBLE PRECISION,ELSE,END DO, END IF, EQUIVALENCE, EXTERNAL, IMPLICIT, INTEGER, LOGICAL, PARAMETER,PROGRAM,REAL, Statement functions.

Table 5.17 summarizes the weights assigned to other program elements such as dummy argument and "built-in" function.

Table 5.17 Weights Assigned to other program Elements

Element		Weights
Use of dummy argument as the name of a function, subroutines, or entry point is assessed an additional weight	-	4.00
Reference to a "built in" function (eg. % LOC)	-	15.00
Poor usage	-	30.00
Label definition that is not higher in value than the previous label defined	-	0.50 (up to a maximum 75.00)
Interface exception	-	30.00

Factors are assigned to increase or decrease the weights assigned to symbolic (Table 5.12) name elements according to the attribute and interrelationships of the symbolic name. Factors assigned to data type are given in table 5.18, while Table 5.19 gave the factors for other attributes.

Table 5.18 Factors for Data type of Symbolic Name Element.

Data Type of Symbolic Name Element	Factor
LOGICAL *1	0.10
LOGICAL *2	0.15
LOGICAL *4	0
LOGICAL *2	0.15
INTEGER *4	0.00
REAL *4	0.05
REAL *8 (same as double precision)	0.00
COMPLEX *8	0.20
COMPLEX *16	0.25

Table 5.19 Factors for other Attributes

Attribute	Factor
(i) Changing a Value	
Applied whenever a statement causes the value associated with symbolic name to change (eg. Assignment, Read, Data Statement).	- 0.15
(ii) Dummy Argument	
The Symbolic name is a dummy argument if it appears in a subroutine or function statement	- 0.30
(iii) Call Argument	
Applied to a symbolic name if it is used as an argument in a call statement	- 0.30
(iv) Implicit	
Applied to a symbolic name if it is implicitly defined	- 0.50

Table 5.19 Continued Factors for other Attributes

Attribute	Factor
(v) Common	
Applied to a symbolic name if it is defined or equivalence into a common block	- 1.00
(vi) Equivalence	
Applied to a symbolic name once for every explicit and implicit association.	- 0.5
(vii) Backward Reference	
Applied to a reference to a table that is defined earlier in the program (unless the table identified a format statement)	- 3.00

The factors as determined by equivalence attributes can be implicit as well as explicit,
eg.

```
REAL *4,A,B,C,D(2)
COMMON A,B,C
EQUIVALENCE (B,D(1))
```

A is not associated, B and C are associated explicitly while D is associated explicitly with B and implicitly with C (i.e. D(1) with B explicitly and D(2) with C implicitly).

The backwards reference which have the highest weighing factor, is to weight down programs with excessive GOTO statement (non-structured programming).

5.6.2. Measurement of Program Efficiency

Program efficiency refers to the use of Computer resources by the program. These resources are principally time and memory.

Assumption:

It was assumed that the time for access of any memory location is the same, such that the effect of COMMON and EQUIVALENCE statements as attribute on variables will have no effect on the execution time.

We also assumed that statement flow is sequential and that all conditional branches will remain the same for all the Benchmarks under comparison.

Method:

(i) The weight of each statement is calculated by finding the product of the weight of each token and multiplying by its factor.

(ii) The individual element weight are calculated: symbolic name, constants, operation, statement types and other program elements (like using dummy argument as name of function or subroutine; referencing built-in function;

backwards table definition; and interface exception).

(iii) Symbolic name weight are increased by multipling the weight by (1 + factor) for each attribute (Tables 5.18 and 5.19). The factors are used to increase/decrease the weight assigned to symbolic names (Table 5.12) according to the attribute interrelationship of the symbolic name. They will bring out the natural complexity of each statement.

(iv) The statement weight is the sum of the weights of the tokens in (ii) and (iii) above.

(v) The program or subprogram weight is the sum of the weights of each statements in it.

Generally, statement weights can be calculated with some or all of the following:

(a) W1 - a weight for the type of statement ranging from 0 to 12 (Table 15.15).

(b) W2 - for each constant element in the statement, it is ranging from 0 to 0.10 (Table 15.13).

(c) W3 - for each operator element in the statement. This weight varies from 0.20 to 1.00 (Table 15.14).

(d) W4 - is the weight assigned to other special program elements such as dummy argument, and built-in functions, backwards label definition and interface exceptions; varies from 0.50 to 30.

If a variable in a statement has some other relationship with other variables, then the factor will now come in, to increase the weight due to the relationships. Each variable may have one or more of the following:-

(e) W5 - the weight for a symbolic name element (Table 5.12) the weight varies from 0 to 2.00.

(f) W6 - (1 + F1) where F1 is the factor for being LOGICAL, REAL, INTEGER or COMPLEX (Table 5.18) F1 varies from 0 to 0.25.

(g) W7 = (1 + F2) where F2 is the factor as a result of one or more of the following:-

(i) when a statement causes the content of a symbolic name to change

(ii) when the symbolic name is a dummy argument and appears in a subroutine or function statement.

(iii) when a symbolic name is used as an argument in a call statement

(iv) when a symbolic name is implicitly defined

(v) when a symbolic name is defined or equivalenced into common

(vi) when a symbolic name have an explicit or implicit alias eg. REAL *4 A,B,C,D(2)

```
COMMON A,B,C
EQUIVALENCE (B,D(1))
```

A has no alias, B and C has one alias, and D has two aliases one explicit (B) and one implicit (c).

(vii) when a reference to a label that is defined earlier in the program (except for labels that identified a FORMAT statement).

The weight of a statement can then be defined as a sum:

STATEMENT WEIGHT = W1 + W2 + W3 + W4 (W5 x W6 x W7)

for any Wr, which is applicable to the statement. Any Wr, 1<r<5, can be zero if it does not apply to the statement while Ws, s = 6 or 7, will result to 1 if it does not apply, since F1, and F2 can be zero for W6 and W7 respectively.

The program segments and subprograms we considered for counting were in line with the subprograms considered in chapter 5; these include:

1. PREPROCESSOR:

   (a) SCON - sets the control parameter

   (b) SMESH - sets up the mesh coordinates

   (c) SELEM - sets up the element data

   (d) SBAND - sets up maximum band width

   (e) SMAT - sets up the material property

   (f) SBC - sets up boundary conditions

2. PROCESSOR:

   (a) FORMKF - formulation of the stiffness matrix and load vector

   (b) APLYBC - applying the boundary conditions to the global stiffness matrix and load vector.

   (c) SOLVE - Solves the linear equations for the nodal point values.

3. POSTPROCESSOR:

(a) EVAL - evaluate and prints finite element and exact solutions

(b) ENORM - evaluate and prints the error if the exact solution exists.

(c) EXACT - calculate the exact solutions.

Table 5.20 Weight of program segments

Module Name:	Weight
PREPROCESSOR:	
SCON	1265.00
SMESH	935.30
SELEM	776.11
SBAND	501.00
SMAT	311.00
SBC	327.30
PREPROCESSOR:	
FORMKF	2241.11
APLYBC	1671.00
SOLVE	2611.70
POSTPROCESSOR:	
EVAL	1095.00
ENORM	822.50
EXACT	251.00

Total weight of SOLVER Program :

The Benchmarks under consideration did not breakdown their weights by subprogram, rather they were given as the total weight of the entire program and estimated execution time. The breakdown in comparison with the Benchmark programs is shown in Table 5.21.

Table 5.21 The comparison of the Benchmarks with the SOLVER

Method	Total weight
FEM1	18,758.00
FEM2	15,235.00
FEM3	13,667.00
SOLVER	12,808.02

CHAPTER 6

SUMMARY AND CONCLUSION

The aim of developing the programming facilities for Finite Element Problems, using the software engineering approach, is an attempt to remove the barrier placed by communication language in handling such problems.

At this stage it is quite obvious that the theoretical development of Finite Element Method had advanced tremendously with a little touch of the programming aspect of the method. We looked at the problem of the early packages like PDEL and LEAN from the classes of problems under consideration. It was found that generalization of software packages will result to wastage of time and storage. Yet an efficient program is desirable, which calls for experienced programming. The general purpose-package have to give way to the development of a programming tool which has to be used to facilitate problem solving in Finite Element Method.

Despite these efforts to develop a unified front for the Finite Element Method the problem still remained that only the experts (i.e. the systems analyst as programmers) can produce efficient and economic packages.

It was found that there was still some barrier between the Finite Element Method and the implementation and this was precisely the communication language in which the

problem solving skills of the finite element analyst does not cognitively fit with that of the computer language. Therefore, there is the need to describe in an explicit manner and in a convenient way the steps and request to realize a Finite Element Method Solver.

We found out that the choice of programming language was out of place as provision were made for adaptable tools that will assist or translate with ease the well organised Finite Element Method into computer equivalent. This had brought about the dynamic usage of electronic sheets for various components of the Finite Element method. These sheets which do not need any specialized language are interactive, such that changes in the problem can be noticed as development of solution method progresses.

Principally based on Software Engineering approach, we have been able to look from theoretical point of view the variation approaches of solving partial differential equation bearing in mind that the theory must not deviate conspicuously from the practical implementation. This approach made use of method of achieving functionals which may actually represent some physical quantities in the problem to obtain lower-order derivations. Though it is not always possible to find a variational formulation to a given differential equation, the approach was adopted as, it was a good starting point to the study of the new approach of Finite Element Method by electronic sheets.

In the general finite element procedure the nodal and element characteristics in a domain have to be efficiently arranged for convenience, reduction of storage requirement and reduction in processing time.

In section 2.3 we have to go through the theory of the variation format of the Finite Element Method, and isolated the basic elements that will assist in the structural design of the communication language. Most of the isolation of basic elements helped in producing some structured approach of the handling to Finite Element Method equations. It can be seen that the definition of the problem domain was possible together with every property. These can be seen in Pseudo code:

```
Type  CODS is structured
        X is array [1..3] of real;
        Y is array [..3] of real;  and structure;
ELEMENTS is structured
        NODE is array [1..3] of integer;
        COORDINATE is CODS
        Kx,Ky,P,Q,P are real;
        and structure;
Variables
  REGION is array [1..E] of ELEMENTS;
```

Every property of equation 2.1 to 2.3 are embedded in this single definition. Moreover, the entire region was

made flexible such that it can vary from problem to problem. For the input control data, a compact use and representation of interrelated data which reflects the physical view of discretized region removed the isolated nature of the element proportion and coordinates of the incident list.

Property defined in equations 2.5, 2.6 and 2.7 were collectively read with a structured procedure which can also be seen in Pseudo code:

```
While not (end of element) loop
 read(e);
 with region[e] do
  loop for n = 1 to 3
   read (node[n]);
  end loop;
  loop for n = 1 to 3
   with coordinate do
    read (X[n],Y[n]);
  end loop;
  read (Kx,Ky,P,Q,alf,beta);
end while;
```

which involved a direct and uniform reading of element number e, the global nodal numbers, by using NODE; the nodal coordinates and the value of the coefficient functions.

The problem of handling the scattered element stiffness matrix and load vector of equation 2.12 to 2.19 was smoothened with the centralized and natural transformation. This had shown that with the same element and its properties the local degree of freedom numbers 1,2 and 3 being under consideration on the element stiffness matrix and load vector the Pseudo code version:-

```
    loop for i = 1 to 3
with region [e] do
    loop for j = 1 to 3
            S[i,j]    =    (Kx*b[i]*b[j]    +
     Ky*c[i]*c[j])/(A*4);
            S[i,j] = S[i,j] = S[i,j]/6
            IF i = j then S[i,j] = S[i,j]/12
      end loop
   V[i] = Q*A/3.0;
end loop;
```

This had made the introduction of the natural boundary condition possible in one step and the language not far from the steps of equation 2.19. The algorithm in Pseudo code is:

```
j = 2; k = 3;
with region [e] do
 loop for n = 1 to 3
   L = ((X[n] - X [3] 2 + (Y[n] - Y [j]^2)^(1/2)
   S[n,n] = S[n,n] + 2* alf* L/6;
   S[n;j] = S[n;j] + alf *L/6;
   S[j,n] = S[n,j] { for symmetry }
   S[j,j]=S[j,j]+2*alf*L/6;
   V[n] = V[n] - beta*4/2;
   V[j] = V[j] - beta*L/2;
 end loop;
```

The advantage of a skyline or profile method of storage over the convectional square matrix of two dimensions array was adopted for the assemblage into global stiffness matrix and load vector. This method had drastically reduced the amount of computer storage wasted. The wastage with the conventional matrix operation is estimated to be N(N-B) where N is the number of nodes in the problem and B is the maximum band width of the global stiffness matrix, since the resulting matrix is always banded.

The major difference between this skyline method and the existing ones is its structured and modular approach. The advantage of the approach over the existing ones is the removal of the required extra 'book-keeping' to maintain and

define points that will locate matrices and the damage of overwriting in attempt to locate implicitly defined storage.

6.1 Program Development

After reviewing most of the available Finite Element computer packages, it became obvious that the early codes were developed at a great cost, yet the developed packages were highly specialized, machine dependent, and even difficult to modify.

Based on this fact we looked at the possibility of developing a package that will remove most of the routine activities of programming only to concentrate on activities which includes comprehension, modification and updating ability. This will enhance the user's familiarity with the internal function of the package to be able to take full advantage of the basic potentials of the Finite Element Method and avoid inappropriate application. These advantages were developed in chapter 3, with concentration on some general class of boundary value problems.

Structurally the Finite Element Method problems were divided into three basic functional stages which falls in line with the functioning of a digital computer:

PREPROCESSING; PROCESSING and POSTPROCESSING.

These three stages are so related that they work automatically with the view of communication tests that can signal the need for changing some parameters in the problem statement and thereby causing returns to either the preprocessor or processor stage.

Changeable variable and parameters are clearly defined in terms of their type, scopes and limitations, moreover algorithms of various procedures, in form of flow diagram are presented.

As the project had limited its scope to procedures of achieving flexibility, it concentrated on the routing that will realize some meaningful solution to boundary value problems, it had therefore made that assumption that the discretization of problem region can be done with other automatic routine or manually. This assumption was based on the fact that there are many good mask generation and refinement routines with the capabilities of reducing the maximum band width, thereby enhancing the efficiency of SBAND routine of section 3.2.6. This routine simply searches the entire problem region to get the maximum bandwidth.

The individual consideration of boundary conditions as prescribed or derivatives can now be done by using one procedure for both one and two dimensional problems. This

approach did not only bring about the reduction of storage wastage as a result of using unprescribed nodes, but had also brought about the reduction of computational time, from $O(N^2)$ to $O(K^2)$ where K << N. K is the number of either prescribed boundary conditions or derivative boundary condition while N is the number of global nodes. This was done by procedure ALPYBC in which the global boundary condition GBNDN and prescribed PBOUND were defined.

During the stages of the design and implementation of this project, we have to look at the objective set out for this project and a little backwards to the classification of the earlier development of various packages and software procedures. These early software fall into four basic forms: Individual programs, software packages, numerical software library and software system.

Individual programs software packages could not meet the objectives of this project, because they fell short of the communication language which will be far from the users method of solving Finite Element Method problems. Moreover, they mostly lack the facilities onto which structured equivalent of Finite Element Method will be translated and they have to subject the user to adapt to the constraints and implicit strategies facilitated by them.

Numerical software library is characterised by a large set of programs so that it will support general numerical computations. It mostly falls short of compatibility due to insufficient and lack of well organized documentation that will enable good selection of parameters. Moreover, numerical software library require an expert experience to be able to invoke most of the procedures.

The software system is generally made up of specialized user interface, like file manipulation and management, stream and graphic displays, memory management, input and output management. These facilities had removed most of the programming effort required by the user, such that consideration will be on Finite Element Method problems.

In section 4.2 the computational SOLVER was discussed. This method of software development was considered the best relative to individual programs software packages and numerical software library, because it does not require the user to know much about the internal working of the computer system and the methods of handling special facilities. The SOLVER concentrated mostly in taking care of these housekeeping facility, which are very sensitive as well as time consuming. The powers of the SOLVER strictly lies on the ability to make use of electronic sheets as a media for representing all parameters connecting the Finite Element

Method. These parameters ranged from the variables, rules, functions, values (lists), tables and arrays; up to, input and output format, numerical units and global specifications. A complete discussion on the capabilities of these sheets and their various subsheets was made in section 4.3.

The sheet approach of the SOLVER had reduced the cost of analysing a problem by allowing the user to only specify the Finite Element Method problem in the rule sheet and function without bothering about the variables that make up the rules and functions; which are automatically inserted in the variable sheet leaving the user with the options of specify the numeric format and units. This had gone to some extent to remove the problem of some of the objectives of the project, the reduction of the much pronounced limitation problem placed by program and computer in the use of computer storage. Moreover it had to some extent bridge the language communication gap between the Finite Element Method and computer implementation by not allowing the user to use specialized language in defining the problem parameters rather the analyst goes ahead to specify his problems.

Another interesting thing about the Solver approach that eases the application of Finite Element Method is with regards to boundary condition. The structured approach of chapter 2 which takes care of both the derivative and

prescribed boundary conditions can be seen to have worked out very well with the heat conduction and convention problem of 5.11. Here the temperature distribution became refined with the increase in the number of elements. These had shown that the Solver has the capability of getting a more accurate solution with its ability to contain more elements then the conventional methods.

The same approach was applied to the problem of Irrotational flow of fluid, problem of one derivations fluid flow between flat permeable plates, and problem on Hydrostatic pad. The results had shown that the solver results are more reliable compared to the results from Pesaro et al (1978) Duchsky (1980) and Molus (1982).

6.2. Experimental Results

This design of the interactive software for Finite Element Method procedures cannot come to a good conclusion without having a look on the experimental results. In Chapter 5 four problem were considered to see to what extent:

Accuracy of Problem

Heat conduction and convention in two dimensional body was considered with results of temperature distribution at various nodes. This had shown a improved refinement of the temperature distribution as the number of elements increases, (Fig 5.2 and Fig. 5.3). This had also shown to some extent the accuracy of Solver approach which had the capability of accommodating more elements, more than other approaches under consideration.

6.3. Expandability

The conventional method of programming which principally involves:

(i) coding, (ii) computation, (iii) linking and loading, and (iv) running, will not allow expansion without going through the three stages of programme development. This had resulted to difficulty in expansion. With the solver this process of computation linkage and loading are not necessary. Additional facilities can be added in form of function subsheets or expansion of original rules and referenced from the appropriate place from the rule sheet or other function subsheets.

6.4 Drawback and Limitations

Most of the research carried out had been based on the Software Development with regards to easing the problem of Finite Element Analyst who is not a Computer expert. Less attention had been paid to other Finite Element Problem principally because they will bring about complication or deviation from the main objective, secondly the time allowed for the research will not give room for such a diversed experiment.

REFERENCES

Aho, A., The Design and Analysis of Computer Algorithms. Addison- Wesley reading, Mass. 1974.

Akin J.E., Application and Implementation of Finite Element methods, Academic Press Inc. London 1982.

Allaire, P.E., Basics of Finite Element Methods, Wm. C. Brown Publishers Dubuque, Lowa, 1985.

Babuska Ivo and Werner, C.R.,: Adaptive Finite Element Process in Structural Mechanics. In: Garrett Birkhoff and Arthur Choenstadt (Ed.) Elliptic Problem Solver II. Proceedings of the Conference held at Naval Postgraduate School Monterey, Calif. Jan. 10 - 12 1983. Academic Press Inc. orlando, Fla 1984 PP. 345 - 377).

Babuska I. The Self Adaptive Approach in the Finite Element Method. In: J. Whitman (Ed.), Mathematics of Finite Element with Applications: MAFELAP 1975, Academic press, New York 1977.

Babuska, I and Szabo, On the rate of conveyance of the Finite Element Method, International Journal for Numerical Method in Engineering 18 323 - 341, 1982.

Babuska, I and Aziz, A.K.: Survey lecturer on the Mathematic Foundation of the Finite Element Method. In: A.K. Aziz (ed). The Mathematical Foundation of the Finite Element Method with application to PDE. Academic press, New York 5 - 359 (1972).

Bathe, k.J., Finite Element procedures in Engineering Analysis, prince-hall, Eaglewood Cliffs, New Jersey, 1982.

Bathe, K.H. and Wilson, E.L. Numerical Methods for Finite Element Analysis, Prentice-Hall, 1976.

Berns, M., Assessing Software Maintainability, Comm. ACM., Vol. 27 (1), 1984.

Berztiss, A.T., Data Structure, Theory and Practice, 2nd ed., Academic press, New York, 1975.

Bettess P. Infinite Elements. Int. J. Num. Math. Eng. 11, 53 - 64, 1977.

Bramble J.H. and H. Schatz, Rayleigh-Ritz-Galerkin: Method for Dirichlet's Problem using subspaces without boundary conditions, Commun. Pure Appl. Maths. 23, 653 - 675, 1970.

Cardenas, A.F. and Kaplus, W.J., PDEL - A language for partial differential equation, Comm. ACM., Vol. 13(3) 184-193, 1970.

Carey, G.F., A mesh refinement scheme for Finite element Computation. Comp. Math. Appl. Mech. Eng. Vol. 7, pp 93 - 105, 1976.

Cherry C., On Human Communication, 2nd ed. MIT press, Cambridge mass. 1970.

Cheung, Y.K. and Yeo, M.F. A practical Introduction to Finite Element Analysis, Pitman Publication Limited 1979.

Cody, W.J. Observation on Mathematical Software Effect In: Cowell, Wayne R. (Ed.) Source and development of Mathematical Software. Printice Hall series in Computational Mathematics. Printice-Hall Inc. Eagle wood Cliffs. N.J. 1984, PP. 1 - 19.

Cody W.J. FUNPACK: A package of special function routine In: Cowell, Wayne R. (Ed.) Source and development of Mathematical Software. Printice Hall series in Computational Mathematics. Printice-Hall Inc. Eagle wood Cliffs. N.J. 1984, PP. 49 - 67.

Cook R.D. Concepts and Application of Finite Element Analysis, John Wiley and Sons Inc. New York, 1974.

DeBoor, C. (Ed.) Mathematical Aspects of Finite Element in partial Differential Equations. Academic press 1974.

Duchsky, S. Quantitative Analysis of Numerical Software, Addision Wesley Reading, Mass 1980.

Eisenstat, S.c., J.A. George, R.G. Grimes, D.E. Kincaid and A.H. Sherman, "Some Comparison of Software packages for large sparse linear systems". In: R. Vichnevetsky and R.S. Stepleman (Eds.) Advances in Computer Methods for partial differential equations III, IMACS, New Brunswick, N.J. 98 - 106, 1979.

Eisenstat, S.C., M.H. Schults and A.H. Sherman Algorithm for sparse Symmetric Gaussian Elimination Research Report, Department of Computer Science, Yale University 1980.

Fix, G.J. Finite Element Approximation to Compressible Flow Problem In: Garrett Birkhoff and Arthur Choenstadt (Ed.) Elliptic Problem Solver II. Proceedings of the Conference held at Naval Postgraduate School Monterey, Calif. Jan. 10 - 12 1983. Academic Press Inc. orlando, Fla 1984 PP. 379 - 393).

Ferrari, D., Computer System performance evaluation, Prentice - Hall Englewood cliff N. J.

Gary J. and R. Helgerson, An extension of FORTRAN containing Finite difference operators. Software practice and experience, Vol. 2 PP. 321-336 1972.

Graham, F.C. and Oden T.J.: Finite Elements Computational Aspect Vol. III Prentice-Hall, Inc. Eaglewood Cliffs, N.J. (1984).

Gregory, R.T. Karney, D.L.: A Collection of Matrices for testing Computational Algorithm, Wiley Interscience, New York, 1969.

Halstad, M. H., Element of software Science, Elsevier North-Holland Inc, New York, 1977.

Hinton, E. and Owen D.R.J. Finite Element programming, Academic Press London 1977.

Irons, B.M. A Frontal solution program for Finite Elements, Int. J. Nuwer Meth. Eng. 2, 5 - 32, 1970.

Jennings, A. and A.D. Tuff. A Direct Method for the solution of large sparse symmetric simulation equations. In: J.K. Raid (Ed.) Large Sparse sets of Linear Equations, Academic press, New York 97 - 104, 1971.

Jones, R.E., A Self-Organizing Mesh Generation Program, ASME Publication 74 - PUP - 13, March 1974.

Laemuel, A. and Shooman, M., Statistical Natural Language Theory and Computer program complexity. POLY-EE/EP-76-020 Report, Aug. 1977.

Machura M. and R.A. Sweet: A Survey of Software for partial Differential Equation, ACM Transactions on Mathematical Software Vol. 6, No. 4 PP. 461 - 488 Dec. 1980.

Madsen, N. and Sincovec, General Software for partial differential equations In: Numerical Methods for differential systems: Recent Development in Software Algorithms and Applications, Academic Press, New York, PP. 229 - 242 1976.

Marek, M. and Sweet, R., Survey of software for Partial diferential equations, ACM Trans. Math. Software, Vol. 6, pp 461 - 488, Dec. 1980.

Mitchell, A.R. Computational Methods in partial differential equations, John Wiley and Sons 1976.

Mitchell, A.R. and Wait R., The Finite Element Method in Partial Differential Equations.

Melosh, R.D. Basics for Derivation of Matrices for the Direct Stiffnes Method. Jounal American Institute for Aeronautics and Astronautics, 1965, Vol. 1, pp 1631 - 37.

Molus K. P. Complexity of Numerical Software, Annual Review of Numerical Computation, Vol. 3, Pergmon press, New York 1982.

Morris, S.M. and Schliesser, W.E. SALCM - A Programming System for the Simulation of Systems described by partial differential equations In: Proc. AFIPS 1968 Fall Joint Compt. Conf. 33 part 1, Thompson Book Co., Washington D.C. PP. 353 - 357.

Newmark, N.M., A Method of Computational Structural Dynamics, J. Eng. Mech. Div. ASCE 85 EM3, 1959.

Norrie, D.H. De Vries G., Finite Element Method: Fundamentals and Applications. Academic Press, New York 1973.

Norrie, D.H. and Vries, G. de, An Introduction to Finite Element Analysis, Academic press, New York, 1975.

Oden, J.T. Reddy J.N. On mixed Finite Element Approximation, SIAM J. Numer. Anal. 13, 393 - 404 1976.

Oden T.J. and G.F. Carey, Finite Element: Mathematical Aspect. Prentice Hall Englewood Cliffs N.T. (1982).

Pesaro, H. D. Metrices for program frequency count, Digest of papers, spring COMPCOM '78 IEEE, New York 1987, pp 202 -206.

Rao, S.S., The Finite Element Methods in Engineering, Pergman International, 1982.

Russel, R.D. Finite Difference for two-point Boundary-value problems, SIAM J. Numer Anal 14, 19-39, 1977.

Seglerlind, L.J., Applied Finite Element Analysis, John Wiley and Sons, Inc. 1976.

Shampine, L.F. and H.A. Watts: Software for ordinary differential equation In: Cowell, Wayne R. (Ed.) Source and development of Mathematical Software. Printice Hall series in Computational Mathematics. Printice-Hall Inc. Eagle wood Cliffs. N.J. 1984, PP. 112 - 133.

Shooman, M.L., Software Engineering: Design, Reliability and Management, McGraw-Hill Inc. 1985.

Sloan, S.W., Randolph, M.F. Automatic Element Reordering for Finite Element Analysis with Frontal Solution Schemes. Int. Journ. Num. Method in Eng. Vol. 19 1153 - 1181 (1983).

Smith G.D. Numerical Solution of Partial Differential Equations. Oxford University Press, New York (1965).

Smith, R.E., (Ed.), Numerical Grid generation Techniques, NASA Conference Publication 2166, 1980.

Strang, G. and Fix, G., An Analysis of the Finite Element Method, Prentice Hall, New York, 1973.

Szabo, B. Some Recent Development in Finite Element Analysis. Compt. Math. Appl., 5, 2, 99 - 116, 1977.

Taylor, R.L., Computer procedure for Finite Element Analysis, In: Finite Element Methods, O.C. Zienkiewicz (ed.), McGraw-Hill Inc., 1977.

Tiechroew, D., Lubin J.F. Computer Simulation - Discussion of the Technique and Comparison of language. Comm. ACM 9, 10 (Oct. 1966) 723 - 741.

Turner, M. J., Clough, R.D., Martin, H.C. and Topp, L.J. Stiffness and Deflection Analysis of Complex Structures. Journal Aeronaurical Science, 1956, Vol. 23 pp 805 - 824.

Vemuri, V. and Kapplus, W.J., Digital Computer Treatment of Partial Differential Equations, 1981.

Wachspress, E, and R. McLeod: Curved Finite Elements. Special issue of Compt. Math. Appl. 5, 4, 1979.

Wang, lie Hang, On the maximum-minimum principle for the linear Finite Element equation for the Drichlet problem of Laplace equation. In: Kang Feng and J.L. Lions (Ed.). On Finite Element Methods proceedings of the China-France Symposium Beijing April 19 - 23rd 1982. Science press, Beijing, Gordon and Breach Science Publishers New York 1983, (PP. 1019 - 1026).

Wirth, N., Weber, H. EULER: A generalization of ALGOL and its formal definition: Pt. I Comm. ACM, 9, 1 (Jan. 1966). 11-23; Pt. II Comm. ACM, 9, 2 (Feb. 1966) 89-99.

Zienkiewicz, O.C., The Finite Element Method in Engineering Science, McGraw-Hill, New York, 1971.

The Main Program for the SOLVER

```
C
C*                                                                      *
C**                                                                    **
C***                                                                  ***
C****          MAIN PROGRAM FOR SOLVER                               ****
C***                                                                  ***
C**                                                                    **
C*                                                                      *
C
C
C
C*          I N T E R O V E R L A Y    P A R A M E T E R S
      PROGRAME SOLVER
C
      PARAMETER  (JNDSH = 30,  JDFSC = 12 * JNDSH,
     + JLD = JDFSC+1,  JSECT = 60,  JRWND = 15,  JRWEL = JRWND-1,
     + JELSH = 28, JZNSH = 1,  JMTSH = 1,
     + JSTRN = 1,    JNDSR = JRWND,  JELSR = JNDSR-1,
     + JMTSR = 3,  JIMSR = 1,  JELISR = 5,  JMTISR = 1,  JRING = 2,
     + JNDRN = 5,  JELRN = 3,  JMTRN = 2,  JIMRN = 15,  JELIRN = 20,
     + JMTIRN = 10,  JGAUS = 12,  JCRSM = 5,  JPRDS = JNDSH,
     + JLDCA = 2,  JCNLD = 5,  JDTLD = 2,  JAULD = 1,  JZNTP = 1,
     + JZNSL = 6,  JDFIF = 12 * JRWND,
     + JICON = 109, JDFSL = 64801)
C
C
C
C          I N T E R - O V E R L A Y    V A R I A B L E S
C
      CHARACTER*10  STRTYP, AUTGN, SLDSP, SLRAC, SLSTS, SLBKL,
     + PLSTR, PLMSH, PLDSP, PLSTS, PLMSP
C
       INTEGER  CSDCL, CSDMC, DSKF1, DSKF2, SLDLD,
     + SLVSC(JLDCA,JZNSL,2), ROWSH(JRWND,2),
     + SHMTI(JZNSH), SHMTJ(JZNSH), CSDSG(JCRSM), RSDDF(JPRDS,12),
     + CSPRD(JPRDS), SCNLI(JLDCA,JCNLD), SCNLJ(JLDCA,JCNLD),
     + CSCNL(JLDCA,JCNLD), TYDTL(JLDCA,JDTLD), DSDTL(JLDCA,JDTLD),
     + DRDTL(JLDCA,JDTLD,5), CSDTL(JLDCA,JDTLD), RFDTL(JLDCA,JDTLD),
     + SCTPI(JLDCA,JZNTP), SCTPJ(JLDCA,JZNTP), SEQND(JNDSH),
     + NDORD(JNDSH), IPNTR(JNDSH), ICONN(JICON), ROWSA(2)
C
       REAL  ORIGN(JRWEL,2), KCINV(JELSH,2,2), KSECT(JDFSL),
     + ITCNL(JLDCA,JCNLD,9), ITDTL(JLDCA,JDTLD,2), ITWSH(JMTSH,3)
C
       DIMENSION  NDDSH(JELSH,3), MTDSH(JZNSH), DNSSH(JELSH),
     + PRPSH(JMTSH,8), MTSHL(JELSH),
     + TSHEL(JMTSH), TRMSH(JMTSH), ARASH(JELSH), NODSR(JELSR,2),
     + MTDSR(JMTSR), ECNSR(JMTSR), TRMSR(JMTSR),
     + PRPSR(JMTSR,9), MTSTR(JELSR), CLSTR(JELSR,7,2),
     + NODISR(JELISR,2), NDIMSR(JIMSR,2), MTDISR(JMTISR),
```

```
     +  ECISR(JMTISR), TMISR(JMTISR),
     +  PRPISR(JMTISR,9), MTISR(JELISR), CLISR(JELISR,7,2),
     +  NDDRN(JELRN,2), MTDRN(JMTRN), TRMRN(JMTRN), PSNRN(JMTRN,3),
     +  PRPRN(JMTRN,9), MTRNG(JELRN), NRNGEL(JRING), NDDIRN(JELIRN,2),
     +  NDIMRN(JIMRN,2), MTDIRN(JMTIRN), PSNIRN(JMTIRN,3),
     +  TMIRN(JMTIRN), PRPIRN(JMTIRN,9), MTIRN(JELIRN), NIRNEL(JIMRN),
     +  CRDCL(JNDSH,3), CRDMC(JNDSH,3), GUSWT(JGAUS), GUSTR(JGAUS,3),
     +  GUSCL(JELSH,JGAUS,3), DVDTL(JLDCA,JDTLD), NDDPD(JPRDS),
     +  PRSDS(JPRDS,12), NREST(JPRDS),
     +  NDCNL(JLDCA,JCNLD), TPNDI(JLDCA,JZNTP), TPNDE(JLDCA,JZNTP),
     +  DRCOS(JCRSM,3,3), LD(JLD), NDJNC(JNDSH), CORLD(JDFSC),
     +  BRHS(JDFSC), X(JDFSC)
C
      LOGICAL  PDCHEK, SINGAB
C
C
C
      COMMON  STRTYP, AUTGN, SLDSP, SLRAC, SLSTS, SLBKL,
     +  PLSTR, PLMSH, PLDSP, PLSTS, PLMSP,
     +  MSECT, MSTRN, MNDSR, MRING, MNDRN, MGAUS,
     +  NNDSH, NELSH, NMTSH, NSTRN, NNDSR, NELSR, NMTSR, NIMSR,
     +  NELISR, NMTISR, NRING, NNDRN, NELRN, NMTRN, NIMRN, NELIRN,
     +  NMTIRN, NBRAC, NELBR, NMTBR, NGAUS, NCRSM, NPRDS, NLDCA,
     +  NSECT, NRWEL, NRWND, NDFND, NNDE1, NNDE2, NNDIN,
     +  NNTSH, NBTSH, NGTSH, NSTSSH, NKCTND, NNTSR, NBTSR, NGTSR,
     +  NTFSR, NSTSSR, NNTISR, NBTISR, NGTISR, NTFISR, NSTSIS, NNTRN,
     +  NBTRN, NGTRN, NTFRN, NSTSRN, NNTIRN, NBTIRN, NGTIRN, NTFIRN,
     +  NSTSIR, NTFBR, NSTSBR, NKRWPD, NCRLD, NFRCN, NABIF, NASCT,
     +  NACNT, NFBSC, NRACT, NKSCT, NKSCF, NKKGW, NKGSC, NKGICL, NKBNIF,
     +  NKGBNI, NEGVC, NKKGCN, NKGCN, NMSND, NMSCT, NKOEE, NKGOEE,
     +  CSDCL, CSDMC, MCCDD, ILDCA, LCNLD, LDTLD, LZNTP, LZNSL,
     +  DSKF1, DSKF2, IN, IO,
     +  FEED, RTOP, RBAS, HITE, ELEV, THATAD,
     +  TITLE(60), LDCAS(2), NCNLD(2), NDTLD(2), NZNTP(2), NZNSL(2),
     +  RATIO(14), INDEX(45), NDCON(200), NDEX1(15), NDEX2(15),
     +  NDINT(15), INTND(35), WKAREA(192)
C
      COMMON  / COMMN1 /
     +  SLVSC, ROWSH, SHMTI, SHMTJ, CSDSG,
     +  RSDDF, CSPRD, SCNLI, SCNLJ, CSCNL,
     +  TYDTL, DSDTL, DRDTL, CSDTL, RFDTL,
     +  SCTPI, SCTPJ, SEQND, NDDRD, IPNTR,
     +  ICONN, ROWSA,
     +  ORIGN, KCINV, KSECT, ITCNL,
     +  ITDTL, ITWSH,
     +  NDDSH, MTDSH, DNSSH, PRPSH, MTSHL,
     +  TSHEL, TRMSH, ARASH,
     +  NDDSR, MTDSR, ECNSR, TRMSR,
     +  PRPSR, MTSTR, CLSTR, NDDISR, NDIMSR, MTDISR,
     +  ECISR, TMISR, PRPISR, MTISR, CLISR, NDDRN,
     +  MTDRN, PSNRN, PRPRN, TRMRN, MTRNG
```

```
      C
            COMMON  / COMMN2 /
           + NRNGEL, NODIRN, NDIMRN, MTDIRN, PSNIRN,
           + TMIRN, PRPIRN, MTIRN, NIRNEL,
           + CRDCL, CRDMC, GUSWT, GUSTR, GUSCL,
           + DVDTL, NODPO,
           + PRSDS, NREST, NDCNL, TPNDI, TPNDE,
           + DRCOS, LD, NDJNC, CORLD, BRHS, X
      C
            COMMON  / COMMN3 /
           + LRDF1, MNDSH, MELSH, MMTSH, MELSR,
           + MMTSR, MIMSR, MELISR, MMTISR, MELRN, MMTRN,
           + MIMRN, MELIRN, MMTIRN, MGUSS, MZNTP, MZNSL, MRWND,
           + MRWEL, MCRSM, MPRDS, MLDCA, MCNLD, MDTLD, MAULD, MDFSC, MDFIF,
           + NDFIF, NDFEX, NDFIN, NDFSC, NINFA, NDFBL, NNDEX, NBEG, NEND, IOP,
           + IBX, PDCHEK, SINGAB, ICOND, IAFILE, IREF, DETCF, SLDLD, NDFSK
      C
      C
      C
      C*                                                                    *
      C**                                                                   **
      C***              O  V  E  R  L  A  Y       1       1                 ***
      C**                                                                   **
      C*                                                                    *
      C
      C
      C
            CALL ONECNE
      C
      C
      C
      C*    IF NO SOLUTION (ZONES) HAS BEEN REQUESTED, STOP.
      C
            IF (NZNSL(1) . NE . 0)                          GO TO 50
      C
      C
      C                                       S  T  O  P
      C
            PRINT*, 'EXECUTION WAS TERMINATED BY VAFA AFTER OVERLAY(1,1), ',
           + 'BECAUSE NO SOLUTION (ZONE) IS REQUESTED.'
      C
      C
            WRITE(IO,20)
         20 FORMAT(1H0,/// 6X,  'EXECUTION WAS TERMINATED BY VAFA AFTER ',
           + 'OVERLAY(1,1) BECAUSE NO SOLUTION (ZONE) IS REQUESTED.' //)
      C
      C
      C
      C***  READ AND WRITE THE DATA DESCRIBING THE PROBLEM (AND GENERATE
      C***  NODAL COORDINATES AND INCIDENCES, PLOT, ETC).
```

```
      C
      50 IF (NELSH . EQ . 0)                                  GO TO 150
      C
      C
      C
      C
      C*                                                                  *
      C**                                                                 **
      C***                 O  V  E  R  L  A  Y       1       2            ***
      C**                                                                 **
      C*                                                                  *
      C
      C***  CALCULATE AND ASSEMBLE THE STIFFNESS MATRICES OF ALL SHELL
      C***  ELEMENTS.
      C
            CALL ONETWO
      C
      C
      C
      150 IF (NELSR . EQ . 0 . AND . NELISR . EQ . 0 . AND .
           + NELRN . EQ . 0 . AND . NELIRN . EQ . 0 . AND . NELBR . EQ . 0)
           +                                                  GO TO 250
      C
      C
      C
      C*                                                                  *
      C**                                                                 **
      C***                 O  V  E  R  L  A  Y       1       3            ***
      C**                                                                 **
      C*                                                                  *
      C
      C***  CALCULATE AND ASSEMBLE THE STIFFNESS MATRICES OF ALL STRINGER,
      C***  INTERMEDIATE STRINGER, RING, AND INTERMEDIATE RING ELEMENTS.
      C
            CALL ONETHR
      C
      C
      C
      C*                                                                  *
      C**                                                                 **
      C***                 O  V  E  R  L  A  Y       1       4            ***
      C**                                                                 **
      C*                                                                  *
      C
      C
      C
      C***  PUT THE NONZERO ENTRIES OF KSECT CORRESPONDING TO A CONSTRAINED
      C***  DEGREE OF FREEDOM ON DISK DSKF1 (ROW BY ROW).
      C***  THEN CONSTRUCT THE CORRECTIVE LOAD VECTOR CORLD. THEN
      C***  SET THE NONDIAGONAL ENTRIES OF KSECT CORRESPONDING TO THE
      C***  PRESCRIBED DISPLACEMENTS EQUAL TO ZERO ( THE CORRESPONDING
```

```
C***  DIAGONAL ENTRIES ARE SET TO ONE ). THEN CONDENSE OUT THE
C***  INTERNAL STIFFNESS MATRIX.
C
  250 CALL ONEFOR
C
      END
C
C*                                                                    *
C**                                                                   **
C***                                                                  ***
C****                 O V E R L A Y      1     1                    ****
C***                                                                  ***
C**                                                                   **
C*                                                                    *
C
      SUBROUTINE ONEDNE
C
C
C*          I N T E R O V E R L A Y   P A R A M E T E R S
C
      PARAMETER  (JNDSH = 30,  JDFSC = 12 * JNDSH,
     + JLD = JDFSC+1,  JSECT = 60,  JRWND = 15,  JRWEL = JRWND-1,
     + JELSH = 28, JZNSH = 1,  JMTSH = 1,
     + JSTRN = 1,   JNDSR = JRWND,  JELSR = JNDSR-1,
     + JMTSR = 3,  JIMSR = 1,  JELISR = 5,  JMTISR = 1,  JRING = 2,
     + JNDRN = 5,  JELRN = 3,  JMTRN = 2,  JIMRN = 15,  JELIRN = 20,
     + JMTIRN = 10,  JGAUS = 12,  JCRSM = 5,  JPRDS = JNDSH,
     + JLDCA = 2,  JCNLD = 5,  JDTLD = 2,  JAULD = 1,  JZNTP = 1,
     + JZNSL = 6,  JDFIF = 12 * JRWND,
     + JICON = 109, JDFSL = 64801)
C
C
C
      DIMENSION  ALPHA(3)
C
C
C
C           I N T E R - O V E R L A Y    V A R I A B L E S
C
      CHARACTER*10  STRTYP, AUTGN, SLDSP, SLRAC, SLSTS, SLBKL,
     + PLSTR, PLMSH, PLDSP, PLSTS, PLMSP
C
      INTEGER  CSDCL, CSDMC, DSKF1, DSKF2, SLDLD,
     + SLVSC(JLDCA,JZNSL,2), ROWSH(JRWND,2),
     + SHMTI(JZNSH), SHMTJ(JZNSH), CSDSG(JCRSM), RSDDF(JPRDS,12),
     + CSPRD(JPRDS), SCNLI(JLDCA,JCNLD), SCNLJ(JLDCA,JCNLD),
     + CSCNL(JLDCA,JCNLD), TYDTL(JLDCA,JDTLD), DSDTL(JLDCA,JDTLD),
     + DRDTL(JLDCA,JDTLD,5), CSDTL(JLDCA,JDTLD), RFDTL(JLDCA,JDTLD),
     + SCTPI(JLDCA,JZNTP), SCTPJ(JLDCA,JZNTP), SEQND(JNDSH),
     + NDORD(JNDSH), IPNTR(JNDSH), ICONN(JICON), ROWSA(2)
C
```

```
      REAL  ORIGN(JRWEL,2), KCINV(JELSH,2,2), KSECT(JDFSL),
     + ITCNL(JLDCA,JCNLD,9), ITDTL(JLDCA,JDTLD,2), ITWSH(JMTSH,3)
C
      DIMENSION  NODSH(JELSH,3), MTDSH(JZNSH), DNSSH(JELSH),
     + PRPSH(JMTSH,8), MTSHL(JELSH),
     + TSHEL(JMTSH), TRMSH(JMTSH), ARASH(JELSH), NODSR(JELSR,2),
     + MTDSR(JMTSR), ECNSR(JMTSR), TRMSR(JMTSR),
     + PRPSR(JMTSR,9), MTSTR(JELSR), CLSTR(JELSR,7,2),
     + NODISR(JELISR,2), NDIMSR(JIMSR,2), MTDISR(JMTISR),
     + ECISR(JMTISR), TMISR(JMTISR),
     + PRPISR(JMTISR,9), MTISR(JELISR), CLISR(JELISR,7,2),
     + NODRN(JELRN,2), MTDRN(JMTRN), TRMRN(JMTRN), PSNRN(JMTRN,3),
     + PRPRN(JMTRN,9), MTRNG(JELRN), NRNGEL(JRING), NODIRN(JELIRN,2),
     + NDIMRN(JIMRN,2), MTDIRN(JMTIRN), PSNIRN(JMTIRN,3),
     + TMIRN(JMTIRN), PRPIRN(JMTIRN,9), MTIRN(JELIRN), NIRNEL(JIMRN),
     + CRDCL(JNDSH,3), CRDMC(JNDSH,3), GUSWT(JGAUS), GUSTR(JGAUS,3),
     + GUSCL(JELSH,JGAUS,3), DVDTL(JLDCA,JDTLD), NODPD(JPRDS),
     + PRSDS(JPRDS,12), NREST(JPRDS),
     + NDCNL(JLDCA,JCNLD), TPNDI(JLDCA,JZNTP), TPNDE(JLDCA,JZNTP),
     + DRCOS(JCRSM,3,3), LD(JLD), NDJNC(JNDSH), CORLD(JDFSC),
     + BRHS(JDFSC), X(JDFSC)
C
      LOGICAL  PDCHEK, SINGAB
C
C
C
      COMMON  STRTYP, AUTGN, SLDSP, SLRAC, SLSTS, SLBKL,
     + PLSTR, PLMSH, PLDSP, PLSTS, PLMSP,
     + MSECT, MSTRN, MNDSR, MRING, MNDRN, MGAUS,
     + NNDSH, NELSH, NMTSH, NSTRN, NNDSR, NELSR, NMTSR, NIMSR,
     + NELISR, NMTISR, NRING, NNDRN, NELRN, NMTRN, NIMRN, NELIRN,
     + NMTIRN, NBRAC, NELBR, NMTBR, NGAUS, NCRSM, NPROS, NLDCA,
     + NSECT, NRWEL, NRWND, NDFND, NNDE1, NNDE2, NNDIN,
     + NNTSH, NBTSH, NGTSH, NSTSSH, NKCTND, NNTSR, NBTSR, NGTSR,
     + NTFSR, NSTSSR, NNTISR, NBTISR, NGTISR, NTFISR, NSTSIS, NNTRN,
     + NBTRN, NGTRN, NTFRN, NSTSRN, NNTIRN, NBTIRN, NGTIRN, NTFIRN,
     + NSTSIR, NTFBR, NSTSBR, NKRWPD, NCRLD, NFRCN, NABIF, NASCT,
     + NACNT, NFBSC, NRACT, NKSCT, NKSCF, NKKGW, NKGSC, NKGICL, NKBNIF,
     + NKGBNI, NEGVC, NKKGCN, NKGCN, NMSND, NMSCT, NKOEE, NKGOEE,
     + CSDCL, CSDMC, MCCOD, ILDCA, LCNLD, LDTLD, LZNTP, LZNSL,
     + DSKF1, DSKF2, IN, IO,
     + FEED, RTOP, RBAS, HITE, ELEV, THATAD,
     + TITLE(60), LDCAS(2), NCNLD(2), NDTLD(2), NZNTP(2), NZNSL(2),
     + RATIO(14), INDEX(45), NDCON(200), NDEX1(15), NDEX2(15),
     + NDINT(15), INTND(35), WKAREA(192)
C
      COMMON  / COMMN1 /
     + SLVSC, ROWSH, SHMTI, SHMTJ, CSDSG,
     + RSDOF, CSPRD, SCNLI, SCNLJ, CSCNL,
     + TYDTL, DSDTL, DRDTL, CSDTL, RFDTL,
     + SCTPI, SCTPJ, SEQND, NDORD, IPNTR,
```

```
     +  ICONN, ROWSA,
     +  ORIGN, KCINV, KSECT, ITCNL,
     +  ITOTL, ITWSH,
     +  NODSH, MTDSH, DNSSH, PRPSH, MTSHL,
     +  TSHEL, TRMSH, ARASH,
     +  NODSR, MTDSR, ECNSR, TRMSR,
     +  PRPSR, MTSTR, CLSTR, NODISR, NDIMSR, MTDISR,
     +  ECISR, TMISR, PRPISR, MTISR, CLISR, NODRN,
     +  MTDRN, PSNRN, PRPRN, TRMRN, MTRNG
C
      COMMON  / COMMN2 /
     +  NRNGEL, NODIRN, NDIMRN, MTDIRN, PSNIRN,
     +  TMIRN, PRPIRN, MTIRN, NIRNEL,
     +  CROCL, CROMC, GUSWT, GUSTR, GUSCL,
     +  DVDTL, NODPD,
     +  PRSDS, NREST, NDCNL, TPNDI, TPNDE,
     +  DRCOS, LD, NDJNC, CORLD, BRHS, X
C
      COMMON  / COMMN3 /
     +  LRDF1, MNDSH, MELSH, MMTSH, MELSR,
     +  MMTSR, MIMSR, MELISR, MMTISR, MELRN, MMTRN,
     +  MIMRN, MELIRN, MMTIRN, MGUSS, MZNTP, MZNSL, MRWND,
     +  MRWEL, MCRSM, MPRDS, MLOCA, MCNLD, MDTLD, MAULD, MDFSC, MDFIF,
     +  NDFIF, NDFEX, NDFIN, NDFSC, NINFA, NDFBL, NNDEX, NBEG, NEND, IOP,
     +  IBX, PDCHEK, SINGAB, ICOND, IAFILE, IREF, DETCF, SLDLD, NDFSK
C
C***  FETCH MAXIMUMS OF VARIABLE.
C
      CALL MAXIMS (JNDSH, JSECT, JRWND, JELSH, JZNSH, JMTSH,
     +  JSTRN, JNDSR, JELSR, JMTSR, JIMSR, JELISR, JMTISR,
     +  JRING, JNDRN, JELRN, JMTRN, JIMRN, JELIRN, JMTIRN,
     +  JGAUS, JCRSM, JPRDS, JLOCA, JCNLD, JDTLD, JAULD,
     +  JZNTP, JZNSL, JICON, MNDSH, MELSH, MZNSH, MMTSH, MSTRN, MNDSR,
     +  MELSR, MMTSR, MIMSR, MELISR, MMTISR, MRING, MNDRN, MELRN, MMTRN,
     +  MIMRN, MELIRN, MMTIRN, MGAUS, MGUSS, MZNTP, MZNSL, MICON, MSECT,
     +  MRWND, MRWEL, MCRSM, MPRDS, MLOCA, MCNLD, MDTLD, MAULD, MDFSC,
     +  MDFIF)
C
C
C***  FETCH THE LOCAL DISK FILE ('TAPE') NUMBERS.
C
      CALL DSKFIL (DSKF1, DSKF2, IN, IO)
C
C
C***  CALL THE SUBROUTINE THAT READS , WRITES AND PLOTS (DESCRIBES)
C***  AND CHECKS THE INPUT DATA.
C
      CALL DESCR1 (MNDSH, MELSH, MRWND, MRWEL, MMTSH, MELSR,
     +  MMTSR, MIMSR, MELISR, MMTISR, MELRN, MMTRN, MIMRN, MELIRN,
     +  MMTIRN, MCRSM, MPRDS, MLOCA, MCNLD, MDTLD, MZNTP, NODSH, MTDSH,
     +  DNSSH, PRPSH, MTSHL, SHMTI, SHMTJ, TSHEL, TRMSH,
```

```
     +  ROWSH, ORIGN, NODSR, MTDSR, ECNSR, TRMSR, PRPSR, MTSTR, NODISR,
     +  NDIMSR, MTDISR, ECISR, TMISR, PRPISR, MTISR, NODRN, MTORN,
     +  PSNRN, TRMRN, PRPRN, MTRNG, NRNGEL, NODIRN, NDIMRN, MTDIRN,
     +  PSNIRN, TMIRN, PRPIRN, MTIRN, NIRNEL, CRDCL, CRDMC, ALPHA)
C
C
C
C***  READ, WRITE (DESCRIBE) , CHECK PRESCRIBED RESTRAINTS /
C***  DISPLACEMENTS, LOADS AND SHELL ELEMENT TEMPERATURE SPECIFICATIONS.
C
      CALL DESCR2 (AUTGN, CSDCL, CSDMC, MCCOD, IN, IO,
     +  MRWND, MCRSM, MPRDS, MLDCA, MCNLD, MDTLD, MZNTP, MZNSL, NSECT,
     +  NRWEL, NNDSH, NCRSM, NPRDS, NLDCA, SLBKL, STRTYP, TITLE, NDEX1,
     +  NDEX2, ROWSH, CSDSG, DRCDS, NODPD, PRSDS, RSDOF, NREST, CSPRD,
     +  LDCAS, NCNLD, NDTLD, NZNTP, NZNSL, SCNLI, SCNLJ, NDCNL,
     +  ITCNL, CSCNL, TYDTL, DSDTL, ITDTL, DRDTL, CSDTL, RFDTL,
     +  DVDTL, SCTPI, SCTPJ, TPNDI, TPNDE, SLVSC)
C
C
C
C***  GET CERTAIN CONSTANTS.
C
      CALL CONSTN (NDFND, NRWND, NSECT, NNDSH, NNDIN, NDFIF,
     +  NDFEX, NDFIN, NDFSC, NINFA, NDFBL, NNDEX, NBEG, NEND, IOP,
     +  IBX, PDCHEK, SINGAB, ICOND, IAFILE, IREF, DETCF)
C
C
C
C***  GENERATE VECTORS SEQND AND NDORD, WHICH GIVE THE
C***  NODAL SEQUENCE AND ORDER, RESPECTIVELY, AND SET UP VECTORS IPNTR
C***  AND ICONN, WHICH TOGETHER PROVIDE THE INFORMATION ABOUT THE NODES
C***  CONNECTED TO A GIVEN NODE, AND ALSO CALCULATE PARAMETER NDFSL,
C***  NUMBER OF DEGREES OF FREEDOM USED IN THE SKYLINE STORAGE PROCESS
C***  IN VECTOR KSECT. ALSO GENERATE VECTOR NDJNC, WHICH DETERMINES
C***  WHETHER A GIVEN NODE IS LOCATED AT A JUNCTURE WHERE THE
C***  MERIDIONAL GEOMETRY OF THE SHELL CHANGES.
C
      CALL SUNDRY (ID, JDFSL, MRWND, MICON, NNDSH, NRWND,
     +  NRWEL, NNDE1, NNDE2, NNDIN, NLDCA, NGAUS, NDFND, NDFBL, NDFSK,
     +  NDFSL, STRTYP, SLDSP, SLSTS, SLBKL, NZNTP, ROWSH, NDEX1, NDEX2,
     +  NDINT, SEQND, NDORD, INDEX, NDCON, IPNTR, ICONN, LD, NDJNC,
     +  KSECT)
C
C
C*    IF BUCKLING SOLUTION HAS NOT BEEN REQUESTED, SKIP.
C
      IF (SLBKL . NE . 'YES')                          GO TO 135
C
C
C*    WHEN BUCKLING SOLUTION HAS BEEN REQUESTED, THE DIMENSION OF
C*    VECTOR KSECT, JDFSL, SHOULD BE AT LEAST   2 * NDFIF * NDFIF + 1 .
```

```
C*     THIS IS BECAUSE VECTOR KGSCT(JFDSL) (SAME PLACE IN MEMORY AS
C*     KSECT) IN SUBROUTINE BUCKLE IS PASSED TO THE COMPLEX MATRIX
C*     EIGVC(NDFIF,NDFIF) AND VECTOR EXTRA( ) IN SUBROUTINE SLVEIG
C*     THROUGH COMMON  COMMN1.
C
       MINDF = 2 * NDFIF * NDFIF + 1
C
       IF (JDFSL . GE . MINDF)                          GO TO 135
C
C
       PRINT*,  'ERROR MESSAGE FROM OVERLAY(1,1):'
       PRINT*, ' EXECUTION IS ABORTED BECAUSE WHEN BUCKLING SOLUTION IS',
      + ' REQUESTED, THE DIMENSION OF VECTOR KSECT SHOULD AT LEAST BE ',
      + MINDF
C
C
       WRITE(IO,130)  MINDF
  130 FORMAT (1H0, 9X, 'ERROR MESSAGE FROM OVERLAY(1,1):',/,
      + 10X, 'EXECUTION IS ABORTED, BECAUSE WHEN BUCKLING SOLUTION IS ',
      + 'REQUESTED THE DIMENSION OF VECTOR KSECT SHOULD ', /,
      + 'BE AT LEAST', 1I10/)
C
C
       STOP
C
C
C*     IF PLOTTING OF THE STRUCTURE IS NOT REQUESTED, SKIP.
C
  135 IF (PLSTR . NE . 'YES')                          GO TO 150
C
C***   PLOT THE ENTIRE STRUCTURE IN 3-DIMENSIONS, SHOWING ALL
C***   STIFFENERS. NOTE THAT IN THIS PLOT THE HIDDEN LINES ARE
C***   REMOVED.
C***   THE ' N C A R ', THE NATIONAL CENTER FOR ATMOSPHERIC RESEARCH'S,
C***   ROUTINES ARE USED HERE FOR ALL GRAPHICS.
C
       CALL PLTSTR (AUTGN, ELEV, STRTYP, MNDSH, NRWND, CRDCL,
      + NDEX1, ALPHA, KSECT, WKAREA)
C
C***   PLOT THE ENTIRE STRUCTURE IN THREE DIMENSIONS WITH HIDDEN
C***   LINES SHOWN
C***   THE ' N C A R ', THE NATIONAL CENTER FOR ATMOSPHERIC RESEARCH'S,
C***   ROUTINES ARE USED HERE FOR ALL GRAPHICS.
C
       CALL PLSTRU (ELEV, HITE, MNDSH, MELSR, MELISR, MIMSR,
      + MIMRN, NSECT, NRWND, NNDSH, NSTRN, NELSR, NIMSR, NELISR,
      + NIMRN, RTOP, RBAS, STRTYP, THATAD, CRDCL, NODSR, NODISR, NDIMSR,
      + NDIMRN, NDEX1, NDJNC)
C
C
C*     IF PLOTTING OF THE MESH HAS NOT BEEN REQUESTED, SKIP.
```

```
C
  150 IF (PLMSH . NE . 'YES')                          GO TO 250
C
C
C***  PLOT THE MESH FOR ONE SECTOR IN THREE DIMENSIONS, SHOWING THE
C***  SHELL, STRINGER, INTERMEDIATE STRINGERS, RINGS, AND INTERMEDIATE
C***  RINGS.
C***  THE ' N C A R ', THE NATIONAL CENTER FOR ATMOSPHERIC RESEARCH'S,
C***  ROUTINES ARE USED HERE FOR ALL GRAPHICS.
C
      CALL PLTMSH (MNDSH, MELSR, MELISR, MELRN, MELIRN,
     + NNDSH, NELSH, NELSR, NELISR, NELRN, NELIRN, NRWND, CRDCL,
     + NODSR, NODISR, NODRN, NODIRN, NDEX1, SEQND, IPNTR, ICONN)
C
C
C
C*    IF THE SOLUTION OF DISPLACEMENTS IS NOT REQUESTED, STOP,
C*    AFTER ISSUING A MESSAGE.
C
  250 IF (SLDSP . EQ . 'YES')                          GO TO 350
C
C
      PRINT*, ' THE PROGRAM STOPS HERE, BECAUSE, ',
     + 'SOLUTION OF THE DISPLACEMENTS IS NOT REQUESTED.'
C
      WRITE(IO,300)
  300 FORMAT (1H0,//,6X, ' THE PROGRAM STOPS HERE, BECAUSE, ',
     + 'SOLUTION OF THE DISPLACEMENTS IS NOT REQUIERED.')
C
C
      STOP
C
C
C***  OPEN ALL DISK FILES, IF NECESSARY ( EXCEPT IN=5 AND IO=6, WHICH
C***  HAVE ALREADY BEEN OPENED).
C
  350 CALL OPENDF (DSKF1, DSKF2, LRDF1, MAULD, MZNTP, MGAUS,
     + NLDCA, NNDIN, NDFIF, NDFSC, SLRAC, SLSTS, SLBKL, NZNTP)
C
C***  CALCULATE THE RECORD NUMBERS OF THE RECORDS LOCATED JUST BEFORE
C***  THE POSITION WHERE THE VARIOUS SETS OF DATA ARE TO BE WRITTEN
C***  OR READ) ON (FROM) DISK FILES.
C
      CALL RECNUM (LRDF1, MLDCA, MZNSL, NDFSC, NDFSL, NDDPD,
     + NREST, SLVSC, SCNLI, SCNLJ, TYDTL, RFDTL, SCTPI, SCTPJ)
C
C
      RETURN
      END
C
C*                                                              *
```

```
C**                                                                  **
C***                                                                 ***
C****          O V E R L A Y     1     2                            ****
C***                                                                 ***
C**                                                                  **
C*                                                                    *
C
      SUBROUTINE ONETWO
C
C
C*          I N T E R O V E R L A Y   P A R A M E T E R S
C
      PARAMETER  (JNDSH = 30,  JDFSC = 12 * JNDSH,
     + JLD = JDFSC+1,  JSECT = 60,  JRWND = 15,  JRWEL = JRWND-1,
     + JELSH = 28, JZNSH = 1,  JMTSH = 1,
     + JSTRN = 1,   JNDSR = JRWND,  JELSR = JNDSR-1,
     + JMTSR = 3,  JIMSR = 1,  JELISR = 5,  JMTISR = 1,  JRING = 2,
     + JNDRN = 5,  JELRN = 3,  JMTRN = 2,  JIMRN = 15,  JELIRN = 20,
     + JMTIRN = 10,  JGAUS = 12,  JCRSM = 5,  JPRDS = JNDSH,
     + JLDCA = 2,  JCNLD = 5,  JDTLD = 2,  JAULD = 1,  JZNTP = 1,
     + JZNSL = 6,  JDFIF = 12 * JRWND,
     + JICON = 109, JDFSL = 64801)
C
C
C             L O C A L    V A R I A B L E S
C
      REAL  NODTR(JGAUS,3), NTSHL(JGAUS,3,38)
C
      DIMENSION  GTSHL(JGAUS,3,38), BTSHL(JGAUS,6,38), CURVT(JGAUS,5),
     + TFMCI(38,38)
C
C
C
C          I N T E R - O V E R L A Y   V A R I A B L E S
C
      CHARACTER*10  STRTYP, AUTGN, SLDSP, SLRAC, SLSTS, SLBKL,
     + PLSTR, PLMSH, PLDSP, PLSTS, PLMSP
C
      INTEGER  CSDCL, CSDMC, DSKF1, DSKF2, SLDLD,
     + SLVSC(JLDCA,JZNSL,2), ROWSH(JRWND,2),
     + SHMTI(JZNSH), SHMTJ(JZNSH), CSDSG(JCRSM), RSDDF(JPRDS,12),
     + CSPRD(JPRDS), SCNLI(JLDCA,JCNLD), SCNLJ(JLDCA,JCNLD),
     + CSCNL(JLDCA,JCNLD), TYDTL(JLDCA,JDTLD), DSDTL(JLDCA,JDTLD),
     + DRDTL(JLDCA,JDTLD,5), CSDTL(JLDCA,JDTLD), RFDTL(JLDCA,JDTLD),
     + SCTPI(JLDCA,JZNTP), SCTPJ(JLDCA,JZNTP), SEQND(JNDSH),
     + NDDRD(JNDSH), IPNTR(JNDSH), ICONN(JICON), ROWSA(2)
C
      REAL  ORIGN(JRWEL,2), KCINV(JELSH,2,2), KSECT(JDFSL),
     + ITCNL(JLDCA,JCNLD,9), ITDTL(JLDCA,JDTLD,2), ITWSH(JMTSH,3)
C
      DIMENSION  NODSH(JELSH,3), MTDSH(JZNSH), DNSSH(JELSH),
```

```
     +  PRPSH(JMTSH,8), MTSHL(JELSH),
     +  TSHEL(JMTSH), TRMSH(JMTSH), ARASH(JELSH), NODSR(JELSR,2),
     +  MTDSR(JMTSR), ECNSR(JMTSR), TRMSR(JMTSR),
     +  PRPSR(JMTSR,9), MTSTR(JELSR), CLSTR(JELSR,7,2),
     +  NODISR(JELISR,2), NDIMSR(JIMSR,2), MTDISR(JMTISR),
     +  ECISR(JMTISR), TMISR(JMTISR),
     +  PRPISR(JMTISR,9), MTISR(JELISR), CLISR(JELISR,7,2),
     +  NODRN(JELRN,2), MTDRN(JMTRN), TRMRN(JMTRN), PSNRN(JMTRN,3),
     +  PRPRN(JMTRN,9), MTRNG(JELRN), NRNGEL(JRING), NODIRN(JELIRN,2),
     +  NDIMRN(JIMRN,2), MTDIRN(JMTIRN), PSNIRN(JMTIRN,3),
     +  TMIRN(JMTIRN), PRPIRN(JMTIRN,9), MTIRN(JELIRN), NIRNEL(JIMRN),
     +  CRDCL(JNDSH,3), CRDMC(JNDSH,3), GUSWT(JGAUS), GUSTR(JGAUS,3),
     +  GUSCL(JELSH,JGAUS,3), DVDTL(JLDCA,JDTLD), NODPD(JPRDS),
     +  PRSDS(JPRDS,12), NREST(JPRDS),
     +  NDCNL(JLDCA,JCNLD), TPNDI(JLDCA,JZNTP), TPNDE(JLDCA,JZNTP),
     +  DRCDS(JCRSM,3,3), LD(JLD), NDJNC(JNDSH), CORLD(JDFSC),
     +  BRHS(JDFSC), X(JDFSC)
C
      LOGICAL  PDCHEK, SINGAB
C
C
C
      COMMON  STRTYP, AUTGN, SLDSP, SLRAC, SLSTS, SLBKL,
     +  PLSTR, PLMSH, PLDSP, PLSTS, PLMSP,
     +  MSECT, MSTRN, MNDSR, MRING, MNDRN, MGAUS,
     +  NNDSH, NELSH, NMTSH, NSTRN, NNDSR, NELSR, NMTSR, NIMSR,
     +  NELISR, NMTISR, NRING, NNDRN, NELRN, NMTRN, NIMRN, NELIRN,
     +  NMTIRN, NBRAC, NELBR, NMTBR, NGAUS, NCRSM, NPRDS, NLDCA,
     +  NSECT, NRWEL, NRWND, NDFND, NNDE1, NNDE2, NNDIN,
     +  NNTSH, NBTSH, NGTSH, NSTSSH, NKCTND, NNTSR, NBTSR, NGTSR,
     +  NTFSR, NSTSSR, NNTISR, NBTISR, NGTISR, NTFISR, NSTSIS, NNTRN,
     +  NBTRN, NGTRN, NTFRN, NSTSRN, NNTIRN, NBTIRN, NGTIRN, NTFIRN,
     +  NSTSIR, NTFBR, NSTSBR, NKRWPD, NCRLD, NFRCN, NABIF, NASCT,
     +  NACNT, NFBSC, NRACT, NKSCT, NKSCF, NKKGW, NKGSC, NKGICL, NKBNIF,
     +  NKGBNI, NEGVC, NKKGCN, NKGCN, NMSND, NMSCT, NKOEE, NKGOEE,
     +  CSDCL, CSDMC, MCCOD, ILDCA, LCNLD, LDTLD, LZNTP, LZNSL,
     +  DSKF1, DSKF2, IN, IO,
     +  FEED, RTOP, RBAS, HITE, ELEV, THATAD,
     +  TITLE(60), LDCAS(2), NCNLD(2), NDTLD(2), NZNTP(2), NZNSL(2),
     +  RATIO(14), INDEX(45), NDCON(200), NDEX1(15), NDEX2(15),
     +  NDINT(15), INTND(35), WKAREA(192)
C
      COMMON  / COMMN1 /
     +  SLVSC, ROWSH, SHMTI, SHMTJ, CSDSG,
     +  RSDOF, CSPRD, SCNLI, SCNLJ, CSCNL,
     +  TYDTL, DSDTL, DRDTL, CSDTL, RFDTL,
     +  SCTPI, SCTPJ, SEQND, NDORD, IPNTR,
     +  ICONN, ROWSA,
     +  ORIGN, KCINV, KSECT, ITCNL,
     +  ITDTL, ITWSH,
     +  NODSH, MTDSH, DNSSH, PRPSH, MTSHL,
```

```
     + TSHEL, TRMSH, ARASH,
     + NODSR, MTDSR, ECNSR, TRMSR,
     + PRPSR, MTSTR, CLSTR, NODISR, NDIMSR, MTDISR,
     + ECISR, TMISR, PRPISR, MTISR, CLISR, NODRN,
     + MTDRN, PSNRN, PRPRN, TRMRN, MTRNG
C
      COMMON  / COMMN2 /
     + NRNGEL, NODIRN, NDIMRN, MTDIRN, PSNIRN,
     + TMIRN, PRPIRN, MTIRN, NIRNEL,
     + CRDCL, CRDMC, GUSWT, GUSTR, GUSCL,
     + DVDTL, NODPD,
     + PRSDS, NREST, NDCNL, TPNDI, TPNDE,
     + DRCOS, LD, NDJNC, CORLD, BRHS, X
C
      COMMON  / COMMN3 /
     + LRDF1, MNDSH, MELSH, MMTSH, MELSR,
     + MMTSR, MIMSR, MELISR, MMTISR, MELRN, MMTRN,
     + MIMRN, MELIRN, MMTIRN, MGUSS, MZNTP, MZNSL, MRWND,
     + MRWEL, MCRSM, MPRDS, MLDCA, MCNLD, MDTLD, MAULD, MDFSC, MDFIF,
     + NDFIF, NDFEX, NDFIN, NDFSC, NINFA, NDFBL, NNDEX, NBEG, NEND, IOP,
     + IBX, PDCHEK, SINGAB, ICOND, IAFILE, IREF, DETCF, SLDLD, NDFSK
C
      EQUIVALENCE  ( BTSHL(1,1,1), TFMCI(1,1) )
C
C
C
C***  CALCULATE THE STIFFNESS MATRIX OF ALL SHELL ELEMENTS
C***  ON NO NODES OF WHICH A DISPLACEMENT IS PRESCRIBED IN
C***  THE NON-MERIDIONAL-CIRCUMFERENTIAL COORDINATE SYSTEM.
C***  THEN ASSEMBLE THEM IN THE SKYLINE FASHION IN VECTOR
C***  KSECT. FOR EACH ELEMENT, CALCULATE THE MATRIX FOR THE
C***  TRANSFORMATION OF COORDINATES TO A GIVEN NONMERIDIONAL-
C***  CIRCUMFERENTIAL SYSTEM. WRITE N(3,38), AND IF NECESSARY
C***  MATRICES B(6,38) AND G(3,38) ON DISK FILE DSKF1.
C
      CALL KSHELL (MNDSH, MELSH, MRWND, MRWEL, MMTSH, MCRSM,
     + MGUSS, MPRDS, NDFSC, NODSH, TSHEL, MTSHL, PRPSH,
     + ARASH, ROWSH, ORIGN, CRDCL, CRDMC, DRCOS,
     + CSDSG, GUSCL, GUSTR, GUSWT, NODTR, NODPD, PRSDS,
     + CSPRD, RSDOF, NREST, SEQND, NDORD, IPNTR, ICONN, LD, NDJNC,
     + KCINV, KSECT, GTSHL, NTSHL, BTSHL, TFMCI, CURVT)
C
C
C
C
      RETURN
      END
C
C*                                                              *
C**                                                            **
C***                                                          ***
```

```
C**** O V E R L A Y 1 3 ****
C*** ***
C** **
C* *
C
C
      SUBROUTINE ONETHR
C
C
C*          I N T E R O V E R L A Y   P A R A M E T E R S
C
       PARAMETER  (JNDSH = 30,  JDFSC = 12 * JNDSH,
     + JLD = JDFSC+1,  JSECT = 60,  JRWND = 15,  JRWEL = JRWND-1,
     + JELSH = 28, JZNSH = 1,  JMTSH = 1,
     + JSTRN = 1,   JNDSR = JRWND,  JELSR = JNDSR-1,
     + JMTSR = 3,  JIMSR = 1,  JELISR = 5,  JMTISR = 1,  JRING = 2,
     + JNDRN = 5,  JELRN = 3,  JMTRN = 2,  JIMRN = 15,  JELIRN = 20,
     + JMTIRN = 10,  JGAUS = 12,  JCRSM = 5,  JPRDS = JNDSH,
     + JLDCA = 2,  JCNLD = 5,  JDTLD = 2,  JAULD = 1,  JZNTP = 1,
     + JZNSL = 6,  JDFIF = 12 * JRWND,
     + JICON = 109, JDFSL = 64801)
C
C
C
C           I N T E R - O V E R L A Y   V A R I A B L E S
C
       CHARACTER*10  STRTYP, AUTGN, SLDSP, SLRAC, SLSTS, SL8KL,
     + PLSTR, PLMSH, PLDSP, PLSTS, PLMSP
C
       INTEGER  CSDCL, CSDMC, DSKF1, DSKF2, SLDLD,
     + SLVSC(JLDCA,JZNSL,2), ROWSH(JRWND,2),
     + SHMTI(JZNSH), SHMTJ(JZNSH), CSDSG(JCRSM), RSDOF(JPRDS,12),
     + CSPRD(JPRDS), SCNLI(JLDCA,JCNLD), SCNLJ(JLDCA,JCNLD),
     + CSCNL(JLDCA,JCNLD), TYDTL(JLDCA,JDTLD), DSDTL(JLDCA,JDTLD),
     + DRDTL(JLDCA,JDTLD,5), CSDTL(JLDCA,JDTLD), RFDTL(JLDCA,JDTLD),
     + SCTPI(JLDCA,JZNTP), SCTPJ(JLDCA,JZNTP), SEQND(JNDSH),
     + NDORD(JNDSH), IPNTR(JNDSH), ICONN(JICON), ROWSA(2)
C
       REAL  ORIGN(JRWEL,2), KCINV(JELSH,2,2), KSECT(JDFSL),
     + ITCNL(JLDCA,JCNLD,9), ITDTL(JLDCA,JDTLD,2), ITWSH(JMTSH,3)
C
       DIMENSION  NODSH(JELSH,3), MTDSH(JZNSH), DNSSH(JELSH),
     + PRPSH(JMTSH,8), MTSHL(JELSH),
     + TSHEL(JMTSH), TRMSH(JMTSH), ARASH(JELSH), NODSR(JELSR,2),
     + MTDSR(JMTSR), ECNSR(JMTSR), TRMSR(JMTSR),
     + PRPSR(JMTSR,9), MTSTR(JELSR), CLSTR(JELSR,7,2),
     + NDDISR(JELISR,2), NDIMSR(JIMSR,2), MTDISR(JMTISR),
     + ECISR(JMTISR), TMISR(JMTISR),
     + PRPISR(JMTISR,9), MTISR(JELISR), CLISR(JELISR,7,2),
     + NODRN(JELRN,2), MTDRN(JMTRN), TRMRN(JMTRN), PSNRN(JMTRN,3),
     + PRPRN(JMTRN,9), MTRNG(JELRN), NRNGEL(JRING), NODIRN(JELIRN,2),
```

```
      + NDIMRN(JIMRN,2), MTDIRN(JMTIRN), PSNIRN(JMTIRN,3),
      + TMIRN(JMTIRN), PRPIRN(JMTIRN,9), MTIRN(JELIRN), NIRNEL(JIMRN),
      + CRDCL(JNDSH,3), CRDMC(JNDSH,3), GUSWT(JGAUS), GUSTR(JGAUS,3),
      + GUSCL(JELSH,JGAUS,3), DVDTL(JLDCA,JDTLD), NDDPD(JPRDS),
      + PRSDS(JPRDS,12), NREST(JPRDS),
      + NDCNL(JLDCA,JCNLD), TPNDI(JLDCA,JZNTP), TPNDE(JLDCA,JZNTP),
      + DRCOS(JCRSM,3,3), LD(JLD), NDJNC(JNDSH), CORLD(JDFSC),
      + BRHS(JDFSC), X(JDFSC)
C
       LOGICAL  PDCHEK, SINGAB
C
C
C
       COMMON  STRTYP, AUTGN, SLDSP, SLRAC, SLSTS, SLBKL,
      + PLSTR, PLMSH, PLDSP, PLSTS, PLMSP,
      + MSECT, MSTRN, MNDSR, MRING, MNDRN, MGAUS,
      + NNDSH, NELSH, NMTSH, NSTRN, NNDSR, NELSR, NMTSR, NIMSR,
      + NELISR, NMTISR, NRING, NNDRN, NELRN, NMTRN, NIMRN, NELIRN,
      + NMTIRN, NBRAC, NELBR, NMTBR, NGAUS, NCRSM, NPRDS, NLDCA,
      + NSECT, NRWEL, NRWND, NDFNC, NNDE1, NNDE2, NNDIN,
      + NNTSH, NBTSH, NGTSH, NSTSSH, NKCTND, NNTSR, NBTSR, NGTSR,
      + NTFSR, NSTSSR, NNTISR, NBTISR, NGTISR, NTFISR, NSTSIS, NNTRN,
      + NBTRN, NGTRN, NTFRN, NSTSRN, NNTIRN, NBTIRN, NGTIRN, NTFIRN,
      + NSTSIR, NTFBR, NSTSBR, NKRWPD, NCRLD, NFRCN, NABIF, NASCT,
      + NACNT, NFBSC, NRACT, NKSCT, NKSCF, NKKGW, NKGSC, NKGICL, NKBNIF,
      + NKGBNI, NEGVC, NKKGCN, NKGCN, NMSND, NMSCT, NKOEE, NKGOEE,
      + CSDCL, CSDMC, MCCOD, ILDCA, LCNLD, LDTLD, LZNTP, LZNSL,
      + DSKF1, DSKF2, IN, IO,
      + FEED, RTOP, RBAS, HITE, ELEV, THATAD,
      + TITLE(60), LDCAS(2), NCNLD(2), NDTLD(2), NZNTP(2), NZNSL(2),
      + RATIO(14), INDEX(45), NDCON(200), NDEX1(15), NDEX2(15),
      + NDINT(15), INTND(35), WKAREA(192)
C
       COMMON  / COMMN1 /
      + SLVSC, ROWSH, SHMTI, SHMTJ, CSDSG,
      + RSDOF, CSPRD, SCNLI, SCNLJ, CSCNL,
      + TYDTL, DSDTL, DRDTL, CSDTL, RFDTL,
      + SCTPI, SCTPJ, SEGND, NDDRD, IPNTP,
      + ICONN, ROWSA,
      + ORIGN, KCINV, KSECT, ITCNL,
      + ITDTL, ITWSH,
      + NODSH, MTDSH, DNSSH, PRPSH, MTSHL,
      + TSHEL, TRMSH, ARASH,
      + NODSR, MTDSR, ECNSR, TRMSR,
      + PRPSR, MTSTR, CLSTR, NDDISR, NDIMSR, MTDISR,
      + ECISR, TMISR, PRPISR, MTISR, CLISR, NODRN,
      + MTDRN, PSNRN, PRPRN, TRMRN, MTRNG
C
       COMMON  / COMMN2 /
      + NRNGEL, NODIRN, NDIMRN, MTDIRN, PSNIRN,
      + TMIRN, PRPIRN, MTIRN, NIRNEL,
```

```
     +  CRDCL, CRDMC, GUSWT, GUSTR, GUSCL,
     +  DVDTL, NODPD,
     +  PRSDS, NREST, NDCNL, TPNDI, TPNDE,
     +  DRCOS, LD, NDJNC, CORLD, BRHS, X
C
      COMMON  / COMMN3 /
     +  LRDF1, MNDSH, MELSH, MMTSH, MELSR,
     +  MMTSR, MIMSR, MELISR, MMTISR, MELRN, MMTRN,
     +  MIMRN, MELIRN, MMTIRN, MGUSS, MZNTP, MZNSL, MRWND,
     +  MRWEL, MCRSM, MPROS, MLDCA, MCNLD, MDTLD, MAULD, MDFSC, MDFIF,
     +  NDFIF, NDFEX, NDFIN, NDFSC, NINFA, NDFBL, NNDEX, NBEG, NEND, IOP,
     +  IBX, POCHEK, SINGAB, ICOND, IAFILE, IREF, DETCF, SLOLD, NDFSK
C
C***  GENERATE THE STIFFNES MATRICES OF ALL STRINGERS (I.E.,
C***  STRINGERS, INTERMEDIATE STRINGERS, RINGS, AND INTERMEDIATE RINGS)
C***  , AND ASSEMBLE THEM INTO KSECT, THE STIFNESS MATRIX OF THE
C***  SECTOR:
C
      CALL KSTFNR (MNDSH, MRWND, MRWEL, MELSR, MMTSR, MELISR,
     +  MMTISR, MELRN, MMTRN, MELIRN, MMTIRN, MCRSM, NDFSK, ROWSH,
     +  ORIGN, NODSR, ECNSR, PRPSR, MTSTR, CLSTR, NODISR, ECISR, PRPISR,
     +  MTISR, CLISR, NODRN, PSNRN, PRPRN, MTRNG, NRNGEL, NODIRN, PSNIRN,
     +  PRPIRN, MTIRN, CRDCL, CRDMC, DRCOS, NODPD, CSPRD, CSDSG, NDDRD,
     +  LD, NDJNC, KSECT)
C
      RETURN
      END
C
C*                                                                    *
C**                                                                  **
C***                                                                ***
C****          O V E R L A Y     1     4                           ****
C***                                                                ***
C**                                                                  **
C*                                                                    *
C
C
      SUBROUTINE ONEFOR
C
C
C*          I N T E R O V E R L A Y   P A R A M E T E R S
C
      PARAMETER  (JNDSH = 30,  JDFSC = 12 * JNDSH,
     +  JLD = JDFSC+1,  JSECT = 60,  JRWND = 15,  JRWEL = JRWND-1,
     +  JELSH = 28, JZNSH = 1,  JMTSH = 1,
     +  JSTRN = 1,   JNDSR = JRWND,  JELSR = JNDSR-1,
     +  JMTSR = 3,  JIMSR = 1,  JELISR = 5,  JMTISR = 1,  JRING = 2,
     +  JNDRN = 5,  JELRN = 3,  JMTRN = 2,  JIMRN = 15,  JELIRN = 20,
     +  JMTIRN = 10,  JGAUS = 12,  JCRSM = 5,  JPROS = JNDSH,
     +  JLDCA = 2,  JCNLD = 5,  JDTLD = 2,  JAULD = 1,  JZNTP = 1,
     +  JZNSL = 6,  JDFIF = 12 * JRWND,
```

```
     +  JICON = 109, JDFSL = 64801)
C
C
C
C                 I N T E R - O V E R L A Y    V A R I A B L E S
C
      CHARACTER*10  STRTYP, AUTGN, SLDSP, SLRAC, SLSTS, SLBKL,
     + PLSTR, PLMSH, PLDSP, PLSTS, PLMSP
C
      INTEGER  CSDCL, CSDMC, DSKF1, DSKF2, SLDLD,
     + SLVSC(JLDCA,JZNSL,2), ROWSH(JRWND,2),
     + SHMTI(JZNSH), SHMTJ(JZNSH), CSDSG(JCRSM), RSDOF(JPRDS,12),
     + CSPRD(JPRDS), SCNLI(JLDCA,JCNLD), SCNLJ(JLDCA,JCNLD),
     + CSCNL(JLDCA,JCNLD), TYDTL(JLDCA,JDTLD), DSDTL(JLDCA,JDTLD),
     + DRDTL(JLDCA,JDTLD,5), CSDTL(JLDCA,JDTLD), RFDTL(JLDCA,JDTLD),
     + SCTPI(JLDCA,JZNTP), SCTPJ(JLDCA,JZNTP), SEQND(JNDSH),
     + NDORD(JNDSH), IPNTR(JNDSH), ICONN(JICON), ROWSA(2)
C
      REAL  ORIGN(JRWEL,2), KCINV(JELSH,2,2), KSECT(JDFSL),
     + ITCNL(JLDCA,JCNLD,9), ITDTL(JLDCA,JDTLD,2), ITWSH(JMTSH,3)
C
      DIMENSION  NODSH(JELSH,3), MTDSH(JZNSH), DNSSH(JELSH),
     + PRPSH(JMTSH,8), MTSHL(JELSH),
     + TSHEL(JMTSH), TRMSH(JMTSH), ARASH(JELSH), NODSR(JELSR,2),
     + MTDSR(JMTSR), ECNSR(JMTSR), TRMSR(JMTSR),
     + PRPSR(JMTSR,9), MTSTR(JELSR), CLSTR(JELSR,7,2),
     + NODISR(JELISR,2), NDIMSR(JIMSR,2), MTDISR(JMTISR),
     + ECISR(JMTISR), TMISR(JMTISR),
     + PRPISR(JMTISR,9), MTISR(JELISR), CLISR(JELISR,7,2),
     + NODRN(JELRN,2), MTDRN(JMTRN), TRMRN(JMTRN), PSNRN(JMTRN,3),
     + PRPRN(JMTRN,9), MTRNG(JELRN), NRNGEL(JRING), NODIRN(JELIRN,2),
     + NDIMRN(JIMRN,2), MTDIRN(JMTIRN), PSNIRN(JMTIRN,3),
     + TMIRN(JMTIRN), PRPIRN(JMTIRN,9), MTIRN(JELIRN), NIRNEL(JIMRN),
     + CRDCL(JNDSH,3), CRDMC(JNDSH,3), GUSWT(JGAUS), GUSTR(JGAUS,3),
     + GUSCL(JELSH,JGAUS,3), DVDTL(JLDCA,JDTLD), NODPD(JPRDS),
     + PRSDS(JPRDS,12), NREST(JPRDS),
     + NDCNL(JLDCA,JCNLD), TPNDI(JLDCA,JZNTP), TPNDE(JLDCA,JZNTP),
     + DRCOS(JCRSM,3,3), LD(JLD), NDJNC(JNDSH), CORLD(JDFSC),
     + BRHS(JDFSC), X(JDFSC)
C
      LOGICAL  PDCHEK, SINGAB
C
C
C
      COMMON  STRTYP, AUTGN, SLDSP, SLRAC, SLSTS, SLBKL,
     + PLSTR, PLMSH, PLDSP, PLSTS, PLMSP,
     + MSECT, MSTRN, MNDSR, MRING, MNDRN, MGAUS,
     + NNDSH, NELSH, NMTSH, NSTRN, NNDSR, NELSR, NMTSR, NIMSR,
     + NELISR, NMTISR, NRING, NNDRN, NELRN, NMTRN, NIMRN, NELIRN,
     + NMTIRN, NSRAC, NELBR, NMTBR, NGAUS, NCRSM, NPRDS, NLDCA,
     + NSECT, NRWEL, NRWND, NDFND, NNDE1, NNDE2, NNDIN,
```

```
     +  NNTSH, NBTSH, NGTSH, NSTSSH, NKCTND, NNTSR, NBTSR, NGTSR,
     +  NTFSR, NSTSSR, NNTISR, NBTISR, NGTISR, NTFISR, NSTSIS, NNTRN,
     +  NBTRN, NGTRN, NTFRN, NSTSRN, NNTIRN, NBTIRN, NGTIRN, NTFIRN,
     +  NSTSIR, NTFBR, NSTSBR, NKRWPD, NCRLD, NFRCN, NABIF, NASCT,
     +  NACNT, NFBSC, NRACT, NKSCT, NKSCF, NKKGW, NKGSC, NKGICL, NKBNIF,
     +  NKGBNI, NEGVC, NKKGCN, NKGCN, NMSND, NMSCT, NKOEE, NKGOEE,
     +  CSDCL, CSDMC, MCCOD, ILDCA, LCNLD, LDTLD, LZNTP, LZNSL,
     +  OSKF1, OSKF2, IN, IO,
     +  FEED, RTOP, RBAS, HITE, ELEV, THATAD,
     +  TITLE(60), LDCAS(2), NCNLD(2), NDTLD(2), NZNTP(2), NZNSL(2),
     +  RATIO(14), INDEX(45), NOCON(200), NDEX1(15), NDEX2(15),
     +  NDINT(15), INTND(35), WKAREA(192)
C
      COMMON  / COMMN1 /
     +  SLVSC, ROWSH, SHMTI, SHMTJ, CSDSG,
     +  RSDOF, CSPRD, SCNLI, SCNLJ, CSCNL,
     +  TYDTL, DSDTL, DRDTL, CSDTL, RFDTL,
     +  SCTPI, SCTPJ, SEQND, NDORD, IPNTR,
     +  ICONN, ROWSA,
     +  ORIGN, KCINV, KSECT, ITCNL,
     +  ITDTL, ITWSH,
     +  NODSH, MTDSH, DNSSH, PRPSH, MTSHL,
     +  TSHEL, TRMSH, ARASH,
     +  NODSR, MTDSR, ECNSR, TRMSR,
     +  PRPSR, MTSTR, CLSTR, NODISR, NDIMSR, MTDISR,
     +  ECISR, TMISR, PRPISR, MTISR, CLISR, NODRN,
     +  MTDRN, PSNRN, PRPRN, TRMRN, MTRNG
C
      COMMON  / COMMN2 /
     +  NRNGEL, NODIRN, NDIMRN, MTDIRN, PSNIRN,
     +  TMIRN, PRPIRN, MTIRN, NIRNEL,
     +  CRDCL, CRDMC, GUSWT, GUSTR, GUSCL,
     +  DVDTL, NODPD,
     +  PRSDS, NREST, NDCNL, TPNDI, TPNDE,
     +  DRCOS, LD, NDJNC, CORLD, BRHS, X
C
      COMMON  / COMMN3 /
     +  LRDF1, MNDSH, MELSH, MMTSH, MELSR,
     +  MMTSR, MIMSR, MELISR, MMTISR, MELRN, MMTRN,
     +  MIMRN, MELIRN, MMTIRN, MGUSS, MZNTP, MZNSL, MRWND,
     +  MRWEL, MCRSM, MPRDS, MLDCA, MCNLD, MDTLD, MAULD, MDFSC, MDFIF,
     +  NDFIF, NDFEX, NDFIN, NDFSC, NINFA, NDFBL, NNDEX, NBEG, NEND, IOP,
     +  IBX, PDCHEK, SINGAB, ICOND, IAFILE, IREF, DETCF, SLDLD, NDFSK
C
C
C
C***  PUT THE NONZERO ENTRIES OF KSECT CORRESPONDING TO A CONSTRAINED
C***  DEGREE OF FREEDOM ON DISK DSKF1 (ROW BY ROW).
C***  THEN CONSTRUCT THE CORRECTIVE LOAD VECTOR CORLD. THEN
C***  SET THE NONDIAGONAL ENTRIES OF KSECT CORRESPONDING TO THE
C***  PRESCRIBED DISPLACEMENTS EQUAL TO ZERO ( THE CORRESPONDING
```

```
C***  DIAGONAL ENTRIES ARE SET TO ONE ). THEN CONDENSE OUT THE
C***  INTERNAL STIFFNESS MATRIX.
C
      CALL MDFKSC (LRDF1, MPRDS, MDFIF, NNDEX, NDFBL, NDFEX,
     + NDFIN, NDFSC, NDFSK, NBEG, NEND, IOP, IBX, PDCHEK, SINGAB, ICOND,
     + IAFILE, IREF, DETCF, NDOPD, PRSDS, RSDOF, NREST, NDORD, SEQND,
     + LD, IPNTR, ICONN, CORLD, KSECT, BRHS, X)
C
C
      RETURN
      END
C
C
C*                                                                    *
C**                                                                  **
C***          O V E R L A Y      2      0                            ****
C**                                                                  **
C*                                                                    *
C
C
C
      SUBROUTINE TWOZRO
C
C
C*          I N T E R O V E R L A Y    P A R A M E T E R S
C
      PARAMETER  (JNDSH = 30,  JDFSC = 12 * JNDSH,
     + JLD = JDFSC+1,  JSECT = 60,  JRWND = 15,  JRWEL = JRWND-1,
     + JELSH = 28, JZNSH = 1,  JMTSH = 1,
     + JSTRN = 1,   JNDSR = JRWND,  JELSR = JNDSR-1,
     + JMTSR = 3,  JIMSR = 1,  JELISR = 5,  JMTISR = 1,  JRING = 2,
     + JNDRN = 5,  JELRN = 3,  JMTRN = 2,  JIMRN = 15,  JELIRN = 20,
     + JMTIRN = 10,  JGAUS = 12,  JCRSM = 5,  JPRDS = JNDSH,
     + JLOCA = 2,  JCNLD = 5,  JDTLD = 2,  JAULD = 1,  JZNTP = 1,
     + JZNSL = 6,  JDFIF = 12 * JRWND,
     + JICON = 109, JDFSL = 64801)
C
C
C*             L O C A L    V A R I A B L E S
C
C
C
      REAL  NTSHL(JGAUS,3,38), NTSTF(7,5,22)
C
      DIMENSION  FRCNL(JDFIF)      .
C
C
C
C*          I N T E R - O V E R L A Y    V A R I A B L E S
C
C*             BETWEEN  OVERLAYS   (0,0), (1,0), (2,0)
```

```
      C
            CHARACTER*10  STRTYP, AUTGN, SLDSP, SLRAC, SLSTS, SLBKL,
           + PLSTR, PLMSH, PLDSP, PLSTS, PLMSP
      C
            INTEGER  CSDCL, CSDMC, DSKF1, DSKF2, SLDLD,
           + SLVSC(JLDCA,JZNSL,2), ROWSH(JRWND,2),
           + SHMTI(JZNSH), SHMTJ(JZNSH), CSDSG(JCRSM), RSDOF(JPRDS,12),
           + CSPRO(JPRDS), SCNLI(JLDCA,JCNLD), SCNLJ(JLDCA,JCNLD),
           + CSCNL(JLDCA,JCNLD), TYDTL(JLDCA,JDTLD), DSDTL(JLDCA,JDTLD),
           + DRDTL(JLDCA,JDTLD,5), CSDTL(JLDCA,JDTLD), RFDTL(JLDCA,JDTLD),
           + SCTPI(JLDCA,JZNTP), SCTPJ(JLDCA,JZNTP), SEQND(JNDSH),
           + NDORD(JNDSH), IPNTR(JNDSH), ICONN(JICON), ROWSA(2)
      C
            REAL  ORIGN(JRWEL,2), KCINV(JELSH,2,2), KSECT(JDFSL),
           + ITCNL(JLDCA,JCNLD,9), ITDTL(JLDCA,JDTLD,2), ITWSH(JMTSH,3)
      C
            DIMENSION  NDDSH(JELSH,3), MTDSH(JZNSH), DNSSH(JELSH),
           + PRPSH(JMTSH,8), MTSHL(JELSH),
           + TSHEL(JMTSH), TRMSH(JMTSH), ARASH(JELSH), NDDSR(JELSR,2),
           + MTDSR(JMTSR), ECNSR(JMTSR), TRMSR(JMTSR),
           + PRPSR(JMTSR,9), MTSTR(JELSR), CLSTR(JELSR,7,2),
           + NDDISR(JELISR,2), NDIMSR(JIMSR,2), MTDISR(JMTISR),
           + ECISR(JMTISR), TMISR(JMTISR),
           + PRPISR(JMTISR,9), MTISR(JELISR), CLISR(JELISR,7,2),
           + NDDRN(JELRN,2), MTDRN(JMTRN), TRMRN(JMTRN), PSNRN(JMTRN,3),
           + PRPRN(JMTRN,9), MTRNG(JELRN), NRNGEL(JRING), NDDIRN(JELIRN,2),
           + NDIMRN(JIMRN,2), MTDIRN(JMTIRN), PSNIRN(JMTIRN,3),
           + TMIRN(JMTIRN), PRPIRN(JMTIRN,9), MTIRN(JELIRN), NIRNEL(JIMRN),
           + CRDCL(JNDSH,3), CRDMC(JNDSH,3), GUSWT(JGAUS), GUSTR(JGAUS,3),
           + GUSCL(JELSH,JGAUS,3), DVDTL(JLDCA,JDTLD), NDDPD(JPRDS),
           + PRSDS(JPRDS,12), NREST(JPRDS),
           + NDCNL(JLDCA,JCNLD), TPNDI(JLDCA,JZNTP), TPNDE(JLDCA,JZNTP),
           + DRCOS(JCRSM,3,3), LD(JLD), NDJNC(JNDSH), CORLD(JDFSC),
           + BRHS(JDFSC), X(JDFSC)
      C
      C
      C       BETWEEN OVERLAYS  (2,0), (2,1), (2,2), (2,3), (2,4), (2,5), (2,6)
      C
            DIMENSION  ENLWT(JDFSC), ECLWT(JELSH,12), ENLAUL(JAULD,JDFSC),
           + ECLAUL(JAULD,JELSH,12), ENLTP(JZNTP,JDFSC),
           + ECLTP(JZNTP,JELSH,12), INDIC(JSECT)
      C
            COMPLEX  ENLSA(4,JDFSC), ECLSA(4,JELSH,12), KBNIF(JDFIF,JDFIF)
      C
            LOGICAL  PDCHEK, SINGAB
      C
      C
            EQUIVALENCE  ( NTSHL(1,1,1), NTSTF(1,1,1) )
      C
      C
      C
```

```
      COMMON  STRTYP, AUTGN, SLDSP, SLRAC, SLSTS, SLBKL,
     + PLSTR, PLMSH, PLDSP, PLSTS, PLMSP,
     + MSECT, MSTRN, MNDSR, MRING, MNDRN, MGAUS,
     + NNDSH, NELSH, NMTSH, NSTRN, NNDSR, NELSR, NMTSR, NIMSR,
     + NELISR, NMTISR, NRING, NNDRN, NELRN, NMTRN, NIMRN, NELIRN,
     + NMTIRN, NBRAC, NELBR, NMTBR, NGAUS, NCRSM, NPRDS, NLDCA,
     + NSECT, NRWEL, NRWND, NDFND, NNDE1, NNDE2, NNDIN,
     + NNTSH, NBTSH, NGTSH, NSTSSH, NKCTND, NNTSR, NBTSR, NGTSR,
     + NTFSR, NSTSSR, NNTISR, NBTISR, NGTISR, NTFISR, NSTSIS, NNTRN,
     + NBTRN, NGTRN, NTFRN, NSTSRN, NNTIRN, NBTIRN, NGTIRN, NTFIRN,
     + NSTSIR, NTFBR, NSTSBR, NKRWPD, NCRLD, NFRCN, NABIF, NASCT,
     + NACNT, NFBSC, NRACT, NKSCT, NKSCF, NKKGW, NKGSC, NKGICL, NKBNIF,
     + NKGBNI, NEGVC, NKKGCN, NKGCN, NMSND, NMSCT, NK0EE, NKG0EE,
     + CSDCL, CSDMC, MCCOD, ILDCA, LCNLD, LDTLD, LZNTP, LZNSL,
     + DSKF1, DSKF2, IN, IO,
     + FEED, RTOP, RBAS, HITE, ELEV, THATAD,
     + TITLE(60), LDCAS(2), NCNLD(2), NDTLD(2), NZNTP(2), NZNSL(2),
     + RATIO(14), INDEX(45), NDCON(200), NDEX1(15), NDEX2(15),
     + NDINT(15), INTND(35), WKAREA(192)
C
C
      COMMON  / COMMN1 /
     + SLVSC, ROWSH, SHMTI, SHMTJ, CSDSG,
     + RSDOF, CSPRD, SCNLI, SCNLJ, C[illegible],
     + TYDTL, DSDTL, DRDTL, ESDTL, RFDTL,
     + SCTPI, SCTPJ, S[illegible], NCCRD, IPNTR,
     + ICONN, ROWSA,
     + DRIGN, KCINV, KSECT, ITCNL,
     + ITDTL, ITWSH,
     + NCDSH, MTDSH, DNSIH, RFRSH, ATSHL,
     + TSHEL, TRMSH, ARASH,
     + NCDSR, MTDSR, RCNDX, TRMIN,
     + PRPSR, MTSTR, CLSTR, NCDISR, NCDMSR, MTDISR,
     + ECISR, TMISH, [illegible], MTDIR, DIFF, DN,
     + MTORN, PSURN, PRPRN, TCMAD, MTHNG
C
      COMMON  / COMMN2 /
     + [illegible], [illegible], [illegible], MTORN, TMIRN,
     + TMIRN, PRPIRN, MTIRN, NIRWEL,
     + CRDCL, CRDMC, SUSAT, SUSTY, RWICL,
     + DVDTL, NCDPD,
     + PRSDS, NRFST, NLCHL, TRNLL, TRNLA,
     + CRCOS, LD, NDJNC, CORLD, ERHF,
C
      COMMON  / COMMN3 /
     + LKOPL, KLSH, [illegible], RTDR, MLA,
     + MMTSR, MIMSR, MELISR, MMTISR, MELSN, MMTRN,
     + MIMRN, MELIRN, MMTIRN, MNDSS, MCLTP, MIMSL, MRWND,
     + MRWEL, MCRSM, MPRDS, MLDCA, MCNLD, MDTLD, MAULD, MDFSC, MDFIF,
     + MDFIF, NDFLY, [illegible], [illegible], [illegible], [illegible], MSECT, MSND, DIP,
     + IBX, PDCMEX, SINGAB, ICOND, IFFILE, IHCF, MEICR, SLOLD, NDFSK
```

```
1071  C
1072  C
1073  C     BETWEEN OVERLAYS  (2,0), (2,1), (2,2), (2,3), (2,4), (2,5), (2,6)
1074  C
1075        COMMON  / COMMNA /  NOPCN, NEIG, ENLWT, ECLWT, ENLAUL, ECLAUL,
1076       +  ENLTP, ECLTP, ENLSA, ECLSA, IMDIC, KBNIP
1077  C
1078  C
1079  C
1080   1000 IF (NLDCA . EQ . 0)                              GO TO 23500
1081  C
1082  C*    IF BUCKLING SOLUTINE HAS BEEN REQUESTED, GO TO THE APPROPRIATE
1083  C*    PROGRAM SEGMENT.
1084  C
1085        IF (SLBKL . EQ . 'YES')                           GO TO 6500
1086  C
1087  C
1088  C                S T R E S S     A N A L Y S I S
1089  C
1090  C
1091  C
1092  C*    CONSIDER EACH LOADING CASE.
1093  C
1094  C
1095        DO 6000 LLDCA = 1 , NLDCA
1096  C
1097        ILDCA = LLDCA
1098        LCNLD = NCNLD(LLDCA)
1099        LDTLD = NDTLD(LLDCA)
1100        LZNTP = NZNTP(LLDCA)
1101        LZNSL = NZNSL(LLDCA)
1102  C
1103  C
1104  C***  IF THE LOADING CASE NUMBER IS GREATER THAN 1, AND PLOTTING OF THE
1105  C***  DISPLACEMENTS AND / OR STRESSES HAS BEEN REQUESTED, READ THE
1106  C***  CONVENTIONAL STIFFNESS MATRIX FROM DISK FILE DSKF1.
1107  C
1108        IF (ILDCA . EQ . 1)                               GO TO 1500
1109  C
1110        IF (PLDSP . NE . 'YES' . AND . PLSTS . NE . 'YES') GO TO 1500
1111  C
1112        IF (NNDIN . NE . 0)                               GO TO 1250
1113  C
1114        CALL REDKSC (DSKF1, NKSCT, 0, LRDF1, NDFSK, KSECT)
1115  C
1116        GO TO 1500
1117  C
1118   1250 CALL REDKSC (DSKF1, NKSCF, 0, LRDF1, NDFSK, KSECT)
1119  C
1120   1500 IF (LCNLD . EQ . 0)                               GO TO 2500
1121  C
```

```
1122  C          C O N C E N T R A T E D    N O D A L    L O A D S
1123  C
1124  C
1125  C***  SET UP PR'S, THE SET OF VECTORS CONTAINING THE PRESCRIBED
1126  C***  CONCENTRATED NODAL LOADS FOR ALL INTERFACES AND PUT THEM ON
1127  C***  DISK FILE DSKF1.
1128  C
1129        CALL NDLDCN (CSDMC, DSKF1, ILDCA, LCNLD, MNDSH, MELSR,
1130       + MELISR, MELRN, MELIRN, MPRDS, MCRSM, MLDCA, MCNLD, MRWND,
1131       + MRWEL, MCCOD, NELSH, NELSR, NELISR, NELRN, NELIRN, NCRSM, NPRDS,
1132       + NLDCA, NSECT, NRWND, NNDIN, NDFIF, NDFSC, NFRCN, STRTYP, NCNLD,
1133       + CRDCL, CRDMC, NODSR, NODISR, NODRN, NODIRN, ROWSH, ORIGN, NDEX1,
1134       + NDEX2, NDINT, INTND, NDORD, CSDSG, NODPD, RSDOF, NREST,
1135       + CSPRD, SCNLI, SCNLJ, NDCNL, ITCNL, CSCNL, DRCOS, LD, KSECT,
1136       + ENLWT, ENLAUL, ENLSA, FRCNL, WKAREA)
1137  C
1138  C*    IF NO SHELL ELEMENTS EXIST, SKIP.
1139  C
1140   2500 IF (NELSH . EQ . 0)                          GO TO 3500
1141  C
1142  C*                                                              *
1143  C**                                                            **
1144  C***     CALCULATE THE EQUIVALENT NODAL LOADS FOR ALL          ***
1145  C***     SHELL ELEMENTS.                                       ***
1146  C**                                                            **
1147  C*                                                              *
1148  C
1149        CALL TWODNE
1150  C
1151  C
1152  C
1153  C*    IF NO STIFFENERS (STRINGER, INTERMEDIATE STRINGER, RING,
1154  C*    INTERMEDIATE RING, OR BRACE) ELEMENTS EXIST, SKIP.
1155  C
1156   3500 IF (NELSR . EQ . 0. AND . NELISR . EQ . 0 . AND .
1157       + NELRN . EQ . 0 . AND . NELIRN . EQ . 0 . AND . NELBR . EQ . 0)
1158       +                                          GO TO 4500
1159  C
1160  C
1161  C*                                                              *
1162  C**                                                            **
1163  C***     CALCULATE THE EQUIVALENT NODAL LOADS FOR ALL          ***
1164  C***     STIFFENER ELEMENTS.                                   ***
1165  C**                                                            **
1166  C*                                                              *
1167  C
1168        CALL TWOTWO
1169  C
1170  C
1171  C*                                                              *
1172  C**                                                            **
```

```
1173  C***      CALCULATE THE DISPLACEMENTS, REACTIONS, AND THE                ***
1174  C***      THE STRESSES.                                                 ***
1175  C**                                                                     **
1176  C*                                                                      *
1177  C
1178   4500 CALL TWOTHR
1179  C
1180  C
1181  C
1182  C*     IF PLOTTING OF DISPLACEMENTS AND STRESSES HAS NOT BEEN REQUESTED,
1183  C*     SKIP.
1184  C
1185         IF (PLDSP . NE . 'YES' . AND . PLSTS . NE . 'YES')  GO TO 6000
1186  C
1187  C
1188  C*                                                                      *
1189  C**                                                                     **
1190  C***      PLOT THE DISPLACEMENTS AND / OR STRESSES DUE TO DEAD          ***
1191  C***      AND LIVE LOADS COMBINED.                                      ***
1192  C**                                                                     **
1193  C*                                                                      *
1194  C
1195         CALL TWOSIX
1196  C
1197  C
1198   6000 CONTINUE
1199  C
1200         RETURN
1201  C
1202  C
1203  C
1204  C
1205  C*     SET UP THE EIGENVALUE PROBLEM:
1206  C                (K+KGDLD) - LAMBDA * KGLLD = 0
1207  C
1208  C*     WHERE
1209  C*     K IS THE CONVENTIONAL STIFFNESS MATRIX OF A SECTOR,
1210  C*     KGDLD IS THE GEOMETRIC STIFFNESS DUE TO DEAD LOAD ALONE,
1211  C*     KGLLD IS THE GEOMETRIC STIFFNESS MATRIX DUE TO LIVE LOAD ALONE.
1212  C
1213  C
1214   6500 DO 23000 LLDCA = 1 , NLDCA
1215  C
1216         ILDCA = LLDCA
1217  C
1218  C***   IF THE LOADING CASE NUMBER IS GREATER THAN 1, AND PLOTTING OF THE
1219  C***   DISPLACEMENTS AND / OR STRESSES HAS BEEN REQUESTED, READ THE
1220  C***   CONVENTIONAL STIFFNESS MATRIX FROM DISK FILE DSKF1.
1221  C
1222         IF (ILDCA . EQ . 1)                              GO TO 6750
1223  C
```

```
      IF (PLDSP . NE . 'YES' . AND . PLSTS . NE . 'YES') GO TO 6750
C
      CALL REDKSC (DSKF1, NKSCT, 0, LRDF1, NDFSK, KSECT)
C
C*    FIND THE NUMBER OF ZONES OF SECTORS FOR WHICH SOLUTION HAS BEEN
C*    REQUESTED.
C
 6750 LZNSL = NZNSL(LLDCA)
C
C
C                          D E A D     L O A D
C
C*    CALCULATE AND SAVE (ON LOCAL DISK FILE) THE GEOMETRIC STIFFNESS
C*    MATRIX DUE TO THE DEAD LOAD ONLY ONCE.
C
      IF (LLDCA . GT . 1)                              GO TO 13500
C
C*    INITIALIZATIONS.
C
C*    SET THE NUMBER OF THE CONCENTRATED NODAL LOADS, THE DISTRIBUTED
C*    LOADS, AND THE TEMPERATURE ZONES EQUAL TO ZERO, TEMPORARILY, FOR
C*    THE CURRENT DEAD LOAD ANALYSIS.
C
      LCNLD = 0
      LDTLD = 0
      LZNTP = 0
C
C*    SIGNAL DEAD LOAD SOLUTION.
C
      SLDLD = 3HYES
C
       IKGOEE = 0
C
C
C
C     KGSCT(NDFSC) DUE TO DEAD LOAD.
C
C
C***  CALCULATE THE EQUIVALENT NODAL LOADS OF A TYPICAL SECTOR DUE
C***  TO DEAD LOAD OF THE SHELL AND THE STIFFENER ELEMENTS,
C***  ENLWT(NDFSC).
C
      CALL ENLDLD (MNDSH, MELSH, MMTSH, MELSR, MMTSR, MELISR,
     + MMTISR, MELRN, MMTRN, MELIRN, MMTIRN, MCRSM, MLDCA, MCNLD, MDTLD,
     + MGUSS, MRWND, MRWEL, NDFSC, NODSH, MTSHL, TSHEL, DNSSH, ARASH,
     + CRDCL, CRDMC, ROWSH, ORIGN, NODSR, PRPSR, MTSTR, CLSTR, NODISR,
     + PRPISR, MTISR, CLISR, NODRN, PSNRN, PRPRN, MTRNG, NODIRN, PSNIRN,
     + PRPIRN, MTIRN, NDORD, CSDSG, DRCOS, ITWSH, NDJNC, GUSWT, GUSCL,
     + NDCNL, ITCNL, CSCNL, ITDTL, DRDTL, CSDTL, KCINV, NTSHL, NTSTF,
     + ENLWT, ECLWT)
C
```

```
C
C*    FOR THE FIRST LOADING CASE, VECTOR KSECT CONTAINS THE DESIRED
C*    CONVENTIONAL STIFFNESS MATRIX. I.E.,
C*    UNFACTORIZED IF NO INTERNAL NODES EXIST, AND
C*    FACTORIZED IF INTERNAL NODES EXIST.
C
      IF (ILDCA . EQ . 1)                              GO TO 8500
C
      IF (NNDIN . NE . 0)                              GO TO 7500
C
C***  IF NO INTERNAL NODES EXIST, READ THE UNFACTORIZED CONVENTIONAL
C***  STIFFNESS MATRIX FROM DISK FILE DSKF1.
C
      CALL REDKSC (DSKF1, NKSCT, 0, LRDF1, NDFSK, KSECT)
C
      GO TO 8500
C
C***  WHEN INTERNAL NODES DO EXIST, READ THE FACTORIZED CONVENTIONAL
C***  STIFFNESS MATRIX FROM DISK FILE DSKF1.
C
 7500 CALL REDKSC (DSKF1, NKSCF, 0, LRDF1, NDFSK, KSECT)
C
C
C*                                                                    *
C**                                                                  **
C***     CALCULATE THE DISPLACEMENTS, REACTIONS, AND THE            ***
C***     THE STRESSES DUE TO THE DEAD LOAD ALONE.                   ***
C**                                                                  **
C*                                                                    *
C
 8500 CALL TWOTHR
C
C
C*    IF NO INTERNAL NODE EXISTS, KSECT ALREADY CONTAINS THE
C*    UNFACTORIZED CONVENTIONAL STIFFNESS MATRIX.
C
      IF (NNDIN . EQ . 0)                              GO TO 9500
C
C
C***  INITIALIZE KSECT(NDFSC) TO THE UNFACTORIZED CONVENTIONAL
C***  SECTOR STIFFNESS MATRIX MATRIX BY READING IT FROM DISK FILE DSKF1.
C
      CALL REDKSC (DSKF1, NKSCT, 0, LRDF1, NDFSK, KSECT)
C
C*                                                                    *
C**                                                                  **
C***     CALCULATE THE GEOMETRIC STIFFNESS MATRIX OF THE SECTOR     ***
C***     DUE TO (CIRCUMFERENTIALLY UNIFORM) DEAD LOADS.             ***
C***     ON RETURN, KSECT = K + KGOLD. I.E., KSECT CONTAINS         ***
C***     THE SUM OF THE CONVENTIONAL STIFFNESS MATRIX AND THE       ***
C***     GEOMETRIC STIFFNESS MATRIX DUE TO THE DEAD LOAD.           ***
```

```
C**                                                                  **
C*                                                                    *
C
 9500 CALL TWOFOR
C
C
C
C
C
C
C
C
C*    SIGNAL LIVE LOAD SOLUTION.
C
13500 SLDLD = 2HNO
C
C
C*    FIND THE NUMBER OF THE CONCENTRATED NODAL LOADS, THE DISTRIBUTED
C*    LOADS, AND THE ZONES OF TEMPERATUER, FOR THE CURRENT LOAD CASE.
C
      LCNLD = NCNLD(LLDCA)
      LDTLD = NDTLD(LLDCA)
      LZNTP = NZNTP(LLDCA)
C
C
      IF (LCNLD . EQ . 0)                          GO TO 14000
C
C           C O N C E N T R A T E D   N O D A L   L O A D S
C
C***  SET UP PR'S, THE SET OF VECTORS CONTAINING THE PRESCRIBED
C***  CONCENTRATED NODAL LOADS FOR ALL INTERFACES AND PUT THEM ON
C***  DISK FILE DSKF1.
C
      CALL NDLDCN (CSDMC, DSKF1, ILDCA, LCNLD, MNDSH, MELSR,
     + MELISR, MELRN, MELIRN, MPRDS, MCRSM, MLDCA, MCNLD, MRWND,
     + MRWEL, MCCOD, NELSH, NELSR, NELISR, NELRN, NELIRN, NCRSM, NPRDS,
     + NLDCA, NSECT, NRWND, NNDIN, NDFIF, NDFSC, NFRCN, STRTYP, NCNLD,
     + CRDCL, CRDMC, NDDSR, NDDISR, NDDRN, NDDIRN, ROWSH, ORIGN, NDEX1,
     + NDEX2, NDINT, INTND, NDORD, CSDSG, NDDPD, RSDOF, NREST,
     + CSPRD, SCNLI, SCNLJ, NDCNL, ITCNL, CSCNL, DRCOS, LD, KSECT,
     + ENLWT, ENLAUL, ENLSA, FRCNL, WKAREA)
C
C
C*    SET VECTOR ENLWT, JUST USED AS A TEMPORARY WORKING AREA,
C*    EQUAL TO ZERO, SIGNIFYING THAT NO DEAD LOAD IS TO BE CONSIDERED
C*    IN THIS PART OF THE ANALYSIS.
C
14000 CALL SETZRO (NDFSC, 1, ENLWT)
C
```

```
C
C*    IF NO SHELL ELEMENTS EXIST, SKIP.
C
      IF (NELSH . EQ . 0)                                  GO TO 14500
C
C*                                                                    *
C**                                                                   **
C***     CALCULATE THE EQUIVALENT NODAL LOADS DUE TO LIVE             ***
C***     LOADS FOR ALL SHELL ELEMENTS.                                ***
C**                                                                   **
C*                                                                    *
C
      CALL TWODNE
C
C
14500 IF (NNDIN . NE . 0)                                  GO TO 15500
C
C***  IF NO INTERNAL NODES EXIST, READ THE UNFACTORIZED CONVENTIONAL
C***  STIFFNESS MATRIX FROM DISK FILE DSKF1.
C
      CALL REDKSC (DSKF1, NKSCT, 0, LRDF1, NDFSK, KSECT)
C
      GO TO 16500
C
C***  WHEN INTERNAL NODES DO EXIST, READ THE FACTORIZED CONVENTIONAL
C***  STIFFNESS MATRIX FROM DISK FILE DSKF1.
C
15500 CALL REDKSC (DSKF1, NKSCF, 0, LRDF1, NDFSK, KSECT)
C
C*                                                                    *
C**                                                                   **
C***     CALCULATE THE DISPLACEMENTS, REACTIONS, AND THE              ***
C***     THE STRESSES.                                                ***
C**                                                                   **
C*                                                                    *
C
16500 CALL TWOTHR
C
C
C*    INITIALIZE KSECT (PASSED TO KGSCT IN THE SUBROUTINES) TO ZERO
C*    BEFORE CALCULATING THE GEOMETRIC STIFFNESS MATRIX.
C
      CALL SETZRO (NDFSK, 1, KSECT)
C
C
C
C
C*                                                                    *
C**                                                                   **
C***     CALCULATE THE GEOMETRIC STIFFNESS MATRIX OF THE SECTOR       ***
C***     DUE TO (CIRCUMFERENTIALLY UNIFORM) LIVE LOADS.               ***
```

```
1428  C**                                                                **
1429  C*                                                                 *
1430  C
1431        CALL TWOFOR
1432  C
1433  C
1434  C*    CALCULATE LAMBDA DEFINED IN THE EIGENVALUE PROBLEM:
1435  C         (KSECT + KGDLD) - LAMBDA * KGLLD = 0.0
1436  C*    BY SOLVING
1437  C      KSNIF(NDFIF, NDFIF) - LAMBDA * KGSNIF(NDFIF, NDFIF) = 0.0
1438  C
1439  C
1440  C
1441  C*                                                                 *
1442  C**                                                                **
1443  C***     SOLVE FOR THE BUCKLING LOAD AND THE ASSOCIATED            ***
1444  C***     MODESHAPES.                                               ***
1445  C**                                                                **
1446  C*                                                                 *
1447  C
1448        CALL TWOFIV
1449  C
1450  C
1451  C
1452  C*    IF PLOTTING OF DISPLACEMENTS AND STRESSES AND MODESHAPES HAS NOT
1453  C*    BEEN REQUESTED, SKIP.
1454  C
1455        IF (PLDSP . NE . 'YES' . AND . PLSTS . NE . 'YES' . AND .
1456       +  PLMSP . NE . 'YES')                          GO TO 23000
1457  C
1458  C
1459  C*                                                                 *
1460  C**                                                                **
1461  C***     PLOT THE DISPLACEMENTS AND / OR STRESSES AND / OR          ***
1462  C***     MODESHAPES DUE TO LIVE LOADS.                             ***
1463  C**                                                                **
1464  C*                                                                 *
1465  C
1466        CALL TWOSIX
1467  C
1468  C
1469  C
1470  23000 CONTINUE
1471  C
1472  C
1473  23500 RETURN
1474        END
1475  C
1476  C
1477  C*                                                                 *
1478  C**                                                                **
```

```
C***                  O V E R L A Y      2      1                    ***
C**                                                                  **
C*                                                                    *
C
C
C
      SUBROUTINE TWOONE
C
C
C*              I N T E R O V E R L A Y    P A R A M E T E R S
C
      PARAMETER (JNDSH = 30,  JDFSC = 12 * JNDSH,
     + JLD = JDFSC+1,  JSECT = 60,  JRWND = 15,  JRWEL = JRWND-1,
     + JELSH = 28, JZNSH = 1,  JMTSH = 1,
     + JSTRN = 1,   JNDSR = JRWND,  JELSR = JNDSR-1,
     + JMTSR = 3,  JIMSR = 1,  JELISR = 5,  JMTISR = 1,  JRING = 2,
     + JNDRN = 5,  JELRN = 3,  JMTRN = 2,  JIMRN = 15,  JELIRN = 20,
     + JMTIRN = 10,  JGAUS = 12,  JCRSM = 5,  JPRDS = JNDSH,
     + JLDCA = 2,  JCNLD = 5,  JDTLD = 2,  JAULD = 1,  JZNTP = 1,
     + JZNSL = 6,  JDFIF = 12 * JRWND,
     + JICON = 109, JDFSL = 64801)
C
C
C
C                    L O C A L    V A R I A B L E S
C
      REAL  NTSHL(JGAUS,3,38)
C
      DIMENSION  BTSHL(JGAUS,6,38), CURVT(JGAUS,5),
     + STSTP(JZNTP,JGAUS,6)
C
C              I N T E R - O V E R L A Y    V A R I A B L E S
C
C                  BETWEEN  OVERLAYS   (0,0), (1,0), (2,0)
C
      CHARACTER*10  STRTYP, AUTGN, SLDSP, SLPAC, SLSTS, SLBKL,
     + PLSTR, PLMSH, PLDSP, PLSTS, PLMSP
C
      INTEGER  CSDCL, CSDMC, DSKF1, DSKF2, SLDLD,
     + SLVSC(JLDCA,JZNSL,2), ROWSH(JRWND,2),
     + SHMTI(JZNSH), SHMTJ(JZNSH), CSDSG(JCRSM), RSDDF(JPRDS,12),
     + CSPRD(JPRDS), SCNLI(JLDCA,JCNLD), SCNLJ(JLDCA,JCNLD),
     + CSCNL(JLDCA,JCNLD), TYDTL(JLDCA,JDTLD), DSDTL(JLDCA,JDTLD),
     + DRDTL(JLDCA,JDTLD,5), CSDTL(JLDCA,JDTLD), RFDTL(JLDCA,JDTLD),
     + SCTPI(JLDCA,JZNTP), SCTPJ(JLDCA,JZNTP), SEQND(JNDSH),
     + NDORD(JNDSH), IPNTR(JNDSH), ICONN(JICON), ROWSA(2)
C
      REAL  ORIGN(JRWEL,2), KCINV(JELSH,2,2), KSECT(JDFSL),
     + ITCNL(JLDCA,JCNLD,9), ITDTL(JLDCA,JDTLD,2), ITWSH(JMTSH,3)
C
      DIMENSION  NODSH(JELSH,3), MTDSH(JZNSH), DNSSH(JELSH),
```

```
     +  PRPSH(JMTSH,8), MTSHL(JELSH),
     +  TSHEL(JMTSH), TRMSH(JMTSH), ARASH(JELSH), NODSR(JELSR,2),
     +  MTDSR(JMTSR), ECNSR(JMTSR), TRMSR(JMTSR),
     +  PRPSR(JMTSR,9), MTSTR(JELSR), CLSTR(JELSR,7,2),
     +  NDDISR(JELISR,2), NDIMSR(JIMSR,2), MTDISR(JMTISR),
     +  ECISR(JMTISR), TMISR(JMTISR),
     +  PRPISR(JMTISR,9), MTISR(JELISR), CLISR(JELISR,7,2),
     +  NODRN(JELRN,2), MTDRN(JMTRN), TRMRN(JMTRN), PSNRN(JMTRN,3),
     +  PRPRN(JMTRN,9), MTRNG(JELRN), NRNGEL(JRING), NDDIRN(JELIRN,2),
     +  NDIMRN(JIMRN,2), MTDIRN(JMTIRN), PSNIRN(JMTIRN,3),
     +  TMIRN(JMTIRN), PRPIRN(JMTIRN,9), MTIRN(JELIRN), NIRNEL(JIMRN),
     +  CRDCL(JNDSH,3), CRDMC(JNDSH,3), GUSWT(JGAUS), GUSTR(JGAUS,3),
     +  GUSCL(JELSH,JGAUS,3), DVDTL(JLDCA,JDTLD), NODPD(JPRDS),
     +  PRSDS(JPRDS,12), NREST(JPRDS),
     +  NDCNL(JLDCA,JCNLD), TPNDI(JLDCA,JZNTP), TPNDE(JLDCA,JZNTP),
     +  DRCDS(JCRSM,3,3), LD(JLD), NDJNC(JNDSH), CDRLD(JDFSC),
     +  BRHS(JDFSC), X(JDFSC)
C
C
C       BETWEEN OVERLAYS  (2,0), (2,1), (2,2), (2,3), (2,4), (2,5), (2,6)
C
      DIMENSION  ENLWT(JDFSC), ECLWT(JELSH,12), ENLAUL(JAULD,JDFSC),
     +  ECLAUL(JAULD,JELSH,12), ENLTP(JZNTP,JDFSC),
     +  ECLTP(JZNTP,JELSH,12), INDIC(JSECT)
C
      COMPLEX  ENLSA(4,JDFSC), ECLSA(4,JELSH,12), KBNIF(JDFIF,JDFIF)
C
C
      LOGICAL  PDCHEK, SINGAB
C
C
C
       COMMON  STRTYP, AUTGN, SLDSP, SLRAC, SLSTS, SLBKL,
     +  PLSTR, PLMSH, PLDSP, PLSTS, PLMSP,
     +  MSECT, MSTRN, MNDSR, MRING, MNDRN, MGAUS,
     +  NNDSH, NELSH, NMTSH, NSTRN, NNDSR, NELSR, NMTSR, NIMSR,
     +  NELISR, NMTISR, NRING, NNDRN, NELRN, NMTRN, NIMRN, NELIRN,
     +  NMTIRN, NBRAC, NELBR, NMTBR, NGAUS, NCRSM, NPRDS, NLDCA,
     +  NSECT, NRWEL, NRWND, NDFND, NNDE1, NNDE2, NNDIN,
     +  NNTSH, NBTSH, NGTSH, NSTSSH, NKCTND, NNTSR, NBTSR, NGTSR,
     +  NTFSR, NSTSSR, NNTISR, NBTISR, NGTISR, NTFISR, NSTSIS, NNTRN,
     +  NBTRN, NGTRN, NTFRN, NSTSRN, NNTIRN, NBTIRN, NGTIRN, NTFIRN,
     +  NSTSIR, NTFBR, NSTSBR, NKRWPD, NCRLD, NFRCN, NABIF, NASCT,
     +  NACNT, NFBSC, NRACT, NKSCT, NKSCF, NKKGW, NKGSC, NKGICL, NKBNIF,
     +  NKGBNI, NEGVC, NKKGCN, NKGCN, NMSND, NMSCT, NKOEE, NKGOEE,
     +  CSDCL, CSDMC, MCCOD, ILDCA, LCNLD, LDTLD, LZNTP, LZNSL,
     +  DSKF1, DSKF2, IN, IO,
     +  FEED, RTOP, RBAS, HITE, ELEV, THATAD,
     +  TITLE(60), LDCAS(2), NCNLD(2), NDTLD(2), NZNTP(2), NZNSL(2),
     +  RATIO(14), INDEX(45), NDCON(200), NDEX1(15), NDEX2(15),
     +  NDINT(15), INTND(35), WKAREA(192)
```

```
      C
      C
            COMMON  / COMMN1 /
           + SLVSC, ROWSH, SHMTI, SHMTJ, CSDSG,
           + RSDOF, CSPRD, SCNLI, SCNLJ, CSCNL,
           + TYDTL, DSDTL, DRDTL, CSDTL, RFDTL,
           + SCTPI, SCTPJ, SEQND, NDORD, IPNTR,
           + ICONN, ROWSA,
           + ORIGN, KCINV, KSECT, ITCNL,
           + ITDTL, ITWSH,
           + NODSH, MTDSH, DNSSH, PRPSH, MTSHL,
           + TSHEL, TRMSH, ARASH,
           + NODSR, MTDSR, ECNSR, TRMSR,
           + PRPSR, MTSTR, CLSTR, NODISR, NDIMSR, MTDISR,
           + ECISR, TMISR, PRPISR, MTISR, CLISR, NODRN,
           + MTDRN, PSNRN, PRPRN, TRMRN, MTRNG
      C
            COMMON  / COMMN2 /
           + NRNGEL, NODIRN, NDIMRN, MTDIRN, PSNIRN,
           + TMIRN, PRPIRN, MTIRN, NIRNEL,
           + CRDCL, CRDMC, GUSWT, GUSTR, GUSCL,
           + DVDTL, NDDPD,
           + PRSDS, NREST, NDCNL, TPNDI, TPNDE,
           + DRCOS, LD, NDJNC, CORLD, BRHS, X
      C
            COMMON  / COMMN3 /
           + LRDF1, MNDSH, MELSH, MMTSH, MELSR,
           + MMTSR, MIMSR, MELISR, MMTISR, MELRN, MMTRN,
           + MIMRN, MELIRN, MMTIRN, MGUSS, MZNTP, MZNSL, MRWND,
           + MRWEL, MCRSM, MPRDS, MLDCA, MCNLD, MDTLD, MAULD, MDFSC, MDFIF,
           + NDFIF, NDFEX, NDFIN, NDFSC, NINFA, NDFBL, NNDEX, NBEG, NEND, IOP,
           + IBX, POCHEK, SINGAB, ICOND, IAFILE, IREF, DETCF, SLDLD, NDFSK
      C
      C
      C       BETWEEN OVERLAYS  (2,0), (2,1), (2,2), (2,3), (2,4), (2,5), (2,6)
      C
            COMMON  / COMMN4 /  NDFCN, NEIG, ENLWT, ECLWT, ENLAUL, ECLAUL,
           + ENLTP, ECLTP, ENLSA, ECLSA, INDIC, KBNIF
      C
      C
      C
      C
      C
      C***  GENERATE THE EQUIVALENT NODAL AND CENTROIDAL LOADS FOR SHELL
      C***  ELEMENTS.
      C
            CALL ENLSHL (MNDSH, MELSH, MMTSH, MZNSL, MCRSM, MPRDS,
           + MLDCA, MAULD, MDTLD, MZNTP, MGUSS, MRWND, MRWEL,
           + NODSH, NDORD, MTSHL, TSHEL, TRMSH, DNSSH, PRPSH, ARASH,
           + CRDCL, CRDMC, ROWSH, ORIGN, SLVSC, CSDSG, DRCOS, NDDPD,
           + RSDOF, NREST, CSPRD, GUSWT, GUSTR, GUSCL, LD, NDJNC, KCINV,
```

```
     + KSECT, TYDTL, ITWSH, ITDTL, DRDTL, CSDTL, RFDTL,
     + DVDTL, TPNDI, TPNDE, NTSHL, BTSHL, CURVT, ENLWT,
     + ECLWT, ENLAUL, ECLAUL, STSTP, ENLTP, ECLTP, ENLSA, ECLSA)
C
C
C
C
      RETURN
      END
C
C
C*                                                                    *
C**                                                                   **
C***          O V E R L A Y      2      2                            ***
C**                                                                   **
C*                                                                    *
C
C
C
      SUBROUTINE TWOTWO
C
C
C*          I N T E R O V E R L A Y   P A R A M E T E R S
C
      PARAMETER  (JNDSH = 30,  JDFSC = 12 * JNDSH,
     + JLD = JDFSC+1,  JSECT = 60,  JRWND = 15,  JRWEL = JRWND-1,
     + JELSH = 28, JZNSH = 1,  JMTSH = 1,
     + JSTRN = 1,   JNDSR = JRWND,  JELSR = JNDSR-1,
     + JMTSR = 3,  JIMSR = 1,  JELISR = 5,  JMTISR = 1,  JRING = 2,
     + JNDRN = 3,  JELRN = 3,  JMTRN = 2,  JIMRN = 15,  JELIRN = 20,
     + JMTIRN = 10,  JGAUS = 12,  JCRSM = 5,  JPRDS = JNDSH,
     + JLDCA = 2,  JCNLD = 5,  JDTLD = 2,  JAULD = 1,  JZNTP = 1,
     + JZNSL = 6,  JDFIF = 12 * JRWND,
     + JICON = 109, JDFSL = 64801)
C
C
C
C                 L  O  C  A  L     V  A  R  I  A  B  L  E  S
C
      DIMENSION  STPST(JZNTP,7,5)
C
C           I N T E R - O V E R L A Y     V A R I A B L E S
C
C               BETWEEN  OVERLAYS   (0,0), (1,0), (2,0)
C
      CHARACTER*10  STRTYP, AUTGN, SLOSP, SLRAC, SLSTS, SLBKL,
     + PLSTR, PLMSH, PLOSP, PLSTS, PLMSP
C
      INTEGER  CSOCL, CSDMC, DSKF1, DSKF2, SLDLD,
     + SLVSC(JLDCA,JZNSL,2), ROWSH(JRWND,2),
     + SHMTI(JZNSH), SHMTJ(JZNSH), CSDSG(JCRSM), RSDOF(JPRDS,12),
```

```
     +  CSPRD(JPRDS), SCNLI(JLDCA,JCNLD), SCNLJ(JLDCA,JCNLD),
     +  CSCNL(JLDCA,JCNLD), TYDTL(JLDCA,JDTLD), DSDTL(JLDCA,JDTLD),
     +  DRDTL(JLDCA,JDTLD,5), CSDTL(JLDCA,JDTLD), RFDTL(JLDCA,JDTLD),
     +  SCTPI(JLDCA,JZNTP), SCTPJ(JLDCA,JZNTP), SEQND(JNDSH),
     +  NDORD(JNDSH), IPNTR(JNDSH), ICONN(JICON), ROWSA(2)
C
      REAL  ORIGN(JRWEL,2), KCINV(JELSH,2,2), KSECT(JDFSL),
     +  ITCNL(JLDCA,JCNLD,9), ITDTL(JLDCA,JDTLD,2), ITWSH(JMTSH,3)
C
      DIMENSION  NODSH(JELSH,3), MTDSH(JZNSH), DNSSH(JELSH),
     +  PRPSH(JMTSH,8), MTSHL(JELSH),
     +  TSHEL(JMTSH), TRMSH(JMTSH), ARASH(JELSH), NODSR(JELSR,2),
     +  MTDSR(JMTSR), ECNSR(JMTSR), TRMSR(JMTSR),
     +  PRPSR(JMTSR,9), MTSTR(JELSR), CLSTR(JELSR,7,2),
     +  NODISR(JELISR,2), NDIMSR(JIMSR,2), MTDISR(JMTISR),
     +  ECISR(JMTISR), TMISR(JMTISR),
     +  PRPISR(JMTISR,9), MTISR(JELISR), CLISR(JELISR,7,2),
     +  NODRN(JELRN,2), MTDRN(JMTRN), TRMRN(JMTRN), PSNRN(JMTRN,3),
     +  PRPRN(JMTRN,9), MTRNG(JELRN), NRNGEL(JRING), NODIRN(JELIRN,2),
     +  NDIMRN(JIMRN,2), MTDIRN(JMTIRN), PSNIRN(JMTIRN,3),
     +  TMIRN(JMTIRN), PRPIRN(JMTIRN,9), MTIRN(JELIRN), NIRNEL(JIMRN),
     +  CRDCL(JNDSH,3), CRDMC(JNDSH,3), GUSWT(JGAUS), GUSTR(JGAUS,3),
     +  GUSCL(JELSH,JGAUS,3), DVDTL(JLDCA,JDTLD), NDDPD(JPRDS),
     +  PRSDS(JPRDS,12), NREST(JPRDS),
     +  NDCNL(JLDCA,JCNLD), TPNDI(JLDCA,JZNTP), TPNDE(JLDCA,JZNTP),
     +  DRCOS(JCRSM,3,3), LD(JLD), NDJNC(JNDSH), CORLD(JDFSC),
     +  BRHS(JDFSC), X(JDFSC)
C
C
C     BETWEEN OVERLAYS  (2,0), (2,1), (2,2), (2,3), (2,4), (2,5), (2,6)
C
      DIMENSION  ENLWT(JDFSC), ECLWT(JELSH,12), ENLAUL(JAULD,JDFSC),
     +  ECLAUL(JAULD,JELSH,12), ENLTP(JZNTP,JDFSC),
     +  ECLTP(JZNTP,JELSH,12), INDIC(JSECT)
C
      COMPLEX  ENLSA(4,JDFSC), ECLSA(4,JELSH,12), KBNIF(JDFIF,JDFIF)
C
      LOGICAL  PDCHEK, SINGAB
C
C
C
      COMMON  STRTYP, AUTGN, SLDSP, SLRAC, SLSTS, SLBKL,
     +  PLSTR, PLMSH, PLDSP, PLSTS, PLMSP,
     +  MSECT, MSTRN, MNDSR, MRING, MNDRN, MGAUS,
     +  NNDSH, NELSH, NMTSH, NSTRN, NNDSR, NELSR, NMTSR, NIMSR,
     +  NELISR, NMTISR, NRING, NNDRN, NELRN, NMTRN, NIMRN, NELIRN,
     +  NMTIRN, NBRAC, NELBR, NMTBR, NGAUS, NCRSM, NPRDS, NLDCA,
     +  NSECT, NRWEL, NRWND, NDFND, NNDE1, NNDE2, NNDIN,
     +  NNTSH, NBTSH, NGTSH, NSTSSH, NKCTND, NNTSR, NBTSR, NGTSR,
     +  NTFSR, NSTSSR, NNTISR, NBTISR, NGTISR, NTFISR, NSTSIS, NNTRN,
     +  NBTRN, NGTRN, NTFRN, NSTSRN, NNTIRN, NBTIRN, NGTIRN, NTFIRN,
```

```
     + NSTSIR, NTFBR, NSTSBR, NKRWPD, NCRLD, NFRCN, NABIF, NASCT,
     + NACNT, NFBSC, NRACT, NKSCT, NKSCF, NKKGW, NKGSC, NKGICL, NKBNIF,
     + NKGBNI, NEGVC, NKKGCN, NKGCN, NMSND, NMSCT, NKOEE, NKGOEE,
     + CSDCL, CSDMC, MCCOD, ILOCA, LCNLD, LDTLD, LZNTP, LZNSL,
     + DSKF1, DSKF2, IN, IO,
     + FEED, RTOP, RBAS, HITE, ELEV, THATAD,
     + TITLE(60), LDCAS(2), NCNLD(2), NDTLD(2), NZNTP(2), NZNSL(2),
     + RATIO(14), INDEX(45), NDCON(200), NDEX1(15), NDEX2(15),
     + NDINT(15), INTND(35), WKAREA(192)
C
C
      COMMON  / COMMN1 /
     + SLVSC, ROWSH, SHMTI, SHMTJ, CSDSG,
     + RSDOF, CSPRD, SCNLI, SCNLJ, CSCNL,
     + TYDTL, DSDTL, DRDTL, CSDTL, RFDTL,
     + SCTPI, SCTPJ, SEQND, NDDRD, IPNTR,
     + ICONN, ROWSA,
     + ORIGN, KCINV, KSECT, ITCNL,
     + ITDTL, ITWSH,
     + NODSH, MTDSH, DNSSH, PRPSH, MTSHL,
     + TSHEL, TRMSH, ARASH,
     + NODSR, MTDSR, ECNSR, TRMSR,
     + PRPSR, MTSTR, CLSTR, NODISR, NDIMSR, MTDISR,
     + ECISR, TMISR, PRPISR, MTISR, CLISR, NODRN,
     + MTDRN, PSNRN, PRPRN, TRMRN, MTRNG
C
      COMMON  / COMMN2 /
     + NRNGEL, NODIRN, NDIMRN, MTDIRN, PSNIRN,
     + TMIRN, PRPIRN, MTIRN, NIRNEL,
     + CRDCL, CRDMC, GUSWT, GUSTR, GUSCL,
     + DVDTL, NODPD,
     + PRSDS, NREST, NDCNL, TPNDI, TPNDE,
     + DRCDS, LD, NDJNC, CORLD, BRHS, X
C
      COMMON  / COMMN3 /
     + LRDF1, MNDSH, MELSH, MMTSH, MELSR,
     + MMTSR, MIMSR, MELISR, MMTISR, MELRN, MMTRN,
     + MIMRN, MELIRN, MMTIRN, MGUSS, MZNTP, MZNSL, MRWND,
     + MRWEL, MCRSM, MPRDS, MLDCA, MCNLD, MDTLD, MAULD, MDFSC, MDFIF,
     + NDFIF, NDFEX, NDFIN, NDFSC, NINFA, NDFBL, NNDEX, NBEG, NEND, IOP,
     + IBX, PDCHEK, SINGAB, ICOND, IAFILE, IREF, DETCF, SLDLD, NDFSK
C
C
C       BETWEEN OVERLAYS  (2,0), (2,1), (2,2), (2,3), (2,4), (2,5), (2,6)
C
      COMMON  / COMMN4 /  NDFCN, NEIG, ENLWT, ECLWT, ENLAUL, ECLAUL,
     + ENLTP, ECLTP, ENLSA, ECLSA, INDIC, KBNIF
C
C
C***  CALCULATE THE EQUIVALENT NODAL LOADS OF ALL STRINGER  (I.E.,
C***  STRINGER, INTERMEDIATE STRINGER, RING, AND INTERMEDIATE RING)
```

```
C***  ELEMENTS DUE TO SELF-WEIGHTS AND THERMAL EFFECTS AND ASSEMBLE
C***  THEM DIRECTLY INTO ENLWT AND ENLTP, RESPECTIVELY.
C
      CALL ENLSTF (MNDSH, MELSR, MMTSR, MELISR, MMTISR, MELRN,
     + MMTRN, MELIRN, MMTIRN, MCRSM, MLDCA, MZNTP, MRWND, MRWEL, NDFSC,
     + SLOLD, CROCL, CROMC, ROWSH, ORIGN, NODSR, PRPSR, MTSTR, TRMSR,
     + CLSTR, NODISR, PRPISR, MTISR, TMISR, CLISR, NODRN, PSNRN,
     + PRPRN, MTRNG, TRMRN, NODIRN, PSNIRN, PRPIRN, MTIRN, TMIRN, DRCOS,
     + TPNDI, TPNDE, NDORD, NDJNC, ENLWT, STPST, ENLTP)
C
C
C
      RETURN
      END
C
C
C*                                                                    *
C**                                                                   **
C***            O V E R L A Y     2     3                            ***
C**                                                                   **
C*                                                                    *
C
C
C
      SUBROUTINE TWOTHR
C
C
C*          I N T E R O V E R L A Y   P A R A M E T E R S
C
      PARAMETER  (JNDSH = 30,  JDFSC = 12 * JNDSH,
     + JLD = JDFSC+1,  JSECT = 60,  JRWND = 15,  JRWEL = JRWND-1,
     + JELSH = 28, JZNSH = 1,  JMTSH = 1,
     + JSTRN = 1,   JNDSR = JRWND,  JELSR = JNDSR-1,
     + JMTSR = 3,  JIMSR = 1,  JELISR = 5,  JMTISR = 1,  JRING = 2,
     + JNDRN = 5,  JELRN = 3,  JMTRN = 2,  JIMRN = 15,  JELIRN = 20,
     + JMTIRN = 10,  JGAUS = 12,  JCRSM = 5,  JPROS = JNDSH,
     + JLDCA = 2,  JCNLD = 5,  JDTLD = 2,  JAULD = 1,  JZNTP = 1,
     + JZNSL = 6,  JDFIF = 12 * JRWND,
     + JICON = 109, JDFSL = 64801)
C
C
C
C                    L O C A L    V A R I A B L E S
C
C
C
      DIMENSION  BTSHL(JGAUS,6,38),
     + CRVSH(JGAUS,5), BTSTF(7,5,22),
     + FRCNL(JDFIF), SPACE(1260),
     + STSTP(JZNTP,JGAUS,6), STPST(JZNTP,7,5)
C
```

```
      COMPLEX  FBNIF(JDFIF), ABNIF(JDFIF)
C
C                I N T E R - O V E R L A Y   V A R I A B L E S
C
C                   BETWEEN  OVERLAYS   (0,0), (1,0), (2,0)
C
      CHARACTER*10  STRTYP, AUTGN, SLDSP, SLRAC, SLSTS, SLBKL,
     + PLSTR, PLMSH, PLDSP, PLSTS, PLMSP
C
      INTEGER  CSDCL, CSDMC, DSKF1, DSKF2, SLDLD,
     + SLVSC(JLDCA,JZNSL,2), ROWSH(JRWND,2),
     + SHMTI(JZNSH), SHMTJ(JZNSH), CSDSG(JCRSM), RSDOF(JPRDS,12),
     + CSPRD(JPRDS), SCNLI(JLDCA,JCNLD), SCNLJ(JLDCA,JCNLD),
     + CSCNL(JLDCA,JCNLD), TYDTL(JLDCA,JDTLD), DSDTL(JLDCA,JDTLD),
     + DRDTL(JLDCA,JDTLD,5), CSDTL(JLDCA,JDTLD), RFDTL(JLDCA,JDTLD),
     + SCTPI(JLDCA,JZNTP), SCTPJ(JLDCA,JZNTP), SEQND(JNDSH),
     + NDORD(JNDSH), IPNTR(JNDSH), ICONN(JICON), ROWSA(2)
C
      REAL  ORIGN(JRWEL,2), KCINV(JELSH,2,2), KSECT(JDFSL),
     + ITCNL(JLDCA,JCNLD,9), ITDTL(JLDCA,JDTLD,2), ITWSH(JMTSH,3)
C
      DIMENSION  NODSH(JELSH,3), MTDSH(JZNSH), DNSSH(JELSH),
     + PRPSH(JMTSH,3), MTSHL(JELSH),
     + TSHEL(JMTSH), TRMSH(JMTSH), ARASH(JELSH), NODSR(JELSR,2),
     + MTDSR(JMTSR), ECNSR(JMTSR), TRMSR(JMTSR),
     + PRPSR(JMTSR,9), MTSTR(JELSR), CLSTR(JELSR,7,2),
     + NODISR(JELISR,2), NDIMSR(JIMSR,2), MTDISR(JMTISR),
     + ECISR(JMTISR), TMISR(JMTISR),
     + PRPISR(JMTISR,9), MTISR(JELISR), CLISR(JELISR,7,2),
     + NODRN(JELRN,2), MTDRN(JMTRN), TRMRN(JMTRN), PSNRN(JMTRN,3),
     + PRPRN(JMTRN,9), MTRNG(JELRN), NRNGEL(JRING), NODIRN(JELIRN,2),
     + NDIMRN(JIMRN,2), MTDIRN(JMTIRN), PSNIRN(JMTIRN,3),
     + TMIRN(JMTIRN), PRPIRN(JMTIRN,9), MTIRN(JELIRN), NIRNEL(JIMRN),
     + CRDCL(JNDSH,3), CRDMC(JNDSH,3), GUSWT(JGAUS), GUSTR(JGAUS,3),
     + GUSCL(JELSH,JGAUS,3), DVDTL(JLDCA,JDTLD), NODPD(JPRDS),
     + PRSDS(JPRDS,12), NREST(JPRDS),
     + NDCNL(JLDCA,JCNLD), TPNDI(JLDCA,JZNTP), TPNDE(JLDCA,JZNTP),
     + DRCOS(JCRSM,3,3), LD(JLD), NDJNC(JNDSH), CORLD(JDFSC),
     + BRHS(JDFSC), X(JDFSC)
C
C       BETWEEN OVERLAYS  (2,0), (2,1), (2,2), (2,3), (2,4), (2,5), (2,6)
C
      DIMENSION  ENLWT(JDFSC), ECLWT(JELSH,12), ENLAUL(JAULD,JDFSC),
     + ECLAUL(JAULD,JELSH,12), ENLTP(JZNTP,JDFSC),
     + ECLTP(JZNTP,JELSH,12), INDIC(JSECT)
C
      COMPLEX  ENLSA(4,JDFSC), ECLSA(4,JELSH,12), KBNIF(JDFIF,JDFIF)
C
C
      LOGICAL  PDCHEK, SINGAB
C
```

```
      EQUIVALENCE  ( FBNIF(1), ABNIF(1) )
C
      EQUIVALENCE ( BTSHL(1,1,1), BTSTF(1,1,1) )
C
      COMMON  STRTYP, AUTGN, SLDSP, SLRAC, SLSTS, SLBKL,
     + PLSTR, PLMSH, PLDSP, PLSTS, PLMSP,
     + MSECT, MSTRN, MNDSR, MRING, MNDRN, MGAUS,
     + NNDSH, NELSH, NMTSH, NSTRN, NNDSR, NELSR, NMTSR, NIMSR,
     + NELISR, NMTISR, NRING, NNDRN, NELRN, NMTRN, NIMRN, NELIRN,
     + NMTIRN, NBRAC, NELBR, NMTBR, NGAUS, NCRSM, NPRDS, NLDCA,
     + NSECT, NRWEL, NRWND, NDFND, NNDE1, NNDE2, NNDIN,
     + NNTSH, NBTSH, NGTSH, NSTSSH, NKCTND, NNTSR, NBTSR, NGTSR,
     + NTFSR, NSTSSR, NNTISR, NBTISR, NGTISR, NTFISR, NSTSIS, NNTRN,
     + NBTRN, NGTRN, NTFRN, NSTSRN, NNTIRN, NBTIRN, NGTIRN, NTFIRN,
     + NSTSIR, NTFBR, NSTSBR, NKRWPD, NCRLD, NFRCN, NABIF, NASCT,
     + NACNT, NFBSC, NRACT, NKSCT, NKSCF, NKKGW, NKGSC, NKGICL, NKBNIF,
     + NKGBNI, NEGVC, NKKGCN, NKGCN, NMSND, NMSCT, NK0EE, NKG0EE,
     + CSDCL, CSDMC, MCCOD, ILDCA, LCNLD, LDTLD, LZNTP, LZNSL,
     + DSKF1, DSKF2, IN, IO,
     + FEED, RTOP, RBAS, HITE, ELEV, THATAD,
     + TITLE(60), LDCAS(2), NCNLO(2), NDTLD(2), NZNTP(2), NZNSL(2),
     + RATIO(14), INDEX(45), NDCON(200), NDEX1(15), NDEX2(15),
     + NDINT(15), INTND(35), WKAREA(192)
C
C
      COMMON  / COMMN1 /
     + SLVSC, ROWSH, SHMTI, SHMTJ, CSDSG,
     + RSDDF, CSPRD, SCNLI, SCNLJ, CSCNL,
     + TYDTL, DSDTL, DRDTL, CSDTL, RFDTL,
     + SCTPI, SCTPJ, SEQND, NDORD, IPNTR,
     + ICONN, ROWSA,
     + ORIGN, KCINV, KSECT, ITCNL,
     + ITDTL, ITWSH,
     + NDDSH, MTDSH, DNSSH, PRPSH, MTSHL,
     + TSHEL, TRMSH, ARASH,
     + NDDSR, MTDSR, ECNSR, TRMSR,
     + PRPSR, MTSTR, CLSTR, NDDISR, NDIMSR, MTDISR,
     + ECISR, TMISR, PRPISR, MTISR, CLISR, NDDRN,
     + MTDRN, PSNRN, PRPRN, TRMRN, MTRNG
C
      COMMON  / COMMN2 /
     + NRNGEL, NDDIRN, NDIMRN, MTDIRN, PSNIRN,
     + TMIRN, PRPIRN, MTIRN, NIRNEL,
     + CRDCL, CRDMC, GUSWT, GUSTR, GUSCL,
     + DVDTL, NDDPD,
     + PRSDS, NREST, NDCNL, TPNDI, TPNDE,
     + DRCOS, LD, NDJNC, CORLD, BRHS, X
C
      COMMON  / COMMN3 /
     + LRDF1, MNDSH, MELSH, MMTSH, MELSR,
     + MMTSR, MIMSR, MELISR, MMTISR, MELRN, MMTRN,
```

```
      + MIMRN, MELIRN, MMTIRN, MGUSS, MZNTP, MZNSL, MRWND,
      + MRWEL, MCRSM, MPRDS, MLDCA, MCNLD, MDTLD, MAULD, MDFSC, MDFIF,
      + NDFIF, NDFEX, NDFIN, NDFSC, NINFA, NDFBL, NNDEX, NBEG, NEND, IOP,
      + IBX, PDCHEK, SINGAB, ICOND, IAFILE, IREF, DETCF, SLDLD, NDFSK
C
C
C       BETWEEN OVERLAYS  (2,0), (2,1), (2,2), (2,3), (2,4), (2,5), (2,6)
C
       COMMON  / COMMN4 /  NDFCN, NEIG, ENLWT, ECLWT, ENLAUL, ECLAUL,
      + ENLTP, ECLTP, ENLSA, ECLSA, INDIC, KBNIF
C
C
C
C
C
C
C***  SOLVE FOR THE DISPLACEMENTS OF THE SECORS FOR WHICH SOLUTION
C***  HAS BEEN REQUESTED. IF PRINTING OF THE DISPLACEMENTS HAS BEEN
C***  SPECIFIED, THE DISPLACEMENTS DUE TO THE COMBINED EFFECTS OF THE
C***  DEAD LOAD AND THE LIVE LOAD ARE PRINTED.
C
C
C
C
       CALL SLVASC (MNDSH, MELSH, MELSR, MELISR, MELPN, MELIRN,
      + MMTIRN, MPRDS, MLDCA, MCNLD, MAULD, MZNTP, MZNSL, MDFIF, NDFIF,
      + NDFSC, NDFSK, NDFCT, NODSH, NDFCN, SLDLD, NODSH, NODSR, NODISR,
      + NODRN, NODIRN, PRPIRN, MTIRN, NODPD,
      + RSDOF, PRSDS, NREST, CSPRD, SCNLI, SCNLJ, NDCNL, ITCNL, TYDTL,
      + RFDTL, DVDTL, SCTPI, SCTPJ, SLVSC, CRDCL, SEQND, NDORD, IPNTR,
      + ICONN, LD, NDJNC, KCINV, KSECT, CORLD, ENLWT, ECLWT,
      + FRCNL, ENLAUL, ECLAUL, ENLSA, ECLSA, ENLTP, ECLTP, ROWSA, INDIC,
      + X, BRHS, FBNIF, KBNIF, ABNIF, SPACE)
C
C
C
C*     IF THE SOLUTION OF THE REACTIONS HAS NOT BEEN REQUESTED, SKIP.
C
       IF (SLRAC . NE . 'YES')                          GO TO 250
C
C*     SOLVE FOR THE REACTIONS DUE TO THE LIVE LOADS FOR THE SECTORS
C*     FOR WHICH SOLUTION HAS BEEN REQUESTED AND PRINT THE REACTIONS
C*     DUE TO THE COMBINED EFFECTS OF THE DEAD LOAD AND THE LIVE LOAD,
C*     FOR THE SAME SECTORS.
C
       CALL SLVRAC (DSKF1, IO, ILDCA, LCNLD, LDTLD, LZNTP,
      + LZNSL, MLDCA, MPRDS, MCNLD, MAULD, MZNTP, MZNSL, NPRDS, NSECT,
      + NDFSC, NNDE2, NNDIN, NKRWPD, NASCT, NFBSC, NRACT, SLBKL, SLDLD,
      + NODPD, PRSDS, RSDOF, NREST, SCNLI, SCNLJ, NDCNL, ITCNL, TYDTL,
      + RFDTL, DVDTL, SCTPI, SCTPJ, SLVSC, NDEX1, NDEX2, NDINT, NDORD,
```

```
1989       + ROWSA, ENLWT, ENLAUL, ENLSA, ENLTP, BRHS, X, CORLD, SPACE)
1990  C
1991  C*    IF STRESSES HAVE NOT BEEN REQUESTED, SKIP.
1992  C
1993    250 IF (SLSTS . NE . 'YES' . AND . SLBKL . NE . 'YES')   GO TO 350
1994  C
1995  C***  SOLVE FOR THE STRESSES DUE TO THE LIVE LOADS ALONE, FOR THE
1996  C***  SECTORS FOR WHICH SOLUTION HAS BEEN REQUESTED. IF PRINTING
1997  C***  OF THE STRESSES HAS BEEN SPECIFIED, PRINT THE STRESSES DUE
1998  C***  TO THE COMBINED EFFECTS OF THE DEAD AND THE LIVE LOADS, FOR
1999  C***  THE SAME SECTORS.
2000  C
2001        CALL SLVSTS (MNDSH, MELSH, MMTSH, MELSR, MMTSR, MELISR,
2002       + MMTISR, MELRN, MMTRN, MELIRN, MMTIRN, MLOCA, MZNTP, MZNSL,
2003       + MGUSS, NDFSC, NDFCT, NCDSH, SLDLD, NODSH, PRPSH,
2004       + MTSHL, TSHEL, TRMSH, ARASH, NODSR, PRPSR, MTSTR,
2005       + TRMSR, NODISR, PRPISR, MTISR, TMISR, NODRN, PRPRN,
2006       + MTRNG, TRMRN, NODIRN, PRPIRN, MTIRN, TMIRN, SCTPI, SCTPJ,
2007       + TPNDI, TPNDE, SLVSC, CROMC, NDDRD, NDJNC, GUSWT, GUSTR,
2008       + GUSCL, STSTP, STPST, X, CORLD, BTSHL, CRVSH, BTSTF, SPACE)
2009  C
2010  C
2011    350 RETURN
2012        END
2013  C
2014  C
2015  C*                                                                *
2016  C**                                                               **
2017  C***            O V E R L A Y     2     4                        ***
2018  C**                                                               **
2019  C*                                                                *
2020  C
2021  C
2022  C
2023        SUBROUTINE TWOFOR
2024  C
2025  C
2026  C*          I N T E R O V E R L A Y    P A R A M E T E R S
2027  C
2028        PARAMETER  (JNDSH = 30,  JDFSC = 12 * JNDSH,
2029       + JLD = JDFSC+1,  JSECT = 60,  JRWND = 15,  JRWEL = JRWND-1,
2030       + JELSH = 28, JZNSH = 1,  JMTSH = 1,
2031       + JSTRN = 1,   JNDSR = JRWND,  JELSR = JNDSR-1,
2032       + JMTSR = 3,  JIMSR = 1,  JELISR = 5,  JMTISR = 1,  JRING = 2,
2033       + JNDRN = 6,  JELRN = 3,  JMTRN = 2,  JIMRN = 15,  JELIRN = 20,
2034       + JMTIRN = 10,  JGAUS = 12,  JCRSM = 5,  JPRDS = JNDSH,
2035       + JLDCA = 2,  JCNLD = 5,  JDTLD = 2,  JAULD = 1,  JZNTP = 1,
2036       + JZNSL = 6,  JDFIF = 12 * JRWND,
2037       + JICON = 109, JDFSL = 64801)
2038  C
2039  C
```

```
2040  C
2041  C                    L O C A L    V A R I A B L E S
2042  C
2043        REAL  KGIRW(12,JDFSC), KGSHL(38,38)
2044  C
2045        DIMENSION  GTSHL(JGAUS,3,38), KGNSH(36,36), SPACE(1260)
2046  C
2047        COMPLEX  KGBNIF(JDFIF,JDFIF)
2048  C
2049  C
2050  C          I N T E R - O V E R L A Y    V A R I A B L E S
2051  C
2052  C                BETWEEN  OVERLAYS   (0,0), (1,0), (2,0)
2053  C
2054        CHARACTER*10  STRTYP, AUTGN, SLDSP, SLRAC, SLSTS, SLBKL,
2055       + PLSTR, PLMSH, PLOSP, PLSTS, PLMSP
2056  C
2057        INTEGER  CSDCL, CSDMC, DSKF1, DSKF2, SLDLD,
2058       + SLVSC(JLDCA,JZNSL,2), ROWSH(JRWND,2),
2059       + SHMTI(JZNSH), SHMTJ(JZNSH), CSDSG(JCRSM), RSDOF(JPRDS,12),
2060       + CSPRD(JPRDS), SCNLI(JLDCA,JCNLD), SCNLJ(JLDCA,JCNLD),
2061       + CSCNL(JLDCA,JCNLD), TYDTL(JLDCA,JDTLD), DSDTL(JLDCA,JDTLD),
2062       + DRDTL(JLDCA,JDTLD,5), CSDTL(JLDCA,JDTLD), RFDTL(JLDCA,JDTLD),
2063       + SCTPI(JLDCA,JZNTP), SCTPJ(JLDCA,JZNTP), SEQND(JNDSH),
2064       + NDORD(JNDSH), IPNTR(JNDSH), ICONN(JICON), ROWSA(2)
2065  C
2066        REAL  ORIGN(JRWEL,2), KCINV(JELSH,2,2), KSECT(JDFSL),
2067       + ITCNL(JLDCA,JCNLD,9), ITDTL(JLDCA,JDTLD,2), ITWSH(JMTSH,3)
2068  C
2069        DIMENSION  NDDSH(JELSH,3), MTDSH(JZNSH), DNSSH(JELSH),
2070       + PRPSH(JMTSH,8), MTSHL(JELSH),
2071       + TSHEL(JMTSH), TRMSH(JMTSH), ARASH(JELSH), NDDSR(JELSR,2),
2072       + MTDSR(JMTSR), ECNSR(JMTSR), TRMSR(JMTSR),
2073       + PRPSR(JMTSR,9), MTSTR(JELSR), CLSTR(JELSR,7,2),
2074       + NDDISR(JELISR,2), NDIMSR(JIMSR,2), MTDISR(JMTISR),
2075       + ECISR(JMTISR), TMISR(JMTISR),
2076       + PRPISR(JMTISR,9), MTISR(JELISR), CLISR(JELISR,7,2),
2077       + NDDRN(JELRN,2), MTDRN(JMTRN), TRMRN(JMTRN), PSNRN(JMTRN,3),
2078       + PRPRN(JMTRN,9), MTRNG(JELRN), NRNGEL(JRING), NDDIRN(JELIRN,2),
2079       + NDIMRN(JIMRN,2), MTDIRN(JMTIRN), PSNIRN(JMTIRN,3),
2080       + TMIRN(JMTIRN), PRPIRN(JMTIRN,9), MTIRN(JELIRN), NIRNEL(JIMRN),
2081       + CRDCL(JNDSH,3), CRDMC(JNDSH,3), GUSWT(JGAUS), GUSTR(JGAUS,3),
2082       + GUSCL(JELSH,JGAUS,3), DVDTL(JLDCA,JDTLD), NDDPD(JPRDS),
2083       + PRSDS(JPRDS,12), NREST(JPRDS),
2084       + NDCNL(JLDCA,JCNLD), TPNDI(JLDCA,JZNTP), TPNDE(JLDCA,JZNTP),
2085       + DRCOS(JCRSM,3,3), LD(JLD), NDJNC(JNDSH), CORLD(JDFSC),
2086       + BRHS(JDFSC), X(JDFSC)
2087  C
2088  C       BETWEEN OVERLAYS  (2,0), (2,1), (2,2), (2,3), (2,4), (2,5), (2,6)
2089  C
2090        DIMENSION  ENLWT(JDFSC), ECLWT(JELSH,12), ENLAUL(JAULD,JDFSC),
```

```
     +  ECLAUL(JAULD,JELSH,12), ENLTP(JZNTP,JDFSC),
     +  ECLTP(JZNTP,JELSH,12), INDIC(JSECT)
C
      COMPLEX  ENLSA(4,JDFSC), ECLSA(4,JELSH,12), KBNIF(JDFIF,JDFIF)
C
C
      LOGICAL  PDCHEK, SINGAB
C
C
C
      COMMON  STRTYP, AUTGN, SLDSP, SLRAC, SLSTS, SLBKL,
     +  PLSTR, PLMSH, PLDSP, PLSTS, PLMSP,
     +  MSECT, MSTRN, MNDSR, MRING, MNDRN, MGAUS,
     +  NNDSH, NELSH, NMTSH, NSTRN, NNDSR, NELSR, NMTSR, NIMSR,
     +  NELISR, NMTISR, NRING, NNDRN, NELRN, NMTRN, NIMRN, NELIRN,
     +  NMTIRN, NBRAC, NELBR, NMTBR, NGAUS, NCRSM, NPRDS, NLOCA,
     +  NSECT, NRWEL, NRWND, NDFND, NNDE1, NNDE2, NNDIN,
     +  NNTSH, NBTSH, NGTSH, NSTSSH, NKCTND, NNTSR, NBTSR, NGTSR,
     +  NTFSR, NSTSSR, NNTISR, NBTISR, NGTISR, NTFISR, NSTSIS, NNTRN,
     +  NBTRN, NGTRN, NTFRN, NSTSRN, NNTIRN, NBTIRN, NGTIRN, NTFIRN,
     +  NSTSIR, NTFBR, NSTSBR, NKRWPD, NCRLD, NFRCN, NABIF, NASCT,
     +  NACNT, NFBSC, NRACT, NKSCT, NKSCF, NKKGW, NKGSC, NKGICL, NKBNIF,
     +  NKGBNI, NEGVC, NKKGCN, NKGCN, NMSND, NMSCT, NKOEE, NKGOEE,
     +  CSDCL, CSDMC, MCCOD, ILOCA, LCNLD, LDTLD, LZNTP, LZNSL,
     +  DSKF1, DSKF2, IN, IO,
     +  FEED, RTOP, RBAS, HITE, ELEV, THATAD,
     +  TITLE(60), LOCAS(2), NCNLD(2), NDTLD(2), NZNTP(2), NZNSL(2),
     +  RATIO(14), INDEX(45), NDCON(200), NDEX1(15), NDEX2(15),
     +  NDINT(15), INTND(35), WKAREA(192)
C
C
      COMMON  / COMMN1 /
     +  SLVSC, ROWSH, SHMTI, SHMTJ, CSDSG,
     +  RSDOF, CSPRD, SCNLI, SCNLJ, CSCNL,
     +  TYDTL, DSDTL, DRDTL, CSDTL, RFDTL,
     +  SCTPI, SCTPJ, SEQND, NDORD, IPNTR,
     +  ICONN, ROWSA,
     +  ORIGN, KCINV, KSECT, ITCNL,
     +  ITDTL, ITWSH,
     +  NODSH, MTDSH, DNSSH, PRPSH, MTSHL,
     +  TSHEL, TRMSH, ARASH,
     +  NODSR, MTDSR, ECNSR, TRMSR,
     +  PRPSR, MTSTR, CLSTR, NODISR, NDIMSR, MTDISR,
     +  ECISR, TMISR, PRPISR, MTISR, CLISR, NODRN,
     +  MTDRN, PSNRN, PRPRN, TRMRN, MTRNG
C
      COMMON  / COMMN2 /
     +  NRNGEL, NODIRN, NDIMRN, MTDIRN, PSNIRN,
     +  TMIRN, PRPIRN, MTIRN, NIRNEL,
     +  CRDCL, CRDMC, GUSWT, GUSTR, GUSCL,
     +  DVDTL, NODPD,
```

```
     +  PRSDS, NREST, NDCNL, TPNDI, TPNDE,
     +  DRCOS, LD, NDJNC, CORLD, BRHS, X
C
      COMMON  / COMMN3 /
     +  LRDF1, MNDSH, MELSH, MMTSH, MELSR,
     +  MMTSR, MIMSR, MELISR, MMTISR, MELPN, MMTRN,
     +  MIMRN, MELIRN, MMTIRN, MGUSS, MZNTP, MZNSL, MRWND,
     +  MRWEL, MCRSM, MPROS, MLOCA, MCNLD, MDTLD, MAULD, MDFSC, MDFIF,
     +  NDFIF, NDFEX, NDFIN, NDFSC, NINFA, NDFBL, NNDEX, NBEG, NEND, IOP,
     +  IBX, PDCHEK, SINGAB, ICOND, IAFILE, IREF, DETCP, SLDLD, NDFSK
C
C
C       BETWEEN OVERLAYS  (2,0), (2,1), (2,2), (2,3), (2,4), (2,5), (2,6)
C
      COMMON  / COMMN4 /  NDFCN, NEIG, ENLWT, ECLWT, ENLAUL, ECLAUL,
     +  ENLTP, ECLTP, ENLSA, ECLSA, INDIC, KBNIF
C
      EQUIVALENCE  ( KGSHL(1,1), KGIRW(1,1) )
C
      EQUIVALENCE  ( GTSHL(1,1,1), KGNSH(1,1) )
C
C
C
C*    NDFRW IS THE NUMBER OF DEGREES OF FREEDOM OF 12 ROWS OF THE
C*    STIFFNESS MATRIX IN ITS CONVENTIONAL SQUARE FORM.
C
      NDFRW = 12 * NDFSC
C
C
C*    NCOLS IS THE INTEGER NUMBER OF COLUMNS OF THE COMPLEX MATRIX
C*    KBNIF(NDFIF,NDFIF), WHICH IS ACCOMODATED IN ONE RECORD OF
C*    DISK FILE DSKF1 (I. E., LRDF1).
C
      NCOLS = LRDF1 / (2 * NDFIF)
C
C*    NR IS THE NUMBER OF COLUMNS OF THE COMPLEX MATRIX
C*    KBNIF(NDFIF,NDFIF) WHICH WILL BE WRITTEN (READ) ON (FROM) DISK
C*    FILE DSKF1 IN GROUPS OF NCOLS.
C
      NR = NDFIF / NCOLS
C
C*    JREMN ( < NCOLS ) IS THE NUMBER OF COLUMNS IN KBNIF(NDFIF,NDFIF)
C*    REMAINING AND NOT COVERED BY THE NR GROUPS OF NCOLS COLUMNS.
C
      JREMN = NDFIF - NR * NCOLS
C
C*    NRECS IS THE TOTAL NUMBER OF RECORDS OF LENGTH LRDF1 IT TAKES TO
C*    WRITE KBNIF(NDFIF,NDFIF) ON DISK FILE DSKF1.
C
      NRECS = NR
C
```

```
      IF (JREMN . NE . 0)                                  NRECS = NR + 1
C
C
C
      IF (NELSH . EQ . 0)                                  GO TO 8500
C
C***  CALCULATE THE GEOMETRIC STIFFNESS MATRIX OF ALL SHELL ELEMENTS
C***  DUE TO THE DEAD LOAD, AND ADD THEM DIRECTLY TO KSECT(NDFSK),
C***  WHICH NOW CONTAINS THE UNFACTORIZIED CONVENTIONAL STIFFNESS
C***  MATRIX OF THE SECTOR. HERE EACH SHELL ELEMENT HAS ONLY 38 DEGREE
C***  OF FREEDOM SHELL ELEMENT.
C
      CALL KGSHEL (DSKF1, MELSH, MGAUS, NNDSH, NELSH, NGAUS,
     + NPRDS, NDFND, NDFSC, NDFSK, NGTSH, NSTSSH,
     + NKCTND, NKKGCN, NKGCN, PLMSP, SLOLD, NODSH, TSHEL, MTSHL,
     + ARASH, NDORD, IPNTR, ICONN, LD, NDJNC, GUSWT,
     + KCINV, GTSHL, KGSHL, KGNSH, KSECT, SPACE, WKAREA)
C
C
C
C
C
C*    IF NO STIFFENER (I.E., STRINGER, INTERMEDIATE STRINGER, RING,
C*    INTERMEDIATE RING) ELEMENTS EXIST, SKIP.
C
 8500 IF (NELSR . EQ . 0 . AND . NELISR . EQ . 0 . AND .
     + NELRN . EQ . 0 . AND . NELIRN . EQ . 0)    GO TO 9500
C
C***  CALCULATE THE GEOMETRIC STIFFNESS MATRIX OF ALL STIFFENER
C***  (STRINGRE, INTERMEDIATE STRINGER, RING, AND INTERMEDIATE
C***  RING AND BRACE) ELEMENTS AND ADD THEM DIRECTLY TO KSECT(NDFSK),
C***  WHICH NOW CONTAINS THE UNFACTORIZED CONVENTIONAL STIFFNESS MATRIX
C***  OF THE SECTOR.
C
      CALL KGSTFN (MNDSH, MELSR, MMTSR, MELISR, MMTISR, MELRN,
     + MMTRN, MELIRN, MMTIRN, NDFSK, SLOLD, NODSR, PRPSR, MTSTR,
     + NODISR, PRPISR, MTISR, NODRN, PRPRN, MTRNG, NODIRN, PRPIRN,
     + MTIRN, CRDMC, NDORD, LD, NDJNC, KSECT, SPACE)
C
C*    IF THE CURRENT SOLUTION IS NOT FOR DEAD LOAD, SKIP.
C
 9500 IF (SLOLD . NE . 3HYES)                             GO TO 13500
C
C
C
C
C                D E A D    L O A D    S O L U T I O N
C
C
C
C***  SET THE NONDIAGONAL ENTRIES OF KSECT CORRESPONDING TO THE
```

```
C***  RESTRAINED DEGREES OF FREEDOM EQUAL TO ZERO. THE ASSOCIATED
C***  DIAGONAL ENTRIES ARE SET EQUAL TO ONE.
C
      CALL MDFKPD (1.0, MPROS, NNOSH, NDFND, NPROS, NNDE1,
     + NDEX1, NDEX2, INTND, INDEX, NDCON, NODPD, NREST, RSDOF, LD,
     + KSECT, NDORD, SEQND, IPNTR, ICONN)
C
C
C
C
C
C
C
C***  WRITE THE SUM OF THE SECTOR CONVENTIONAL STIFFNESS MATRIX AND
C***  THE SECTOR GEOMETRIC STIFFNESS MATRIX DUE TO DEAD LOAD (BOTH
C***  UNFACTORIZED) ON DISK FILE DSKF1. THIS SUM WILL LATER BE USED
C***  IN CALCULATING THE MODE SHAPES OF THE INTERNAL NODES (SEE
C***  SUBROUTINE BUCKLE). HERE VECTOR KSECT(NDFSK) WILL BE WRITTEN IN
C***  WORD GROUPS OF 1260 ON THE DISK FILE.
C
      CALL WRTKSC (DSKF1, NKKGW, 0, 1260, NDFSK, KSECT)
C
C*    IF THERE ARE NO INTERNAL NODES, FACTORIZATION AND STATIC
C*    CONDESATION OF KSECT ARE NOT NEEDED.
C
      IF (NNDIN . EQ . 0)                          GO TO 12500
C
C
      WRITE(6,*) ' NBEG=',NBEG,' NEND=',NEND,' NDFSC=',NDFSC,
     + ' LD',LD,' SINGAB=',SINGAB,' PDCHEK=',PDCHEK,' ICOND=',ICOND,
     + ' IAFILE=',IAFILE,' ACOND',ACOND,' DETCF=',DETCF,' NEGEIG=',
     + NEGEIG,' IFAIL=',IFAIL
C
C***  FACTORIZE KSECT ( = KSECT + KGDLD ). THE FACTORIZED MATRIX
C***  WILL BE USED TO CONDENSE OUT THE INTERNAL NODES OF THE SECTOR.
C
      CALL SKYFAC (KSECT, NBEG, NEND, NDFSC, LD, CORLD, SINGAB,
     + PDCHEK, ICOND, IAFILE, ACOND, DETCF, IDETEX, NEGEIG, IFAIL)
C
C
      PRINT*, 'IFAIL=',IFAIL,'  NEGEIG=',NEGEIG
      WRITE(6,*) 'NEGEIG=',NEGEIG,'IFAIL=',IFAIL
C
      IF (IFAIL . EQ . 0)                GO TO 10500
C
      WRITE(6,*) 'EXECUTION WAS DISCONTINUED BECAUSE FACTORIZATION OF '
     + ,'KSECT + KGDLD (DUE TO DEAD LOAD) HAS FAILED'
C
      PRINT*,    'EXECUTION WAS DISCONTINUED BECAUSE FACTORIZATION OF '
     + ,'KSECT + KGDLD (DUE TO DEAD LOAD) HAS FAILED'
C
```

```
      STOP
C
C
10500 IKGOEE = 0
C
C*    CONSIDER ALL EXTERNAL 2 AND EXTERNAL 1 NODES, RESPECTIVELY.
C
      DO 12400 IEX = 1 , 2
C
      DO 12000 IRWND = 1 , NRWND
C
C*    FIND THE CURRENT EXTERNAL NODE.
C
      INDRW = NRWND - IRWND + 1
C
      NDEXT = (2 - IEX) * NDEX2(INDRW) + (IEX - 1) * NDEX1(INDRW)
C
C*    FIND THE ORDER OF THE CURRENT EXTERNAL NODE.
C
      IDREX = NDORD(NDEXT)
C
C*    CONSIDER ALL DEGREES OF FREEDOM OF THE CURRENT EXTERNAL NODE.
C
      DO 11000 IDFEX = 1 , NDFND
C
      IROW = (IDREX-1) * NDFND + IDFEX
C
C***  CONSTRUCT ROW NUMBER IROW OF KEI (COLUMN NUMBER IROW OF KIE)
C***  AND PUT IT IN BRHS. ON RETURN THE ENTRIES OF BRHS CORRESPONDING
C***  TO THE EXTERNAL NODES CONTAIN ZEROS.
C
       CALL KEIBRS (IDREX, IDFEX, IROW, NNDSH, NDFSC, NDFND,
     + NDFEX, NDFIN, NNDEX, NNDIN, NDINT, INTND, NDORD, SEQND, INDEX,
     + NDCON, LD, KSECT, BRHS)
C
C***  TAKE THE ENTRIES OF BRHS WHICH CORRESPOND TO THE INTERNAL NODES,
C***  AND CONTAIN THE IROW-TH COLUMN OF KIE, AND ON RETURN PUT THE
C***  PRODUCT  KEI * KII-INV * KIE  IN THE EXTERNAL ENTRIES OF BRHS.
C
       CALL SKYSOL (KSECT, NDFSC, LD, IDP, IBX, BRHS, X,
     + IREF ,IAFILE, WKAREA, DELTA)
C
C***  ARRANGE ALL ENTRIES OF VECTOR BRHS CORRESPONDING TO THE
C***  INTERNAL NODES, SO THAT, ON RETURN, ITS TOP NDFEX ENTRIES
C***  ARE ASSOCIATED WITH THE EXTERNAL 2 AND EXTERNAL 1 NODES,
C***  RESPECTIVELY.
C
       CALL ARRANG (NRWND, NDFSC, NDEX1, NDEX2, NDORD, X, BRHS)
C
C
      IKGOEE = IKGOEE + 1
```

```
C
C*    PUT THE TOP NDFEX ENTRIES OF BRHS, WHICH IS THE ICOL-TH COLUMN OF
C*    ( KEI * KII-INV * KIE ) ON DISK FILE DSKF2.
C
      WRITE (DSKF2, REC=NKGOEE+IKGOEE)  (BRHS(I), I=1, NDFEX)
C
11000 CONTINUE
C
12000 CONTINUE
C
12400 CONTINUE
C
C
C
C*    GENERATE THE COMPLEX MATRIX KBNIF(NDFIF,NDFIF) FOR ALL SECTORS,
C*    AND WRITE THEM ON DISK FILE DSKF1 TO BE USED LATER FOR THE
C*    EIGENVALUE PROBLEM.
C
C*    CONSIDER ALL INTERFACES.
C
12500 DO 13000 N = 1 , NINFA
C
C
C          K B N I F  +  K G B N I F   D U E   T O   D E A D   L O A D
C
C***  GENERATE THE COMPLEX MATRIX  KBNIF(NDFIF,NDFIF)  DIRECTLY
C***  FROM VECTOR KSECT(NDFSK) ACCORDING TO:
C
C      KBNIF(NDFIF,NDFIF) =
C      KE2(NDFIF,NDFIF) + KE1(NDFIF,NDFIF) + KE21(NDFIF,NDFIF) *
C       CEXP(I*2*PI*N/NINFA) + KE12(NDFIF,NDFIF) * CEXP(-I*2*PI*N/NINFA)
C
C***  NOTE, HERE KE2, KE1, KE21, AND KE12 ARE NOT CONDENSED. ALSO
C***  NOTE, THAT THE ASSEMBLY OF KE2 OF SECTOR N WITH KE1 OF SECTOR
C***  (N+1) IS IMPLIED IN THE ABOVE OPERATION.
C
      CALL KBNINF (MDFIF, NNDSM, NNDE1, NDFND, NNDEX, NDFIN,
     + NDFIF, NINFA, N, NDEX1, NDEX2, INTND, SEQND, NDORD, IPNTR, ICONN,
     + LD, KSECT, KBNIF)
C
C
C
C     IF NO INTERNAL NODES EXIST, THERE IS NO STATIC CONDENSATION.
C
      IF (NNDIN . EQ . 0)                          GO TO 12700
C
C
C
C
C
C
```

```
C                          K  O  E  E  S
C
C
C***  COMPLETE THE GENERATION OF KBNIF(NDFIF,NDFIF), BY SUBTRACTING
C***  THE TERM
C                KOEES(NDFIF,NDFIF) = KEI * KII-INV * KIE
C***  ARISING FROM THE STATIC CONDENSATION OF THE INTERNAL DEGREES OF
C***  FREEDOM. NOTE THAT THE ASSEMBLY OF KE2 OF SECTOR N AND KE1 OF
C***  SECTOR (N+1) IS CARRIED OUT SIMULTANEOUSLY.
C
      CALL KBNIF0 (DSKF2, MDFIF, N, NDFEX, NINFA, NDFIF, NKGOEE,
     + CORLD, KBNIF)
C
C***  WRITE THE NDFIF BY NDFIF PART OF THE COMPLEX MATRIX
C***  KBNIF(JDFIF,JDFIF) ON DISK FILE DSKF1 USING THE RECORD
C***  LENGTH OF LRDF1.
C
12700 CALL WRTKBN (DSKF1, JREMN, MDFIF, N, NCOLS, NDFIF,
     + NKBNIF, NR, NRECS, KBNIF)
C
C
13000 CONTINUE
C
C
C
      GO TO 15170
C
C
C
C                          L I V E   L O A D   S O L U T I O N
C
C
C***  SET THE THE ROWS AND COLUMNS OF THE GEOMETRIC STIFFNESS MATRIX
C***  CORRESPONDING TO ALL DEGREES OF FREEDOM EQUAL TO ZERO,
C***  EXCEPT THE DIAGONAL ENTRIES WHICH ARE SET TO 1.0.
C
13500 CALL MDFKPO (1.0, MPROS, MADSH, NDFND, NPROS, NNDE1,
     + NDEX1, NDEX2, ENTH2, ELDA, PAIN, NODPO, IPRST, ASECP, LD,
     + KSECT, NDORD, SECND, IPNTR, ICONN)
C
C
C***  WRITE THE NDFIF BY NDFIF PART OF THE COMPLEX MATRIX
C***  ( NOW IN KSECT(NDFSQ) ) ON DISK FILE DSKF1. THIS WILL BE USED
C***  LATER IN CALCULATING THE MODE SHAPES OF THE INTERNAL NODES.
C
C
C
      CALL WRTKBC (DSKF1, JREMN, NDFIF, N, NCOLS, KSECT)
C
```

```
C
C
C*    IF NO INTERNAL NODES EXIST, THERE IS NO NEED FOR STATIC
C*    CONDENSATION.
C
      IF (NNDIN . EQ . 0)                                  GO TO 15150
C
C
C                              G  U  Y  A  N
C
C
C
C
C*    IN THE FOLLOWING THE TERM
C
C       KGOEE = KIE-T * KII-1 * KGIE + KGEI * KII-1 * KIE -
C       KIE-T * KII-1 * KGII * KII-1 * KIE
C
C*    WHICH IS DEVELOPED BY GUYAN WILL BE CALCULATED AND WILL BE
C*    WRITTEN ON DISK FILE DSKF2 AFTER REARRANGEMENT OF ITS ROWS
C*    AND COLUMNS. EACH COLUMN OF KGOEE IS CALCULATED SEPARATELY
C*    IN ORDER TO USE LEAST AMOUNT OF CENTRAL MEMORY.
C
C
      IKGOEE = 0
C
C*    AT THIS POINT VECTOR KSECT CONTAINS KGSCT, THE GEOMETRIC
C*    STIFFNESS MATRIX.
C
C***  CONSIDER ALL INTERNAL NODES IN THE ORDER THEY APPEAR IN THE
C***  GEOMETRIC STIFFNESS MATRIX, AND WRITE THE (NDFSC BY 12) COLUMNS
C***  ASSOCIATED WITH EACH NODE ON DISK FILE DSKF1.
C
      CALL KGINRW (DSKF1, LRDF1, MDFSC, NDFND, NDFSC, NNDIN,
     + NKGICL, INDEX, NDCON, NDINT, LD, KSECT, NDORD, SEQND, IPNTR,
     + ICONN, KGIRW)
C
C*    READ THE FACTORIZED CONVENTIONAL STIFFNESS MATRIX FROM DISK FILE
C*    DSKF1.
C
      CALL REDKSC (DSKF1, NKSCF, 0, LRDF1, NDFSK, KSECT)
C
C
C*    CONSIDER ALL EXTERNAL 2 AND EXTERNAL 1 NODES, RESPECTIVELY.
C
      DO 15120 IEX = 1 , 2
C
      DO 15100 IRWND = 1 , NRWND
C
```

```
C*    FIND THE CURRENT EXTERNAL NODE.
C
      INDRW = NRWND - IRWND + 1
C
      NDEXT = (2 - IEX) * NDEX2(INDRW) + (IEX - 1) * NDEX1(INDRW)
C
C*    FIND THE ORDER OF THE CURRENT EXTERNAL NODE.
C
      IOREX = NDORD(NDEXT)
C
C*    CONSIDER ALL DEGREES OF FREEDOM OF THE CURRENT EXTERNAL NODE.
C
      DO 15000 IDFEX = 1 , NDFND
C
      IROW = (IOREX-1) * NDFND + IDFEX
C
C***  CONSTRUCT ROW NUMBER IROW OF KGEI (COLUMN NUMBER IROW OF KGIE)
C***  AND PUT IT IN THE INTERNAL ENTRIES OF BRHS. ON RETURN THE EXTERNAL
C***  ENTRIES WILL CONTAIN ZEROS.
C
      CALL KEIBRS (IOREX, IDFEX, IROW, NNDSH, NDFSC, NDFND,
     + NDFEX, NDFIN, NNDEX, NNDIN, NDINT, INTND, NDORD, SEQND, INDEX,
     + NDCON, LD, KSECT, BRHS)
C
C
C
C***  TAKE THE INTERNAL ENTRIES OF BRHS, WHICH CONTAIN THE IROW-TH
C***  COLUMN OF KGIE, AND ON RETURN PUT THE PRODUCT  KII-INV * KIE
C***  IN THE EXTERNAL ENTRIES OF BRHS. (IOP = 0, IBX = 0)
C***  AT THIS POINT THE EXTERNAL ENTRIES CONTAIN ZEROS IN ORDER TO
C***  CARRY OUT THE ABOVE MULTIPLICATION.
C***  WHEN IBX IS EQAUL TO ZERO, INDICATING A REQUEST FOR A NORNAL
C***  SOLUTION, VECTORS BRHS AND X MUST BE DECLARED EQUIVALENT THROUGH
C***  AN EQUIVALENT STATEMENT. THEN THE SOLUTION X OVERWRITES BRHS.
C***  THE SAME RESULT COULD BE EFFECTED IF BRHS IS PASSED TWICE TO
C***  SUBROUTINE SKYSOL, THE SECOND TIME OF WHICH CORRESPONDS TO X.
C
      CALL SKYSOL (KSECT, NDFSC, LD, ICP, 0, BRHS, BRHS,
     + IREF ,IAFILE, WKAREA, DELTA)
C
C
C
C***  SET ALL ENTRIES OF VECTOR X ( = Y1 ) AND VECTOR CORLD ( = ONE
C***  COLUMN OF KGIE) EQUAL TO ZERO FOR THE CURRENT (IROW-TH) COLUMN
C***  OF KGOEE.
C
      CALL SETZRO (NDFSC, 1, X)
C
      CALL SETZRO (NDFSC, 1, CORLD)
C
C*    TO PRODUCE ONE COLUMN OF X ( = Y1) ALL COLUMNS OF KGSCT
```

```
C*    CORRESPONDING TO THE INTERNAL NODES WILL BE READ FROM DISK FILE
C*    DSKF1 IN THE ORDER THEY APPEAR IN THE STIFFNESS MATRIX.
C
C*    CONSIDER ALL INTERNAL NODES.
C
      DO 14400 INDIN = 1 , NNDIN
C
C*    FIND THE INDIN-TH INTERNAL NODE.
C
      NDIN = NDINT(NNDIN + 1 - INDIN)
C
C*    DETERMINE THE POSITION IN BRHS ( = X1 ) JUST BEFORE THOSE
C*    CORRESPONDING TO THE CURRENT INTERNAL NODE, AND THE POSITION OF
C*    OF THE BEGINNING AND THE ENDING ENTRIES OF THE CURRENT INTERNAL
C*    NODE.
C
      IBFR = ( NDORD(NDIN) - 1 ) * 12
      IBIND = IBFR + 1
      IEIND = IBFR + 12
C
C***  READ THE (NDFSC BY 12) COLUMN OF KGSCT ASSOCIATED WITH THE CURRENT
C***  INTERNAL NODE FROM DISK FILE DSKF1.
C
C
      IR = (INDIN - 1) * (NDFRW / LRDF1 + 1)
C
      CALL REDKSC (DSKF1, NKGICL, IR, LRDF1, NDFRW, KGIRW)
C
C
C*    PUT THE COLUMN  OF KGIRW ASSOCIATED WITH THE CURRENT INTERNAL NODE
C*    IN THE CORRESPONDING INTERNAL ENTRIES OF VECTOR CORLD.
C*    THIS ROW CORRESPONDS TO THE PORTION OF A COLUMN OF KGIF ASSOCIATED
C*    WITH THE CURRENT IROW AT THE CURRENT INTERNAL NODE.
C
      DO 13800 I = 1 , 12
C
      CORLD(IBFR+I) = KGIRW(I, IROW)
C
13800 CONTINUE
C
C*    NOW CARRY OUT THE PRODUCT:
C
C       KGIRW(12,NDFSC) * BRHS (ONLY THE 12 ENTRIES ASSOCIATED
C       WITH NDIN)
C
C*    NOTE THAT ZERO BLOCKS OF KGIRW WILL NOT BE CONSIDERED IN THE
C*    MULTIPLICATION.
C
      IPNT = INDEX(NDIN)
      NCND = NDCON(IPNT)
      IBEG = IPNT + 1
```

```
2601          IEND = IPNT + NCND
2602    C
2603    C*    CONSIDER ALL NODES CONNECTED TO NDIN.
2604    C
2605          DO 14100 IPOS = IBEG , IEND
2606    C
2607    C*    DETERMINE THE CURRNET NODE CONNECTED TO NDIN.
2608    C
2609          NDCN = NDCON(IPOS)
2610    C
2611    C*    FIND THE BEGINNING AND THE ENDING ROW NUMBERS OF THE BLOCK
2612    C*    NDCN-NDIN IN KGIRW.
2613    C
2614          IBRW = ( NDORD(NDCN) - 1 ) * 12 + 1
2615          IERW = IBRW + 11
2616    C
2617    C*    MULTIPLY BLOCK NDCN-NDIN BY THE ROWS OF BRHS ( = X1 )
2618    C*    CORRESPONDING TO NODE NDIN.
2619    C
2620          DO 14000 I = IBRW , IERW
2621    C
2622          DO 13900 J = 1    ,   12
2623    C
2624          X(I) = X(I) + KGIRW(J,I) * BRHS(IBFR+J)
2625    C
2626    13900 CONTINUE
2627    C
2628    14000 CONTINUE
2629    C
2630    14100 CONTINUE
2631    C
2632    C*    CARRY OUT THE MULTIPLICATION OF THE BLOCK NDIN-NDIN AND THE
2633    C*    ASSOCIATED ENTRIES IN BRHS ( = X1 )
2634    C
2635          DO 14200 I = IBIND , IEIND
2636    C
2637          DO 14200 J = 1     ,    12
2638    C
2639          X(I) = X(I) + KGIRW(J,I) * BRHS(IBFR+J)
2640    C
2641    14200 CONTINUE
2642    C
2643    14300 CONTINUE
2644    C
2645    C
2646    14400 CONTINUE
2647    C
2648    C*    CONSIDER ALL NODES OF THE SECTOR IN THE ORDER THEY APPEAR IN
2649    C*    THE STIFFNESS MATRIX.
2650    C
2651          DO 14700 IORND = 1 , NNDSH
```

```
C
C*    DETERMINE THE INODE-TH NODE.
C
      INODE = SEQND(IORND)
C
C*    DETERMINE THE BEGINNING AND THE ENDING POSITIONS OF THE DEGREES
C*    OF FREEDOM OF THE CURRENT NODE IN VECTORS CORLD, BRHS, AND X.
C
      IBRW = (IORND - 1) * 12 + 1
      IERW = IBRW + 11
C
C*    IF THE NODE IS AN INTERNAL NODE, GO TO THE APPROPRIATE PROGRAM
C*    SEGMENT.
C
      IF ( INTND(INODE) . EQ . 3HYES )            GO TO 14550
C
C*    INODE IS EXTERNAL.
C
C*    PUT THE ENTRIES OF X ( = Y1 ) CORRESPONDING TO THE CURRENT
C*    EXTERNAL NODE IN VECTOR BRHS ( = X ) AND THEN SET THE SAME ENTRIE
C*    OF X EQUAL TO ZERO TO PREPARE IT FOR CARRING OUT THE
C*    MULTIPLICATION:
C
C                          KIE-T * KII-INV * X-I
C
C*    BY SUBROUTINE SKYSOL, WHERE  X-I ARE THE INTERNAL ENTRIES OF X.
C
      DO 14500 I = IBRW , IERW
C
      BRHS(I) = X(I)
      X(I) = 0.0
C
14500 CONTINUE
C
      GO TO 14700
C
C*    INTERNAL NODES.
C
C*    CARRY OUT THE SUBTRACTION
C
C                       KGIE - KGII * KII-INV * KIE
C
C*    FOR THE IROW-TH COLUMN OF KGIE. PUT THE FINAL RESULT IN THE
C*    INTERNAL ENTRIES OF X ( = Y1 ) NOW CONTAINING THE PRODUCT
C*    KGII * KII-INV * KIE
C
14550 DO 14600 I = IBRW , IERW
C
      X(I) = CORLD(I) - X(I)
C
14600 CONTINUE
```

```
C
14700 CONTINUE
C
C*    NOW THE INTERNAL ENTRIES OF VECTOR X CONTAIN THE IROW-TH COLUMN
C*    OF   SUM-I = (KGIE - KGII * KII-INV * KIE)  AND THE EXTERNAL
C*    ENTRIES CONTAIN ZEROS. VECTOR CORLD IS NOT NEEDED AND UNTIL THE
C*    NEXT IROW AND CAN BE USED AS WORKSPACE.
C
C
C***  PASS VECTOR X TO SUBROUTINE SKYSOL WITH PARAMETER IBX SET EQAUL TO
C***  1   . ON RETURN, THE EXTERNAL ENTRIES OF X CONTAIN THE IROW-TH
C***  COLUMN OF THE PRODUCT
C
C                KIE-T - KII-INV * SUM-I
C
      CALL SKYSOL (KSECT, NOFSC, LD, IOP, 1, X, CORLD,
     + IREF, IAFILE, WKAREA, DELTA)
C
C*    ADD THE EXTERNAL ENTRIES OF X TO THE CORRESPONDING ENTRIES IN
C*    BRHS.
C
C*    CONSIDER ALL SHELL NODES IN THE ORDER THEY APPEAR IN THE STIFFNESS
C*    MATRIX.
C
      DO 14900 IORND = 1 , NNDSH
C
C*    DETERMINE THE IORND-TH NODE.
C
      INODE = SEQND(IORND)
C
C*    IF THE NODE IS INTERNAL, SKIP.
C
      IF ( INTND(INODE) . EQ . 3HYES )          GO TO 14900
C
C*    FIND THE BEGINING AND ENDING ROW NUMBERS ASSOCIATED WITH THE
C*    CURRENT EXTERNAL NODE.
C
      IBRW = ( NDORD(INODE) - 1 ) * 12 + 1
      IERW = IBRW + 11
C
C*    ADD THE EXTERNAL ENTRIES OF X TO THE ASSOCIATED ENTRIES OF BRHS,
C*    NOW CONTAINING THE PRODUCT
C
C                      KGEI * KII-INV * KIE
C
      DO 14800 I = IBRW , IERW
C
      BRHS(I) = BRHS(I) + X(I)
C
14800 CONTINUE
C
```

```
14900 CONTINUE
C
C*    NOW THE EXTERNAL ENTRIES OF BRHS(NDFSC) CONTAIN THE IROW-TH
C*    COLUMN OF KG0EE(NDFEX,NDFEX).
C
C
C***  NOW REARRANGE THE ENTRIES OF BRHS SUCH THAT THE ENTRIES
C***  ASSOCIATED WITH THE EXTERNAL 2 NODES APPEAR FIRST FOLLOWED BY
C***  THOSE OF THE EXTERNAL 1 NODE.
C
      CALL ARRANG (NRWND, NDFSC, NDEX1, NDEX2, NDORD, X, BRHS)
C
      IKG0EE = IKG0EE + 1
C
C*    PUT THE TOP NDFEX ENTRIES OF BRHS, WHICH IS THE ICOL-TH COLUMN OF
C*    KG0EE =  KGEI * KGII-INV * KGIE  ON DISK FILE DSKF1.
C
      WRITE (DSKF2, REC=NKG0EE+IKG0EE)  (BRHS(I), I=1, NDFEX)
C
15000 CONTINUE
C
15100 CONTINUE
C
15120 CONTINUE
C
C
C
C***  READ THE GEOMETRIC STIFFNESS MATRIX DUE TO THE LIVE LOADS
C***  FROM DISK FILE DSKF1, BECAUSE, NOW KSECT IS THE FACTORIZED
C***  CONVENTIONAL STIFFNESS MATRIX.
C
      CALL REDKSC (DSKF1, NKGSC, 0, LRDF1, NDFSK, KSECT)
C
C
C
C
C
C***  NOW KSECT CONTAINS THE GEOMETRIC STIFFNESS MATRIX OF THE
C***  STRUCTURE DUE TO LIVE LOADS.
C***  SET THE ROWS AND COLUMNS OF KSECT CORRESPONDING TO THE
C***  PRESCRIBED DISPLACEMENTS EQUAL TO ZERO (AND THE DIAGONAL
C***  ENTRIES EQUAL TO ONE) BEFORE FORMING KGBNIF(NDFIF,NDFIF).
C
C
C
C***  WITH THIS MODIFICATION THE EIGENPROBLEMS SLOVED IN SLVEIG
C***  APPEARS AS FOLLOWS:
C
C       KBNIF(NDFIF,NDFIF) - LAMBDA * KGBNIF(NDFIF,NDFIF) = 0.0
C
```

```
2805  C
2806  C
2807  C          (  I  .  .  .  I                    I  .  0  .  I  )
2808  C          (  I  .  1  .  I     +    L  *     I  0  1  0  I  )  *  X = 0
2809  C          (  I  .  .  .  I                    I  .  0  .  I  )
2810  C
2811  C
2812  C
2813  C
2814  C
2815  C
2816  C
2817  C
2818  C*    LOOP NINFA (NUMBER OF INTERFACES) TIMES.
2819  C
2820  15150 DO 15170 N = 1 , NINFA
2821  C
2822  C
2823  C                                K  G  B  N  I  F
2824  C
2825  C***  GENERATE THE COMPLEX MATRIX  KGBNIF(NDFIF,NDFIF)  DIRECTLY
2826  C***  FROM VECTOR KGLLD(NDFSK) ACCORDING TO:
2827  C
2828  C       KGBNIF(NDFIF,NDFIF) =
2829  C       KGE2(NDFIF,NDFIF) + KGE1(NDFIF,NDFIF) + KGE21(NDFIF,NDFIF) *
2830  C       CEXP(I*2*PI*N/NINFA) + KGE12(NDFIF,NDFIF) * CEXP(-I*2*PI*N/NINFA)
2831  C
2832  C***  NOTE, HERE KGE2, KGE1, KGE21, AND KGE12 ARE NOT CONDENSED. ALSO
2833  C***  NOTE, THAT THE ASSEMBLY OF KGE2 OF SECTOR N WITH KGE1 OF SECTOR
2834  C***  (N+1) IS IMPLIED IN THE ABOVE OPERATION.
2835  C
2836        CALL KBNINF (MDFIF, NNDSH, NNDE1, NDFND, NNDEX, NDFIN,
2837       + NDFIF, NINFA, N, NDEX1, NDEX2, INTND, SEQND, NDORD, IPNTR, ICONN,
2838       + LD, KSECT, KGBNIF)
2839  C
2840  C
2841  C
2842  C*    IF NO INTERNAL NODES EXIST, THERE IS NO STATIC CONDENSATION.
2843  C
2844        IF (NNDIN . EQ . 0)                          GO TO 15165
2845  C
2846  C
2847  C
2848  C
2849  C
2850  C
2851  C                              K  G  0  E  E  S
2852  C
2853  C
2854  C***  COMPLETE THE GENERATION OF KGBNIF(NDFIF,NDFIF), BY SUBTRACTING
2855  C***  THE TERM
```

```
C                KGOEES(NDFIF,NDFIF) = KGEI * KGII-INV * KGIE
C***  ARISING FROM THE STATIC CONDENSATION OF THE INTERNAL DEGREES OF
C***  FREEDOM. NOTE THAT THE ASSEMBLY OF KGE2 OF SECTOR N AND KGE1 OF
C***  SECTOR (N+1) IS CARRIED OUT SIMULTANEOUSLY.
C
      CALL KBNIFO (DSKF2, MDFIF, N, NDFEX, NINFA, NDFIF, NKGOEE,
     + CORLD, KGBNIF)
C
C***  WRITE MATRIX KGBNIF(NDFIF, NDFIF) CALCULATED FOR THE CURRENT N
C***  ON DISK FILE DSKF1 USING THE RECORD LENGTH OF LRDF1.
C
15165 CALL WRTKBN (DSKF1, JREMN, MDFIF, N, NCOLS, NDFIF,
     + NKGBNI, NR, NRECS, KGBNIF)
C
C
15170 CONTINUE
C
C
15175 RETURN
      END
C
C
C
C*                                                                    *
C**                                                                   **
C***              O V E R L A Y      2      5                         ***
C**                                                                   **
C*                                                                    *
C
C
C
      SUBROUTINE TWOFIV
C
C
C*          I N T E R O V E R L A Y   P A R A M E T E R S
C
      PARAMETER  (JNDSH = 30,  JDFSC = 12 * JNDSH,
     + JLD = JDFSC+1,  JSECT = 60,  JRWND = 15,  JRWEL = JRWND-1,
     + JELSH = 28, JZNSH = 1,  JMTSH = 1,
     + JSTRN = 1,   JNDSR = JRWND,  JELSR = JNDSR-1,
     + JMTSR = 3,  JIMSR = 1,  JELISR = 5,  JMTISR = 1,  JRING = 2,
     + JNDRN = 5,  JELRN = 3,  JMTRN = 2,  JIMRN = 15,  JELIRN = 20,
     + JMTIRN = 10,  JGAUS = 12,  JCRSM = 5,  JPRDS = JNDSH,
     + JLDCA = 2,  JCNLD = 5,  JDTLD = 2,  J4ULD = 1,  JZNTP = 1,
     + JZNSL = 6,  JDFIF = 12 * JRWND,
     + JICON = 109, JDFSL = 64801)
C
C
C
C                    L O C A L    V A R I A B L E S
C
```

```
      DIMENSION  SPACE(1260)
C
      COMPLEX  KGBNIF(JDFIF,JDFIF)
C
C                 I N T E R - O V E R L A Y    V A R I A B L E S
C
C                      BETWEEN  OVRELAYS   (0,0), (1,0), (2,0)
C
      CHARACTER*10  STRTYP, AUTGN, SLDSP, SLRAC, SLSTS, SLBKL,
     + PLSTR, PLMSH, PLDSP, PLSTS, PLMSP
C
      INTEGER  CSDCL, CSDMC, DSKF1, DSKF2, SLDLD,
     + SLVSC(JLDCA,JZNSL,2), ROWSH(JRWND,2),
     + SHMTI(JZNSH), SHMTJ(JZNSH), CSDSG(JCRSM), RSDOF(JPRDS,12),
     + CSPRD(JPRDS), SCNLI(JLDCA,JCNLD), SCNLJ(JLDCA,JCNLD),
     + CSCNL(JLDCA,JCNLD), TYDTL(JLDCA,JDTLD), DSDTL(JLDCA,JDTLD),
     + DRDTL(JLDCA,JDTLD,5), CSDTL(JLDCA,JDTLD), RFDTL(JLDCA,JDTLD),
     + SCTPI(JLDCA,JZNTP), SCTPJ(JLDCA,JZNTP), SEQND(JNDSH),
     + NDORD(JNDSH), IPNTR(JNDSH), ICONN(JICON), ROWSA(2)
C
      REAL  ORIGN(JRWEL,2), KCINV(JELSH,2,2), KSECT(JDFSL),
     + ITCNL(JLDCA,JCNLD,9), ITDTL(JLDCA,JDTLD,2), ITWSH(JMTSH,3)
C
      DIMENSION  NODSH(JELSH,3), MTDSH(JZNSH), DNSSH(JELSH),
     + PRPSH(JMTSH,8), MTSHL(JELSH),
     + TSHEL(JMTSH), TRMSH(JMTSH), ARASH(JELSH), NODSR(JELSR,2),
     + MTDSR(JMTSR), ECNSR(JMTSR), TRMSR(JMTSR),
     + PRPSR(JMTSR,9), MTSTR(JELSR), CLSTR(JELSR,7,2),
     + NODISR(JELISR,2), NDIMSR(JIMSR,2), MTDISR(JMTISR),
     + ECISR(JMTISR), TMISR(JMTISR),
     + PRPISR(JMTISR,9), MTISR(JELISR), CLISR(JELISR,7,2),
     + NODRN(JELRN,2), MTDRN(JMTRN), TRMRN(JMTRN), PSNRN(JMTRN,3),
     + PRPRN(JMTRN,9), MTRNG(JELRN), NRNGEL(JRING), NODIRN(JELIRN,2),
     + NDIMRN(JIMRN,2), MTDIRN(JMTIRN), PSNIRN(JMTIRN,3),
     + TMIRN(JMTIRN), PRPIRN(JMTIRN,9), MTIRN(JELIRN), NIRNEL(JIMRN),
     + CRDCL(JNDSH,3), CRDMC(JNDSH,3), GUSWT(JGAUS), GUSTR(JGAUS,3),
     + GUSCL(JELSH,JGAUS,3), DVDTL(JLDCA,JDTLD), NODPD(JPRDS),
     + PRSDS(JPRDS,12), NREST(JPRDS),
     + NDCNL(JLDCA,JCNLD), TPNDI(JLDCA,JZNTP), TPNDE(JLDCA,JZNTP),
     + DRCOS(JCRSM,3,3), LD(JLD), NDJNC(JNDSH), CORLD(JDFSC),
     + BRHS(JDFSC), X(JDFSC)
C
C       BETWEEN OVERLAYS  (2,0), (2,1), (2,2), (2,3), (2,4), (2,5), (2,6)
C
      DIMENSION  ENLWT(JDFSC), ECLWT(JELSH,12), ENLAUL(JAULD,JDFSC),
     + ECLAUL(JAULD,JELSH,12), ENLTP(JZNTP,JDFSC),
     + ECLTP(JZNTP,JELSH,12), INDIC(JSECT)
C
      COMPLEX  ENLSA(4,JDFSC), ECLSA(4,JELSH,12), KBNIF(JDFIF,JDFIF)
C
C
```

```
      LOGICAL  PDCHEK, SINGAB
C
      EQUIVALENCE  ( KBNIF(1,1), SPACE(1) )
C
C
      COMMON  STRTYP, AUTGN, SLDSP, SLRAC, SLSTS, SLBKL,
     + PLSTR, PLMSH, PLDSP, PLSTS, PLMSP,
     + MSECT, MSTRN, MNDSR, MRING, MNDRN, MGAUS,
     + NNDSH, NELSH, NMTSH, NSTRN, NNDSR, NELSR, NMTSR, NIMSR,
     + NELISR, NMTISR, NRING, NNDRN, NELRN, NMTRN, NIMRN, NELIRN,
     + NMTIRN, NBRAC, NELBR, NMTBR, NGAUS, NCPSM, NPRDS, NLDCA,
     + NSECT, NRWEL, NRWND, NOFND, NNDE1, NNDE2, NNDIN,
     + NNTSH, NBTSH, NGTSH, NSTSSH, NKCTND, NNTSR, NBTSR, NGTSR,
     + NTFSR, NSTSSR, NNTISR, NBTISR, NGTISR, NTFISR, NSTSIS, NNTRN,
     + NBTRN, NGTRN, NTFRN, NSTSRN, NNTIRN, NBTIRN, NGTIRN, NTFIRN,
     + NSTSIR, NTFBR, NSTSBR, NKRWPD, NCRLD, NFRCN, NABIF, NASCT,
     + NACNT, NFBSC, NRACT, NKSCT, NKSCF, NKKGW, NKGSC, NKGICL, NKBNIF,
     + NKGBNI, NEGVC, NKKGCN, NKGCN, NMSND, NMSCT, NKOEE, NKGOEE,
     + CSDCL, CSDMC, MCCOD, ILDCA, LCNLD, LDTLD, LZNTP, LZNSL,
     + DSKF1, DSKF2, IN, IO,
     + FEED, RTOP, RBAS, HITE, ELEV, THATAD,
     + TITLE(60), LDCAS(2), NCNLD(2), NDTLD(2), NZNTP(2), NZNSL(2),
     + RATIO(14), INDEX(45), NDCON(200), NDEX1(15), NDEX2(15),
     + NDINT(15), INTNO(36), WKAREA(192)
C
C
      COMMON  / COMMN1 /
     + SLVSC, ROWSH, SHMTI, SHMTJ, CSDSG,
     + RSDOF, CSPRD, SCNLI, SCNLJ, CSCNL,
     + TYDTL, DSDTL, DRDTL, CSDTL, RFDTL,
     + SCTPI, SCTPJ, SEQND, NDORD, IPNTR,
     + ICONN, ROWSA,
     + ORIGN, KCINV, KSECT, ITCNL,
     + ITDTL, ITWSH,
     + NODSH, MTDSH, DNSSH, PRPSH, MTSHL,
     + TSHEL, TRMSH, ARASH,
     + NODSR, MTDSR, ECNSR, TRMSR,
     + PRPSR, MTSTR, CLSTR, NODISR, NDIMSR, MTDISR,
     + ECISR, TMISR, PRPISR, MTISR, CLISR, NODRN,
     + MTDRN, PSNRN, PRPRN, TRMRN, MTRNG
C
      COMMON  / COMMN2 /
     + NRNGEL, NDDIRN, NDIMRN, MTDIRN, PSNIRN,
     + TMIRN, PRPIRN, MTIRN, NIRNEL,
     + CRDCL, CRDMC, GUSWT, GUSTR, GUSCL,
     + DVDTL, NODPD,
     + PRSDS, NREST, NDCNL, TPNDI, TPNDE,
     + DRCDS, LD, NDJNC, CORLD, BRHS, X
C
      COMMON  / COMMN3 /
     + LROF1, MNDSH, MELSH, MMTSH, MELSR,
```

```
     +  MMTSR, MIMSR, MELISR, MMTISR, MELRN, MMTRN,
     +  MIMRN, MELIRN, MMTIRN, MGUSS, MZNTP, MZNSL, MRWND,
     +  MRWEL, MCRSM, MPROS, MLOCA, MCNLD, MDTLD, MAULD, MDFSC, MDFIF,
     +  NDFIF, NDFEX, NDFIN, NDFSC, NINFA, NDFBL, NNDEX, NBEG, NEND, IOP,
     +  IBX, PDCHEK, SINGAB, ICOND, IAFILE, IREF, DETCF, SLDLD, NDFSK
C
C       BETWEEN OVERLAYS  (2,0), (2,1), (2,2), (2,3), (2,4), (2,5), (2,6)
C
      COMMON  / COMMN4 /  NDFCN, NEIG, ENLWT, ECLWT, ENLAUL, ECLAUL,
     +  ENLTP, ECLTP, ENLSA, ECLSA, INDIC, KBNIF
C
C
C
C*    CALCULATE LAMBDA DEFINED IN THE EIGENVALUE PROBLEM:
C             (KSECT + KGDLD) - LAMBDA * KGLLD = 0.0
C     BY SOLVING
C      KBNIF(NDFIF, NDFIF) - LAMBDA * KGBNIF(NDFIF, NDFIF) = 0.0
C
C***  FOR ALL N'S ( N = 1 , ... NSECT), AND PRINT THE SMALLEST IN
C***  ABSOLUTE VALUE. LAMBDA MULTIPLIED BY THE LIVE LOADS GIVES THE
C***  APPLIED BUCKLING LOAD. ALSO THE MODES SHAPES OF THE BUCKLED
C***  STATE, ASSOCIATED WITH THIS LAMBDA, ARE PRINTED, FOR THE
C***  SECTORS FOR WHICH SOLUTION HAS BEEN REQUESTED.
C
      CALL BUCKLE (LRDF1, MNDSH, MELSR, MELISR, MIMSR, MELRN,
     +  MIMRN, MELIRN, MCRSM, MRWND, MRWEL,
     +  MLOCA, MZNSL, MDFIF, NDFSC, NDFSK,
     +  NDFIF, NBEG, NEND, SINGAB, PDCHEK, ICOND, IAFILE, DETCF,
     +  SPACE, KGBNIF)
C
C
      RETURN
      END
C
C
C*                                                                    *
C**                                                                   **
C***          O V E R L A Y     2     6                              ***
C**                                                                   **
C*                                                                    *
C
C
C
      SUBROUTINE TWOSIX
C
C
C*          I N T E R O V E R L A Y   P A R A M E T E R S
C
      PARAMETER  (JNDSH = 30,  JDFSC = 12 * JNDSH,
     +  JLD = JDFSC+1,  JSECT = 60,  JRWND = 15,  JRWEL = JRWND-1,
     +  JELSH = 28, JZNSH = 1,  JMTSH = 1,
```

```
     + JSTRN = 1,   JNDSR = JRWND,  JELSR = JNDSR-1,
     + JMTSR = 3,  JIMSR = 1,  JELISR = 5,  JMTISR = 1,  JRING = 2,
     + JNDRN = 5,  JELRN = 3,  JMTRN = 2,  JIMRN = 15,  JELIRN = 20,
     + JMTIRN = 10,  JGAUS = 12,  JCRSM = 5,  JPRDS = JNDSH,
     + JLDCA = 2,  JCNLD = 5,  JDTLD = 2,  JAULD = 1,  JZNTP = 1,
     + JZNSL = 6,  JDFIF = 12 * JRWND,
     + JICON = 109, JDFSL = 64801)
C
C
C                L O C A L   V A R I A B L E S
C
C
C
C
      DIMENSION  STRES(1260), TRFST(18,18)
C
C
C              I N T E R - O V E R L A Y   V A R I A B L E S
C
C                 BETWEEN  OVERLAYS   (0,0), (1,0), (2,0)
C
      CHARACTER*10  STRTYP, AUTGN, SLDSP, SLRAC, SLSTS, SLBKL,
     + PLSTR, PLMSH, PLDSP, PLSTS, PLMSP
C
      INTEGER  CSDCL, CSDMC, DSKF1, DSKF2, SLDLD,
     + SLVSC(JLDCA,JZNSL,2), ROWSH(JRWND,2),
     + SHMTI(JZNSH), SHMTJ(JZNSH), CSDSG(JCRSM), RSDDF(JPRDS,12),
     + CSPRD(JPRDS), SCNLI(JLDCA,JCNLD), SCNLJ(JLDCA,JCNLD),
     + CSCNL(JLDCA,JCNLD), TYDTL(JLDCA,JDTLD), DSDTL(JLDCA,JDTLD),
     + DRDTL(JLDCA,JDTLD,5), CSDTL(JLDCA,JDTLD), RFDTL(JLDCA,JDTLD),
     + SCTPI(JLDCA,JZNTP), SCTPJ(JLDCA,JZNTP), SEQND(JNDSH),
     + NDORD(JNDSH), IPNTR(JNDSH), ICONN(JICON), ROWSA(2)
C
      REAL  ORIGN(JRWEL,2), KCINV(JELSH,2,2), KSECT(JDFSL),
     + ITCNL(JLDCA,JCNLD,9), ITDTL(JLDCA,JDTLD,2), ITWSH(JMTSH,3)
C
      DIMENSION  NODSH(JELSH,3), MTDSH(JZNSH), DNSSH(JELSH),
     + PRPSH(JMTSH,8), MTSHL(JELSH),
     + TSHEL(JMTSH), TRMSH(JMTSH), ARASH(JELSH), NODSR(JELSR,2),
     + MTDSR(JMTSR), ECNSR(JMTSR), TRMSR(JMTSR),
     + PRPSR(JMTSR,9), MTSTR(JELSR), CLSTR(JELSR,7,2),
     + NODISR(JELISR,2), NDIMSR(JIMSR,2), MTDISR(JMTISR),
     + ECISR(JMTISR), TMISR(JMTISR),
     + PRPISR(JMTISR,9), MTISR(JELISR), CLISR(JELISR,7,2),
     + NODRN(JELRN,2), MTDRN(JMTRN), TRMRN(JMTRN), PSNRN(JMTRN,3),
     + PRPRN(JMTRN,9), MTRNG(JELRN), NRNGEL(JRING), NODIRN(JELIRN,2),
     + NDIMRN(JIMRN,2), MTDIRN(JMTIRN), PSNIRN(JMTIRN,3),
     + TMIRN(JMTIRN), PRPIRN(JMTIRN,9), MTIRN(JELIRN), NIRNEL(JIMRN),
     + CRDCL(JNDSH,3), CRDMC(JNDSH,3), GUSWT(JGAUS), GUSTR(JGAUS,3),
     + GUSCL(JELSH,JGAUS,3), DVDTL(JLDCA,JDTLD), NODPD(JPRDS),
     + PRSDS(JPRDS,12), NREST(JPRDS),
```

```
3111          + NDCNL(JLDCA,JCNLD), TPNDI(JLDCA,JZNTP), TPNDE(JLDCA,JZNTP),
3112          + DRCOS(JCRSM,3,3), LD(JLD), NDJNC(JNDSH), CORLD(JDFSC),
3113          + BRHS(JDFSC), X(JDFSC)
3114    C
3115    C
3116    C       BETWEEN OVERLAYS  (2,0), (2,1), (2,2), (2,3), (2,4), (2,5), (2,6)
3117    C
3118           DIMENSION  ENLWT(JDFSC), ECLWT(JELSH,12), ENLAUL(JAULD,JDFSC),
3119          + ECLAUL(JAULD,JELSH,12), ENLTP(JZNTP,JDFSC),
3120          + ECLTP(JZNTP,JELSH,12), INDIC(JSECT)
3121    C
3122           COMPLEX  ENLSA(4,JDFSC), ECLSA(4,JELSH,12), KBNIF(JDFIF,JDFIF)
3123    C
3124           LOGICAL  PDCHEK, SINGAB
3125    C
3126    C
3127    C
3128    C
3129           COMMON  STRTYP, AUTGN, SLDSP, SLRAC, SLSTS, SLBKL,
3130          + PLSTR, PLMSH, PLDSP, PLSTS, PLMSP,
3131          + MSECT, MSTRN, MNDSR, MRING, MNDRN, MGAUS,
3132          + NNDSH, NELSH, NMTSH, NSTRN, NNDSR, NELSR, NMTSR, NIMSR,
3133          + NELISR, NMTISR, NRING, NNDRN, NELRN, NMTRN, NIMRN, NELIRN,
3134          + NMTIRN, NBRAC, NELBR, NMTBR, NGAUS, NCRSM, NPRDS, NLDCA,
3135          + NSECT, NRWEL, NRWND, NDFND, NNDE1, NNDE2, NNDIN,
3136          + NNTSH, NBTSH, NGTSH, NSTSSH, NKCTND, NNTSR, NBTSR, NGTSR,
3137          + NTFSR, NSTSSR, NNTISR, NBTISR, NGTISR, NTFISR, NSTSIS, NNTRN,
3138          + NBTRN, NGTRN, NTFRN, NSTSRN, NNTIRN, NBTIRN, NGTIRN, NTFIRN,
3139          + NSTSIR, NTFBR, NSTSBR, NKRWPD, NCRLD, NFRCN, NABIF, NASCT,
3140          + NACNT, NFBSC, NRACT, NKSCT, NKSCF, NKKGW, NKGSC, NKGICL, NKBNIF,
3141          + NKGBNI, NEGVC, NKKGCN, NKGCN, NMSND, NMSCT, NKOEE, NKGOEE,
3142          + CSOCL, CSOMC, MCCOD, ILDCA, LCNLD, LDTLD, LZNTP, LZNSL,
3143          + DSKF1, DSKF2, IN, IO,
3144          + FEED, RTOP, RBAS, HITE, ELEV, THATAD,
3145          + TITLE(60), LDCAS(2), NCNLD(2), NDTLD(2), NZNTP(2), NZNSL(2),
3146          + RATIO(14), INDEX(45), NDCON(200), NDEX1(15), NDEX2(15),
3147          + NDINT(15), INTND(35), WKAREA(192)
3148    C
3149    C
3150           COMMON  / COMMN1 /
3151          + SLVSC, ROWSH, SHMTI, SHMTJ, CSDSG,
3152          + RSDOF, CSPRO, SCNLI, SCNLJ, CSCNL,
3153          + TYDTL, DSDTL, DRDTL, CSDTL, RFDTL,
3154          + SCTPI, SCTPJ, SEQND, NDDRD, IPNTR,
3155          + ICONN, ROWSA,
3156          + ORIGN, KCINV, KSECT, ITCNL,
3157          + ITDTL, ITWSH,
3158          + NDDSH, MTDSH, DNSSH, PRPSH, MTSHL,
3159          + TSHEL, TRMSH, ARASH,
3160          + NDDSR, MTDSR, ECNSR, TRMSR,
3161          + PRPSR, MTSTR, CLSTR, NDDISR, NDIMSR, MTDISR,
```

```
3162          + ECISR, TMISR, PRPISR, MTISR, CLISR, NODRN,
3163          + MTDRN, PSNRN, PRPRN, TRMRN, MTRNG
3164    C
3165           COMMON  / COMMN2 /
3166          + NRNGEL, NODIRN, NDIMRN, MTDIRN, PSNIRN,
3167          + TMIRN, PRPIRN, MTIRN, NIRNEL,
3168          + CRDCL, CRDMC, GUSWT, GUSTR, GUSCL,
3169          + DVDTL, NODPD,
3170          + PRSDS, NREST, NDCNL, TPNDI, TPNDE,
3171          + DRCDS, LD, NDJNC, CORLD, BRHS, X
3172    C
3173           COMMON  / COMMN3 /
3174          + LRDF1, MNDSH, MELSH, MMTSH, MELSR,
3175          + MMTSR, MIMSR, MELISR, MMTISR, MELRN, MMTRN,
3176          + MIMRN, MELIRN, MMTIRN, MGUSS, MZNTP, MZNSL, MRWND,
3177          + MRWEL, MCRSM, MPRDS, MLDCA, MCNLD, MDTLD, MAULD, MDFSC, MDFIF,
3178          + NDFIF, NDFEX, NDFIN, NDFSC, NINFA, NDFBL, NNDEX, NBEG, NEND, IOP,
3179          + IBX, PDCHEK, SINGAB, ICOND, IAFILE, IREF, DETCF, SLDLD, NDFSK
3180    C
3181    C
3182    C       BETWEEN OVERLAYS  (2,0), (2,1), (2,2), (2,3), (2,4), (2,5), (2,6)
3183    C
3184           COMMON  / COMMN4 /  NDFCN, NEIG, ENLWT, ECLWT, ENLAUL, ECLAUL,
3185          + ENLTP, ECLTP, ENLSA, ECLSA, INDIC, KBNIF
3186    C
3187    C
3188           LOGICAL  SYMLD
3189    C
3190    C
3191           SYMLD = .TRUE.
3192    C
3193    C*     IF THE CURRENT SOLUTION IS FOR DEAD LOAD ALONE, SKIP OVER THE
3194    C*     FOLLOWING PORTION, BECAUSE THE DEAD LOAD IS ALREADY KNOWN TO
3195    C*     BE CIRCUMFERENTIALLY UNIFORM AND SYMMETRIC.
3196    C
3197           IF (SLDLD . EQ . 3HYES)                          GO TO 350
3198    C
3199    C*     DETERMINE IF THE CURRENT SPECIFIED LIVE LOADS ARE CIRCUMFERENIALLY
3200    C*     UNIFORM AND SYMMETRIC OR NOT. THIS INFORMATION WILL BE USED TO
3201    C*     DECIDE WHETHER A 2-DIMENSIONAL OR A CONTOUR SHOULD BE PLOTTED FOR
3202    C*     DISPLACEMENTS, STRESSES AND MODESHAPES.
3203    C
3204    C*     IF NO CONCENTRATED LOADS EXIST FOR THE CURRENT LOADING CASE, SKIP.
3205    C
3206           IF (LCNLD . EQ . 0)                              GO TO 150
3207    C
3208    C
3209    C*     CONSIDER ALL CONCENTRATED NODAL LOADS OF THE CURRENT LOADING CASE.
3210    C
3211           DO 100 ICNLD = 1 , LCNLD
3212    C
```

```
3213          ISC = SCNLI(ILDCA,ICNLD)
3214          JSC = SCNLJ(ILDCA,ICNLD)
3215    C
3216          IF (ISC . LE . JSC . AND . (JSC-ISC+1) . NE . NSECT)
3217         +                                                  SYMLD = . FALSE .
3218    C
3219          IF (ISC . GT . JSC . AND . (NSECT-JSC+ISC-1) . NE . NSECT)
3220         +                                                  SYMLD = . FALSE .
3221    C
3222          IF ( . NOT . SYMLD )                              GO TO 350
3223    C
3224      100 CONTINUE
3225    C
3226    C
3227    C*    IF NO DISTRIBUTED LOADS EXIST FOR THE CURRENT LOADING CASE, SKIP.
3228    C
3229      150 IF (LDTLD . EQ . 0)                               GO TO 250
3230    C
3231    C*    CONSIDER ALL DISTRIBUTED LOADS.
3232    C
3233          DO 200 IDTLD = 1 , LDTLD
3234    C
3235          IF (TYDTL(ILDCA,IDTLD) . NE . 4HUNIFM . AND . TYDTL(ILDCA,IDTLD)
3236         +    . NE . 4HLINP . AND . TYDTL(ILDCA,IDTLD) . NE . 4HSNOW)
3237         +                                                  SYMLD = . FALSE .
3238    C
3239          IF (RFDTL(ILDCA,IDTLD) . NE . 1)                  SYMLD = . FALSE .
3240    C
3241          IF ( . NOT . SYMLD )                              GO TO 350
3242    C
3243      200 CONTINUE
3244    C
3245    C
3246    C*    IF NO THERMAL ZONES EXIST FOR THE CURRENT LOADING CASE, SKIP.
3247    C
3248      250 IF (LZNTP . EQ . 0)                               GO TO 350
3249    C
3250    C
3251    C*    CONSIDER ALL THERMAL ZONES.
3252    C
3253          DO 300 IZNTP = 1 , LZNTP
3254    C
3255          ISC = SCTPI(ILDCA,IZNTP)
3256          JSC = SCTPJ(ILDCA,IZNTP)
3257    C
3258          IF (ISC . LE . JSC . AND . (JSC-ISC+1) . NE . NSECT)
3259         +                                                  SYMLD = . FALSE .
3260    C
3261          IF (ISC . GT . JSC . AND . (NSECT-JSC+ISC-1) . NE . NSECT)
3262         +                                                  SYMLD = . FALSE .
3263    C
```

```
      IF ( . NOT . SYMLD )                                  GO TO 350
C
C
  300 CONTINUE
C
C
C*    IF PLOTTING OF THE DISPLACEMENTS HAS NOT BEEN REQUESTED, SKIP
C
  350 IF (PLOSP . NE . 'YES')                               GO TO 450
C
C
C
C
C*    IF NO STRINGERS, INTERMEDIATE STRINGERS, RINGS OR INTERMEDIATE
C*    RINGS EXIST, SKIP.
C
      IF (NSTRN . EQ . 0 . AND . NIMSR . EQ . 0 . AND . NRING .
     + EQ . 0 . AND . NIMRN . EQ . 0)                       GO TO 425
C
C
C*** PLOT THE DEFORMED SHAPE OF THE STRUCTURE (IN SOLID LINES)
C*** ALONG WIDTH THE ORIGINAL UNDEFORMED SHAPE (IN DASH LINES).
C*** NOTE: DISPLACEMENTS ARE PLOTTED ONLY FOR THE SECTORS FOR
C*** WHICH SOLUTION HAS BEEN REQUESTED.
C
      CALL PLOFSH (CSDMC, DSKF1, ELEV, HITE, ILOCA, LZNSL,
     + MNDSH, MELSR, MELISR, MIMSR, MELRN, MIMRN, MELIRN, MCRSM, MLOCA,
     + MZNSL, MRWND, MRWEL, MCCOD, NSECT, NRWND, NNDSH, NSTRN, NELSR,
     + NIMSR, NELISR, NRING, NELRN, NIMRN, NELIRN, NCRSM, NPRDS, NNDIN,
     + NASCT, NOFIF, NOFSC, RTOP, RBAS, STRTYP, THATAD, SLVSC, CRDCL,
     + ROWSH, ORIGN, NODSR, NODISR, NDIMSR,
     + NODRN, NDIMRN, NODIRN, CSDSG, NDDPD, CSPRD,
     + NDEX1, NDEX2, INTND, NDJNC, DRCOS, NDORD, X, BRHS, WKAREA)
C
      RETURN
C
C*    NEIG SIMPLY INDICATES THAT THE DISPLACEMENTS ARE TO
C*    BE PLOTTED ONLY ONCE.
C
  425 NEIG = 4HPLOS
C
C
C
C                         D I S P L A C E M E N T S
C
C
C
C
C*** PLOT THE NODAL IN-PLANE DISPLACEMENTS OF THE SHELL, AND THE
C*** NODAL DISPLACEMENTS OF STRINGERS AND INTERMEDIATE STRINGERS
C*** RINGS AND INTERMEDIATE RINGS FOR THE CURRENT LOADING CASE.
```

```
C
C     S H E L L.
C
C***  WHEN THE DISPLACEMENTS ARE UNIFORM CIRCUMFERENTIALLY WITHIN A
C***  SECTOR, ONLY THE MERIDIONAL (U) AND THE NORMAL (W), NODAL
C***  DISPLACEMENTS OF THE SHELL MID-SURFACE, ARE PLOTTED IN A SIMPLE
C***  2-DIMENSIONAL GRAPH, VERSUS THE MERIDIONAL COORDINATE.
C***  IN THIS CASE, V = 0, EVERYWHERE.
C***  WHEN THE DISPLACEMENTS ARE NOT CIRCUMFERENTIALLY UNIFORM
C***  WITHIN A SECTOR, BUT ARE THE SAME FROM SECTOR TO SECTOR, CONTOURS
C***  OF DISPLACEMENTS U, V, AND W ARE PLOTTED FOR A TYPICAL SECTOR.
C***  WHEN DISPLACEMENTS VARY CIRCUMFERENTIALLY WITHIN THE SECTOR
C***  AS WELL AS FORM SECTOR TO SECTOR, CONTOURS OF DISPLACEMENTS
C***  ARE PLOTTED FOR THE SECTORS FOR WHICH SOLUTION HAS BEEN REQUESTED.
C
C     S T R I N G E R S  &  I N T E R M E D I A T E  S T R I N G E R S.
C
C***  WHEN THE DISPLACEMENTS DO NOT VARY FROM SECTOR TO SECTOR,
C***  U AND W, THE MERIDIONAL AND THE NORMAL DISPLACEMENTS,
C***  RESPECTIVELY, OF THE THE STRINGERS AND INTERMEDIATE STRINGERS
C***  ARE PLOTTED VERSUS THIER MERIDIONAL COORDINATE XI, IN A
C***  2-DIMENSIONAL GRAPH, FOR A TYPICAL SECTOR.
C***  IF HOWEVER, THE DISPLACEMENTS DO NOT VARY FROM SECTOR TO SECTOR,
C***  2-DIMENSIONAL GRAPHS OF U, V, AND W, THE MERIDIONAL, THE
C***  CIRCUMFERENTIAL AND THE NORMAL DISPLACEMENTS, RESPECTIVELY,
C***  ARE PLOTTED VERSUS THE MERIDIONAL COORDINATE, FOR ALL SECTORS
C***  FOR WHICH SOLUTION HAS BEEN REQUESTED.
C
C     R I N G S  &  I N T E R M E D I A T E  R I N G S.
C
C***  WHEN THE DISPLACEMENTS DO NOT VARY CIRCUMFERENTIALLY, NO PLOT
C***  IS PRODUCED FOR RINGS AND INTERMEDIATE RINGS.
C***  WHEN THE DISPLACEMENTS WITHIN A SECTOR DO VARY CIRCUMFERENTIALLY,
C***  BUT THE DISPLACEMENTS OF ALL SECTORS ARE THE SAME, 2-DIMENSIONAL
C***  GRAPHS OF V AND W, THE CIRCUMFERENTIAL AND NORMAL DISPLACEMENTS,
C***  RESPECTIVELY, ARE PLOTTED VERSUS ETA, THE CIRCUMFERENTIAL
C***  COORDINATE, FOR ONLY A TYPICAL SECTOR.
C***  WHEN THE DISPLACEMENTS VARY CIRCUMFERENTIALLY WITHIN A SECTOR
C***  AS WELL AS FROM SECTOR TO SECTOR, CONTOURS OF NODAL DISPLACEMENTS
C***  U, V, AND W (MERIDIONAL, CIRCUMFERENTIAL, AND NORMAL,
C***  RESPECTIVELY) ARE PLOTTED FOR ALL SECTORS FOR WHICH SOLUTION HAS
C***  BEEN REQUESTED.
C
C
C        THE 'SCIENTIFIC GRAPHICS SYSTEM', SDG, IS UTILIZED
C        HERE TO PLOT THE CURVES AND CONTOURS. THIS SOFTWARE
C        HAS BEEN DEVELOPED AT 'NCAR', 'NATIONAL CENTER FOR
C        ATMOSPHERIC RESEARCH', IN BOULDER COLORADO.
C
C
      CALL PLTDIS (DSKF1, ILOCA, LCNLD, LINSL, MNDSH, MELSH,
```

```
3366        + MELSR, MIMSR, MELISR, MELRN, MIMRN, MELIRN, MRWND, MLDCA,
3367        + MZNSL, NSECT, NNDSH, NELSH, NASCT, NACNT, NSTRN, NELSR, NTFSR,
3368        + NIMSR, NELISR, NTFISR, NRING, NELRN, NTFRN, NIMRN, NELIRN,
3369        + NTFIRN, NRWND, NRWEL, NDFSC, STRTYP, SLRAC, SYMLD, NODSH,
3370        + CRDCL, CRDMC, NODSR, NODISR, NDIMSR, NODRN, NRNGEL, NODIRN,
3371        + NDIMRN, SLVSC, NDEX1, NDEX2, NDORD, ROWSH, NDJNC, TRFST,
3372        + CORLD, KSECT)
3373  C
3374  C
3375  C*    IF PLOTTING OF THE GENERALIZED STRESSES HAS NOT BEEN REQUESTED,
3376  C*    SKIP.
3377  C
3378    450 IF (PLSTS . NE . 'YES')                          GO TO 550
3379  C
3380  C
3381  C                              S T R E S S E S
3382  C
3383  C
3384  C***  PLOT THE GENERALIZED STRESSES FOR THE SHELL, STRINGER,
3385  C***  INTERMEDIATE STRINGERS, RINGS, AND THE INTERMEDIATE
3386  C***  RINGS.
3387  C
3388  C     S H E L L.
3389  C
3390  C***  WHEN THE GENERALIZED STRESSES ARE UNIFORM CIRCUMFERENTIALLY
3391  C***  WITHIN A SECTOR, ONLY THE MERIDIONAL AND CIRCUMFERENTIAL
3392  C***  PRINCIPAL STRESSES AND THE MERIDIONAL AND CIRCUMFERENTIAL
3393  C***  MOMENTS OF THE SHELL MID-SURFACE, ARE PLOTTED IN A
3394  C***  2-DIMENSIONAL GRAPH, VERSUS THE MERIDIONAL COORDINATE, XI
3395  C***  IN THIS CASE, THE SHEAR STRESS AND THE TWIST ARE ZERO, EVERYWHERE.
3396  C***  WHEN THE GENERALIZED MODE SHAPES ARE NOT CIRCUMFERENTIALLY
3397  C***  UNIFORM WITHIN A SECTOR, BUT ARE THE SAME FROM SECTOR TO SECTOR,
3398  C***  CONTOURS OF THE 4 GENERALIZED STRESSES ARE PLOTTED FOR A
3399  C***  TYPICAL SECTOR.
3400  C***  WHEN THE GENERALIZED STRESSES VARY CIRCUMFERENTIALLY WITHIN
3401  C***  A SECTOR, AS WELL AS FORM SECTOR TO SECTOR, CONTOURS OF
3402  C***  THE 6 GENERALIZED MODE SHAPES ARE PLOTTED FOR THE SECTORS FOR
3403  C***  FOR WHICH SOLUTION HAS BEEN REQUESTED.
3404  C
3405  C     S T R I N G E R S  &  I N T E R M E D I A T E  S T R I N G E R S
3406  C
3407  C***  WHEN THE GENERALIZED STRESSES DO NOT VARY FROM SECTOR TO SECTOR,
3408  C***  SIGMA-XI AND M-ETA THE MERIDIONAL STRESS AND MOMENT,
3409  C***  RESPECTIVELY, OF THE THE STRINGERS AND INTERMEDIATE STRINGERS
3410  C***  ARE PLOTTED VERSUS THIER MERIDIONAL COORDINATE XI, IN A
3411  C***  2-DIMENSIONAL GRAPH, FOR A TYPICAL SECTOR.
3412  C***  IF HOWEVER, THE MODE SHAPES DO VARY WITHIN THE SECTOR AND
3413  C***  FROM SECTOR TO SECTOR 2-DIMENSIONAL GRAPHS OF SIGMA-XI, M-XI,
3414  C***  M-ZETA, AND T THE MERIDIONAL STRESS AND THE IN-PLANE AND
3415  C***  OUT-OF-PLANE MOMENTS AND TWIST, RESPECTIVELY ARE PLOTTED FOR ALL
3416  C***  SECTORS FOR WHICH SOLUTION HAS BEEN REQUESTED.
```

```
C
C     R I N G S  &  I N T E R M E D I A T E   R I N G S.
C
C***  WHEN THE GENERALIZED STRESSES DO NOT VARY CIRCUMFERENTIALLY,
C***  NO PLOT IS PRODUCED FOR RINGS AND INTERMEDIATE RINGS.
C***  WHEN THE GENERALIZED STRESSES  WITHIN A SECTOR DO VARY
C***  CIRCUMFERENTIALLY, BUT THE GENERALIZED STRESSES OF ALL SECTORS
C***  ARE THE SAME, GRAPHS OF SIGMA-ETA AND M-XI, THE AXIAL STRESS
C***  AND THE IN-PLANE MOMENT, RESPECTIVELY, ARE PLOTTED VERSUS
C***  ETA, THE CIRCUMFERENTIAL COORDINATE.
C***  WHEN THE GENERALIZED STRESSES ARE NOT UNIFORM WITHIN A SECTOR
C***  AND VARY FORM SECTOR TO SECTOR, 2-DIMENSIONAL PLOTS OF EACH OF
C***  THE FOUR GENERALIZED STRESSES ARE PRODUCED FOR EACH RING AND
C***  INTERMEDIATE RING, FOR ALL SECTORS FOR WHICH SOLUTION HAS BEEN
C***  REQUESTED.
C
C
C        THE 'SCIENTIFIC GRAPHICS SYSTEM', SDG, IS UTILIZED
C        HERE TO PLOT THE CURVES AND CONTOURS. THIS SOFTWARE
C        HAS BEEN DEVELOPED AT 'NCAR', 'NATIONAL CENTER FOR
C        ATMOSPHERIC RESEARCH', IN BOULDER COLORADO.
C
C
      CALL PLTSTS (DSKF1, ILOCA, LCNLD, LINSL, MGAUS, MNDSH,
     + MELSH, MELSR, MIMSR, MELISR, MELRN, MIMRN, MELIRN, MRWND,
     + MLOCA, MINSL, NSECT, NGAUS, NNDSH, NELSH, NSTSSH, NSTRN, NELSR,
     + NSTSSR, NIMSR, NELISR, NSTSIS, NRING, NELRN,
     + NSTSRN, NIMRN, NELIRN, NSTSIR, NRWEL, NACNT, STRTYP, SYMLD,
     + CRDCL, CRDMC, NODSH, NODSR, NODISR, NDIMSR, NODRN, NRNGEL,
     + NODIRN, NDIMRN, SLVSC, NDEX1, NDEX2, NDJNC, ROWSH, GUSTR, GUSCL,
     + X, BRHS, KSECT, STRES)
C
C
C
C*    IF PLOTTING OF THE MODE SHAPES HAS NOT BEEN REQUESTED, OR THE
C*    CURRENT SOLUTION IS DUE TO DEAD LOAD, SKIP.
C
  550 IF (PLMSP . NE . 'YES' . OR . SLOLD . EQ . 3HYES)   GO TO 650
C
      SYMLD = . FALSE .
C
C***  PLOT THE DEFORMED SHAPE OF THE STRUCTURE (IN SOLID LINES)
C***  ALONG WITH THE UNDEFORMED ORIGINAL SHAPE (IN DASH LINES).
C***  NOTE: THIS ROUTINE ONLY PLOTS THE DEFORMED SHAPES OF
C***  THE STIFFENERS AND DOES NOT CONSIDER THE PRESENCE OF THE SHELL.
C
C
C                          M O D E   S H A P E S
C
C
C
```

```
C
C
C
C***  PLOT THE NODAL IN-PLANE MODE SHAPES OF THE SHELL, AND THE
C***  NODAL MODE SHAPES OF STRINGERS AND INTERMEDIATE STRINGERS
C***  RINGS AND INTERMEDIATE RINGS FOR THE CURRENT LOADING CASE.
C
C     S H E L L.
C
C***  WHEN THE MODE SHAPES ARE UNIFORM CIRCUMFERENTIALLY WITHIN A
C***  SECTOR, ONLY THE MERIDIONAL (U) AND THE NORMAL (W), NODAL
C***  MODE SHAPES OF THE SHELL MID-SURFACE, ARE PLOTTED IN A SIMPLE
C***  2-DIMENSIONAL GRAPH, VERSUS THE MERIDIONAL COORDINATE.
C***  IN THIS CASE, V = 0, EVERYWHERE.
C***  WHEN THE MODE SHAPES ARE NOT CIRCUMFERENTIALLY UNIFORM
C***  WITHIN A SECTOR, BUT ARE THE SAME FROM SECTOR TO SECTOR, CONTOURS
C***  OF MODE SHAPES U, V, AND W ARE PLOTTED FOR A TYPICAL SECTOR.
C***  WHEN MODE SHAPES VARY CIRCUMFERENTIALLY WITHIN THE SECTOR
C***  AS WELL AS FORM SECTOR TO SECTOR, CONTOURS OF MODE SHAPES
C***  ARE PLOTTED FOR THE SECTORS FOR WHICH SOLUTION HAS BEEN REQUESTED.
C
C     S T R I N G E R S  &  I N T E R M E D I A T E  S T R I N G E R S.
C
C***  WHEN THE MODE SHAPES DO NOT VARY FROM SECTOR TO SECTOR,
C***  U AND W, THE MERIDIONAL AND THE NORMAL MODE SHAPES,
C***  RESPECTIVELY, OF THE THE STRINGERS AND INTERMEDIATE STRINGERS
C***  ARE PLOTTED VERSUS THIER MERIDIONAL COORDINATE XI, IN A
C***  2-DIMENSIONAL GRAPH, FOR A TYPICAL SECTOR.
C***  IF HOWEVER, THE MODE SHAPES DO NOT VARY FROM SECTOR TO SECTOR,
C***  2-DIMENSIONAL GRAPHS OF U, V, AND W, THE MERIDIONAL, THE
C***  CIRCUMFERENTIAL AND THE NORMAL MODE SHAPES, RESPECTIVELY,
C***  ARE PLOTTED VERSUS THE MERIDIONAL COORDINATE, FOR ALL SECTORS
C***  FOR WHICH SOLUTION HAS BEEN REQUESTED.
C
C     R I N G S  &  I N T E R M E D I A T E  R I N G S.
C
C***  WHEN THE MODE SHAPES DO NOT VARY CIRCUMFERENTIALLY, NO PLOT
C***  IS PRODUCED FOR RINGS AND INTERMEDIATE RINGS.
C***  WHEN THE MODE SHAPES WITHIN A SECTOR DO VARY CIRCUMFERENTIALLY,
C***  BUT THE MODE SHAPES OF ALL SECTORS ARE THE SAME, 2-DIMENSIONAL
C***  GRAPHS OF V AND W, THE CIRCUMFERENTIAL AND NORMAL MODE SHAPES,
C***  RESPECTIVELY, ARE PLOTTED VERSUS ETA, THE CIRCUMFERENTIAL
C***  COORDINATE, FOR ONLY A TYPICAL SECTOR.
C***  WHEN THE MODE SHAPES VARY CIRCUMFERENTIALLY WITHIN A SECTOR
C***  AS WELL AS FROM SECTOR TO SECTOR, CONTOURS OF NODAL MODE SHAPES
C***  U, V, AND W (MERIDIONAL, CIRCUMFERENTIAL, AND NORMAL,
C***  RESPECTIVELY) ARE PLOTTED FOR ALL SECTORS FOR WHICH SOLUTION HAS
C***  BEEN REQUESTED.
C
C
C        THE 'SCIENTIFIC GRAPHICS SYSTEM', SDG, IS UTILIZED
```

```
C         HERE TO PLOT THE CURVES AND CONTOURS. THIS SOFTWARE
C         HAS BEEN DEVELOPED AT 'NCAR', 'NATIONAL CENTER FOR
C         ATMOSPHERIC RESEARCH', IN BOULDER COLORADO.
C
C
      CALL PLTDIS (DSKF1, ILOCA, LCNLO, LZNSL, MNDSH, MELSH,
     + MELSR, MIMSR, MELISR, MELRN, MIMRN, MELIRN, MRWND, MLOCA,
     + MZNSL, NSECT, NNDSH, NELSH, NMSND, NMSCT, NSTRN, NELSR, NTFSR,
     + NIMSR, NELISR, NTFISR, NRING, NELRN, NTFRN, NIMRN, NELIRN,
     + NTFIRN, NRWND, NRWEL, NDFSC, STRTYP, SLRAC, SYMLO, NODSH,
     + CRDCL, CRDMC, NODSR, NODISR, NDIMSR, NODRN, NRNGEL, NODIRN,
     + NDIMRN, SLVSC, NDEX1, NDEX2, NOORD, ROWSH, NDJNC, TRFST,
     + CORLD, KSECT)
C
C
C
C
C***  CALLING ' F L U S H ' FORCES ' N C A R ' TO COMPLETE THE LAST
C***  PLOT.
C
  650 CALL FLUSH
      RETURN
      END
C
```

The Preprocessor Subprogram

```
      C     ---------------------------------------
      C
      C
      C
      C***  THIS SUBROUTINE CONTAINS THE MAXIMUM OF THE DIMENSIONS OF
      C***  VARIABLES.
      C
            SUBROUTINE MAXIMS (JNDSH, JSECT, JRWND, JELSH, JZNSH, JMTSH,
     +  JSTRN, JNDSR, JELSR, JMTSR, JIMSR, JELISR, JMTISR,
     +  JRING, JNDRN, JELRN, JMTRN, JIMRN, JELIRN, JMTIRN,
     +  JGAUS, JCRSM, JPROS, JLOCA, JCNLD, JDTLD, JAULD,
     +  JZNTP, JZNSL, JICON, MNDSH, MELSH, MZNSH, MMTSH, MSTRN, MNDSR,
     +  MELSR, MMTSR, MIMSR, MELISR, MMTISR, MRING, MNDRN, MELRN, MMTRN,
     +  MIMRN, MELIRN, MMTIRN, MGAUS, MGUSS, MZNTP, MZNSL, MICON, MSECT,
     +  MRWND, MRWEL, MCRSM, MPROS, MLOCA, MCNLD, MDTLD, MAULD, MDFSC,
     +  MDFIF)
      C
            MNDSH = JNDSH
            MDFSC = 12 * MNDSH
            MSECT = JSECT
            MRWND = JRWND
            MRWEL = MRWND - 1
            MELSH = JELSH
            MZNSH = JZNSH
            MMTSH = JMTSH
            MSTRN = JSTRN
            MNDSR = JNDSR
            MELSR = JELSR
            MMTSR = JMTSR
            MIMSR = JIMSR
            MELISR = JELISR
            MMTISR = JMTISR
            MRING = JRING
            MNDRN = JNDRN
            MELRN = JELRN
            MMTRN = JMTRN
            MIMRN = JIMRN
            MELIRN = JELIRN
            MMTIRN = JMTIRN
            MGAUS = JGAUS
            MGUSS = MGAUS
            MCRSM = JCRSM
            MPROS = JPROS
            MLOCA = JLOCA
            MCNLD = JCNLD
            MDTLD = JDTLD
            MAULD = JAULD
            MZNTP = JZNTP
            MZNSL = JZNSL
```

```
      MICON = JICON
      NOFND = 12
      MOFIF = NOFND * MRWND
C
      RETURN
      END
C
C
C
C     ------------------------------------------------------------
C
C
C
C***  THIS SUBROUTINE DEFINES THE VALUES REPRESENTING THE DISK FILE
C***  UNIT NUMBERS.
C
      SUBROUTINE DSKFIL (DSKF1, DSKF2, IN, IO)
C
      INTEGER  DSKF1, DSKF2
C
      DSKF1 = 1
      DSKF2 = 2
      IN = 5
      IO = 6
C
      RETURN
      END
C
C
C
C***  THIS SUBROUTINE READS AND WRITES AND PLOTS AND CHECKS
C***  THE INPUT DATA.
C
       SUBROUTINE DESCR1(MNDSH, MELSH, MRWND, MRWEL, MMTSH, MELSR,
     + MMTSR, MIMSR, MELISR, MMTISR, MELRN, MMTRN, MIMRN, MELIRN,
     + MMTIRN, MCRSM, MPRDS, MLOCA, MCNLO, MDTLO, MZNTP, NODSH, MTDSH,
     + DNSSH, PRPSH, MTSHL, SHMTI, SHMTJ, TSHEL, TRMSH,
     + ROWSH, ORIGN, NODSR, MTDSR, ECNSR, TRMSR, PRPSR, MTSTR, NODISR,
     + NDIMSR, MTDISR, ECISR, TMISR, PRPISR, MTISR, NODRN, MTDRN, PSNRN,
     + TRMRN, PRPRN, MTRNG, NRNGEL, NODIRN, NDIMRN, MTDIRN, PSNIRN,
     + TMIRN, PRPIRN, MTIRN, NIRNEL, CRDCL, CRDMC, ALPHA)
C
       INTEGER  STRTYP, AUTGN, SLOSP, SLRAC, SLSTS,
     + SLBKL, PLSTR, PLMSH, PLOSP, PLSTS, PLMSP,
     + ROWSH(MRWND,2), SHMTI(1), SHMTJ(1), ERDR2(20)
C
       REAL ORIGN(MRWEL,1), AMTRX(5,5), COEFF(5)
C
```

```
      DIMENSION NODSH(MELSH,1), MTDSH(1), MTSHL(1), TSHEL(1),
     + TRMSH(1), DNSSH(1), PRPSH(MMTSH,1),
     + NODSR(MELSR,1), MTDSR(1), ECNSR(1), TRMSR(1), PRPSR(MMTSR,1),
     + MTSTR(1), NODISR(MELISR,1), NDIMSR(MIMSR,1), MTDISR(1)
     + ,ECISR(1), TMISR(1), PRPISR(MMTISR,1), MTISR(1), NODRN(MELRN,1),
     + MTDRN(1), PSNRN(MMTRN,1), TRMRN(1), PRPRN(MMTRN,1), MTRNG(1),
     + NRNGEL(1),
     + NODIRN(MELIRN,1), NDIMRN(MIMRN,1), MTDIRN(1), PSNIRN(MMTIRN,1),
     + TMIRN(1), PRPIRN(MMTIRN,1), MTIRN(1), NIRNEL(1),
     + CRDCL(MNDSH,1), CRDMC(MNDSH,1), ALPHA(3)
C
      COMMON  STRTYP, AUTGN, SLDSP, SLRAC, SLSTS, SLBKL,
     + PLSTR, PLMSH, PLDSP, PLSTS, PLMSP,
     + MSECT, MSTRN, MNDSR, MRING, MNDRN, MGAUS,
     + NNDSH, NELSH, NMTSH, NSTRN, NNDSR, NELSR, NMTSR, NIMSR,
     + NELISR, NMTISR, NRING, NNDRN, NELRN, NMTRN, NIMRN, NELIRN,
     + NMTIRN, NBRAC, NELBR, NMTBR, NGAUS, NCRSM, NPRDS, NLDCA,
     + NSECT, NRWEL, NRWND, NDFND, NNDE1, NNDE2, NNDIN,
     + NNTSH, NBTSH, NGTSH, NSTSSH, NKCTND, NNTSR, NBTSR, NGTSR,
     + NTFSR, NSTSSR, NNTISR, NBTISR, NGTISR, NTFISR, NSTSIS, NNTRN,
     + NBTRN, NGTRN, NTFRN, NSTSRN, NNTIRN, NBTIRN, NGTIRN, NTFIRN,
     + NSTSIR, NTFER, NSTSBR, NKRWPD, NCRLD, NFRCN, NABIF, NASCT,
     + NACNT, NFBSC, NRACT, NKSCT, NKSCF, NKKGW, NKGSC, NKGICL, NKBNIF,
     + NKGBNI, NEGVC, NKKGCN, NKGCN, NMSND, NMSCT, NKOEE, NKGOEE,
     + CSDCL, CSDMC, MCCOD, ILDCA, LCNLD, LDTLD, LZNTP, LZNSL,
     + DSKF1, DSKF2, IN, IO,
     + FEED, RTOP, RBAS, HITE, ELEV, THATAD,
     + TITLE(30), LDCAS(2), NCNLD(2), NDTLD(2), NZNTP(2), NZNSL(2),
     + RATIO(14), INDEX(45), NDCON(200), NDEX1(15), NDEX2(15),
     + NDINT(15), INTND(35), WKAREA(192)
C
      LSHLJ = 0
C
C***  INITIALIZE ERROR2 TO ZERO.
C
      CALL SETZRO (20, 1, ERROR2)
C
C***  READ AND WRITE THE CONTROL DATA.
C
      CALL CONTRL (IN, IO, STRTYP, AUTGN, SLDSP, SLRAC,
     + SLSTS, SLBKL, PLSTR, PLMSH, PLDSP, PLSTS, PLMSP,
     + NDFND, NDFEX, NNDSH, NELSH, NMTSH, NNDLR, NSTRN, NNDSR,
     + NELSR, NMTSR, NIMSR, NELISR, NMTISR, NRING, NNDRN, NELRN,
     + NMTRN, NIMRN, NELIRN, NMTIRN, NBRAC, NELBR, NMTBR, NGAUS,
     + NCRSM, NPRDS, NLDCA, NSECT, NRWEL, NRWND, TITLE, LDCAS, NCNLD,
     + NDTLD, NZNTP, NZNSL)
C
C***  CHECK THE CONTROL DATA AND PRINT ERROR MESSAGES IF ANY :
C
      CALL CHECK1 (KERR1, MNDSH, MELSH, MMTSH, MELSR, MMTSR,
     + MIMSR, MELISR, MMTISR, MELRN, MMTRN, MIMRN, MELIRN, MMTIRN,
```

```
      + MRWND, MRWEL, MCRSM, MPROS, MLOCA, MCNLD, MDTLD, MZNTP)
C
      IF (KERR1 . EQ . 0)                                  GO TO 60
C
C***  IF AN ERROR HAS BEEN DETECTED IN CHECK1 CALL ECHO TO ECHO THE
C***  REMAINDER OF THE DATA. NOTE THAT EXECUTION WILL BE ABORTED BY
C***  VAX DUE TO ERROR NUMBER 65.
C
      CALL ECHO (IN, IO)
C
C
   60 IF ( STRTYP . EQ . 7HGENERAL )                       GO TO 70
C
C***  READ AND WRITE DATA ON ROWS OF ELEMENTS:
C
      CALL ROWSHL (IN, IO, MRWND, MRWEL, NNDSH, NRWEL, NNDLR,
     + AUTGN, TITLE, NDEX1, ROWSH, ORIGN, RATIO, EROR2)
C
C
   70 CALL GEODTA (IN, IO, AUTGN, NSECT, NRWND, NRWEL, TANFEE,
     + SINFEE, FEED, THATAD, THATAR, ELEV, RTOP, RBAS, HITE, TITLE)
C
      IF (NELSH . EQ . 0)                                  GO TO 75
C
C
C***  READ AND WRITE MATERIAL PROPERTIES :
C
      CALL MTPRSH (IN,IO,MMTSH,NMTSH,EROR2,TITLE,TSHEL,TRMSH,
     +DNSSH,PRPSH)
C
C***  READ MATERIAL SPECIFICATIONS FOR  ELEMENTS :
C
      CALL MTSPSH (IN,LSHLJ,LZNSH,NELSH,TITLE,MTOSH,SHMTI,SHMTJ,ERORZ)
C
C***  GENERATE OR READ NODAL INCIDENCES FOR EACH   ELEMENT .
C***  THEN PRINT THE NODAL INCIDENCES FOR EACH   ELEMENT
C***  ALONG WITH ITS MATERIAL DESIGNATION :
C
      CALL NDINSH (AUTGN, IN, IO, LZNSH, MRWEL, MELSH,
     + NRWND, NRWEL, TITLE, NDOSH, MTOSH, SHMTI, SHMTJ, MTSHL, ROWSH)
C
C***  FIND AND PRINT ALL OF THE NODES CONNECTED TO A CERTAIN NODE.
C
      CALL NDCNSH (IO, MELSH, NNOSH, NELSH, NODSH, INDEX, NDCON)
C
C***  GENERATE THE CYLINDRICAL AND MERIDIONAL-CIRCUMFERENTIAL
C***  COORDINATES OF ALL NODES AUTOMATICALLY, OR READ THEM DIRECTLY
C***  THROUGH THE NONAUTOMATIC VERSION. THEN PRINT THESE COORDINATES.
C
C
   75 CALL CORDSH (IN, IO, PLSTR, STRTYP, AUTGN, MNDSH, MRWND,
```

```
      + MRWEL, NRWND, NRWEL, NNDSH, NNDE1, RTOP, RBAS, HITE, ELEV,
      + THATAD, TITLE, ROWSH, ORIGN, RATIO, NDEX1, NDEX2, WKAREA, AMTRX,
      + COEFF, CRDCL, CRDMC, ALPHA)
C
C***  THIS SUBROUTINE IDENTIFIES EXTERNAL AND INTERNAL NODES OF A
C***  TYPICAL SECTOR AND PRINTS THEM.
C
       CALL EXINND (IO,MNDSH,NNDSH,NNDE1,NNDE2,NNDIN,CRDCL,NDEX1,NDEX2,
      + NDINT, INTND)
C
C*    IF NO STRINGERS AND NO INTERM. STRINGERS EXIST SKIP :
       IF(NSTRN.EQ.0.AND. NIMSR.EQ.0) GO TO 200
C
C***  READ MATERIAL PROPERTIES FOR STRINGERS AND INTERMEDIATE
C***  STRINGERS AND PRINT THEM ONLY FOR STRINGERS :
C
       CALL MTPRSR (IN, IO, MMTSR, MMTISR, NSTRN, NMTSR, NIMSR, NMTISR,
      + ERDR2, TITLE, ECNSR, TRMSR, PRPSR, MTDSR, ECISR, TMISR, PRPISR,
      + MTDISR)
C
C*    IF NO STRINGERS EXIST SKIP :
       IF(NSTRN.EQ.0) GO TO 100
C
C***  GENERATE OR READ NODAL INCIDENCES ,GEOMETRIC DATA AND MATERIAL
C***  SPECIFICATION FOR STRINGER ELEMENTS AND THEN PRINTS NODAL
C***  INCIDENCES ALONG WITH MATERIAL DESIGNATION FOR EACH ELEMENT :
C
       CALL NDINSR (IN, IO, STRTYP, AUTGN, LELSR, MELSR,
      + NELSR, NMTSR, NRWND, NRWEL, TITLE, NDEX1, NDEX2, NODSR,
      + MTDSR, MTSTR)
C
C***  PRINT NODAL COORDINATES FOR STRINGERS :
C
       CALL PTCRSR (IN,IO,MNDSH,MELSR,NELSR,CRDCL,CRDMC,NODSR)
C
C*    IF NO INTERM. STRINGERS EXIST SKIP :
  100 IF(NIMSR.EQ.0) GO TO 200
C
C***  WRITE MATERIAL SPECIFICATIONS FOR INTERMEDIATE STRINGERS :
C
       CALL PPRISR (IO, MMTISR, NMTISR, MTDISR, ECISR, TMISR,
      + PRPISR, MTISR)
C
C***  GENERATE OR READ NODAL INCIDENCES AND MATERIAL SPECIFICATIONS
C***  FOR INTERM. STRINGERS AND THEN PRINT THEN FOR EACH ELEMENT :
C
       CALL NDIISR (IN, IO, STRTYP, AUTGN, LELISR, MNDSH,
      + MIMSR, MELISR, NELSR, NIMSR, NELISR, NMTISR, NRWND, TITLE, CRDCL,
      + NDEX1, NDEX2, INTND, MTISR, MTDISR, NDIMSR, NODISR)
C
C***  PRINT NODAL COORDINATES FOR INTERM. STRINGERS :
```

```
C
      CALL PCRISR (IN,IO,MELISR,MNDSH,NIMSR,NELISR,NODISR,CRDCL,CRDMC)
C
C*    IF NO RINGS AND NO INTERM. RINGS EXIST :
  200 IF(NRING . EQ . 0 . AND . NIMRN . EQ . 0)   GO TO 400
C
C***  READ AND WRITE MATERIAL PROPERTIES FOR RINGS :
C
      CALL MTPRRN (IN, IO, MMTRN, MMTIRN, NRING, NMTRN, NIMRN,
     + NMTIRN, EROR2, TITLE, MTDRN, PSNRN, TRMRN, PRPRN, MTDIRN, PSNIRN,
     + TMIRN, PRPIRN)
C
C*    IF NO RINGS EXIST SKIP :
      IF(NRING.EQ.0) GO TO 300
C
C***  GENERATE OR READ NODAL INCIDENCES, GEOMETRIC DATA AND MATERIAL
C***  SPECIFICATIONS FOR DOMAIN AND PRINT THE FOR EACH ELEMENT :
C
C
      CALL NDINRN (IN, IO, STRTYP, AUTGN, LELRN, MELRN ,
     + NRING, NELRN, NMTRN, NRWND, NRWEL, TITLE, NDEX1, NDEX2, NODRN,
     + MTRNG, MTDRN, NRNGEL)
C
C***  PRINT NODAL COORDINATES FOR DOMAIN :
C
      CALL PTCRRN (IN, IO, STRTYP, AUTGN, MELRN, MNDSH, NRING, NELRN,
     + NRNGEL, NODRN, CRDCL, CRDMC)
C
C*    IF NO INTERMEDIATE ELEMENT SKIP :
C
      IF (NIMRN . EQ . 0)                              GO TO 400
C
C***  PRINT MATERIAL PROPERTIES FOR INTERMEDIATE ELEMENT :
C
  300 CALL PPRIRN (IO, MMTIRN, NIMRN, NMTIRN, MTDIRN, PSNIRN, TMIRN,
     + PRPIRN)
C
C***  GENERATE OR READ GEOMETRIC DATA, NODAL INCIDENCES AND MATERIAL
C***  SPECIFICATIONS FOR INTERM.
C
      CALL NDIIRN (IN ,IO ,STRTYP ,AUTGN ,LELIRN ,MIMRN ,MELIRN ,
     + NELRN, NELIRN, NIMRN, NMTIRN, TITLE, NDIMRN, NODIRN, NIRNEL,
     + MTIRN, MTDIRN)
C
C***  PRINT NODAL COORDINATES FOR INTERM.
C
      CALL PCRIRN (IN, IO, AUTGN, MNDSH, MELIRN, MIMRN, NIMRN, NDIMRN,
     + NIRNEL, NODIRN, CRDCL, CRDMC)
C
  400 IF (NBRAC . EQ . 0)                              GO TO 425
C
```

```
C***  READ AND WRITE MATERIAL PROPERTIES FOR BRACES.
C
      CALL MTPRBR (IN, IO, MMTIRN, NMTIRN, NMTBR, TITLE,
     + MTDIRN, PSNIRN, TMIRN, PRPIRN)
C
C***  GENERATE OR READ GEOMETRIC DATA, NODAL INCIDENCES AND MATERIAL
C***  SPECIFICATIONS FOR BRACES AND PRINT THEM ELEMENT BY ELEMENT.
C
      CALL NDINBR (AUTGN, IN, IO, LELBR, MIMRN, MELIRN,
     + NELIRN, NIMRN, NMTIRN, NBRAC, NELBR, NMTBR, TITLE, NDIMRN,
     + NODIRN, NIRNEL, MTIRN, MTDIRN)
C
  425 IF (NELSH . NE . 0)                          GO TO 430
C
C***  GENERATE VECTORS INDEX AND NDCON, WHICH TOGETHER GIVE THE NODES
C***  CONNECTED TO A GIVEN NODE. NOTE THAT THIS SUBROUTINE IS USED
C***  IN LIEU OF SUBROUTINE NDCNSH, WHEN NO   ELEMENTS EXIST.
C
      CALL NDCNST (IO, MELSR, MELISR, MELRN, MELIRN, NNDSH,
     + NELSR, NELISR, NELRN, NELIRN, NELBR, NRWND, NODSR, NODISR,
     + NODRN, NODIRN, NDEX1, NDEX2, INTND, INDEX, NDCON)
C
C***  CHECK DATA GENERATED OR READ VERSUS KNOWN DATA :
C
  430 CALL CHECK2 (IO,KERR2,AUTGN,LSHLJ,LELSR,LELISR,LELRN,LELIRN,
     + MNDSH,MELSH,NNDSH,NELSH,NMTSH,NSTRN,NELSR,NMTSR,NIMSR,NELISR,
     + NMTISR,NRING,NELRN,NMTRN,NIMRN,NELIRN,NMTIRN,NRWEL,ERDR2,
     + CRDCL,NODSH)
C
      IF (KERR2 . EQ . 0)                          GO TO 450
C
C***  IF AN ERROR HAS BEEN DETECTED BY CHECK2 THEN CALL ECHO TO
C***  READ AND WRITE THE REMAINDER OF THE INPUT DATA. NOTE EXECUTION
C***  WILL BE ABROTED BY VAX DUE TO ERROR NUMBER 65.
C
      CALL ECHO(IN,IO)
C
C
C
C
  450 RETURN
      END
C
C
C
C     ------------------------------------------
C
C
C
C***  THIS SUBROUTINE READS AND WRITES THE CONTROL DATA :
C
```

```
      SUBROUTINE CONTRL (IN, IO, STRTYP, AUTGN, SLDSP, SLRAC,
     + SLSTS, SLBKL, PLSTR, PLMSH, PLDSP, PLSTS, PLMSP,
     + NDFND, NDFEX, NNDSH, NELSH, NMTSH, NNDLR, NSTRN, NNDSR,
     + NELSR, NMTSR, NIMSR, NELISR, NMTISR, NRING, NNDRN, NELRN,
     + NMTRN, NIMRN, NELIRN, NMTIRN, NBRAC, NELBR, NMTBR, NGAUS,
     + NCRSM, NPRDS, NLOCA, NSECT, NRWEL, NRWND, TITLE, LDCAS, NCNLD,
     + NDTLD, NZNTP, NZNSL)
C
      CHARACTER*10,  CSTRTP, CAUTGN, CSLDSP, CSLRAC, CSLSTS,
     + CSLBKL, CPLSTR, CPLMSH, CPLDSP, CPLSTS, CPLMSP
C
      INTEGER  STRTYP, AUTGN, SLDSP, SLRAC, SLSTS, SLBKL, PLSTR,
     + PLMSH, PLDSP, PLSTS, PLMSP
C
      DIMENSION  TITLE(8), LDCAS(1), NCNLD(1), NDTLD(1), NZNTP(1),
     + NZNSL(1)
C
      READ(IN,100) TITLE
C
      WRITE(IO,10)
   10 FORMAT(1H1,////////////////////// T62, 'O  U  T  P  U  T'/////)
C
      CALL TIME(T)
      CALL DATE(D)
C
      WRITE(IO,20) T, D
   20 FORMAT (1H0, T110, 'TIME :', 3X, 1A10/ T110, 'KKKK' //
     + 1H0, T110, 'DATE :', 3X, 1A10/ T110, 'KKKK'////)
C
      WRITE(IO,30) TITLE
   30 FORMAT (1H0, ///// 6X, 8A10 ///)
C
      READ (IN,90) (TITLE(I),I=1,16)
C
   90 FORMAT(8A10/8A10)
  100 FORMAT(8A10)
C
C*    READ CONTROL DATA
C
      READ*, CSTRTP, CAUTGN, CSLDSP, CSLRAC, CSLSTS, CSLBKL
      READ(IN,100) TITLE
      READ*,  CPLSTR, CPLMSH, CPLDSP, CPLSTS, CPLMSP
      READ(IN,100) TITLE
      READ*, NSECT, NNDSH, NNDLR, NRWND
      READ(IN,100) TITLE
      READ*, NELSH, NMTSH, NGAUS
      READ(IN,100) TITLE
      READ*, NSTRN, NNDSR, NELSR, NMTSR
      READ(IN,100) TITLE
      READ*, NIMSR, NELISR, NMTISR
      READ(IN,100) TITLE
```

```
      READ*, NRING, NNDRN, NELRN, NMTRN
      READ (IN,100) TITLE
      READ*, NIMRN, NELIRN, NMTIRN
      READ (IN,100) TITLE
      READ*, NBRAC, NELBR, NMTBR
      READ(IN,100) TITLE
      READ*, NNCRSM, NPROS, NLDCA
C
      IF ( NLDCA . EQ . 0 )                          GO TO 107
C
      READ(IN,100) TITLE
C
      DO 105 ILDCA = 1 , NLDCA
C
      READ*, LDCAS(ILDCA), NCNLD(ILDCA), NDTLD(ILDCA), NZNTP(ILDCA),
     + NZNSL(ILDCA)
C
  105 CONTINUE
C
C***  CONVERT THE ALPHA-NUMERIC VARIABLES FROM THE CHARACTER FROM TO
C***  THE EQUIVALENT HOLLERITH FROM USING THE LIBRARY FUNCTION BOOL,
C***  WHERE,
C                          XXXXX = BOOL(CXXXXX)
C***  NOTE FUNCTION BOOL IS A FEATURE OF THE VAX ONLY.
C
  107 CALL CHRHOL (CSTRTP, CAUTGN, CSLDSP, CSLRAC, CSLSTS,
     + CSLBKL, CPLSTR, CPLMSH, CPLDSP, CPLSTS, CPLMSP, STRTYP, AUTGN,
     + SLDSP, SLRAC, SLSTS, SLBKL, PLSTR, PLMSH, PLDSP, PLSTS, PLMSP)
C
C*    PRINT CONTROL DATA
C
      WRITE(IO,120)  STRTYP, AUTGN, SLDSP, SLRAC, SLSTS, SLBKL
  120 FORMAT(1H0,5X,'CONTROL DATA '/1H ,5X,'KKKKKKKKKKKK'////
     + 1H0, 5X, 'STRUCTURE', T27, 'AUTOMATIC', T48, 'DISPLACEMENTS',
     + T69, 'REACTIONS', T90, 'STRESSES', T111, 'BUCKLING'/
     + 1H0, T8, A10, T30, A10, T53, A10, T72, A10, T93, A10, T114,
     + A10///)
C
      WRITE(IO,125)  PLSTR, PLMSH, PLDSP, PLSTS, PLMSP
  125 FORMAT (1H0, 5X, 'PLOT STRUCTURE', T31, 'PLOT MESH', T56,
     + 'PLOT DISPLACEMENTS', T81, 'PLOT STRESSES', T106,
     + 'PLOT MODE SHAPES'/
     + 1H0, T13, A10, T35, A10, T65, A10, T87, A10, T114, A10////)
C
      WRITE(IO,130)  NSECT, NNDSH, NNDLR, NRWND, NELSH, NMTSH, NGAUS,
     + NSTRN, NNDSR, NELSR, NMTSR, NIMSR, NELISR, NMTISR, NRING, NNDRN,
     + NELRN, NMTRN, NIMRN, NELIRN, NMTIRN, NBRAC, NELBR, NMTBR, NNCRSM,
     + NPROS, NLDCA
C
  130 FORMAT (1H0, T7, 'SECTORS', T38, 'NODES', T69, 'NO. NODES LAST ',
     +  'OF ROW', T100, 'ROWS OF NODES' ,
```

```
459         +  /1H0,  T7, I4, T38, I4, T76, I4, T103, I4////
460         + 1H0, T7, '  ELEMENTS', T38, 'MATERIALS', T69,
461         + 'GAUSS POINTS'/1H0,  T11, I4, T39, I4, T72, I4 ////
462         + 1H0, T7, 'STRINGERS', T38, 'NODES', T69, 'ELEMENTS', T100,
463         + 'MATERIALS'/1H0,  T9, I4, T38, I4, T70, I4, T101, I4 ////
464         + 1H0, T7, 'INTERMEDIATE STRINGERS', T38, 'ELEMENTS',  T69,
465         + 'MATERIALS'/1H0,  T15, I4, T39, I4, T70, I4 ////
466         + 1H0, T7, 'RINGS', T38, 'NODES', T69, 'ELEMENTS', T100,
467         + 'MATERIALS'/1H0,  T6, I4, T38, I4, T70, I4, T101, I4 ////
468         + 1H0, T7, 'INTERMEDIATE ELEMENTS', T38, 'ELEMENTS', T69,
469         + 'MATERIALS'/1H0,  T12, I4, T39, I4, T70, I4 ////
470         + 1H0, T7, 'BRACES', T38, 'ELEMENTS', T69,
471         + 'MATERIALS'/1H0,  T12, I4, T39, I4, T70, I4 ////
472         + 1H0, T7, 'NEW COORDINATES', T38, 'PRESCRIBED DISPLACEMENTS',
473         + T69, 'LOADING CASES'/1H0,  T11, I4, T47, I4, T72, I4/////)
474   C
475          IF ( NLDCA . EQ . 0 )                          GO TO 165
476   C
477          WRITE(IO,151)
478       151 FORMAT(1H0,////,6X,'LOADING CASE',T31,'CONCEN. NODAL LOADS',T56,
479         +'DISTRIBUTED LOADS',T81,'TEMPERATURE ZONES',T106,'SOLUTION ZONES')
480   C
481          DO 155 ILDCA = 1 , NLDCA
482   C
483          WRITE(IO, 152)  LDCAS(ILDCA), NCNLD(ILDCA), NDTLD(ILDCA),
484         + NZNTP(ILDCA), NZNSL(ILDCA)
485       152 FORMAT (1H0, 9X, I3, T39, I3, T62, I3, T87, I3, T111, I3/)
486   C
487       155 CONTINUE
488   C
489   C*    DETERMINE NCRSM, THE TOTAL NUMBER OF COORDINATE SYSTEMS WHICH IS
490   C*    EQUAL TO THE NUMBER OF NEW SYSTEMS PLUS THE CYLINDRICAL AND THE
491   C*    MERIDIONAL-CIRCUMFERENTIAL SYSTEMS.
492   C
493       165 NCRSM = NNCRSM + 2
494   C
495   C*    COMPUTE NRWEL, AND NDFEX, NUMBER OF ROWS OF ELEMENTS, AND NUMBER
496   C*    OF EXTERNAL DEGREES OF FREEDOM, RESPECTIVELY.
497   C
498          NDFND = 12
499          NRWEL = NRWND - 1
500          IF (NRWND . EQ . 1)                            NRWEL = 1
501          NDFEX = 2 * NDFND * NRWND
502          RETURN
503          END
504
505   C
506   C
507   C
508   C      --------------------    C  H  E  C  K  1    --------------------
509   C
```

```
      C
      C
C***  THIS SUBROUTINE CHECKS THE CONTROL DATA OF PHASE 1 AND
C***  PRINTS DIAGNOSTICS IF ANY :
      C
       SUBROUTINE CHECK1 (KERR1, MNDSH, MELSH, MMTSH, MELSR, MMTSR,
     + MIMSR, MELISR, MMTISR, MELRN, MMTRN, MIMRN, MELIRN, MMTIRN,
     + MRWND, MRWEL, MCRSM, MPRDS, MLDCA, MCNLD, MDTLD, MZNTP)
      C
       INTEGER  ERDR1(40), STRTYP, AUTGN, SLDSP, SLRAC, SLSTS, SLBKL,
     + PLSTR, PLMSH, PLDSP, PLSTS, PLMSP
      C
       COMMON  STRTYP, AUTGN, SLDSP, SLRAC, SLSTS, SLBKL,
     + PLSTR, PLMSH, PLDSP, PLSTS, PLMSP,
     + MSECT, MSTRN, MNDSR, MRING, MNDRN, MGAUS,
     + NNDSH, NELSH, NMTSH, NSTRN, NNDSR, NELSR, NMTSR, NIMSR,
     + NELISR, NMTISR, NRING, NNDRN, NELRN, NMTRN, NIMRN, NELIRN,
     + NMTIRN, NBRAC, NELBR, NMTBR, NGAUS, NCRSM, NPRDS, NLDCA,
     + NSECT, NRWEL, NRWND, NDFND, NNDE1, NNDE2, NNDIN,
     + NNTSH, NBTSH, NGTSH, NSTSSH, NKCTND, NNTSR, NBTSR, NGTSR,
     + NTFSR, NSTSSR, NNTISR, NBTISR, NGTISR, NTFISR, NSTSIS, NNTRN,
     + NBTRN, NGTRN, NTFRN, NSTSRN, NNTIRN, NBTIRN, NGTIRN, NTFIRN,
     + NSTSIR, NTFBR, NSTSBR, NKRWPD, NCRLD, NFRCN, NABIF, NASCT,
     + NACNT, NFBSC, NRACT, NKSCT, NKSCF, NKKGW, NKGSC, NKGICL, NKBNIF,
     + NKGBNI, NEGVC, NKKGCN, NKGCN, NMSND, NMSCT, NKOEE, NKGOEE,
     + CSDCL, CSDMC, MCCOD, ILDCA, LCNLD, LDTLD, LZNTP, LZNSL,
     + DSKF1, DSKF2, IN, IO,
     + FEED, RTOP, RBAS, HITE, ELEV, THATAD,
     + TITLE(30), LDCAS(2), NCNLD(2), NDTLD(2), NZNTP(2), NZNSL(2),
     + RATIO(14), INDEX(45), NDCON(200), NDEX1(15), NDEX2(15),
     + NDINT(15), INTND(35), WKAREA(192)
      C
       KERR1=0
      C
       DO 80 I = 1 , 36
    80 ERDR1(I) = 0
      C
C*    IF ANY ERRORS ARE DETECTED , SET UP FLAGS FOR THE CORRESPONDING
C*    ERROR MESSAGE :
      C
       IF ( STRTYP . NE . 4HCONE . AND . STRTYP . NE . 6HSPHERE . AND .
     + STRTYP . NE . 9HELLIPSOID . AND . STRTYP . NE . 10HHYPERBOLOI .
     + AND . STRTYP . NE . 10HPARABOLOID . AND . STRTYP . NE .
     + 8HCYLINDER . AND . STRTYP . NE . 6HHYBRID . AND . STRTYP . NE .
     + 7HGENERAL )              ERDR1(1)  =  1
      C
       IF(AUTGN.NE.3HYES.AND.AUTGN.NE.2HNO) ERDR1(2) = 1
      C
       IF (SLDSP . NE . 3HYES . AND . SLDSP . NE . 2HNO . AND .
     + SLRAC . NE . 3HYES . AND . SLRAC . NE . 2HNO . AND .
     + SLSTS . NE . 3HYES . AND . SLSTS . NE . 3HNO . AND .
```

```
      + SLBKL . NE . 3HYES . AND . SLBKL . NE . 2HNO )  ERDR1(3) = 1
C
       IF (PLSTR . NE . 3HYES . AND . PLSTR . NE . 2HNO . AND .
      + PLMSH . NE . 3HYES . AND . PLMSH . NE . 2HNO . AND .
      + PLDSP . NE . 3HYES . AND . PLDSP . NE . 2HNO . AND .
      + PLSTS . NE . 3HYES . AND . PLSTS . NE . 2HNO . AND .
      + PLMSP . NE . 3HYES . AND . PLMSP . NE . 2HNO )  ERDR1(4) = 1
C
       IF (NMTIRN . LT . 0 . OR . NMTIRN . GT . MMTIRN) ERDR1(5) = 1
       IF (NNDSH . LT . 2 . OR . NNDSH . GT . MNDSH) ERDR1(6) = 1
C
C
C*     3 IN THE FOLLOWING IS NO .  OF NODES IN AN ELEMENT  .
C
       KELSH = 3*NELSH
C
       IF (NELSH . NE . 0 . AND . KELSH . LT . NNDSH) ERDR1(7) = 1
C
       IF (NELSH . LT . 0 . OR . NELSH . GT . MELSH) ERDR1(8) = 1
C
       IF (NMTSH . LT . 0 . OR . NMTSH . GT . MMTSH) ERDR1(9) = 1
C
       IF (NSTRN . LT . 0 . OR . NSTRN . GT . MSTRN) ERDR1(10) = 1
C
       IF (NNDSR . LT . 0 . OR . NNDSR . GT . MNDSR) ERDR1(11) = 1
C
       IF (NELSR . NE . 0 . AND . NELSR . GT . NRWEL) ERDR1(12) = 1
C
       IF (NMTSR . LT . 0 . OR . NMTSR . GT . MMTSR) ERDR1(13) = 1
C
       IF (NIMSR . LT . 0 . OR . NIMSR . GT . MIMSR) ERDR1(14) = 1
C
       IF (NELISR . LT . 0 . OR . NELISR . GT . MELISR) ERDR1(15) = 1
C
       IF (NMTISR . LT . 0 . OR . NMTISR . GT . MMTISR) ERDR1(16) = 1
C
       IF (NRING . LT . 0 . OR . NRING . GT . MRING) ERDR1(17) = 1
C
       IF (NNDRN . LT . 0 . OR . NNDRN . GT . MNDRN) ERDR1(18) = 1
C
       IF (NELRN . LT . 0 . OR . NELRN . GT . MELRN) ERDR1(19) = 1
C
       IF (NMTRN . LT . 0 . OR . NMTRN . GT . MMTRN) ERDR1(20) = 1
C
       IF ( (NIMRN+NBRAC) . LT . 0 . OR . (NIMRN+NBRAC) . GT . MIMRN )
      +                                                  ERDR1(21) = 1
C
       IF ( (NELIRN+NELBR) . LT . 0 . OR . (NELIRN+NELBR) . GT . MELIRN )
      +                                                  ERDR1(22) = 1
C
       IF ( (NMTIRN+NMTBR) . LT . 0 . OR . (NMTIRN+NMTBR) . GT . MMTIRN )
```

```
     +                                                    ERDR1(23) = 1
C
      IF (NCRSM . LT . 0 . OR . NCRSM . GT . MCRSM) ERDR1(24) = 1
C
      IF (NPRDS . LT . 0 . OR . NPRDS . GT . MPRDS) ERDR1(25) = 1
C
      IF (NGAUS . LT . 0 . OR . NGAUS . GT . MGAUS) ERDR1(26) = 1
C
      IF (NSECT . LE . 0 . OR . NSECT . GT . MSECT) ERDR1(27) = 1
C
      IF (NSTRN . LT . 0)                             ERDR1(28) = 1
C
      IF (NRWEL . GT . MRWEL)                         ERDR1(29) = 1
C
      IF (NRWND . EQ . 1 . AND . STRTYP . NE . 8HCYLINDER)
     +                                                 ERDR1(30) = 1
C
      IF (NIMSR . NE . 0 . AND . NRWND . LT . 2)  ERDR1(31) = 1
C
      IF (NIMRN . NE . 0 . AND . NRWND . LT . 3)  ERDR1(32) = 1
C
      IF (NPRDS . EQ . 0 )                            ERDR1(33) = 1
C
C
      IF (NELSH . EQ . 0 . AND . NELSR . EQ . 0 . AND . NELISR .
     + EQ . 0 . AND . NELRN . EQ . 0 . AND . NELIRN . EQ . 0 .
     + AND . NELBR . EQ . 0)                          ERDR1(34) = 1
C
C
      IF (NNDSH . EQ . 0 . AND . NNDSR . EQ . 0 . AND . NNDISR .
     + EQ . 0 . AND . NNDRN . EQ . 0 . AND . NNDIRN . EQ . 0 .
     + AND . NNDBR . EQ . 0)                          ERDR1(35) = 1
C
      IF (NMTSH . EQ . 0 . AND . NMTSR . EQ . 0 . AND . NMTISR .
     + EQ . 0 . AND . NMTRN . EQ . 0 . AND . NMTIRN . EQ . 0 .
     + AND . NMTBR . EQ . 0)                          ERDR1(36) = 1
C
C
C
C*    PRINT AN ERROR MESSAGE IF A FLAG HAS BEEN SET UP I.E, ERDR1( )=1
C
      DO 150 IERR1 = 1 , 36
C
      IF (ERDR1(IERR1) . EQ . 0)                    GO TO 150
C
      IF (KERR1 . EQ . 1)                           GO TO 99
C
      KERR1 = 1
C
      WRITE(IO,90)
   90 FORMAT(1H1,T47,'*****  ERROR MESSAGES FROM CHECK1 : *****'////)
```

```
C
 99 WRITE(IO,100)
100 FORMAT(1H0,5X,'*')
C
      GO TO (101,102,103,104,105,106,107,108,109,110,111,112,113,
     + 114,115,116,117,118,119,120,121,122,123,124,125,126,127,128,
     + 129,130,131,132,133,134,135,136)  IERR1
C
101 WRITE(IO,1) STRTYP
C
  1 FORMAT(1H0,9X,'TYPE OF CONE NOT RECOGNIZED I.E, IT',
     +' IS NEITHER ''CONE'' NOR ''SPHERE'' NOR '/6X,
     + ''''ELLIPSOID'' NOR ''HYPERBOLOI'' NOR ''PARABOLOID'' NOR ',
     + ''''CYLINDRICAL'' NOR ''HYBRID'' NOR ''GENERAL'' , RATHER: ',
     + 'STRTYP = ', A10)
C
      GO TO 150
C
102 WRITE(IO,2) AUTGN
  2 FORMAT(1H0,9X,'AUTOMATIC DATA GENERATION IS NEITHER ''YES'' NOR',
     +''''NO'' , RATHER : ',A10)
C
      GO TO 150
C
103 WRITE (IO,3)  SLOSP, SLRAC, SLSTS, SLBKL
  3 FORMAT(1H0, 9X, 'THE INPUT FOR AT LEAST ONE OF THE FOLLOWING:',
     + / 45X,''''DISPLACEMENTS'', ''REACTIONS'', ''STRESSES'', OR ',
     + ''''BUCKLING'''',/10X, 'ANALYSIS IS NEITHER ''YES'', NOR ''NO''. ',
     + 'THEY ARE:', 2X, 4(A10,3X) )
C
      GO TO 150
C
104 WRITE(IO,4)  PLSTR, PLMSH, PLDSP, PLSTS, PLMSP
  4 FORMAT(1H0, 9X, 'THE INPUT FOR PLOTTING AT LEAST ONE OF THE',
     + 'FOLLOWING:'/59X, ' ''STRUCTURE'', ''MESH'', ''DISPLACEMENTS'',',
     + ' ''STRESSES'', ''MODE SHAPES'' '/9X, 'IS NEITHER ''YES'', NOR ',
     + ''''NO'', THEY ARE: ',2X, 5(A10,3X) )
C
      GO TO 150
C
105 WRITE (IO,5) NMTIRN, MMTIRN
  5 FORMAT(1H0,9X,'NO. OF MATERIALS IN THE INTERMEDIATE DOMAIN , ',
     +'NMTIRN = ',I4,' , IS EITHER NEGATIVE OR GREATER THAN ITS ',
     +'MAXIMUM , MMTIRN =',I4)
C
      GO TO 150
C
106 WRITE(IO,6) NNDSH,MNDSH
  6 FORMAT(1H0,9X,'NO. OF   NODES , NNDSH = ',I4,' , IS EITHER ',
     +'LESS THAN 2 OR GREATER THAN ITS MAXIMUM , MNDSH = ',I4)
C
```

```
      GO TO 150
C
  107 WRITE(IO,7) KELSH,NNDSH
    7 FORMAT(1H0,9X,'3*NELSH = ',I4,' , IS LESS THAN NO. OF NODES IN ',
     +'  , NNDSH =',I4)
C
      GO TO 150
C
  108 WRITE(IO,8) NELSH,MELSH
    8 FORMAT(1H0,9X,'NO. OF   ELEMENTS , NELSH = ',I4,' , IS ',
     +'EITHER NEGATIVE OR GREATER THAN ITS MAXIMUM , MELSH =',I4)
C
      GO TO 150
C
  109 WRITE(IO,9) NMTSH,MMTSH
    9 FORMAT(1H0,9X,'NO. OF   MATERIALS , NMTSH =',I4,' , IS ',
     +'EITHER NEGATIVE OR GREATER THAN ITS MAXIMUM  , MMTSH =',I4)
C
      GO TO 150
C
  110 WRITE(IO,10) NSTRN,MSTRN
   10 FORMAT(1H0,9X,'NO. OF STRINGERS , NSTRN = ',I4,' , IS ',
     +'EITHER NEGATIVE OR GREATER THAN ITS MAXIMUM , MSTRN =',I4)
C
      GO TO 150
C
  111 WRITE(IO,11) NNDSR,MNDSR
   11 FORMAT(1H0,9X,'NO. OF NODES IN THE STRINGERS , NNDSR =',I4,
     +' , IS EITHER NEGATIVE OR GREATER THAN ITS MAXIMUM , MNDSR =',I4)
C
      GO TO 150
C
  112 WRITE(IO,12)  NELSR,NRWEL
   12 FORMAT(1H0,9X,'NO. OF ELEMENTS IN ONE STRINGER , NELSR = ', I3,
     +' , IS GREATER THAN ITS NO. OF ROWS OF ELEMENTS , NRWEL = ',I4)
C
      GO TO 150
C
  113 WRITE(IO,13) NMTSR,MMTSR
   13 FORMAT(1H0,9X,'NO. OF STRINGER MATERIALS , NMTSR = ',I4,
     +' , IS EITHER NEGATIVE OR GREATER THAN ITS MAXIMUM , MMTSR = ',I4)
C
      GO TO 150
C
  114 WRITE(IO,14) NIMSR,MIMSR
   14 FORMAT(1H0,9X,'NO. OF INTERM. STRINGERS , NIMSR = ',I4,' , IS',
     +' EITHER NEGATIVE OR GREATER THAN ITS MAXIMUM , MIMSR =',I4)
C
      GO TO 150
C
  115 WRITE(IO,15) NELISR,MELISR
```

```
765      15 FORMAT(1H0,9X,'NO. OF ELEMENTS IN THE INTERM. STRINGERS , ',
766        +'NELISR = ',I4,' , IS EITHER NEGATIVE OR GREATER THAN ITS ',
767        +'MAXIMUM , MELISR =',I4)
768   C
769         GO TO 150
770   C
771     116 WRITE(IO,16) NMTISR, MMTISR
772      16 FORMAT(1H0,9X,'NO. OF MATERIALS IN THE INTERM. STRINGERS , ',
773        +'NMTISR = ',I4,' , IS EITHER NEGATIVE OR GREATER THAN ITS ',
774        +'MAXIMUM , MMTISR =',I4)
775   C
776         GO TO 150
777   C
778     117 WRITE(IO,17) NRING,MRING
779      17 FORMAT(1H0,9X,'NO. OF ELEMET ,NRING =',I4,' , IS EITHER ',
780        +'NEGATIVE OR GREATER THAN ITS MAXIMUM , MRING =',I4)
781   C
782         GO TO 150
783   C
784     118 WRITE(IO,18) NNDRN,MNDRN
785      18 FORMAT(1H0,9X,'NO. OF RING NODES , NNDRN =',I4,' , IS EITHER ',
786        +'NEGATIVE OR GREATER THAN ITS MAXIMUM , MNDRN =',I4)
787   C
788         GO TO 150
789   C
790     119 WRITE(IO,19) NELRN,MELRN
791      19 FORMAT(1H0,9X,'NO. OF RING ELEMENTS , NELRN =',I4,' , IS EITHER ',
792        +'NEGATIVE OR GREATER THAN ITS MAXIMUM , MELRN =',I4)
793   C
794         GO TO 150
795   C
796     120 WRITE(IO,20) NMTRN,MMTRN
797      20 FORMAT(1H0,9X,'NO. OF RING MATERIALS , NMTRN =',I4,' , IS EITHER',
798        +' NEGATIVE OR GREATER THAN ITS MAXIMUM , NMTRN =',I4)
799   C
800         GO TO 150
801   C
802     121 WRITE(IO,21) NIMRN,NBRAC,MIMRN
803      21 FORMAT(1H0,9X,'NO. OF INTERM. ELEMENTS, NIMRN =',I4,' , PLUS ',
804        +'NO. OF BRACES, NBRAC =',I4, ' , IS EITHER ', / 1H0, 9X,
805        +'NEGATIVE OR GREATER THAN ITS MAXIMUM , MIMRN =',I4)
806   C
807         GO TO 150
808   C
809     122 WRITE(IO,22) NELIRN,NELBR,MELIRN
810      22 FORMAT(1H0,9X,'NO. OF ELEMENTS IN INTERM. NELIRN =',I4,
811        + ' PLUS NO. OF ELEMENTS IN BRACES, NELBR =', I4,/1H0,9X,
812        +' , IS EITHER NEGATIVE OR GREATER THAN ITS MAXIMUM , MELIRN =',
813        +I4)
814   C
815         GO TO 150
```

```
C
  123 WRITE(IO,23) NMTIRN,NMTBR,MMTIRN
   23 FORMAT(1H0,9X,'NO. OF MATERIALS IN INTERM. NMTIRN =',I4,
     + ' PLUS NO. OF MATERIALS IN BRACES, NMTBR =', I4,/1H0,9X,
     +' , IS EITHER NEGATIVE OR GREATER THAN ITS MAXIMUM , MMTIRN =',
     +I4)
C
      GO TO 150
C
  124 WRITE(IO,24) NCRSM,MCRSM
   24 FORMAT(1H0,9X,'NUMBER OF COORDINATE SYSTEMS , NCRSM =',I4,
     +' , IS EITHER NEGATIVE OR GREATER THAN ITS MAXIMUM , MCRSM =',I4)
C
      GO TO 150
C
  125 WRITE(IO,25) NPRDS,MPRDS
   25 FORMAT(1H0,9X,'NUMBER OF PRESCRIBED DISPLACEMENTS , NPRDS =',I4
     +,' , IS EITHER NEGATIVE OR GREATER THAN ITS MAXIMUM , MPRDS =',I4)
C
      GO TO 150
C
  126 WRITE(IO,26) NGAUS,MGAUS
   26 FORMAT(1H0,9X,'NO. OF GAUSS QUADRATURE POINTS , NGAUS =',I4,
     +' IS EITHER NEGATIVE OR GREATER THAN ITS MAXIMUM , MGAUS =',I4)
C
      GO TO 150
C
  127 WRITE(IO,27) NSECT,MSECT
   27 FORMAT(1H0,9X,'NUMBER OF SECTORS , NSECT = ',I4,' , IS EITHER',
     +' LESS THAN OR EQUAL TO ZERO OR GREATER THAN ITS MAXIMUM, ',
     +'MSECT = ',I4,)
C
      GO TO 150
C
  128 WRITE(IO,28) NSTRN
   28 FORMAT(1H0,9X,'NUMBER OF STRINGERS, NSTRN = ',I4,' , IS NOT ',
     + 'POSITIVE')
C
      GO TO 150
C
  129 WRITE(IO,29) NRWEL, MRWEL
   29 FORMAT(1H0,9X,'NO. OF ROWS OF ELEMENTS, NRWEL =',I4,
     + ', IS GREATER THAN ITS MAXIMUM ,MRWEL =',I4,)
C
      GO TO 150
C
  130 WRITE(IO,30)
   30 FORMAT (1H0, 9X, 'WHEN NUMBER OF ROWS OF NODES IS 1, ',
     + 'STRUCTURE TYPE SHOULD BE KCYLINDERK  ')
C
      GO TO 150
```

```
C
  131 WRITE(IO,31)
   31 FORMAT(1HO, 9X, 'WHEN INTERMEDIATE STRINGERS EXIST, NUMBER',
     + ' OF ROWS OF NODES SHOULD AT LEAST BE 2')
C
      GO TO 150
C
  132 WRITE(IO,32)
   32 FORMAT(1HO, 9X, 'WHEN INTERMEDIATE ELEMENTS EXIST, NUMBER',
     + ' OF ROWS OF NODES SHOULD AT LEAST BE 3')
C
      GO TO 150
C
  133 WRITE(IO,33)
   33 FORMAT(1HO, 9X, 'NUMBER OF PRESCRIBED DISPLACEMENTS CANNOT',
     + ' BE ZERO, AS THIS WILL MAKE THE STRUCTURE UNSTABLE')
C
  134 WRITE(IO,34)
   34 FORMAT(1HO, 9X, 'NUMBER OF ELEMENT INTERMEDIATE',
     + ' STRINGER, RING, INTERMEDIATE RING, AND BRACE ELEMENTS',
     + ' CANNOT ALL BE ZERO')
C
  135 WRITE(IO,35)
   35 FORMAT(1HO, 9X, 'NUMBER OF ELEMENT , STRINGER, INTERMEDIATE',
     + ' STRINGER, RING, INTERMEDIATE RING, AND BRACE NODES',
     + ' CANNOT ALL BE ZERO')
C
  136 WRITE(IO,36)
   36 FORMAT(1HO, 9X, 'NUMBER OF ELEMENT, STRINGER, INTERMEDIATE',
     + ' STRINGER, RING, INTERMEDIATE RING, AND BRACE MATERIALS',
     + ' CANNOT ALL BE ZERO')
C
  150 CONTINUE
C
      DO 200 JLDCA = 1 , NLDCA
C
      DO 160 I = 36 , 39
C
  160 EROR1(I) = 0
C
      ICASE = LDCAS(JLDCA)
C
      IF(NCNLD(JLDCA).LT.0.OR.NCNLD(JLDCA).GT.MCNLD) EROR1(36) = 1
C
      IF(NDTLD(JLDCA).LT.0.OR.NDTLD(JLDCA).GT.MDTLD) EROR1(37) = 1
C
      IF(NELSH.EQ.0.AND.NDTLD(JLDCA).NE.0)           EROR1(38) = 1
C
      IF(NZNTP(JLDCA).LT.0.OR.NZNTP(JLDCA).GT.MZNTP) EROR1(39) = 1
C
C
```

```
      DO 190 IERR1 = 36 , 39
C
      IF(EROR1(IERR1).EQ.0)                                  GO TO 190
C
      IF(KERR1.EQ.1)                                         GO TO 165
C
      KERR1 = 1
C
      WRITE(IO,90)
  165 WRITE(IO,100)
C
      JERR1 = IERR1 - 35
C
      GO TO (171,172,173,174) , JERR1
C
  171 WRITE(IO,71) ICASE, NCNLO(JLOCA), MCNLO
   71 FORMAT(1H0,9X,'NUMBER OF CONCENTRATED NODAL LOADS IN LOADING',
     +' CASE ',I3,', NCNLO =',I4/1H ,9X,' , IS EITHER NEGATIVE OR',
     +' GREATER THAN ITS MAXIMUM , MCNLO =',I2)
C
      GO TO 190
C
  172 WRITE(IO,72) ICASE, NDTLO(JLOCA), MDTLO
   72 FORMAT(1H0,9X,'NUMBER OF DISTRIBUTED LOADINGS IN LOADING CASE',
     + I3,', NDTLO = ',I4,' , IS EITHER NEGATIVE OR GREATER THAN ITS',
     +' MAXIMUM , MDTLO = ',I2)
C
      GO TO 190
C
  173 WRITE(IO,73)
   73 FORMAT(1H0, 9X, 'WHEN NO   ELEMENTS EXIST, DISTRIBUTED',
     + ' LOADS CANNOT BE SPECIFIED (EXCEPT FOR WEIGHTS) ')
C
C
      GO TO 190
C
  174 WRITE(IO,74) ICASE, NZNTP(JLOCA), MZNTP
   74 FORMAT(1H0,9X,'NUMBER OF TEMPERATURE ZONES IN LOADING CASE',I3,
     +' , NZNTP = ',I4,' , IS EITHER NEGATIVE OR GREATER THAN ITS',
     +' MAXIMUM : MZNTP = ',I2)
C
  190 CONTINUE
C
  200 CONTINUE
C
      IF(KERR1 . EQ . 0)                                     RETURN
C
      WRITE(IO,205)
  205 FORMAT (1H0,5X,'*',//////,T44,
     +'*****  END OF ERROR MESSAGES FROM CHECK1  *****'/////)
      PRINT*,'**   MESSAGE FROM VAFA : EXPECT ERROR NUMBER',
```

```
      +' 65 FROM VAX BECAUSE SUBROUTINE ECHO TRIES TO KEEP ON ',
      +'READING AFTER THE END OF THE DATA FILE IS ENCOUNTERED. ',
      +'TO SEE THE ERROR MESSAGES FROM VARS ''TYPE'' OR ''PRINT'' ',
      +'FILE ''RESULT''.  '/'
C
      RETURN
      END
C
C
C
C     ------------------------------------------
C
C
C
C***  THIS SUBROUTINE READS DATA AND PRINTS IT IMMEDIATELY UNTIL
C***  INPUT DATA IS EXHAUSTED .
C
      SUBROUTINE ECHO(IN,IO)
C
      DIMENSION DATA(10)
C
      WRITE(IO,10)
   10 FORMAT(1H0,131,'######  THIS SUBROUTINE IS ECHOING IN A LISTING OF THE',
     +' POST-DISSECT DATA FOR A VARS RUN.'///)
   20 READ(IN,30) (DATA(I),I=1,8)
   30 FORMAT(8A10)
      WRITE(IO,40) (DATA(I),I=1,8)
   40 FORMAT(1X,8A10,2X,'|')
      GO TO 20
C
      END
C
C
C
C     ------------------------------------------
C
C
C
C***  THIS SUBROUTINE READS AND WRITES THE DATA ON ROWS OF ELEMENTS.
C
      SUBROUTINE ROWSHL(IN, IO, MRWND, MRWEL, NNDSH, NRWEL, NNDLR,
     + AUTGN, TITLE, RWSHL, ROWSH, ORIGN, RATIO, EROR2)
C
      CHARACTER*10,  CROWSH
C
      INTEGER AUTGN, TITLE(30), ROWSH(MRWND,2), EROR2(1), RWSHL(1)
C
      REAL ORIGN(MRWEL,1), RATIO(14)
C
      READ(IN,10) (TITLE(I), I=1, 16)
```

```
1020      10 FORMAT(8A10 / 8A10)
1021   C
1022         LNDSH = 0
1023   C
1024         DO 16 IRWEL = 1 , NRWEL
1025      16 RWSHL(IRWEL) = 0
1026   C
1027   C
1028         IF( AUTGN . NE . 3HYES )              GO TO 45
1029   C
1030   C                    A U T O M A T I C
1031   C
1032         WRITE(IO,20)
1033      20 FORMAT(1H ,///1H0,5X,'DATA ON ROWS OF ELEMENTS'/1H ,5X,
1034        + 'KKKKKKKKKKKKKKKKKKKKKKKKK'///T6,'ELEMENT ROW',T27,
1035        + 'NO. OF TOP NODES',T55,'RATIO'/)
1036   C
1037      30 FORMAT(1H0, T10, I2, T34, I2, T54, E12.4/)
1038   C
1039         DO 40 JRWEL = 1 , NRWEL
1040   C
1041         READ*, IRWEL, ROWSH(IRWEL,1), RATIO(IRWEL)
1042   C
1043         RWSHL(IRWEL) = 1
1044   C
1045         WRITE(IO,30) IRWEL, ROWSH(IRWEL,1), RATIO(IRWEL)
1046         ORIGN(IRWEL,1) = 0.0
1047         ORIGN(IRWEL,2) = 0.0
1048   C*    LNDSH IS THE TOTAL NUMBER OF NODES ON THE TOP ROWS OF ELEMENTS
1049   C*    SO FAR.
1050         LNDSH = LNDSH + ROWSH(IRWEL,1)
1051   C
1052         IF ( ROWSH(IRWEL,1) . LT . 2 . OR . ROWSH(IRWEL,1) . GT . NNDSH )
1053        +                                              EROR2(2) = 1
1054   C
1055      40 CONTINUE
1056   C
1057   C*    ENTER THE NUMBER OF NODES ON THE LAST ROW OF NODES IN ROWSH.
1058   C
1059         ROWSH(IRWEL+1, 1) = NNDSH - LNDSH
1060   C
1061         IF ( NNDLR . NE . ROWSH(IRWEL+1, 1) )        EROR2(15) = 1
1062   C
1063         GO TO 75
1064   C
1065   C              H Y B R I D   ( N O N A U T O M A T I C )
1066   C
1067      45 WRITE(IO,50)
1068      50 FORMAT(1H ,///1H0,5X,'DATA ON ROWS OF ELEMENTS'/1H ,5X,
1069        + 'KKKKKKKKKKKKKKKKKKKKKKKKK'///T6,'ELEMENT ROW',T27,
1070        + 'MERIDIONAL SHAPE', T53, 'LOCAL X',T70, 'LOCAL Z'/)
```

```
   60 FORMAT(1H0, T10, I2, T31, 1A10, T53, F7.3, T70, F7.3/)
C
      DO 70 JRWEL = 1 , NRWEL
C
      READ*, IRWEL, CROWSH, (ORIGN(IRWEL,J),J=1,2)
C
      RWSHL(IRWEL) = 1
C
C*    NOTE FUNCTION BOOL IS A FEATURE OF THE VAX WHICH TRANSFORMS
C*    CHARACTER TYPE DATA INTO INTEGER.
C
      ROWSH(IRWEL,2) = BOOL(CROWSH)
      WRITE(IO,60)  IRWEL, ROWSH(IRWEL,2), (ORIGN(IRWEL,J),J=1,2)
C
C*    LNDSH IS THE TOTAL NUMBER OF THE NODES ON THE TOP ROWS OF ELEMENTS
C
      LNDSH = LNDSH + ROWSH(IRWEL, 1)
      MERID = ROWSH(IRWEL, 2)
C
      IF( MERID . NE . 5HCONIC . AND . MERID . NE . 7HSPHERIC . AND .
     + MERID . NE . 8HELLIPTIC . AND . MERID . NE . 10HHYPERBOLIC .
     + AND . MERID . NE . 9HPARABOLIC . AND . MERID . NE . 9HCYLINDRIC .
     + AND . MERID . NE . 7HGENERIC )          EROR2(14) = 1
C
   70 CONTINUE
C
C*    CONSIDER ALL ROWS OF ELEMENTS AND CHECK TO SEE IF THE DATA
C*    FOR A GIVEN ROW IS MISSING.
C
   75 DO 80 IRWEL = 1 , NRWEL
C
      IF (RWSHL(IRWEL) . NE . 0)                    GO TO 80
C
      PRINT*, '    ****    ERROR MESSAGE FROM SUBROUTINE ROWSHL    ',
     + '*****'
C
      PRINT*, ' DATA ON ELEMENT ROW NUMBER ', IRWEL , ' IS ',
     + 'NOT PROVIDED'
C
C
      WRITE(IO,76) IRWEL
   76 FORMAT(1H0,10X,////'*****    ERROR MESSAGE FROM SUBROUTINE ',
     + 'ROWSHL    *****'///6X, '*'////1H0, 9X,
     + 'DATA ON ELEMENT ROW NUMBER ', I5, ' IS NOT PROVIDED'//////)
C
      PRINT*, ' EXPECT ERROR NUMBER 65 FORM THE VAX BECAUSE ',
     + ' SUBROUTINE ECHO TRIES TO KEEP ON READING AFTER THE END',
     + ' OF THE DATA FILE IS ENCOUNTERED, YOU MAY ALSO TYPE',
     + ' OR PRINT FILE  K RESULT K TO SEE THIS MESSAGE.'
C
C***  READ AND WRITE THE REMAINING DATA.
```

```
1122  C
1123        CALL ECHO(IN,IO)
1124  C
1125        GO TO 85
1126  C
1127     80 CONTINUE
1128  C
1129     85 RETURN
1130        END
1131  C
1132  C
1133  C
1134  C     --------------------    G  E  O  D  T  A    --------------------
1135  C
1136  C
1137  C
1138  C***  THIS SUBROUTINE READS, COMPUTES AND WRITES SOME GEOMETRIC
1139  C***  DATA :
1140  C
1141        SUBROUTINE GEODTA(IN, IO, AUTGN, NSECT, NRWND, NRWEL, TANFEE,
1142       + SINFEE, FEED, THATAD, THATAR, ELEV, RTOP, RBAS, HITE, TITLE)
1143  C
1144        INTEGER  AUTGN
1145  C
1146        DIMENSION TITLE(30)
1147  C
1148        READ(IN,100) (TITLE(I),I=1,16)
1149    100 FORMAT(8A10/8A10)
1150        CNVRT = 180. / ACOS(-1.0)
1151  C*
1152        IF(AUTGN.EQ.2HNO) GO TO 130
1153  C*    READ THE GEOMETRIC DATA AND PRINT IT :
1154        READ*,ELEV,DTOP,DBAS,HITE
1155        RTOP = DTOP/2
1156        RBAS = DBAS/2
1157  C*    FEE = ANGLE FORMED BY AXIS AND MERIDIAN IN RADIANS.
1158  C*    THATAD = ANGLE BETWEEN TWO ADJACENT STRINGERS , IN DEGREES .
1159  C*    THATAR = ANGLE BTWEEN TWO ADJACENT STRINGERS , IN DEGREES .
1160        TANFEE=(RBAS - RTOP)/HITE
1161        FEE = ATAN(TANFEE)
1162        SINFEE = SIN(FEE)
1163        FEED = CNVRT * FEE
1164        THATAD = 360.0 / NSECT
1165        THATAR= 2.0 * ACOS(-1.0) / NSECT
1166  C*    IF NONAUTOMATIC GENERATION IS REQUESTED SKIP :
1167        IF(AUTGN.EQ.2HNO) GO TO 130
1168        WRITE(IO,110) ELEV,DTOP,DBAS,HITE
1169    110 FORMAT(1H ,////1H0,5X,'GEOMETRIC DATA '
1170       +/1H ,5X,'KKKKKKKKKKKKKKKKKKKKKKKK'////6X,'ELEVATION',13X,'TOP ',
1171       + 'DIAMETER',9X,'BASE DIAMETER',13X,'HEIGHT',/1H0,4X,3(E12.5,10X),
1172       + E12.5/)
```

```
      WRITE(IO,120) THATAD,FEED,NRWND,NRWEL
  120 FORMAT(1H ,////,6X,'ANGLE BETW. TWO ADJC. STIFF''S',6X,'ANGLE ',
     +'BETW. AXIS AND MERID',7X,'NO. OF ROWS OF NODES',7X,'NO. OF ROWS',
     +' OF ELEM''S'/1H0,14X,F10.4,25X,F10.4,2X,2(24X,I2)/////)
C
      RETURN
C*    READ AND WRITE GEOMETRIC DATA FOR THE NONAUTOMATIC CASE :
  130 READ*,  ELEV, DTOP, DBAS, HITE
      WRITE(IO, 110) ELEV, DTOP, DBAS, HITE
C*    THATAD = ANGLE BETWEEN TWO ADJACENT STRINGERS , IN DEGREES .
C*    THATAR = ANGLE BTWEEN TWO ADJACENT STRINGERS , IN DEGREES .
      RTOP = DTOP/2
      RBAS = DBAS/2
      TANFEE=(RBAS - RTOP)/HITE
      FEE = ATAN(TANFEE)
      SINFEE = SIN(FEE)
      FEED = CNVRT * FEE
      THATAD = 360.0 / NSECT
      THATAR= 2.0 * ACOS(-1.0) / NSECT
      WRITE(IO,120) THATAD,FEED,NRWND,NRWEL
C
      RETURN
      END
C
C
C
C     --------------------    N  O  I  N  S  H    --------------------
C
C
C
C***  THIS SUBROUTINE GENERATES THE NODAL INCIDENCES ACCORDING TO
C***  THE SPECIFIED NUMBER OF NODES PER ROW. THEN IT PRINTS NODAL
C***  INCIDENCE DATA FOR EACH ELEMENT ALONG WITH ITS MATERIAL TYPE
C***  DESIGNATION.
C
      SUBROUTINE NDINSH (AUTGN, IN, IO, LZNSH, MRWEL, MELSH,
     + NRWND, NRWEL, TITLE, NODSH, MTDSH, SHMTI, SHMTJ, MTSHL, ROWSH)
C
      INTEGER AUTGN, MELT1, MELT2, NODSH(MELSH,1), MTDSH(1), MTSHL(1),
     + SHMTI(1), SHMTJ(1), ROWSH(MRWEL,1)
C
      REAL TITLE(30)
C
C*    IF NONAUTOMATIC DATA INPUT IS REQUESTED SKIP:
      IF(AUTGN . EQ . 2HNO)    GO TO 160
      NDBGUP = 1
      LASTEL = 0
C*    CONSIDER ALL ROWS OF   ELEMENTS.
C
      DO 110 IRWEL = 1 , NRWEL
C
```

```
      NNDUP = ROWSH(IRWEL, 1)
      NNDLW = ROWSH(IRWEL+1 ,1)
C*    'NENDEL' IS THE NUMBER OF THE ENDING ELEMENT OF THE CURRENT ROW.
      NENDEL = LASTEL + ROWSH(IRWEL ,1) + ROWSH(IRWEL+1 ,1) - 2
      NDEDUP = ROWSH(IRWEL, 1)
      NDBGLW = NDEDUP + 1
      NDEDLW = NDEDUP + ROWSH(IRWEL+1, 1)
      IF(IRWEL . EQ . 1)                    GO TO 10
C*    LASTEL IS THE NUMBER OF THE ENDING ELEMENT OF THE PREVIOUS ROW.
      LASTEL = LASTEL + ROWSH(IRWEL-1 ,1) + ROWSH(IRWEL ,1) - 2
C*    'NENDEL' IS THE NUMBER OF THE ENDING ELEMENT OF THE CURRENT ROW.
      NENDEL = LASTEL + ROWSH(IRWEL ,1) + ROWSH(IRWEL+1 ,1) - 2
C*    DETERMINE THE BEGGINING AND ENDING NODES OF THE UPPER AND LOWER
C*    ROWS.
      NDBGUP = NDBGUP + ROWSH(IRWEL-1, 1)
      NDEDUP = NDBGUP + ROWSH(IRWEL, 1) - 1
      NDBGLW = NDEDUP + 1
      NDEDLW = NDEDUP + ROWSH(IRWEL+1, 1)
C*    DETERMINE THE NUMBER OF TYPE 1 AND TYPE 2 ELEMENTS.
   10 NELT1 = NNDLW - 1
      NELT2 = NNDUP - 1
      IF(NNDUP . GT . NNDLW)                GO TO 45
      IF(NNDUP . EQ . NNDLW)                GO TO 95
C*    CASE 1: NUMBER OF UPPER NODES LESS THAN NUMBER OF LOWER NODES.
C*    CONSIDER TYPE 1 AND TYPE 2 ELEMENTS SYMMETRICALLY.
      HELT2 = NELT2 / 2
      LELT2 = HELT2 + 1
C
      DO 25 I = 1 , LELT2
C
      II = (I-1) * 2 + 1
C*    ELEMENT NUMBERS AT THE LEFT AND THE RIGHT OF THE CENTER, CURRENTLY
C*    BEING CONSIDERD.
      NELLT = LASTEL + II
      NELRT = NENDEL + 1 - II
      IF(NELT2 . GT . 1)                    GO TO 15
C*    WHEN ONLY ONE TYPE 2 ELEMENT:
C*    TYPE 1 ELEMENTS LEFT OF CENTER:
      NODSH(NELLT, 1) = NDBGUP
      NODSH(NELLT, 2) = NDBGLW
      NODSH(NELLT, 3) = NDBGLW + 1
C*    TYPE 1 ELEMENTS RIGHT OF CENTER:
      NODSH(NELRT, 1) = NDEDUP
      NODSH(NELRT, 2) = NDEDLW - 1
      NODSH(NELRT, 3) = NDEDLW
C*    THE TYPE 2 ELEMENT.
      IT2 = LASTEL + 1 + NELT1 - NELT1 / 2
      NODSH(IT2, 1) = NDBGLW + NDEDLW - (NDBGLW + NDEDLW) / 2
      NODSH(IT2, 2) = NDEDUP
      NODSH(IT2, 3) = NDBGUP
C*    SEE IF ANY TYPE 1 ELEMENTS ARE LEFT.
```

```
      NT1 = NELT1 - 2
      IF(NT1 . EQ . 0)                          GO TO 15
C*    NUMBER OF TYPE 1 ELEMENTS ON THE LEFT SIDE OF THE CENTER:
      NT1LT = NT1 - NT1 / 2
C
      DO 20 IT1 = 1 , NT1LT
C
      NELLT = LASTEL + 1 + IT1
      NELRT = NENDEL - IT1
C*    REMAINING TYPE 1 ELEMENTS, LEFT OF CENTER:
      NODSH(NELLT, 1) = NDBGUP
      NODSH(NELLT, 2) = NDBGLW + IT1
      NODSH(NELLT, 3) = NDBGLW + 1 + IT1
      IF(NT1LT . NE . NT1/2 . AND . IT1 . EQ . NT1LT)   GO TO 20
C*    REMAINING TYPE 1 ELEMENT, RIGHT OF CENTER:
      NODSH(NELRT, 1) = NDEDUP
      NODSH(NELRT, 2) = NDEDLW - 1 - IT1
      NODSH(NELRT, 3) = NDEDLW - IT1
C
   20 CONTINUE
C
C
      GO TO 110
C
C*    TYPE 1 ELEMENTS LEFT OF CENTER:
   15 NODSH(NELLT, 1) = NDBGUP + I - 1
      NODSH(NELLT, 2) = NDBGLW + I - 1
      NODSH(NELLT, 3) = NDBGLW + I
C*    TYPE 1 ELEMENTS RIGHT OF CENTER:
      NODSH(NELRT, 1) = NDEDUP + 1 - I
      NODSH(NELRT, 2) = NDEDLW - I
      NODSH(NELRT, 3) = NDEDLW + 1 - I
      IF(I . EQ . LELT2)                        GO TO 25
C*    TYPE 2 ELEMENTS, LEFT OF CENTER:
      NODSH(NELLT+1, 1) = NDBGLW + I
      NODSH(NELLT+1, 2) = NDBGUP + I
      NODSH(NELLT+1, 3) = NDBGUP - 1 + I
C*    TYPE 2 ELEMENTS, RIGHT OF CENTER:
      NODSH(NELRT-1, 1) = NDEDLW - I
      NODSH(NELRT-1, 2) = NDEDUP + 1 - I
      NODSH(NELRT-1, 3) = NDEDUP - I
C
   25 CONTINUE
C
C*    CONSIDER THE LAST TYPE 2 ELEMENT, IF ANY LEFT (THIS HAPPENS
C*    ONLY WHEN THE NUMBER OF TYPE 2 ELEMENTS IS ODD).
      IF( (2 * NELT2) . EQ . NELT2)      GO TO 30
      IT2 = (LASTEL + 1 + NENDEL) / 2
      NODSH(IT2, 1) = (NDBGLW + NDEDLW) / 2
      NODSH(IT2, 2) = (NDBGUP + NDEDUP) / 2 + 1
      NODSH(IT2, 3) = (NDBGUP + NDEDUP) / 2
```

```
1326  C*     FIND THE NUMBER OF THE REMAINING TYPE 1 ELEMENTS.
1327       30 NT1 = NELT1 - 2 * LELT2
1328          IF(NT1 . LE . 0)                          GO TO 110
1329  C*     FOR NUMBER OF TYPE 2 ELEMENTS EVEN OR ODD, NUMBER OF TYPE 1
1330  C*     ELEMENTS EVEN OR ODD.
1331  C*     LEFT OF CENTER
1332          NLT1 = NT1 / 2
1333          IF(NLT1 . EQ . 0)                         GO TO 37
1334          NDUP = (NDBGUP + NDEDUP) / 2
1335          NDLW = (NDBGLW + NDEDLW) / 2
1336  C
1337          DO 35 ILT1 = 1 , NLT1
1338  C
1339          NELLT = LASTEL + 2 * HELT2 + 1 + ILT1
1340          NODSH(NELLT, 1) = NDUP
1341          NODSH(NELLT, 2) = NDLW - NLT1 - 1 + ILT1
1342          NODSH(NELLT, 3) = NDLW - NLT1 + ILT1
1343  C
1344       35 CONTINUE
1345  C
1346  C*     RIGHT OF CENTER:
1347       37 NRT1 = NT1 - NLT1
1348          NDUP = (NDBGUP + NDEDUP) / 2 + 1
1349          NDLW = (NDBGLW + NDEDLW) / 2
1350  C
1351          DO 40 IRT1 = 1 , NRT1
1352  C
1353          NELRT = NENDEL - 2 * HELT2 - IRT1
1354          NODSH(NELRT, 1) = NDUP
1355          NODSH(NELRT, 2) = NDLW + NRT1 - IRT1
1356          NODSH(NELRT, 3) = NDLW + NRT1 - IRT1 + 1
1357  C
1358       40 CONTINUE
1359  C
1360  C
1361          GO TO 110
1362  C
1363  C*     CASE 2: NUMBER OF UPPER NODES GREATER THAN NUMBER OF LOWER NODES.
1364  C*     CONSIDER TYPE 1 AND TYPE 2 ELEMENTS SYMMETRICALLY.
1365       45 HELT1 = NELT1 / 2
1366          LELT1 = HELT1 + 1
1367  C
1368          DO 60 I = 1 , LELT1
1369  C
1370          II = (I-1) * 2 + 1
1371  C*     ELEMENT NUMBERS AT THE LEFT AND THE RIGHT OF CENTER, CURRENTLY
1372  C*     BEING CONSIDERED:
1373          NELLT = LASTEL + II
1374          NELRT = NENDEL + 1 - II
1375  C*     TYPE 2 ELEMENTS, LEFT OF CENTER:
1376          NODSH(NELLT, 1) = NDBGLW - 1 + I
```

```
      NODSH(NELLT, 2) = NDBGUP + I
      NODSH(NELLT, 3) = NDBGUP - 1 + I
C*    TYPE 2 ELEMENTS, RIGHT OF CENTER:
      NODSH(NELRT, 1) = NDEDLW + 1 - I
      NODSH(NELRT, 2) = NDEDUP + 1 - I
      NODSH(NELRT, 3) = NDEDUP - I
      IF(I . EQ . LELT1)                    GO TO 60
C*    TYPE 1 ELEMENTS, LEFT OF CENTER:
      NODSH(NELLT+1, 1) = NDBGUP + I
      NODSH(NELLT+1, 2) = NDBGLW + I - 1
      NODSH(NELLT+1, 3) = NDBGLW + I
C*    TYPE 1 ELEMENTS, RIGHT OF CENTER:
      NODSH(NELRT-1, 1) = NDEDUP - I
      NODSH(NELRT-1, 2) = NDEDLW - I
      NODSH(NELRT-1, 3) = NDEDLW + 1 - I
C
   60 CONTINUE
C
C*    CONSIDER THE LAST TYPE 1 ELEMENT, IF ANY LEFT.
      IF( (2 * HELT1) . EQ . NELT1 )     GO TO 70
      IT1 = LASTEL + 1 + NENDEL - (LASTEL + 1 + NENDEL) / 2
      NODSH(IT1, 1) = NDBGUP + NDEDUP - (NDBGUP + NDEDUP) / 2
      NODSH(IT1, 2) = NDBGLW + HELT1
      NODSH(IT1, 3) = NDEDLW - HELT1
C*    NOW CONSIDER THE REMAINIG TYPE 2 ELEMENTS, IF ANY.
   70 NT2 = NELT2 - 2 * LELT1
      IF(NT2 . LE . 0)                      GO TO 110
C*    TYPE 2 ELEM3NTS, LEFT OF CENTER:
      NLT2 = NT2 - NT2 / 2
C
      DO 80 ILT2 = 1 , NLT2
C
      NELLT = LASTEL + 2 * HELT1 + 1 + ILT2
      NODSH(NELLT, 1) = NDBGLW + HELT1
      NODSH(NELLT, 2) = NDBGUP + LELT1 + ILT2
      NODSH(NELLT, 3) = NDBGUP + LELT1 - 1 + ILT2
C
   80 CONTINUE
C
C*    TYPE 2 ELEMENTS, RIGHT OF CENTER:
      NRT2 = NT2 / 2
      IF(NRT2 . EQ . 0)                     GO TO 110
C
      DO 90 IRT2 = 1 , NRT2
C
      NELRT = NENDEL - 2 * HELT1 - IRT2
      NODSH(NELRT, 1) = NDEDLW - HELT1
      NODSH(NELRT, 2) = NDEDUP - LELT1 + 1 - IRT2
      NODSH(NELRT, 3) = NDEDUP - LELT1 - IRT2
C
   90 CONTINUE
```

```
1428  C
1429  C
1430        GO TO 110
1431  C
1432  C*    CASE 3: NUMBER OF UPPER NODES IS EQUAL TO NUMBER OF LOWER NODES:
1433  C
1434  C*    FIND THE NUMBER OF RECTANGLES COMPOSED OF A PAIR OF TYPE 1 AND
1435  C     TYPE 2 ELEMETS IN THE CURRENT ROW OF ELEMENTS.
1436  C
1437     95 NPAIR = NDSOUP - NDSGUP
1438  C
1439  C*    CONSIDER ALL PAIRS OF TYPE 1 AND TYPE 2 ELEMENTS IN THE CURRENT
1440  C*    ROW OF ELEMENTS.
1441  C
1442        DO 100 IPAIR = 1 , NPAIR
1443  C
1444        II = 2 * (IPAIR-1) + 1
1445        NT1 = LASTEL + II
1446        NT2 = NT1 + 1
1447        N1 = NDSGUP + IPAIR - 1
1448        N2 = N1 + 1
1449        N3 = NDSGLW + IPAIR - 1
1450        N4 = N3 + 1
1451  C
1452        NODSH(NT1,1) = N1
1453        NODSH(NT1,2) = N3
1454        NODSH(NT1,3) = N4
1455  C
1456        NODSH(NT2,1) = N4
1457        NODSH(NT2,2) = N2
1458        NODSH(NT2,3) = N1
1459  C
1460     100 CONTINUE
1461  C
1462     110 CONTINUE
1463  C
1464  C*    PRINT NODAL INCIDENCES JUST GENERATED AUTOMATICALLY FOR EACH
1465  C*      ELEMENT ALONG WITH ITS MATERIAL TYPE :
1466        WRITE(IO,120)
1467     120 FORMAT(1H0,5X,'ELEMENT NODAL INCIDENCES AND MATERIAL ',
1468       +'DESIGNATION FOR ELEMENT'/1H ,5X,'KKKKKKKKKKKKKKKKKKKKKKKKKKKKK',
1469       +'KKKKKKKKKKKKKKKKKKKKKKKKKKKKK'/////1H ,5X,'ELEMENT',15X,
1470       +'NODE I',15X,'NODE J',15X,'NODE K',15X,'MATERIAL DESIGNATION'//)
1471  C
1472        DO 150 IZNSH = 1 , LZNSH
1473  C
1474        ISHLI = SHMTI(IZNSH)
1475        ISHLJ = SHMTJ(IZNSH)
1476        IMTSH = MTDSH(IZNSH)
1477  C
1478        DO 140 IELSH = ISHLI , ISHLJ
```

```
      C
            MTSHL(IELSH) = IMTSH
            WRITE(IO,130)IELSH,NODSH(IELSH,1),NODSH(IELSH,2),NODSH(IELSH,3),
           +IMTSH
        130 FORMAT(1H0,7X,3(I8,18X),I3,25X,I3/)
      C
        140 CONTINUE
      C
      C
        150 CONTINUE
      C
      C
            RETURN
      C
      C                              N O N A U T O M A T I C
      C
      C*    READ THE NODAL INCIDENCE DATA FOR EACH   ELEMENT AND WRITE
      C*    IT ALONG WITH ITS MATERIAL DESIGNATION :
      C
        160 WRITE(IO,120)
            READ(IN,170) (TITLE(I),I=1,16)
        170 FORMAT(8A10/8A10)
      C
            DO 190 IZNSH = 1 , LZNSH
      C
            ISHLI = SHMTI(IZNSH)
            ISHLJ = SHMTJ(IZNSH)
            IMTSH = MTDSH(IZNSH)
      C
            DO 180 IELSH = ISHLI , ISHLJ
      C
            READ*, KELSH,NDSH1,NDSH2,NDSH3
            NODSH(KELSH,1) = NDSH1
            NODSH(KELSH,2) = NDSH2
            NODSH(KELSH,3) = NDSH3
            MTSHL(KELSH) = IMTSH
            WRITE(IO,130) KELSH,NDSH1,NDSH2,NDSH3,IMTSH
      C
        180 CONTINUE
      C
        190 CONTINUE
      C
      C
            RETURN
            END
      C
      C
      C
      C     --------------------    N  O  C  N  S  H    --------------------
      C
      C
```

The Processor Subprogram

```
C
C***  THIS SUBROUTINE FINDS AND PRINTS ALL OF THE NODES CONNECTED
C***  A CERTAIN NODE.
C
      SUBROUTINE NDCNSH (ID, MELSH, NNDSH, NELSH, NODSH, INDEX, NDCON)
C
      DIMENSION  NODSH(MELSH,1), INDEX(1), NDCON(1), KOUNT(100)
C
      LOGICAL REPEAT
C
C*    INITIALIZE VARIABLES
      KNT = 1
C
      DO 10 I = 1 , NNDSH
   10 KOUNT(I) = 0
C
      INDEX(1) = 1
C
      DO 50 INDSH = 1 , NNDSH
C
      DO 40 IELSH = 1 , NELSH
C
      DO 30 J = 1 , 3
C
      IF(INDSH.NE.NODSH(IELSH,J)) GO TO 30
C
      DO 20 I = 1 , 3
C
      IF(INDSH.EQ.NODSH(IELSH,I)) GO TO 20
C
      IBEG = INDEX(INDSH) + 1
      IEND = KNT
      REPEAT = .FALSE.
C
      DO 15 K = IBEG , IEND
      IF(NODSH(IELSH,I).EQ.NDCON(K)) REPEAT = .TRUE.
   15 CONTINUE
C
      IF(REPEAT) GO TO 20
C
      KOUNT(INDSH) = KOUNT(INDSH) + 1
      KNT = INDEX(INDSH) + KOUNT(INDSH)
      NDCON(KNT) = NODSH(IELSH,I)
C
   20 CONTINUE
C
      GO TO 40
C
   30 CONTINUE
C
   40 CONTINUE
```

```
C
      IPOS = KNT - KOUNT(INDSH)
      NDCON(IPOS) = KOUNT(INDSH)
      INDEX(INDSH+1) = KNT + 1
C
   50 CONTINUE
C
      WRITE(IO,60)
   60 FORMAT(1H ,/////6X,'NODAL CONNECTIONS'/1H ,5X,'KKKKKKKKKKKKKKKKK'
     +/////1H ,5X,'NODE',13X,'CONNECTED NODES')
C
      DO 80 INDSH = 1 , NNDSH
C
      IPOS = INDEX(INDSH)
      NND = NDCON(IPOS)
      ISEG = IPOS + 1
      IEND = IPOS + NND
      WRITE(IO,70) INDSH,(NDCON(I),I=ISEG,IEND)
   70 FORMAT(1H0,5X,I3,11X,10(I3,3X))
C
   80 CONTINUE
C
      WRITE(IO, 90)  IEND
C
   90 FORMAT (1H0, //, 6X, 'MINIMUM REQUIRED DIMENSION OF VECTOR',
     + ' NDCON :', I4//)
C
      RETURN
      END


C     ---------------------    C  O  N  S  T  N    ---------------------
C
C
C
C***  THIS SUBROUTINE SETS UP CERTAIN CONSTANTS.
C
      SUBROUTINE CONSTN (NDFND, NRWND, NSECT, NNDSH, NNDIN, NDFIF,
     + NDFEX, NDFIN, NDFSC, NINFA, NDFBL, NNDEX, NBEG, NEND, IOP,
     + IBX, PDCHEK, SINGAB, ICOND, IAFILE, IREF, DETCF)
C
      LOGICAL  PDCHEK, SINGAB
C
      NDFND = 12
      NDFIF = NDFND * NRWND
      NDFEX = 2 * NDFIF
      NDFIN = NDFND * NNDIN
      NDFSC = NDFND * NNDSH
      NINFA = NSECT
      NDFBL = NDFND * NDFND
      NNDEX = NNDSH - NNDIN
```

```
C*    PARAMETERS NEEDED IN SUBROUTINES SKYFAC AND SKYSOL.
      NBEG = 0
      NEND = NDFSC
      IOP = 0
      IBX = 1
      PDCHEK = .TRUE.
      SINGAB = .TRUE.
      ICOND = 0
      IAFILE = 0
      IREF = -1
      DETCF = 0
C
      RETURN
      END


C
C
C     --------------------    S E Q O R D    --------------------
C
C
C
C***  THIS SUBROUTINE GENERATES VECTORS SEQND AND NDORD. GIVEN THE
C***  ORDER SEQND GIVES THE NODE: GIVEN THE NODE NDORD GIVES THE ORDER.
C
      SUBROUTINE SEQORD (NNDSH, SEQND, NDORD)
C
      INTEGER  SEQND(1), NDORD(1)
C
C*    CONSIDER ALL NODES OF A SECTOR.
C
      DO 10 INDSH = 1 , NNDSH
C
      IORD = NNDSH + 1 - INDSH
      NDORD (INDSH) = IORD
      SEQND (IORD) = INDSH
C
   10 CONTINUE
C
      RETURN
      END
C
C
C
C     --------------------    I P N I C N    --------------------
C
C
C
C***  THIS SUBROUTINE CONSTRUCTS VECTORS ICONN AND IPNTR.ICONN GIVES THE
C***  NUMBER AND THE LIST OF NODES CONNECTED TO A GIVEN NODE, SAY INODE,
```

```
C***  AND APPEARING BEFORE INODE IN THE STIFFNESS MATRIX.
C***  IPNTR (INODE) IS THE POINTER TO ICONN FOR INFORMATION ON INODE.
C
      SUBROUTINE IPNICN (ID, MICON, NNOSH, SEGND, INDEX, NOCON,
     + IPNTR, ICONN)
C
      INTEGER SEGND(1), INDEX(1), NOCON(1), IPNTR(1), ICONN(1)
C
C*    IENT = IPNTR(NODE) ; NUM = ICONN(IENT) , WHERE NUM GIVES THE NEXT
C*    NUM ENTRIES OF ICONN, WHICH ARE THE NODES CONNECTED TO 'NODE' AND
C*    COME BEFORE IT IN ORDER.  IPOS = IPTK(NODE) , WHERE IPOS IS THE
C*    POSITION IN KSECT WHERE STIFFNESS COEFFICIENTS OF THE NODES
C*    CONNECTED TO 'NODE' AND LOWER IN ORDER START.
      IENT = 1
      ISTR = 1
C
      DO 30 IORDR = 1 , NNOSH
C
C*    FIND THE CURRENT NODE.
      NODE = SEGND(IORDR)
      IPNTR(NODE) = IENT
C*    FIND THE NODES CONNECTED TO THE CURRENT NODE.
      IPOS = INDEX(NODE)
      NNO = NOCON(IPOS)
      IBEG = IPOS + 1
      IEND = IPOS + NNO
C*    CONSIDER ALL NODES THAT HAVE A LOWER ORDER THAN 'NODE', AND SEE
C*    IF ANY OF THEM ARE AMONG THE NODES CONNECTED TO 'NODE' .
      NLOND = IORDR - 1
      KOUNT = 0
      ICONN(IENT) = 0
C
      DO 25 ILOND = 1 , NLOND
C
      LOWND = SEGND(ILOND)
C
      DO 20 INDCN = IBEG , IEND
C
      NDCN = NOCON(INDCN)
C
      IF(LOWND.NE.NDCN) GO TO 20
C
      ICONN(IENT) = KOUNT + 1
      KOUNT = KOUNT + 1
      ICONN(IENT+KOUNT) = NDCN
C
      GO TO 25
C
   20 CONTINUE
C
   25 CONTINUE
```

```
C
      IENT = IENT + KOUNT + 1
C
   30 CONTINUE
C
C
      IF (MICON . GE . IENT)                          GO TO 45
C
      WRITE(IO, 40)  IENT, MICON
C
   40 FORMAT (1H0, //, 6X, 'EXECUTION WAS ABORTED BY PHOENIX FROM ',
     + 'SUBROUTINE IPNICN BECAUSE ', / 6X, 'THE MINIMUM DIMENSION ',
     + 'REQUIRED FOR VECTOR ICONN , ', I4, ' IS GREATER THAN THE ',
     + 'DIMENSION CURRENTLY SET FOR IT, MICON = ', I4//)
C
      PRINT*, 'EXECUTION WAS ABORTED BY PHOENIX FROM ',
     + 'SUBROUTINE IPNICN BECAUSE THE MINIMUM DIMENSION ',
     + 'REQUIRED FOR VECTOR ICONN , ', IENT ,' IS GREATER THAN THE ',
     + 'DIMENSION CURRENTLY SET FOR IT, MICON = ', MICON
C
      STOP
C
C
   45 WRITE(IO,55)  IENT
   55 FORMAT (1H0, //, 6X, 'MINIMUM REQUIRED DIMENSION OF VECTOR',
     + ' ICONN :', I4, //)
C
C
      RETURN
      END
C
C
C
C     ---------------------    N  D  F  S  K  L    ---------------------
C
C
C
C***  THIS SUBROUTINE CALCULATES NDFSL. NDFSL IS THE NUMBER OF
C***  DEGREES OF FREEDOM IN THE SKY-LINE -STORED STIFFNESS MATRIX
C***  (KSECT). I.E.,THE NUMBER OF ENTRIES UNDER THE SKYLINE.
C
      SUBROUTINE NDFSKL (IO, NNOSH, NDFND, NDFSL, NDFSL, NOORD,
     + IPNTR, ICONN)
C
      INTEGER NOORD(1), IPNTR(1), ICONN(1)
C
C*    INITIALIZE NDFSL TO THE SUM OF THE ENTRIES OF THE NNOSH UPPER-
C*    TRIANGULAR MATRICES CORRESPONDING TO NODE I-NODE I BLOCKS.
      NDFSL = NNOSH * NDFND * (NDFND + 1) / 2
C
C*    CONSIDER ALL NODES.
```

```
C
      DO 10 INDSH = 1 , NNDSH
C
C*    FIND THE NUMBER OF ''NDFSC BY NDFS'' BLOCKS UNDER THE SKY-LINE.
C*    FIND THE ORDER OF THE FIRST BLOCK (NODE)UNDER THE SKY-LINE
C*    ASSOCIATED WITH INDSH.
      IPOS = IPNTR(INDSH)
      NCNDS = ICONN(IPOS)
C
      IF (NCNDS . EQ . 0)                    GO TO 10
C
      ND1 = ICONN(IPOS + 1)
      IORND1 = NDORD(ND1)
      IORND = NDORD(INDSH)
      NBLKS = IORND - IORND1
      NDFSL = NDFSL + NDFBL * NBLKS
C
   10 CONTINUE
C
      WRITE(IO,20)  NDFSL
C
   20 FORMAT (1H0, //, 6X, 'MINIMUM REQUIRED DIMENSION OF THE ',
     + 'STIFFNESS VECTOR KSECT, NDFSL = NDFSK= ', I7//)
C
      RETURN
      END
C
C
C
C     ---------------------    V E C T L D    --------------------
C
C
C
C***  THIS SUBROUTINE CONSTRUCTS VECTOR LD. LD(I+1) POINTS TO THE
C***  POSITION IN KSECT WHERE THE DIAGONAL ENTRY (I,I) IS LOCATED.
C
      SUBROUTINE VECTLD (NNDSH, NDFND, INDEX, NDCON, SEQND, LD)
      INTEGER INDEX(1), NDCON(1), SEQND(1), LD(1)
      LD(1) = 0
      ILD = 1
      NUNDR = 0
      IDIAG = 0
      DO 60 IORDR = 1 , NNDSH
      NODE = SEQND(IORDR)
      IPOS = INDEX(NODE)
      NCND = NDCON(IPOS)
      IBEG = IPOS + 1
      IEND = IPOS + NCND
C*    FIND THE NUMBER OF NODES THAT ARE LOWER IN ORDER THAN 'NODE'.
      NLWOR = IORDR - 1
      IF(NLWOR.EQ.0) GO TO 53
```

```
      DO 50 ILWOR = 1 , NLWOR
      LOWND = SEQND(ILWOR)
C*    SEE IF LOWND IS CONNECTED TO NODE.
      DO 45 JPOS = IBEG , IEND
      NDCN = NDCON(JPOS)
      IF(LOWND.NE.NDCN) GO TO 45
C*    NUNDR IS THE NUMBER OF BLOCKS IN THE COLUMN CORRESPONDING TO
C*    'NODE', UNDER THE SKY-LINE.
      NUNDR = NLWOR - ILWOR + 1
      GO TO 53
   45 CONTINUE
      NUNDR = NLWOR - ILWOR
   50 CONTINUE
   53 DO 55 IDFND = 1 , NDFND
      IDIAG = IDIAG + NDFND * NUNDR + IDFND
      ILD = ILD + 1
      LD(ILD) = IDIAG
   55 CONTINUE
   60 CONTINUE
      RETURN
      END
C
C
C
C     --------------------    O P E N D F    --------------------
C
C
C
C***  THIS SUBROUTINE OPENS ALL DISK FILES, IF NECESSARY (EXCEPT FOR
C***  IN=5 AND IO=6, WHICH HAVE ALREADY BEEN OPENED THROUGH THE PROGRAM
C***  STATEMENT).
C
      SUBROUTINE OPENDF (DSKF1, DSKF2, LRDF1, MAULD, MZNTP, MGAUS,
     + NLOCA, NNDIN, NDFIF, NDFSC, SLRAC, SLSTS, SLBKL, NZNTP)
C
      INTEGER  DSKF1, DSKF2, SLRAC, SLSTS, SLBKL
C
      DIMENSION  NZNTP(1)
C
      LOGICAL  SAVBT
C
C*    INITIALIZATIONS.
C
      SAVBT = . FALSE .
C
C
C
C*    SEE IF MATRIX BT NEEDS TO BE SAVED ON DISK FILE.
C
      IF (SLSTS . EQ . 3HYES . OR . SLBKL . EQ . 3HYES ) SAVBT = .TRUE.
C
```

```
1887  C*    CONSIDER ALL LOADING CASES.
1888  C
1889        DO 80 I = 1 , NLDCA
1890        IF ( NZNTP(I) . NE . 0 )                          SAVBT = . TRUE .
1891     80 CONTINUE
1892  C
1893  C
1894  C*    LRDF1 IS THE RECORD LENGTH OF DISK FILE DISK FILE 1.
1895  C
1896        LRDF1 = MGAUS * 3 * 38
1897  C
1898  C
1899  C
1900  C*    ADD 5 * MGAUS TO LBTSH FOR MATRIX CURVT(MGAUS,5).
1901  C
1902  C     LBTSH = MGAUS * 6 * 38 + MGAUS * 5
1903  C
1904        LBTSH = 233 * MGAUS
1905  C
1906  C
1907        IF (SAVBT)                                        LRDF1 = LBTSH
1908  C
1909        NDFEX = 2 * NDFIF
1910  C
1911  C*    THE FOLLOWING IF STATEMENT INSURES THAT THE NUMBER OF WORDS
1912  C*    IN A SINGLE COLUMN OF THE COMPLEX MATRIX KBNIF(NDFIF,NDFIF)
1913  C*    IS LESS THAN OR EQUAL TO THE RECORD LENGTH OF DISK FILE DSKF1.
1914  C
1915        IF (LRDF1 . LT . NDFEX)                           LRDF1 = NDFEX
1916  C
1917        IF (SLRAC . NE . 3HYES)                           GO TO 85
1918  C
1919        LFBSC = NDFSC * (1 + MAULD + MZNTP)
1920  C
1921        IF (LFBSC . GT . LRDF1)                           LRDF1 = LFBSC
1922  C
1923        LFBSC = 8 * NDFSC
1924  C
1925        IF (LFBSC . GT . LRDF1)                           LRDF1 = LFBSC
1926  C
1927     85 IF (LRDF1 . LT . 1260)                            LRDF1 = 1260
1928  C
1929  C*    OPEN DISK FILE DSKF1.
1930  C
1931        OPEN (DSKF1, STATUS='UNKNOWN',ACCESS='DIRECT',
1932       + FORM='UNFORMATTED', RECL=LRDF1)
1933  C
1934  C*    IF NO INTERNAL NODES EXIST, DO NOT OPEN DISK FILE DSKF2.
1935  C
1936        IF (NNDIN . EQ . 0)                               GO TO 95
1937  C
```

```
      OPEN (DSKF2, STATUS='UNKNOWN', ACCESS='DIRECT',
     + FORM='UNFORMATTED', RECL=NDFEX)
C
   95 RETURN
      END
C
C
C
C     --------------------    R  E  C  N  U  M    --------------------
C
C
C
C***  CALCULATE THE RECORD NUMBERS OF THE RECORDS LOCATED JUST BEFORE
C***  THE POSITION WHERE VARIOUS SETS OF DATA ARE TO BE WRITTEN
C***  (OR READ) ON (FROM) THE DISK FILES.
C
      SUBROUTINE RECNUM (LRDF1, MLOCA, MZNSL, NDFSC, NDFSL, NODPD,
     + NREST, SLVSC, SCNLI, SCNLJ, TYDTL, RFDTL, SCTPI, SCTPJ)
C
      INTEGER  SLOSP, SLRAC, SLSTS, SLBKL, PLOSP, PLSTS, PLMSP,
     + SLVSC(MLOCA,MZNSL,2), SCNLI(MLOCA,1), SCNLJ(MLOCA,1),
     + TYDTL(MLOCA,1), RFDTL(MLOCA,1), SCTPI(MLOCA,1), SCTPJ(MLOCA,1)
C
      DIMENSION  NODPD(1), NREST(1)
C
      LOGICAL  SAVBT
C
      COMMON  STRTYP, AUTGN, SLOSP, SLRAC, SLSTS, SLBKL,
     + PLSTR, PLMSM, PLOSP, PLSTS, PLMSP,
     + MSECT, MSTRN, MNDSR, MRING, MNDRN, MGAUS,
     + NNDSH, NELSH, NMTSH, NSTRN, NNDSR, NELSR, NMTSR, NIMSR,
     + NELISR, NMTISR, NRING, NNDRN, NELRN, NMTRN, NIMRN, NELIRN,
     + NMTIRN, NBRAC, NELBR, NMTBR, NGAUS, NCRSM, NPROS, NLOCA,
     + NSECT, NRWEL, NRWND, NDFND, NNDE1, NNDE2, NNDIN,
     + NNTSH, NBTSH, NGTSH, NSTSSH, NKCTND, NNTSR, NBTSR, NGTSR,
     + NTFSR, NSTSSR, NNTISR, NBTISR, NGTISR, NTFISR, NSTSIS, NNTRN,
     + NBTRN, NGTRN, NTFRN, NSTSRN, NNTIRN, NBTIRN, NGTIRN, NTFIRN,
     + NSTSIR, NTFBR, NSTSBR, NKRWPD, NCRLD, NFRCN, NABIF, NASCT,
     + NACNT, NFBSC, NRACT, NKSCT, NKSCF, NKKGW, NKGSC, NKGICL, NKBNIF,
     + NKGBNI, NEGVC, NKKGCN, NKGCN, NMSND, NMSCT, NKOEE, NKGOEE,
     + CSDCL, CSDMC, MCCOD, ILOCA, LCNLD, LDTLD, LZNTP, LZNSL,
     + DSKF1, DSKF2, IN, IO,
     + FEED, RTOP, RBAS, HITE, ELEV, THATAD,
     + TITLE(30), LOCAS(2), NCNLD(2), NDTLD(2), NZNTP(2), NZNSL(2),
     + RATIO(14), INDEX(45), NOCON(200), NDEX1(15), NDEX2(15),
     + NOINT(15), INTND(35), WKAREA(192)
C
C*    INITIALIZATION.
C
      NDFIF = NNDE1 * NDFND
      NDFEX = (NNDE1 + NNDE2) * NDFND
```

```
      SAVBT = . FALSE .
      LKBNIF = NNDE1 * NDFND * NDFEX
C
      KNSCT = 1
      MNSCT = 0
C
C*    FIND MNSCT, THE MAXIMUM NUMBER OF SECTORS FOR WHICH SOLUTION
C*    HAS BEEN REQUESTED.
C
C*    CONSIDER ALL LOADING CASES.
C
      DO 150 KLDCA = 1 , NLDCA
C
      IF (SLRAC . NE . 3HYES)                          GO TO 138
C
      KDTLD = NDTLD(KLDCA)
C
C*    CONSIDER ALL DISTRIBUTED LOADS.
C
      DO 136 IDT = 1 , KDTLD
C
      IF (TYDTL(KLDCA,IDT) . EQ . 4HSINE . OR .
     + TYDTL(KLDCA,IDT) . EQ . 4HAFRC )              NFB = 2
C
  136 CONTINUE
C
C*    FIND NSC, THE TOTAL NUMBER OF SECTORS FOR WHICH SOLUTION
C*    HAS BEEN REQUESTED FOR THE CURRENT LOADING CASE.
C
  138 NSC = 0
      KZNSL = NZNSL (KLDCA)
C
C*    CONSIDER ALL ZONES FOR WHICH SOLUTION HAS BEEN REQUESTED FOR THE
C*    CURRENT LOADING CASE.
C
      DO 140 IZNSL = 1 , KZNSL
C
      ISC = SLVSC(KLDCA,IZNSL,1)
      JSC = SLVSC(KLDCA,IZNSL,2)
C
      IF (ISC . LE . JSC)              NSC = NSC + JSC - ISC + 1
      IF (ISC . GT . JSC)              NSC = NSC+NSECT-ISC+JSC+1
C
  140 CONTINUE
C
C*    UPDATE THE MAXIMUM OF NSC.
C
      IF (KNSCT . LT . NSC)                          KNSCT = NSC
C
C     IF CALCULATION OF REACTIONS HAS BEEN REQUESTED, ADD 2 RECORDS TO
C*    MNSCT FOR EACH SOLUTION ZONE. THIS IS TO ACCOUNT FOR THE DISPLACE-
```

```
2040  C*    MENTS OF THE TWO SECTORS ON EITHER SIDE OF THE SOLUTION ZONE.
2041  C
2042        LNSCT = NSC + 2 * KZNSL
2043  C
2044        IF (SLRAC . EQ . 3HYES . AND . MNSCT . LT . LNSCT)  MNSCT = LNSCT
2045  C
2046    150 CONTINUE
2047  C
2048  C*    KSC IS THE NUMBER OF SECTORS FOR WHICH SOLUTION HAS BEEN
2049  C*    REQUESTED.
2050  C
2051        KSC = 1
2052        IF (PLSTS . EQ . 3HYES)                          KSC = KNSCT
2053  C
2054  C
2055  C*    DETERMINE IF MATRICES BT SHOULD BE SAVED.
2056  C
2057  C*    CONSIDER ALL LOADING CASES.
2058  C
2059        DO 10 I = 1 , NLOCA
2060  C
2061        IF (SLSTS . NE . 3HYES . AND . SLBKL . NE . 3HYES . AND .
2062       + NZNTP(I) . EQ . 0)                              GO TO 10
2063  C
2064        SAVBT = . TRUE .
2065  C
2066    10  CONTINUE
2067  C
2068  C*    BEGINNING POINTS ON DSKF1.
2069  C
2070  C*      ELEMENTS.
2071  C
2072        NKCTND = 0
2073        IF (NELSH . EQ . 0)                                GO TO 15
2074        NNTSH = 0
2075        NBTSH = NNTSH + NELSH
2076        NGTSH = NBTSH
2077        IF (SAVBT)                              NGTSH = NBTSH + NELSH
2078        NSTSSH = NGTSH
2079        IF (SLBKL . EQ . 3HYES)                 NSTSSH = NGTSH + NELSH
2080        NRECS = 0
2081  C
2082  C*    NRECS IS THE NUMBER OF 1260-WORD RECORDS OF STRESSES TO BE PLACED
2083  C*    ON DISK FILE.
2084  C
2085        IF (SLBKL . EQ . 3HYES . OR . PLSTS . NE . 2HNO)
2086       +                          NRECS = KSC * ( (6*NGAUS*NELSH/1260) + 1 )
2087  C
2088        NKCTND = NSTSSH + NRECS
2089  C
2090  C*    STRINGERS.
```

```
C
   15 NNTSR = NKCTND + NELSH
      NSTSSR = NNTSR
      NRECS = 0
      IF (NELSR . EQ . 0)                              GO TO 25
C
      NBTSR = NNTSR + NELSR
      NGTSR = NBTSR
      IF (SAVBT)                          NGTSR = NBTSR + NELSR
      NTFSR = NGTSR
      IF (SLBKL . EQ . 3HYES)             NTFSR = NGTSR + NELSR
      NSTSSR = NTFSR
      IF (SLDSP . EQ . 3HYES)             NSTSSR = NTFSR + NELSR
C
      IF (SLBKL . EQ . 3HYES . OR . PLSTS . NE . 2HNO)
     +                          NRECS = KSC * ( (4 * 7 * NELSR / 1260) + 1 )
C
C*    INTERMEDIATE STRINGERS.
C
   25 NNTISR = NSTSSR + NRECS
      NSTSIS = NNTISR
      NRECS = 0
      IF (NELISR . EQ . 0)                             GO TO 35
C
      NBTISR = NNTISR + NELISR
      NGTISR = NBTISR
      IF (SAVBT)                          NGTISR = NBTISR + NELISR
      NTFISR = NGTISR
      IF (SLBKL . EQ . 3HYES)             NTFISR = NGTISR + NELISR
      NSTSIS = NTFISR
      IF (SLDSP . EQ . 3HYES)             NSTSIS = NTFISR + NELISR
C
      IF (SLBKL . EQ . 3HYES . OR . PLSTS . NE . 2HNO)
     +                          NRECS = KSC * ( (4 * 7 * NELISR / 1260) + 1 )
C
C*
C
   35 NNTRN = NSTSIS + NRECS
      NSTSRN = NNTRN
      NRECS = 0
      IF (NELRN . EQ . 0)                              GO TO 45
C
      NBTRN = NNTRN + NELRN
      NGTRN = NBTRN
      IF (SAVBT)                          NGTRN = NBTRN + NELRN
      NTFRN = NGTRN
      IF (SLBKL . EQ . 3HYES)             NTFRN = NGTRN + NELRN
      NSTSRN = NTFRN
      IF (SLDSP . EQ . 3HYES)             NSTSRN = NTFRN + NELRN
      IF (SLBKL . EQ . 3HYES . OR . PLSTS . NE . 2HNO)
     +                          NRECS = KSC * ( (4 * 7 * NELRN / 1260) + 1 )
```

```
C
C*    INTERMEDIATE ELEMENTS.
C
   45 NNTIRN = NSTSRN + NRECS
      NSTSIR = NNTIRN
      NRECS = 0
C
      IF (NELIRN . EQ . 0)                                  GO TO 50
C
      NGTIRN = NNTIRN + NELIRN
      NGTIRN = NGTIRN
      IF (SAVGT)                                NGTIRN = NSTIRN + NELIRN
      NTFIRN = NGTIRN
      IF (SLBKL . EQ . 3HYES)                   NTFIRN = NGTIRN + NELIRN
      NSTSIR = NTFIRN
      IF (SLOSP . EQ . 3HYES)                   NSTSIR = NTFIRN + NELIRN
C
      IF (SLBKL . EQ . 3HYES . OR . SLSTS . NE . 2HNO)
     +                                    NRECS = ( (6 * 7 * NELIRN / 1260) + 1 )
C
C
C*    BRACES.
C
   50 NTFBR = NSTSIR + NRECS
      NSTSBR = NTFBR
      NRECS = 0
C
      IF (NELBR . EQ . 0)                                   GO TO 55
C
      NSTSBA = NTFBR
C
      IF (SLOSP . EQ . 3HYES . OR . SLSTS . EQ . 3HYES)
     +                                    NRECS = KTC * ( (NELBR/1260) + 1 )
C
   55 NKRWPD = NSTSBA + NRECS
      NRDOF = 0
C
C*    IF NO DISPLACEMENTS ARE PRESCRIBED, OR CALCULATION OF THE
C*    REACTIONS IS NOT REQUESTED, SKIP.
C
      IF (NPRDS . EQ . 0 . OR . SLRAC . NE . 3HYES)  GO TO 75
C
C*    FIND NRDF, TWICE THE NUMBER OF THE RESTRAINED DEGREES OF FREEDOM
C*    OF THE EXTERNAL 1 AND 2 NODES SPECIFICALLY MENTIONED PLUS THE
C*    NUMBER OF THE RESTRAINED DEGREES OF FREEDOM ON THE INTERNAL NODES.
C*    ALSO FIND NRDOF, THE SPECIFIED NUMBER OF DEGREES OF FREEDOM.
```

```
C
      DO 70 IPRDS = 1 , NPRDS
C
      NODPD = NODPD(IPRDS)
C
      IF (INTNO(NODPD) . NE . 3HYES)                    GO TO 65
C*    INTERNAL NODES.
      NRDF = NRDF + NREST(IPRDS)
      NRDOF = NRDOF + NREST(IPRDS)
C
      GO TO 70
C
C*    EXTERNAL NODES.
C
   65 NRDF = NRDF + 2* NREST(IPRDS)
C
      NRDOF = NRDOF + NREST(IPRDS)
C
   70 CONTINUE
C
   75 NCRLD = NKRWPD + NRDF
      NFRCN = NCRLD + 1
      NABIF = NFRCN + 1
C
      LSCCN = 1
C
C*    FOR THE FOURIER COEFFIENTS OF THE CONCENTRATED NODAL LOADS,
C*    ALLOCATE 1 RECORD IF ALL SPECIFIED CONCENTRATED LOADS ARE
C*    SYMMETRIC, AND NSECT RECORDS, OTHERWISE.
C
C*    CONSIDER ALL LOADING CASES.
C
      DO 90 I = 1 , NLDCA
C
      KCNLD = NCNLD(I)
      IF (KCNLD . EQ . 0)                               GO TO 90
C
      DO 80 J = 1 , KCNLD
C
      ISC = SCNLI(I,J)
      JSC = SCNLJ(I,J)
C
      IF (ISC . LE . JSC . AND . (JSC-ISC+1) . NE . NSECT)
     +                                                  LSCCN = NSECT
      IF (ISC . GT . JSC . AND . (NSECT-JSC+ISC-1) . NE . NSECT)
     +                                                  LSCCN = NSECT
C
   80 CONTINUE
C
   90 CONTINUE
C
```

```
C*    FIND THE NUMBER OF FOURIER COEFFICIENTS OF DISPLACEMENTS TO BE
C*    PLACED ON DISK FILE.
C
      NABIF = NFRCN + LSCCN
      NASCT = NABIF + LSCCN
C
      IF (LSCCN . EQ . NSECT)                          GO TO 135
C
      LSECT = NSECT / 2
      LSCDT = 1
C
C*    DISTRIBUTED LOADS.
C
C*    CONSIDER ALL LOADING CASES, AND DETERMINE THE MAXIMUM NUMBER OF
C*    NONZERO FOURIER TRANSFORMATIONS OF DISPLACEMENTS.
C
      DO 110 I = 1 , NLDCA
C
      KDTLD = NDTLD(I)
      IF (KDTLD . EQ . 0)                              GO TO 110
C
      DO 100 J = 1 , KDTLD
C
      IF ( TYDTL(I,J) . NE . 7HUNIFORM . AND . TYDTL(I,J) . NE .
     + 6HLINEAR . AND . TYDTL(I,J) . NE . 4HSNOW ) GO TO 95
C
C*    FOR ANTISYMMETRIC UNIFORMLY OR LINEARLY DISTRIBUTED LOADS.
C
      IF (RFDTL(I,J) . NE . -1)                        GO TO 95
C
      IF ( (2 * LSECT) . EQ . NSECT . AND . LSCDT . LT . (LSECT+1) )
     +                                                 LSCDT = LSECT + 1
C
      IF ( (2 * LSECT) . NE . NSECT )                  LSCDT = NSECT
C
C*    FOR SINE OR AFRO LOADS.
C
   95 IF ( TYDTL(I,J) . EQ . 4HSINE . AND . LSCDT . LT . 3 )  LSCDT = 3
C
      IF ( TYDTL(I,J) . EQ . 4HAFRO . AND . LSCDT . LT . 5 )  LSCDT = 5
C
  100 CONTINUE
C
  110 CONTINUE
C
      IF (LSCDT . GT . LSCCN)          NABIF = NFRCN + LSCDT
      IF (LSCDT . GT . LSCCN)          NASCT = NABIF + LSCDT
      IF (LSCDT . EQ . NSECT)                          GO TO 135
C
      LSCTP = 1
C
```

```
C*    THERMAL LOADS.
C
C*    CONSIDER ALL LOADING CASES.
C
      DO 130 I = 1 , NLDCA
C
      KZNTP = NZNTP(I)
      IF (KZNTP . EQ . 0)                                  GO TO 130
C
      DO 120 J = 1 , KZNTP
C
      ISC = SCTPI(I,J)
      JSC = SCTPJ(I,J)
C
      IF (ISC . LE . JSC . AND . (JSC-ISC+1) . NE . NSECT)
     +                                                    LSCTP = NSECT
C
      IF (ISC . GT . JSC . AND . (NSECT-JSC+ISC-1) . NE . NSECT)
     +                                                    LSCTP = NSECT
C
  120 CONTINUE
C
  130 CONTINUE
C
      IF (LSCTP . GT . LSCDT)          NABIF = NFRCN + LSCTP
      IF (LSCTP . GT . LSCDT)          NASCT = NABIF + LSCTP
C
  135 NACNT = NASCT
C
      NFB = 1
C
      IF (SLSTS . EQ . 3HYES . OR . PLDSP . EQ . 3HYES . OR . PLSTS .
     +    EQ . 3HYES . OR . SLBKL . EQ . 3HYES)     NACNT = NASCT + KNSCT
C
      IF (SLRAC . EQ . 3HYES)          NACNT = NASCT + MNSCT
C
      NRACT = NACNT
C
      IF (SLSTS . EQ . 3HYES . OR . PLDSP . EQ . 3HYES . OR . PLSTS .
     +    EQ . 3HYES . OR . SLBKL . EQ . 3HYES)     NRACT = NACNT + KNSCT
C
      IF (SLRAC . EQ . 3HYES)          NRACT = NACNT + MNSCT
C
      NRBCS = 0
C
      IF (SLRAC . NE . 3HYES)                              GO TO 155
C
      NFBSC = NACNT
C
      IF (SLSTS . EQ . 3HYES . OR . PLDSP . EQ . 3HYES . OR . PLSTS .
     +    EQ . 3HYES . OR . SLBKL . EQ . 3HYES)     NFBSC = NACNT + MNSCT
```

```
C
      NRACT = NFBSC + NFB
C
C*    FIND THE NUMBER OF 1260-WORD RECORDS, CONTAINING THE REACTIONS, TO
C*    BE PLACED ON DISK FILE.
C
      NRECS = (12 * NRDOF / 1260) + 1
C
C*    IF BUCKLING SOLUTION HAS BEEN REQUESTED, THE CONVENTIONAL
C*    STIFFNESS MATRIX SHOULD BE WRITTEN ON DISK FILE DSKF1.
C
  155 IF (SLBKL . EQ . 3HYES)                       NKSCT = NRACT + NRECS
C
C*    IF PLOTTING OF THE DISPLACEMENTS HAS BEEN REQUESTED, AND THE
C*    NUMBER OF OF LOADING CASES IS GREATER THAN 1, THE CONVENTIONAL
C*    STIFFNESS MATRIX SHOULD BE WRITTEN ON DISK FILE DSKF1.
C
      IF (PLDSP . EQ . 3HYES . AND . NLOCA . GT . 1)
     +                                             NKSCT = NRACT + NRECS
C
C*    IF PLOTTING OF THE STRESSES HAS BEEN REQUESTED, AND THE
C*    NUMBER OF OF LOADING CASES IS GREATER THAN 1, THE CONVENTIONAL
C*    STIFFNESS MATRIX SHOULD BE WRITTEN ON DISK FILE DSKF1.
C
      IF (PLSTS . EQ . 3HYES . AND . NLOCA . GT . 1)
     +                                             NKSCT = NRACT + NRECS
C
C
C
      NEGVC = NKSCT
C
C*    IF PLOTTING OF DISPLACEMENTS OR STRESSES HAS BEEN REQUESTED,
C*    AND THE NUMBER OF THE LOADING CASES IS GREATER THAN 1, AND
C*    INTERNAL NODES EXIST, THE FACTORIZED CONVENTIONAL STIFFNESS
C*    MATRIX WILL BE WRITTEN ON DISK FILE DSKF1.
C
      IF (PLDSP . NE . 3HYES . OR . NLOCA . LE . 1 . OR . NNDIN . EQ
     +    . 0)                                     GO TO 162
C
      NKSCF = NKSCT + (NDFSL / LRDF1) + 1
C
      NEGVC = NKSCF
C
  162 IF (PLSTS . NE . 3HYES . OR . NLOCA . LE . 1 . OR . NNDIN . EQ
     +    . 0)                                     GO TO 163
C
      NKSCF = NKSCT + (NDFSL / LRDF1) + 1
C
      NEGVC = NKSCF
C
C*    IF NO BUCKLING SOLUTION HAS BEEN REQUESTED, SKIP.
```

```
C
  163 IF (SLBKL . EQ . 2HNO)                                  GO TO 165
C
      NKSCF = NKSCT + (NDFSL / LRDF1) + 1
      NKKGW = NKSCF + (NDFSL / LRDF1) + 1
C
C*    THE SUM OF THE CONVENTIONAL STIFFNESS MATRIX AND THE GEOMETRIC
C*    STIFFNESS MATRIX DUE TO THE DEAD LOAD IS WRITTEN (IN OVERLAY(2,0))
C*    AND READ (IN SUBROUTINE BUCKLE) IN GROUPS OF 1260 WORDS, STARTING
C     FROM RECORD NUMBER NKKGW+1 OF DISK FILE DSKF1.
C
      NKGSC = NKKGW + (NDFSL / 1260) + 1
C
      NKGICL = NKGSC + (NDFSL / LRDF1) + 1
C
      NKBNIF = NKGICL
C
      IF (NNDIN . EQ . 0)                                     GO TO 164
C
      NKBNIF = NKGICL + NNDIN * (12 * NDFSC / LRDF1 + 1)
C
C*    NCOLS IS THE INTEGER NUMBER OF COLUMNS OF THE COMPLEX MATRIX
C*    KBNIF(NDFIF,NDFIF), WHICH CAN BE ACCOMODATED IN ONE RECORD OF
C*    DISK FILE DSKF1 (I. E., LRDF1).
C
  164 NCOLS = LRDF1 / (2 * NDFIF)
C
C*    NRECS IS THE NUMBER OF RECORDS IT TAKES TO WRITE (READ) THE
C*    COMPLEX MATRIX KBNIF(NDFIF,NDFIF) ON (FROM) DISK FILE DSKF1.
C
      NRECS = NDFIF / NCOLS
C
      IF ( (NRECS*NCOLS) . NE . NDFIF )                       NRECS = NRECS + 1
C
      NKGBNI = NKBNIF + NSECT * NRECS
      NEGVC  = NKGBNI + NSECT * NRECS
C
C
C*    IF PLOTTING OF THE MODE SHAPES HAS NOT BEEN REQUESTED, SKIP.
C
  165 IF (PLMSP . NE . 3HYES)                                 GO TO 405
C
C
C*    IN SUBROUTINE SLVEIG A MAXIMUM OF 20 MODE SHAPES ARE ALLOWED TO
C*    BE WRITTEN ON DISK FILE DSKF1.
C
C*    FOR KCTND-CONVENTIONAL + KGCTND-DEAD LOAD.
C
      NKKGCN = NEGVC + 20
C
C*    FOR KGCTND-LIVE LOAD.
```

```
C
      NKGCN = NKKGCN + NELSH
C
C
C*    FIND THE MAXIMUM NUMBER OF MODE SHAPES THAT ARE GOING TO
C*    BE WRITTEN ON DISK FILE DSKF1.
C
      NMDSP = 0
C
C*    CONSIDER ALL LOADING CASES.
C
      DO 300 JLOCA = 1 , NLOCA
C
      IMDSP = 0
C
      KZNSL = NZNSL(JLOCA)
C
C*    CONSIDER THE MAXIMUM NUMBER OF BUCKLING LOAD COEFFICIENTS.
C*    THE NUMBER IS SPECIFIED AS 20 IN SUBROUTINE SLVEIG.
C
      DO 280 IEIG = 1 , 20
C
C*    FOR EACH EIGENVALUE (BUCKLING LOAD) FIND THE TWO POSSIBLE
C*    MODE SHAPES. THESE TWO MODE SHAPES ARISE FROM THE REAL AND
C*    THE COMPLEX PARTS OF THE FOURIER EIGENVECTOR ASSOCIATED WITH
C*    THE CURRENT EIGENVALUE. (IPART 1 INDICATES REAL PART; IPART 2
C*    INDICATES COMPLEX PART)
C
      DO 260 IPART = 1 , 2
C
C
C*    CONSIDER ALL SOLUTION ZONES.
C
      DO 240 IZNSL = 1 , KZNSL
C
C*    FIND THE BEGINNING AND THE ENDING SECTORS OF THE CURRENT
C*    SOLUTION ZONE.
C
      JSC = SLVSC(JLOCA,IZNSL,1)
      KSC = SLVSC(JLOCA,IZNSL,2)
C
C
C*    IF JSC K KSC, CONSIDER 2 RANGES.
C
      DO 230 IRANG = 1 , 2
C
      IF ( SLVSC(JLOCA,IZNSL,1) . LE . SLVSC(JLOCA,IZNSL,2) ) GO TO 215
C
      JSC = SLVSC(JLOCA,IZNSL,1) * (2 - IRANG) + IRANG - 1
      KSC = NSECT * (2 - IRANG) + SLVSC(JLOCA,IZNSL,2) * (IRANG - 1)
C
```

```
C*     CONSIDER ALL OF THE SECTORS FOR THE CURRENT ZONE.
C
  215 DO 220 ISC = JSC , KSC
C
      IMDSP = IMDSP + 1
C
  220 CONTINUE
C
  230 CONTINUE
C
  240 CONTINUE
C
  260 CONTINUE
C
  280 CONTINUE
C
      IF (IMDSP . GT . NMDSP)                          NMDSP = IMDSP
C
  300 CONTINUE
C
C*     FOR MODE SHAPES OF NODES.
C
      NMSND = NKGCN + NELSH
C
C
C*     FOR MODE SHAPES OF CENTROIDS.
C
      NMSCT = NMSND + NMDSP
C
C
C*     IF NO INTERNAL NODES EXIST, SKIP.
C
  405 IF (NNDIN . EQ . 0)                              GO TO 415
C
C*     BEGINNING POINTS ON DISK FILE DSKF2.
C
      NKOEE = 0
      NKGOEE = NKOEE + NDFEX
C
C
  415 RETURN
      END
C
C
C
C
```

```
C     --------------------    G  A  U  S  T  R    --------------------
C
C
C
C
C
      SUBROUTINE GAUSTR (MGAUS, NGAUS, GUSWT, GUSTR)
C
      DIMENSION  GUSWT(1), GUSTR(MGAUS,1), CONST(2,2), COORD(3),
     + IPT7(7), IPT12(12)
C
      DATA  (COORD(I), I=1,3)/ 0.636502499121399, 0.310352451033785,
     + 0.053145049844816 /
C
      DATA  (IPT7(I), I=1, 7)/ 4, 7, 2, 3, 1, 5, 6 /
C
      DATA  (IPT12(I), I=1, 12)/ 1, 9, 10, 4, 7, 8, 2, 3, 5, 6, 11, 12 /
C
      IF (NGAUS . NE . 7)                          GO TO 25
C
C*    THE 7-POINT RULE.
C
C*    ASSIGN WEIGHTS
C
      GUSWT(1) = 0.1259391805
      GUSWT(2) = 0.1323941527
      GUSWT(3) = GUSWT(2)
      GUSWT(4) = 0.225
      GUSWT(5) = GUSWT(1)
      GUSWT(6) = GUSWT(1)
      GUSWT(7) = GUSWT(2)
      CONST(1,1) = 0.0597158717
      CONST(1,2) = 0.4701420641
      CONST(2,1) = 0.7974269853
      CONST(2,2) = 0.1012865073
C
C*    ASSIGN THE TRIANGULAR COORDINATES OF THE QUADRATUR POINTS
C
      GUSTR(4,1) = 1./3.
      GUSTR(4,2) = GUSTR(4,1)
      GUSTR(4,3) = GUSTR(4,1)
C
      DO 20 K = 1 , 2
C
      DO 10 I = 2 , 4
C
      IPONT = (K-1) * 3 + I
C
C*    USE POINTER IPT7 TO ASSIGN THE CURRENT GAUSS POINT COORDINATES
C*    TO A POINT SUCH THAT FOR TYPE 1 SHELL ELEMENTS, THE MERIDIONAL
C*    COORDINATE XI INCREASES WITH GAUSS POINT NUMBER, IGAUS.
```

```
C
      IGAUS = IPT7(IPONT)
C
      DO 10 J = 1 , 3
C
      GUSTR(IGAUS,J) = CONST(K,2)
C
      IF ( J . EQ . (I-1) )          GUSTR(IGAUS,J) = CONST(K,1)
C
   10 CONTINUE
C
   20 CONTINUE
C
   25 IF (NGAUS . NE . 12)                              GO TO 65
C
C*    THE 12-POINT RULE.
C
C*    6 GAUSS POINTS.
C
      DO 40 JGAUS = 1 , 3
C
C*    USE POINTER IPT12 TO ASSIGN THE CURRENT GAUSS POINT COORDINATES
C*    AND WEIGHTS TO A POINT SUCH THAT FOR TYPE 1 SHELL ELEMENTS, THE
C*    MERIDIONAL COORDINATE XI INCREASES WITH GAUSS POINT NUMBER, IGAUS.
C
      IGAUS = IPT12(JGAUS)
      KGAUS = IPT12(JGAUS+3)
C
      DO 30     J = 1 , 3
C
      GUSTR(IGAUS,J) = 0.063089014491502
      GUSTR(KGAUS,J) = 0.249286745170910
C
   30 CONTINUE
C
      GUSTR(IGAUS,JGAUS) = 0.873821971016996
      GUSWT(IGAUS) = 0.050844906370207
C
      GUSTR(KGAUS,JGAUS) = 0.501426509658179
      GUSWT(KGAUS) = 0.116786275726379
C
   40 CONTINUE
C
      JGAUS = 6
C
C*    6 MORE GAUSS POINTS.
C
      DO 60 I = 1 , 3
C
      DO 50 L = 1 , 2
C
```

```
      JGAUS = JGAUS + 1
C
C*    USE POINTER IPT12 TO ASSIGN THE CURRENT GAUSS POINT COORDINATES
C*    AND WEIGHTS TO A POINT SUCH THAT FOR TYPE 1 SHELL ELEMENTS, THE
C*    MERIDIONAL COORDINATE XI INCREASES WITH GAUSS POINT NUMBER, IGAUS.
C
      IGAUS = IPT12(JGAUS)
C
C
      J = (I-2) * (I-3) * 0.5 * (2-L) * 2 + (I-2) * (I-3) * 0.5 *
     K (L-1) * 3 + (I-1) * (3-I) * (2-L) * 1 + (I-1) * (3-I) *
     K (L-1) * 3 + (I-1) * (I-2) * 0.5 * (2-L) * 1 +
     K (I-1) * (I-2) * 0.5 * (L-1) * 2
C
      K = 6 - I - J
C
      GUSTR(IGAUS,1) = COORD(I)
      GUSTR(IGAUS,2) = COORD(J)
      GUSTR(IGAUS,3) = COORD(K)
      GUSWT(IGAUS) = 0.082851075618374
C
   50 CONTINUE
   60 CONTINUE
C
   65 RETURN
      END


C -------------------    C N D S H L    -------------------
C
C
C
C
C
      SUBROUTINE CNDSHL (DSKF1, IELSH, MELSH, NKCTND, NDFCD, KSHEL,
     + KCINV, KNSHL, WKAREA)
C
      INTEGER DSKF1
C
      REAL KSHEL(38,38), KCINV(MELSH,2,2), KCDIV(12,12),
     + KCDND(12,36), KNSHL(36,36)
C
      DIMENSION  WKAREA(1), INDEX(19)
C
      DATA  INDEX / 14, 18, 22, 23, 24, 26, 30, 34, 35, 36, 37, 38,
     + 2, 6, 10, 11, 12, 37, 38 /
C
C
      IF (NDFCD . NE . 2)                          GO TO 65
C
```

```
2703  C*     IN THE FOLLOWING ND K CD STAND FOR NODAL AND CONDENSED,
2704  C*     RESPECTIVELY.
2705  C                                  K                          K
2706  C                                 1                            1
2707  C                                 1   KNDND          KNDCT    1
2708  C                                 1                            1
2709  C             KSHEL(38,38) =      1                            1
2710  C                                 1                            1
2711  C                                 1   KCDND          KCDIV    1
2712  C                                 1K                          K1
2713  C
2714  C*     PUT ROWS 37-38 AND COLUMNS 37-38 OF KSHEL IN KCDIV(2,2),
2715  C*     AND ROWS 37-38 COLUMNS 1-36 OF KSHEL IN KCNND(3,36).
2716  C
2717         DO 30 I = 1 , 2
2718  C
2719         DO 20 J = 1 , 2
2720         KCDIV(I,J) = KSHEL(36+I,36+J)
2721     20 CONTINUE
2722  C
2723         DO 30 K = 1 , 36
2724         KCDND(I,K) = KSHEL(36+I, K)
2725     30 CONTINUE
2726  C
2727  C***   OBTAIN THE INVERSE OF KCDIV. ON RETURN KCDIV IS REPLACED BY ITS
2728  C***   INVERSE.
2729  C
2730  C
2731         CALL INVERT (12, 2, KCDIV, WKAREA(1), WKAREA(2))
2732  C
2733  C*     STORE THE INVERSE OF THE CENTROIDAL STIFFNESS MATRIX FOR THE
2734  C*     CURRENT ELEMENT.
2735  C
2736         DO 40 J = 1 , 2
2737         DO 40 I = 1 , 2
2738         KCINV(IELSH,I,J) = KCDIV(I,J)
2739     40 CONTINUE
2740  C
2741  C*     CONDENSE OUT THE CENTROIDAL DEGREES OF FREEDOM FROM KSHEL. STORE
2742  C*     THE CONDENSED MATRIX IN KNSHL.
2743  C
2744         DO 60 J = 1 , 36
2745         DO 60 I = J , 36
2746         KNSHL(I,J) = KSHEL(I,J)
2747  C
2748         DO 50 K = 1 ,  2
2749         DO 50 L = 1 ,  2
2750  C
2751         KNSHL(I,J) = KNSHL(I,J) - KCDND(L,I) * KCDIV(L,K) * KCDND(K,J)
2752  C
2753     50 CONTINUE
```

```
C
      KNSHL(J,I) = KNSHL(I,J)
C
   60 CONTINUE
C
C*    PUT KCOND(2,36) ON DISK FILE DSKF1:
C
      WRITE (DSKF1,REC = NKCTND + IELSH)  KCOND
C
      GO TO 145
C
C
C
C*    SECOND DERIVATIVES SHOULD BE CONDENSED IN ADDITION TO THE
C*    CENTRODAL ONES.
C
   65 IBFR = (144 - 12 * NDFCD) / 3
C
C*    SET UP KCDIV(NDFCD,NDFCD).
C
      DO 90 J = 1 , NDFCD
C
C*    FIND THE COLUMN NUMBER OF ALL CONDENSED DEGREES OF FREEDOM.
C
      ICOL = INDEX(IBFR+J)
C
      DO 70 I = J , NDFCD
C
      IROW = INDEX(IBFR+I)
C
      KCDIV(I,J) = KSHEL(IROW,ICOL)
      KCDIV(J,I) = KCDIV(I,J)
C
      KSHEL(IROW,ICOL) = 0.0
      KSHEL(ICOL,IROW) = 0.0
C
   70 CONTINUE
C
   80 CONTINUE
C
C
C*    SET UP KCOND(NDFCD,36).
C
      DO 100 J = 1 , NDFCD
C
C*    FIND THE COLUMN NUMBER OF ALL CONDENSED DEGREES OF FREEDOM.
C
      ICOL = INDEX(IBFR+J)
C
      DO 90 I = 1 , 36
C
```

```
2805          KCOND(J,I) = KSHEL(I,ICOL)
2806    C
2807       90 CONTINUE
2808    C
2809      100 CONTINUE
2810    C
2811    C***  OBTAIN THE INVERSE OF KCDIV(NDFCD,NDFCD). ON RETURN KCDIV CONTAINS
2812    C***  THE INVERSE OF KCDIV.
2813    C
2814    C
2815          CALL INVERT (12, NDFCD, KCDIV, WKAREA(1), WKAREA(NDFCD+1))
2816    C
2817    C*    CONDENSE OUT THE DEGREES OF FREEDOM FROM KSHEL, AND STORE
2818    C*    THE CONDENSED MATRIX IN KNSHL(36,36).
2819    C
2820          DO 120 J = 1 , 36
2821          DO 120 I = J , 36
2822    C
2823          KNSHL(I,J) = KSHEL(I,J)
2824    C
2825          DO 110 K = 1 , NDFCD
2826          DO 110 L = 1 , NDFCD
2827    C
2828          KNSHL(I,J) = KNSHL(I,J) - KCOND(L,I) * KCDIV(L,K) * KCOND(K,J)
2829    C
2830      110 CONTINUE
2831    C
2832          KNSHL(J,I) = KNSHL(I,J)
2833    C
2834      120 CONTINUE
2835    C
2836    C
2837          NDF = NDFCD - 2
2838    C
2839    C
2840    C*    SET THE ROWS AND COLUMNS CORRESPONDING TO THE CONDENSED DEGREES
2841    C*    FREEDOM EQUAL TO ZERO, AND THE ASSOCIATED DIAGONAL TERMS EQUAL
2842    C*    TO ONE.
2843    C
2844          DO 140 I = 1 , NDF
2845    C
2846          J = INDEX(IBFR+I)
2847    C
2848          DO 130 K = 1 , 36
2849    C
2850          KNSHL(J,K) = KNSHL(K,J) = 0.0
2851    C
2852      130 CONTINUE
2853    C
2854          KNSHL(J,J) = 1.0
2855    C
```

```
2856     140 CONTINUE
2857   C
2858   C*     WRITE KCOND AND KCDIV ON DISK FILE DSKF1.
2859   C
2860         WRITE (DSKF1, REC = NKCTND + IRLSH)  KCOND, KCDIV
2861   C
2862     145 RETURN
2863         END
2864
2865   C
2866   C
2867   C
2868   C
2869   C
2870   C
2871   C
2872   C
2873   C
2874   C***   THIS SUBROUTINE PUTS THE NONZERO ENTRIES OF KSECT CORRESPONDING
2875   C***   TO A COSTRAINED DEGREE OF FREEDOM ON DISK DSKF1 (ROW BY ROW).
2876   C***   THEN IT CONSTRUCTS THE CORRECTIVE LOAD VECTOR CORLD. THEN
2877   C***   IT SETS THE NONDIAGONAL ENTRIES OF KSECT CORRESPONDING TO THE
2878   C***   PRESCRIBED DISPLACEMENTS EQUAL TO ZERO ( THE CORRESPONDING
2879   C***   DIAGONAL ENTRIES ARE SET TO ONE ). THEN IT CONDENSES OUT THE
2880   C***   INTERNAL STIFFNESS MATRIX.
2881   C
2882         SUBROUTINE MDFKSC (LROF1, MPRDS, MDFIF, NNDEX, NDFBL, NDFEX,
2883        + NDFIN, NDFSC, NDFSL, NBEG, NEND, IOP, IBX, PDCHEK, SINGAB, ICOND,
2884        + IAFILE, IREF, DETCF, NODPD, PRSDS, RSDOF, NREST, NDORD, SEQND,
2885        + LD, IPNTR, ICONN, CORLD, KSECT, BRHS, X)
2886   C
2887         INTEGER  DSKF1, DSKF2, SLRAC, SLBKL, PLDSP, PLSTS, NODPD(1),
2888        + RSDOF(MPRDS,1), NREST(1), NDORD(1), SEQND(1), LD(1), IPNTR(1),
2889        + ICONN(1)
2890   C
2891         REAL  PRSDS(MPRDS,1), KSECT(NDFSL)
2892   C
2893         DIMENSION  CORLD(NDFSC), BRHS(NDFEX), X(1)
2894   C
2895         LOGICAL SINGAB, PDCHEK
2896   C
2897         COMMON  STRTYP, AUTGN, SLDSP, SLRAC, SLSTS, SLBKL,
2898        + PLSTR, PLMSH, PLDSP, PLSTS, PLMSP,
2899        + MSECT, MSTRN, MNDSR, MRING, MNDRN, MGAUS,
2900        + NNDSH, NELSH, NMTSH, NSTRN, NNDSR, NELSR, NMTSR, NIMSR,
2901        + NELISR, NMTISR, NRING, NNDRN, NELRN, NMTRN, NIMRN, NELIRN,
2902        + NMTIRN, NBRAC, NELBR, NMTBR, NGAUS, NCRSM, NPRDS, NLDCA,
2903        + NSECT, NRWEL, NRWND, NDFND, NNDE1, NNDE2, NNDIN,
2904        + NNTSH, NBTSH, NGTSH, NSTSSH, NKCTND, NNTSR, NBTSR, NGTSR,
2905        + NTFSR, NSTSSR, NNTISR, NBTISR, NGTISR, NTFISR, NSTSIS, NNTRN,
2906        + NBTRN, NGTRN, NTFRN, NSTSRN, NNTIRN, NBTIRN, NGTIRN, NTFIRN,
```

```
     + NSTSIR, NTFBR, NSTSBR, NKRWPD, NCRLD, NFRCN, NABIF, NASCT,
     + NACNT, NFBSC, NRACT, NKSCT, NKSCF, NKKGW, NKGSC, NKGICL, NKBNIF,
     + NKGBNI, NEGVC, NKKGCN, NKGCN, NMSND, NMSCT, NKOEE, NKGOEE,
     + CSOCL, CSDMC, MCCOD, ILOCA, LCNLD, LDTLD, LZNTP, LZNSL,
     + DSKF1, DSKF2, IN, IO,
     + FEED, RTOP, RBAS, HITE, ELEV, THATAD,
     + TITLE(30), LOCAS(2), NCNLD(2), NDTLD(2), NZNTP(2), NZNSL(2),
     + RATIO(14), INDEX(45), NDCON(200), NDEX1(15), NDEX2(15),
     + NDINT(15), INTND(36), WKAREA(192)
C
C*    INITIALIZATIONS.
C
      IKOEE = 0
C
C*    IF NO DISPLACEMENTS ARE PRESCRIBED SKIP.
C
      IF (NPRDS . EQ . 0)                              GO TO 85
C
C*    IF CALCULATION OF REACTIONS IS REQUESTED, SKIP.
C
      IF (SLRAC . NE . 3HYES)                          GO TO 75
C
C
C***  PUT THE NONZERO ENTRIES OF KSECT CORRESPONDING TO A CONSTRAINED
C***  DEGREE OF FREEDOM ON DISK DSKF1 (ROW BY ROW).
C
      CALL ROWKPD (DSKF1, MPRDS, NNDSH, NDFND, NPRDS, NRWND,
     + NDFSC, NKRWPD, INDEX, NDCON, NODPD, NREST, RSDOF, NDEX1, NDEX2,
     + LD, KSECT, NDORD, SEQND, IPNTR, ICONN, CORLD)
C
C***  CONSTRUCT THE CORRECTIVE LOAD VECTOR, AND WRITE IT ON DISK FILE
C***  TCRLD. ENTRY EQUAL TO THE NEGATIVE OF THE SUM OF (K,IJ * U,J),
C***  WHERE J IS THE RESTRAINED DEGREE OF FREEDOM. 'I' IS
C***  ASSOCIATED WITH THE FREE DEGREE OF FREEDOM. ENTRY  'J' OF CORLD
C***  CONTAINS THE J-TH RESTRAINED DEGREE OF FREEDOM. ENTRIES OF CORLD
C***  CORRESPONDING TO THE SYMMETRIC COUNTERPART OF AN EXTERNAL
C***  RESTRAINED NODE (I.E., EXTERNAL NODE ON THE OTHER INTERFACE OF
C***  THE SECTOR, LOCATED SYMMETRICALLY TO A GIVEN NODE.) ARE ALSO
C***  ASSUMED TO HAVE THE SAME PRESCRIBED DISPLACEMENTS AS THE
C***  RESTRAINED NODE.
C
   75 CALL CORCLD (DSKF1, MPRDS, NNDSH, NPRDS, NRWND, NDFND,
     + NDFSC, NCRLD, NODPD, NREST, RSDOF, PRSDS, INDEX, NDCON,
     + NDEX1, NDEX2, INTND, NDORD, LD, KSECT, CORLD)
C
C***  SET THE NONDIAGONAL ENTRIES OF KSECT CORRESPONDING TO THE
C***  PRESCRIBED DISPLACEMENTS EQUAL TO ZERO. THE CORRESPONDING
C***  DIAGONAL ENTRIES ARE SET EQUAL TO 1.0. IF THE NODE AT WHICH A
C***  DISPLACEMENT IS PRESCRIBED IS AN EXTERNAL 1 (2) TYPE NODE, CARRY
C***  OUT THE SAME PROCESS FOR ITS SYMMETRICAL COUNTERPART LOCATED
C***  ON THE EXTERNAL 2 (1) INTERFACE.
```

```
C
      CALL MDFKPD (1.0, MPROS, NNOSH, NDFNO, NPROS, NNDE1,
     + NDEX1, NDEX2, INTND, INDEX, NOCON, NOOPD, NREST, RSDOF,
     + LD, KSECT, NOORD, SEQND, IPNTR, ICONN)
C
C
C***  IF THE NUMBER OF LOADING CASES IS GREATER THAN 1, AND PLOTTING
C***  OF THE DISPLACEMENTS AND / OR STRESSES HAS BEEN REQUESTED, WRITE
C***  THE CONVENTIONAL STIFFNESS MATRIX ON DISK FILE DSKF1, BECAUSE
C***  KSECT IS OVERWRITTEN IN THE PLOTTING SUBROUTINES.
C
   85 IF (NLOCA . LE . 1)                          GO TO 105
C
      IF (PLOSP . EQ . 3HYES . OR . PLSTS . EQ . 3HYES) GO TO 115
C
C***  IF BUKLING SOLUTION HAS BEEN REQUESTED, WRITE THE SECTOR
C***  STIFFNESS MATRIX ON DISK FILE DSKF1.
C
  105 IF (SLBKL . NE . 3HYES)                      GO TO 150
C
  115 CALL WRTKSC (DSKF1, NKSCT, 0, LRDF1, NDFSL, KSECT)
C
C*    IF THERE ARE NO INTERNAL NODES, NO FACTORIZATION AND STATIC
C*    CONDENSATION IS NEEDED.
C
  150 IF (NNDIN . EQ . 0)                          GO TO 550
C
C*    SET THE ENTRIES OF LD CORRESPONDING TO THE EXTERNAL NODES,
C*    WHICH ARE NOT TO BE FACTORIZED, EQUAL TO THEIR NEGATIVE.
C
C
C*    CONSIDER ALL NODES OF THE SECTOR.
C
      DO 220 INDSH = 1 , NNDSH
C
      IF (INTND(INDSH) . EQ . 3HYES)               GO TO 220
C
      IORD = NOORD(INDSH)
      LSTR = 12 * (IORD - 1) + 2
      LFIN = LSTR + 11
C
      DO 200 I = LSTR , LFIN
      LD(I) = - LD(I)
  200 CONTINUE
C
  220 CONTINUE
C
C
C                CONDENSE OUT THE INTERNAL STIFFNESS MATRIX.
```

```
C***  FACTORIZE THE SKYLINE-STORED VECTOR KSECT BYE SKYFAC.
      WRITE(6,*) ' NBEG=',NBEG,' NEND=',N,' NDFSC=',NDFSC,
     + ' LD',LD,' SINGAB=',SINGAB,' PDCHEK=',PDCHEK,' ICOND=',ICOND,
     + ' IAFILE=',IAFILE,' ACOND',ACOND,' DETCF=',DETCF,' NEGEIG=',
     + NEGEIG,' IFAIL=',IFAIL
C
C
      CALL SKYFAC (KSECT, NBEG, NEND, NDFSC, LD, CORLD, SINGAB,
     + PDCHEK, ICOND, IAFILE, ACOND, DETCF, IDETEX, NEGEIG, IFAIL)
C
C
      PRINT*, 'IFAIL=',IFAIL,'  NEGEIG=',NEGEIG
      WRITE(6,*) 'NEGEIG=',NEGEIG,'IFAIL=',IFAIL
C
      IF (IFAIL . EQ . 0)            GO TO 250
C
      WRITE(6,*) 'EXECUTION WAS DISCONTINUED BECAUSE FACTORIZATION OF '
     + ,'KSCET HAS FAILED'
      PRINT*,    'EXECUTION WAS DISCONTINUED BECAUSE FACTORIZATION OF '
     + ,'KSCET HAS FAILED'
C
      STOP
C
C*    CONSIDER ALL EXTERNAL 2 AND EXTERNAL 1 NODES, RESPECTIVELY.
C
  250 DO 540 IEX = 1 , 2
C
      DO 500 IRWND = 1 , NRWND
C*    FIND THE CURRENT EXTERNAL NODE.
C
      INDRW = NRWND - IRWND + 1
C
      NDEXT = (2 - IEX) * NDEX2(INDRW) + (IEX - 1) * NDEX1(INDRW)
C
C*    FIND THE ORDER OF THE CURRENT EXTERNAL NODE.
C
      IOREX = NDORD(NDEXT)
C
C*    CONSIDER ALL DEGREES OF FREEDOM OF THE CURRENT EXTERNAL NODE.
C
      DO 300 IDFEX = 1 , NDFND
C
      IROW = (IOREX-1) * NDFND + IDFEX
C
C***  CONSTRUCT ROW NUMBER IROW OF KEI (COLUMN NUMBER IROW OF KIE)
C***  AND PUT IT IN BRHS. ON RETURN THE ENTRIES OF BRHS CORRESPONDING
C***  TO THE EXTERNAL NODES CONTAIN ZEROS.
C
      CALL KEIBRS (IOREX, IDFEX, IROW, NNDSH, NDFSC, NDFND,
     + NDFEX, NDFIN, NNDEX, NNDIN, NDINT, INTND, NDORD, SEQND, INDEX,
```

```
3060       + NDCON, LD, KSECT, BRHS)
3061  C
3062  C***  TAKE THE NONZERO ENTRIES OF BRHS, WHICH CONTAIN THE IROW-TH
3063  C***  COLUMN OF KIE, AND ON RETURN PUT THE PRODUCT  KEI * KII-INV * KIE
3064  C***  IN THE ENTRIES OF BRHS CORRESPONDING TO THE EXTERNAL NODES. AS A
3065  C***  REQUIREMENT FOR THIS OPERATION BY SKYSOL, THE EXTERNAL ENTRIES
3066  C***  OF BRHS NOW CONTAIN ZEROS. (IOP = 0; IBX = 1)
3067  C
3068        CALL SKYSOL (KSECT, NDFSC, LD, IOP, IBX, BRHS, X,
3069       K IREF ,IAFILE, WKAREA, DELTA)
3070  C
3071  C***  ARRANGE ALL ENTRIES OF VECTOR BRHS CORRESPONDING TO THE
3072  C***  INTERNAL NODES, SO THAT, ON RETURN, ITS TOP NDFEX ENTRIES
3073  C***  ARE ASSOCIATED WITH THE EXTERNAL 2 AND EXTERNAL 1 NODES,
3074  C***  RESPECTIVELY.
3075  C
3076        CALL ARRANG (NRWND, NDFSC, NDEX1, NDEX2, NDORD, X, BRHS)
3077  C
3078  C
3079        IKOEE = IKOEE + 1
3080  C
3081  C*    PUT THE TOP NDFEX ENTRIES OF BRHS, WHICH IS THE ICOL-TH COLUMN OF
3082  C*    ( KEI * KII-INV * KIE ) ON DISK FILE DSKF2.
3083  C
3084        WRITE (DSKF2, REC=NKOEE+IKOEE)  BRHS
3085  C
3086     300 CONTINUE
3087  C
3088     500 CONTINUE
3089  C
3090     540 CONTINUE
3091  C
3092  C***  IF THE NUMBER OF LOADING CASES IS GREATER THAN 1, AND PLOTTING
3093  C***  OF THE DISPLACEMENTS AND / OR STRESSES HAS BEEN REQUESTED, WRITE
3094  C***  THE FACTORIZED CONVENTIONAL STIFFNESS MATRIX ON DISK FILE DSKF1,
3095  C***  BECAUSE KSECT IS OVERWRITTEN IN THE PLOTTING SUBROUTINES.
3096  C
3097        IF (NLDCA . LE . 1)                              GO TO 545
3098  C
3099        IF (PLDSP . EQ . 3HYES . OR . PLSTS . EQ . 3HYES) GO TO 547
3100  C
3101  C*    IF BUCKLING SOLUTION HAS BEEN REQUESTED, WRITE THE FACTORIZED
3102  C*    STIFFNESS MATRIX ON DISK FILE DSKF1.
3103  C
3104     545 IF (SLBKL . NE . 3HYES)                         GO TO 550
3105  C
3106     547 CALL WRTKSC (DSKF1, NKSCF, 0, LRDF1, NDFSL, KSECT)
3107  C
3108  C
3109     550 RETURN
3110        END
```

```
C
C
C
C -----------------------------------------------------------------
C
C
C***   THIS SUBROUTINE PUTS THE NONZERO ENTRIES OF KSECT CORRESPONDING
C***   TO A CONSTRAINED DEGREE OF FREEDOM ON DISK (ROW BY ROW).
C***   THE ENTRIES ARE SAVED ON DISK FILE DSKF1 TO BE USED LATER FOR
C***   CALCULATING REACTIONS.
C***   IF THE RESTRAINED NODE IS AN EXTERNAL 1 TYPE (EXTERNAL 2 TYPE)
C***   NODE, CARRY OUT THE SAME PROCESS FOR ITS SYMMETRICAL
C***   COUNTERPART, LOCATED ON THE EXTERNAL 2 (EXTERNAL 1) INTERFACE.
C
      SUBROUTINE ROWKPD (DSKF1, MPRDS, NNDSH, NDFND, NPRDS, NRWND,
     + NDFSC, NKRWPD, INDEX, NDCON, NDDPD, NREST, RSDOF, NDEX1, NDEX2,
     + LD, KSECT, NDORD, SEQND, IPNTR, ICONN, KRWPD)
C
      INTEGER  DSKF1, RSDOF(MPRDS,1), SEQND(1)
C
      REAL  KSECT(1), KRWPD(NDFSC)
C
      DIMENSION  NDDPD(1), NREST(1), NDEX1(1), NDEX2(1), LD(1),
     + NDORD(1), INDEX(1), NDCON(1), IPNTR(1), ICONN(1)
C
      LOGICAL  INTER
C
C*    INITIALIZATION.
C
      IKRWPD = 0
C
C*    CONSIDER EVERY NODE WITH A PRESCRIBED DISPLACEMENT.
C
      DO 130 IPRDS = 1 , NPRDS
C
      NDPD = NDDPD (IPRDS)
C
      INTER = . TRUE .
C
C*    FIND OUT IF NDPD IS AN INTERNAL OR AN EXTERNAL NODE. IF AN
C*    EXTERNAL NODE, FIND NDSM, ITS SYMMETRICAL COUNTERPART.
C
C*    CONSIDER ALL EXTERNAL NODES.
C
      DO 20 IRWND = 1 , NRWND
C
      IF ( NDPD . NE . NDEX1(IRWND) )              GO TO 15
      NDSM = NDEX2(IRWND)
      INTER = . FALSE .
C
      GO TO 25
```

```
C
   15 IF ( NDPD . NE . NDEX2(IRWND) )                GO TO 20
      NDSM = NDEX1(IRWND)
      INTER = . FALSE .
C
      GO TO 25
C
   20 CONTINUE
C
C*    IN THE FOLLOWING DO LOOP, THE INDEX ITIME ASSUMES VALUES OF 1
C*    AND 2 WHEN NDPD IS AN EXTERNAL ( 1 OR 2 ) NODE. WHEN  ITIME = 2
C*    NDPD IS SET EQUAL TO THE SYMMETRICAL COUNTERPART OF NODPD(IPRDS).
C*    I.E., THE EXTERNAL NODE LOCATED ON THE OTHER END OF THE ROW OF
C*    NODES ON WHICH NODPD(IPRDS) IS LOCATED. WHEN NDPD IS AN INTERNAL
C*    NODE, THE LOOP IS EXITED BEFORE ITIME ASSUMES VALUE OF 2.
C
   25 DO 120 ITIME = 1 , 2
C
C
C*    WHEN ITIME IS EQUAL TO 2 SET NDPD EQUAL TO THE SYMMETRICAL
C*    COUNTERPART OF THE CURRENT  NDPD ( = NODPD(IPRDS) ) AND REPEAT
C*    THE PROCESS.
C
      IF (ITIME . EQ . 2)                             NDPD = NDSM
C
      NRST = NREST (IPRDS)
      IORPD = NDORD (NDPD)
      IPOS = INDEX (NDPD)
      NND = NDCON (IPOS)
      IBEG = IPOS + 1
      IEND = IPOS + NND
C*    FIND 'LWSTOR' THE LOWEST ORDER AMONG THE NODES CONNECTED TO NDPD
C*    AND LOWER IN ORDER.
      KPOS = IPNTR (NDPD)
      NCND = ICONN (KPOS)
      IF (NCND . EQ . 0)                              GO TO 45
      ND1 = ICONN (KPOS+1)
      LWSTOR = NDORD (ND1)
C
      GO TO 55
C
   45 LWSTOR = IORPD
C
C
   55 IF (NRST . EQ . 0)                              GO TO 115
C
C
C
C*    CONSIDER EVERY RESTRAINED DOF AT NDPD.
C
      DO 110 IRST = 1 , NRST
```

```
C
C
C*    INITIALIZE KRWPD TO ZERO FOR EACH RESTRAINT.
C
      DO 60 I = 1 , NDFSC
      KRWPD(I) = 0.0
   60 CONTINUE
C
      IDOF = RSDOF (IPRDS,IRST)
      ILD = (IDRPD-1) * NDFND + IDOF + 1
      ICOL = ILD - 1
C*    FOR EACH IRST KRWPD WILL CONTAIN THE NONZERO ENTRIES OF ROW
C*    NUMBER 'ICOL' OF KSECT. A SET OF NDFND ENTRIES CORRESPOND TO A
C*    NODE CONNECTED TO NDPD, AND ARE LOCATED IN THE ORDER GIVEN BY
C*    NDCON. THE NDFND ENTRIES CORRESPONDING TO NDPD-NDPD BLOCK COME
C*    LAST. CONSIDER ALL NODES CONNECTED TO NDPD.
C
      DO 90 JPOS = IBEG , IEND
C
      NDCN = NDCON (JPOS)
C*    FIND THE ORDER OF 'NDCN' IN THE STIFFNESS MATRIX OF THE SECTOR.
      IDRCN = NDORD (NDCN)
C*    FIND THE ENTRY JUST BEFORE THOSE OF 'NDCN' IN KRWPD.
      IKRW = 12 * (IDRCN - 1)
C*    IF THE ORDER OF THE CONNECTED NODE IS LESS THAN THAT OF NDPD,
C*    ENTRIES ARE READ COLUMN BY COLUMN FROM KSECT.
      IF (IDRCN.GT.IDRPD)       GO TO 75
C*    IDRCN LESS THAN IDRPD.
C*    'LAST' IS THE POSITION OF THE LAST DIAGONAL ENTRY OF THE COLUMN
C*    JUST BEFORE ICOL.
      LAST = LD (ICOL)
C
      DO 70 IDFRW = 1 , NDFND
C
      IKSC = LAST + (IDRCN - LWSTOR) * NDFND + IDFRW
      KRWPD(IKRW+IDFRW) = KSECT(IKSC)
C
   70 CONTINUE
C
      GO TO 90
C
C*    IDRCN GREATER THAN IDRPD
C*    NOW CONSIDER ROW NUMBER ICOL OF KSECT AND PUT ITS NONZERO ENTRIES
C*    IN KRWPD.
C*    'MINORD' IS THE MINIMUM OF THE ORDERS OF THE NODES CONNECTED TO
C*    NDCN.
   75 NOD1 = ICONN (IPNTR(NDCN) + 1)
      MINORD = NDORD(NOD1)
C
      DO 80 IDFCL = 1 , NDFND
C
```

```
C*     'LAST' IS THE POSITION OF THE LAST (DIAGONAL) ENTRY OF THE
C*     COLUMN JUST BEFORE THE ONE IN WHICH THE PRESENT ENTRY IS BEING
C*     CONSIDERED, IN KSECT.
       ILD = (IORCN - 1) * NDFND + IDFCL
       LAST = LD (ILD)
       IKSC = LAST + (IORPD - MINORD) * NDFND + IDOF
       KRWPD(IKRW+IDFCL) = KSECT(IKSC)
C
   80 CONTINUE
C
C
   90 CONTINUE
C
C*     FINALLY PUT THE ENTRIES OF ROW 'ICOL', OF KSECT, CORRESPONDING
C*     TO BLOCK NDPD-NDPD IN THE LAST NDFND ENTRIES OF KRWPD.
C*     'LSTCL' IS THE POSITION OF THE LAST COLUMN OF THE PREVIOUS
C*     NODE, IN KSECT.
       LSTCL = (IORPD - 1) * NDFND
C*     LAST IS THE POSITION OF THE LAST DIAGONAL ENTRY ON COLUMN ICOL-1
       LAST = LD (ICOL)
C*     FIND THE ENTRY JUST BEFORE THOSE OF 'NDPD' IN KRWPD.
       IKRW = 12 * (IORPD -.1)
C
       DO 100 IDFPD = 1 , NDFND
C
       JJ = LSTCL + IDFPD
       IF (JJ . GT . ICOL)       GO TO 95
C*     READ COLUMN-WISE.
       IKSC = LAST + (IORPD-LWSTOR) * NDFND + IDFPD
       KRWPD(IKRW+IDFPD) = KSECT(IKSC)
C
       GO TO 100
C
C*     READ ROW-WISE.
C*     'LAST' IS THE LAST (DIAGONAL) POSITION OF COLUMN  JJ-1.
    95 LAST = LD (JJ)
       IKSC = LAST + (IORPD - LWSTOR) * NDFND + ICOL - LSTCL
       KRWPD(IKRW+IDFPD) = KSECT(IKSC)
C
  100 CONTINUE
C
       IKRWPD = IKRWPD + 1
C
C*     WRITE THE STIFFNESS COEFFICIENTS CORRESPONDING TO THE CURRENT
C*     RESTRAINED DEGREE OF FREEDOM ON DISK FILE DSKF1.
C
       WRITE (DSKF1, REC=NKRWPD+IKRWPD)  KRWPD
C
  110 CONTINUE
C
  115 IF (INTER)                                  GO TO 130
```

```
C
  120 CONTINUE
C
  130 CONTINUE
C
      RETURN
      END
C
C
C
C
C-----------------------------------------------------------------
C
C
C***  THIS SUBRORTINE CONSTRUCTS THE CORRECTIVE LOAD VECTOR, AND
C***  WRITES IT ON DISK FILE TCRLD. ENTRY 'I' OF CORLD
C***  IS EQUAL TO THE NEGATIVE OF THE SUM OF (K,IJ * U,J)
C***  WHERE J IS THE RESTRAINED DOF. 'I' IS ASSOCIATED WITH THE FREE
C***  DEGREES OF FREEDOM. ENTRY 'J' OF CORLD CONTAINS THE J-TH
C***  RESTRAINED DEGREE OF FREEDOM.

      SUBROUTINE CORCLD (DSKF1, MPRDS, NNDSH, NPRDS, NRWND, NDFND,
     + NDFSC, NCRLD, NODPD, NREST, RSDOF, PRSDS, INDEX, NDCON,
     + NDEX1, NDEX2, INTND, NDORD, LD, KSECT, CORLD)
C
      REAL KSECT(1), CORLD(NDFSC), PRSDS(MPRDS,1)
C
      INTEGER  DSKF1, NODPD(1), NREST(1), RSDOF(MPRDS,1),
     + INDEX(1), NDCON(1), NDEX1(1), NDEX2(1), INTND(1), NDORD(1), LD(1)
C
      LOGICAL  NDFREE
C
C*    CONSIDER ALL NODES AND THEIR NDFND DEGREES OF FREEDOM.
C
      DO 110 INDSH = 1 , NNDSH
C
      IORND = NDORD(INDSH)
      NDFREE = .TRUE.
      IPOS = INDEX(INDSH)
      NCND = NDCON(IPOS)
      IBEG = IPOS + 1
      IEND = IPOS + NCND
C
C*    SEE IF THE CURRENT NODE IS RESTRAINED.
C
      DO 20 IPRDS = 1 , NPRDS
C
      NDPD = NODPD(IPRDS)
C
C*    IF NDPD IS AN EXTERNAL NODE, FIND ITS SYMMETRICAL COUNTERPART,
```

```
C*     NDSYM.
C
       IF (INTND(NDPD) . NE . 3HYES)                    GO TO 11
C*     NDPD IS INTERNAL.
       IF ( INDSH . NE . NDPD )                         GO TO 20
       NDFREE = . FALSE .
       KPROS = IPROS
       GO TO 23
C
C*     NDPD BEING AN EXTERNAL NODE, FIND ITS SYMMETRICAL COUNTERPART,
C*     NDSYM.
C
   11  DO 14 IRWND = 1 , NRWND
C
       IF ( NDPD . NE . NDEX1(IRWND) )                  GO TO 13
       NDSYM = NDEX2(IRWND)
       GO TO 15
C
   13  IF ( NDPD . NE . NDEX2(IRWND) )                  GO TO 14
       NDSYM = NDEX1(IRWND)
       GO TO 15
C
   14  CONTINUE
C
   15  IF ( INDSH . NE . NDPD . AND . INDSH . NE . NDSYM ) GO TO 20
       NDFREE = .FALSE.
       KPROS = IPROS
C
       GO TO 23
C
   20  CONTINUE
C
C*     NOW CONSIDER ALL FREE DEGREES OF FREEDOM OF NODE INDSH.
C
   23  DO 100 IDFND = 1 , NDFND
C
       IROW = (IDRND - 1) * NDFND + IDFND
C
C*     INITIALIZE THE CURRENT ROW OF CORLD TO ZERO.
       CORLD(IROW) = 0.0
       IF (NDFREE)                                      GO TO 35
C*     NOW THAT INDSH IS RESTRAINED (OR PARTIALLY SO) CHECK IF IDFND IS
C*     FREE.
C
       NRST = NREST (KPROS)
C
C
       IF (NRST . EQ . 0)                               GO TO 35
C
       DO 30 IRST = 1 , NRST
C
```

```
      IDOF = RSDOF (KPROS,IRST)
C
C*    IF IDFND IS RESTRAINED, CONSIDER THE NEXT IDFND.
C
      IF (IDFND - IDOF)                                  30, 25, 30
C
C*    PUT THE PRESCRIBED DISPLACEMENTS ASSOCIATED WITH IDOF IN CORLD.
C
   25 CORLD (IROW) = PRSDS (KPROS,IDOF)
C
      GO TO 100
C
   30 CONTINUE
C
C*    NOW WE KNOW IDFND IS A FREE DEGREE OF FREEDOM.
C
C*    CONSIDER ALL NODES WITH RESTRAINED DEGREES OF FREEDOM CONNECTED
C*    TO INDSH.
C
   35 DO 90 IPROS = 1 , NPROS
C
      NDPD = NDDPD (IPROS)
      IORPD = NDORD (NDPD)
C
C*    IF NDPD IS AN EXTERNAL NODE, REPEAT THE FOLLOWING FOR ITS
C*    SYMMETRIC COUNTERPART.
C
      DO 85 ITIME = 1 , 2
C
      IF ( ITIME . EQ . 1 )                              GO TO 47
C
C*    IF NDPD IS AN INTERNAL NODE, SKIP.
C
      IF (INTND(NDPD) . EQ . 3HYES)                      GO TO 85
C
C
C*    NDPD BEING AN EXTERNAL NODE, FIND ITS SYMMETRIC COUNTERPART,
C*    NDSYM.
C
      DO 44 IRWND = 1 , NRWND
C
      IF ( NDPD . NE . NDEX1(IRWND) )                    GO TO 42
      NDSYM = NDEX2(IRWND)
      GO TO 45
C
   42 IF ( NDPD . NE . NDEX2(IRWND) )                    GO TO 44
      NDSYM = NDEX1(IRWND)
      GO TO 45
C
   44 CONTINUE
C
```

```
      45 NDPD = NDSYM
         IORPD = NDORD(NDPD)
C
      47 DO 60 JPOS = ISEG , IEND
C
         NDCN = NDCON (JPOS)
         NRST = NREST(IPRDS)
C
         IF (NDPD . NE . NDCN)                          GO TO 60
C
C
         IF (NRST . EQ . 0)                             GO TO 60
C
C*       NOW WE KNOW INDSH IS CONNECTED TO NDPD.
C
C*       CALCULATE THE NEGATIVE OF THE SUM OF THE PRODUCTS OF THE
C*       PRESCRIBED DEGREES OF FREEDOM  OF NDPD'S AND THE CORRESPONDING
C*       ENTRIES OF IROW AND PUT IT IN THE CURRENT ROW OF CORLD.
C
         DO 50 IRST = 1 , NRST
C
         IDOF = RSDOF (IPRDS,IRST)
         ICOL = (IORPD - 1) * NDFND + IDOF
         IF (IROW. GT. ICOL)                    GO TO 48
C*       IDIAG IS THE POSITION OF THE DIAGONAL ENTRY ICOL-ICOL IN KSECT.
         IDIAG = LD (ICOL + 1)
         IKSC = IDIAG - (ICOL - IROW)
         CORLD (IROW) = CORLD (IROW) - KSECT (IKSC) * PRSDS (IPRDS,IDOF)
         GO TO 50
C*       IROW GREATER THAN ICOL
C*       IDIAG IS THE POSITION OF THE DIAGONAL ENTRY IROW-IROW IN KSECT.
      48 IDIAG = LD (IROW + 1)
         IKSC = IDIAG - (IROW - ICOL)
         CORLD (IROW) = CORLD (IROW) - KSECT (IKSC) * PRSDS (IPRDS,IDOF)
C
      50 CONTINUE
C
      60 CONTINUE
C
C*       NOW CALCULATE THE SUM OF (K,IJ * U,J) WHEN 'I' CORRESPONDS TO
C*       ONE OF THE FREE DOF'S OF THE DIAGONAL BLOCK NDPD-NDPD.
C
         IF (IORPD . NE . IORND)                        GO TO 85
C
C
         IF (NRST . EQ . 0)                             GO TO 85
C
         DO 80 IRST = 1 , NRST
C
         IDOF = RSDOF (IPRDS,IRST)
         ICOL = (IORPD - 1) * NDFND + IDOF
```

```
      IF (IROW. GT. ICOL)                    GO TO 70
C*    IDIAG IS THE POSITION OF THE DIAGONAL ENTRY ICOL-ICOL IN KSECT.
      IDIAG = LD (ICOL + 1)
      IKSC = IDIAG - (ICOL - IROW)
      CORLD (IROW) = CORLD (IROW) - KSECT (IKSC) * PRSDS (IPRDS,IDOF)
      GO TO 80
C*    IROW GREATER THAN ICOL
C*    IDIAG IS THE POSITION OF THE DIAGONAL ENTRY IROW-IROW IN KSECT.
   70 IDIAG = LD (IROW + 1)
      IKSC = IDIAG - (IROW - ICOL)
      CORLD (IROW) = CORLD (IROW) - KSECT (IKSC) * PRSDS (IPROS,IDOF)
C
   80 CONTINUE
C
   85 CONTINUE
C
   90 CONTINUE
C
  100 CONTINUE
C
C
  110 CONTINUE
C
C
C*    WRITE THE CORRECTIVE LOAD VECTOR, CORLD(NDFSC), ON DISK FILE
C*    DSKF1.
C
      WRITE(DSKF1, REC=NCRLD+1)  CORLD
C
      RETURN
      END
C
C
C
C
C-----------------------------------------------------------------------
C
C
C***  THIS SUBROUTINE SETS THE NONDIAGONAL ENTRIES OF KSECT
C***  CORRESPONDING TO THE PRESCRIBED DISPLACEMENTS EQUAL TO ZERO.
C***  THE CORRESPONDING DIAGONAL ENTRIES ARE SET TO 1.0. IF THE NODE
C***  AT WHICH A DISPLACEMENT IS PRESCRIBED IS AN EXTERNAL 1 (2) TYPE,
C***  IT CARRIES OUT THE SAME PROCESS FOR ITS SYMMETRICAL COUNTERPART
C***  LOCATED ON THE EXTERNAL 2 (1) INTERFACE.
C***  ZO MAY ASSUME VALUES OF EITHER ZERO (0.0) OR ONE (1.0) TO BE
C***  PLACED ON THE DIAGONAL ENTRIES OF KSECT.
C
      SUBROUTINE MODAP (LD, NPROS, NPDOF, NDFNO, NPROS, NNOE1,
     + NLOAD, NLOAD, 1.0, 0.0, 0.0, 0.0, NPST, RSDOF, LO,
     + KSECT, NDOFSC, 0.0D0, 0.0D0, DSKF1)
C
```



```
C
C*    FOR EACH IRST 'STORE' WILL CONTAIN THE NONZERO ENTRIES OF ROW
C*    NUMBER 'ICOL' OF KSECT. EVERY NOFND ENTRIES CORRESPOND TO A NODE
C*    CONNECTED TO NOPO, AND ARE LOCATED IN THE ORDER GIVEN BY NOCON.
C*    THE NOFND ENTRIES CORRESPONDING TO NOPO-NOPO BLOCK COME LAST.
C*    CONSIDER ALL NODES CONNECTED TO NOPO.
C
      DO 30 JPOS = IBEG , IEND
C
      NOCN = NOCON (JPOS)
      IORCN = NOORD (NOCN)
C
C*    IF THE ORDER OF THE CONNECTED NODE IS LESS THAN THAT OF NOPO,
C*    ENTRIES ARE READ COLUMN BY COLUMN FROM KSECT.
C
      IF (IORCN . GT . IORPO)                          GO TO 15
C
C*    IORCN LESS THAN IORPO .
C*    'LAST' IS THE POSITION OF THE LAST DIAGONAL ENTRY OF THE COLUMN
C*    JUST BEFORE ICOL, IN ORDER.
C
      LAST = IABS( LO (ICOL) )
C
      DO 10 IOFRW = 1 , NOFND
C
      IKSC = LAST + (IORCN - LWSTOR) * NOFND + IOFRW
      KSECT (IKSC) = 0.0
C
   10 CONTINUE
C
      GO TO 30
C
C*    IORCN GREATER THAN IORPO
C*    NOW CONSIDER ROW NUMBER ICOL OF KSECT AND PUT ITS NONZERO ENTRIES
C*    IN STORE.
C*    'MINORD' IS THE MINIMUM OF THE ORDERS OF THE NODES CONNECTED TO
C*    NOCN.
C
   15 NOD1 = ICONN (IPNTR(NOCN) + 1)
C
      MINORD = NOORD(NOD1)
C
      DO 20 IOFCL = 1 , NOFND
C
C*    'LAST' IS THE POSITION OF THE LAST (DIAGONAL) ENTRY OF THE
C*    COLUMN JUST BEFORE THE ONE IN WHICH THE PRESENT ENTRY IS BEING
C*    CONSIDERED, IN KSECT.
C
      ILO = (IORCN - 1) * NOFND + IOFCL
      LAST = IABS( LO (ILO) )
      IKSC = LAST + (IORPO - MINORD) * NOFND + IOOF
```

```
3672          KSECT (IKSC) = 0.0
3673    C
3674       20 CONTINUE
3675    C
3676       30 CONTINUE
3677    C
3678    C*    FINALLY PUT THE ENTRIES OF ROW 'ICOL', OF KSECT, CORRESPONDING
3679    C*    TO BLOCK NDPD-NDPD IN THE LAST NDFND ENTRIES OF STORE.
3680    C*    'LSTCL' IS THE POSITION OF THE LAST COLUMN OF THE PREVIOUS
3681    C*    NODE, IN KSECT.
3682    C
3683          LSTCL = (IORPD - 1) * NDFND
3684    C
3685    C*    LAST IS THE POSITION OF THE LAST DIAGONAL ENTRY ON COLUMN ICOL-1
3686    C
3687          LAST = IABS( LD (ICOL) )
3688    C
3689          DO 40 IDFPD = 1 , NDFND
3690    C
3691          JJ = LSTCL + IDFPD
3692    C
3693    C
3694          IF (JJ . GT . ICOL)                          GO TO 35
3695    C
3696    C*    READ COLUMN-WISE.
3697    C
3698          IKSC = LAST + (IORPD - LWSTOR) * NDFND + IDFPD
3699    C
3700          KSECT (IKSC) = 0.0
3701    C
3702          IF (JJ . EQ . ICOL)                      KSECT (IKSC) = ZO
3703    C
3704          GO TO 40
3705    C
3706    C*    READ ROW-WISE.
3707    C*    'LAST' IS THE LAST (DIAGONAL) POSITION OF COLUMN  JJ-1.
3708    C
3709       35 LAST = IABS( LO (JJ) )
3710    C
3711          IKSC = LAST + (IORPD - LWSTOR) * NDFND + ICOL - LSTCL
3712          KSECT (IKSC) = 0.0
3713    C
3714       40 CONTINUE
3715    C
3716       50 CONTINUE
3717    C
3718    C*    IF NDPD IS THE SYMMETRIC NODE, SKIP.
3719    C
3720       53 IF (SYMNO)                                  GO TO 60
3721    C
3722    C*    IF NDPD IS AN INTERNAL NODE, SKIP.
```

```
C
      IF (INTND(NDPD) . EQ . 3HYES)                    GO TO 60
C
C*    SINCE NDPD IS AN EXTERNAL NODE, FIND ITS SYMMETRICAL COUNTERPART
C*    LOCATED ON THE OTHER SIDE OF THE SECTOR.
C
   55 DO 58 I =  1 , NNDE1
C
      IF ( NDPD . NE . NDEX1(I) )                      GO TO 56
      NODE = NDEX2(I)
      GO TO 59
C
   56 IF ( NDPD . NE . NDEX2(I) )                      GO TO 58
      NODE = NDEX1(I)
      GO TO 59
C
   58 CONTINUE
C
C*    MODIFY THE ROW AND THE COLUMN OF KSECT CORRESPONDING TO 'NODE'
C*    THE SAME WAY AS WAS DONE FOR THOSE CORRESPONDING TO 'NDPD'.
C
   59 NDPD = NODE
C
      SYMND = . TRUE .
C
      GO TO 2
C
   60 CONTINUE
C
      RETURN
      END
C
C
C
C-----------------------------------------------------------------------
C
C
C
C*    NOW CONSTRUCT COLUMNS OF KIE FROM KSECT AND GIVE THEM TO
C*    SUBROUTINE SKYSOL TO SOLVE FOR THE FIRST COLUMN OF BRHS.
C*    KEE* IS OBTAINED BY SUBTRACTING THESE BRHS'S FROM THE
C*    THE CORRESPONDING COLUMNS OF KEE (AS THESE BRHS'S ARE
C*    BEING PUT OUT BY SKYSOL).
C*    ON INPUT THE EXTERNAL ENTRIES OF BRHS ARE ZERO AND THE INTERNAL
C*    ENTRIES CONTAIN A COLUMN OF KIE.
C*
C
      SUBROUTINE SKYFAC (A, NDEG, NEND, N, LD, V, SINGAB,
     + PDCHEK, ICOND, IAFILE, ACOND, DETCF, IDETEX, NEGEIG, IFAIL)
C
```

```
C*                    ----------------------                                  *
C*                    - D I S C U S S I O N -                                  *
C*                    ----------------------                                  *
C*                                                                            *
C*     THE UPPER TRIANGLE OF THE INPUT SYMMETRIC MATRIX (A) IS STORED         *
C*     IN COLUMN-WISE SKYLINE FASHION IN THE FIRST NWA WORDS OF               *
C*     ARRAY (A). THE STORAGE SCHEME IS BEST ILLUSTRATED BY THE               *
C*     (6 * 6) MATRIX (DOTS DENOTE ZEROS OUTSIDE SKYLINE) :                   *
C*                                  -                          -              *
C*                                 I  A11   .     A13   .   .     A16  I      *
C*                                 I        A22   0     A24  .     0    I      *
C*     (A)   =                     I              A33   A34  .     0    I      (1)  *
C*                                 I                    A44  .     A46  I      *
C*                                 I      (SYMM)              A55   A56  I      *
C*                                 I                          A66   I      *
C*                                  -                          -              *
C*     THIS IS PRESENTED TO SKYFAC AS A 15-WORD STRING :                      *
C*     A11, A22,A13,0, A33, A24,A34,A44,A55,A16,0,0, A46, A56, A66 (2)        *
C*                                                                            *
C*     IN ARRAY A, THE POSITION OF THE DIAGONAL ELEMENTS IS MARKED BY         *
C*     A (N+1)-WORD INTEGER ARRAY LD, SUCH THAT LD(1) = 0 AND                 *
C*     LD(I+1) POINTS TO A(I,I) IN ARRAY A FOR I = 1(1)N.                     *
C*                                                                            *
C*                          LD =  0, 1, 2, 5, 8, 9, 15                (2A)  *
C*                                                                            *
C*     THE SYMMETRIC DECOMPOSITION                                            *
C*                                                                            *
C*                             (A) = (L) (D) (U)                      (3)     *
C*                                                                            *
C*     WHERE (U) IS THE TRASPOSE OF (L), WHICH IS A UNIT LOWER                *
C*     TRIANGULAR MATRIX, AND (D) IS A DIAGONAL MATRIX, IS PERFORMED.         *
C*     THE STRICT UPPER TRIANGLE OF (U) OVERWRITES THE OFF-DIAGONAL           *
C*     ENTRIES OF (A). WHEREAS THE RECIPROCALS OF THE ELEMENTS OF             *
C*     (D) REPLACE THE DIAGONAL ELEMENTS OF (A). THE OUTPUT                   *
C*     STORAGE SCHEME FOR THE EXAMPLE MATRIX (1) IS THEN:                     *
C*                                                                            *
C*                                  -                          -              *
C*                                 I  1/D1          U13               U16  I      *
C*                                 I        1/D2    U23    U24        U26  I      *
C*                                 I                1/D3   U34        U36  I      *
C*                                 I                       1/D4       U46  I      *
C*                                 I                              1/D5 U56  I      *
C*                                 I                                   1/D6 I      *
C*                                  -                          -              *
C*                                                                            *
C*     THE DECOMPOSITION PROCESS MAY BE ABORTED UNDER EITHER OF THE           *
C*     FOLLOWING CONDITIONS :                                                 *
C*                                                                            *
C*      (A) THE MATRIX (A), MODIFIED BY THE ROUNDING ERRORS, APPEARS          *
C*          SINGULAR. THE SIGULARITY TEST USED AT THE J-TH STEP IS            *
```

```
C*                                                                      *
C*                        ABS(D(J)) LE 8*EPSMAC*R(J)                (5) *
C*                                                                      *
C*    WHERE R(J) DENOTES THE EUCLIDEAN NORM OF THE J-TH ROW OF          *
C*    (A), AND EPSMAC IS THE SMALLEST POSITIVE FLOATING-POINT           *
C*    NUMBER SUCH THAT 1.0+EPSMAC GT 1.0 ON THE COMPUTER                *
C*    BEING USED. THIS SIGULARITY TEST IS ONLY ENFORCED IF              *
C*    INPUT FLAG SINGAB IS .TRUE. (SEE INPUT ARGUMENTS) .               *
C*     (B) THE MATRIX (A) IS INDEFINITE, I.E. ,                         *
C*                                                                      *
C*                           D(J) LT 8*EPSMAK*R(J)                (6) *
C*                                                                      *
C*    THE POSITIVE DEFINITENESS TEST IS ONLY ENFORCED IF INPUT          *
C*    FLAG PDCHEK IS .TRUE. (SEE INPUT ARGUMENTS). IT SHOULD BE         *
C*    STRESSED THAT THE NUMERIC STABILITY OF THE DECOMPOSITION          *
C*     (B) IS NOT GUARANTEED IF MATRIX (A) IS INDEFINITE.               *
C*                                                                      *
C*    THIS SUBROUTINE MAY ALSO BE UTILIZED TO HANDLE THE MORE           *
C*    GENERAL CASE OF A CONSTRAINED LINEAR SYSTEM, IN WHICH SOME OF     *
C*    THE X-ENTRIES ARE PRESCRIBED. A CONSTRAINED SYSTEM MAY BE         *
C*    FORMALLY WRITTEN IN THE PARTITIONED MATRIX FORM :                 *
C*                                                                      *
C*                      (AFF) (XF) + (AFC) (XC) = (BF)                  *
C*                                                                  (7) *
C*                      (ACF) (XF) + (ACC) (XC) = (BC)                  *
C*                                                                      *
C*                                                                      *
C*    WHERE (BF) AND (XC) ARE GIVEN AND (XF) AND (BC) ARE TO BE         *
C*    DETERMINED (SUFFICES ''F'' AND ''C'' STAND FOR ''FREE''           *
C*    AND ''CONSTRAINED'', RESPECTIVELY. HOWEVER, IT MUST BE            *
C*    EMPHASIZED THAT (A) SHOULD NOT BE PHYSICALLY ARRANGED TO          *
C*    CONFORM TO (7).                                                   *
C*                                                                      *
C*    IF X(I) IS TO BE PRESCRIBED, THE I-TH ROW AND AND COLUMN OF (A)   *
C*    ARE SAID TO BE CONSTRAINED. SUCH CONDITION IS MARKED BY A         *
C*    NEGATIVE VALUE OF LD(I+1). FOR INSTANCE, IF ROWS 3 AND 6 OF THE   *
C*    EXAMPLE MATRIX (1) ARE CONSTRAINED, THE LD ARRAY MUST BE :        *
C*                                                                      *
C*                        LD = 0, 1, 2, -5, 8, 9, -15               (7A) *
C*                                                                      *
C*    CONSTRAINED ROWS AND COLUMNS ARE IGNORED IN THE FACTORIZATION     *
C*    OF A CONSTRAINED MATRIX AND VALUES THEREIN RETURN UNALTERED.      *
C*    FOR INSTANCE, THE OUTPUT FROM SKYFAC FOR THE EXAMPLE MATRIX       *
C*     (1) WITH THE LD ARRAY (7A) WOULD BE :                            *
C*                                                                      *
C*          -                                          -                *
C*         I  1/D1          A13          A16  I                        *
C*         I       1/D2     A23   U23     0   I                        *
C*         I                A33   U34     0   I                   (8)  *
C*         I              1/D4          A46  I                         *
C*         I                     1/D5   A56  I                         *
```

```
C*            I                                   A66  I                  *
C*              -                                    -                    *
C*    THAT IS, ONLY THE (UNREARRANGED) FREE-FREE PORTION (AFF) IS          *
C*    EFFECTIVELY FACTORIZED INTO (LFF) (DFF) (UFF).                       *
C*                                                                         *
C*                                                                         *
C*    ADDITIONAL FEATURES OF THE SUBROUTINE INCLUDE:                       *
C*                                                                         *
C*     1.  THE DETERMINANT OF (A) (OR (AF) IF CONSTRAINED) MAY BE          *
C*         COMPUTED ON REQUEST.                                            *
C*                                                                         *
C*     2.  AN ESTIMATE OF THE EUCLIDEAN CONDITION NUMBER OF (A) (OR        *
C*         (AFF) IF CONSTRAINED) MAY BE OBTAINED.                          *
C*                                                                         *
C*     3.  ACCUMULATION OF INNER PRODUCTS IN DIFFERENT PRECISION           *
C*         MODES IS FACILITATED BY INCLUDING THE INNERPODUCT FUNCTION       *
C*         IN THE FORMAL ARGUMENT LIST. VARIOUS ENTRY POINT NAMES          *
C*         MAY THUS BE USED WITHOUT EDITING/RECOMPILING.                   *
C*                                                                         *
C*     4. A CAPABILITY FOR PARTIAL FACTORIZATION IS INCLUDED. THE          *
C*         FACTORIZATION (3) IS ACUALLY CARRIED OUT FROM EQUATION          *
C*         (NBEG+1) TROUGH NEND, WHERE NBEG, NEND ARE TWO INPUT            *
C*         PARAMETERS. NORMALLY NBEG=0 AND NEND=N (MATRIX ORDER).          *
C*         HOWEVER, IN CERTAIN CASES (E.G., LOCAL STABILITY ANALYSIS,      *
C*         LOCAL NONLINEARITIES), A PARTIAL FACTORIZATION MAY BE           *
C*         USEFUL.                                                         *
C*                                                                         *
C***********************************************************************
C*                                                                         *
C*                          ------------                                   *
C*                          - U S A G E -                                  *
C*                          ------------                                   *
C*                                                                         *
C**** 1. INPUT ARGUMENTS -                                                 *
C*                                                                         *
C*     NBEG,NEND      TWO PARAMETERS SPECIFYING FACTORIZATION RANGE         *
C*                    AS EXTENDING FROM EQUATION (NBEG+1) THROUGH NEND.    *
C*                    (SEE DISCUSSION). COMPLETE FACTORIZATION OF (A)      *
C*                    IS OBTAINED WITH NBEG=0 AND NEND=N (MATRIX ORDER)    *
C*                                                                         *
C*     N              ORDER OF MATRIX (A)                                  *
C*                                                                         *
C*                                                                         *
C*     LD             (N+1)-WORD ARRAY OF DIAGONAL LOCATIONPOINTERS        *
C*                    (SEE DISCUSSION)                                     *
C*                                                                         *
C*     V              A SCRATCH FLOATING-POINT ARRAY OF LENGTH AT LEAST    *
C*                    N IF ICOND=0, 4*N IF ICOND IS POSITIVE.              *
C*                                                                         *
C*     DOTPRD         FUNCTION RETURNING INNER (DOT, SCALAR) PRODUCT       *
C*                    OF TWO VECTORS. THIS FUNCTION IS INVOKED BY          *
C*                    DOTPRD(P,Q,M) AND SHOULD EVELUATE THE INNER          *
```

```
C*                    PRODUCT OF THE TWO M-WORD VECTORS P AND Q.          *
C*                                                                        *
C*     SINGAB         LOGICAL FLAG CONTROLLING ACTION TO BE TAKEN IF      *
C*                    THE SINGULARITY TEST (5) IS VERIFIED:               *
C*                                                                        *
C*                    SINGAB = .TRUE.   ABORT SETTING IFAIL = J           *
C*                    SINGAB = .FALSE.  SET D(J) TO 8*EPSMAC*R(J) AND     *
C*                    PROCEED.                                            *
C*                                                                        *
C*                                                                        *
C*                    FOR NORMAL USE OF SKYFAC IN EQUATION SOLVING, ONE   *
C*                    SHOULD ALWAYS SET SINGAB = .TRUE.. THE .FALSE.      *
C*                    SETTING IS RECOMMENDED ONLY IF THE DECOMPOSITION    *
C*                    (3) IS TO BE USED IN THE CALCULATION OF EIGEN-      *
C*                    VECTORS OF (AFF) BY INVERSE POWER ITERATION.        *
C*                                                                        *
C*     PDCHEK         LOGICAL FLAG INDICATING WHETHER (A) (OR (AFF) IF    *
C*                    CONSTRAINED) IS TO BE CHECKED FOR POSITIVE          *
C*                    DEFINITENESS, I.E., CONDITION (6) TESTED :          *
C*                                                                        *
C*                    PDCHEK = .TRUE.   ABORT SETTING IFAIL = -J          *
C*                    PDCHEK = .FALSE.  DON'T TEST FOR INDEFINITENESS.    *
C*                                                                        *
C*     ICOND          IF POSITIVE, A LOWER BOUND ESTIMATE OF THE EUCLI-   *
C*                    DEAN CONDITION NUMBER C(A) OF (A) (OR (AFF) IF      *
C*                    CONSTRAINED) WILL BE CALCULATED BY PERFORMING ONE   *
C*                    ITERATIVE REFINEMENT CYCLE ON ICOND ''RANDOM''      *
C*                    RHS VECTORS. THE LARGER ICOND, THE BETTER THE       *
C*                    ESTIMATE. HOWEVER, FOR PRACTICAL A PRIORI           *
C*                    ASSESMENT OF SOLUTION ACCURACY, ICOND = 2 IS        *
C*                    OFTEN SUFFICIENT.                                   *
C*                                                                        *
C*     IAFILE         IF NONZERO, COPIES OF (A) AND ITS FACTORIZATION     *
C*                    WILL BE WRITTEN IN LOGICAL UNIT IAFILE(SEE          *
C*                    ''OUTPUT FILES'' BELOW). IF ZERO, NO COPIES         *
C*                    ARE WRITTEN. A NONZERO IAFAIL IS REQUIRED           *
C*                    IN THE FOLLOWING TWO CASES :                        *
C*                                                                        *
C*                     (A) ICOND IS POSITIVE (C(A) REQUESTED)             *
C*                     (B) ITERATIVE REFINEMENT OF SOLUTION VECTORS       *
C*                         WILL BE REQUESTED WHEN CALLING SKYSOL.         *
C*                                                                        *
C*                    WARNING - DO NOT USE FEATURES (A) OR (B) IF THE     *
C*                    PARTIAL FACTORIZATION OPTION IS BEING EXERCISED.    *
C*                                                                        *
C**** 2. INPUT-OUTPUT ARGUMENTS -                                        *
C*                                                                        *
C*     A              ARRAY CONTAINING INPUT MATRIX TO BE FACTORIZED      *
C*                    (OR UNFACTORIZED PORTION THEREOF IF NBEG GT 0).     *
C*                    OUTPUT FACTORIZATION-DEFINITION ELEMENTS , AS       *
C*                    DESCRIBED IN THE DISCUSSION.                        *
```

```
C*                                                                      *
C***** 3. OUTPUT ARGUMENTS -                                            *
C*                                                                      *
C*     ACOND          ESTIMATE OF C(A) IF ICOND GT 0 (AND IFAIL = 0)    *
C*                                                                      *
C*  DETCF,IDETEX      TWO NUMBERS (REAL,INTEGER), WHICH TOGETHER GIVE   *
C*                    THE DETEMINANT OF (A) (OR (AFF) IF (A) IS         *
C*                    CONSTRAINED) IN THE FORM                          *
C*                                                                      *
C*                       DET(A) = DETCF  * 2**IDETEX              (10)  *
C*                                                                      *
C*                    WHERE DETCF IS IN THE RANGE (1/16,1), PROVIDED    *
C*                    DETCF IS PRESET TO 1.0 ON INPUT. IF DETCF IS ZERO *
C*                    ON INPUT, THE DETERMINANT IS NOT CALCULATED.      *
C*                    IF OUTPUT IFAIL = +J, DETCF IS SET TO ZERO.       *
C*                    IF OUTPUT IFAIL == -J, (J GT 0), THE DETERMINANT  *
C*                    (10) IS THAT OF THE UPPER (J * J) PRINCIPAL MINOR *
C*                                                                      *
C*     NEGEIG         COUNT OF NEGATIVE EIGENVALUES OF (A) (OR (AFF) IF *
C*                    (A) IS CONSTRAINED) IF IFAIL = 0 ON OUTPUT.       *
C*                                                                      *
C*     IFAIL          FACTORIZATION FAILURE INDICATOR:                  *
C*                                                                      *
C*                    IFAIL = 0   FACTORIZATION SUCCESSFULLY EXECUTED.  *
C*                    IFAIL = J   SIGULARITY CONDITION (5) DETECTED.    *
C*                    IFAIL = -J  INDEFINITE CONDITION (6) DETECTED.    *
C*                                                                      *
C***** 4. COMMON INPUTS -                                               *
C*                    NONE.                                             *
C*                                                                      *
C***** 5. COMMON INPUT-OUTPUTS -                                        *
C*                    NONE.                                             *
C*                                                                      *
C***** 6. COMMON OUTPUTS -                                              *
C*                    NONE.                                             *
C*                                                                      *
C***** 7. INPUT FILES -                                                 *
C*                    NONE.                                             *
C*                                                                      *
C*****  8. OUTPUT FILES -                                              *
C*        IAFILE      IF NONZERO, LOGICAL UNIT IAFILE RECEIVES COPIES   *
C*                    OF THE ORIGINAL AND OUTPUT CONTENTS OF ARRAY A,   *
C*                    SAVED AS 2 FORTRAN BINARY RECORDS OF SIZE LD(N+1).*
C*                                                                      *
C*****  9. SCRATCH FILES -                                             *
C*                    NONE.                                             *
C*                                                                      *
C***** 10. PROGRAM SUBROUTINES -                                        *
C*                    DOTPRD, SKYROM, SKYSOL                            *
C*                                                                      *
C***** 11. LIBRARY SUBROUTINES -                                        *
```

```
C*        SQRT, FORTRAN BINARY I/O                                             *
C*                                                                             *
C*******************************************************************************
C
C     T Y P E   A N D   D I M E N S I O N   S T A T E M E N T S
C
      REAL         A(1),      V(1)
      REAL         AIJ2,      D,        DETCF,        DOTPRD,       EPSMAC
      INTEGER      LD(1)
      LOGICAL      PDCHEK,       SINGAB
      EQUIVALENCE    (AIJ2,D,DMAX)
C
C     NOTE- APPROPRIATE VALUES OF EPSMAC FOR VARIOUS COMPUTERS ARE:
C     CDC 6000/7000 SERIES                  7.11E-15 (SINGLE PRECISION)
C     CDC 170       SERIES                  3.13E-294(SINGLE PRECISION)
C     IBM 360/370   SERIES                  9.54E-07 (REAL*4 PRECISION)
C     IBM 360/370   SERIES                  2.22E-16 (REAL*8 PRECISION)
C     UNIVAC 1108/1110, IBM 7094            1.49E-08 (SINGLE PRECISION)

      DATA           EPSMAC/3.13E-294/

C
C                I N I T I A L I Z A T I O N
C

      IFAIL =  0
      IDETEX = 0
      NEGEIG = 0
      SINGAB = .TRUE.

C
C     SAVE ORIGINAL MATRIX IF IAFILE NE 0 AND NBEG = 0
C

      NWA = IABS(LD(N+1))
      IF(IAFILE.EQ.0)                    GO TO 200
      IF(NBEG.NE.0)                      GO TO 200
      REWIND IAFILE
      WRITE (IAFILE)      (A(J),J=1,NWA)
C     COMPUTE SQUARED LENGTH OF UNCONSTRAINED ROWS NBEG+1 THRU N
C
  200 NBEGP1 = NBEG + 1
      DO 1000 I = NBEGP1,N
        II = LD(I+1)
        IF(II)           1000,1000,400
  400   V(I) = A(II)**2.
        M = II - I
        K = MAX0 (NBEGP1,IABS(LD(I))-M+1)
        L = MIN0 (NEND,I) - 1
        IF(K-L)          500,500,1000
  500   DO 800 J = K,L
```

```
4080              IF(LD(J+1))   800,800,600
4081     600      AIJ2 = A(M+J)**2.
4082              V(I) = V(I) + AIJ2
4083              V(J) = V(J) + AIJ2
4084     800      CONTINUE
4085    1000 CONTINUE
4086  C
4087
4088
4089  C              F A C T O R I Z A T I O N    S E C T I O N
4090  C
4091            DO 4000 J = NSEGP1,NEND
4092  C
4093  C       COMPUTE KU SUPERDIAGONAL ENTRIES OF J-TH COLUMN OF (U)
4094  C       IF UNCONSTRAINED
4095  C
4096              JJ = LD(J+1)
4097              IF(JJ)           4000,4000,1200
4098     1200     D = A(JJ)
4099              JMJ = IABS(LD(J))
4100              JK = JJ - JMJ
4101              KU = JK - 1
4102              IF(KU.EQ.0)                      GO TO 2200
4103              DO 2000 K = 1,KU
4104                I = J - JK + K
4105                V(K) = 0.0
4106                II = LD(I+1)
4107                IF(II)         2000,2000,1800
4108     1800       M = MIND (II-IABS(LD(I)),K) - 1
4109                IJ = JMJ + K
4110                V(K) = A(IJ) - DOTPRD (A(II-M),V(K-M),M)
4111                A(IJ) = V(K) * A(II)
4112     2000       CONTINUE
4113  C
4114  C       COMPUTE DIAGONAL ELEMENT  D(J)
4115  C
4116              D = D - DOTPRD(A(JMJ+1),V,KU)
4117  C
4118  C       SINGULARITY TEST
4119  C
4120     2200     TOLROW = 8.*EPSMAC*SQRT(V(J))
4121              IF(ABS(D).GT.TOLROW)             GO TO 2500
4122              IF(SINGAB)                       GO TO 6000
4123              D = TOLROW
4124     2500     A(JJ) = 1.0/D
4125  C
4126  C       UPDATE DETERMIANT IF DETCF IS NONZERO
4127  C
4128     3000     IF(DETCF.EQ.0.0)                 GO TO 3500
4129              DETCF = DETCF*D
4130     3200     IF(ABS(DETCF).LT.1.0)            GO TO 3400
```

```
      DETCF = DETCF*.0625
      IDETEX = IDETEX + 4
      GO TO 3200
 3400 IF (ABS(DETCF).GE.  .0625)     GO TO 3500
      DETCF = DETCF*16.
      IDETEX = IDETEX - 4
      GO TO 3400
C
C     POSITIVE DEFINETNESS CHECK  (IF PDCHEK = .TRUE.)
C
 3500 IF(D.GT.0.0)                   GO TO 4000
      NEGEIG = NEGEIG + 1
      IF(PDCHEK)                     GO TO 6500
 4000 CONTINUE
C
C     SAVE FACTORIZATION IF IAFILE NE 0 AND N = NEND
C
      IF (IAFILE.EQ.0)               GO TO 5000
      IF (NEND.NE.N)                 GO TO 5000
      WRITE (IAFILE)    (A(J),J=1,NWA)
      IF(ICOND.EQ.0)     GO TO 5000
C
C     M A T R I X   C O N D I T I O N   E S T I M A T I O N
C
      K = 0
      DMAX = 0.0
      DO 4200 I = 1,ICOND
C
C     SET UP A RANDOM RHS VECTOR IN V
C
        CALL SKYRDM (V, NWA, 0.0, K)
C
C     SOLVE FOR V AND DO ONE CYCLE OF ITERATIVE REFINEMENT TO
C     OBTAIN 2-NORM RELATIVE SOLUTION ERROR IN DELTA
C
        CALL SKYSOL (A, N, LD, 0, -1, V, V(N+1),
     *    1, IAFILE, V(2*N+1), DELTA)
        DMAX = AMAX1 (DELTA,DMAX)
 4200   CONTINUE
C
C     NOW ESTIMATE C(A) FROM LARGEST DELTA AND MACHINE PRECISION
C
      ACOND = DMAX/((1.0+DMAX)*EPSMAC)
C
 5000 RETURN
C
C                    E R R O R   E X I T
C
C     SINGULAR MATRIX (SINGAB = .TRUE.)
 6000 IFAIL = J
      DETCF = 0.0
```

The Postprocessor Subprogram

```
      GO TO 5000
C     INDEFINATE MATRIX (PDCHEK = .TRUE.)
 6500 IFAIL = -J
      GO TO 5000
      END
C
C
C
C-----------------------------------------------------------------
C
C
C***  THIS SUBROUTINE CONSTRUCTS ROW NUMBER IROW OF KEI (COLUMN
C***  NUMBER IROW OF KIE) AND PUTS IT IN THE ENTRIES OF BRHS
C***  ASSOCIATED WITH THE INTERNAL NODES.
C
      SUBROUTINE KEIBRS (IOREX, IDFEX, IROW, NNDSH, NDFSC, NDFND,
     + NDFEX, NDFIN, NNDEX, NNDIN, NDINT, INTND, NDORD, SEQND, INDEX,
     + NDCON, LD, KSECT, BRHS)
C
      INTEGER  SEQND(1)
C
      REAL  KSECT(1)
C
      DIMENSION  NDINT(1), INTND(1), NDORD(1), INDEX(1), NDCON(1),
     + LD(1), BRHS(1)
C
C*    FIND THE CURRENT EXTERNAL NODE.
      NDEX = SEQND(IOREX)
C
C*    INITIALIZE BRHS TO ZERO.
C
      CALL SETZRO (NDFSC, 1, BRHS)
C
C*    CONSIDER ALL INTERNAL NODES.
C
      DO 170 INDIN = 1 , NNDIN
C
C*    DETERMINE THE CURRENT INTERNAL NODE.
C
      NDIN = NDINT(INDIN)
      IORIN = NDORD(NDIN)
C
C*    DETERMINE N3, THE POSITION IN BRHS, BEFORE THOSE TO BE FILLED
C*    BY 12 ENTRIES FROM BLOCK NDEX-NDIN OR NDIN-NDEX.
C
      N3 = NDFND * (IORIN - 1)
C
      IPNT = INDEX(NDIN)
      NCND = NDCON(IPNT)
      IBEG = IPNT + 1
      IEND = IPNT + NCND
```

```
C
      IF (IOREX . GT . IORIN)                          GO TO 145
C
C*    IOREX < IORIN
C
C*    CONSIDER ALL NODES CONNECTED TO NDIN AND SEE IF NDEX IS AMONG
C*    THEM.
C
      DO 150 IPOS = IBEG , IEND
C
      NDCN = NDCON(IPOS)
C
      IF (INTND(NDCN) . EQ . 3HYES . OR . NDEX . NE . NDCN)  GO TO 150
C
      N1 = NDFND * (IORIN - IOREX)
C
      N2 = N1 - IOFEX
C
C*    PUT THE IOFEX-TH ROW OF BLOCK NDEX-NDIN IN BRHS.
C
      DO 140 IOF = 1 , NDFND
C
      IKSC = LD(N3+1+IOF) - N2 - IOF
      IBRHS = N3 + IOF
C
      BRHS(IBRHS) = KSECT(IKSC)
C
  140 CONTINUE
C
      GO TO 170
C
  150 CONTINUE
C
C*    IOREX K IORIN
C
C*    CONSIDER ALL NODES CONNECTED TO NDIN, AND SEE IF NDEX IS AMONG
C*    THEM.
C
  145 DO 168 IPOS = IBEG , IEND
C
      NDCN = NDCON(IPOS)
C
      IF (INTND(NDCN) . EQ . 3HYES . OR . NDEX . NE . NDCN)  GO TO 168
C
      N1 = NDFND * (IOREX - IORIN)
C
C*    DETERMINE N2, THE POSITION IN KSECT JUST BEFORE THE FIRST
C*    ENTRY OF THE COLUMN TO BE CONSIDERED.
C
      N2 = IABS( LD(IROW+1) ) - N1 - IOFEX
C
```

```
C*     PUT THE IDPEX-TH COLUMN OF BLOCK NDIN-NDEX IN BRHS.
C
       DO 160 IDF = 1 , NDFND
C
       IKSC = N2 + IDF
       IBRHS = N3 + IDF
       BRHS(IBRHS) = KSECT(IKSC)
C
  160 CONTINUE
C
       GO TO 170
C
  168 CONTINUE
C
  170 CONTINUE
C
       RETURN
       END
C
C
C
C-----------------------------------------------------------------
C
C
C***  WHEN CALLED FROM SUBROUTINE STIFSH,
C***  THIS SUBROUTINE TAKES THE INTERNAL ENTRIES OF BRHS, WHICH
C***  CONTAIN THE IROW-TH COLUMN OF KIE, AND ON RETURN PUTS THE
C***  PRODUCT  KEI * KII-INV * KIE  IN THE EXTERNAL ENTRIES OF BRHS.
C
C
       SUBROUTINE SKYSOL (A, N, LD, IOP, IBX, B, X,
     * IREF, IAFILE, V, DELTA)
C
C*****************************************************************
C*                                                               *
C*                        ----------                             *
C*                        - P U R P O S E -                      *
C*                        ----------                             *
C*                                                               *
C*    TO SOLVE THE LINEAR SYSTEM (A) (X) = (B), (A) BEING A SKYLINE-*
C*    STORED SYMMETRIC MATRIX PREFACTORED BY SUBROUTINE SKYFAC.   *
C*                                                               *
C*                     ----------                                *
C*                     - D I S C U S S I O N -                   *
C*                     ----------                                *
C*                                                               *
C*    SKYSOL CAN DEAL EITHER WITH THE UNCONSTRAINED LINEAR SYSTEM *
C*                                                               *
C*                          (A) (X) = (B)                        *
```

```
C*    WHERE THE ENTRIES RHS (B) IS PRESCRIBED AND (A) HAS BEEN PRE-    *
C*    FACTORED INTO (L) (D) (U) BY SKYFAC, OR WITH THE MORE GENERAL    *
C*    CASE OF A CONSTRAINED SYSTEM (CF.EQ.(7) IN SKYFAC):             *
C*                                                                    *
C*                        (AFF) (XF) + (AFC) (XC) = (BF)              *
C*                        (ACF) (XF) + (ACC) (XC) = (BC)              *
C*                                                                    *
C*    WHERE (BF) AND (BC) ARE PRESCRIBED AND THE FREE-FREE PORTION    *
C*    (AFF) HAS BEEN PREFACTORIZED INTO (LFF) (DFF) (UFF) BY SKYFAC.  *
C*    SINCE THE CONSTRAINED CASE EMBODIES THE UNCONSTRAINED ONE, THE  *
C*    THE PROCESS DESCRIPTION BELOW ASSUMED A CONSTRAINED SYSTEM,     *
C*                                                                    *
C* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -*
C*                                                                    *
C*    THE SOLUTION PROCESS IN SKYSOL CONSISTS OF UP TO SIX STAGES -   *
C*                                                                    *
C*    1. RHS MODIFICATION : (BF) IF REPLACED BY (BF) - (AFC) (XC).    *
C*       THIS STEP IS SKIPPED IF THE LINEAR SYSTEM IS UNCONSTRAINED   *
C*       OR HOMOGENEOUSLY CONSTRAINED ((XC) = (0)).                   *
C*                                                                    *
C*    2. FORWARD REDUCTION: THE TRIANGULAR SYSTEM (LFF) (ZF) = (BF)   *
C*       IS SOLVED FOR (ZF).                                          *
C*                                                                    *
C*    3. SCALING : THE DIAGONAL SYSTEM (DFF) (YF) = (ZF) IS SOLVED    *
C*                                                                    *
C*    4. BACKSUBSTITUTION : THE TRIANGULAR SYSTEM (UFF) (XF) = (YF)   *
C*       IS SOLVED FOR (XF).                                          *
C*                                                                    *
C*    5. ITERATIVE REFINEMENT : IF ARGUMENT IREF GT 0, THE RESIDUAL   *
C*       (RF) = (BF) - (AFF) (XF) - (AFC) (XC)   IS COMPUTED USING    *
C*       HIGHER PRECISION ARITHMATIC, AND STAGES 2-4 REPEATED TO      *
C*       SOLVE (AFF) (DFF) = (RF)   (NOTE THAT (RC) = (0))    . THE   *
C*       CORRECTION (DXF) IS ADDED TO (XF) AND THE PROCESS REPEATED   *
C*       IREF TIMES.                                                  *
C*                                                                    *
C*    6. COSTRAINED RHS RECOVERY: (BC) = (AFC) (XF) + (ACC) (XC)      *
C*       IS CALCULATED IF REQUESTED (SEE INPUT ARGUMENT IBX)          *
C*                                                                    *
C**********************************************************************
C*                            -------------                           *
C*                            - U S A G E -                           *
C*                            -------------                           *
C*                                                                    *
C**** 1. INPUT ARGUMENTS -                                           *
C*                                                                    *
C*                                                                    *
C*       A          OUTPUT MATRIX FACTORS PRODUCED BY SKYFAC         *
C*                                                                    *
C*       N          SYSTEM ORDR                                      *
C*                                                                    *
C*       LD         (N+1)-WORD ARRAY OF DIAGONAL LOCATION POINTERS   *
```

```
C*                                                                      *
C*      DOTPRD      ENTRY POINT OF INNERPODUCT FUNCTION (CF. SKYFAC)    *
C*                                                                      *
C*      IOP         OPERATION CONTROL FLAG                              *
C*                                                                      *
C*                  IOP = 0  : NORMAL SOLUTION PROCESS                  *
C*                  IOP = -1 : EXECUTE STAGES 1-2 AND EXIT (USFUL       *
C*                  IN CERTAIN SUBSTRUCTURING APPLICATIONS)             *
C*                  IOP = 1  : EXECUTE STAGES 3-4 AND EXIT (USFUL IN    *
C*                  ON STARTING AN INVERSE ITERATION PROCESS)           *
C*                                                                      *
C*      IBX         CONTROLLING OUTPUT CONFIGURATION OF ARRAY B.        *
C*                                                                      *
C*                  IBX = 0 : B AND X ARE IDENTIFIED IN THE (CALLING    *
C*                  PROGRAM. THE SOLUTION VECTOR X THEN OVERWRITES B.   *
C*                  THIS OPTION MAY BE USED ONLY IF                     *
C*                  (A) THE SYSTEM IS UNCONSTRAINED, OR HOMOGENEOUSLY   *
C*                      CONSTRAINED WITH NO RECOVERY OF (BC) DESIRED    *
C*                  (B) NO ITERATIVE REFINEMENT REQUESTED (IREF=0)      *
C*                                                                      *
C*                  IBX = -1 : B AND X ARE NOT IDENTIFIED IN THE        *
C*                  CALLING PROGRAM BUT COMPUTATION OF (BC) IS NOT      *
C*                  DESIRED.                                            *
C*                                                                      *
C*                  IBX = 1 : B AND X ARE NOT IDENTIFIED IN THE         *
C*                  CALLING PROGRAM AND RECOVERY OF (BC) IS REQUESED.   *
C*                                                                      *
C*                  NOTE - : IF THE INPUT SYSTEM CONTAINS NONHOMO-      *
C*                  GENEOUS CONSTRAINTS, IBX MUST BE -1 OR +1, I.E.,    *
C*                  ARRAYS B AND X MAY NOT SHARE THE SAME STORAGE       *
C*                  SPACE.                                              *
C*                                                                      *
C*      IREF        IF GT 0, PERFORM IREF CYCLES OF ITERATIVE           *
C*                  REFINEMENT OF THE SOLUTION VECTOR. IN PRACTICE      *
C*                  ONE SHOULD NOT REQUEST IREF GT 1 UNLESS THE         *
C*                  ENTRIES OF (A) HAPPEN TO BE EXACT NUMBERS.          *
C*                                                                      *
C*      IAFILE      LOGICAL UNIT CONTAINING COPIES OF (A) AND ITS       *
C*                  FACTORIZATION (SEE SKYFAC). REQUIRED IF IREF GT 0   *
C*                                                                      *
C*      V           SCRATCH FLOATING POINT ARRAY OF LENGH 2*N           *
C*                  TO BE USED IN ITERATIVE REFINEMENT PROCESS.         *
C*                  MAY BE A DUMMY ARGUMENT IF IREF = 0                 *
C*                                                                      *
C**** 2. INPUT-OUTPUT ARGUMENTS -                                       *
C*                                                                      *
C*      B           ON INPUT , ARRAY B CONTAINS PRESCRIBED COMPONENTS   *
C*                  OF (B) AND (X), I.E., IF THE ITH EQUATION IS FREE   *
C*                  (CONSTRAINED), B(I) STORES THE RHS (LHS) VALUE.     *
C*                  THE OUTPUT CONFIGURATION OF B IS CONTROLLED BY      *
C*                  INPUT FLAG IBX (SEE INPUT ARGUMENTS).               *
```

```
4437  C*                                                                    *
4438  C**** 4. OUTPUT ARGUMENTS -                                           *
4439  C*                                                                    *
4440  C*       X           COMPUTED SOLUTION VECTOR                         *
4441  C*                                                                    *
4442  C*       DELTA       IF IREF GT 0, RATIO OF LAST CORRECTION LENGTH    *
4443  C*                    TO FINAL SOLUTION LENGTH. NOT USED IF IREF = 0   *
4444  C*                                                                    *
4445  C**** 4. COMMON INPUTS -                                              *
4446  C*                    NONE.                                           *
4447  C*                                                                    *
4448  C**** 5. COMMON INPUT-OUTPUTS -                                       *
4449  C*                    NONE.                                           *
4450  C*                                                                    *
4451  C**** 6. COMMON OUTPUTS -                                             *
4452  C*                    NONE.                                           *
4453  C*                                                                    *
4454  C**** 7. INPUT FILES -                                                *
4455  C*                                                                    *
4456  C*       IAFILE      SEE SUBROUTINE SKYFAC.                           *
4457  C*                                                                    *
4458  C**** 8. OUTPUT FILES -                                               *
4459  C*                    NONE.                                           *
4460  C*                                                                    *
4461  C**** 9. SCRATCH FILES -                                              *
4462  C*                    NONE.                                           *
4463  C*                                                                    *
4464  C*****10. PROGRAM SUBROUTINES -                                       *
4465  C*                    DOTPRD, SKYMUL                                  *
4466  C*                                                                    *
4467  C****11. LIBRARY SUBROUTINES -                                        *
4468  C*                    SQRT, FORTRAN BINARY I/O                        *
4469  C*                                                                    *
4470  C**********************************************************************
4471  C
4472  C     T Y P E   A N D   D I M E N S I O N   S T A T E M E N T S
4473  C
4474        INTEGER       LD(1)
4475        REAL          A(1),     B(1),     BI,     BXFAC,     DELTA
4476        REAL          RNORM,    V(1),     X(1),   XI,        XNORM
4477        EQUIVALENCE   (BI,XI,XNORM)
4478  C
4479  C               I N I T I A L I Z A T I O N
4480  C
4481        ASSIGN 3100 TO NEXT
4482        KREF = 0
4483        BXFAC = 0.0
4484        IF(IBX.EQ.0)                    GO TO 200
4485        BXFAC = 1.0
4486        DO 150 I = 1,N
4487   150    X(I) = B(I)
```

```
  200 IF (IDP.GT.0)                          GO TO 1800
      IF (IBX.EQ.0)                          GO TO 1100
C
C                R H S   M O D I F I C A T I O N
C
      DO 1000  I = 1,N
        II = LD(I+1)
        IF(II)                               300,1000,1000
  300   BI = B(I)
        IF(BI.EQ.0.0)                        GO TO 1000
        II = -II
        K = I - II + IABS(LD(I)) + 1
      DO 900 J = K,N
        JJ = LD(J+1)
      IF(JJ)                                 900,900,400
  400   M = J - I
        IF(M)                                500,600,600
  500   X(J) = X(J) - A(II+M)*BI
        GO TO 900
  600   IJ = JJ - M
        IF(IJ-IABS(LD(J)))                   900,900,800
  800   X(J) = X(J) - A(IJ)*BI
  900   CONTINUE
 1000 CONTINUE
C
C     F O R W A R D    S U B S T I T U T I O N   P A S S
C
 1100 DO 1500  I = 1,N
        II = LD(I+1)
        IF(II)                               1200,1200,1300
 1200   X(I) = 0.0
        GO TO 1500
 1300   IMI = IABS(LD(I))
        M = II - IMI - 1
        X(I) = X(I) - DOTPRO (A(IMI+1),X(I-M),M)
 1500   CONTINUE
      IF(IDP.NE.0)                           GO TO 5000
C
C                  S C A L I N G    P A S S
C
 1800 DO 2000  I = 1,N
        II = IABS(LD(I+1))
 2000   X(I) = A(II)*X(I)
C
C     B A C K   S U B S T I T U T I O N   P A S S
C
      I = N
      DO 3000  K = 1,N
        II = LD(I+1)
        IF(II)                               2200,2200,2400
 2200   X(I) = BXFAC*B(I)
```

```
4539            GO TO 2800
4540    2400    M = II - IABS(LD(I)) - 1
4541            IF (M.EQ.0)                         GO TO 2800
4542            DO 2500  J = 1,M
4543              X(I-J) = X(I-J) - A(II-J)*X(I)
4544    2500      CONTINUE
4545    2800    I = I - 1
4546    3000    CONTINUE
4547          IF(IBX*IAFILE.EQ.0)                   GO TO 4000
4548          GO TO NEXT,  (3100,3500)
4549    C
4550    C     I T E R A T I V E   R E F I N E M E N T   S E C T I O N
4551    C
4552    3100  KREF = KREF + 1
4553          IF (KREF-IREF)                        3200,3200,4000
4554    C
4555    C     CALCULATE RESIDUAL VECTOR (R) = (B) - (A) (X) USING SKYMUL.
4556    C     (R) RETURNS IN (X), AND OLD (X) IN V
4557    C
4558    3200  REWIND IAFILE
4559          NWA = IABS(LD(N+1))
4560          READ (IAFILE)                  (A(J),J=1,NWA)
4561          CALL SKYMUL (A, N, LD, X, V, -1, B, V)
4562    C
4563    C     SOLVE FOR CORRECTION (DX) (WHICH APPEARS IN X), CORRECT
4564    C     OLD SOLUTION AND EVALUATE 2-NORM RELATIVE ERROR DELTA
4565    C
4566          READ (IAFILE)                  (A(J),J=1,NWA)
4567          BXFAC = 0.0
4568          ASSIGN 3500 TO NEXT
4569          GO TO 1100
4570    3500  XNORM = 0.0
4571          RNORM = 0.0
4572          DO 3800  I = 1,N
4573            RNORM = RNORM + X(I) * X(I)
4574            X(I) = V(I) + X(I)
4575            XNORM = XNORM + X(I) * X(I)
4576    3800    CONTINUE
4577          DELTA = 0.0
4578          IF (XNORM.NE.0.0)                   DELTA = SQRT(RNORM/XNORM)
4579          IF (KREF-4)                         3100,3100,4000
4580    C
4581    C     C O N S T R A I N E D    R H S    R E C O V E R Y
4582    C
4583    4000  IF(IBX.LE.0)                        GO TO 5000
4584          DO 4800  I = 1,N
4585            II = LD(I+1)
4586            IF (II)                           4200,4200,4800
4587    4200    IMI = IABS(LD(I))
4588            M = -II - IMI - 1
4589            B(I) = DOTPRD (A(IMI+1),X(I-M),M)
```

```
      DO 4600  J = I,N
        IJ = IABS(LD(J+1)) + I - J
        IF(IJ-IABS(LD(J)))            4600,4600,4400
4400    B(I) = B(I) + A(IJ) * X(J)
4600    CONTINUE
4800  CONTINUE
5000 RETURN
      END
C
C
C
C     --------------------    S K Y M U L    --------------------
C
C
C
C     SUBROUTINE SKYMUL (A, N, LD, X, AX, IOP, B, R)
C
C     THIS SUBROUTINE POSTMULTIPLIES SKYLINE-STORED MATRIX (A) BY
C     VECTOR (X).
C     FURTHER OPERATIONS ARE CONTROLLED BYE INPUT FLAG IOP :
C       IOP = 0       EXIT AFTER OBTAINING (AX)
C       IOP = 1       FORM RESIDUAL VECTOR (R) = (B) - (AX) IN R
C       IOP =-1       MOVE (X) TO (R), AND FORM (B) - (AX) IN (X).
C
C     ARRAYS (AX) (D.P.) AND R MAY MAY BE EQUIVALENT IN THE CALLING PROGRA
C
      DOUBLE PRECISION       AX(1),    AIJ,    AXI
      INTEGER       LD(1)
      REAL          A(1),     B(1),     X(1),   R(1)
C
      DO 2000 I = 1,N
        II = IABS(LD(I+1))
        AX(I) = DBLE(A(II))*DBLE(X(I))
        M = II - IABS(LD(I)) - 1
        IF(M.EQ.0)                          GO TO 2000
        DO 1500  K = 1,M
          J = I - K
          AIJ = A(II-K)
          AX(I) = AX(I) + AIJ*DBLE(X(J))
          AX(J) = AX(J) + AIJ*DBLE(X(I))
 1500   CONTINUE
 2000 CONTINUE
C
      IF (IOP)                              2500,5000,3500
 2500 DO 2800  I = 1,N
        AXI = AX(I)
        R(I) = X(I)
        X(I) = DBLE(B(I)) - AXI
        IF (LD(I+1).LE.0)                   X(I) = 0.0
 2800 CONTINUE
      GO TO 5000
```

```
      3500 DO 3800  I = 1,N
             R(I) = DBLE(B(I)) - AX(I)
             IF(LD(I+1).LE.0)                R(I) = 0.0
      3800   CONTINUE
      5000 RETURN
           END
C
C
C
C
C
C
C
C     **********************    S  L  V  A  S  C    **********************
C
C
C
C***  THIS SUBROUTINE CALCULATES THE DISPLACEMENTS OF ALL SECTORS
C***  FOR WHICH SOLUTION IS REQUESTED. TO DO SO THE FOLLOWING PROCEDURE
C***  IS PERSUED:
C
C
C
C
C     1. GENERATE VECTOR ROWSA(2), WHICH DETERMINES IF ANY SINE OR AFRO
C        LOADS ARE PRESCRIBED.
C
C     2. IF CALCULATION OF REACTIONS IS REQUESTED,
C        A)  WRITE ALL OF THE SECTOR FOURIER LOAD COEFFICIENTS ON
C            DISK FILE TFBSC.
C
C        B)  REPLACE THE ENTRIES OF THE FOURIER LOAD COEFFICIENTS,
C            CORRESPONDING TO A RESTAINED DEGREE OF FREEDOM, WITH ZERO.
C
C     3. CONDENSE OUT THE FOURIER COEFFICIENTS OF THE INTERNAL
C        EQUIVALENT NODAL LOADS, AND THE THERMAL LOADS, AND THE
C        CORRECTIVE LOAD VECTOR. TO SAVE CENTERAL MEMORY SPACE,
C        PLACE THE CONDENSED VECTOR IN THE TOP NDFEX ENTRIES OF
C        ITSELF. THE NEXT NDFIN ENTRIES OF EACH VECTOR REMAINS INTACT.
C
C     4. GENERATE VECTOR INDIC(NINFA). WHERE, INDIC(I) = 0 INDICATES
C        THAT THE I-TH FOURIER COEFFICIENT OF THE INTERFACE
C        EQUIVALENT NODAL LOADS IS ZERO, AND NO SOLUTION FOR THE
C        I-TH FOURIER DISPLACEMENT COEFFICIENTS IS NECESSARY.
C
C     5. LOOP NINFA (NUMBER OF INTERFACES) TIMES, AND IF INDIC(IINFA)
C        IS EQUAL TO ZERO, SKIP STEPS A, B, C, AND D.
C
C     DO    IINFA = 1 , NINFA
C
C        A)  CALCULATE FBNIF(NDFIF), THE N-TH FOURIER COEFFICIENT OF THE
```

```
C              NINFA  INTERFACE EQUIVALENT NODAL LOADS.
C
C         B)  GENERATE, DIRECTLY FROM KSECT(NDFSL), KBNIF(NDFIF,NDFIF),
C             THE N-TH DIAGONAL SUBMATRIX OF KSECT, AFTER FOURIER
C             TRANSFORMATION (N=IINFA).
C
C         C)  SOLVE FOR ABNIF(NDFIF), THE N-TH FOURIER COEFFICIENT OF THE
C             NINFA DISPLACEMENTS.
C
C         D)  WRITE THE CURRENT ABNIF(NDFSC) ON DISK FILE TABIF.
C
C      CONTINUE  IINFA
C
C
C     6. LOOP OVER ALL ZONES FOR WHICH SOLUTION HAS BEEN REQUESTED
C
C      DO     IZNSL = 1 , LZNSL
C
C     7. CONSIDER ALL SECTORS IN THE CURRENT SOLUTION ZONE, IZNSL
C
C      DO     R = JSC  ,  KSC
C
C         A)  LOOPING NINFA TIMES FOR EACH R, CALCULATE THE  EXTERNAL
C             DISPLACEMENTS OF SECTOR R (INTERFACES R, AND R-1) AND PUT
C             THEM IN THE TOP NDFEX ENTRIES OF VECTOR ASECT(NDFSC).
C
C         B)  CALCULATE THE EQUIVALENT LOADS OF THE INTERNAL NODES OF
C             SECTOR R, AND PLACE THEM IN ENTRIES  (NDFEX+1)  TO NDFSC
C             OF KSECT(NDFSC).
C
C         C)  SOLVE FOR THE INTERNAL DISPLACEMENTS OF SECTOR R, BY
C             PASSING KSECT(NDFSL), ASECT(NDFSC), AND FSECT(NDFSC) TO
C             SUBROUTINE SKYSOL. ON RETURN, ENTREIS  (NDFEX+1)
C             TO NDFSC OF VECTOR ASECT(NDFSC) CONTAIN THE INTERNAL
C             NODE DISPLACEMENTS.
C
C       8. IF DISPLACEMENT OR STRESS ANALYSIS IS REQUESTED,
C          CONSIDER ALL OF THE ELEMENTS OF THE CURRENT SECTOR, R:
C
C      DO     IELSH = 1 , NELSH
C
C         A)  CALCULATE THE EQUIVALENT CENTROIDAL LOADS OF THE CURRENT
C             ELEMENT, IELSH.
C
C         B)  SOLVE FOR CENTROIDAL DISPLACEMENTS OF THE CURRENT ELEMENT,
C             IELSH.
C
C      CONTINUE   IELSH
C
C     9. IF REQUESTED, PRINT THE NODAL AND CENTROIDAL DISPLACEMENTS OF
C        THE CURRENT SECTOR.
```

```
C
C     10. IF STRESS ANALYSIS IS REQUESTED, WRITE THE NODAL AND THE
C         CENTROIDAL DISPLACEMENTS OF THE CURRENT SECTOR, R,
C         ON DISK FILES TASCT AND TACNT, RESPECTIVELY.
C
C     11. IF REQUESTED, CALCULATE AND PRINT THE DISPLACEMENTS OF
C         STRINGERS, INTERMEDIATE STRINGERS, RINGS, AND INTERMEDIATE
C         RINGS.
C
C      CONTINUE   R
C
C      CONTINUE   IZNSL
C
C
C
      SUBROUTINE SLVASC (MNDSH, MELSH, MELSR, MELISR, MELRN, MELIRN,
     + MMTIRN, MPROS, MLOCA, MCNLD, MAULD, MZNTP, MZNSL, MDFIF, NDFIF,
     + NDFSC, NDFSL, NDFCT, NCDSH, NDFCN, SLDLD, NODSH, NODSR, NODISR,
     + NODRN, NODIRN, PRPIRN, MTIRN, NODPD,
     + RSDOF, PRSDS, NREST, CSPRD, SCNLI, SCNLJ, NDCNL, ITCNL, TYDTL,
     + RFDTL, DVDTL, SCTPI, SCTPJ, SLVSC, CRDCL, SEQND, NDORD, IPNTR,
     + ICONN, LD, NDJNC, KCINV, KSECT, CORLD, ENLWT, ECLWT,
     + FRCNL, ENLAUL, ECLAUL, ENLSA, ECLSA, ENLTP, ECLTP, ROWSA, INDIC,
     + ASECT, FSECT, FBNIF, KBNIF, ABNIF, DISPL)
C
      INTEGER  CSDMC, DSKF1, DSKF2, R, SLDSP, SLRAC, SLSTS, SLDLD,
     + SLBKL, PLDSP, TYPEL, RSDOF(MPROS,1), CSPRD(1), SCNLI(MLOCA,1),
     + SCNLJ(MLOCA,1), TYDTL(MLOCA,1), RFDTL(MLOCA,1), SCTPI(MLOCA,1),
     + SCTPJ(MLOCA,1), SLVSC(MLOCA,MZNSL,2), SEQND(1), DRDSR(9),
     + DRDRN(9), DRDBR(6), ROWSA(2)
C
      REAL  ITCNL(MLOCA,MCNLD,1), KCINV(MELSH,2,2), KSECT(NDFSL),
     + KCOND(12,36), KCDIV(12,12)
C
      DIMENSION  NODSH(MELSH,1), NODSR(MELSR,1), NODISR(MELISR,1),
     + NODRN(MELRN,1), NODIRN(MELIRN,1), PRPIRN(MMTIRN,1),
     + MTIRN(1), NODPD(1), PRSDS(MPROS,1),
     + NREST(1), NDCNL(MLOCA,1), DVDTL(MLOCA,1), CRDCL(MNDSH,1),
     + NDORD(1), IPNTR(1), ICONN(1), LD(1), CORLD(1),
     + TRFST(18,18), ENLWT(1), ECLWT(MELSH,1), FRCNL(1),
     + ENLAUL(MAULD,1), ECLAUL(MAULD,MELSH,1), ENLTP(MZNTP,1),
     + ECLTP(MZNTP,MELSH,1), INDIC(1), ASECT(NDFSC), FSECT(NDFSC),
     + FCNTR(12), DISPL(1260), TBRAC(6,12), NDJNC(1)
C
      COMPLEX  ENR, ENRM1, ARG,
     + ENLSA(4,1), ECLSA(4,MELSH,1), FBNIF(NDFIF), KBNIF(MDFIF,MDFIF),
     + ABNIF(NDFIF)
C
      EQUIVALENCE ( KCOND(1,1), TRFST(1,1) )
C
      COMMON  STRTYP, AUTGN, SLDSP, SLRAC, SLSTS, SLBKL,
```

```
     +  PLSTR, PLMSH, PLDSP, PLSTS, PLMSP,
     +  MSECT, MSTRN, MNDSR, MRING, MNDRN, MGAUS,
     +  NNDSH, NELSH, NMTSH, NSTRN, NNDSR, NELSR, NMTSR, NIMSR,
     +  NELISR, NMTISR, NRING, NNDRN, NELRN, NMTRN, NIMRN, NELIRN,
     +  NMTIRN, NSRAC, NELBR, NMTBR, NGAUS, NCRSM, NPRDS, NLOCA,
     +  NSECT, NRWEL, NRWND, NDFND, NNDE1, NNDE2, NNDIN,
     +  NNTSH, NBTSH, NGTSH, NSTSSH, NKCTND, NNTSR, NBTSR, NGTSR,
     +  NTFSR, NSTSSR, NNTISR, NBTISR, NGTISR, NTFISR, NSTSIS, NNTRN,
     +  NBTRN, NGTRN, NTFRN, NSTSRN, NNTIRN, NBTIRN, NGTIRN, NTFIRN,
     +  NSTSIR, NTFBR, NSTSBR, NKRWPD, NCRLD, NFRCN, NABIF, NASCT,
     +  NACNT, NFBSC, NRACT, NKSCT, NKSCF, NKKGW, NKGSC, NKGICL, NKBNIF,
     +  NKGBNI, NEGVC, NKKGCN, NKGCN, NMSND, NMSCT, NK0EE, NKG0EE,
     +  CSDCL, CSDMC, MCCDD, ILDCA, LCNLD, LDTLD, LZNTP, LZNSL,
     +  DSKF1, DSKF2, IN, IO,
     +  FEED, RTOP, REAS, HITE, ELEV, THATAD,
     +  TITLE(30), LDCAS(2), NCNLD(2), NDTLD(2), NZNTP(2), NZNSL(2),
     +  RATIO(14), INDEX(45), NDCON(200), NDEX1(15), NDEX2(15),
     +  NDINT(15), INTND(35), WKAREA(192)
C
C
C
C
      DATA  (GRDSR(I), I=1, 9) / 1, 2, 4, 5, 7, 9, 10, 9, 11 /
C
      DATA  (GRDRN(I), I=1, 9) / 1, 3, 4, 6, 7, 9, 12, 8, 11 /
C
      DATA  (GRDBR(I), I=1, 6) / 1, 4, 7, 9, 8, 3 /
C
C*    I N I T I A L I Z A T I O N S:
C
      NNDEX = NNDSH - NNDIN
      NDFEX = 2 * NDFIF
      NDFIN = NDFSC - NDFEX
      NDFIF = NDFND * NNDE1
      NINFA = NSECT
C
      TETAR = 2.0 * ACOS(-1.0) / NINFA
C*    IOP = 0  FOR NORMAL OPERATION PROCESS (BY SKYSOL).
      IOP = 0
C*    IBX = -1  FOR NO BACKSUBSTITUTION (BY SKYSOL).
      IBX = -1
      IREF = -1
      IAFILE = 0
      IABIF = 0
      IASCT = 0
C
C*    INITIALIZATIONS.
C
C*    IF THE DISPLACEMENTS, OR THE BUCKLING SOLUTION HAS NOT BEEN
C*    REQUESTED, SKIP.
```

```
C
      IF (SLOSP . NE . 3HYES . OR . SLSKL . NE . 3HYES)   GO TO 90
C
      IF (SLOLD . EQ . 3HYES)                              WRITE(IO,30)
   30 FORMAT(1H0, //,T38, '     K K    DISPLACEMENTS DUE TO THE',
     + ' DEAD LOAD ALONE    < <      '//)
C
      IF (SLOLD . EQ . 2HNO)                               WRITE(IO,60)
   60 FORMAT(1H0, //,T30, '     K K    DISPLACEMENTS DUE TO THE',
     + ' DEAD AND THE LIVE LOADS COMBINED    < < '//)
C
C***  GENERATE VECTOR ROWSA(2).
C***  IF ROWSA(1) = 1, ANTISYMMETRIC-SINE LOADS EXIST (EITHER
C***  INDEPENDENTLY OR FROM AFRO LOADS).
C***  IF ROWSA(2) = 1, SYMMETRIC-SINE LOADS EXIST (EITHER INDEPENDENTLY
C***  OR FROM AFRO LOADS).
C***  IF ROWSA(1) = 1 AND ROWSA(2) = 1, BOTH THE ANTISYMMETRIC-SINE
C***  AND SYMMETRIC-SINE  AND/OR THE AFRO LOAD MAY EXIST.
C
   90 CALL ROWSNA (ILOCA, LOTLD, MLOCA, TYDTL, RFDTL, ROWSA)
C
C
C***  IF CALCULATION OF REACTIONS HAS BEEN REQUESTED, WRITE THE FOURIER
C***  COEFFICIENTS OF ALL EQUIVALENT NODAL LOADS, AS WELL AS THE THERMAL
C***  EQUIVALENT NODAL LOADS ON DISK FILE TFESC. THESE COEFFICIENTS
C***  WILL LATER BE USED TO CALCULATE THE ACTUAL EQUIVALENT
C***  NODAL LOADS CORRESPONDING TO A DEGREE OF FREEDOM FOR WHICH A
C***  A DISPLACEMENT IS PRESCRIBED. THEN THE FOURIER COEFFICIENTS, AND
C***  THE THERMAL EQUIVALENT NODAL LOADS CORRESPONDING TO SUCH DEGREES
C***  OF FREEDOM ARE SET TO ZERO, EXCEPT FOR SUCH ENTRIES IN ENLWT
C***  WHICH ARE REPLACED WITH THE ACTUAL PRESCRIBD DISPLACEMENTS. THE
C***  SAME ZEROING OF THE PARTICULAR ENTRIES IS DONE TO THE CORRECTIVE
C***  LOAD VECTOR, CORLD. FOR THE INTERFACE CONCENTRATED NODAL LOADS,
C***  NOW ON DISK FILE TFRCN, THE ZEROING PROCESS HAS ALREADY BEEN
C***  CARRIED OUT IN SUBROUTINE NDLDCN.
C
      CALL MDFFSN (DSKF1, ILOCA, LOTLD, LZNTP, MLOCA, MPRDS,
     + MAULD, MZNTP, NPRDS, NNDE2, NNDIN, NDFSC, NCRLD, NFBSC, SLRAC,
     + NODPD, PRSDS, RSDOF, NREST, TYDTL, RFDTL, NDEX1, NDEX2, NDINT,
     + INTND, NDORD, ROWSA, CORLD, ENLWT, ENLAUL, ENLSA, ENLTP)
C
C
C
C                    S T A T I C    C O N D E N S A T I O N
C
C
C***  CONDENSE OUT THE FOURIER COEFFICIENTS OF THE INTERNAL EQUIVALENT
C***  NODAL LOADS FOR CIRCUMFERENTIALLY CONSTANT LOADS (AS DEFINED
C***  ABOVE), ANTISYMMETRIC-UNIFORM , ANTISYMMETRIC-LINEAR, SYMMETRIC-
C***  SINE, ANTISYMMETRIC SINE, AND THE AFRO DISTRIBUTED LOADS, AND
C***  THE THERMAL LOADS, IF ANY.
```

```
C
      CALL CNDENL (ILDCA, LDTLD, LZNTP, MELSH, MLDCA, MAULD,
     + MZNTP, NSECT, NNDF1, NNDIN, NDFIN, NDFEX, NDFIF, NDFSC,
     + NDEX1, NDEX2, NDINT, INTND, NDORD,
     + TYDTL, RFDTL, LD, ROWSA, KSECT, CORLD, ENLWT,
     + ENLAUL, ENLSA, ENLTP, ASECT, FSECT, WKAREA)
C
C***  SET UP VECTOR INDIC. INDIC(N) INDICATES IF THE N-TH FOURIER
C***  COEFFICIENT OF THE INTERFACE EQUIVALENT NODAL LOADS IS ZERO OR
C***  NOT.
C
      CALL INDICT (ILDCA, LCNLD, LDTLD, LZNTP, MLDCA, NLDCA,
     + NSECT, NCNLD, NDTLD, NZNTP, SCNLI, SCNLJ, TYDTL, RFDTL,
     + SCTPI, SCTPJ, ROWSA, INDIC)
C
C
C
C                F O U R I E R   C O E F F I C I E N T S   O F
C
C     I N T E R F A C E   L O A D S  K  D I S P L A C E M E M T S
C
C
C*    TO CALCULATE FOURIER DISPLACEMENTS, FIRST ALL NONZERO FOURIER
C*    COEFFICIENTS OF THE INTERFACE EQUIVALENT NODAL LOADS ARE OBTAINED,
C*    AND THEN THE CORRESPONDING DISPLACEMENTS ARE SOLVED FOR AND PUT ON
C*    DISK FILE TABIF.
C
C*    CONSIDER ALL INTERFACES:
      DO 100 N = 1 , NINFA
C
C     IF INDIC(N) IS ZERO, THEN THE N-TH FOURIER COEFFICIENT OF THE
C*    INTERFACE LOADS IS ZERO, AND SOLUTION FOR THE CORRESPONDING
C*    FOURIER COEFFICIENTS OF DISPLACEMENTS IS NOT NECESSARY.
C
      IF (INDIC(N) . EQ . 0)                                  GO TO 100
C
C
C*                                                  I N F
C
C***  CALCULATE VECTOR FORCE(NLDOF), THE N-TH FOURIER COEFFICIENT OF THE
C***  NINFA INTERFACE EQUIVALENT NODAL LOAD VECTORS.
C
      CALL FRNINF (CGSKFI, ILDCA, LCNLD, LDTLD, LZNTP, MELSH,
     + MLDCA, MAULD, MZNTP, N, NINFA,
     + NDFIF, NFDOF, NDFSC, NDFIN, NDFEX,
     + TYDTL, RFDTL, LD, NSECT, SCNLI, SCNLJ, SCTPI, SCTPJ,
     + ENLWT, KSECT, ENLAUL, ENLSA, ENLTP, FRNIF, INDIC)
C
C
C
```

```
C*     IF INDIC(*) HAS BEEN SET TO ZERO, IT IS SUBSTITUTED FROMINF, AVOID
C*     SOLVING EQUATION FOR THE CORRECT D -AS IT IS ZERO.
C
      IF (INDIC(N) . EQ . 0)                          GO TO 100
C
C
C
C
C
C
C
C***  GENERATE THE CONDENSED MATRIX  KBNIF(NDFIF,NDFIF)  DIRECTLY
C***  [illegible] THE FULL ROW AND COLUMN IS TO:
C
C        KBNIF(I,J) =
C     +  KE22(NDFIF,NDFIF) + KE21(NDFIF,NDFIF) + KE21(NDFIF,NDFIF) *
C     +     [illegible] / NINFA) + KE12(NDFIF,NDFIF) * CEXP(I*2*PI*N/NINFA)
C
C***  NOTE, HERE KE2, KE1, KE21, AND KE12 ARE NOT CONDENSED. ALSO
C***  NOTE, THAT THE ASSEMLY OF KE2 OF SECTOR N WITH KE1 OF SECTOR
C***  (N+1) IS IMPLIED IN THE ABOVE OPERATION.
C
      CALL KBNINF (NDFIF, NNDSH, NNDE1, NDFND, NNDEX, NDFIN,
     + NDFIF, NINFA, N, NDEX1, NDEX2, INTND, SEGND, NDORD, IPNTR,
     + ICONN, LD, KSECT, KBNIF)
C
C
C
C*     IF NO INTERNAL NODES (HENCE DISPLACEMENTS) EXIST, THERE IS
C*     NO STATIC CONDENSATION.
C
      IF (NNDIN . EQ . 0)                             GO TO 95
C
C
C
C
C
C                                  K  O  E  E  S
C
C
C***  COMPLETE THE GENERATION OF KBNIF(NDFIF,NDFIF), BY SUBTRACTION
C***  OF THE TERM
C                 KOEES(NDFIF,NDFIF) = KE1 * KII-INV * KIE
C***  ARISING FROM THE STATIC CONDENSATION OF THE INTERNAL DEGREES OF
C***  FREEDOM. NOTE THAT THE ASSEMBLY OF KE2 OF SECTOR N AND KE1 OF
C***  SECTOR (N+1) IS CARRIED OUT SIMULTANEOUSLY.
C
      CALL KBNIFO (OSKE2, NDFIF, N, NDFEX, NINFA, NDFIF, NKOEE,
     + CORLD, KBNIF)
C
```

```
C
C
C                                   A  B  N  I  F
C
C***  NOW, SOLVE FOR ABNIF, THE N-TH FOURIER COEFFICIENT VECTOR OF
C***  THE NINFA INTERFACE DISPLACEMENT VECTORS.
C
C              KBNIF(NDFIF,NDFIF) * ABNIF(NDFIF) = FBNIF(NDFIF)
C
C***  ON OUTPUT, THE SOLUTION VECTOR, ABNIF, REPLACES FBNIF, WHICH IS
C***  EQUIVALENCED TO ABNIF.
C
C***  THE SOLUTION IS DONE IN TWO STEPS BY TWO ROUTINES FROM THE
C***  LINPAK LIBRARY.
C
C
C***  FACTORIZE KBNIF.
C
   95 CALL CGEFA (KBNIF, MDFIF, NDFIF, IPVT, INFO)
C
C
C***  COMPLETE SOLUTION.
C
      CALL CGESL (KBNIF, MDFIF, NDFIF, IPVT, FBNIF, 0)
C
      IABIF = IABIF + 1
C
C*    PUT THE N-TH FOURIER COEFFICIENT OF THE INTERFACE DISPLACEMENTS
C*    ON DISK FILE TABIF:
C
C     WRITE (DSKF1, REC=NABIF+IABIF)  ABNIF
C
C***  TO DO THIS CALL SUBROUTINE WRITEA, WHICH USES A REAL VECTOR
C***  EQUIVALENT (IN NUMBER OF WORDS) TO THE COMPLEX MATRIX PASSED
C***  TO IT.
C***  THE REASON FOR DOING THIS IS THAT THE CURRENT CDC-FTN5 COMPILER
C***  IS NOT CAPABLE OF WRITING A COMPLEX VECTOR OR MATRIX ON DISK FILE.
C
      CALL WRITEA (IABIF, NDFEX, NABIF, DSKF1, ABNIF)
C
C
  100 CONTINUE
C
C
C
C                              D I S P L A C E M E N T S
C
C
C*    NOW, THAT THE FOURIER COEFFICIENTS OF THE INTERFACE DISPLACEMENTS
C*    ARE AVAILABLE (ON DISK FILE TABIF), CALCULATE THE ACTUAL DISPLACE-
C*    MENTS, FOR THE SECTORS FOR WHICH SOLUTION HAS BEEN REQUESTED.
```

```
C
C*    CONSIDER ALL SECTORS ZONES FOR WHICH SOLUTION IS REQUESTED:
C
      DO 1700 IZNSL = 1 , LZNSL
C
      ITIME = 0
C
C*    FIND THE BEGINNING AND ENDING SECTORS FOR THE CURRENT SOLUTION
C*    ZONE:
C
      JSC = SLVSC(ILDCA,IZNSL,1)
      KSC = SLVSC(ILDCA,IZNSL,2)
C
C*    IF JSC K KSC, CONSIDER 2 RANGES.
C
      DO 1600 IRANG = 1 , 2
C
      IF (SLVSC(ILDCA,IZNSL,1).LE.SLVSC(ILDCA,IZNSL,2))  GO TO 150
C
      JSC = SLVSC(ILDCA,IZNSL,1) * (2-IRANG) + IRANG - 1
      KSC = NSECT * (2-IRANG) + SLVSC(ILDCA,IZNSL,2) * (IRANG-1)
C
C*    CONSIDER ALL SECTORS IN THE CURRENT SOLUTION ZONE:
C
  150 DO 1500 ISC = JSC , KSC
C
      R = ISC
C
      NDFCT = 0
C
  175 IASCT = IASCT + 1
C
C
C
C
C                 I N T E R F A C E    D I S P L A C E M E N T S
C
C
C
      DO 200 I = 1 , NDFSC
      ASECT(I) = 0.0
  200 CONTINUE
C
C
C*    CALCULATE THE EXTERNAL DISPLACEMENTS OF SECTOR R, I.E., DISPLACE-
C*    MENTS OF INTERFACES (R-1) AND R.
C
C     FOR SECTOR R :
C
C       ASECT(NDFIF) = SIGMA (ABNIF * CEXP (I * 2 * PI * R * N/NINFA))
C
C                          WITH,    N = 1 ... NINFA
```

```
C
      IABIF = 0
      DO 400 N = 1 , NINFA
C
      IF (INDIC(N) . EQ . 0)                          GO TO 400
C
      ARG = CMPLX(0.0, TETAR * R * N)
      ENR = CEXP(ARG)
C
      ARG = CMPLX(0.0, TETAR * (R-1) * N)
      ENRM1 = CEXP(ARG)
C
      IABIF = IABIF + 1
C
C*    READ THE N-TH FOURIER COEFFICIENTS OF THE INTERFACE DISPLACEMENTS.
C
C     READ (DSKF1, REC=NABIF+IABIF)  ABNIF
C
C***  TO DO THIS CALL SUBROUTINE READA, WHICH USES A REAL VECTOR
C***  EQUIVALENT (IN NUMBER OF WORDS) TO THE COMPLEX MATRIX PASSED
C***  TO IT.
C***  THE REASON FOR DOING THIS IS THAT THE CURRENT CDC-FTN5 COMPILER
C***  IS NOT CAPALBE OF READING A COMPLEX VECTOR OR MATRIX FROM DISK
C***  FILE.
C
      CALL READA (IABIF, NOFEX, NABIF, DSKF1, ABNIF)
C
C
C*    ADD THE CONTRIBUTION OF THE CURRNET FOURIER DISPLACEMENTS OF
C*    THE INTERFACES TO ASECT.
C*    NOTE, IF R = 1, (R - 1) IS EQUAL TO ZERO, AND IS EQUIVALENT TO  N.
C
C*    CONSIDER EXTERNAL 1 AND EXTERNAL 2 NODES.
C
      DO 360 IRWND = 1 , NRWND
C
      LAB = 12 * (NRWND - IRWND)
C
      IDR1 = NDORD( NDEX1(IRWND) )
      IDR2 = NDORD( NDEX2(IRWND) )
C
      L1 = 12 * (IDR1 - 1)
      L2 = 12 * (IDR2 - 1)
C
C*    INTERFACE (R-1) NODES.
C
      DO 300 I = 1 , 12
      ASECT(L1+I) = ASECT(L1+I) + ABNIF(LAB+I) * ENRM1
  300 CONTINUE
C
C*    INTERFACE R NODES.
```

```
C
      DO 310 I = 1 , 12
      ASECT(L2+I) = ASECT(L2+I) + ABNIF(LA8+I) * ENR
  310 CONTINUE
C
  360 CONTINUE
C
  400 CONTINUE
C
C
C
C*    IF NO INTERNAL NODES EXIST, SKIP.
C
      IF (NNDIN . EQ . 0)                          GO TO 425
C
C
C
C       I N T E R N A L   E Q U I V A L E N T   N O D A L   L O A D S
C
C
C***  BEFORE CALCULATING THE DISPLACEMENTS OF INTERNAL NODES,
C***  CALCULATE THE INTERNAL EQUIVALENT LOADS OF THE CURRENT SECTOR.
C***  NOTE THAT EXCEPT FOR THE SYMMETRIC AND ANTISYMMETRIC SINE, AND
C***  AFRO LOADINGS, WHOSE EQUIVALENT LOADS SHOULD BE CALCULATED FROM
C***  THEIR FOURIER COEFFICIENTS, THE EQUIVALENT NODAL LOADS DUE TO
C***  THE OTHER TYPES OF LOADING ARE READILY AVAILABLE.
C***  ON RETURN, THE INTERNAL ENTRIES OF FSECT CONTAIN THE INTERNAL
C***  EQUIVALENT NODAL LOADS OF SECTOR R.
C
      CALL ENLINT (ILDCA, LCNLD, LDTLD, LZNTP, MNDSH, MLDCA,
     + MCNLD, MAULD, MZNTP, NPROS, NSECT, NDFSC, NNDEX, NNDIN, NDFEX,
     + NDFIN, R, CROCL, NDINT, INTND, NDORD, NDDPD, SCNLI, SCNLJ,
     + NDCNL, ITCNL, TYDTL, RFDTL, DVDTL, SCTPI, SCTPJ, ROWSA, ENLWT,
     + FRCNL, ENLAUL, ENLSA, ENLTP, FSECT)
C
C
C       I N T E R N A L    D I S P L A C E M E N T S
C
C
C*    PUT THE EXTERNAL DISPLACEMENTS IN THE EXTERNAL ENTRIES OF FSECT.
C*    THIS IS REQUIRED IN CALCULATING THE DISPLACEMENTS OF THE INTERNAL
C*    NODES BY SKYSOL.
C
C
C*    CONSIDER ALL NODES OF THE SECTOR.
C
      DO 420 INDSH = 1 , NNDSH
C
C*    IF INDSH IS AN INTERNAL NODE, SKIP.
C
      IF (INTND(INDSH) . EQ . 3HYES)               GO TO 420
```

```
C
      IORD = NDORD(INDSH)
      LA = 12 * (IORD - 1)
C
      DO 410 I = 1 , 12
      FSECT(LA+I) = ASECT(LA+I)
  410 CONTINUE
C
  420 CONTINUE
C
C
C*** TO SOLVE FOR THE DISPLACEMENTS OF THE INTERNAL NODES OF SECTOR
C*** R, SUBROUTINE SKYSOL WILL BE USED AGAIN. BEFORE SOLUTION, THE
C*** EXTERNAL ENTRIES OF FSECT(NDFSC) CONTAIN THE EXTERNAL
C*** DISPLACEMENTS OF THE R-TH AND THE (R-1)-TH INTERFACES, AND THE
C*** INTERNAL ENTRIES OF IT CONTAIN THE INTERNAL EQUIVALENT NODAL
C*** LOADS OF SECTOR R. (IOP = 0; ISX = -1)
C*** THUS THE DISPLACEMENTS OF THE INTERNAL NODES ARE SOLVED FOR
C*** FROM
C
C          ASECT(INTERNAL) = K-INT(NDFIN,NDFIN)-INVERSE *
C     ( FSECT(INTERNAL) - K-INEX(NDFIN,NDFEX) * ASECT(EXTERNAL) )
C
C*** ON RETURN, THE INTERNAL ENTRIES OF ASECT WILL CONTAIN THE
C*** INTERNAL DISPLACEMENTS, AND THE EXTERNAL ENTRIES OF IT WILL
C*** CONTAIN THE EXTERNAL DISPLACEMENTS.
C
C
      CALL SKYSOL (KSECT, NDFSC, LD, IOP, ISX, FSECT, ASECT,
     + IREF, IAFILE, WKAREA, DELTA)
C
  425 IF (ITIME . NE . 0)                                  GO TO 1400
C
C*    IF NEITHER DISPLACEMENT ANALYSIS NOR STRESS ANALYSIS NOR
C*    BUCKLING ANALYSIS HAS BEEN REQUESTED, DO NOT SOLVE FOR THE
C*    CENTROIDAL DISPLACEMENTS.
C
      IF (SLOSP . NE . 3HYES . AND . SLSTS . NE . 3HYES . AND .
     + SLBKL . NE . 3HYES)                                 GO TO 1500
C
C     IF THERE ARE NO SHELL ELEMENTS, DO NOT SOLVE FOR CENTROIDAL
C*    DISPLACEMENTS.
C
      IF (NELSH . EQ . 0)                                  GO TO 550
C
C
C           C E N T R O I D A L   D I S P L A C E M E N T S
C
C                  S E C O N D   D E R I V A T I V E S
C
C
```

```
C
C*    CALCULATE THE CENTROIDAL DISPLACEMENTS (OR THE CENTROIDAL AND THE
C*    2ND DERIVATIVES OF THE NODES OF THE JUNCTURE) OF ALL THE ELEMENTS
C*    WITHIN THE SECTOR, R, AND PRINT THEM. IF STRESS ANALYSIS IS
C*    REQUESTED, SAVE THESE CENTROIDAL DISPLACEMENTS OF EACH SECTOR ON
C*    DISK FILE DSKF1. TO CALCULATE THE CENTROIDAL DISPLACEMENTS, FIRST
C*    THE CENTROIDAL EQUIVALENT NODAL LOADS MUST BE CALCULATED.
C
C*    CONSIDER ALL SHELL ELEMENTS OF THE CURRENT SECTOR, R.
C
      DO 500 IELSH = 1 , NELSH
C
C*    DETERMINE IF ANY OF THE NODES OF THE CURRENT ELEMENT BELONG TO
C*    A JUNCTURE OF TWO SHELLS WITH DIFFERENT MERIDIONAL GEOMETRY.
C
      NDFCD = 2
C
      IF (NDJNC(NODSH(IELSH,2)) . EQ . 3HYES)     NDFCD = 12
C
      IF (NDJNC(NODSH(IELSH,1)) . EQ . 3HYES)     NDFCD = 7
C
C
      IF (NDFCD . NE . 2)                         GO TO 435
C
C
C*    READ KCDND(2,36) FROM DISK FILE DSKF1.
C
      READ (DSKF1, REC=NKCTND+IELSH)  KCDND
C
C
      DO 430 J = 1 , 2
      DO 430 I = 1 , 2
C
      KCDIV(I,J) = KCINV(IELSH,I,J)
C
  430 CONTINUE
C
      GO TO 455
C
C
C*     S E C O N D   D E R I V A T I V E S   C O N D E N S E D
C
C
C*    READ KCDND(NDFCD,36) AND KCDIV(NDFCD,NDFCD) FROM DISK FILE
C*    DSKF1.
C
  435 READ (DSKF1, REC = NKCTND + IELSH)  KCDND, KCDIV
C
C
C
C
```

```
C     C E N T R O I D A L   E Q U I V A L E N T   N O D A L   L O A D S
C
C
C***  CALCULATE THE EQUIVALENT NODAL LOADS OF THE CURRENT ELEMENT,
C***  IELSH, AND PUT THEM IN THIER CORRESPONDING ENTRIES OF VECTOR
C***  CORLD (USED ONLY AS STORAGE). NOTE, EXCEPT FOR THE SYMMETRIC-
C***  SIN, THE ANTISYMMETRIC-SINE, AND THE AERO LOADINGS, WHOSE
C***  EQUIVALENT NODAL LOADS ARE CALCULATED FROM THIER FOURIER
C***  COEFFICIENTS, THE EQUIVALENT NODAL LOADS DUE TO THE OTHER
C***  TYPES OF LOADING ARE READILY AVAILABLE.
C
  455 CALL FCENTR (IELSH, ILOCA, LCNLD, LDTLD, LZNTP, MELSH,
     + MLOCA, MCNLD, MAULD, MZNTP, NSECT, NDFCD, R, TYDTL, RFDTL, DVDTL,
     + SCTPI, SCTPJ, ROWSA, ECLWT, ECLAUL, ECLSA, ECLTP, FCNTR)
C
C
C           C E N T R O I D A L   D I S P L A C E M E N T S
C
C
C
C
C
C
C
C***  CALCULATE THE CENTROIDAL DISPLACEMENTS OF THE CURRENT ELEMENT,
C***  IELSH FROM
C
C       ACNTR(NDFCD) = KCINV(NDFCD,NDFCD) *
C                    ( FCNTR(NDFCD) - KCOND(NDFCD,NDFUC) * ASHEL(NDFUC) )
C*    NDFUC IS NUMBER OF DEGREES OF FREEDOM UNCONDENSED, AND IS
C*    NDFUC = 38 - NDFCD.
C
C***  AND PLACE THEM IN VECTOR CORLD(NDFSC) (USED ONLY AS A
C***  TEMPORARY STORAGE) IN THE SEQUENTIAL ORDER OF THE ELEMENTS.
C
        CALL ACENTR (IELSH, IO, MELSH, NDFCD, NDFCT, NODSH,
     +  NDORD, KCOND, KCDIV, ASECT, FCNTR, CORLD)
C
C
C
  500 CONTINUE
C
C
C*    NCDSH IS THE NUMBER OF CONDENSED GENERALIZED DISPLACEMENTS OF
C*    SHELL ALONE.
C
  550 NCDSH = NDFCT
C
        IF (SLOSP . NE . 3HYES . OR . SLBKL . NE . 3HYES . OR .
     + SLOLD . EQ . 3HYES)                                  GO TO 650
C
```

```
C
C
C*    IF THE DISPLACEMENTS JUST SOLVED FOR ARE DUE TO THE LIVE LOADS
C*    ALONE, READ THE NODAL AND THE CENTROIDAL DISPLACEMENTS OF THE
C*    CURRENT SECTOR DUE TO THE DEAD LOAD BEFORE THEY ARE OVERWRITTEN.
C*    (VECTORS FSECT AND DISPL ARE USED AS A TEMPORARY STORAGE)
C
      READ (DSKF1, REC=NASCT+IASCT)  FSECT
C
C
      IF (NELSH . EQ . 0)                                  GO TO 650
C
      IF (SLSTS . NE . 3HYES . AND . SLBKL . NE . 3HYES . AND .
     + PLDSP . NE . 3HYES)                                 GO TO 650
C
      IF (NDFCT . LE . 1260)                               GO TO 600
C
      PRINT*, ' DIMENSION OF VECTOR',
     + ' DISPL MUST BE INCREASED FROM 1260 TO NDFCT =',NDFCT,
     + ' ISSUED FROM SUBROUTINE SLVASC, SHORTLY BEFORE DELIBERATE',
     + ' ABORTION OF THE PROGRAM'
C
      STOP
C
C
600 READ (DSKF1, REC = NACNT + IASCT)  (DISPL(I), I = 1, NDFCN)
C
C
650 IF (SLRAC . EQ . 2HNO . AND . SLSTS . EQ . 2HNO . AND .
     + SLBKL . EQ . 2HNO . AND . PLDSP . EQ . 2HNO)  GO TO 760
C
C*    IF REACTION OR STRESS OR BUCKLING ANALYSIS OR PLOTTING OF THE
C*    DISPLACEMENTS HAS BEEN REQUESTED, WRITE THE NODAL
C*    DISPLACEMENTS OF THE CURRENT SECTOR ON DISK FILE DSKF1.
C
      WRITE (DSKF1, REC = NASCT + IASCT)  ASECT
C
C*    IF SOLUTION OF DISPLACEMEMTS HAS NOT BEEN REQUESTED, DO NOT PRINT
C*    THEM.
C
760 IF (SLDSP . NE . 3HYES)                                GO TO 1350
C
      IF (SLBKL . NE . 3HYES . OR . SLDLD . EQ . 3HYES) GO TO 950
C
C*    ADD THE NODAL DISPLACEMENTS DUE TO THE DEAD
C*    LOAD TO THOSE DUE TO THE LIVE LOAD.
C
      DO 800 I = 1 , NDFSC
      ASECT(I) = ASECT(I) + FSECT(I)
800 CONTINUE
C
```

```
5406          IF (NELSH . EQ . 0)                                GO TO 950
5407    C
5408          DO 900 I = 1 , NDFCT
5409          DISPL(I) = DISPL(I) + CORLD(I)
5410       900 CONTINUE
5411    C
5412    C
5413    C
5414    C***  PRINT THE NODAL DISPLACEMENTS OF THE CURRENT SECTOR.
5415    C
5416       950 CALL PRNASC (ID, NNDSH, P, NDORD, ASECT)
5417    C
5418          IF (NELSH . EQ . 0)                                GO TO 1005
5419    C
5420    C
5421    C***  PRINT THE CENTROIDAL DISPLACEMENTS OF THE CURRENT SECTOR.
5422    C
5423          IF (SLBKL . NE . 3HYES . OR . SLOLD . EQ . 3HYES)
5424         + CALL PRNACT (IELSH, ID, MELSH, NELSH, R, NODSH, NDJNC, CORLD)
5425    C
5426          IF (SLBKL . EQ . 3HYES . AND . SLOLD . NE . 3HYES)
5427         + CALL PRNACT (IELSH, ID, MELSH, NELSH, R, NODSH, NDJNC, DISPL)
5428    C
5429    C
5430     1005 IF (NELSR . EQ . 0)                                GO TO 1050
5431    C
5432    C
5433    C                S T R I N G E R     D I S P L A C E M E N T S
5434    C
5435    C
5436    C
5437          TYPEL = 6HSTRING
5438    C
5439    C***  CALCULATE THE NODAL DISPLACEMENTS OF ALL STRINGER ELEMENTS
5440    C***  OF THE CURRENT SECTOR, IN THEIR LOCAL COORDINATES, FROM THE
5441    C***  CORRESPONDING SHELL NODE DISPLACEMENTS, ACCORDING TO THE
5442    C***  FOLLOWING TRANSFORMATION
5443    C
5444    C               ASTRN(18) = TRFSR(18,18) * ASHEL(18)
5445    C
5446    C***  AND PRINT THEM.
5447    C
5448          CALL DSPSTR (DSKP1, ID, MELSR, NELSR, NTFSR, NDFCT,
5449         + SLOLD, SLBKL, TYPEL, NODSR, NDORD, NDJNC, ORDSR, TRFST, ASECT,
5450         + CORLD)
5451    C
5452    C
5453     1050 IF (NELISR . EQ . 0)                               GO TO 1150
5454    C
5455    C
5456    C               I N T E R M E D I A T E     S T R I N G E R S
```

```
C
C
      TYPEL = 6HIMSTRN
C
C
C***  CALCULATE THE NODAL DISPLACEMENTS OF ALL INTERMEDIATE STRINGER
C***  ELEMENTS OF THE CURRENT SECTOR, IN THEIR LOCAL COORDINATES, FROM
C***  THE CORRESPONDING SHELL NODE DISPLACEMENTS, ACCORDING TO THE
C***  FOLLOWING TRANSFORMATION
C
C           ASTRN(18) = TRFSR(18,18) * ASHEL(18)
C
C***  AND PRINT THEM.
C
      CALL DSPSTR (DSKF1, IO, MELISR, NELISR, NTFISR, NDFCT,
     + SLOLD, SL8KL, TYPEL, NODISR, NOORD, NDJNC, ORDSR, TRFST, ASECT,
     + CORLD)
C
 1150 IF (NELRN . EQ . 0)                          GO TO 1250
C
C
C                R I N G     D I S P L A C E M E N T S
C
C
      TYPEL = 4HRING
C
C***  CALCULATE THE NODAL DISPLACEMENTS OF ALL RING ELEMENTS OF THE
C***  CURRENT SECTOR, IN THEIR LOCAL COORDINATES, FROM THE CORRESPONDING
C***  SHELL NODE DISPLACEMENTS, ACCORDING TO THE FOLLOWING
C***  TRANSFORMATION
C
C           ARING(18) = TRFRN(18,18) * ASHEL(18)
C
C***  AND PRINT THEM.
C
      CALL DSPRNG (DSKF1, IO, MELRN, NELRN, NTFRN, NDFCT,
     + SLOLD, SL8KL, TYPEL, NODRN, NOORD, NDJNC, ORDRN, TRFST, ASECT,
     + CORLD)
C
C
 1250 IF (NELIRN . EQ . 0)                         GO TO 1255
C
C
C      I N T E R M E D I A T E   R I N G   D I S P L A C E M E N T S
C
C
      TYPEL = 6HIMRING
C
C***  CALCULATE THE NODAL DISPLACEMENTS OF ALL INTERMEDIATE RING
C***  ELEMENTS OF THE CURRENT SECTOR, IN THEIR LOCAL COORDINATES,
C***  FROM THE CORRESPONDING SHELL NODE COORDINATES, ACCORDING TO
```

```
C***  THE FOLLOWING TRANSFORMATION
C
C              ARING(18) = TFRN(18,18) * ASHEL(18)
C
C***  AND PRINT THEM.
C
      CALL DSPRNG (DSKF1, ID, MELIRN, NELIRN, NTFIRN, NDFCT,
     + SLDLO, SLBKL, TYPEL, NODIRN, NDORD, NOJNC, ORDRN, TRFST, ASECT,
     + CORLO)
C
C
 1255 IF (NELBR . EQ . 0)                          GO TO 1265
C
C
C
C
C                  B R A C E    D I S P L A C E M E N T S
C
C
C
C***  CALCULATE THE NODAL DISPLACEMENTS OF ALL BRACE ELEMENTS OF THE
C***  CURRENT SECTOR, IN THEIR LOCAL COORDINATES, FROM THE CORRESPONDING
C***  SHELL NODE COORDINATES, ACCORDING TO THE FOLLOWING TRANSFORMATION
C
C                  ABRAC(6) = TRFRN(6,12) * ASHEL(12)
C
C***  AND PRINT THEM.
C
      CALL DSPBRC (DSKF1, IO, SLDSP, MELIRN, MMTIRN, NELIRN,
     + NELBR, NTFBR, NSTSBR, NODIRN, PRPIRN, MTIRN, NDORD, ORDBR, TBRAC,
     + ASECT)
C
C
C
 1265 IF (NDFCT . EQ . 0)                          GO TO 1350
C
      IF (SLSTS . NE . 3HYES . AND . SLBKL . NE . 3HYES . AND .
     + PLDSP . NE . 3HYES)                         GO TO 1350
C
      IF (NDFCT . LE . 1260)                       GO TO 1266
C
      PRINT*, ' DIMENSION OF VECTOR',
     + ' DISPL MUST BE INCREASED FROM 1260 TO NDFCT =',NDFCT,
     + ' ISSUED FROM SUBROUTINE SLVASC, SHORTLY BEFORE DELIBERATE',
     + ' ABORTION OF THE PROGRAM'
C
      STOP
C
C
C***  IF NO BUCKLING SOLUTION HAS BEEN REQUESTED, SKIP.
C
```

```
 1266 IF (SLBKL . NE . 3HYES)                          GO TO 1300
C
C
C*    IF THE CURRENT SOLUTION IS FOR THE DEAD LOAD, SKIP.
C
      IF (SLDLD . EQ . 3HYES)                          GO TO 1300
C
C*    IF NO STIFFENERS EXIST, SKIP.
C
      IF (NELSR . EQ . 0 . AND . NELISR . EQ . 0 . AND . NELRN . EQ .
     +  0 . AND . NELIRN . EQ . 0)                     GO TO 1300
C
C*    AT THIS POINT, THE ENTRIES OF CORLD CORRESPONDING TO THE
C*    CONDENSED GENERALIZED DISPLACEMENTS  AT THE JUNCTURE (IF ANY)
C*    OF THE STIFFENERS ARE THOSE DUE TO DEAD LOAD PLUS LIVE LOAD
C*    COMBINED. THIS IS BECAUSE ASECT, WHICH WAS USED TO CALCULATE
C*    THESE CONDENSED DEGREES OF FREEDOM IS ITSELF THE COMBINED NODAL
C*    DISPLACEMENT VECTOR (I.E., DUE TO DEAD LOAD + LIVE LOAD). WE
C*    SHOULD NOW SUBTRACT THE CONDENSED GENERALIZED  DISPLACEMENTS
C*    DUE TO DEAD LOAD FORM THE COMBINED GENERALIZED DISPLACEMENTS,
C*    WRITING THEM ON DISK FILE DSKF1.
C
C
C*    FIRST CHECK ALL NODES OF INTERFACE 1 TO SEE IF ANY OF THE NODES
C*    IS A JUNCTURE NODE.
C
      DO 1270 IRWND = 1 , NRWND
C
      IF (NDJNC ( NDEX1(IRWND) ) . NE . 3HYES)     GO TO 1270
C
      GO TO 1280
C
 1270 CONTINUE
C
      GO TO 1300
C
C*    READ THE DEAD LOAD CONDENSED GENERALIZED DISPLACEMENTS FORM
C*    DISK FILE DSKF1.
C
 1280 READ (DSKF1, REC = NACNT + IASCT) (DISPL(I), I = 1, NDFCT)
C
C
      NDFP1 = NCOSH + 1
C
C*    NOW SUBTRACT THE DEAD LOAD GENERALIZED DISPLACEMENTS FORM THE
C*    TOTAL OF THE DEAD LOAD PLUS THE LIVE LOAD FOR THE STIFFENERS ONLY.
C
      DO 1290 I = NDFP1 , NDFCN
C
      CORLD(I) = CORLD(I) - DISPL(I)
C
```

```
 1290 CONTINUE
C
C
C
C*    IF STRESS OR BUCKLING ANALYSIS IS REQUESTED, WRITE THE CENTROIDAL
C*    DISPLACEMENTS OF THE CURRENT SECTOR ON DISK FILE DSKF1.
C
 1300 WRITE (DSKF1, REC=NACNT+IASCT)  (CORLD(I), I=1,NDFCT)
C
      NDFCN = NDFCT
C
 1350 IF (SLRAC . NE . 3HYES)                          GO TO 1500
C
      IF( ISC . NE . SLVSC(ILDCA,IZNSL,2) )            GO TO 1500
C
C*    IF CALCULATION OF REACTIONS HAS BEEN REQUESTED, COMPUTE THE
C*    DISPLACEMENTS OF THE TWO SECTORS ON EITHER SIDE OF THE CURRENT
C*    SOLUTION ZONE AND WRITE THEM ON DISK FILE DSKF1.
C
C*    THE ORDER IN WHICH THE NODAL DISPLACEMENTS OF THE SECTORS ARE
C*    WRITTEN IS:
C*    FIRST, THE DISPLACEMENTS OF THE CURRENT SOLUTION ZONE, JSC TO KSC;
C*    NEXT, IN THE FOLLOWING THE DISPLACEMENTS OF SECTORS JSC-1 AND
C*    KSC+1, RESPECTIVELY; THEN THE DISPLACEMENTS OF THE NEXT SOLUTION
C*    ZONE, IN THE SAME MANNER.
C
C
      ITIME = 1
C
 1400 R = (2 - ITIME) * ( SLVSC(ILDCA,IZNSL,1) - 1 ) +
     K    (ITIME - 1) * ( SLVSC(ILDCA,IZNSL,2) + 1 )
C
C*    R = 0 IS THE SAME AS R = NSECT; R = NSECT + 1 IS THE SAME AS R = 1
C
      IF ( R . EQ . 0 )                                 R = NSECT
      IF ( R . EQ . (NSECT+1) )                         R = 1
C
      IF (ITIME . EQ . 1)                               GO TO 1485
C
C*    WRITE THE NODAL DISPLACEMENTS OF SECTOR R ON DISK FILE DSKF1.
C*    NODAL DISPLACEMENTS.
C
      WRITE (DSKF1, REC=NASCT+IASCT)  ASECT
C
 1485 ITIME = ITIME + 1
C
      IF (ITIME . LT . 4)                               GO TO 175
C
 1500 CONTINUE
C
      IF (SLVSC(ILDCA,IZNSL,1).LE.SLVSC(ILDCA,IZNSL,2)) GO TO 1700
```

```
5661  C
5662   1600 CONTINUE
5663  C
5664   1700 CONTINUE
5665  C
5666         RETURN
5667         END
5668  C
5669  C
5670  C
5671  C     --------------------    R  O  W  S  N  A    --------------------
5672  C
5673  C
5674  C
5675  C***   THIS SUBROUTINES GENERATE VECTOR ROWSA(2).
5676  C***   IF ROWSA(1) = 1, ANTISYMMETRIC-SINE LOADS EXIST (EITHER
5677  C***   INDEPENDENTLY OR FROM AFRO LOADS).
5678  C***   IF ROWSA(2) = 1, SYMMETRIC-SINE LOADS EXIST (EITHER INDEPENDENTLY
5679  C***   OR FROM AFRO LOADS).
5680  C***   IF ROWSA(1) = 1 AND ROWSA(2) = 1, BOTH THE ANTISYMMETRIC-SINE
5681  C***   AND SYMMETRIC-SINE  AND/OR THE AFRO LOAD MAY EXIST.
5682  C
5683         SUBROUTINE ROWSNA (ILDCA, LDTLD, MLDCA, TYDTL, RFDTL, ROWSA)
5684  C
5685         INTEGER  TYPE, PROJ, TYDTL(MLDCA,1), RFDTL(MLDCA,1), ROWSA(2)
5686  C
5687         ROWSA(1) = 0
5688         ROWSA(2) = 0
5689  C
5690  C      CONSIDER ALL DISTRIBUTED LOADS IN THE CURRENT LOADING CASE.
5691  C
5692         DO 10 IDTLD = 1 , LDTLD
5693  C
5694         TYPE = TYDTL(ILDCA,IDTLD)
5695         PROJ = RFDTL(ILDCA,IDTLD)
5696  C
5697         IF ( TYPE . EQ . 4HSINE . AND . PROJ . EQ . -1 )  ROWSA(1) = 1
5698         IF ( TYPE . EQ . 4HSINE . AND . PROJ . EQ . 1 )   ROWSA(2) = 1
5699         IF ( TYPE . EQ . 4HAFRO )                         ROWSA(1) = 1
5700         IF ( TYPE . EQ . 4HAFRO )                         ROWSA(2) = 1
5701  C
5702     10 CONTINUE
5703  C
5704         RETURN
5705         END
5706  C
5707  C
5708  C
5709  C     --------------------    K  B  N  I  F  O    --------------------
5710  C
5711  C
```

```
C
C***  THIS SUBROUTINE COMPLETES THE GENERATION OF KBNIF(NDFIF,NDFIF),
C***  BY SUBTRACTION OF THE TERM
C***                KOEES(NDFEX,NDFEX) = KEI * KII-INV * KIE
C***  ARISING FROM THE STATIC CONDENSATION OF THE INTERNAL DEDREES OF
C***  FREEDOM, FROM KBNIF. THE COLUMNS OF KOEES(NDFEX,NDFEX) ARE READ
C***  FROM DISK FILE DSKF2. KOEES IS PARTITIONED AS
C
C
C
C                      -                    -
C                     I   KOE2   .   KOE21   I
C                     I . . . . . . . . . . I
C                     I          .         I
C                     I   KOE12  .   KOE1   I
C                     IK         .        KI
C
C
C***  THE CONDENSATION IS CARRIED OUT ACCORDING TO
C
C     KBNIF(NDFIF,NDFIF) = KBNIF(NDFIF,NDFIF) - ( KOE2(NDFIF,NDFIF) +
C     KOE1(NDFIF,NDFIF) ) -  KOE21(NDFIF,NDFIF) * CEXP(-I*2*PI*N/NINFA)
C     - KOE12(NDFIF,NDFIF) * CEXP(I*2*PI*N/NINFA)
C
C
C***  NOTE, THAT THE ASSEMBLY OF KE2 OF SECTOR N WITH WITH KE1 OF
C***  SECTOR (N+1) IS IMPLIED IN THE ABOVE OPERATION
C
      SUBROUTINE KBNIF0 (DSKF2, NDFIF, N, NDFEX, NINFA, NDFIF, NKOEE,
     + KOEE, KBNIF)
C
      INTEGER  DSKF2
C
      REAL  KOEE(NDFEX)
C
      COMPLEX  ARG, EN, ENCJ, CEXP, KBNIF(NDFIF,1)
C
C*    INITIALIZATIONS
      TETAR = 2.0 * 3.14159265359 / NINFA
      ARG = CMPLX(0.0, TETAR * N)
      EN = CEXP(ARG)
      ENCJ = CEXP(-ARG)
C
      IKOEE = 0
C
C*    CONSIDER ALL COLUMNS OF KOEES(NDFEX,NDFEX), I.E., KOEE(NDFEX)'S.
C
      DO 30 ICOL = 1 , NDFEX
C
      IKOEE = IKOEE + 1
C
```

```
C*    READ EACH COLUMN OF KOEES(NDFEX,NDFEX), FROM DISK FILE DSKF2, AND
C*    PLACE IT IN KOEE(NDFEX).
C
      READ (DSKF2, REC=NKOEE+IKOEE)  KOEE
C
C*    CARRY OUT THE CONDENSATION OF THE ICOL-TH COLUMN OF KBNIF,
C*    ACCORDING TO THE FORMULA GIVEN IN THE SUBROUTINE HEADING.
C
      IF ( ICOL . GT . NDFIF )                          GO TO 15
C
C*    IN THIS CASE, THE TOP NDFIF ENTRIES OF KOEE CONSTITUTE A COLUMN
C*    OF KOE2, AND THE NEXT NDFIF ENTRIES CONSTITUTE A COLUMN OF KE12.
C
C*    CONSIDER ALL OF THE DEGREES OF FREEDOM OF ONE COLUMN OF KBNIF.
C
      DO 10 IRW = 1 , NDFIF
C
      KBNIF(IRW,ICOL) = KBNIF(IRW,ICOL)-KOEE(IRW)-KOEE(NDFIF+IRW) * EN
C
   10 CONTINUE
C
      GO TO 30
C
C*    IN THIS CASE, THE TOP NDFIF ENTRIES OF KOEE CONSTITUE THE A
C*    COLUMN OF KOE21, AND THE NEXT NDFIF ENTRIES CONSTITUE A COLUMN
C*    OF KOE1.
C
   15 ICL = ICOL - NDFIF
C
C*    CONSIDER ALL OF THE DEGREES OF FREEDOM OF THE ICL-TH COLUMN OF
C*    KBNIF.
C
      DO 20 IRW = 1 , NDFIF
C
      KBNIF(IRW,ICL) = KBNIF(IRW,ICL) - KOEE(NDFIF+IRW) - KOEE(IRW)*ENCJ
C
   20 CONTINUE
C
   30 CONTINUE
C
      RETURN
      END
C
C
C
```

VVVVVVV 1111111111111111111111111111111111111111111111111111111111111111111111111111
VVVVVVV 1111111111111111111111 Digital Equipment Corporation - VAX/VMS Ver
VVVVVVV 1111111111111111111111111111111111111111111111111111111111111111111111111111