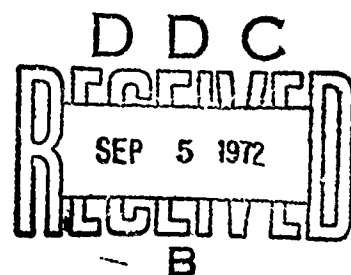


AD 747523

NAVAL POSTGRADUATE SCHOOL

Monterey, California



THESIS

FINITE ELEMENT SOLUTION FOR
AXISYMMETRIC TRANSIENT THERMAL STRESSES

by

Manouchehr Bakhshandehpour

Thesis Advisor:

R. E. Newton

Reproduced by
NATIONAL TECHNICAL
INFORMATION SERVICE

U.S. Department of Commerce
Springfield MA 01101

June 1972

Approved for public release; distribution unlimited.

174

UNCLASSIFIED

Security Classification

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY (Corporate author)		2a. REPORT SECURITY CLASSIFICATION	
Naval Postgraduate School Monterey, California 93940		Unclassified	
		2b. GROUP	
3. REPORT TITLE			
FINITE ELEMENT SOLUTION FOR AXISYMMETRIC TRANSIENT THERMAL STRESSES			
4. DESCRIPTIVE NOTES (Type of report and, inclusive dates)			
Mechanical Engineer's Thesis; June 1972			
5. AUTHOR(S) (First name, middle initial, last name)			
Manouchehr Bakhshandehpour; Lieutenant, Imperial Iranian Navy			
6. REPORT DATE		7a. TOTAL NO. OF PAGES	7b. NO. OF REF.
June 1972		174	9
8a. CONTRACT OR GRANT NO.		9a. ORIGINATOR'S REPORT NUMBER(S)	
b. PROJECT NO.			
c.		9b. OTHER REPORT NO(S) (Any other numbers that may be assigned this report)	
d.			
10. DISTRIBUTION STATEMENT			
Approved for public release; distribution unlimited.			
11. SUPPLEMENTARY NOTES		12. SPONSORING MILITARY ACTIVITY	
		Naval Postgraduate School Monterey, California 93940	

13. ABSTRACT

A finite element formulation for solving axisymmetric transient heat conduction and thermal stress problems is developed in this thesis. The governing equations of uncoupled, linear, isotropic thermoelasticity are discretized using quadratic isoparametric elements. A FORTRAN IV program, using double precision arithmetic, is presented. Compact storage techniques for banded symmetric matrices are used.

Comparisons between exact and computer solutions demonstrate close agreement for a number of test problems. Detailed instructions for using the program are included.

UNCLASSIFIED

Security Classification

14 KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
AXISYMMETRIC TRANSIENT HEAT CONDUCTION						
FINITE ELEMENT HEAT CONDUCTION						
FINITE ELEMENT TRANSIENT THERMAL STRESSES						
AXISYMMETRIC TRANSIENT THERMAL STRESSES						

UNCLASSIFIED

Security Classification

1-11400

Finite Element Solution for
Axisymmetric Transient Thermal Stresses

by

Manouchehr Bakhshandehpour
Lieutenant, Imperial Iranian Navy
B.S., Italian Naval Academy, 1960

Submitted in partial fulfillment of the
requirements for the degrees of

MECHANICAL ENGINEER
and
MASTER OF SCIENCE IN MECHANICAL ENGINEERING

from the
NAVAL POSTGRADUATE SCHOOL
June 1972

Author

M. Bakhshandehpour

Approved by:

R. E. Newton

Thesis Advisor

Robert A. Munn

Chairman, Department of Mechanical Engineering

Milton H. Claman

Academic Dean

ABSTRACT

A finite element formulation for solving axisymmetric transient heat conduction and thermal stress problems is developed in this thesis. The governing equations of uncoupled, linear, isotropic thermoelasticity are discretized using quadratic isoparametric elements. A FORTRAN IV program, using double precision arithmetic, is presented. Compact storage techniques for banded symmetric matrices are used.

Comparisons between exact and computer solutions demonstrate close agreement for a number of test problems. Detailed instructions for using the program are included.

TABLE OF CONTENTS

I.	INTRODUCTION -----	12
II.	FINITE ELEMENT FORMULATION OF HEAT CONDUCTION IN AXISYMMETRIC BODIES -----	14
	A. METHOD OF FORMULATION -----	14
	B. THERMAL BOUNDARY CONDITIONS -----	16
	C. EXACT TIME SOLUTION WITH SPATIAL DISCRETIZATION -----	18
	D. TIME INTEGRATION -----	20
	E. ESTIMATION OF EXTREME EIGENVALUES -----	26
III.	ONE-DIMENSIONAL HEAT CONDUCTION -----	29
IV.	STRESS PROBLEM -----	32
	A. STIFFNESS MATRIX -----	33
	B. THERMAL LOAD VECTOR -----	34
	C. PRESSURE LOAD VECTOR -----	36
	D. CENTRIFUGAL LOAD VECTOR -----	37
	E. STRUCTURAL BOUNDARY CONDITIONS -----	38
	F. SYSTEM EQUATION SOLVER -----	39
	G. PRINCIPLE OF SUPERPOSITION -----	39
	H. STRESS EVALUATION -----	39
V.	ONE-DIMENSIONAL TRANSIENT STRESSES -----	41
VI.	TEST PROBLEMS -----	42
VII.	CONCLUSIONS AND RECOMMENDATIONS -----	54
	APPENDIX A: APPLICABLE FORMULAS AND EQUATIONS -----	56
	APPENDIX B: LIST OF THE PROGRAM -----	59
	APPENDIX C: USER'S MANUAL -----	135

APPENDIX D: PROGRAMMING -----	160
LIST OF REFERENCES -----	171
INITIAL DISTRIBUTION LIST -----	172
DD FORM 1473 -----	173

LIST OF FIGURES

1. Slab with One Face Insulated and Another in Contact with Fluid -----	29
2. Quadrilateral Element Representation -----	33
3. Boundary Element Under Pressure -----	36
4. Two Radial Elements Representation in Thick Cylinder -----	43
5. Five Radial Elements Representation in Thick Cylinder -----	43
6. Stresses in Thick Cylinder Under Thermal Loading -----	44
7. Stresses in Thick Cylinder Under Uniform Internal Pressure -----	45
8. Stresses in Rotating Thick Cylinder -----	47
9. Hollow Semi-sphere 32 Elements Representation -----	48
10. Thermal Stresses in Hollow Sphere -----	49
11. Nozzle Geometry -----	50
12. Fluid Temperature-Time Histories -----	52
13. Element Representation in Nozzle -----	53
14. Longitudinal Cross Section -----	137
15. Subdivision into Elements -----	138
16. Element and Node Numbering -----	139
17. Entry Temperature-Time Variation -----	151
18. Deck Set-up for Using CHECK Program -----	155
19. Deck Set-up for Using AXITTS Program -----	157
20. Deck Set-up for Obtaining a Listing of the Program AXITTS -----	157
21. Deck Set-up for Obtaining a Punched Deck of the Program AXITTS -----	158

22.	Functional Flow Chart of CANDY -----	162
23.	Fluid Node Representation for Inside Flow -----	163
24.	Functional Flow Chart of STIFF -----	167
25.	Functional Flow Chart of FØRMF -----	168

LIST OF TABLES

I.	Attenuation Factor Comparison, Fourth Order Runge-Kutta Algorithm -----	23
II.	Attenuation Factor Comparison, Trapezoidal Integration -----	25
III.	Effect of Using Irons' Correction at Every 10 Steps of Integration -----	27
IV.	Comparison of One-dimensional Transient Temperatures -----	31
V.	Units for Input Data -----	136
VI.	Maximum Values for Program Parameters -----	170

LIST OF SYMBOLS

Note: A single underline is used to denote a column vector and a double underline denotes a rectangular matrix. The symbols used in computer program are described in the beginning of Appendix B.

a	Entrance fluid cross-sectional area
$\underline{\underline{A}}$	Linear combination of $\underline{\underline{C}}$ and $\underline{\underline{Y}}$ matrices
$\underline{b}$	A constant vector
$\underline{\underline{B}}$	Standard rectangular strain-displacement matrix
$\underline{\underline{C}}$	Thermal capacitance matrix
c	Specific heat
$\underline{\underline{D}}$	Standard elasticity matrix
E	Young's modulus of elasticity
e	Superscript designating element contribution
$\underline{F}$	Load vector
$\underline{F}_T^e$	Element thermal load vector
$\underline{F}_p^e$	Element pressure load vector
$\underline{F}_c^e$	Element centrifugal load vector
$\underline{\underline{G}}$	Linear combination of $\underline{\underline{C}}$ and $\underline{\underline{Y}}$ matrices
h	Surface heat transfer coefficient
$\underline{\underline{I}}$	Identity matrix
$\underline{\underline{J}}$	Jacobian coordinate transformation matrix
$\underline{\underline{K}}$	System stiffness matrix
$\underline{\underline{K}}^e$	Element stiffness matrix
k	Thermal conductivity
L	Thickness of slab

l	Arc length along the side of quadrilateral
N_i	Shape function
m	Total number of nodes
n	Outward normal or number of nodes per element
P	Pressure
R	Radial coordinate
S	Surface area
$\underline{T}$	Nodal temperature vector
T	Temperature or, when used as a superscript, transpose of a matrix
T_{avg}	Average temperature
T_f	Fluid temperature (used in one-dimensional example)
$\underline{U}$	A vector defined as $\langle 1 \ 1 \ 1 \ 0 \rangle^T$
u	Radial displacement
$\underline{v}$	Right-hand side vector in conduction equation
V	Volume
W	Work done by loads
$\underline{W}$	Modal matrix
w	Axial displacement
$\underline{w}$	Eigenvector
$\underline{Y}, \underline{Y}^+$	Thermal admittance matrix
y	Dependent variable
y_{ij}^*	Element ij of the matrix $\underline{Y}^*$
$[\]$	Matrix representation
$\langle \ \rangle$	Row vector
$\bar{\nabla}$	Gradient operator
α	Coefficient of thermal expansion
$\underline{\beta}$	Constant coefficient vector

$\underline{\delta}$	Nodal displacement vector
$\underline{\delta}^e$	Element displacement vector
$\underline{\epsilon}$	Strain vector
μ	Eigenvalue of one-dimensional transient temperature solution
$\underline{\epsilon}_0$	Thermal strain vector
$\underline{\epsilon}^e$	Element strain vector
η	Local element coordinate
ν	Poisson's ratio
λ	Eigenvalue
ρ	Material density
$\underline{\sigma}$	Stress vector
ξ	Local element coordinate
τ	Time
$\Delta\tau$	Step size of numerical time integration
τ_{RZ}	Shearing stress component
θ	Fluid temperature, or angle
$\underline{\Lambda}$	Spectral matrix
Ω	Speed of rotation

ACKNOWLEDGEMENTS

The author would like to express his gratitude to the Imperial Iranian Navy for having made it possible for him to undertake the course of graduate study leading to the present text.

To Dr. Robert E. Newton, Professor of Mechanical Engineering, the author wishes to express deep appreciation for his inspiring advice and guidance. Without his diligence and patience this study could not have been accomplished in its present form. The author is obliged to Dr. Gilles Cantin, Professor of Mechanical Engineering, for his generous advice during the course of study and computer programming. Thanks are also due to the personnel at the computer center of Naval Postgraduate School.

Finally the author should thank his wife for her patience and understanding throughout his work at this institution.

I. INTRODUCTION

Thermal stresses have become increasingly important in engineering practice during recent years. In power generation higher cycle temperatures and use of nuclear fission are largely responsible for this trend. This thesis describes a computer program for finding temperatures and stresses in bodies having axisymmetric geometry and loading. The governing equations are those of isotropic, uncoupled, quasi-static, linear thermoelasticity. They are discretized by using the finite element method. A FORTRAN IV computer program using double-precision arithmetic has been written to solve problems of the following kinds.

A. TEMPERATURE PROBLEMS

The transient temperature vector, evaluated at the nodal points, may be obtained for an axisymmetric body with the combination of insulated, convection, or constant temperature boundary conditions. For the convection thermal boundary condition, however, we may have fluid flowing with entry temperature prescribed as a linear function of time (RAMP). The program can handle up to 15 different ramps, each having a different flow velocity, and with discontinuities between successive ramps.

B. STRESS PROBLEMS

The program will generate load vectors for pressure loading, centrifugal loading, and axial force. Provision is

made for direct input of one additional load vector. Stresses may be found for any combination of these loadings.

C. THERMAL STRESS PROBLEMS

Thermal stresses may be found for as many as 20 different temperature vectors which may be output of the temperature problem or direct input. In short, in this part any combination of the temperature and stress problems may be used.

II. FINITE ELEMENT FORMULATION OF HEAT CONDUCTION IN AXISYMMETRIC BODIES

A. METHOD OF FORMULATION

For bodies of revolution under axisymmetric loading the mathematical problems presented are two-dimensional. The governing equation for non-steady heat conduction is

$$\bar{\nabla} \cdot k \bar{\nabla} T = \rho c \dot{T}, \quad (1)$$

where k is the thermal conductivity, ρ the density, c the specific heat, T the temperature and $\bar{\nabla}$ the gradient operator. The superior dot denotes a time derivative.

Applying Galerkin's principle [1] gives

$$\int_V N_i \bar{\nabla} \cdot k \bar{\nabla} T \, dV = \int_V \rho c N_i \dot{T} \, dV, \quad (2)$$

where the integral is over the volume V of the conducting body and N_i is a "shape function" used in representing the temperature distribution. If S is the surface of the body and n the outward normal to surface, then using Gauss' theorem we can write

$$\int_V \bar{\nabla} \cdot (N_i k \bar{\nabla} T) \, dV = \int_S N_i k \frac{\partial T}{\partial n} \, dS. \quad (3)$$

Since

$$\begin{aligned} \int_V \bar{\nabla} \cdot (N_i k \bar{\nabla} T) \, dV &= \int_V (\bar{\nabla} N_i) \cdot (k \bar{\nabla} T) \, dV \\ &+ \int_V N_i \bar{\nabla} \cdot (k \bar{\nabla} T) \, dV, \end{aligned} \quad (4)$$

we can combine Eqs. 2, 3 and 4 to get

$$\int_V \rho c N_i \dot{T} dV + \int_V (\bar{\nabla} N_i) \cdot (k \bar{\nabla} T) dV = \int_S N_i k \frac{\partial T}{\partial n} dS. \quad (5)$$

Each node of the solid region has a separate discretized linear equation calculated from Eq. 5 using the appropriate shape function N_i . Thus each of the volume integrals on the left hand side of Eq. 5 yields a square coefficient matrix in the assembled set of equations. Calculation of these matrices is a standard process. Details are given by Zienkiewicz [1].

The discretized set of equations takes the form

$$\underline{\underline{C}} \dot{\underline{T}} + \underline{\underline{Y}} \underline{T} = \underline{v}, \quad (6)^\vee$$

where there is a term by term correspondence with Eq. 5. The real symmetric matrices $\underline{\underline{C}}$ and $\underline{\underline{Y}}$ represent, respectively, the thermal capacitance and thermal admittance. The elements of the vector $\underline{T}$ are nodal temperatures. The vector $\underline{v}$, discussed in the following section, depends upon the thermal boundary conditions.

In the present development piecewise constant material properties have been assumed, i.e., k , ρ and c are constant within each element, but may vary from element to element. Also two-dimensional isoparametric elements are used. Applicable equations are summarized in Appendix A.

[✓]In this text a double underline denotes a rectangular matrix and single underline denotes a column vector.

B. THERMAL BOUNDARY CONDITIONS

Thermal boundary conditions affect only those scalar equations of Eq. 6 which correspond to boundary nodes. Accordingly, the vector $\underline{v}$ is sparse. Also, in a single problem it is common to have different thermal boundary conditions on individual portions of the boundary. In what follows the subvectors of $\underline{v}$ (distinguished by individual superscripts) which correspond to separate boundary conditions are treated individually.

1. Insulated

It is clear that for insulated boundary conditions the subvector $\underline{v}^{(1)}$ of the right hand side of Eq. 6 corresponding to this boundary condition is zero, since $\frac{\partial T}{\partial n} = 0$.

2. Convection

The heat transfer mechanism occurs in the interface of the solid and fluid. If the fluid temperature is θ and the heat transfer coefficient is h , then equating heat conducted away from the surface to the efflux from the solid [2] gives

$$-k \left(\frac{\partial T}{\partial n} \right)_S = h(T - \theta), \quad (7)$$

where the subscript S means that the derivative is evaluated at the surface.

For constant h :

$$\int_S N_i k \frac{\partial T}{\partial n} dS = h \int_S N_i (\theta - T) dS. \quad (8)$$

In what follows the fluid temperature θ is taken to be a specified function of position and time. For purposes of discretization, the fluid temperature is specified at a discrete number of fluid "nodes." If θ_j represents the fluid temperature at fluid node j , then the fluid temperature along the boundary may be represented by

$$\theta = \sum N_j \theta_j, \quad (9)$$

where the N_j are one-dimensional forms of the shape functions used for the solid. Substitution in Eq. 8 gives

$$\int_S N_i k \frac{\partial T}{\partial n} dS = \sum_j y_{ij}^* (\theta_j - T_j), \quad (10)$$

where the summation extends over the surface nodes and the coefficients y_{ij}^* are given by

$$y_{ij}^* = h \int_S N_i N_j dS. \quad (11)$$

Assembling the contributions from Eq. 10, the subvector $v^{(2)}$ for the convection boundary condition may be written

$$v^{(2)} = \underline{Y}^* \underline{\theta}. \quad (12)$$

The contributions $-\sum y_{ij}^* T_j$ from Eq. 10 are included by augmenting the matrix $\underline{Y}$ (see Eq. 13 below).

3. Constant Temperature

If θ represents the constant temperature desired at the wetted surface, then we can use the convection boundary condition and replace h by a big number (say 10^{20}). Since h is very large, then for thermal equilibrium the temperature

T at the surface will be forced to equal θ . So the subvector $\underline{v}^{(3)}$ for the portion corresponding to the constant temperature boundary condition can be obtained from Eq. 12.

Upon the application of these boundary conditions in a single problem, the right-hand side vector $\underline{v}$ will be combined from the corresponding subvectors and the finite element discretized equation becomes

$$\underline{C} \dot{\underline{T}} + \underline{Y}^+ \underline{T} = \underline{v}, \quad (13)$$

where $\underline{Y}^+ = \underline{Y} + \underline{Y}^*$.

C. EXACT TIME SOLUTION WITH SPATIAL DISCRETIZATION

We consider only the solution of Eq. 13 for $\underline{v} = \text{constant}$ with $\underline{T} = \underline{a}$ at time $\tau = 0$. Let $\underline{T}_s$ be a particular solution (steady state) with $\dot{\underline{T}}_s = 0$ so that

$$\underline{T}_s = (\underline{Y}^+)^{-1} \underline{v}. \quad (14)$$

For the homogeneous equation

$$\underline{C} \dot{\underline{T}} + \underline{Y}^+ \underline{T} = 0 \quad (15)$$

the assumption $\underline{T} = \underline{w} \exp(-\lambda\tau)$, where $\underline{w}$ is a vector and λ is a scalar, yields the form

$$\underline{Y}^+ \underline{w} = \lambda \underline{C} \underline{w}. \quad (16)$$

It is apparent that Eq. 16 defines an eigenvalue problem. Let $\underline{\Lambda}$ be the spectral matrix and $\underline{W}$ the modal matrix with normalization according to

$$\underline{W}^T \underline{C} \underline{W} = \underline{I}, \quad (17)$$

where $\underline{I}$ is the identity matrix of the same order as $\underline{C}$. Now let

$$\underline{T} = \underline{W} \exp(-\underline{\Lambda}\tau) \underline{b} \quad (18)$$

where $\underline{b}$ is a constant vector. Substituting this in Eq. 15 gives

$$\underline{Y}^+ \underline{W} \exp(-\underline{\Lambda}\tau) \underline{b} = \underline{C} \underline{W} \underline{\Lambda} \exp(-\underline{\Lambda}\tau) \underline{b}. \quad (19)$$

Now Eq. 19 is satisfied for all $\underline{b}$ if

$$\underline{W}^T \underline{Y}^+ \underline{W} = \underline{\Lambda}, \quad (19')$$

and this is guaranteed to be satisfied since $\underline{\Lambda}$ and $\underline{W}$ are spectral and modal matrices for the eigenvalue problem of Eq. 16 with $\underline{W}$ normalized according to Eq. 17.

Returning to the original problem (Eq. 13), the complete solution may be written as

$$\underline{T} = \underline{W} (\underline{\beta} + \exp(-\underline{\Lambda}\tau) \underline{b}) \quad (20)$$

where

$$\underline{T}_s = \underline{W} \underline{\beta}, \quad (21)$$

and $\underline{\beta}$ is a constant vector. Now $\underline{\beta}$ may be found (using Eq. 14) to be

$$\underline{\beta} = \underline{\Lambda}^{-1} \underline{W}^T \underline{v}. \quad (22)$$

Substituting this result into Eq. 20 and using the initial condition gives the result

$$\underline{b} = \underline{W}^{-1} \underline{a} - \underline{\Lambda}^{-1} \underline{W}^T \underline{v}. \quad (23)$$

The general solution of Eq. 13 may thus be written as

$$\underline{T} = \underline{W}(\underline{I} - \exp(-\underline{\Lambda}\tau)) \underline{\Lambda}^{-1} \underline{W}^T \underline{y} + \underline{W} \exp(-\underline{\Lambda}\tau) \underline{W}^{-1} \underline{a}. \quad (24)$$

For purposes of the present program non-zero components of vector $\underline{y}$ are to be specified as piecewise linear functions of time. During each segment of time history of $\underline{y}$ an analytical solution of Eq. 13 is possible in the form of a particular solution plus a complementary solution such as Eq. 20. At each node the corresponding time variation of temperature will consist of a linear part contributed by the particular solution and a sum of n terms representing the complementary part. Each of these n terms decays exponentially with a separate time constant. In principle it is a straightforward process to find each particular solution and accompanying complementary solution.

Contemplated problems may typically have from 10 to 40 segments required to represent the piecewise linear variation of $\underline{y}$. The number of body nodes n will be of the order of 100 or more. In view of the number of particular solutions required, each accompanied by an individual complementary solution of the form given by Eq. 20, it was concluded that a numerical solution of Eq. 13 would be considerably more economical than an analytic one such as that given by Eq. 24.

D. TIME INTEGRATION

In this section the relative merits of the Runge-Kutta and trapezoidal methods of time integration are discussed. Since either of these methods will give an exact result if the solution is a linear function of time, investigation

is confined to performance on a single scalar equation

$$\dot{y} + \lambda y = 0 \quad (25)$$

whose solution $y = y_0 \exp(-\lambda\tau)$ is of the same form as the components of the complementary solution (Eq. 20).

1. Runge-Kutta Method

A method introduced by Runge and subsequently elaborated by Heun and Kutta [3] is widely used for the numerical solution of first order ordinary differential equations.

This algorithm prescribes a sequence of calculations for determining the ordinate y_{i+1} at time $\tau_{i+1} = \tau_i + \Delta\tau$ in terms of y_i and values of $\dot{y}$ at intermediate and end points of the interval $\Delta\tau$. The fourth-order form, which requires four evaluations of $\dot{y}$, gives for Eq. 25 the result

$$\frac{y_{i+1}}{y_i} = 1 - \lambda\Delta\tau + \frac{(\lambda\Delta\tau)^2}{2!} - \frac{(\lambda\Delta\tau)^3}{3!} + \frac{(\lambda\Delta\tau)^4}{4!} . \quad (26)$$

The right-hand side of Eq. 26 represents the first five terms of the Taylor expansion of the exact solution

($y_{i+1}/y_i = \exp(-\lambda\Delta\tau)$) so we may conclude that the relative error in each time step is less than modulus of the next term: $(\lambda\Delta\tau)^5/5!$.

In addition to providing the apparent prospect for high precision indicated by this error bound, the Runge-Kutta method also permits changes of time increment during the integration process without requiring additional computationally expensive matrix decompositions. The attractiveness of these two features dictated a thorough exploration of the potential of this method for the present application. The

disqualifying defect which emerged after studying a number of examples is readily appreciated from examination of Table I. For values of $\lambda\Delta\tau$ less than 0.5 it is apparent that Runge-Kutta scheme affords acceptable engineering accuracy. However, when the method is applied to solution of Eq. 13 we must deal with a number of λ 's equal to the number of nodes (see Eq. 20). This number may be greater than 100 and the ratio of the largest λ to the smallest may easily exceed 1000. Although the solution is dominated by the contributions of the eigenvectors corresponding to the smaller λ s, it is clear that the solution will be unstable if the largest $\lambda\Delta\tau$ exceeds about 2.7. Because an unacceptably small $\Delta\tau$ is required in typical problems, the Runge-Kutta method was rejected.

2. Trapezoidal Method

The trapezoidal method estimates y_{i+1} from the formula

$$y_{i+1} = y_i + \frac{\Delta\tau}{2} (\dot{y}_i + \dot{y}_{i+1}). \quad (27)$$

Substituting for $\dot{y}_i$ and $\dot{y}_{i+1}$ from Eq. 25 and rearranging gives

$$\frac{y_{i+1}}{y_i} = \frac{2 - \lambda\Delta\tau}{2 + \lambda\Delta\tau}. \quad (28)$$

Series expansion of the right-hand side provides an error bound (per step):

$$(\lambda\Delta\tau)^3/12.$$

From the point of view of the size of $\lambda\Delta\tau$ this method has no stability limit, but has slow attenuation with alteration

TABLE I
ATTENUATION FACTOR COMPARISON
Fourth Order Runge-Kutta Algorithm

$\lambda\Delta\tau$	y_{i+1}/y_i (Runge-Kutta)	$\exp(-\lambda\Delta\tau)$ (Exact)
.0001	.9999	.9999
.001	.9990	.9990
.01	.9901	.9901
.1	.9048	.9048
.2	.8187	.8187
.5	.6068	.6065
1.0	.3750	.3679
2.0	.3333	.1353
2.5	.6484	.0821
3.0	1.3750	.0498
4.0	5.0000	.0183
8.0	110.3333	.0003
10.0	291.0000	.0000
20.0	5514.3333	.0000
50.0	240784.3333	.0000
100.0	4004901.0000	.0000

in sign for large $\lambda\Delta\tau$. Table II shows this behavior.

Since a wide usable range of $\lambda\Delta\tau$ is essential and the stability of trapezoidal integration is guaranteed, this method is chosen for the present program.

Applying the trapezoidal algorithm to Eq. 13 yields

$$\underline{\underline{A}} \underline{\underline{T}}^{i+1} = \underline{\underline{G}} \underline{\underline{T}}^i + \frac{\Delta\tau}{2} (\underline{\underline{v}}^{i+1} + \underline{\underline{v}}^i), \quad (29)$$

where

$$\underline{\underline{A}} = \underline{\underline{C}} + \frac{\Delta\tau}{2} \underline{\underline{Y}}^+,$$

$$\underline{\underline{G}} = \underline{\underline{C}} - \frac{\Delta\tau}{2} \underline{\underline{Y}}^+.$$

and the superscripts denote evaluation at discrete time intervals $\Delta\tau$. If m is the order of the capacitance and admittance matrices, $\underline{\underline{C}}$ and $\underline{\underline{Y}}$, then once a certain step size $\Delta\tau$ is chosen, it requires $m^3/3$ operations to perform the needed triangular decomposition of $\underline{\underline{A}}$. Thus, for large m , a change of step size $\Delta\tau$ becomes costly from the point of view of computer time. Accordingly, in the present program only one time step size is used throughout each problem.

Also, for assuring sufficiently rapid attenuation of the components corresponding to the large $\lambda\Delta\tau$, the following correction is utilized.

3. Irons' Correction

Irons proposed a scheme [4] for augmenting the attenuation of the contributions of those eigenvectors for which $\lambda\Delta\tau$ is large. Define

TABLE II
ATTENUATION FACTOR COMPARISON
Trapezoidal Integration

$\lambda\Delta\tau$	y_{i+1}/y_i (Trapezoidal)	$e^{-\lambda\Delta\tau}$ (Exact)
.0001	.9999	.9999
.001	.9990	.9990
.01	.9901	.9901
.1	.9048	.9048
.2	.8182	.8187
.3	.7391	.7408
.5	.6000	.6065
1.0	.3333	.3679
2.0	.0000	.1353
2.5	-.1111	.0821
3.0	-.2000	.0498
4.0	-.3333	.0183
8.0	-.6000	.0003
10.0	-.6667	.0000
20.0	-.8182	.0000
50.0	-.9231	.0000
100.0	-.9608	.0000

$$y_i^* = .25 y_{i-1} + .5 y_i + .25 y_{i+1}, \quad (30)$$

where y_i and y_{i+1} are obtained from y_{i-1} by trapezoidal integration.

In the program presented in Appendix B, Eq. 30 is used after every 10 steps of time integration. Table III shows the resulting modifications.

E. ESTIMATION OF EXTREME EIGENVALUES

Analytic results for one-dimensional heat conduction give, for an eigenvalue,

$$\lambda = \frac{\pi^2 k}{4\rho c} \frac{1}{d^2}, \quad (31)$$

where d is the distance between points of extreme temperature and zero temperature.

If we use this to estimate the smallest λ in cylindrical coordinates, two modifications are recommended.

1. Assume that the point of zero temperature is in the fluid at a distance from the wall equal to k/h , where h is the surface heat transfer coefficient.
2. If there are two approximately orthogonal paths for heat flow from the (single) maximum temperature point, then replace $1/d^2$ in the above formula by

$$\frac{1}{d^2} = \frac{1}{d_{\min}^2} + \frac{1}{d_{\max}^2}. \quad (32)$$

For estimating the largest λ , the surface heat transfer coefficient has no significant effect. We may continue to

TABLE III
EFFECTS OF USING IRONS' CORRECTION
AFTER 10 STEPS OF INTEGRATION

$\lambda\Delta\tau$	$\exp(-10\lambda\Delta\tau)$ Exact	y_{10}/y_0 Trapezoidal	y_{10}^*/y_0 Corrected
0.00010	0.99900	0.99900	0.99900
0.00020	0.99800	0.99800	0.99800
0.00100	0.99005	0.99005	0.99005
0.01000	0.90484	0.90484	0.90486
0.10000	0.36788	0.36757	0.36849
0.20000	0.13534	0.13443	0.13579
0.30000	0.04979	0.04866	0.04978
0.50000	0.00674	0.00605	0.00645
1.00000	0.00005	0.00002	0.00002
2.00000	0.00000	0.00000	0.00000
4.00000	0.00000	0.00002	-0.00001
8.00000	0.00000	0.00605	-0.00040
10.00000	0.00000	0.01734	-0.00072
20.00000	0.00000	0.13443	-0.00136
50.00000	0.00000	0.44914	-0.00072
100.00000	0.00000	0.67028	-0.00027

use the same formula for $\frac{1}{d^2}$, but now consider only the smallest element and take

$$d_{\min} = \frac{\text{length of smallest side}}{4},$$
$$d_{\max} = \frac{\text{length of largest side}}{4}.$$
(32')

A comparison of estimates based on Eq. 31, 32, 32' with the exact solution of the eigenvalue problem has been carried out for several examples. Based on these comparisons, it is believed that these estimates are sufficiently accurate for choosing a time step and estimating the time of occurrence of the extreme stresses.

III. ONE-DIMENSIONAL HEAT CONDUCTION

For comparison of numerical (time) integration methods, studies of one-dimensional heat conduction were made. In this section numerical results for trapezoidal time integration using Irons' correction are compared with the exact transient temperature solution.

Consider a flat slab of thickness L with conductivity k , density ρ , specific heat c , zero initial temperature, one face insulated and the other in contact with fluid at temperature T_f (Fig. 1). The surface heat transfer is h . The exact transient heat conduction solution is available [5] as

$$T = T_f \left\{ 1 - 2 \sum_{i=1}^{\infty} \frac{\sin \mu_i L \cos \mu_i x}{\mu_i L + \sin \mu_i L \cos \mu_i L} \cdot e^{-\mu_i^2 \frac{k}{\rho c} \tau} \right\} \quad (33)$$

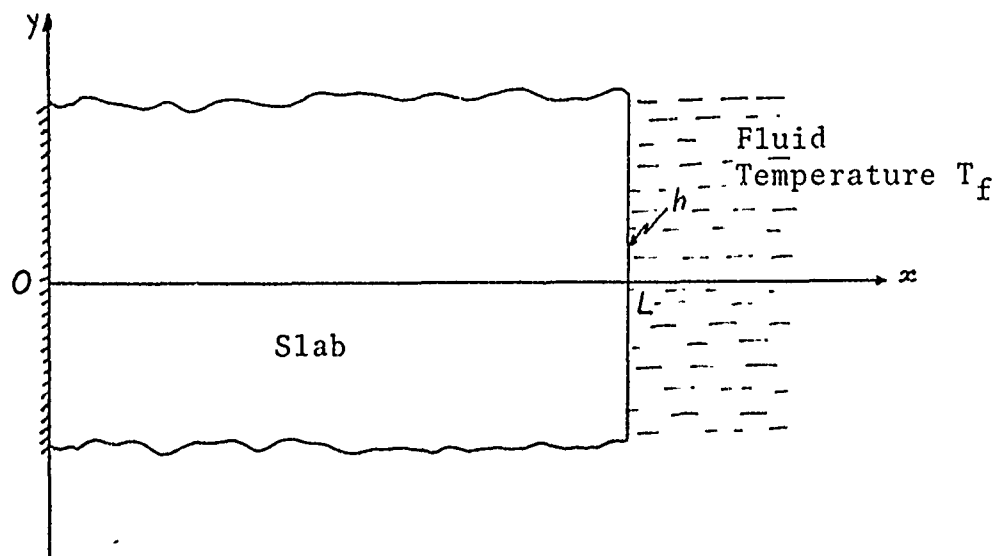


Figure 1. Slab with One Face Insulated and Another in Contact with Fluid.

where $k/\rho c$ is the diffusivity and the μ_i are the solutions of the transcendental equation

$$\tan \mu L = \frac{hL}{k} \frac{1}{\mu L} . \quad (34)$$

For the finite element comparisons we subdivide the distance L into m one-dimensional 3-noded elements and use the corresponding isoparametric shape functions. (The working equations, shape functions and the element capacitance and admittance matrices are given in Appendix A.)

In the course of this investigation separate computer programs were written to evaluate nodal transient temperatures using the following methods:

- (a) exact transient temperature solution, Eq. 33;
- (b) exact time solution with spatial discretization (section II.C, Eq. 24);
- (c) Runge-Kutta time integration (section II.D, part a);
- (d) trapezoidal time integration (section II.D, part b);
- (e) trapezoidal time integration with Irons' correction (section II.D, part c).

For an initial step change of fluid temperature from zero to 1, transient temperatures have been found. For these comparisons the parameters (in consistent units) were taken to be:

$$L = 8, \rho = 25, k = 8, c = 5 \text{ and } h = 5$$

The distance L was subdivided into $m = 3$ elements. For the present purpose, comparison is confined to the exact solution (item (a)), and the finally adopted system (item (e)).

In Table IV, temperatures at the two faces ($x = 0$, $x = 8$) are compared for various times. The trapezoidal integration has been performed using the constant time increment unity and the Irons' correction is applied after every 10 increments.

It is believed that Table IV demonstrates that the numerical integration method gives adequate accuracy for engineering applications.

TABLE IV
COMPARISON OF ONE-DIMENSIONAL TRANSIENT TEMPERATURES

Method	x	Time=0	Time=10	Time=20	Time=30	Time=50	Time=70
Exact	0.	.00000	.00000	.00000	.00002	.00095	.00560
Trapezoidal with Irons' ✓	0.	.00000	.00028	.00007	.00066	.00207	.00619
Exact	8.	.00000	.38431	.47684	.53284	.60264	.64685
Trapezoidal with Irons' ✓	8.	.00000	.38720	.47716	.53262	.60256	.64686

✓ Irons' correction is used after every 10 steps of trapezoidal integration.

IV. STRESS PROBLEM

For bodies of revolution deformed symmetrically under axisymmetric loading, the stress distribution is two-dimensional. Since deformation is symmetric about the axis of revolution, cylindrical coordinates (R, Z, θ) are used. It follows that the stress components are independent of the angle θ and all derivatives with respect to θ are zero. Also the components of shearing stress $\tau_{R\theta}$ and $\tau_{Z\theta}$ vanish on account of the symmetry. But since any radial displacement induces a strain ϵ_θ in the circumferential direction, this non-zero component of strain and the three in-plane components $(\epsilon_Z, \epsilon_R, \gamma_{RZ})$, complete the state of strain at a point in any axisymmetric situation. Hence the state of stress for an axisymmetric body under axisymmetric loading is given by

$$\underline{\sigma} = \langle \sigma_Z \ \sigma_R \ \sigma_\theta \ \tau_{RZ} \rangle^T . \quad (35)$$

In this chapter the stiffness matrix of an axisymmetric body and the thermal, pressure, and centrifugal load vectors are formulated and, finally, evaluation of stresses at a point is discussed. The treatment closely follows that of Zienkiewicz [1] and this reference should be consulted for further details.

A. STIFFNESS MATRIX

The elements used are bodies of revolution (about the Z axis). For analysis it is sufficient to describe the cross-section in the R,Z plane. In Fig. 2 such an element and the local ξ, η coordinates are shown.

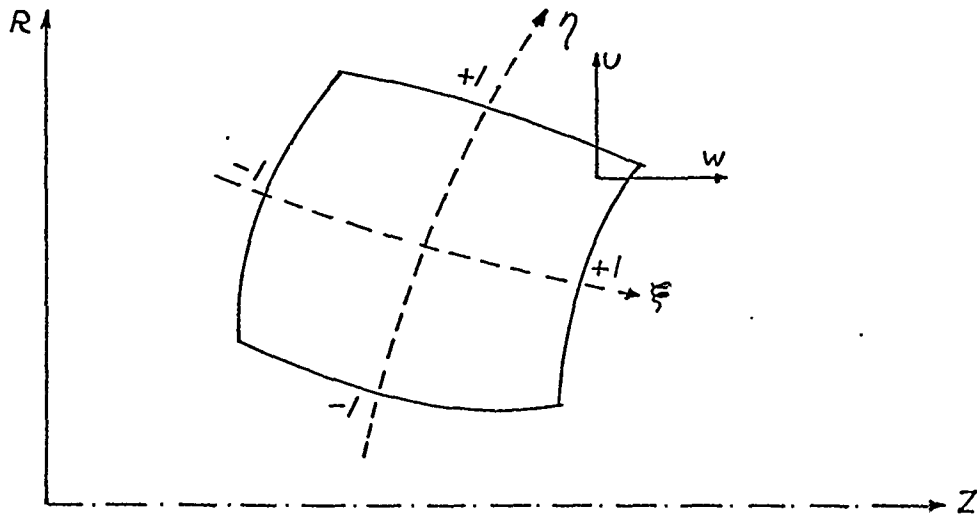


Figure 2. Quadrilateral Element Representation.

If u and w are the displacement components at a point in the directions of R and Z respectively, then these displacement components may be defined in terms of the nodal displacements by the appropriate isoparametric shape functions as

$$u = \sum_{i=1}^8 N_i u_i, \quad w = \sum_{i=1}^8 N_i w_i, \quad (36)$$

where N_i , a function of ξ and η , is the shape function for element node i and u_i, w_i , are the nodal displacement components. The strain-displacement relations [6] can now be used to obtain the components of the strain vector. Thus

$$\underline{\underline{\epsilon}} = \underline{\underline{B}} \underline{\underline{\delta}} , \quad (37)$$

where $\underline{\underline{B}}$ is the standard rectangular strain-displacement matrix of any finite element formulation, a function of the local coordinates ξ and η , and $\underline{\underline{\delta}}$ is the vector of nodal displacement. (See Appendix A, part 3, where the applicable formulas and useful equations are summarized).

If the elasticity matrix for an isotropic material is $\underline{\underline{D}}$, then the stress vector $\underline{\underline{\sigma}}$ at a point is given by

$$\underline{\underline{\sigma}} = \underline{\underline{D}} \underline{\underline{\epsilon}} . \quad (38)$$

Now, by evaluation of the total strain energy in the element, the element stiffness matrix can readily be obtained as

$$\underline{\underline{K}}^e = \int \underline{\underline{B}}^T \underline{\underline{D}} \underline{\underline{B}} dV , \quad (39)$$

where the integration extends over the volume of the element.

In the present program the upper triangle of each element stiffness matrix is evaluated by numerical integration using four Gauss points within the range of ξ and of η [1]. The element contribution is immediately placed in the system stiffness matrix, which is stored in banded form.

B. THERMAL LOAD VECTOR

If we denote T as the difference between local temperature and reference temperature, then the thermal strain $\underline{\underline{\epsilon}}_0$ is given as

$$\underline{\underline{\epsilon}}_0 = \underline{\underline{U}} \alpha T, \quad (40)$$

where

$$\underline{U} = \langle 1 \ 1 \ 1 \ 0 \rangle^T$$

and α is the coefficient of thermal expansion.

The thermal load vector $\underline{F}_T^e$ is given by Zienkiewicz [1] as

$$\underline{F}_T^e = \int \underline{B}^T \underline{D} \underline{\epsilon}_0^e dV . \quad (41)$$

From the point of view of the numerical evaluation it is interesting to note, however, that the product $\underline{D} \underline{\epsilon}_0^e$ in Eq. 41 will reduce to

$$\underline{D} \underline{\epsilon}_0^e = \frac{E \alpha T}{1-2\nu} \underline{U} , \quad (42)$$

where E is the modulus of elasticity and ν is Poisson's ratio. Thus

$$\underline{F}_T^e = \frac{E \alpha}{1-2\nu} \int T \underline{B}^T \underline{U} dV , \quad (43)$$

where $T = \sum_{i=1}^n N_i T_i$ and n is the number of nodes in each element.

In the attached computer program in Appendix B the advantage of the simplicity of Eq. 42 has been utilized. Also, since $\underline{B}$ has some zero components, in the process of multiplication of $\underline{B}^T \underline{U}$, simply the addition of the appropriate non-zero components of each column of the $\underline{B}^T$ has been performed. Finally, Eq. 43 has been integrated with four Gauss points.

C. PRESSURE LOAD VECTOR

Consider a quadrilateral element as in Fig. 3 on the boundary of the axisymmetric cross-section where constant pressure P is applied. The infinitesimal force dF due to the normal pressure acting on the inner infinitesimal circumferential surface dS is

$$dF = PdS = 2 \pi RP \, d\ell , \quad (44)$$

where $d\ell$ is the infinitesimal length along the side of quadrilateral.

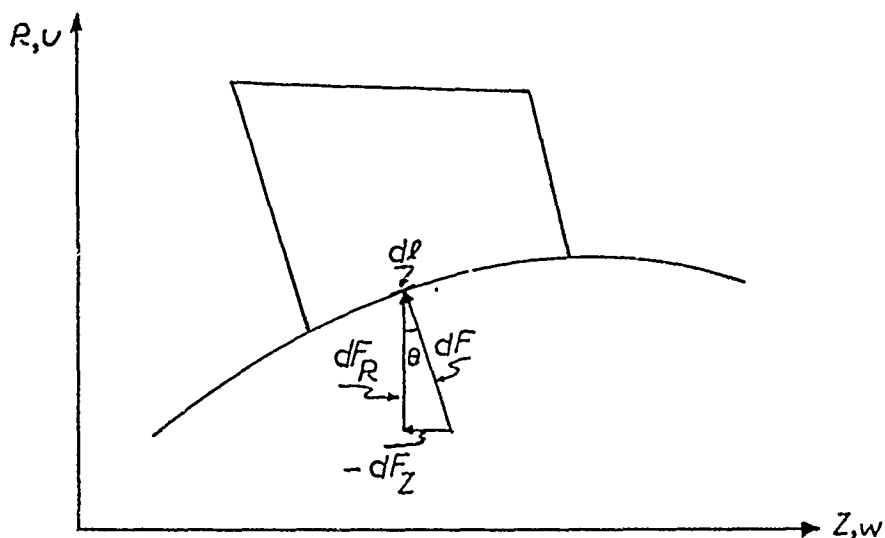


Figure 3. Boundary Element Under Pressure.

Let F_R and F_Z be the components of the pressure force in the R and Z directions respectively, then

$$dF_R = dF \cos \theta, \quad dF_Z = -dF \sin \theta , \quad (45)$$

where $\sin \theta = \frac{dR}{d\ell}$ and $\cos \theta = \frac{dZ}{d\ell}$. Therefore

$$dF_R = (2 \pi RP) dZ , \quad dF_Z = -(2 \pi RP) dR. \quad (46)$$

Since the total work W done by the normal force F is equal to the sum of the work done in R and Z directions,

$$\begin{aligned} W &= \int u \, dF_R + \int w \, dF_Z \\ &= 2\pi P \left(\int R \, u \, dZ - \int R \, w \, dR \right). \end{aligned} \quad (47)$$

Now, by using the appropriate shape functions, each component of Eq. 47 may be defined in terms of the nodal values. Since

$$W = (\underline{\delta}^e)^T \underline{F}_p^e, \quad (48)$$

where $\underline{F}_p^e$ is the element pressure load vector vector contributed by node j is explicitly given as

$$\underline{F}_{jp}^e = \begin{bmatrix} F_R \\ F_Z \end{bmatrix}_{jp}^e = 2\pi P \int_{-1}^1 (\Sigma N_i R_i) \begin{bmatrix} \Sigma \frac{\partial N_i}{\partial \xi} Z_i \\ -\Sigma \frac{\partial N_i}{\partial \xi} R_i \end{bmatrix} N_j \, d\xi, \quad (49)$$

where N_j in Eq. 49 is the appropriate shape function for node j .

D. CENTRIFUGAL LOAD VECTOR

Refer again to Fig. 2 and assume that the body is rotating about the Z axis with angular speed Ω . Then the centrifugal force per unit volume at a point distant R from the Z axis will be $\rho\Omega^2 R$, where ρ is the density of the material. The work done in this case is

$$W = \int_V \rho\Omega^2 R \, u \, dV. \quad (50)$$

For constant $\rho\Omega^2$ the corresponding element centrifugal load vector is readily obtained by evaluation of the components of the integral in Eq. 50 in terms of the nodal variables, i.e.,

$$F_{j_c}^e = 2\pi\rho\Omega^2 \int_{-1}^1 \int_{-1}^1 (\sum N_i R_i)^2 \det \underline{J} N_j d\xi d\eta, \quad (51)$$

where $\det \underline{J}$ is the determinant of the Jacobian coordinate transformation matrix (see Appendix A).

E. STRUCTURAL BOUNDARY CONDITIONS

The structural boundary conditions implemented in the program are:

- (a) one or more nodes prevented from moving axially;
- (b) one or more nodes prevented from moving radially;
- (c) the right-hand end cross-sectional plane remains plane and the transmitted axial force is specified. Hereinafter this will be referred to as the plane-end boundary condition.

The computer program presented in Appendix B has the capability of handling any combination of the above mentioned structural boundary conditions. For boundary conditions of the types (a) and (b), simply multiplying the corresponding diagonal component of the stiffness matrix by 10^{20} gives zero displacement [8] (for practical purposes). For the boundary condition of type (c) both ends are initially fixed axially for all solutions. An additional solution is obtained for unit axial displacement of one end. The axial force is evaluated for each solution. Superposition is performed by adding the displacement vectors for the given

loadings (thermal plus mechanical), plus an appropriate fraction of the vector found for unit axial displacement. This fraction is chosen so that the resultant axial load has the specified value.

F. SYSTEM EQUATION SOLVER

Once the desired structural boundary conditions are applied, then the problem is to find the nodal displacement vector $\underline{\delta}$, corresponding to a given number of load vectors. We have

$$\underline{K} \underline{\delta} = \underline{F} , \quad (52)$$

where $\underline{K}$ is the system stiffness matrix in banded form and $\underline{F}$ is a load vector. In the present computer program a single Cholesky decomposition is performed on $\underline{K}$. Then, by a process of forward and back substitution, each load vector is replaced by the corresponding displacement vector.

G. PRINCIPLE OF SUPERPOSITION

Upon the evaluation of the displacement vectors due to the various types of loading, the principle of superposition can be applied on the displacement vectors. On each thermal displacement vector the displacement due to any other type of loading is superimposed and, as the result, the number of displacement vectors is reduced to the number of thermal load vectors.

H. STRESS EVALUATION

From the system displacement vector $\underline{\delta}$, the displacement vector of each element $\underline{\delta}^e$ may be obtained easily. Then the

corresponding element strain vector $\underline{\epsilon}^e$ at any point can be found from

$$\underline{\epsilon}^e = \underline{B} \underline{\delta}^e . \quad (53)$$

Finally, the corresponding element stress vector $\underline{\sigma}^e$ is obtained by

$$\underline{\sigma}^e = \underline{D} (\underline{\epsilon}^e - \underline{\epsilon}_0^e) . \quad (54)$$

Since normally the stresses on the inner and outer surfaces of axisymmetric bodies are desired, in the computer program presented in the Appendix B provision has been made to calculate the stresses at the two Gauss points corresponding to $\xi = \pm \frac{1}{\sqrt{3}}$ on the inner and outer boundaries of each element (where $\eta = \pm 1$).

Upon the evaluation of the stresses at each point the mean stress and the octahedral shearing stress [7] are calculated. The program gives as output the extreme values of these stresses, the R and Z coordinates of the corresponding points, and the times of occurrence.

V. ONE-DIMENSIONAL TRANSIENT STRESSES

In this section a one-dimensional comparison of stresses is made between exact and finite element results. The transient temperature problem is the one previously described in Section III.

If the slab edges are free to translate in the plane of the slab, but are prevented from rotating, the exact solution for thermal stress [9] is

$$\sigma_y = \sigma_z = \frac{E\alpha}{1-\nu} (T_{\text{avg}} - T) , \quad (55)$$

where T is the local temperature (Eq. 33), and

T_{avg} is the average temperature in the slab

If we choose $E = 2$, $\alpha = .50$, and $\nu = 0$ (all in consistent units), then the maximum stress obtained from the exact solution is

$$\sigma_{\text{max}} = -.477786$$

and it occurs at $x = 8$, $\tau = 73$.

Using the finite element technique with trapezoidal time integration and Irons' correction every 10 steps, the maximum stress is found to be

$$\sigma_{\text{max}} = -.477778$$

and it also occurs at $x = 8$, $\tau = 73$, as before.

It is observed that the method chosen gives excellent results.

VI. TEST PROBLEMS

Program integrity and accuracy have been verified by solving a number of test problems. Since stresses, whose evaluation depends upon derivatives of displacements, are known to be less accurate than temperatures, comparisons with exact results are confined to stresses. Individual problems are described below.

A. THICK CYLINDER

Consider a thick cylinder with inside radius 30 inches and outside radius 50 inches and the following material properties.

Modulus of elasticity	$E = 28.9 \times 10^6$	Psi
Poisson's ratio	$\nu = .28$	-
Coefficient of thermal expansion	$\alpha = 7.22 \times 10^{-6}$	$1/F^\circ$
Thermal conductivity	$k = 28.$	$\frac{\text{Btu}}{\text{hr. ft}^\circ\text{F}}$
Density	$\rho = 489.$	Lbm/ft^3
Specific heat	$c = .111$	$\frac{\text{Btu}}{\text{Lbm.}^\circ\text{F}}$

An arbitrary length of 25 inches has been selected for the cylinder and it has been subdivided into two different element representations as in Figs. 4 and 5. The plane-end boundary condition with zero axial force is used. The stresses for various types of loading are compared with the corresponding exact analytic solutions as described below.

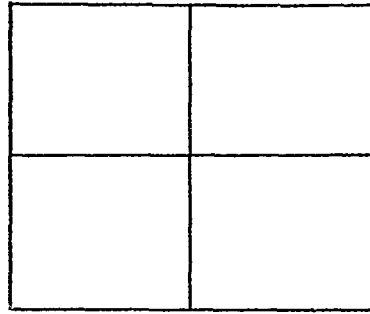


Figure 4. Two Radial Elements Representation of Thick Cylinder.

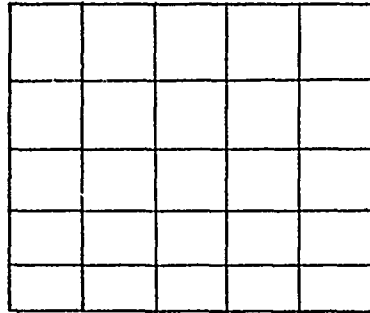


Figure 5. Five Radial Element Representation of Thick Cylinder.

1. Thermal Loading

For a linear variation of the temperature $T = 20R$ the stresses obtained by the finite element method for the above representations are compared with the exact analytic solution in Fig. 6. The τ_{RZ} for this problem clearly is zero and the one obtained by the program was 10^{-9} . The accuracy of the other results is clearly satisfactory.

2. Pressure Loading

A uniform pressure of 1000 psi. acts on the inner surface of the cylinder. Again, τ_{RZ} is zero and the program gives 10^{-10} . In Fig. 7 the other stresses induced by this uniform pressure are compared with the exact solutions. Here also the accuracy of the results, even with two radial elements, is adequate.

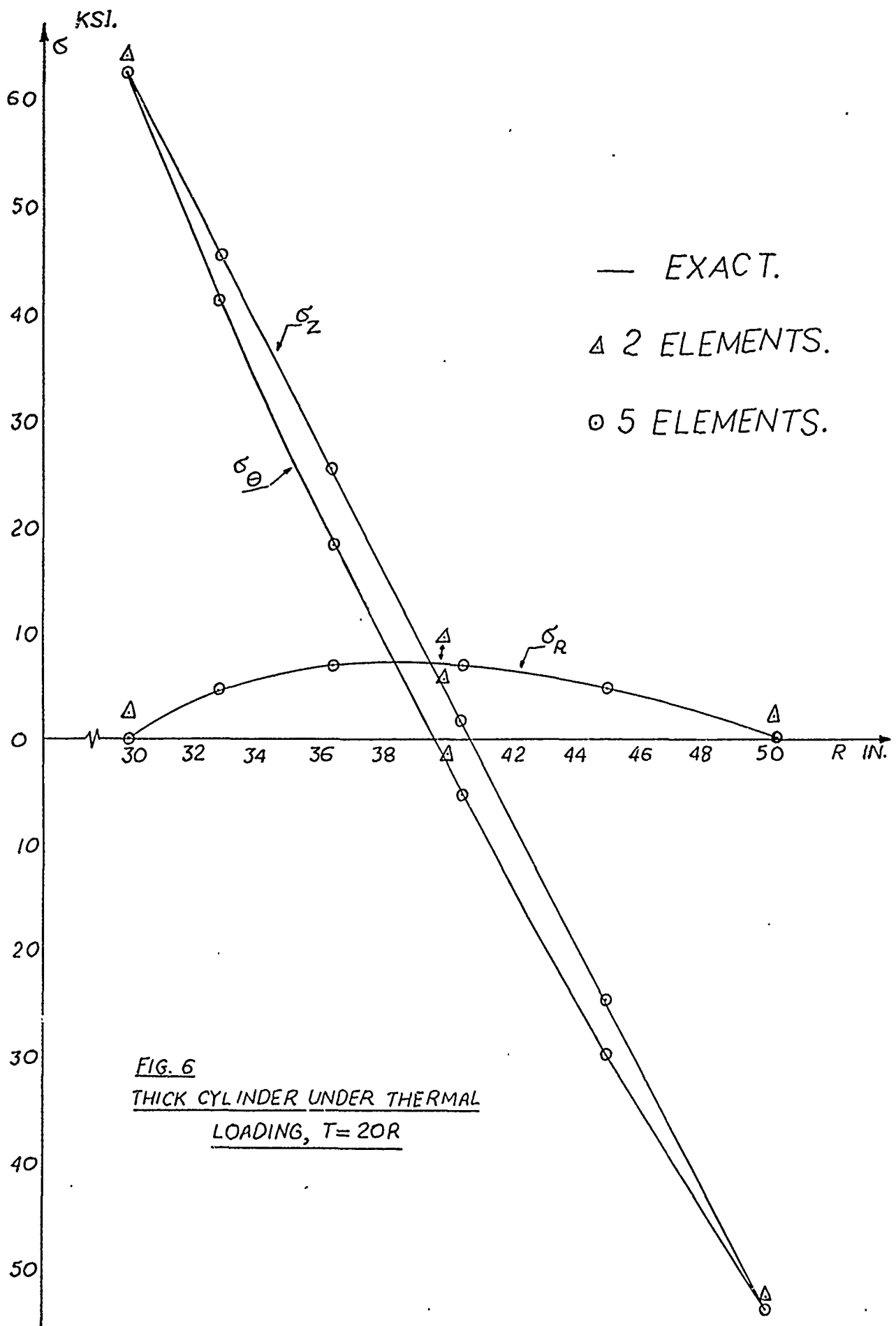


FIG. 6
 THICK CYLINDER UNDER THERMAL
 LOADING, $T = 20R$

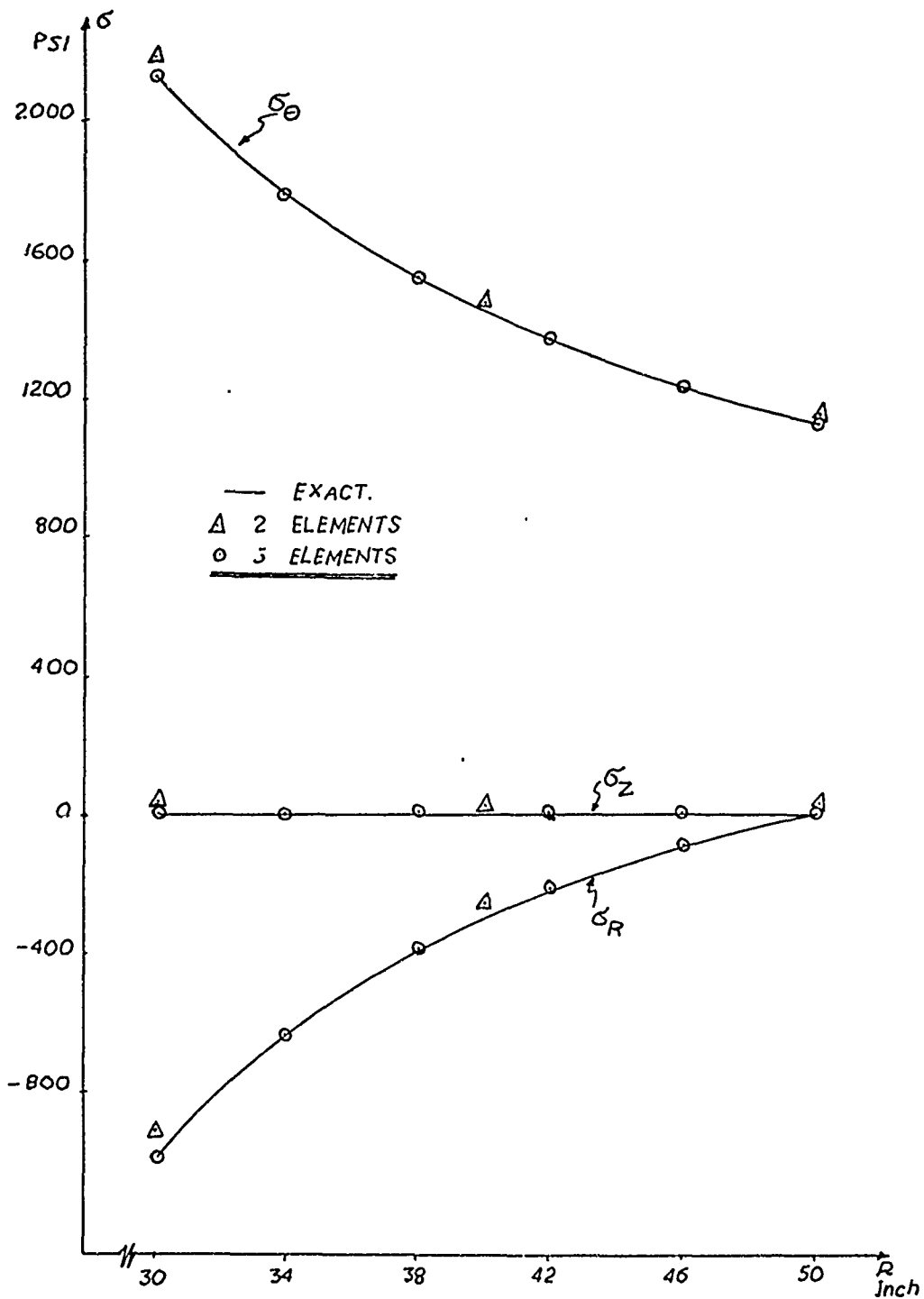


FIG. 7 THICK CYLINDER UNDER INTERNAL PRESSURE.

3. Centrifugal Loading

The speed of rotation has been assumed to be 500 revolutions per minute. The stresses obtained by the program are compared with the analytic solution in Fig. 8. In this case also the results obtained by the program, even with only two radial elements, are very close to the exact solution.

B. HOLLOW SPHERE

We consider a hollow sphere with the same material properties as in the thick cylinder with inside spherical radius 30 inches and outside 50 inches. The loading is thermal with $T = 20 \cdot (\text{spherical radius})$. Symmetry permits using only half of the sphere for the computer analysis. The elements representation is given in Fig. 9. Since the program gives stresses in cylindrical coordinates, these have been transformed to the spherical coordinates for comparison with the exact solution in Fig. 10. The accuracy of the results is noteworthy.

C. THERMAL STRESSES IN NOZZLE

This problem concerns thermal stresses near the intersection of a cylindrical pipe and the spherical vessel. Fig. 11 gives the cross-section of the structure which is to be analysed. The material properties are:

Modulus of elasticity	$E = 29.3 \times 10^6$	Psi
Poisson's ratio	$\nu = .30$	-
Thermal expansion coefficient	$\alpha = 7.6 \times 10^{-6}$	$1/F^\circ$

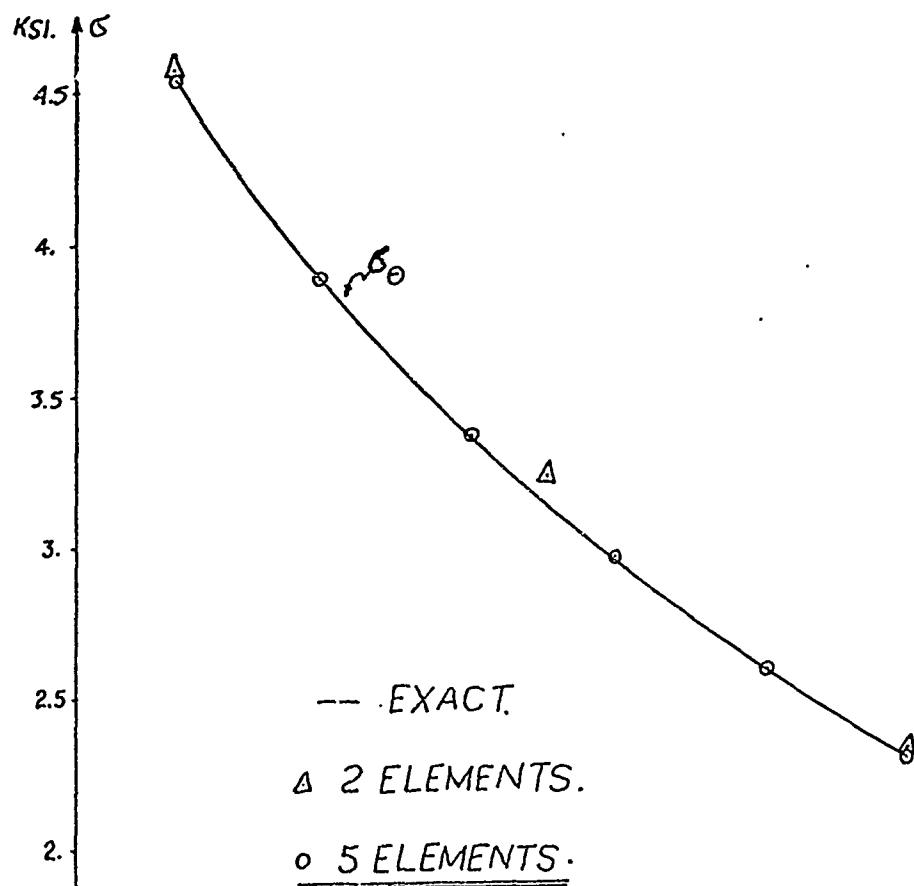


FIG. 8

ROTATING CYLINDER

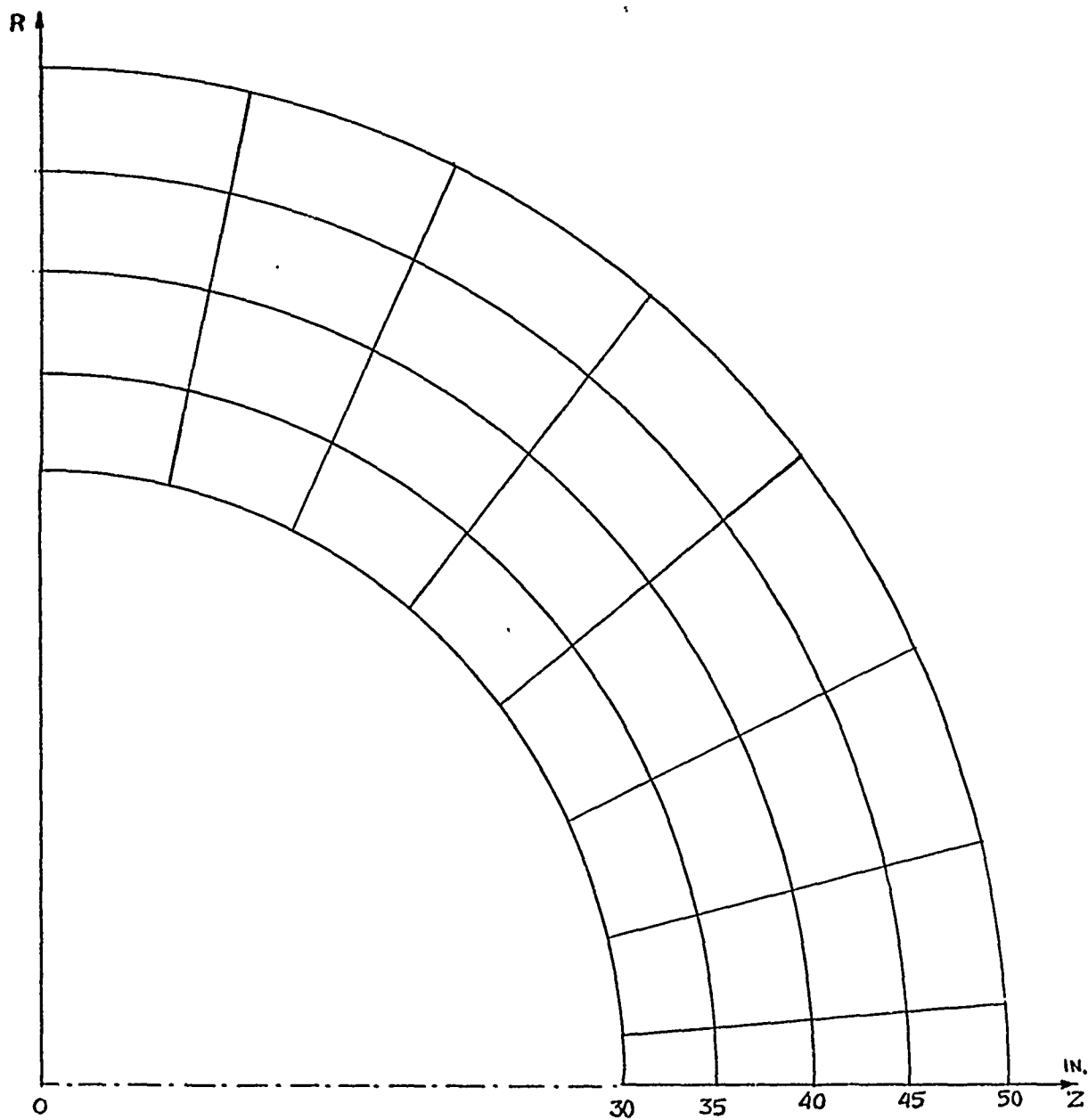


Figure 9. Hollow Semi-sphere 32 Elements Representation.

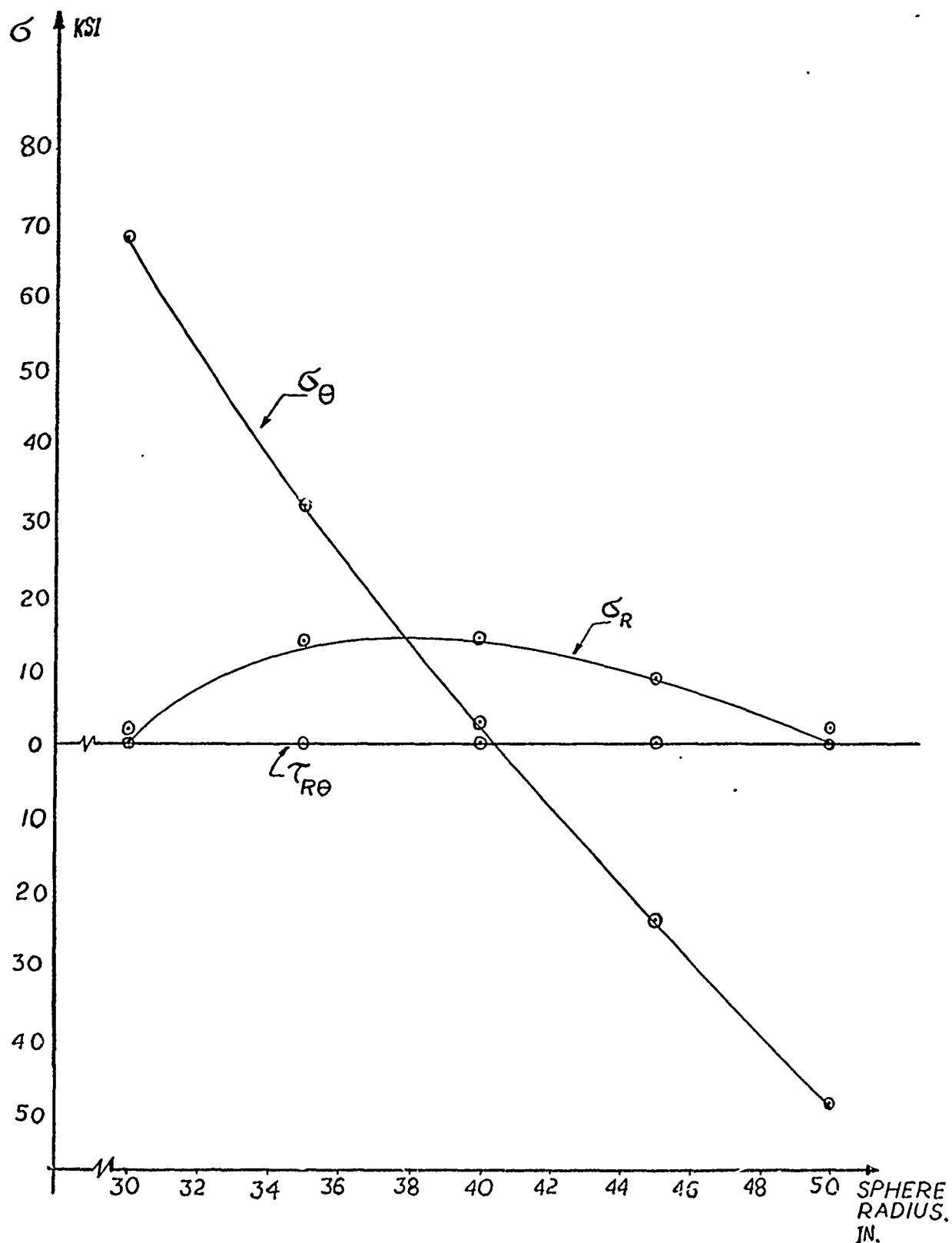
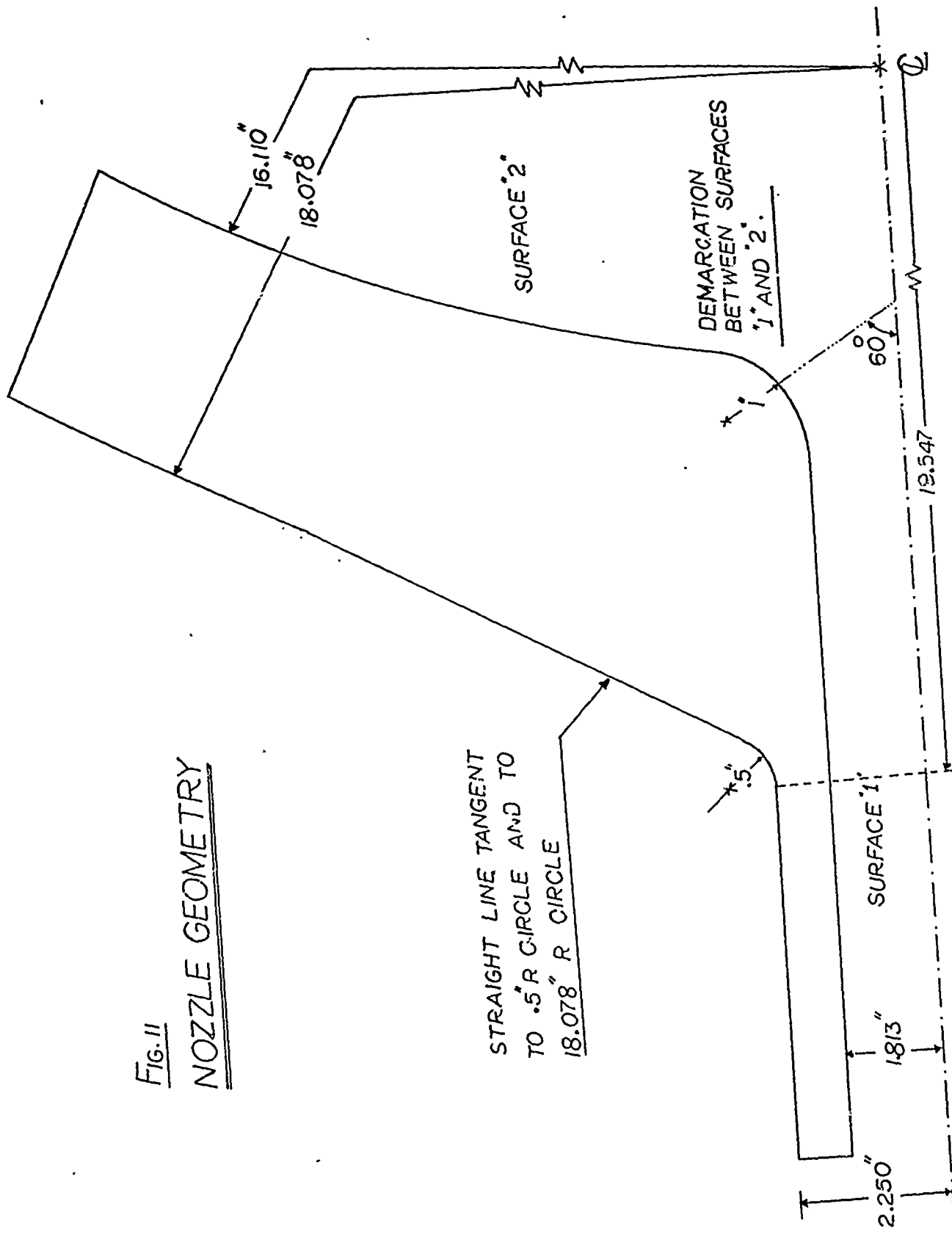


Figure 10. Thermal Stresses in Hollow Sphere
 $T = 20$ (spherical radius).

FIG. 11
NOZZLE GEOMETRY



Thermal conductivity	$k = 10.25 \text{ Btu/ft.hr.}^{\circ}\text{F}$
Density	$\rho = 530.5 \text{ Lbm/ft}^3$
Specific heat	$c = .128 \text{ Btu/Lbm}^{\circ}\text{F}$

The loading results from the thermal transient in the fluid contained in the nozzle. This fluid is in contact with the structure on surface "1" (Fig. 11) and has the entry temperature time variation as in Fig. 12. The fluid in the sphere, which is in contact with the structure on surface "2" (Fig. 11), has the constant temperature 478°F. At $\tau = 0$ structure has a uniform temperature of 478°F and is stress free. Exterior surface of the structure is insulated. The flow velocity past surface 1 is 8.5 ft/sec (inward) and there is no flow past surface 2. The surface heat transfer coefficients are 1393 and 2910 Btu/hr.ft²°F for surfaces "1" and "2" respectively.

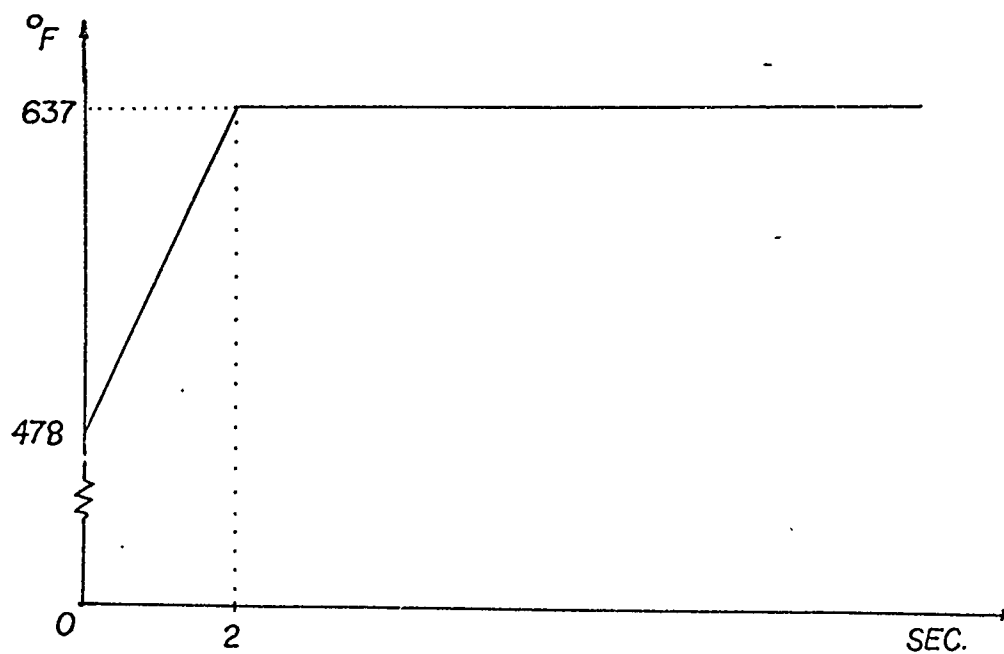
For the structural boundary condition it is assumed that the nodes on the left end cross-section of Fig. 13 are prevented from any axial displacement.

The maximum thermal stresses obtained by the program occur in element 14 of Fig. 13 as follows:

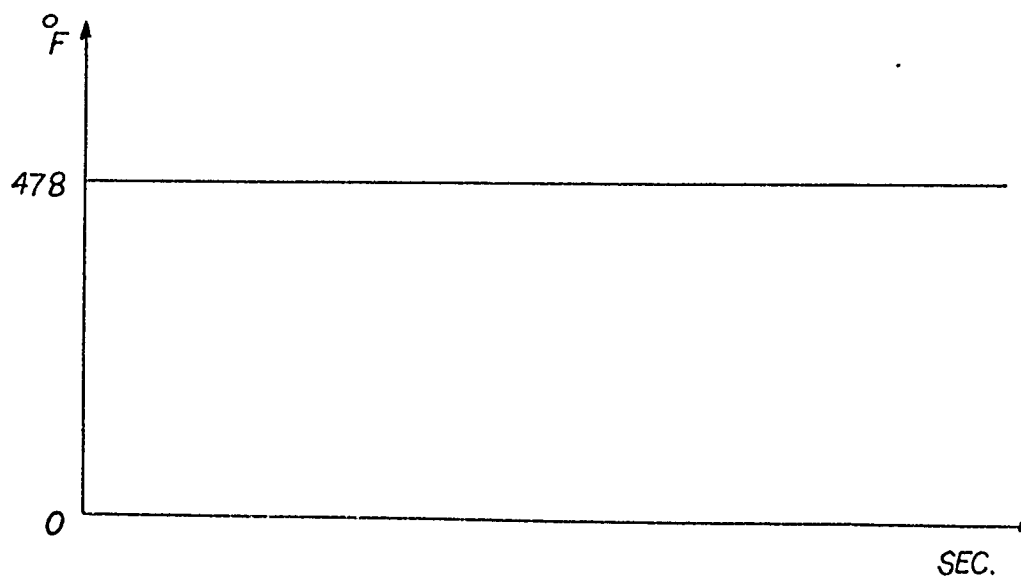
Maximum mean stress = 18.81 ksi. at time = 4 seconds;

Maximum octahedral shearing stress = 11.5 ksi. at time
= 18 seconds.

These results appear to be reasonable, but no suitable comparison solution is available.



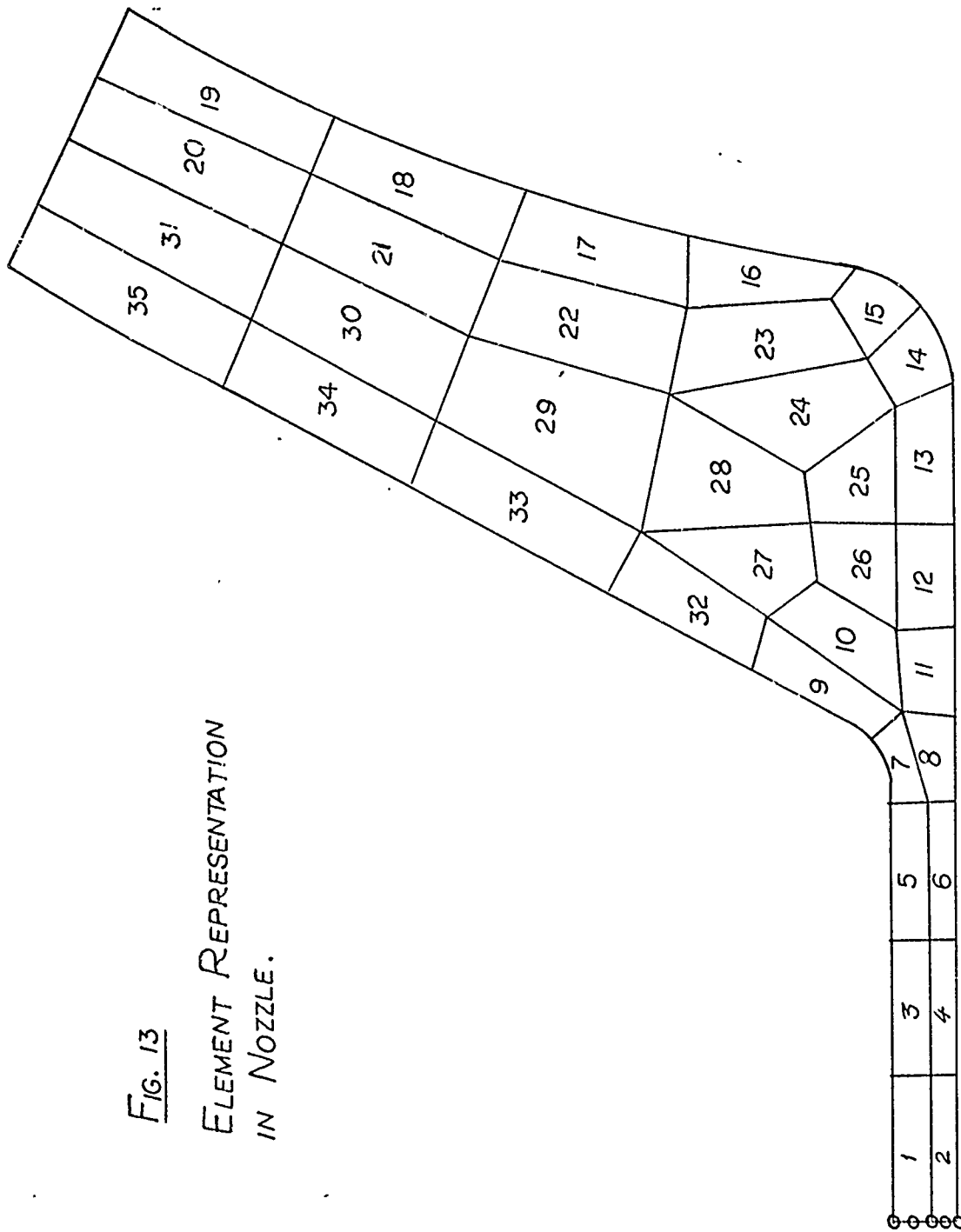
VARIATION OF ENTRY FLUID TEMPERATURE
IN NOZZLE. (SURFACE "1".)



FLUID PAST SURFACE "2".

FIG. 12. Fluid Temperature - Time Histories.

FIG. 13
ELEMENT REPRESENTATION
IN NOZZLE.



VII. CONCLUSIONS AND RECOMMENDATIONS

A. CONCLUSIONS

A computer program has been developed for the solution of axisymmetric transient heat conduction and thermal stress problems. The system will accommodate a wide variety of geometric arrangements, thermal and structural boundary conditions, and mechanical loadings.

Although double-precision arithmetic is employed throughout, efficient algorithms for the manipulation and storage of large symmetric banded matrices allow in-core solutions with modest time requirements.

The quadratic isoparametric elements used allow accurate representation of curvilinear boundaries and the stress field. Examples presented show that a small number of elements is generally sufficient to determine stresses with good precision.

B. RECOMMENDATIONS

Incorporation of several additional features would significantly increase the program capabilities. The following extensions are recommended.

1. Material thermal and elastic properties have been assumed constant within each element. Since such properties are generally temperature dependent, provisions should be made for periodically "updating" them during both temperature and stress solutions.

2. The surface heat transfer coefficient, taken as constant in the program, is a function of temperature and flow velocity. Provision should be made to include these effects.

3. The program presently starts every temperature solution with constant initial solid temperature. Provision for an externally specified initial temperature vector should be included.

4. The large quantity of temperature and stress results generated by the program is currently presented as digital printout. Graphical output in the form of two-dimensional contour plots of temperatures and stresses should be provided.

APPENDIX A
APPLICABLE FORMULAS AND EQUATIONS

PART 1

(a): Two dimensional 8-noded (parabolic) isoparametric shape functions.

Corner nodes:

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

Mid nodes:

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

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

where

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

(b): Temperature at a point in terms of the nodal temperatures.

$$T = \sum_{i=1}^8 N_i T_i$$

(c): Coordinates at a point in terms of the nodal coordinates.

$$R = \sum_{i=1}^8 N_i R_i$$

$$Z = \sum_{i=1}^8 N_i Z_i$$

(d): The Jacobian coordinate transformation matrix.

$$\underline{\underline{J}} = \begin{bmatrix} \frac{\partial Z}{\partial \xi} & \frac{\partial R}{\partial \xi} \\ \frac{\partial Z}{\partial \eta} & \frac{\partial R}{\partial \eta} \end{bmatrix}$$

(e): Element of the finite element capacitance matrix.

$$C_{ij}^e = \rho c \int N_i N_j dV$$

(f): Element of the finite element admittance matrix.

$$y_{ij}^e = k \int \bar{\nabla} N_i \cdot \bar{\nabla} N_j dV$$

(g): Elemental volume.

$$dV = 2\pi R \det \underline{\underline{J}} d\xi d\eta$$

PART 2

(a): One-dimensional 3-noded (parabolic) isoparametric shape functions.

$$\text{End nodes} \quad N_i = \frac{1}{2} \xi_0 (1 + \xi_0)$$

$$\text{Mid node} \quad N_i = (1 - \xi^2)$$

$$\text{where, again, } \xi_0 = \xi \xi_i$$

(b): One-dimensional capacitance and admittance matrices.

$$\underline{\underline{C}}^e = \frac{\rho c \ell}{30} \begin{bmatrix} 4 & 2 & -1 \\ 2 & 16 & 2 \\ -1 & 2 & 4 \end{bmatrix}, \quad \underline{\underline{Y}}^{+e} = \frac{k}{3\ell} \begin{bmatrix} 7 & -8 & 1 \\ -8 & 16 & -8 \\ 1 & -8 & 7 \end{bmatrix} + \begin{bmatrix} h & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

where ℓ is the length of the element.

(c): The element $\underline{v}$ vector for the one-dimensional case.

$$\underline{v}^e = \langle h T_f \ 0 \ 0 \rangle^T$$

PART 3

(a): Strain-displacement relations.

$$\epsilon_Z = \frac{\partial w}{\partial Z} = \sum \frac{\partial N_i}{\partial Z} w_i$$

$$\epsilon_R = \frac{\partial u}{\partial R} = \sum \frac{\partial N_i}{\partial R} u_i$$

$$\epsilon_\theta = \frac{u}{R} = \frac{\sum N_i U_i}{\sum N_i R_i}$$

$$\dot{\gamma}_{RZ} = \frac{\partial u}{\partial Z} + \frac{\partial w}{\partial R} = \sum \frac{\partial N_i}{\partial Z} U_i + \sum \frac{\partial N_i}{\partial R} w_i$$

where N_i are the same isoparametric shape function as in Part 1, (a).

(b): The B matrix.

$$\underline{\underline{B}} = \begin{bmatrix} 0 & \frac{\partial N_1}{\partial Z} & 0 & \frac{\partial N_2}{\partial Z} & \dots\dots\dots & \frac{\partial N_8}{\partial Z} \\ \frac{\partial N_1}{\partial R} & 0 & \frac{\partial N_2}{\partial Z} & 0 & \dots\dots\dots & 0 \\ \frac{N_1}{R} & 0 & \frac{N_2}{R} & 0 & \dots\dots\dots & 0 \\ \frac{\partial N_1}{\partial Z} & \frac{\partial N_1}{\partial R} & \frac{\partial N_2}{\partial Z} & \frac{\partial N_2}{\partial R} & \dots\dots\dots & \frac{\partial N_8}{\partial R} \end{bmatrix}$$

(c): The element displacement vector.

$$\underline{\underline{\delta}}^e = \langle u_1 \ w_1 \ u_2 \ w_2 \ \dots\dots\dots u_8 \ w_8 \rangle^T$$

(d): Elasticity matrix D for an isotropic material.

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

000000030	000000040	000000050	000000060	000000070	000000080	000000090	000000100	000000110	000000120	000000130	000000140	000000150	000000160	000000170	000000180	000000190	000000200	000000210	000000220	000000230	000000240	000000250	000000260	000000270	000000280	000000290	000000300	000000310	000000320	000000330	000000340	000000350	000000360	000000370	000000380	000000390	000000400	000000410	000000420	000000430	000000440	000000450	000000460	000000470	000000480	000000490	000000500
-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------

3711

AXIAL SYMMETRIC TRANSIENT
(THERMAL STRESSES.)

COMPLETION DATE

ADVISER

DR. ROBERT E. NEWTON
PROF. MECH. ENG. N. P. G. S.

B. OBJECTIVE

THIS PROGRAM IS USED FOR THE FOLLOWING CASES:

1- TO EVALUATE TRANSIENT TEMPERATURE IN AN AXISYMMETRIC BODY FOR THE COMBINATION OF THE CONSTANT TEMPERATURE, INSULATED AND CONVECTION THERMAL BOUNDARY CONDITION.

2-- TO EVALUATE THE STRESSES IN AN AXISYMMETRIC BODY FOR ANY COMBINATION OF AXIAL, PRESSURE, CENTRIFUGAL OR ANY EXTERNALLY SPECIFIED LOAD VECTOR.

3-- TO EVALUATE TRANSIENT THERMAL STRESSES IN AN AXISYMMETRIC BODY FOR ANY KNOWN TRANSIENT TEMPERATURE OR USING THE RESULTS OF STEP 1. THE COMBINATION OF THERMAL LOADING WITH ANY OTHER LOADING AS IN STEP 2 MAY BE USED ALSO.

C. METHOD OF CALCULATION

THE FINITE ELEMENT METHOD WITH EIGHT NODDED ISO-PARAMETRIC ELEMENTS IS APPLIED FOR THE ANALYSIS.

D. USAGE

REFER TO THE (APPENDIX C) OF THE MECHANICAL ENG.
THESIS WRITTEN BY THE PROGRAMMER.

[illegible]

```

00002290
00002300
00002310
00002320
00002330
00002340
00002350
00002360
00002370
00002380
00002390
00002400
00002410
00002420
00002430
00002440
00002450
00002460
00002470
00002480
00002490
00002500
00002510
00002520
00002530
00002540
00002550
00002560
00002570
00002580
00002590
00002600
00002610
00002620
00002630
00002640
00002650
00002660
00002670
00002680
00002690
00002700
00002710
00002720
00002730
00002740

SUBROUTINE INPUT
*****
      THIS SUBROUTINE READS AND WRITES THE INPUT DATA OF
      THE GEOMETRY AND MATERIAL PROPERTIES. IT CALCULATES
      THE MID NODE COORDINATES BASED ON STRAIGHT AND THE
      SYSTEM OF UNITS IS SELECTED HERE. STIFFNESS MATRIX
      AT THE END THE BANDWIDTH OF THE STIFFNESS MATRIX
      IS DETERMINED.
*****

      IMPLICIT REAL * 8 (A-H,O-Z)
      REAL * 4 HCO,VCO,PLOTZR

      COMMON /ONE/ AJ(2,2),B(4,16),COPY(37,298),D(4,4),DL(4,4),
1DNZR(2,8),ESM(16,16),F(24,298),SSM(298,66),STCR(20,150)

      COMMON /TWO/ AL(5),DENS(5),DETA(8),DXI(8),E(5),EPSO(4),
1FE(16),GC(4),GX(4),POI(5),R(149),SHP(8),SHT(5),TK(5),Z(149)

      COMMON /TRE/ AXIALF,DTI,EVERY,FDRSL,OMEGA,HTCI,HTCO,
1POST,PRES,Q1,Q2,Q3,SHMAX,SMAX,TIME,TIMEL,TINIT,TMAXM,TREF,
2TMAXO,RMAXM,RMAXO,ZMAXM,ZMAXO,Q4,BIG

      COMMON /FOR/ NN(40,9)

      COMMON /FIV/ NCFI(37),NCFO(37),NNR(37),NRE(37),NNL(37),
1NPN(37)
      COMMON /SIX/ IBAND,IEXT,INIP,IVEC,JVEC,KK,LOAD,NCFIT,NCFOT,
1NCHECK,NET,NDF,NGP,NNE,NNLT,NNRT,NPASS,NPROB,NRPPI,NRPPO,
3NNI,NNRE,NPNT,IPLANE,NBAND

      EQUIVALENCE (C,SSM), (Y,SSM(1,18)), (T,SSM(1,35)),
1(BTE,SSM(1,36)), (TEMP,SSM(1,37)), (V,SSM(1,38)),
2(FIT,SSM(1,39)), (FIT2,SSM(1,40)), (CE,SSM(1,41)),
3(YE,SSM(1,42)), (OFF1,SSM(1,43)), (OFF2,SSM(1,44)),
4(DIAG,SSM(1,45)), (BDRY1,SSM(1,46)), (BDRYJ,SSM(1,47)),
5(ENTEI,SSM(1,48)), (ENTY1,SSM(1,49)), (ENTEI,SSM(1,50)),
6(ENTEQ,SSM(1,51)), (VCL,SSM(1,52)), (VCO,SSM(1,53))
      EQUIVALENCE (HCO,SSM(1,54)), (VCO,SSM(1,55))

      DIMENSION C(149,33),Y(149,33),T(149),BTE(149),TEMP(149),

```



```

DO 1 I = 1, NMAT
1 READ (5,303) E(I),AL(I),POI(I),TK(I),DENS(I),SHT(I)
303 FORMAT (2D12.4,4F8.3)
C
C
C ARE THE UNITS IN METRIC OR BRITISH UNIT SYSTEM?
READ (5,310) Q1
10 FORMAT (A4)
C
WRITE (6,200)
200 FORMAT (1H1,45X,'** TRANSIENT THERMAL STRESS PROBLEM',
1, **',/,49X,32('),//)
C
C
WRITE (6,201) NET,NNT,NCN,NMAT,NPROB
201 FORMAT (/,15X,'TOTAL NUMBER OF ELEMENTS',15,
1/,15X,'TOTAL NUMBER OF NODES',15,/,15X,
2,'TOTAL NUMBER OF CORNER NODES',15,/,15X,'TOTAL',
3,'NUMBER OF MATERIALS',15,/,15X,'TOTAL NUMBER',
4,'OF PROBLEMS',15,//)
C
C
WRITE (6,204)
204 FORMAT (/,15X,'ELEMENT NO.',8X,'GLOBAL NODE NUMBER',
132X,'MATERIAL NO.',/)
C
C
DO 2 I=1, NET
2 WRITE (6,205) I,(NN(I,J),J=1,K)
205 FORMAT (15X,I5,10X,9(15,2X))
C
C
IF (Q1.NE.BRIT) GO TO 10
C
WRITE (6,211)
WRITE (6,213)
GO TO 11
C
C
10 WRITE (6,212)
11 WRITE (6,215)
WRITE (6,202)
C
C
202 FORMAT (/,15X,'THE RADIAL AND LONGITUDINAL',
1, COORDINATES',//,15X,'NODE NO.',14X,'**R**',
212X,'**Z**',/)

```

```

00003230
00003240
00003250
00003260
00003270
00003280
00003290
00003300
00003310
00003320
00003330
00003340
00003350
00003360
00003370
00003380
00003390
00003400
00003410
00003420
00003430
00003440
00003450
00003460
00003470
00003480
00003490
00003500
00003510
00003520
00003530
00003540
00003550
00003560
00003570
00003580
00003590
00003600
00003610
00003620
00003630
00003640
00003650
00003660
00003670
00003680
00003690
00003700

```


00004670
00004680
00004690
00004700
00004710
00004720
00004730
00004740
00004750
00004760

```
C      NDF = 2 * NNT
      WRITE (6,220) IBAND, NDF
220  FORMAT (/,'15X',HALF-BAND-WIDTH OF THE STIFFNESS MATRIX IS',I5,/)
C      118,/,15X,NUMBER OF DEGREES OF FREEDOM OF THE SYSTEM IS',I5,/)
C      RETURN
C      END
```

```

00004770
00004780
00004790
00004800
00004810
00004820
00004830
00004840
00004850
00004860
00004870
00004880
00004890
00004900
00004910
00004920
00004930
00004940
00004950
00004960
00004970
00004980
00004990
00005000
00005010
00005020
00005030
00005040
00005050
00005060
00005070
00005080
00005090
00005100
00005110
00005120
00005130
00005140
00005150
00005160
00005170
00005180
00005190
00005200
00005210
00005220

SUBROUTINE PROB
*****
FOR EACH PROBLEM, IN THIS SUBROUTINE THE THERMAL
INITIAL AND BOUNDARY CONDITIONS AND THE VARIOUS
OTHER TYPES OF LOADS WITH SPECIFIED STRUCTURAL
BOUNDARY CONDITIONS ARE BEING READ IN AND PRINTED
OUT. THE STEP SIZE OF TIME INTEGRATION, THE INTERVAL
OF TEMPERATURE PRINTS IF REQUIRED, AND THE NUMBER
OF THERMAL LOAD VECTORS WANTED ARE READ IN HERE.
*****
IMPLICIT REAL * 8 (A-H,O-Z)
COMMON /ONE/ AJ(2,2),B(4,16),COPY(37,298),D(4,4),DL(4,4),
1DNZR(2,8),ESM(16,16),F(24,298),SSM(298,66),STOR(20,150)
COMMON /TWO/ AL(5),DENS(5),DETA(8),DXI(8),E(5),EPSO(4),
1FE(16),GC(4),GX(4),PCI(5),R(149),SHP(8),SHT(5),TK(5),Z(149)
COMMON /TRE/ AXIALF,DTI,EVERY,FORS1,OMEGA,HICI,HICO,
1POSI,PRES,Q1,Q2,Q3,SHMAX,SMAX,TIME,TIME1,INIT,TMAXM,TREF,
2TMAXG,RMAXM,RMAXO,ZMAXM,ZMAXO,Q4,BIG
COMMON /FOR/ NN(40,9)
COMMON /FIV/ NCFI(37),NCFO(37),NNR(37),NRE(37),NNL(37),
1NPN(37)
COMMON /SIX/ IBAND,IEXT,INTP,IVEC,JVEC,KK,LOAD,NCFIT,NCFOI,
1NCHECK,NET,NDF,NGP,NNE,NNLT,NNRT,NPASS,NPRCB,NRPPI,NRPPO,
3NNNT,NNRE,NPNT,IPLANE,NBAND
EQUIVALENCE (C,SSM), (Y,SSM(1,18)), (T,SSM(1,35)),
1(BTE,SSM(1,36)), (TEMP,SSM(1,37)), (V,SSM(1,38)),
2(FI1,SSM(1,39)), (FI2,SSM(1,40)), (CE,SSM(1,41)),
3(YE,SSM(1,42)), (OFF1,SSM(1,43)), (OFF2,SSM(1,44)),
4(DIAG,SSM(1,45)), (DDRY1,SSM(1,46)), (BDRYO,SSM(1,47)),
5(ENT11,SSM(1,48)), (ENT1IO,SSM(1,49)), (ENT1EI,SSM(1,50)),
6(ENT1EO,SSM(1,51)), (VCL,SSM(1,52)), (VOLO,SSM(1,53))
DIMENSION C(149,33),Y(149,33),T(149),8TE(149),TEMP(149),
1V(149),FI1(90),FI2(90),CE(8,8),YE(8,8),OFF1(37),OFF2(37),

```

```

C      2DIAG(37),BDRYI(16,5),BDRYO(16,5),VOL(36),VOLO(36),ENTE(75),
C      3ENTEI(75),ENTIO(75),ENTII(75)
C      DATA BRIT/'BRIT'//
C      TIME = 0.00
C      JVEC = 0
C      SHMAX = 0.00
C      SHMAX = 0.00
C      NGP = 4
C      GX(1) = -.861136311594053
C      GX(2) = -.GX(1)
C      GX(3) = .339981043584856
C      GX(4) = -.GX(3)
C      GC(1) = .347854845137454
C      GC(2) = .GC(1)
C      GC(3) = .652145154862546
C      GC(4) = .GC(3)
C      PAI = 3.141592653589793
C      NNRE IS THE TOTAL NUMBER OF NODES AT RIGHT HAND END.
C      NPRE(I) IS THE ARRAY OF GLOBAL NODE NUMBER FOR THE
C      RIGHT HAND END.
C      IPLANE.GT.0 IF PLANES REMAIN PLANE AT THE ENDS.
C      IPLANE.EQ.0 IF ONE END IS FREE OR RADIALY FIXED.
C      IEXT IS AN INDICATION IF TEMPERATURE VECTORS ARE
C      GENERATED OR BEING READ IN AS FOLLOWS.
C      IEXT.EQ.0 TEMPERATURE VECTORS ARE GENERATED.
C      IEXT.GT.0 TEMPERATURE VECTORS ARE INPUTED.
C      READ (5,101)OMEGA,PRES,IEXT,LOAD,NNLT,NNRT,IPLANE
C      IF (OMEGA.EQ.0.00) GO TO 2
C      JVEC = JVEC + 1
C      WRITE (6,203) OMEGA
C      2 N = 0
C      IF (PRES.EQ.0.00) GO TO 6
C      READ (5,102) NPNT,(NPN(I),I=1,NPNT)
C      NPNT IS THE TOTAL NUMBER OF PRESSURE NODES
C      NPN(I) IS THE GLOBAL NODE NUMBER OF THE PRESSURE NODE I .
C      WRITE (6,444) NPNT,(NPN(I),I=1,NPNT)

```

```

00005230
00005240
00005250
00005260
00005270
00005280
00005290
00005300
00005310
00005320
00005330
00005340
00005350
00005360
00005370
00005380
00005390
00005400
00005410
00005420
00005430
00005440
00005450
00005460
00005470
00005480
00005490
00005500
00005510
00005520
00005530
00005540
00005550
00005560
00005570
00005580
00005590
00005600
00005610
00005620
00005630
00005640
00005650
00005660
00005670
00005680
00005690
00005700

```

```

444 FORMAT (15X,'TOTAL NUMBER OF PRESSURE NODES IS ',I5,' ',
1,AND THEY ARE AS :',/,3(15X,20(I3,' ',),/),/),
JVEC = JVEC + 1
WRITE (6,207) PRES
5 CONTINUE
C
IF (LOAD.EQ.0) GO TO 50
JVEC = JVEC + 1
L = IVEC + JVEC
READ (5,108) (F(L,I), I=1, NDF)
WRITE (6,333) (F(L,I), I=1, NDF)
333 FORMAT (/,15X,'THE EXTERNAL LOAD VECTOR IN ORDER OF ',
1,THE NODE NUMBERS IS :',/,5(15X,F10.4),/),
50 CONTINUE
C
IF (NNLT.NE.0) READ (5,304) (NNL(I), I=1, NNLT)
IF (NNLT.NE.0) WRITE (6,308) (NNL(I), I=1, NNLT)
308 FORMAT (/,15X,'TOTAL NUMBER OF NODES FIXED IN THE ',
1,LONGITUDINAL DIRECTION IS ',I5,' AND THEY ARE AS :',
2/,3(15X,20(I3,' ',),/),/),
C
IF (NNRT.NE.0) READ (5,304) (NNR(I), I=1, NNRT)
IF (NNRT.NE.0) WRITE (6,309) (NNR(I), I=1, NNRT)
309 FORMAT (/,15X,'TOTAL NUMBER OF NODES FIXED IN THE ',
1,RADIAL DIRECTION ARE ',I5,' AND THEY ARE AS :',
23(15X,20(I3,' ',),/),/),
304 FORMAT (10I5)
C
IF (IPLANE.NE.0) READ (5,102) NNRE, (NRE(I), I=1, NNRE)
IF (IPLANE.NE.0) WRITE (6,214) NNRE, (NRE(I), I=1, NNRE)
214 FORMAT (/,15X,'TOTAL NUMBER OF NODES FIXED IN LONGIT',
1,UDINAL DIRECTION OF RIGHT HAND END IS ',I5,' AND TH',
3,EY ARE AS :',/,2(15X,20(I3,' ',),/),/),
C
READ (5,100) TINIT, Q2, Q3, Q4, AXIALF
C
IF (AXIALF.EQ.0.D0) WRITE (6,220)
220 FORMAT (/,15X,'THERE IS NO AXIAL FORCE ON THE BODY',/)
221 FORMAT (/,15X,'AXIAL FORCE = ',F11.4,/),
C
IF (IEXT.NE.0) GO TO 52
IF (Q4.LT.0.D0) GO TO 57
C
IF CONVECTION BDY. OUTSIDE Q2 > 0.D0
IF INSULATED BDY. OUTSIDE Q2 = 0.D0

```

```

00005710
00005720
00005730
00005740
00005750
00005760
00005770
00005780
00005790
00005800
00005810
00005820
00005830
00005840
00005850
00005860
00005870
00005880
00005890
00005900
00005910
00005920
00005930
00005940
00005950
00005960
00005970
00005980
00005990
00006000
00006010
00006020
00006030
00006040
00006050
00006060
00006070
00006080
00006090
00006100
00006110
00006120
00006130
00006140
00006150
00006160
00006170
00006180

```

```

CCCCCCCCCCCCC
C
IF CONVECTION BDRY. INSIDE Q3 > 0.00
IF INSULATED BDRY. Q3 = 0.00

Q4.GT.0.00 TEMPERATURE PROBLEM IS TO BE SOLVED ONLY
Q4 = 0.00 STRESS PROBLEM IS TO BE SOLVED ALSO.
Q4.LT.0.00 STRESS PROBLEM IS TO BE SOLVED ONLY.

IF (Q2.GT. 0.00) GO TO 1
GO TO 3
1 CONTINUE

READ (5,103) HTCO , TEMPO , NCFOT , NRAMPO
N = NCFOT
READ (5,102) (NCFO(I), I = 1 , NCFOT)
NRPPO = NRAMPO + 1

READ (5,104) ((BDRYO(I,J),J=1,3),BDRYO(I,5),I = 2,NRPPO)
BDRYO(I,1) IS THE INITIAL TIME, TO BE READ IN FOR
EACH RAMP SEGMENT.
BDRYO(I,2) IS THE INITIAL TEMPERATURE, TO BE READ IN.
BDRYO(I,3) IS THE FINAL TEMPERATURE TO BE READ IN IF IT DIFFERS
FROM THE NEXT INITIAL TEMPERATURE.
BDRYO(I,4) IS THE SLOPE, WHICH IS CALCULATED BY THE
PROGRAM.
BDRYO(I,5) IS THE VELOCITY OF THE FLUID TO BE READ IN
IF IT DIFFERS FROM THE PREVIOUS RAMP VELOCITY

WRITE (6,201) HTCO

DO 4 I = 1, NCFOT
FT2(I) = TEMPO
FT1(I) = TEMPO
4

WRITE (6,201) (NCFO(I),FT1(I),I=1,NCFOT)

BDRYO(1,1) = 0.00
BDRYO(1,2) = TEMPO
BDRYO(1,4) = 0.00

```

```

00006190
00006200
00006210
00006220
00006230
00006240
00006250
00006260
00006270
00006280
00006290
00006300
00006310
00006320
00006330
00006340
00006350
00006360
00006370
00006380
00006390
00006400
00006410
00006420
00006430
00006440
00006450
00006460
00006470
00006480
00006490
00006500
00006510
00006520
00006530
00006540
00006550
00006560
00006570
00006580
00006590
00006600
00006610
00006620
00006630
00006640
00006650
00006660

```

```

C      BDRYO(1,5) = 0.D0
      BDRYO(NRPPG,4) = 0.D0
      BDRYO(NRPPG+1,1) = 999999.D0

      DO 5 I = 2, NRAMPO
      IF (BDRYO(I,3).EQ.0.D0) BDRYO(I,3) = BDRYO(I+1,2)
      IF (BDRYO(I,5).EQ.0.D0) BDRYO(I,5) = BDRYO(I-1,5)
      DUM = BDRYO(I,3) - BDRYO(I,2)
      5 BDRYC(I,4) = DUM/(BDRYO(I+1,1) - BDRYO(I,1))

C      WRITE (6,202) (BDRYO(I,1),BDRYO(I,2),BDRYO(I+1,1),
      1BDRYO(I,3),BDRYO(I,4),BDRYO(I,5),I = 2,NRPPG)

C      IF (Q1.EQ.8RIT) GO TO 20
      DO 21 I = 2, NRPPG
      21 BDRYO(I,5) = BDRYO(I,5) * 100.D0
      GO TO 18
      20 DO 22 I = 2, NRPPG
      22 BDRYO(I,5) = BDRYO(I,5) * 12.D0
      GO TO 18

C      3 WRITE (6,1000)
      18 CONTINUE

C      IF (Q3.GT.0D0) GO TO 7
      WRITE (6,1001)
      GO TO 9

C      7 READ (5,103) HTCI, TEMPI, NCFIT, NRAMPI
      READ (5,102) (NCFI(I), I = 1, NCFIT)

C      NRPPI = NRAMPI + 1
      READ (5,104) ((BDRYI(I,J),J=1,3),BDRYI(I,5),I=2,NRPPI)

C      WRITE (6,2022) HTCI

C      DO 10 I = 1, NCFIT
      K = N + I
      FT1(K) = TEMPI
      10 FT2(K) = TEMPI

C      DO 19 I = 1, NCFIT

```

```

00006670
00006680
00006690
00006700
00006710
00006720
00006730
00006740
00006750
00006760
00006770
00006780
00006790
00006800
00006810
00006820
00006830
00006840
00006850
00006860
00006870
00006880
00006890
00006900
00006910
00006920
00006930
00006940
00006950
00006960
00006970
00006980
00006990
00007000
00007010
00007020
00007030
00007040
00007050
00007060
00007070
00007080
00007090
00007100
00007110
00007120
00007130
00007140

```

```

C
C
      K = N + I
      19 WRITE (6,201) NCFI(I) , FT1(K)

      BDRYI(1,1) = 0.00
      BDRYI(1,2) = TEMPI
      BDRYI(1,4) = 0.00
      BDRYI(1,5) = 0.00
      BDRYI(NRPP1,4) = 0.00
      BDRYI(NRPP1+1,1) = 999999.00
C
C
      DO 11 I = 2 , NRAMPI
      IF (BDRYI(1,3).EQ.0.00) BDRYI(1,3) = BDRYI(I+1,2)
      IF (BDRYI(1,5).EQ.0.00) BDRYI(1,5) = BDRYI(I-1,5)
      DUM = BDRYI(1,3) - BDRYI(1,2)
      11 BDRYI(1,4) = DUM / (BDRYI(I+1,1)-BDRYI(1,1))
C
      WRITE (6,202) (BDRYI(1,1),BDRYI(1,2),BDRYI(I+1,1),
      1 BDRYI(1,3),BDRYI(1,4),BDRYI(1,5),I=2,NRPP1)
C
      IF (Q1.EQ.BRIT) GO TO 23
      DO 24 I = 2 , NRPP1
      24 BDRYI(1,5) = BDRYI(1,5) * 100.00
      GO TO 9
      23 DO 25 I = 2 , NRPP1
      25 BDRYI(1,5) = BDRYI(1,5) * 12.00
C
      9 READ (5,106) DTI , TIME1 , EVERY , IVEC , INTP
C
C
      TS = TIME1 + EVERY * (IVEC-1)
C
      WRITE (6,215) DTI
      215 FORMAT (15X,'STEP SIZE OF TIME INTEGRATION IS ',F10.4,/)
C
      DO 14 I = 1,NNT
      TEMP(I) = TINIT
      WRITE (6,204) TINIT
      WRITE (6,205) TREF
      WRITE (6,206) IVEC
C
      WRITE (6,208) TIME1
      WRITE (6,209) EVERY
      IF (INTP.EQ.0) GO TO 40
      WRITE (6,210) INTP
      GO TO 41

```

```

000071150
000071160
000071170
000071180
000071190
00007200
00007210
00007220
00007230
00007240
00007250
00007260
00007270
00007280
00007290
00007300
00007310
00007320
00007330
00007340
00007350
00007360
00007370
00007380
00007390
00007400
00007410
00007420
00007430
00007440
00007450
00007460
00007470
00007480
00007490
00007500
00007510
00007520
00007530
00007540
00007550
00007560
00007570
00007580
00007590
00007600
00007610
00007620

```



```

115X, 'AND THE HEAT TRANSFER COEF. IS ', 1PD14.6, //)
2022 FORMAT (//, 15X, 'INSIDE BDRY. COND. IS CONVECTION', //,
115X, 'AND THE HEAT TRANSFER COEF. IS ', 1PD14.6, //)
C
C
52 IF (IEXT.EQ.0) GO TO 54
NNT1 = NNT + 1
READ (5, 109) ((STOR(I, J), J=1, NNT1), I=1, IEXT)
IVEC = IEXT
WRITE (6, 212) IEXT
212 FORMAT (//, 15X, 15, 'TEMPERATURE VECTORS ARE READ IN AND',
17, 15X, 'TEMPERATURES ARE IN ORDER OF GLOBAL NODE',
2, 'NUMBERS :', //)
C
DO 56 I = 1, IEXT
DO 56 J = 1, NNT
WRITE (6, 213) J, STOR(I, J), F10.3)
56
213 FORMAT (15X, 'NODE(', I3, ')', F10.3)
54 CONTINUE
54 IF (01.EQ.BRIT) GO TO 60
GO TO 62
C
C
60 IF (Q2.GT.0.D0) HTCO = HTCO / 518400.D0
IF (Q3.GT.0.D0) HTCI = HTCI / 518400.D0
C
C
GO TO 58
C
57 IVEC = 1
STOR(1, NNT+1) = 0.D0
DO 59 J = 1, NNT
STOR(I, J) = TREF
59 CONTINUE
62 OMEGA = OMEGA * (PAI / 30.D0)
C
BIG = 10.D0**20
100 FORMAT (5F10.4)
101 FORMAT (2F10.4, 5I5)
C
102 FORMAT (12I5)
103 FORMAT (D16.4, F10.4, 2I5)
C
104 FORMAT (4F10.4)
106 FORMAT (3F10.4, 2I5)
C
108 FORMAT (6F10.4)
109 FORMAT (6F10.3)
C
RETURN
END
00008110
00008120
00008130
00008140
00008150
00008160
00008170
00008180
00008190
00008200
00008210
00008220
00008230
00008240
00008250
00008260
00008270
00008280
00008290
00008300
00008310
00008320
00008330
00008340
00008350
00008360
00008370
00008380
00008390
00008400
00008410
00008420
00008430
00008440
00008450
00008460
00008470
00008480
00008490
00008500
00008510
00008520
00008530
00008540
00008550
00008560
00008570
00008580
00008590
00008600

```



```

00009070
00009080
00009090
00009100
00009110
00009120
00009130
00009140
00009150
00009160
00009170
00009180
00009190
00009200
00009210
00009220
00009230
00009240
00009250
00009260
00009270
00009280
00009290
00009300
00009310
00009320
00009330
00009340
00009350
00009360
00009370
00009380
00009390
00009400
00009410
00009420
00009430
00009440
00009450
00009460
00009470
00009480
00009490
00009500
00009510
00009520
00009530
00009540

```

```

C      DIFY = YMAX - YMIN
      FACTX = 100.0 / DIFY
      FACTY = 70.0 / DIFY

      DO 2 I = 1, N
        X(I) = FACTX * (X(I) - XMIN)
        Y(I) = 1.0 + FACTY * (Y(I) - YMIN)
      2 CONTINUE

C      WRITE (6,87) YMAX

C      JDUM(1) = IDQT
      IDUM(103) = IBLANK
      K = 71
      DO 9 I = 1, 71
        JDUM(104) = IDOT
      9 CONTINUE
      DO 6 J = 2, 103
        JDUM(J) = IBLANK
      6 IDUM(J) = IBLANK

C      DO 7 J = 1, N
      M = Y(J)
      IF (M.EQ.K) GO TO 8
      GO TO 7
      8 NUM = X(J) + 2
      IDUM(NUM) = ISTAR

C      K1 = J/100
      IF (K1.NE.0) GO TO 15
      K1 = J/10
      IF (K1.NE.0) GO TO 10
      JDUM(NUM) = NU(J)
      GO TO 7
      10 K2 = J - 10*K1
      JDUM(NUM) = NU(K1)
      IF (K2.EQ.0) K2 = 10
      JDUM(NUM+1) = NU(K2)
      GO TO 7

C      15 K3 = J - 100*K1
      K2 = K3 / 10
      K3 = K3 - 10*K2
      JDUM(NUM) = NU(K1)
      IF (K2.EQ.0) K2 = 10
      JDUM(NUM+1) = NU(K2)

```



```

C
77 IF (LL.NE.0) WRITE (6,77) IDUM
   FORMAT (1H1,/,42X,INSIDE FLUID ENTRY TIME TEMPERA-
   1,TURE PLOT,/,43X,40(,*,),/,7X,TIME,5X,TEMPERATURE,
   290X,TEMPERATURE,/,29X,102A1)
C
3 DO 3 I = 1, 101
  IDUM(I) = IBLANK
C
4 DO 4 I = 1, 75
  NUM = 1 + 100 * Y(I)/YMAX
  IDUM(NUM) = ISTAR
  WRITE (6,88) X(I),Y(I),IDUM
  IDUM(NUM) = IBLANK
48 CONTINUE
   FORMAT (5X,F8.2,4X,F10.3, .,102A1)
C
6 DO 6 I = 1, 101
  IDUM(I) = IDOT
  WRITE (6,99) IDUM
99 FORMAT (29X,102A1)
50 RETURN
C
END
00010360
00010370
00010380
00010390
00010400
00010410
00010420
00010430
00010440
00010450
00010460
00010470
00010480
00010490
00010500
00010510
00010520
00010530
00010540
00010550
00010560
00010570
00010580
00010590
00010600
00010610

```

```

00010620
00010630
00010640
00010650
00010660
00010670
00010680
00010690
00010700
00010710
00010720
00010730
00010740
00010750
00010760
00010770
00010780
00010790
00010800
00010810
00010820
00010830
00010840
00010850
00010860
00010870
00010880
00010890
00010900
00010910
00010920
00010930
00010940
00010950
00010960
00010970
00010980
00010990
00011000
00011010

```

```

SUBROUTINE LLT (N,A,WL)

```

```

*****
*
* THIS SUBROUTINE WILL DECOMPOSE ANY N BY N REAL
* SYMMETRIC MATRIX A INTO A LOWER TRIANGULAR MATRIX
* WL, OF THE SAME ORDER N, SUCH THAT THE PRODUCT
* OF WL AND ITS TRANSPOSE IS THE GIVEN MATRIX A.
* (THIS IS CHOLESKY DECOMPOSITION.)
*
*****

```

```

IMPLICIT REAL *8 (A-H,O-Z)
DIMENSION A(N,N),WL(N,N)
DO 7 I=1,N
  DO 8 J=1,N
    WL(I,J)=0.0DO
  CONTINUE
  WL(1,1)=DSQRT(A(1,1))
  DO 1 I=2,N
    WL(I,1)=A(I,1)/WL(1,1)
    NI=N-1
    M=3
    DO 2 I=2,N
      I1=I-1
      DO 3 J=1,I1
        WL(I,I1)=WL(I1,I)+WL(I,J)*WL(I,J)**2
        WL(I,I)=A(I,I)-WL(I,I1)*WL(I,I1)
        IF (M.GT.N) GO TO 9
      J=I
      I1=M
      DO 5 I1=M,N
        L=1,I1
        WL(I,I1)=A(I,I1)-WL(I,I1)*WL(I,I1)
        M=M+1
      RETURN
    END
  2 9

```

```

CCCCCCCCCCCCCCCC

```



```

C      2DIAG(37),BDRYI(16,5),BDRYQ(16,5),VOL(36),VOLO(36),ENTED(75),
3ENTEI(75),ENTIO(75),ENTII(75)
      DO 12 I = 1, NNT
      V(I) = 0.00
      DO 12 J = 1, NBAND
      C(I,J) = 0.00
      Y(I,J) = 0.00
12 CONTINUE
C
C      PAI = 3.141592653589793
      DO 25 L = 1, NET
C
C      M1 = NN(L,9)
      FACTC = 2.00 * PAI * DENS(M1) * SHT(M1)
      FACTY = 2.00 * PAI * TK(M1)
C
C      DO 7 I = 1, NNE
      DO 7 J = 1, NNE
      CE(I,J) = 0.00
7 YE(I,J) = 0.00
C
C      DO 9 N = 1, NGP
      ETA = GX(N)
C
C      DO 9 M = 1, NGP
      XI = GX(M)
C
C      SHAP = FUNCTIONS.
      SHP(1) = .2500 * (1.00-XI) * (1.00-ETA) * (-XI-ETA-1.00)
      SHP(2) = .5000 * (1.00-XI**2) * (1.00-ETA) * (XI-ETA-1.00)
      SHP(3) = .2500 * (1.00+XI) * (1.00-ETA) * (XI+ETA-1.00)
      SHP(4) = .5000 * (1.00-ETA**2) * (1.00+XI) * (-XI+ETA-1.00)
      SHP(5) = .2500 * (1.00+XI) * (1.00+ETA) * (-XI+ETA-1.00)
      SHP(6) = .5000 * (1.00-YI**2) * (1.00+ETA) * (-XI+ETA-1.00)
      SHP(7) = .2500 * (1.00-XI) * (1.00-ETA) * (-XI+ETA-1.00)
      SHP(8) = .5000 * (1.00-ETA**2) * (1.00-XI) * (-XI+ETA-1.00)

```

```

000111480
000111490
000111500
000111510
000111520
000111530
000111540
000111550
000111560
000111570
000111580
000111590
000111600
000111610
000111620
000111630
000111640
000111650
000111660
000111670
000111680
000111690
000111700
000111710
000111720
000111730
000111740
000111750
000111760
000111770
000111780
000111790
000111800
000111810
000111820
000111830
000111840
000111850
000111860
000111870
000111880
000111890
000111900
000111910
000111920
000111930
000111940
000111950

```

00011960
00011970
00011980
00011990
00012000
00012010
00012020
00012030
00012040
00012050
00012060
00012070
00012080
00012090
00012100
00012110
00012120
00012130
00012140
00012150
00012160
00012170
00012180
00012190
00012200
00012210
00012220
00012230
00012240
00012250
00012260
00012270
00012280
00012290
00012300
00012310
00012320
00012330
00012340
00012350
00012360
00012370
00012380
00012390
00012400
00012410
00012420
00012430

DERIVATIVES OF THE SHAPE FUNCTIONS WITH RESPECT TO XI.

```

DXI(1) = .25D0 * (1.D0-ETA) * (2.D0*XI+ETA)
DXI(2) = -XI * (1.D0-ETA)
DXI(3) = .25D0 * (1.D0-ETA) * (2.D0*XI-ETA)
DXI(4) = .5D0 * (1.D0-ETA**2)
DXI(5) = .25D0 * (1.D0+ETA) * (2.D0*XI+ETA)
DXI(6) = -XI * (1.D0+ETA)
DXI(7) = .25D0 * (1.D0+ETA) * (2.D0*XI-ETA)
DXI(8) = -.5D0 * (1.D0-ETA**2)

```

DERIVATIVES OF THE SHAPE FUNCTIONS WITH RESPECT TO ETA.

```

DETA(1) = .25D0 * (1.D0-XI) * (2.D0*ETA+XI)
DETA(2) = -.5D0 * (1.D0-XI**2)
DETA(3) = .25D0 * (1.D0+XI) * (2.D0*ETA-XI)
DETA(4) = -ETA * (1.D0+XI)
DETA(5) = .25D0 * (1.D0+XI) * (2.D0*ETA+XI)
DETA(6) = .5D0 * (1.D0-XI**2)
DETA(7) = .25D0 * (1.D0-XI) * (2.D0*ETA-XI)
DETA(8) = -ETA * (1.D0-XI)

```

CONSTRUCTION OF THE JACOBIAN MATRIX.

```

DO 5 I = 1, 2
  DO 5 J = 1, 2
    5 AJ(I,J) = 0.D0

```

```

DO 22 I = 1, NNE
  AJ(1,1) = AJ(1,1) + DXI(I) * Z(NN(L,I))
  AJ(1,2) = AJ(1,2) + DXI(I) * R(NN(L,I))
  AJ(2,1) = AJ(2,1) + DETA(I) * Z(NN(L,I))
  AJ(2,2) = AJ(2,2) + DETA(I) * R(NN(L,I))
22 CONTINUE

```

```

DET = AJ(1,1) * AJ(2,2) - AJ(1,2) * AJ(2,1)
DEI = 1.D0 / DET

```

CONSTRUCTION OF INVERSE OF JACOBIAN

```

C      DUM = AJ(1,1) * DEI
      AJ(1,1) = AJ(2,2) * DEI
      AJ(1,2) = -AJ(1,2) * DEI
      AJ(2,1) = -AJ(2,1) * DEI
      AJ(2,2) = DUM
C      CR = 0.0
      DO 21 I = 1, NNE
      ML = NN(L,I)
      21 CR = CR + SHP(I) * R(ML)
C      EVALUATION OF THE UPPER TRIANGLE OF CE.
C      DUM = FACTC * DET * CR * GC(N) * GC(M)
      DO 23 I = 1, NNE
      DO 23 J = I, NNE
      CE(I,J) = CE(I,J) + DUM * SHP(I) * SHP(J)
      23 CONTINUE
C      CONSTRUCTION OF DNZR : THE MATRIX OF THE PARTIAL
      DERIVATIVES OF SHAPE FUNCTIONS RESPECT TO Z AND R
      DO 30 I = 1, NNE
      DNZR(1,I) = AJ(1,1) * DXI(I) + AJ(1,2) * DETA(I)
      DNZR(2,I) = AJ(2,1) * DXI(I) + AJ(2,2) * DETA(I)
      30 CONTINUE
C      EVALUATION OF UPPER TRIANGLE OF YE.
C      DUM = FACTY * CR * DET * GC(N) * GC(M)
      DO 26 I = 1, NNE
      DO 26 J = I, NNE
      DUM1 = DNZR(1,I) * DNZR(1,J)
      DUM2 = DNZR(2,I) * DNZR(2,J)
      YE(I,J) = YE(I,J) + (DUM1+DUM2) * DUM
      26 CONTINUE
C      9 CONTINUE

```

```

C      DO 27 I = 1, NNE
      DO 27 J = 1, NNE
      CE(J,I) = CE(I,J)
      27 YE(J,I) = YE(I,J)
C
C      GENERATING THE THERMAL CAPACITANCE AND ADMITTANCE
C      MATRICES IN THE BANDED FORM.
C
      14 I = 1, NNE
      II = NN(L,I) - 1
      DO 14 J = 1, NNE
      JJ = NN(L,J) - 1
      IF = (II.GT.JJ) GO TO 14
      J1 = JJ - II
      JJ = JJ + I
      KK = JJ + 1
      C(JJ,KK) = C(JJ,KK) + CE(I,J)
      Y(JJ,KK) = Y(JJ,KK) + YE(I,J)
      14 CONTINUE
C
      25 CONTINUE
C
      IF (Q2.GT.0.D0) GO TO 40
      NC = 0
      GO TO 50
      40 NC = NCFDT
      DO 64 I = 1, NCFOT
      DIAG(I) = 0.D0
      OFF1(I) = 0.D0
      OFF2(I) = 0.D0
      64 CONTINUE
C
      NECFO = (NCFOT-1) / 2
      DO 60 M = 1, NECFO
      K = 2 * M - 1
      JL = NCFOT(K)
      JM = NCFOT(K+1)
      JR = NCFOT(K+2)
      DO 60 N = 1, NGP
      XI = GX(N)

```

```

C      SHP1      = -.5D0 * (1.D0-XI) * XI
C      SHP2      = 1.D0 - XI**2
C      SHP3      = .5D0 * (1.D0+XI) * XI
C      DXI1      = -.5D0 + XI
C      DXI2      = -2.D0 * XI
C      DXI3      = .5D0 + XI
C      RR = SHP3*R(JL) + SHP2*R(JM) + SHP1*R(JR)
C      DRXI = DXI3*R(JL) + DXI2*R(JM) + DXI1*R(JR)
C      DZXI = DXI3*Z(JL) + DXI2*Z(JM) + DXI1*Z(JR)
C      DRXI = DRXI**2
C      DZXI = DZXI**2
C      DUMY = RR * GC(N) * DSQRT(DRXI+DZXI)
C      DIAG( K ) = DIAG( K ) + SHP3*SHP3 * DUMY
C      DIAC(K+1) = DIAG(K+1) + SHP2*SHP2 * DUMY
C      DIAG(K+2) = DIAG(K+2) + SHP1*SHP1 * DUMY
C      OFF1( K ) = OFF1( K ) + SHP3*SHP2 * DUMY
C      OFF1(K+1) = OFF1(K+1) + SHP2*SHP1 * DUMY
C      OFF2( K ) = OFF2( K ) + SHP1*SHP3 * DUMY
C      60 CONTINUE
C      CONST = 2.D0 * PAI * HTCO
C      DO 67 I = 1, NCFOT
C      DIAG(I) = DIAG(I) * CONST
C      OFF1(I) = OFF1(I) * CONST
C      67 OFF2(I) = OFF2(I) * CONST
C      ADDING THE CONTRIBUTION OF Y STAR TO Y MATRIX
C      FCR OUTSIDE CONVECTION
C      DO 68 I = 1, NCFOT
C      K = NCFOT(I)
C      68 Y(K,1) = Y(K,1) + DIAG(I)
C      N1 = NCFOT - I
C      DO 69 I = 1, N1

```

00013880
00013890
00013900
00013910
00013920
00013930
00013940
00013950
00013960
00013970
00013980
00013990
00014000
00014010
00014020
00014030
00014040
00014050
00014060
00014070
00014080
00014090
00014100
00014110
00014120
00014130
00014140
00014150
00014160
00014170
00014180
00014190
00014200
00014210
00014220
00014230
00014240
00014250
00014260
00014270
00014280
00014290
00014300
00014310
00014320
00014330
00014340
00014350

K1 = NCF0(I)
K2 = NCF0(I+1)
K = K2 - K1 + 1
IF (K, LE, 0) K = - K + 2
K1 = MINO(K1, K2)
Y(K1, K) = Y(K1, K) + OFF1(I)
69 CONTINUE

C

N2 = NCFOT-2
DO 66 I = 1, N2, 2
K1 = NCF0(I)
K2 = NCF0(I+2)
K = K2 - K1 + 1
IF (K, LE, 0) K = - K + 2
K1 = MINO(K1, K2)
Y(K1, K) = Y(K1, K) + OFF2(I)
66 CONTINUE

C
C
C

50 IF (Q3, GT, 0.00) GO TO 41
GO TO 83
41 DO 54 I = 1, NCFIT
J = NC + I
DIAG(J) = 0.00
OFF1(J) = 0.00
54 OFF2(J) = 0.00

C
C

NECFI = (NCFIT - 1) / 2
DO 59 M = 1, NECFI

C
C
C

K = 2 * M - 1

JL = NCFI(K)
JM = NCFI(K+1)
JR = NCFI(K+2)

C

DO 59 N = 1, NGP

C

XI = GX(N)

C
C

SHP1 = -.500 * (1.00 - XI) * XI
SHP2 = 1.00 - XI * 2
SHP3 = .500 * (1.00 + XI) * XI
DX11 = -.500 + XI
DX12 = -2.00 * XI


```

      K = K2 - K1 + 1
      IF (K.LE.0) K = - K + 2
      K1 = MINO(K1,K2)
      Y(K1,K) = Y(K1,K) + OFF1(J)
55 CONTINUE
      M2 = NCFIT - 2
      DO 56 I = 1, M2, 2
      J = NC + I
      K1 = NCFI(I)
      K2 = NCFI(I+2)
      K = K2 - K1 + 1
      IF (K.LE.0) K = - K + 2
      K1 = MINO(K1,K2)
      Y(K1,K) = Y(K1,K) + OFF2(J)
56 CONTINUE
83 CONTINUE

```

EVALUATION OF A AND G MATRICES

```

DUM = 5000 * DTI
DO 1 I=1,NNT
DO 1 J = 1, NBAND
  EL = DUM * Y(I,J)
  C1 = C(I,J) + EL
  C2 = C(I,J) - EL
  C(I,J) = C1
  Y(I,J) = C2
1 CONTINUE

```

C NOW CONTAINS C+.5*DTI*Y AND Y CONTAINS C-.5*DTI*Y

THIS PART WILL DO THE CHOLESKY DECOMPOSITION OF THE MATRIX X<C> WHICH IS ALREADY PUT IN THE BANDED FORM. ORIGINAL VERSION WAS CODED BY PROF. CAN, IN

```

DO 300 I = 1, NNT
DUM = C(I,I)
DUM = DSQRT(DUM)

```

```

00014840
00014850
00014860
00014870
00014880
00014890
00014900
00014910
00014920
00014930
00014940
00014950
00014960
00014970
00014980
00014990
00015000
00015010
00015020
00015030
00015040
00015050
00015060
00015070
00015080
00015090
00015100
00015110
00015120
00015130
00015140
00015150
00015160
00015170
00015180
00015190
00015200
00015210
00015220
00015230
00015240
00015250
00015260
00015270
00015280
00015290
00015300
00015310

```

```

00015320
00015330
00015340
00015350
00015360
00015370
00015380
00015390
00015400
00015410
00015420
00015430
00015440
00015450
00015460
00015470
00015480

```

```

      DO 200 J = 1, NBAND
200  C(I,J) = C(I,J) / DUM
      C
      DO 260 J = 2, NBAND
      L = I + J - 1
      IF (L.GT.NNT) GO TO 260
      AA = C(I,J)
      IF (AA.EQ.0.0) GO TO 260
      DO 250 K = J, NBAND
      M = K - J + 1
      C(L,M) = C(L,M) - AA * C(I,K)
250  CONTINUE
260  CONTINUE
300  CONTINUE
      C
      C
      RETURN
      END

```

```
C      SUBROUTINE FLOW
C
C      *****
C      FOR A GIVEN OUTSIDE OR INSIDE ENTRY TIME VARIATION OF
C      FLUID TEMPERATURE, THE SUBROUTINE FLOW WILL EVALUATE
C      THE TEMPERATURE OF THE FLUID NODES AT A GIVEN TIME .
C      THE CALCULATION IS BASED UPON CONSTANT FLUX.
C      *****
C
C      IMPLICIT REAL * 8 (A-H,O-Z)
C      REAL * 4 ENTEO,ENTEI,ENTIO,ENTII
C
C      COMMON /ONE/ AJ(2,2),B(4,16),COPY(37,298),D(4,4),DL(4,4),
C      IDNZR(2,8),ESM(16,16),F(24,298),SSM(298,66),STOR(20,150)
C
C      COMMON /TWO/ AL(5),DENS(5),DETA(8),DXI(8),E(5),EPSO(4),
C      IFE(16),GC(4),GX(4),PCI(5),R(149),SHP(8),SHF(5),TK(5),Z(149)
C
C      COMMON /TRE/ AXIALF,DTI,EVERY,FORSI,OMEGA,HICI,HICQ,
C      IPOSI,PRES,Q1,Q2,Q3,SHMAX,SMAX,TIME,TIMEI,TINIT,TMAXM,TREF,
C      ZTMXOX,RMAXM,RMAXO,ZMAXM,ZMAXG,Q4,BIG
C
C      COMMON /FOR/ NN(40,9)
C
C      COMMON /FIV/ NCFO(37),NCFO(37),NNR(37),NRE(37),NNL(37),
C      INPN(37)
C      COMMON /SIX/ IBAND,IEXT,INTP,IVEC,JVEC,KK,LOAD,NCFIT,NCFOI,
C      INCHECK,NET,NDF,NGP,NNE,NNLT,NNRT,NPASS,NPROB,NRPPI,NRPPO,
C      3NNNT,NNRE,NPNT,IPLANE,NBAND
C
C      EQUIVALENCE (C,SSM),(Y,SSM(1,18)),(T,SSM(1,35)),
C      1(BTE,SSM(1,36)),(TEMP,SSM(1,37)),(V,SSM(1,38)),
C      2(FIT,SSM(1,39)),(FI2,SSM(1,40)),(CE,SSM(1,41)),
C      3(YE,SSM(1,42)),(OFF1,SSM(1,43)),(OFF2,SSM(1,44)),
C      4(DIAG,SSM(1,45)),(BDRY1,SSM(1,46)),(BDRYO,SSM(1,47)),
C      5(ENTII,SSM(1,48)),(ENTIO,SSM(1,49)),(ENTEI,SSM(1,50)),
C      6(ENTEO,SSM(1,51)),(VOL,SSM(1,52)),(VLO,SSM(1,53))
C
C      DIMENSION C(149,33),Y(149,33),T(149),BIE(149),TEMP(149),
C      IV(149),FT1(90),FI2(90),CE(8,8),YE(8,8),OFF1(37),OFF2(37),
```

```

2DIAG(37),BDRYI(16,5),BDRYO(16,5),VOL(36),VOLO(36),ENTEO(75),
3ENTEI(75),ENTIO(75),ENTII(75)

```

```

C1 = 81157810217736320
C2 = 1.6755160819145570
C3 = 0.26179938779914950
C4 = 1.2042771838760882
C5 = -2.094395102393196
C6 = -3.665191429188094

```

```

PAI = 3.141592653589793

```

```

FLOW IN THE LONGITUDINAL DIRECTION , AS IN PIPE

```

```

N = 0
IF (Q2.EQ.0.D0) GO TO 46
N = NCFOT
IF (KK.EQ.1) GO TO 40
IF (NCFOT.EQ.0) GO TO 45

```

```

DO 41 I = 1, NCFOT
41 FT1(I) = FT2(I)

```

```

40 TIME = KK * DTI

```

```

M = KK / 2 * M
I = KK - 2 * M
IF ((KK.LE.150).AND.(I.EQ.1)) ENTEO(M) = FT1(1)
IF ((KK.LE.150).AND.(I.EQ.1)) ENTIO(M) = TIME-DTI

```

```

DO 42 I = 2, NRPP0
42 IF (TIME.GT.BDRYO(I,1)) VEL = BDRYO(I,5)

```

```

IF (VEL.EQ.0.D0) GO TO 46

```

```

IF (KK.GT.1) GO TO 75
NECFO = (NCFOT-1) / 2

```

```

DUM = 0.D0
DO 10 I = 2, NCFOT

```

```

00015950
00015960
00015970
00015980
00015990
00016000
00016010
00016020
00016030
00016040
00016050
00016060
00016070
00016080
00016090
00016100
00016110
00016120
00016130
00016140
00016150
00016160
00016170
00016180
00016190
00016200
00016210
00016220
00016230
00016240
00016250
00016260
00016270
00016280
00016290
00016300
00016310
00016320
00016330
00016340
00016350
00016360
00016370
00016380
00016390
00016400
00016410
00016420

```

```

R1 = R(NCFO(I-1))
R2 = R(NCFO(I))
IF (R1.GE.R2) DUM = R1
10 CONTINUE
IF (DUM.EQ.O.DO) DUM = R(NCFO(NCFOT))
C
C
DUM = DUM + 1.DO
AREA0 = PAI * DUM**2
VOLO(1) = O.DO
DO 70 I = 1, NECFO
K = 2 * I - 1
J1 = NCFO(K)
J2 = NCFO(K+1)
J3 = NCFO(K+2)
DUM = C1*R(J1)**2+C2*R(J2)**2+C3*R(J3)**2
DUM=DUM+C4*R(J1)*R(J2)+C5*R(J1)*R(J3)+C6*R(J2)*R(J3)
VOLO(K+1) = (AREA0-DUM)*DABS(Z(J2)-Z(J1))
C
C
DUM = C3*R(J1)**2+C2*R(J2)**2+C1*R(J3)**2
DUM=DUM+C6*R(J1)*R(J2)+C5*R(J1)*R(J3)+C4*R(J2)*R(J3)
VOLO(K+2) = (AREA0-DUM)*DABS(Z(J3)-Z(J2))
70 CONTINUE
C
C
DO 73 I = 2, NRPP0
IF (TIME.GT.BDRYO(I,1)) KK2 = I
73 CONTINUE
TIM = TIME
TF = BDRYO(KK2,4) * (TIM-BDRYO(KK2,1))
FT2(1) = BDRYO(KK2,2) + TF
C
C
DUMY = VEL*(AREA0-PAI*R(NCFO(1))**2)
C
C
DO 74 I = 2, NCFQT
TIM1 = VOLO(I) / DUMY
TIM = TIM - TIM1
DO 76 J = 2, NRPP0
IF (TIM.GE.BDRYO(J,1)) KK2 = J
76 CONTINUE
C
C
TF = BDRYO(KK2,4) * (TIM-BDRYO(KK2,1))

```

```

FT2(I) = BDRYO(KK,2) + TF
74 CONTINUE
GO TO 46
45 TIME = KK * DTI
C
46 IF (Q3.EQ.0.D0) GO TO 56
C
      IF (KK.EQ.1)
      IF (NCHECK.EQ.0) GO TO 55
C
      DO 51 I = 1, NCFIT
      J = N + I
      51 FT1(J) = FT2(J)
C
      50 TIME = KK * DTI
C
      M = KK / 2 * M
      I = KK - 1
      M = M + 1
      IF ((KK.LE.150).AND.(I.EQ.1)) ENTEI(M) = FT1(N+1)
      IF ((KK.LE.150).AND.(I.EQ.1)) ENTEII(M) = TIME-DTI
C
      DO 52 I = 2, NRPPI
      52 IF (TIME.GT.BDRYI(I,1)) VEL = BDRYI(I,5)
C
      IF (VEL.EQ.0.D0) GO TO 56
C
      IF (KK.GT.1) GO TO 82
      NECFI = (NCFIT-1)/2
      AREA = PAI * R(NCFI(1))**2
      VOL(1) = 0.D0
C
      DO 81 I = 1, NECFI
      K = 2 * I - 1
      J1 = NCFI(K)
      J2 = NCFI(K+1)
      J3 = NCFI(K+2)
C
      DUM = C1*R(J1)**2+C2*R(J2)**2+C3*R(J3)**2
      DUM=DUM+C4*R(J1)*R(J2)+C5*R(J1)*R(J3)+C6*R(J2)*R(J3)
      VOL(K+1) = DUM * DABS(Z(J2)-Z(J1))
C
      DUM = C3*R(J1)**2+C2*R(J2)**2+C1*R(J3)**2
C

```

```

00016910
00016920
00016930
00016940
00016950
00016960
00016970
00016980
00016990
00017000
00017010
00017020
00017030
00017040
00017050
00017060
00017070
00017080
00017090
00017100
00017110
00017120
00017130
00017140
00017150
00017160
00017170
00017180
00017190
00017200
00017210
00017220
00017230
00017240
00017250
00017260
00017270
00017280
00017290
00017300
00017310
00017320
00017330
00017340
00017350
00017360
00017370
00017380

```

```

      DUM=DUM+C6*R(J1)*R(J2)+C5*R(J1)*R(J3)+C4*R(J2)*R(J3)
      VOL(K+2) = DUM * DABS(Z(J3) - Z(J2))
81 CONTINUE
C
C
C
C
      DO 83 I = 2 , NRPPI
      IF (TIME.GT .BDRYI(I,1)) KK2 = I
83 CONTINUE
C
      TIME = TIME
      TF = BDRYI(KK2,4) * (TIM-BDRYI(KK2,1))
      FT2(N+1) = BDRYI(KK2,2) + TF
C
      DUMY = VEL * AREA
C
C
      DO 84 I = 2 , NCFIT
      TIM1 = VOL(I) / DUMY
      TIM = TIM - TIM1
      DO 85 J = 2 , NRPPI
      IF (TIM.GE.BDRYI(J,1)) KK2 = J
85 CONTINUE
      TF = BDRYI(KK2,4) * (TIM-BDRYI(KK2,1))
      FT2(N+1) = BDRYI(KK2,2) + TF
84 GO TO 56
C
      TIME = KK * DTI
56 RETURN
      END

```

```

00017390
00017400
00017410
00017420
00017430
00017440
00017450
00017460
00017470
00017480
00017490
00017500
00017510
00017520
00017530
00017540
00017550
00017560
00017570
00017580
00017590
00017600
00017610
00017620
00017630
00017640
00017650
00017660
00017670
00017680
00017690
00017700
00017710
00017720
00017730
00017740

```


000182200
000182230
000182240
000182250
000182260
000182270
000182280
000182290
000182300
000182310
000182320
000182330
000182340
000182350
000182360
000182370
000182380
000182390
000182400
000182410
000182420
000182430
000182440
000182450
000182460
000182470
000182480
000182490
000182500
000182510
000182520
000182530
000182540
000182550
000182560
000182570
000182580
000182590
000182600
000182610
000182620
000182630
000182640
000182650
000182660
000182670
000182680

```

C
N=0
IF (Q2.GT.0.00) GO TO 5
GO TO 6
5 N= NCFOT
DO I=1, NCFOT
J= NCFOT(I)
1 V(J) = 0.00
DO 2 J=1, NCFOT
I= NCFOT(J)
2 V(I) = V(I) + DIAG(J) * (FT1(J)+FT2(J))
C
N1= NCFOT - 1
DO 3 I=1, N1
I1= I + 1
K1= NCFOT(I)
K2= NCFOT(I+1)
3 V(K1) = V(K1) + OFF1(I) * (FT1(I1)+FT2(I1))
V(K2) = V(K2) + OFF1(I) * (FT1(I1)+FT2(I1))
C
N2= NCFOT - 2
DO 4 I=1, N2, 2
I2= I + 2
K2= NCFOT(I+2)
J= NCFOT(I)
4 V(J) = V(J) + OFF2(I) * (FT1(I2)+FT2(I2))
V(K2) = V(K2) + OFF2(I) * (FT1(I2)+FT2(I2))
C
6 IF (Q3.GT.0.00) GO TO 11
GO TO 12
11 DO 7 I=1, NCFIT
J= NCFIT(I)
7 V(J) = 0.00
C
DO 8 J=1, NCFIT
I= NCFIT(J)
K= N + J
8 V(I) = V(I) + DIAG(K) * (FT1(K)+FT2(K))
C
N1= NCFIT - 1
DO 9 I=1, N1
K= N + I
I1= K + 1
K1= NCFIT(I)
K2= NCFIT(I+1)

```

00018690
00018700
00018710
00018720
00018730
00018740
00018750
00018760
00018770
00018780
00018790
00018800
00018810
00018820
00018830

```

C
  V(K1) = V(K1) + OFF1(K) * (FT1(I1)+FT2(I1))
  V(K2) = V(K2) + OFF1(K) * (FT1(K)+FT2(K))
  N2 = NCFIT - 2
  DO 10 I = 1, N2, 2
    K = N + I
    I2 = K + 2
    J = NCFI(I, I)
    K2 = NCFI(I+2, I)
    V(J) = V(J) + OFF2(K) * (FT1(I2)+FT2(I2))
    V(K2) = V(K2) + OFF2(K) * (FT1(K)+FT2(K))
  10
  12 CONTINUE
  RETURN
  END
C

```

```

SUBROUTINE TEMPER
*****
IN THIS SUBROUTINE THE NODAL TEMPERATURES ARE
EVALUATED BY TRAPEZOIDAL INTEGRATION. ALSO FOR
EVERY 10 STEPS OF TIME INTEGRATION THE IRONS
CORRECTION IS APPLIED.
*****

IMPLICIT REAL * 8 (A-H,O-Z)

COMMON /ONE/ AJ(2,2),B(4,16),COPY(37,298),D(4,4),DL(4,4),
1DNZR(2,8),ESM(16,16),F(24,298),SSM(298,66),STOR(20,150)

COMMON /TWO/ AL(5),DENS(5),DEIA(8),DXI(8),E(5),EPSO(4),
1FE(16),GC(4),GX(4),POI(5),R(149),SHP(8),SHI(5),TK(5),Z(149)

COMMON /TRE/ AXIALF,DTI,EVERY,FORS1,OMEGA,HICI,HTCO,
1POSI,PRES,Q1,Q2,Q3,SHMAX,SMAX,TIME,tIMEI,tINIT,tMAXM,TREF,
2TMAXO,RMAXM,RMAXO,ZMAXM,ZMAXO,Q4,BIG

COMMON /FOR/ NN(40,9)

COMMON /FIV/ NCFI(37),NCFO(37),NNR(37),NRE(37),NNL(37),
1NPN(37)
COMMON /SIX/ IBAND,IEXT,INTP,IVEC,JVEC,KK,LOAD,NCFIT,NCFOI,
1NCHECK,NET,NDF,NGP,NNE,NNLT,NNRT,NPASS,NPROB,NRPPI,NRPP0,
3NNNT,NNRE,NPNT,IPLANE,NBAND

EQUIVALENCE (C,SSM), (Y,SSM(1,18)), (T,SSM(1,35)),
1(BTE,SSM(1,36)), (TEMP,SSM(1,37)), (V,SSM(1,38)),
2(FI1,SSM(1,39)), (FI2,SSM(1,40)), (CE,SSM(1,41)),
3(YE,SSM(1,42)), (OFF1,SSM(1,43)), (OFF2,SSM(1,44)),
4(DIAG,SSM(1,45)), (BDRYI,SSM(1,46)), (BDRYO,SSM(1,47)),
5(ENTEI,SSM(1,48)), (ENTIO,SSM(1,49)), (ENTEI,SSM(1,50)),
6(ENTEO,SSM(1,51)), (VOL,SSM(1,52)), (VOLD,SSM(1,53))

DIMENSION C(149,33),Y(149,33),T(149),BTE(149),TEMP(149),
1V(149),FI1(90),FI2(90),CE(8,8),YE(8,8),OFF2(37),
2DIAG(37),BDRYI(16,5),BDRYO(16,5),VOL(36),VOLD(36),ENTEO(75),
3BENTEI(75),ENTIO(75),ENTEI(75)

```

```

CCCCCCCCCCCC C
C
C
C
C
C
C
C
C
C

```

```

CC      NCHECK = 1
C      DUM = DTI / 2. DO
C      DO 1 I=1,NNT
C      1 V(I) = DUM * V(I)
C
C      THIS PART WILL MULTIPLY THE BANDED MATRIX Y BY THE
C      COLUMN VECTOR OF TEMPERATURE AND ADD TO V.
C
C      DO 7 J = 1, NNT
C      BTE(J) = 0. DO
C      M = MINO(NBAND, NNT+1-J)
C      DO 5 K = 1, M
C      L = J + K - 1
C      BTE(J) = BTE(J) + Y(J,K) * TEMP(L)
C      5 CONTINUE
C
C      IF (J.LT.2) GO TO 7
C      L = MINO(J, NBAND)
C      DO 6 K = 2, L
C      M = J - K + 1
C      BTE(J) = BTE(J) + Y(M,K) * TEMP(M)
C      6 CONTINUE
C      BTE(J) = BTE(J) + V(J)
C
C      SOLVING FOR TEMPERATURE BY FORWARD AND BACK SUBSTITUTION
C
C      DO 32 I = 1, NNT
C      BTE(I) = BTE(I) / C(I,1)
C      DO 31 J = 2, NBAND
C      L = I + J - 1
C      IF (L.GT.NNT) GO TO 32
C      DUM = C(I,J)
C      BTE(L) = BTE(L) - DUM * BTE(I)
C      31 CONTINUE
C      32 CONTINUE
C
C      BTE(NNT) = BTE(NNT) / C(NNT,1)
C      DO 34 L = 2, NNT
C      K = NNT - L + 1
C      SUM = 0. DO
C      DO 33 J = 2, NBAND

```

```

00019300
00019310
00019320
00019330
00019340
00019350
00019360
00019370
00019380
00019390
00019400
00019410
00019420
00019430
00019440
00019450
00019460
00019470
00019480
00019490
00019500
00019510
00019520
00019530
00019540
00019550
00019560
00019570
00019580
00019590
00019600
00019610
00019620
00019630
00019640
00019650
00019660
00019670
00019680
00019690
00019700
00019710
00019720
00019730
00019740
00019750
00019760
00019770

```

```

C      M = J + K - 1
C      IF (NNT, LT, M) GO TO 34
C      SUM = SUM + C(K, J) * BTE(I)
C      BTE(K) = (BTE(K) - SUM) / C(K, 1)
C
C      IF (KK.GT.2) GO TO 2
C      KO = 0
C
C      APPLICATION OF IRONS CORRECTION.
C
C      2 K1 = KO + 12 - KK
C      GO TO (70, 60, 50), K1
C      GO TO 14
C
C      50 DO 51 I = 1, NNT
C      51 T(I) = .25D0 * BTE(I)
C      GO TO 14
C
C      60 DO 61 I = 1, NNT
C      61 T(I) = T(I) + .5D0 * BTE(I)
C      GO TO 14
C
C      70 DO 71 I = 1, NNT
C      71 TEMP(I) = T(I) + .25D0 * BTE(I)
C      TEMPERATURE IS CORRECTED NOW
C
C      NCHECK = 0
C      KO = KO + 10
C      KK = KK - 1
C      TIME = KK*DTI
C      IP = 0
C
C      GO TO 16
C      14 DO 14 I = 1, NNT
C      14 TEMP(I) = BTE(I)
C      IP = 1
C
C      16 CONTINUE
C
C      C1 = 1000.D0 * TIME
C      DO 21 J = 1, IVEC
C      C2 = (TIME1 + EVERY * (J-1)) * 1000.D0
C      DC=DABS(C1-C2)
C      IF (DC.LE..0001) GO TO 20
C      21 CONTINUE

```

00020260
00020270
00020280
00020290
00020300
00020310
00020320
00020330
00020340
00020350
00020360
00020370
00020380
00020390
00020400
00020410
00020420
00020430
00020440
00020450
00020460
00020470
00020480
00020490
00020500
00020510
00020520
00020530
00020540
00020550
00020560
00020570

```

C      GO TO 22
C      DO 40 I = 1, NNT
C      40 STOR(J,I) = TEMP(I)
C      STOR(J,NNT+1) = TIME
C      22 IF (INTP.EQ.0) GO TO 81
C      INT1 = INTP
C      83 KUNT = KK / INT1 GO TO 81
C      KUNT = KK - KUNT * INT1
C      IF (KUNT.EQ.0) GO TO 82
C      INT1 = INT1 + INTP
C      GO TO 83
C      82 ID = INTP / 10 * 10
C      ID = INTP - ID
C      IF ((ID.EQ.0).AND.(IP.NE.0)) GO TO 81
C      101 WRITE (6,101) AT TIME=,F8.2, THE TEMPERATURES AT THE ,
C      1,NODES ARE:,,/
C      222 WRITE (6,222) (I,TEMP(I),I=1,NNT)
C      81 FORMAT (7(1X, '(,I3,')',1PD13.5))
C      KK = KK + 1
C      RETURN
C      END
C

```



```

C
      EL = POI(K) / EL
      D(1,1) = FACT
      D(1,2) = EL* FACT
      D(1,3) = D(1,2)
      D(1,4) = 0.D0
      D(2,2) = FACT
      D(2,3) = D(1,2)
      D(2,4) = 0.D0
      D(3,3) = FACT
      D(3,4) = 0.D0
      D(4,4) = E(K)/(2.D0*(1.D0+POI(K)))
CC
      DO 27 I = 1, 4
      DO 27 J = 1, 4
      27 D(J,I) = D(I,J)
CC
      CALL LLT ('D,DL)
CC
      DL IS THE LOWER TRIANGLE OF DECOMPOSED D.
CC
      3 CONTINUE
      NNE2 = 2 * NNE
      DO 11 I = 1, NNE2
      DO 11 J = 1, NNE2
      11 ESM(I,J) = 0.D0
CC
      DO 9 N = 1,NGP
      ETA = GX(N)
      DO 8 M = 1,NGP
      XI = GX(M)
CC
      DO 7 I = 1,4
      DO 7 J = 1,NNE2
      7 B(I,J) = 0.D0
CC
      SHP(1) = .25D0 * (1.D0-XI) * (1.D0-ETA) * (-XI-ETA-1.D0)
      SHP(2) = .5D0 * (1.D0-XI**2) * (1.D0-ETA)
      SHP(3) = .25D0 * (1.D0+XI) * (1.D0-ETA) * (XI-ETA-1.D0)

```

```

00021040
00021050
00021060
00021070
00021080
00021090
00021100
00021110
00021120
00021130
00021140
00021150
00021160
00021170
00021180
00021190
00021200
00021210
00021220
00021230
00021240
00021250
00021260
00021270
00021280
00021290
00021300
00021310
00021320
00021330
00021340
00021350
00021360
00021370
00021380
00021390
00021400
00021410
00021420
00021430
00021440
00021450
00021460
00021470
00021480
00021490
00021500
00021510

```

```

SHP(4) = .5D0 * (1.D0-ETA**2) * (1.D0+XI) (XI+ETA-1.D0)
SHP(5) = .25D0 * (1.D0+XI) * (1.D0+ETA) *
SHP(6) = .5D0 * (1.D0-XI**2) * (1.D0+ETA) (-XI+ETA-1.D0)
SHP(7) = .25D0 * (1.D0-XI) * (1.D0+ETA) *
SHP(8) = .5D0 * (1.D0-ETA**2) * (1.D0-XI)

```

DERIVATIVES OF THE SHAPE FUNCTIONS WITH RESPECT TO XI.

```

DXI(1) = .25D0 * (1.D0-ETA) * (2.D0*XI+ETA)
DXI(2) = -XI * (1.D0-ETA)
DXI(3) = .25D0 * (1.D0-ETA) * (2.D0*XI-ETA)
DXI(4) = .5D0 * (1.D0-ETA**2)
DXI(5) = .25D0 * (1.D0+ETA) * (2.D0*XI+ETA)
DXI(6) = -XI * (1.D0+ETA)
DXI(7) = .25D0 * (1.D0+ETA) * (2.D0*XI-ETA)
DXI(8) = -.5D0 * (1.D0-ETA**2)

```

DERIVATIVES OF THE SHAPE FUNCTIONS WITH RESPECT TO ETA.

```

DETA(1) = .25D0 * (1.D0-XI) * (2.D0*ETA+XI)
DETA(2) = -.5D0 * (1.D0-XI**2)
DETA(3) = .25D0 * (1.D0+XI) * (2.D0*ETA-XI)
DETA(4) = -ETA * (1.D0+XI)
DETA(5) = .25D0 * (1.D0+XI) * (2.D0*ETA+XI)
DETA(6) = .5D0 * (1.D0-XI**2)
DETA(7) = .25D0 * (1.D0-XI) * (2.D0*ETA-XI)
DETA(8) = -ETA * (1.D0-XI)

```

CONSTRUCTION OF THE JACOBIAN MATRIX.

```

DO 20 I = 1, 2
DO 20 J = 1, 2
20 AJ(I,J) = 0.D0

```

```

DO 22 I = 1, NNE
AJ(1,1) = AJ(1,1) + Z(NN(L,I))
AJ(1,2) = AJ(1,2) + DXI(1)
AJ(2,1) = AJ(2,1) + DETA(1)
AJ(2,2) = AJ(2,2) + DETA(1)
22 CONTINUE

```

```

00021520
00021530
00021540
00021550
00021560
00021570
00021580
00021590
00021600
00021610
00021620
00021630
00021640
00021650
00021660
00021670
00021680
00021690
00021700
00021710
00021720
00021730
00021740
00021750
00021760
00021770
00021780
00021790
00021800
00021810
00021820
00021830
00021840
00021850
00021860
00021870
00021880
00021890
00021900
00021910
00021920
00021930
00021940
00021950
00021960
00021970
00021980
00021990

```

```

C      DET =AJ(1,1) * AJ(2,2) - AJ(1,2) * AJ(2,1)
C      DEI =1.000/DET
C      INVERTING THE JACOBIAN AND STORING IN AJ AGAIN.
C
C      DUM = AJ(1,1) * DEI
C      AJ(1,1) = AJ(2,2) * DEI
C      AJ(1,2) = -AJ(1,2) * DEI
C      AJ(2,1) = -AJ(2,1) * DEI
C      AJ(2,2) = DUM
C
C      CR = 0.00
C      DO 2 I = 1, NNE
C      M1 = NN(L, I)
C      CR = CR + SHP(I) * R(M1)
C      2
C      EVALUATION OF THE B MATRIX
C
C      DO 15 I = 1, NNE
C      K1 = 2 * I - 1
C      K2 = K1 + 1
C      B(1, K2) = AJ(1,1) * DXI(I) + AJ(1,2) * DETA(I)
C      B(2, K1) = AJ(2,1) * DXI(I) + AJ(2,2) * DETA(I)
C      B(3, K1) = SHP(I) / CR
C      B(4, K1) = B(1, K2)
C      B(4, K2) = B(2, K1)
C      15 CONTINUE
C
C      CALCULATION OF (DL TRANSPOSE)*B AND STORING IN <B>
C
C      DO 16 I = 1, 4
C      DO 13 J = 1, NNE2
C      DUM = 0.000
C      DO 17 K = I, 4
C      DUM = DUM + DL(K, I) * B(K, J)
C      17 B(I, J) = DUM
C      13 CONTINUE
C      16
C

```



```

C 31 CONTINUE
C IF (IPLANE.EQ.0) GO TO 49
C
C PART OF SSM IS BEING STORED IN COPY.
C
C DO 41 J = 1, NDF
C DO 41 I = 1, NNRE
C 41 COPY(I,J) = 0.DO
C
C DO 44 II = 1, NNRE
C K = 2 * NRE(II)
C JJ = K
C DO 42 J = 1, IBAND
C IF (JJ.GT.NDF) GO TO 45
C COPY(II,JJ) = SS(K,J)
C 42 JJ = JJ + 1
C
C 45 LL = K - 1
C IF (LL.EQ.0) GO TO 44
C JJ = LL
C
C DO 43 J = 1, LL
C IF (J.EQ.IBAND) GO TO 44
C COPY(II,JJ) = SSM(JJ,J+1)
C 43 JJ = JJ - 1
C 44 CONTINUE
C
C 49 CONTINUE
C IF (NNRT.EQ.0) GO TO 33
C DO 34 I = 1, NNRT
C K = 2 * NNR(I) - 1
C 34 SSM(K,1) = SSM(K,1) * BIG
C
C 33 IF (NNLT.EQ.0) GO TO 35
C DO 36 I = 1, NNLT
C K = 2 * NNLT(I)
C 36 SSM(K,1) = SSM(K,1) * BIG
C 35 CONTINUE

```

```

00022960
00022970
00022980
00022990
00023000
00023010
00023020
00023030
00023040
00023050
00023060
00023070
00023080
00023090
00023100
00023110
00023120
00023130
00023140
00023150
00023160
00023170
00023180
00023190
00023200
00023210
00023220
00023230
00023240
00023250
00023260
00023270
00023280
00023290
00023300
00023310
00023320
00023330
00023340
00023350
00023360
00023370
00023380
00023390
00023400
00023410
00023420
00023430

```

```

CCCCC
      DECOMPOSING SYSTEM STIFFNESS MATRIX
      DO 300 I = 1 , NDF
      DUM = SSM(I,1)
      DUM = DSQRT(DUM)
      DO 200 J = 1 , IBAND
      200 SSM(I,J) = SSM(I,J) / DUM
      C
      DO 260 J = 2 , IBAND
      L = I + J - 1
      IF (L.GT.NDF) GO TO 260
      AA = SSM(I,J)
      IF (AA.EQ.0.DO) GO TO 260
      DO 250 K = J , IBAND
      M = K - J + 1
      250 SSM(L,M) = SSM(L,M) - AA * SSM(I,K)
      260 CONTINUE
      300 CONTINUE
      350 RETURN
      END
00023440
00023450
00023460
00023470
00023480
00023490
00023500
00023510
00023520
00023530
00023540
00023550
00023560
00023570
00023580
00023590
00023600
00023610
00023620
00023630
00023640
00023650
00023660

```



```

00024130
00024140
00024150
00024160
00024170
00024180
00024190
00024200
00024210
00024220
00024230
00024240
00024250
00024260
00024270
00024280
00024290
00024300
00024310
00024320
00024330
00024340
00024350
00024360
00024370
00024380
00024390
00024400
00024410
00024420
00024430
00024440
00024450
00024460
00024470
00024480
00024490
00024500
00024510
00024520
00024530
00024540
00024550
00024560
00024570
00024580
00024590
00024600

```

```

DO 25 L = 1, NET

```

```

DO 9 N = 1, NGP
ETA = GX(N)
DO 8 M = 1, NGP
XI = GX(M)

```

```

DO 7 J = 1, NNE2
DO 7 I, J = 0, DO
7 B(I, J) = 0. DO

```

```

SHP(1) = .25DO * (1. DO - XI) * (1. DO - ETA) * (-XI - ETA - 1. DO)
SHP(2) = .5DO * (1. DO - XI) * (1. DO - ETA) * (XI - ETA - 1. DO)
SHP(3) = .25DO * (1. DO - XI) * (1. DO - ETA) * (XI + ETA - 1. DO)
SHP(4) = .5DO * (1. DO - XI) * (1. DO - ETA) * (-XI + ETA - 1. DO)
SHP(5) = .25DO * (1. DO - XI) * (1. DO - ETA) * (1. DO - XI)
SHP(6) = .5DO * (1. DO - XI) * (1. DO - ETA) * (1. DO - XI)
SHP(7) = .25DO * (1. DO - XI) * (1. DO - ETA) * (1. DO - XI)
SHP(8) = .5DO * (1. DO - XI) * (1. DO - ETA) * (1. DO - XI)

```

DERIVATIVES OF THE SHAPE FUNCTIONS WITH RESPECT TO XI.

```

DXI(1) = .25DO * (1. DO - ETA) * (2. DO * XI + ETA)
DXI(2) = -XI * (1. DO - ETA)
DXI(3) = .25DO * (1. DO - ETA) * (2. DO * XI - ETA)
DXI(4) = .5DO * (1. DO - ETA) * 2
DXI(5) = .25DO * (1. DO - ETA) * (2. DO * XI + ETA)
DXI(6) = -XI * (1. DO - ETA)
DXI(7) = .25DO * (1. DO - ETA) * (2. DO * XI - ETA)
DXI(8) = -.5DO * (1. DO - ETA) * 2

```

DERIVATIVES OF THE SHAPE FUNCTIONS WITH RESPECT TO ETA.

```

DETA(1) = .25DO * (1. DO - XI) * (2. DO * ETA + XI)
DETA(2) = -.5DO * (1. DO - XI) * 2
DETA(3) = .25DO * (1. DO - XI) * (2. DO * ETA - XI)
DETA(4) = -ETA * (1. DO - XI)
DETA(5) = .25DO * (1. DO - XI) * (2. DO * ETA + XI)

```

```

CCCCC      DETA(6)      = .5D0 * (1.D0-XI**2)
            DETA(7)      = .25D0 * (1.D0-XI) * (2.D0*ETA-XI)
            DETA(8)      = -ETA * (1.D0-XI)

            CONSTRUCTION OF THE JACOBIAN MATRIX.

DO 20 I = 1, 2
DO 20 J = 1, 2
20 AJ(I,J) = 0.D0

DO 22 I = 1, NNE
AJ(1,1) = AJ(1,1) + DXI(I) * Z(NN(L,I))
AJ(1,2) = AJ(1,2) + DXI(I) * R(NN(L,I))
AJ(2,1) = AJ(2,1) + DETA(I) * Z(NN(L,I))
AJ(2,2) = AJ(2,2) + DETA(I) * R(NN(L,I))
22 CONTINUE

DET = AJ(1,1) * AJ(2,2) - AJ(1,2) * AJ(2,1)
DEI = 1.D0/DET

INVERTING THE JACOBIAN AND STORING IN AJ AGAIN.

DUM = AJ(1,1) * DEI
AJ(1,1) = AJ(2,2) * DEI
AJ(1,2) = -AJ(1,2) * DEI
AJ(2,1) = -AJ(2,1) * DEI
AJ(2,2) = DUM

CR = 0.D0
DO 2 I = 1, NNE
MI = NN(L,I)
2 CR = CR + SHP(I) * R(MI)

EVALUATION OF THE B MATRIX

DO 15 I = 1, NNE
K1 = 2 * I - 1
K2 = K1 + 1
B(1,K2) = AJ(1,1) * DXI(I) + AJ(1,2) * DETA(I)

```

```

00025090
00025100
00025110
00025120
00025130
00025140
00025150
00025160
00025170
00025180
00025190
00025200
00025210
00025220
00025230
00025240
00025250
00025260
00025270
00025280
00025290
00025300
00025310
00025320
00025330
00025340
00025350
00025360
00025370
00025380
00025390
00025400
00025410
00025420
00025430
00025440
00025450
00025460
00025470
00025480
00025490
00025500
00025510
00025520
00025530
00025540
00025550
00025560

```

```

B(2,K1) = AJ(2,1) * DXI(I) + AJ(2,2) * DETA(I)
B(3,K1) = SHP(I) / CR
B(4,K1) = B(1,K2)
B(4,K2) = B(2,K1)
15 CONTINUE

```

```

DUM1 = 2.00 * PAI * CR * DET * GC(N) * GC(M)

```

```

DO 30 KK = 1, IVEC

```

```

EVALUATING TEMPERATURE AT GAUSS POINT

```

```

TGP = 0.00
DO 3 I = 1, NNE
  MI = NN(L, I)
  3 TGP = TGP + SHP(I) * (STOR(KK,M1)-TREF)
  TGP = AL(NN(L,9)) * TGP

```

```

CALCULATION OF ELEMENT INITIAL STRAIN VECTOR <EPSO>

```

```

DO 6 I = 1, 3
  6 EPSO(I) = TGP

```

```

NOTE EPSO(4) = 0.00

```

```

PRODUCT OF <BTRANSPOSE> * <D> * <EPSO>
CALCULATION OF ELEMENT LOAD VECTOR ,

```

```

DUM = (TGP * E(NN(L,9))) / (1.00 - 2.00 * POI(NN(L,9)))

```

```

DUM = DUM * DUM1
DO 17 I = 1, NNE2, 2
  J = I + 1
  FE(I) = (B(2,I) + B(3,I)) * DUM
  17 FE(J) = B(1,J) * DUM

```

```

CONSTRUCTION OF SYSTEM LOAD VECTORS F

```

```

DO 18 I = 1, NNE

```

```

CC CC CCCC C CCCCCC CCCCCCCC C CCCCC

```

```

00025570
00025580
00025590
00025600
00025610
00025620
00025630
00025640
00025650
00025660
00025670
00025680
00025690
00025700
00025710
00025720
00025730
00025740
00025750
00025760
00025770
00025780
00025790
00025800
00025810
00025820
00025830
00025840

```

```

I2 = 2 * I - 1
I1 = NN(L,I) * 2 - 1
F(KK,I1) = F(KK,I1) + FE(I2)
F(KK,I1+1) = F(KK,I1+1) + FE(I2+1)

```

C

```

18 CONTINUE
30 CONTINUE
8 CONTINUE
9 CONTINUE
25 CONTINUE

```

C

```

IF (IPLANE.EQ.0) GO TO 40
LOAD VECTOR FOR UNIT END DISPLACEMENT

```

C

C

```

K = IDUM

```

C

```

DO 41 I = 1, NNRE
L = 2 * NRE(I)
F(K,L) = CCOPY(I,L) * BIG

```

41

```

40 CONTINUE
RETURN
END

```

C

C

```

SUBROUTINE CENTF
**
**
**
**
SUBROUTINE CENTF CALCULATES THE CENTRIFUGAL LOAD
VECTOR FOR A GIVEN NUMBER OF REVOLUTIONS PER MINUTE
**
**
**
**
IMPLICIT REAL * 8 (A-H,O-Z)
COMMON /CNE/ AJ(2,2),B(4,16),COPY(37,298),D(4,4),DL(4,4),
1DNZR(2,8),ESM(16,16),F(24,298),SSM(298,66),STOR(20,150)
COMMON /TWO/ AL(5),DENS(5),DETA(8),DXI(8),E(5),EPSO(4),
1FE(16),GC(4),GX(4),PCI(5),R(149),SHP(8),SHI(5),TK(5),Z(149)
COMMON /TRE/ AXIALF,DTI,EVERY,FORS1,OMEGA,HTCI,HTCO,
1POSI,PRES,Q1,Q2,Q3,SHMAX,SHAX,TIME,TIMEL,TINIT,TMAXM,TREF,
2TMAXO,RMAXM,RMAXO,ZMAXM,ZMAXO,Q4,BIG
COMMON /FOR/ NN(50,9)
COMMON /FIV/ NCFI(37),NCFO(37),NNR(37),NRE(37),NNL(37),
1NPNT(37),
COMMON /SIX/ IBAND,IEXT,INIP,IVEC,JVEC,KK,LOAD,NCFIT,NCFOT,
1NCHECK,NET,NDF,NGP,NNE,NNL,
3NNT,NNRE,NPNT,IPLANE,NBAND
DATA BRIT,'BRIT' /
PAI = 3.141592653589793
II = IVEC + 1
DO 25 L = 1, NET
NNE2 = 2 * NNE
DO 11 I = 1, NNE2, 2
11 FE(I) = 0.D0
K = NN(L,9)

```

```

CCCCCCCCCCCC C
C
C
C
C
C
C
C
C
C
C
C
C

```

```

C      DUM1 = 2.00 * PAI * DENS(K) * OMEGA**2
C      IF {Q1.EQ.BRIT} DUM1 = DUM1 / 386.0885827
C      IF {Q1.NE.BRIT} DUM1 = DUM1 / 580.665
C      DO 9 N = 1,NGP
C      ETA = GX(N)
C      DO 8 M = 1,NGP
C      XI = GX(M)
C      SHP(1) = .25D0 * (1.D0-XI) * (1.D0-ETA) * (-XI-ETA-1.D0)
C      SHP(2) = .5D0 * (1.D0-XI**2) * (1.D0-ETA)
C      SHP(3) = .25D0 * (1.D0+XI) * (1.D0-ETA) * (XI-ETA-1.D0)
C      SHP(4) = .5D0 * (1.D0-ETA**2) * (1.D0+XI)
C      SHP(5) = .25D0 * (1.D0+XI) * (1.D0+ETA) * (XI+ETA-1.D0)
C      SHP(6) = .5D0 * (1.D0-XI**2) * (1.D0+ETA)
C      SHP(7) = .25D0 * (1.D0-XI) * (1.D0+ETA) * (-XI+ETA-1.D0)
C      SHP(8) = .5D0 * (1.D0-ETA**2) * (1.D0-XI)
C
C      DERIVATIVES OF THE SHAPE FUNCTIONS WITH RESPECT TO XI.
C
C      DXI(1) = .25D0 * (1.D0-ETA) * (2.D0*XI+ETA)
C      DXI(2) = -.XI * (1.D0-ETA)
C      DXI(3) = .25D0 * (1.D0-ETA) * (2.D0*XI-ETA)
C      DXI(4) = .5D0 * (1.D0-ETA**2)
C      DXI(5) = .25D0 * (1.D0+ETA) * (2.D0*XI+ETA)
C      DXI(6) = -.XI * (1.D0+ETA)
C      DXI(7) = .25D0 * (1.D0+ETA) * (2.D0*XI-ETA)
C      DXI(8) = -.5D0 * (1.D0-ETA**2)
C
C      DERIVATIVES OF THE SHAPE FUNCTIONS WITH RESPECT TO ETA.
C
C      DETA(1) = .25D0 * (1.D0-XI) * (2.D0*ETA+XI)
C      DETA(2) = -.5D0 * (1.D0-XI**2)
C      DETA(3) = .25D0 * (1.D0+XI) * (2.D0*ETA-XI)
C      DETA(4) = -.ETA * (1.D0+XI)
C      DETA(5) = .25D0 * (1.D0+XI) * (2.D0*ETA+XI)
C      DETA(6) = .5D0 * (1.D0-XI**2)
C      DETA(7) = .25D0 * (1.D0-XI) * (2.D0*ETA-XI)
C      DETA(8) = -.ETA * (1.D0-XI)
C
C      CONSTRUCTION OF THE JACOBIAN MATRIX.

```

```

C C      DO 20 I = 1, 2
C C      DO 20 J = 1, 2
C C      20 AJ(I,J) = 0.00
C C
C C      DO 22 I = 1, NNE
C C      AJ(1,1) = AJ(1,1) + DXI(I) * Z(NN(L,I))
C C      AJ(1,2) = AJ(1,2) + DXI(I) * Z(NN(L,I))
C C      AJ(2,1) = AJ(2,1) + DETA(I) * Z(NN(L,I))
C C      AJ(2,2) = AJ(2,2) + DETA(I) * Z(NN(L,I))
C C      22 CONTINUE
C C
C C      DET = AJ(1,1) * AJ(2,2) - AJ(1,2) * AJ(2,1)
C C      DEI = 1.000/DET
C C
C C      INVERTING THE JACOBIAN AND STORING IN AJ AGAIN.
C C
C C      DUM = AJ(1,1) * DEI
C C      AJ(1,1) = AJ(2,2) * DEI
C C      AJ(1,2) = -AJ(1,2) * DEI
C C      AJ(2,1) = -AJ(2,1) * DEI
C C      AJ(2,2) = DUM
C C
C C      CR = 0.00
C C      DO 2 I = 1, NNE
C C      M1 = NN(L,I)
C C      2 CR = CR + SHP(I) * R(M1)
C C
C C      DUM = DUM1 * CR * DET * GC(N) * GC(M)
C C
C C      DO 17 I = 1, NNE
C C      K = 2 * I - 1
C C      FE(K) = FE(K) + SHP(I) * DUM
C C      17 CONTINUE
C C
C C      8 CONTINUE
C C      9 CONTINUE
C C

```

```

C C C C C
      CONSTRUCTION OF SYSTEM CENTRIFUGAL LOAD VECTOR
      DO 13 I = 1, NNE
      I2 = 2 * I - 1
      I1 = NN(L, I) * 2 - 1
      F(I1, I1) = F(I1, I1) + FE(I2)
      18 CONTINUE
      25 CONTINUE
      RETURN
      END
C C C C C

```



```

CC      FACT = 2.D0 * PAI * PRES
      DO 10 L = 1, NEUP
      K = 2 * L - 1
      DO 2 I = 1, NNE
      2 FE(I) = 0.D0
CC
      DO 9 N = 1, NGP
      XI = GX(N)
CC
      SHP(1) = -.5D0 * (1.D0 - XI) * XI
      SHP(2) = 1.D0 - XI**2
      SHP(3) = .5D0 * (1.D0 + XI) * XI
CC
      DXI(1) = XI - .5D0
      DXI(2) = -2.D0 * XI
      DXI(3) = XI + .5D0
CC
      RR = 0.D0
      DNZ = 0.D0
      DNR = 0.D0
CC
      DO 3 I = 1, 3
      J = NP(N(K+I-1))
      RR = RR + SHP(I) * R(J)
      DNZ = DNZ + DXI(I) * Z(J)
      DNR = DNR + DXI(I) * R(J)
      3 CONTINUE
CC
      DO 4 I = 1, 3
      J1 = 2 * I - 1
      J2 = 2 * I
      FE(J1) = FE(J1) + RR * DNZ * SHP(I) * GC(N)
      FE(J2) = FE(J2) - RR * DNR * SHP(I) * GC(N)
      4 CONTINUE
      9 CONTINUE
CC      CONSTRUCTION OF SYSTEM PRESSURE LOAD VECTOR
CCCC

```

```

C
      5 CONTINUE
      10 CONTINUE
      F(I1,J) = F(I1,J+1) + FE(I1,J+1) * FACT
      J2=2*J - 1 NPN(K+I-1) + FE(J1)* FACT
      J1=2*I-1
      DO 5 I=1,3
      RETURN
      END

```

```

00028380
00028390
00028400
00028410
00028420
00028430
00028440
00028450
00028460
00028470
00028480
00028490

```

```

CCCCCCCCCCCC C C C C C C C C C C C
SUBROUTINE DISPL
*****
THIS SUBROUTINE SOLVES FOR DISPLACEMENTS, ONCE GIVEN
THE DECOMPOSED SYSTEM STIFFNESS MATRIX AND THE LOAD
VECTORS. THE LOAD VECTORS ARE REPLACED BY THE
SOLUTIONS OF THE DISPLACEMENTS.
*****
IMPLICIT REAL * 8 (A-H,O-Z)
COMMON /ONE/ AJ(2,2), B(4,16), COPY(37,298), D(4,4), DL(4,4),
1DNZR(2,8), ESM(16,16), F(24,298), SSM(298,66), STOR(20,150)
COMMON /TWO/ AL(5), DENS(5), DELTA(8), DXI(8), E(5), EPSO(4),
1FE(16), GC(4), GX(4), POI(5), R(149), SHP(8), SHF(5), TK(5), Z(149)
COMMON /TRE/ AXIALF, DTI, EVERY, FORSL, OMEGA, HTCI, HTCO,
1POSI, PRES, Q1, Q2, Q3, SHMAX, SMAX, TIME, TIME1, TINIT, TMAXM, TREF,
2TMAXO, RMAXM, RMAXO, ZMAXM, ZMAXO, Q4, BIG
COMMON /FOR/ NN(40,9)
COMMON /FIV/ NCFI(37), NCFO(37), NNR(37), NRE(37),>NNL(37),
1NPN(37)
COMMON /SIX/ IBAND, IEXT, INTP, IVEC, JVEC, KK, LOAD, NCFIT, NCFOI,
1INCHECK, NET, NDF, NGP, NNE,>NNLT, NNRT, NPASS, NPROB, NRPP1, NRPP0,
3NNNT, NNRE, NPNT, IPLANE, NBAND
ZERO = 0.00
II = IVEC + JVEC + 1
IF (IPLANE.EQ.0) II = II - 1
DO 500 LL = 1, II
DO 200 I = 1, NDF
F(LL,I) = F(LL,I) / SSM(I,1)
DO 100 J = 2, IBAND
L = I + J - 1
IF (L.GT.NDF) GO TO 200

```

```

DUM = SSM(I,J)
100 F(LL,L) = F(LL,L) - DUM * F(LL,I)
200 CONTINUE
CCC

F(LL,NDF) = F(LL,NDF) / SSM(NDF,I)
DO 400 L=2, NDF
K = NDF - L + 1
SUM = ZERO
DO 300 J=2, IBAND
M = J + K - 1
IF (NCF.LT.M) GO TO 400
SUM = SUM + SSM(K,J) * F(LL,M)
300 F(LL,K) = (F(LL,K) - SUM) / SSM(K,I)
400 CONTINUE
CCC

500 CONTINUE
IF (IPLANE.EQ.0) GO TO 20
CCCCC

CORRECTING THE DISPLACEMENTS

FORSL = 0.00
DO 7 I=1, NNRE
DUM = 0.00
DO 6 J=1, NDF
DUM = DUM + COPY(I,J) * F(II,J)
6 FORSL = FORSL + DUM
CCC

IF (JVEC.EQ.0) GO TO 10
C

DO 8 I=1, JVEC
SUM = 0.00
DO 9 J=1, NNRE
K = 2 * NRE(J)
SUM = SUM + COPY(J,K) * F(IVC+I,K) * BIG
9 CONTINUE
C

AXIALF = AXIALF + SUM
C

10 CONTINUE

```

```

C      C
      DO 11 I = 1, IVEC
      SUM = 0.0
      DO 12 J = 1, NNRE
      K = 2 * NRE(J)
      SUM = SUM + COPY(J,K) * F(I,K) * BIG
12 CONTINUE
C
      FACT = (AXIALF+ SUM)/FORSL
C
      DO 28 L = 1, NDF
      F(I,L) = F(I,L) + FACT * F(II,L)
28 CONTINUE
11 CONTINUE
C      C      C      C
      USING THE PRINCIPLE OF SUPERPOSITION
C      C
20 M = 0
   IF (LOAD.NE.0) M = M + 1
   IF (PRES.NE.0) M = M + 1
   IF (OMEGA.NE.0) M = M + 1
   IF (M.EQ.0) GO TO 40
   DO 41 I = 1, IVEC
   DO 41 J = 1, NDF
   DO 41 K = 1, M
   F(I,J) = F(I,J) + F(IVEC+K,J)
41 CONTINUE
C
40 CONTINUE
RETURN
END

```

```

00029440
00029450
00029460
00029470
00029480
00029490
00029500
00029510
00029520
00029530
00029540
00029550
00029560
00029570
00029580
00029590
00029600
00029610
00029620
00029630
00029640
00029650
00029660
00029670
00029680
00029690
00029700
00029710
00029720
00029730
00029740
00029750
00029760
00029770
00029780

```

```

00029790
00029800
00029810
00029820
00029830
00029840
00029850
00029860
00029870
00029880
00029890
00029900
00029910
00029920
00029930
00029940
00029950
00029960
00029970
00029980
00029990
00030000
00030010
00030020
00030030
00030040
00030050
00030060
00030070
00030080
00030090
00030100
00030110
00030120
00030130
00030140
00030150
00030160
00030170
00030180
00030190
00030200
00030210
00030220
00030230
00030240

SUBROUTINE STRESS
*****
AT THE TWO GAUSS POINTS ON THE TWO BOUNDARIES OF
EACH ELEMENT (WHERE ETA=+1 AND ETA=-1) STRESSES
ARE CALCULATED IN THIS SUBROUTINE, AND ALSO THE
MAXIMUM, MEAN AND OCTAHEDRAL SHEAR STRESSES
ARE EVALUATED AND THEN THE TIME AND LOCATION OF
THE MAXIMUM STRESSES ARE DISTINGUISHED HERE.
*****
IMPLICIT REAL * 8 (A-H,O-Z)
COMMON /ONE/ AJ(2,2),B(4,16),COPY(37,298),D(4,4),DL(4,4),
1DNZR(2,8),ESM(16,16),F(24,298),SSM(298,66),STOR(20,150)
COMMON /TWO/ AL(5),DENS(5),DETA(8),DXI(8),E(5),EPSO(4),
1FE(16),GC(4),GX(4),PCI(5),R(149),SHP(8),SHT(5),TK(5),Z(149)
COMMON /TRE/ AXIALF,CTI,EVERY,FORS1,OMEGA,HICI,HICO,
1POSI,PKES,Q1,Q2,Q3,SHMAX,SMAX,TIME,TIMEI,TINIT,TMAXM,TREF,
2TMAXO,RMAXM,RMAXO,ZMAXM,ZMAXO,Q4,BIG
COMMON /FOR/ NN(40,9)
COMMON /FIV/ NCFI(37),NCFO(37),NNR(37),NRE(37),NNL(37),
1NPN(37)
COMMON /SIX/ IBAND,IEXT,INTP,IVEC,JVEC,KK,LOAD,NCFIT,NCFOT,
1NCHECK,NET,NDF,NGP,NNE,NNLT,NNRT,NPASS,NPROB,NRPPI,NRPP0,
3NNT,NNRE,NPNT,IPLANE,NBAND
DIMENSION EDISP(16),SIG(4),EPS(4)
GX(1) = -.577350269189626
GX(2) = -GX(1)
GC(1) = 1.0D0
GC(2) = GC(1)
PAI = 3.141592653589793
NNE2 = 2.*NNE
DO 25 LL=1,NET

```


CCC

```

SHP(1) = .25D0 * (1.D0-XI) * (1.D0-ETA) * (-XI-ETA-1.D0)
SHP(2) = .5D0 * (1.D0-XI**2) * (1.D0-ETA)
SHP(3) = .25D0 * (1.D0+XI) * (1.D0-ETA) * (XI-ETA-1.D0)
SHP(4) = .5D0 * (1.D0-ETA**2) * (1.D0+XI)
SHP(5) = .25D0 * (1.D0+XI) * (1.D0+ETA) * (XI+ETA-1.D0)
SHP(6) = .5D0 * (1.D0-XI**2) * (1.D0+ETA)
SHP(7) = .25D0 * (1.D0-XI) * (1.D0+ETA) * (-XI+ETA-1.D0)
SHP(8) = .5D0 * (1.D0-ETA**2) * (1.D0-XI)

```

CCCCC

DERIVATIVES OF THE SHAPE FUNCTIONS WITH RESPECT TO XI.

```

DXI(1) = .25D0 * (1.D0-ETA) * (2.D0*XI+ETA)
DXI(2) = -XI * (1.D0-ETA)
DXI(3) = .25D0 * (1.D0-ETA) * (2.D0*XI-ETA)
DXI(4) = .5D0 * (1.D0-ETA**2)
DXI(5) = .25D0 * (1.D0+ETA) * (2.D0*XI+ETA)
DXI(6) = -XI * (1.D0+ETA)
DXI(7) = .25D0 * (1.D0+ETA) * (2.D0*XI-ETA)
DXI(8) = -.5D0 * (1.D0-ETA**2)

```

CCCCC

DERIVATIVES OF THE SHAPE FUNCTIONS WITH RESPECT TO ETA.

```

DETA(1) = .25D0 * (1.D0-XI) * (2.D0*ETA+XI)
DETA(2) = -.5D0 * (1.D0-XI**2)
DETA(3) = .25D0 * (1.D0+XI) * (2.D0*ETA-XI)
DETA(4) = -ETA * (1.D0+XI)
DETA(5) = .25D0 * (1.D0+XI) * (2.D0*ETA+XI)
DETA(6) = .5D0 * (1.D0-XI**2)
DETA(7) = .25D0 * (1.D0-XI) * (2.D0*ETA-XI)
DETA(8) = -ETA * (1.D0-XI)

```

CCCCC

CONSTRUCTION OF THE JACOBIAN MATRIX.

```

DO 20 I = 1, 2
DO 20 J = 1, 2
20 AJ(I,J) = 0.D0

```

CC

```

DO 22 I = 1, NNE

```

00030730
00030740
00030750
00030760
00030770
00030780
00030790
00030800
00030810
00030820
00030830
00030840
00030850
00030860
00030870
00030880
00030890
00030900
00030910
00030920
00030930
00030940
00030950
00030960
00030970
00030980
00030990
00031000
00031010
00031020
00031030
00031040
00031050
00031060
00031070
00031080
00031090
00031100
00031110
00031120
00031130
00031140
00031150
00031160
00031170
00031180
00031190
00031200

```

C      AJ(1,1) = AJ(1,1) + DXI(I) * Z(NN(L,I))
C      AJ(1,2) = AJ(1,2) + DXI(I) * Z(NN(L,I))
C      AJ(2,1) = AJ(2,1) + DETA(I) * Z(NN(L,I))
C      AJ(2,2) = AJ(2,2) + DETA(I) * Z(NN(L,I))
C      22 CONTINUE
C
C      DET = AJ(1,1) * AJ(2,2) - AJ(1,2) * AJ(2,1)
C      DEI = 1.000/DET
C
C      INVERTING THE JACOBIAN AND STORING IN AJ AGAIN.
C
C      DUM = AJ(1,1) * DEI
C      AJ(1,1) = AJ(2,2) * DEI
C      AJ(1,2) = -AJ(1,2) * DEI
C      AJ(2,1) = -AJ(2,1) * DEI
C      AJ(2,2) = DUM
C
C      CR = 0.00
C      DO 2 I = 1, NNE
C      M1 = NN(L,I)
C      2 CR = CR + SHP(I) * R(M1)
C
C      EVALUATION OF THE B MATRIX
C
C      DO 15 I = 1, NNE
C      K1 = 2 * I - 1
C      K2 = K1 + 1
C      B(1,K1) = AJ(1,1) * DXI(I) + AJ(1,2) * DETA(I)
C      B(2,K1) = AJ(2,1) * DXI(I) + AJ(2,2) * DETA(I)
C      B(3,K1) = SHP(I) / CR
C      B(4,K1) = B(1,K2)
C      B(4,K2) = B(2,K1)
C      15 CONTINUE
C
C      POSITION OF GAUSS POINT
C
C      RPOS = 0.00
C      ZPOS = 0.00
C      DO 8 I = 1, NNE

```

```

C      RPOS = RPOS + SHP(I) * R(NN(L,I))
      ZPOS = ZPOS + SHP(I) * Z(NN(L,I))

      WRITE (6,201) XI, ETA, RPOS, ZPOS
201  FORMAT (//,40X,'AT LOCATION XI=',IPD14.7,' AND ETA=',
2,TIME,8X,'SIGMA Z',9X,'SIGMA R',8X,'SIGMA THETA',7X,
3,TAU R Z,7X,'TEMPERATURE',6X,'MEAN STRESS',7X,'OCTA',
4,SHEAR',//)
C      DO 30 KK = 1, IVEC
      EVALUATION OF THE DISPLACEMENT OF ELEMENT L

      DO 4 I = 1, NNE
      I2 = I * 2
      I1 = I2 - 1
      J2 = NN(L,I) *
      J1 = J2 - 1
      EDISP(I1) = F(KK,J1)
      EDISP(I2) = F(KK,J2)
4

      EVALUATING TEMPERATURE AT GAUSS POINT

      TGP = 0.00
      DO 31 I = 1, NNE
      M1 = NN(L,I)
31  TGP = TGP + SHP(I) * STOR(KK,M1)

      TGP = TGP - TREF
      TGP = AL(NN(L,9)) * TGP

      EVALUATION OF STRAIN VECTOR FOR A GAUSS POINT ON
      THE BOUNDARY OF ELEMENT L.

      DO 5 I = 1, 4
      G = 0.00
      DO 5 J = 1, NNE2
      G = G + B(I,J) * EDISP(J)
5  EPS(I) = G

```

```

00031690
00031700
00031710
00031720
00031730
00031740
00031750
00031760
00031770
00031780
00031790
00031800
00031810
00031820
00031830
00031840
00031850
00031860
00031870
00031880
00031890
00031900
00031910
00031920
00031930
00031940
00031950
00031960
00031970
00031980
00031990
00032000
00032010
00032020
00032030
00032040
00032050
00032060
00032070
00032080
00032090
00032100
00032110
00032120
00032130
00032140
00032150
00032160

```

```

C      NOTE: INITIAL STRAIN {4} IS ZERO AND IS NOT
C      BEING SUBTRACTED FROM EPS{4}
C
C      DO 6 I = 1, 3
C      6 EPS(I) = EPS(I) - TGP
C
C      DO 10 I = 1, 4
C      G = 0.00
C      DO 10 J = 1, 4
C      G = G + D(I,J) * EPS(J)
C      10 SIG(I) = G
C
C      SMEAN IS THE MEAN STRESS
C      TMAXM IS THE TIME WHEN MAX. MEAN STRESS OCCURS.
C      OCTA IS THE OCTAHEDRAL SHEAR STRESS
C      TMAXO IS THE TIME WHEN MAX. OCTAHEDRAL SHEAR
C      STRESS OCCURS.
C
C      SMEAN = (SIG(1)+SIG(2)+SIG(3))/3.00
C
C      CALCULATION OF OCTAHEDRAL SHEAR STRESS
C
C      OCTA = 0.00
C      OCTA = OCTA + (SIG(1)-SIG(2))**2 + (SIG(2)-SIG(3))**2
C      OCTA = (OCTA + (SIG(3)-SIG(1))**2) / 9.00
C      OCTA = OCTA + (2.00/3.00) * SIG(4)**2
C      OCTA = DSQRT(OCTA)
C
C      TGP = TGP / AL(NN(L,9)) + TREF
C      TIME = STGR(KK,NNT+1)
C
C      202 WRITE (6,202) (TIME,{SIG(I),I=1,4},TGP,SMEAN,OCTA)
C      FORMAT (3X,F7.2,7(3X,IPD14.6})
C
C      FINDING MAX. MEAN STRESS
C
C      SMEAN = DABS(SMEAN)
C      IF (SMEAN.GT.SMAX) GO TO 27

```

```

00032170
00032180
00032190
00032200
00032210
00032220
00032230
00032240
00032250
00032260
00032270
00032280
00032290
00032300
00032310
00032320
00032330
00032340
00032350
00032360
00032370
00032380
00032390
00032400
00032410
00032420
00032430
00032440
00032450
00032460
00032470
00032480
00032490
00032500
00032510
00032520
00032530
00032540
00032550
00032560
00032570
00032580
00032590
00032600
00032610
00032620
00032630
00032640

```

```

27 GO TO 29
   SMAX = SMEAN
   TMAXM = STOR(KK, NNT+1)
   RMAXM = RPOS
   ZMAXM = ZPOS
29 CONTINUE

```

CCCC

FINDING MAX. OCTAHEDRAL SHEAR STRESS.

```

28 IF (OCTA.GT.SHMAX) GO TO 28
   GO TO 30
   SHMAX = OCTA
   TMAXO = STOR(KK, NNT+1)
   RMAXO = RPOS
   ZMAXO = ZPOS

```

CCC

```

30 CONTINUE
13 CONTINUE
25 CONTINUE

```

CC

```

RETURN
END

```

```

00032650
00032660
00032670
00032680
00032690
00032700
00032710
00032720
00032730
00032740
00032750
00032760
00032770
00032780
00032790
00032800
00032810
00032820
00032830
00032840
00032850
00032860
00032870
00032880
00032890
00032900
00032910
00032920

```

APPENDIX C

USER'S MANUAL

In this section the procedure for the use of the present computer program is described. It is intended that a person with minimum familiarity with the details of the finite element method and computer programming be able to use this program.

The steps below are in order and the user is advised to follow them carefully.

For finding the thermal stresses in an axisymmetric body under axisymmetric loading, the user has the choice of using either the Metric or British system of units. The program will handle both systems and the necessary conversions are made automatically within the program. However, the units used in each system must be consistent and they should be as listed below in Table V.

Now for preparation of the data input for a given axisymmetric geometry with prescribed thermal and structural boundary conditions we go through the details with a simple example.

Step 1:

Draw the longitudinal cross-section of the body to scale. Identify the cylindrical coordinates R and Z , with the origin of the Z axis on the most left-hand point of the

TABLE V
UNITS FOR INPUT DATA

Note: an input card specifies whether British or Metric data is being used.

Quantity	British Unit Syst.	Metric Syst.
Coordinates	in.	cm
Time	sec.	sec.
Temperature	°F	°C
Velocity	ft./sec.	m/sec.
Axial force [✓]	lbf	kg
Pressure	lbf/in ²	kg/cm ²
Density	lbm/ft ³	gm/cm ³
Specific heat	Btu/lbm°F	cal/kg°C
Mod. of elasticity	lbf/in ²	kg/mm ²
Coef. of thermal exp.	1/°F	1/°C
Thermal conductivity	Btu/hr.ft.°F	cal/sec.cm.°C
Heat transfer coef.	Btu/hr.ft. ² °F	cal/sec.cm. ² °C
Load vector	lbf	kg
Rotational speed	Revolutions/min.	Revolutions/min.

[✓]lbf is pound force and lbm is pound mass.

body. The entire cross section must lie in the first quadrant of the coordinate plane.

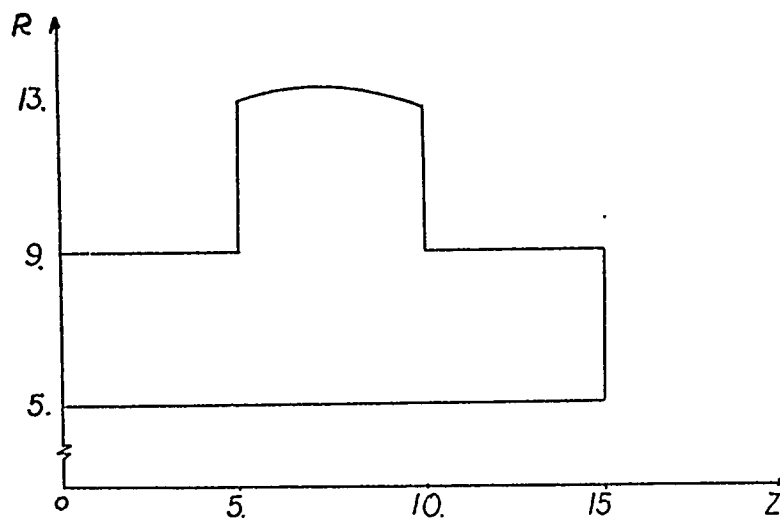


Figure 14. Longitudinal Cross Section.

Step 2:

Subdivide the cross section into a maximum of 40 quadrilateral elements. This subdivision is extremely important and we should provide smaller elements for the parts of the cross section where we expect the largest stresses or temperature gradient. If the body is made from several materials, elements should be chosen so that each element contains only one material. Any two adjacent elements must share one complete side of the quadrilateral. Number the elements, starting from 1, in any arbitrary manner (see Fig. 15).

Step 3:

Since eight-noded elements are used in the program, identify these nodal points around the boundary of each element. Four nodes will be at the corners and the other four will be

at the mid-points of the sides. Number sequentially these nodes starting from 1 and increasing in the direction where the number of elements is the least. (The numbers thus assigned are called global node numbers.) For clarity of this step assume a cross-section as in Figure 15 such that the maximum number of elements in one direction is less than the maximum number of elements in other direction. (In Fig. 15 we have maxima of 2 and 3 elements in R and Z directions respectively.) Thus we number the nodes beginning in the R direction. See Fig. 16. This method will give the minimum band width of the stiffness matrix and will save computer execution time.

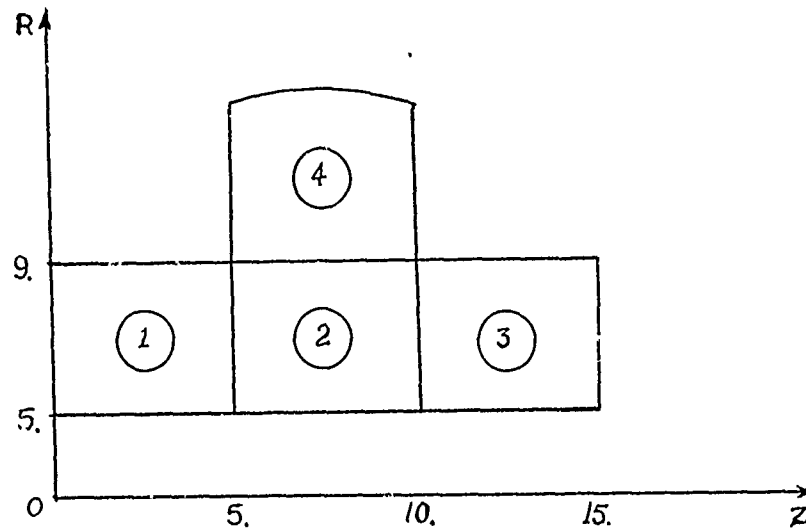
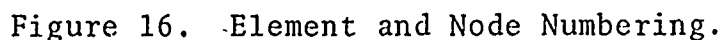


Figure 15. Subdivision into Elements.

Step 4:

At this point we are ready to prepare the first data card. Input quantities are:



NET	The total number of elements
NNT	Total number of nodes
NCN	Total number of corner nodes
NØFN	Total number of mid-side nodes not lying on the straight line joining the corner nodes (number of "Off" nodes)
NMAT	Number of different materials
NPRØB	Number of problems to be solved

For the present we take $\text{NPR}\emptyset\text{B} = 1$.

Prepare a single card that reads

NET, NNT, NCN, NØFN, NMAT, NPRØB

with the format (8I5). For our example, if all the elements are from the same material this data card reads:

4 23 10 1 1 1

[illegible]

On this card, or any succeeding one, if an input quantity (e.g., NØFN) is zero, the corresponding field may have a 0 or be left blank, unless otherwise stated.

Step 5:

For each corner node a card should be punched which reads: The node number I, the R coordinate and the Z coordinate of the node. The format is (I5, 2F10.5).

It is not necessary to sort these cards in the order of increasing or decreasing corner node numbers, they can be put together in any arbitrary order. A typical card is illustrated in step 7. There must be as many punched cards as the number of corner nodes (NCN).

Step 6:

In this step we read in the connectivity array, i.e., for each element we prepare one card which gives the global node numbers and the material identification number.

For each element, start from the lower left-hand node and move in the counterclockwise direction within that element and punch the global node numbers in order. The format is (9I5) where the last I5 is the material identification number.

It is very important that the cards prepared for this step be put together in order of elements, i.e., the first card for element 1, the second card for element 2, etc. For ease of sorting the connectivity cards the element number may be punched after column 50. As an example, for element 2 of Fig. 16 the connectivity card should read:

[illegible]

Step 7:

For each mid-side node that is on a curved element edge a card is prepared with format (I5,2F10.5) to read the global node number and the R and Z coordinates. Here, as in step 5, these cards may be put together in any arbitrary order. For the case of Fig.16, there will be only one card to be punched and it would read:

13 13.7500 7.500

[illegible]

141

Step 8:

The properties of the different materials are specified in this step. For each material a card must be prepared that gives the modulus of elasticity (E), coefficient of thermal expansion (AL), Poisson's ratio (PØI), thermal conductivity (TK), density (DENS) and specific heat (SHT).

I.e., read:

E, AL, PØI, TK, DENS, SHT

with the format (2D12.4, 4F8.3). The first card will be for material number 1, the second card for material number 2, etc.

Step 9:

For this step a single card must be punched, starting in column one, which reads the word BRITISH or METRIC in accordance with the system of units used.

This completes the input of geometric and material property data.

Step 10:

For a transient temperature problem only (no stress calculations) leave a blank card for this step and proceed directly to step 16.

For a stress problem or thermal stress problem, in this step we specify the type of problem and the structural boundary conditions.

The following quantities, when pertinent, are to be specified.

ØMEGA	The speed of rotation, about the Z axis, in revolutions per minute
PRES	The external pressure applied to a boundary segment
IEXT	The number of known temperature vectors for which evaluation of thermal stresses is desired
LØAD	The indication for an additional known load vector. If there is one, LØAD = 1, otherwise LØAD = 0
NNLT	Total number of nodes fixed in the longitudinal direction
NNRT	Total number of nodes fixed in the radial direction
IPLANE	An indication for the plane-end boundary condition. If desired IPLANE = 1

If any of these items is not applicable, the corresponding field is left blank.

Now we have to punch a single card for this step which reads:

ØMEGA, PRES, IEXT, LØAD, NNLT, NNRT, IPLANE

with format (2F10.4, 5I5).

Step 11:

If there is no pressure loading (PRES=0), omit this step. For non-zero pressure, the total number of nodes on pressure loading boundary segment (NPNT) and the global node numbers of these (pressure) nodes (NPN(I)) must be specified here. Only one segment is permitted.

We read in:

NPNT, (NPN(I), I=1, NPNT)

with the format (12I5).

7 1 4 6 11 14 19 21

[illegible]

If there is no additional load vector ($LOAD = 0$), omit this step.

Step 13:

```
READ:  (NN(I),I=1,NNLT)
```

144

Step 14:

For $NNRT > 0$ we must read in the global numbers (NNR(I)) of those nodes which are to be fixed against radial motion. The format is (10I5), i.e.,

If node 1 and 21 of our example are also fixed in the R direction, then the card for this step would read exactly as the one in Step 13.

Step 15:

For the case of end-planes-remain-plane boundary condition, i.e., (IPLANE = 1), we have to specify the total number of nodes of the right-hand end (NNRE) and the global node numbers of this end (NNR(I)).

NNRE, (NNR(I), I=1, NNRE)

For the case of Fig. 16, if it is desired to have end-planes-remain-plane boundary condition then the data for this step will be:

Step 16:

Thermal boundary conditions are imposed along discrete segments of the boundary. Each such segment must begin and end at a corner node of a boundary element. We consider separately the imposition of the various kinds of thermal boundary conditions.

Since the fluid temperature for convection is determined from a prescribed temperature-time history at entry section and a specified flow velocity (see Appendix D, Section 5), it is necessary that the terms "inside," i.e., adjacent to the symmetry axis, and "outside" have their ordinary meanings. Thus on the inside surface the convection boundary condition may be applied to a single (continuous) segment. A similar prescription may be employed for the outside portion. The entry section used for flow calculations is at the upstream end of corresponding segment.

To specify constant temperature on a portion of the boundary, the designator "inside" or "outside" may be used. Such a constant temperature portion may consist of several

discrete segments. If two different constant temperature portions are prescribed, one may be designated "inside" and the other "outside."

(c) Combinations of convection and constant temperature

The convection and constant temperature conditions can be used together, but the portion designated "inside" must have only one of these conditions prescribed and the same restriction applies to the "outside."

(d) Insulated

All portions of the boundary not included in the segments specifically identified as "inside" and "outside" are considered insulated.

The following items, when pertinent, are to be given as specified below:

TINIT The constant initial body temperature
Q2=0. For insulated boundary condition outside
Q2 > 0. For convection or constant temperature boundary condition outside
Q3=0. For insulated boundary condition inside
Q3 > 0. For convection or constant temperature boundary condition inside
Q4 < 0. If solving stress problem only
Q4 > 0. If solving temperature problem only
Q4=0. If solving thermal stress problem
AXIALF The axial force in the Z direction. As usual, + (plus) for tension and - (minus) for compression.

So, we read:

TINIT, Q2, Q3, Q4, AXIALF

with the format (5F10.4).

In case of Fig. 16, if the initial solid temperature is 60° and we have insulated outside and convection boundary condition inside with 120 kg axial compressive force, then the card for this step will be:

60.00 1.0 120.00

000 00 000000000000000000 000000029600000000 00 0000000000000001
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 6
.....

where the blanks left in the columns 31 through 40 indicate that we want to solve a thermal stress problem.

Step 17:

No action is taken in this step - we merely choose between proceeding to step 18 or jumping to step 26.

If there are no known temperature vectors for which evaluation of thermal stresses is desired, proceed to step 13.

For IEXT > 0, i.e., when some known temperature vectors are to be entered for thermal stress analysis alone or combined with some other loadings, proceed directly to step 26.

Step 18:

If (Q4 < 0); i.e., we are to solve only a stress problem, proceed directly to step 27.

Step 19:

If there is an insulated boundary condition outside ($Q2 = 0$), omit the following steps and start from step 22.

For the case of a convection or constant temperature boundary condition outside ($Q2 > 0$), we have to specify the outside heat transfer coefficient ($HTC\emptyset$) (for constant temperature $HTC\emptyset = 10^{20}$), the constant outside initial fluid temperature ($TEMP\emptyset$) which may be equal to the solid's initial temperature, the total number of nodes in contact with fluid outside ($NCF\emptyset T$), and the number of temperature ramps for outside flow ($NRAMP\emptyset$). We read:

$HTC\emptyset, TEMP\emptyset, NCF\emptyset T, NRAMP\emptyset$

with the format (D16.4,F10.4,2I5). See example in step 22.

Step 20:

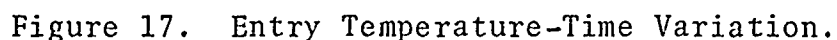
Here we read in the global node numbers of the nodes in contact with outside fluid ($NCF\emptyset(I)$). The sequence of these node numbers must be in the direction of flow velocity. Read:

$(NCF\emptyset(1), I=1, NCF\emptyset T)$

with the format (12I5) (see example in step 23).

Step 21:

For each ramp of outside flow we read in the time when the ramp starts $BDRY\emptyset(I,1)$, the initial temperature of the ramp $BDRY\emptyset(I,2)$, the final temperature of the ramp (specify only if it differs from the initial temperature of the next



Step 23:

For example, if the inside flow of Fig. 16 is from left to right, then for this step the card reads:

151

Step 24:

In this step data for inside flow are given. For each ramp I we read: the time when the ramp starts BDRYI(I,1), the initial temperature of ramp I BDRYI(I,2), the final temperature of ramp I (specify only if it differs from the initial temperature of the next ramp) BDRYI(I,3), and the velocity of the fluid for this ramp BDRYI(I,4). We read in:

$$((\text{BDRYI}(I,J), J=1..4), I=1, \text{NRAMPI})$$

with format (4F10.4).

For example of Fig. 17, let the flow velocity for the first ramp be 15 and that for the second and third segment be 20. Three cards are needed:

180.00 300.00 300.00 20.00

3^{ed} CARD

0000 00 0000 00 00000 00 000000 00 000000000000000000
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52

100.00 165.00 200.00 20.00

2nd CARD

000 00 00000000 000000 00 00002 00 00000000000000000000
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58

0.DD 80.DD 15.DD

1st CARD

[illegible]

Step 25:

In this step we must specify the following items:

DTI The time increment (step size) of trapezoidal
 integration

TIME1 The time when the first calculated temperature vector is to be stored for thermal stress evaluation

Step 26:

If there are no known temperature vectors ($IEXT = 0$) for which thermal stress evaluation is desired, omit this step.

For $IEXT > 0$ we must specify the nodal temperatures which are to be stored for thermal stress evaluation. We read:

((STØR (I,J) , J=1,NNT), I=1,IEXT)

with the format (6F10.4).

This means that we enter all the components of the first nodal temperature vector, followed by the components of the second nodal temperature vector and so on until IEXT such vectors have been entered.

Step 27:

The data cards for this problem are completed now. If no more problems are to be solved for the same geometry, omit this step.

If another problem is to be solved for the same geometry and spatial discretization, increment by 1 the NPRØB in step 4 and start the input of the new problem from step 10.

Step 28:

If there is no access to the IBM 360 computer of the Naval Postgraduate School, omit the following steps and start from step 32.

For the convenience of the user, the author has put a so-called CHECK program and the program presented in

Appendix B in the computer system at N.P.S. (Naval Post-graduate School). It is advised, however, that the user check his input data deck with the CHECK program before attempting to solve the problem.

For using the CHECK program prepare the following control cards.

```
//XXXX0000 JØB (0000,0000FT,XX00),'NAME',TIME=1
//JØBLIB DD DSN=F0609.BAKH,DISP=SIR,UNIT=2314,VØL=SER=DUFFY
//GØ EXEC PGM=CHECK,REGION=100K
//FT06F001 DD SYSØUT=A,DCB=(RECFM=FBA,LRECL=133,BLKSIZE=3325),
//          UNIT=SYSØUT
//FT05F001 DD *
```

where the first card is the regular FORTRAN job card used at this institution.

Prepare your deck as Fig. 18 and it may be read in from the hot card reader.

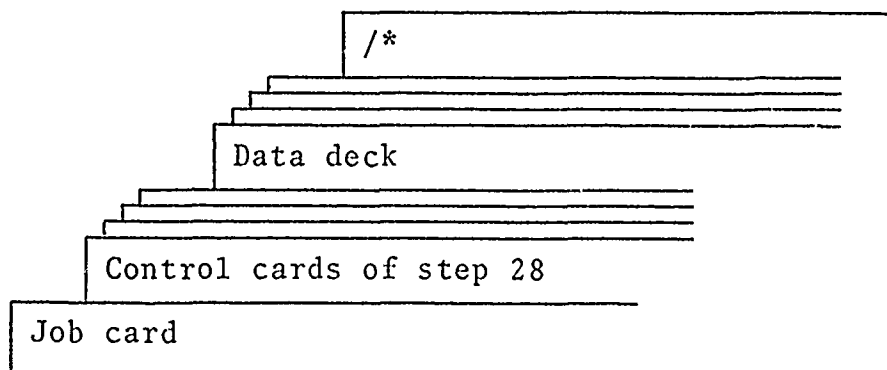


Figure 18. Set-up for Using CHECK Program.

If the output of the CHECK program has any error messages or has not been run, then there are some mistakes in the data deck or control cards.

If there are no error messages on the output of CHECK program, then study carefully the output and make sure it is the same problem that has been intended to be solved. Once the correctness of the input data has been insured, proceed to the next step.

Step 29:

AXITTS stands for Axisymmetric Transient Thermal Stress and this is the name given to the program presented in Appendix B when stored in the computer. For using that we must prepare the following control cards.

```
//XXXX0000 JØB (0000,000FT,XX00)'NAME',TIME=10
//JØBLIB DD DSN=F0609.BAKH,DISP=SHR,UNIT=2314,VØL=SER=DUFFY
//GØ EXEC PGM=AXITTS,REGION=425K
//FT06F001 DD SYSØUT=A,DCB=(RECFM=FBA,LRECL=133,BLKSIZE=3325),
//          UNIT=SYSØUT,SPACE=(CYL,(6,1))
//FT05F001 DD *
```

where the first card is the regular FØRTRAN job card. It is advised to ask for 10 minutes time, i.e., TIME=10, since this would not affect the priority of the job within the class K jobs.

Prepare the deck as in Fig. 19 (it may be read into the computer from the hot card reader) and submit a so-called

service request card, available in the computer center, to the operator on duty.

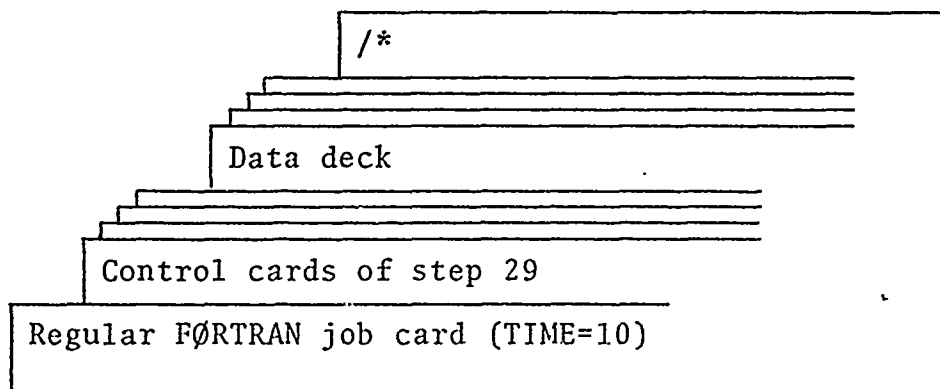


Figure 19. Deck Set-up for Using AXITTS Program.

Step 30:

If the user wishes to have a listing of the program he may prepare the following control cards as in Fig. 20 since the program is also listed on the data cell for this purpose.

Regular FØRTRAN job card

//PRNT EXEC PGM=IEBPTCH

//SYSPRINT DD DUMMY

//SYSUT1 DD DSN=F0609.BAKH,DISP=ØLD,UNIT=2321,

// VØL=SER=CEL003,DCB=(RECFM=FB,LRECL=80,BLKSIZE=2000)

//SYSUT2 DD SYSØUT=A,SPACE=(CYL,6)

//SYSIN DD *

PRINT TYPØRG=PØ,MAXFLDS=1,MAXNAME=1

MEMBER NAME=AXITTS

RECØRD FIELD=(80)

/*

Figure 20. Deck Set-up for Obtaining a Listing of the Program AXITTS.

The set-up deck of Fig. 20 may be read in from the hot card reader to get a listing of the program.

Step 31:

If the user wishes to obtain a deck of the program AXITTS he may prepare a deck as in Fig. 21 and read it in from the hot card reader. He must notify the operator on duty to be prepared for punching almost two boxes of IBM cards.

Regular FORTRAN job card

```
//PUNCH EXEC PGM=IEBPTCH
```

```
//SYSPRINT DD DUMMY
```

```
//SYSUT1 DD DSN=F-0909.BAKH,DISP=OLD,UNIT=2321,
```

```
// VOL=SER=CEL003,DCB=(RECFM=FB,LRECL=80,BLKSIZE=2000)
```

```
//SYSUT2 DD SYSOUT=B,SPACE=(CYL,6)
```

```
//SYSIN DD *
```

```
PUNCH TYPORG=P0,MAXFLDS=1,MAXNAME=1
```

```
MEMBER NAME=AXITTS
```

```
RECORD FIELD=(80)
```

```
/*
```

Figure 21. Deck Set-up for Obtaining a Punched Deck of the Program AXITTS

Step 32:

For using the program presented in Appendix B, if the user does not have access to the computer facility at N.P.S., he must punch a copy of the program. (Good luck!)

For the storage location 425K bytes are required and since the output may be longer than what usually is allowed, three additional cylinders are recommended. The execution time required depends on the size of the problem and the number of time integration steps; however, 10 minutes computer time would be sufficient for a problem of 140 nodes and 500 steps of time integration, with 20 temperature vectors stored for stress calculation.

APPENDIX D

PROGRAMMING

The computer program presented in Appendix B is formed from one main program and fifteen different subroutines. The communication between the main program and the subroutines is handled through the common blocks. In order to minimize storage requirements, most of the storage location of the system stiffness matrix is used for temperature calculations. This is accomplished by use of EQUIVALENCE statements. The total storage requirement is 425K bytes.

In this section the function of some parts of the program is discussed and the assumptions used are brought to attention.

1. MAIN PROGRAM

The main program is simply calling different subroutines when they are needed. The subroutines which are called in main program are:

INPUT, PRØB, CANDY, FLØW, FØRMV, TEMPER, PLØT, STIFF, FØRMF, CENTF, PRESS, DISPL, AND STRESS.

2. SUBROUTINE INPUT

In this subroutine the data regarding the geometry of the body, subdivisions into elements, and the material properties are read in and printed out. Calculation of the mid-side coordinates based on the straight line is done

here. This subroutine calls for subroutine PLØTRZ once to produce a graphical representation of the nodal points then the half-band-width of the system stiffness matrix is calculated from the connectivity array. Also, depending upon the choice of the system of units, the necessary conversions in each system of units are accomplished and the reference temperature (70°F or 20°C) based on the system of units is selected here.

3. SUBROUTINE PRØB

For each problem this subroutine is called once by the main program to read in and print out the information regarding the nature of the problem. Based upon the given information the type of the problem is distinguished here and for the thermal problems the length of time integration is calculated.

4. SUBROUTINE CANDY

Subroutine CANDY (C and Y) evaluates the capacitance matrix $\underline{C}$ and admittance matrix $\underline{Y}^+$ of Eq. 13, in banded form. For each problem this subroutine is called once. The functional flow chart of the subroutine CANDY is given in Fig. 22. Since the only non-zero elements of the symmetric matrix $\underline{Y}^*$ (Eq. 11) are the diagonal, first and second super-diagonals, in order to conserve storage and reduce the number of arithmetic operations, the non-zero elements are stored in three different vectors (DIAG, ØFF1, ØFF2).

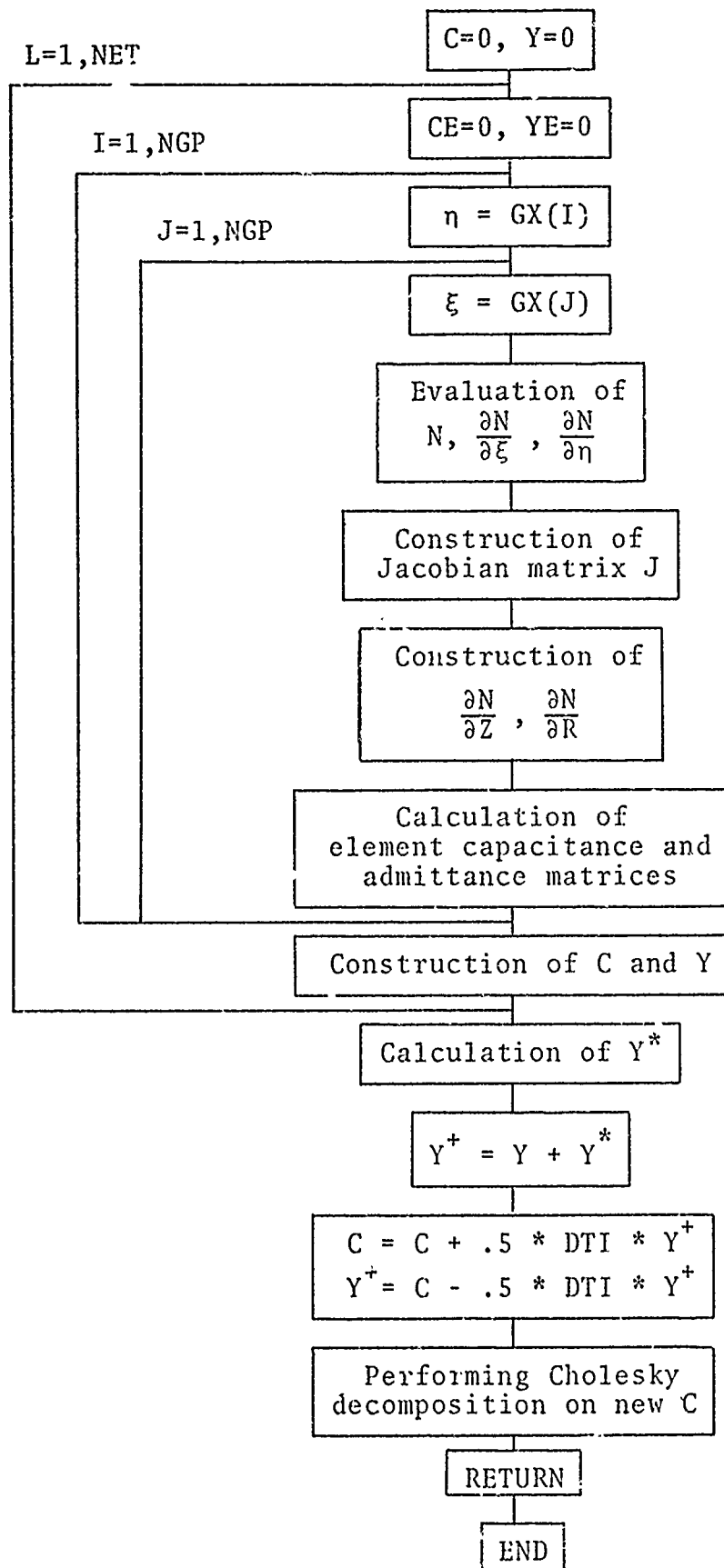


Figure 22. Functional Flow Chart of CANDY.

Once the system capacitance and admittance matrices $\underline{C}$ and $\underline{Y}^+$ are formed, then the matrices $\underline{A}$ and $\underline{G}$ (Eq. 29) are evaluated and replace $\underline{C}$ and $\underline{Y}$ respectively.

5. SUBROUTINE FLOW

If there is any convection or constant temperature thermal boundary condition, this subroutine is called from the main program for each step of time integration in order to evaluate the temperature of the fluid nodes. The calculation is based upon the constant fluid flux assumption (for both inside and outside flow). Consider a section of an irregular cylindrical pipe as Fig. 23 and focus attention on element $i+1$ on the inner boundary.

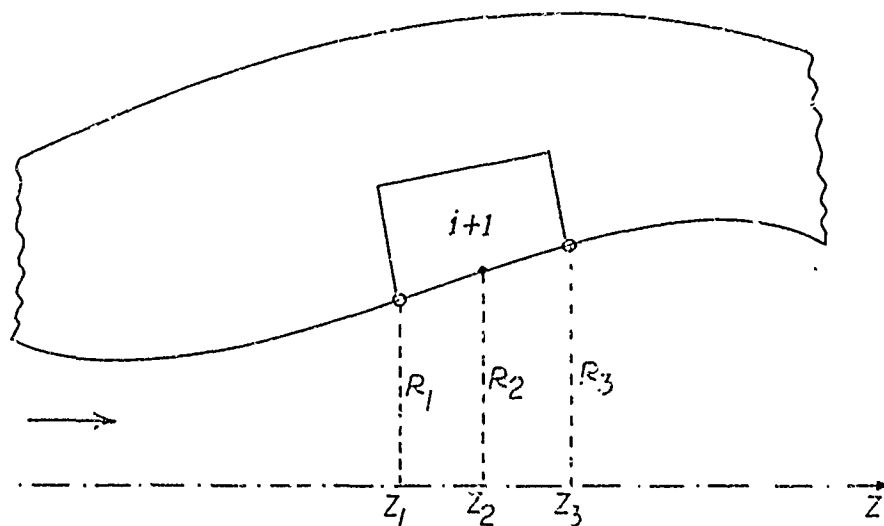


Figure 23. Fluid Node Representation for Inside Flow.

If fluid is moving from left to right we may number the R and Z coordinates of the fluid nodes of the element $i+1$ as in Fig. 23, then the volume of fluid pumped into the pipe at time τ is given by

$$Q_{in} = a v \tau, \quad (56)$$

where a and v are, respectively, the area and velocity at the entrance section. Also the volume of the pipe up to the mid-node of element $i+1$ is

$$Q_1 = V_i + \int_{Z_1}^{Z_2} \pi R^2 dZ, \quad (57)$$

where V_i is the volume of the pipe between the entrance section and element $i+1$ on the inner boundary.

Now we can write

$$R = \sum_{i=1}^3 R_i N_i, \quad (58)$$

where N_i are the one dimensional shape functions (given in Appendix A).

If we substitute Eq. 58 into 57 and make the coordinate transformation, after integration we get

$$Q_1 = V_i + \frac{\pi}{120} (Z_2 - Z_1) (31R_1^2 + 64R_2^2 + R_3^2 + 46R_1R_2 - 8R_1R_3 - 14R_2R_3). \quad (59)$$

Now by equating Eq. 59 to Eq. 56 at any time τ we may determine whether the front of the flow has already passed the mid-node of the wetted side of the element $i+1$.

A similar argument can be carried out for the end node of the element $i+1$. In that case we also get an equation similar to Eq. 59 with different numerical coefficients.

For the case of the outside flow it has also been assumed to have a constant fluid flux and a fictitious

entrance circular cross-sectional area (whose radius is the largest R plus unity) has been assumed.

The numerical coefficients of R 's in Eq. 59 are used in subroutine FLØW and it has been assumed that the temperature of a fluid "particle" does not change during passage through the active section. This is believed to be an acceptable approximation for representative values of flow velocity and active length. With this assumption, for a given entry time-temperature relation (inside and outside flow) this subroutine evaluates the fluid nodal temperatures at any time.

6. SUBROUTINE FØRMV

At every step of time integration this subroutine is called by the main program to form the vector $\underline{v}$ of the right-hand side of the discretized finite element (Eq. 13) for the given thermal boundary conditions. The program itself is self explanatory.

7. SUBROUTINE TEMPER

The nodal temperatures are calculated in this subroutine with the trapezoidal time integrations. Irons' correction is applied here after every 10 steps of time integration. The temperature vectors selected for the stress analysis are stored. The transient temperatures will be printed out at the desired interval of time.

8. SUBROUTINE STIFF

For any stress or thermal stress problem subroutine STIFF is called once by the main program to evaluate the stiffness matrix of the system. Once the stiffness matrix of the system is calculated then the desired structural boundary conditions are applied and, at the end, the Cholesky decomposition is performed. The functional flow chart of subroutine STIFF is given in Fig. 24.

9. FØRMF

In subroutine FØRMF the thermal load vectors are calculated for as many as (IVEC) given temperature vectors. These thermal load vectors are the columns of a rectangular matrix F. The provision is made that no matter how many temperature vectors are given (always $IVEC \leq 20$) the IVEC number of columns of the F are filled with the thermal load vectors and the last column of F will contain the load vector corresponding to the unit end displacement for zero axial force when the plane-end boundary condition is applied. The flow chart of subroutine FØRMF is given in Fig. 25.

10. SUBROUTINE CENTF

If the system is rotating about the axis of revolution, this subroutine is called once by the main program to evaluate the centrifugal load vector. This load vector is always placed in the first column after the thermal load vectors (IVEC+1 position) in F.

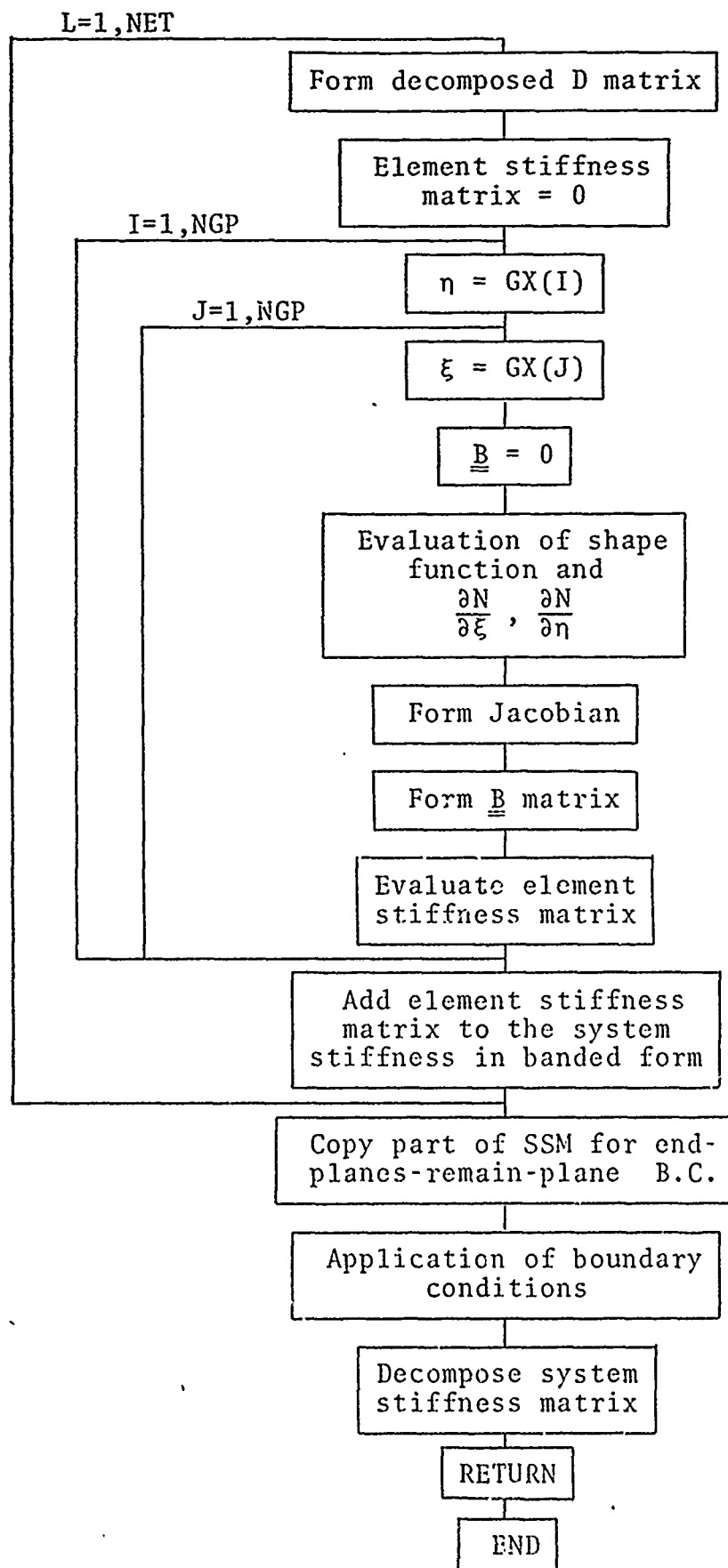


Figure 24. Functional Flow Chart of STIFF.

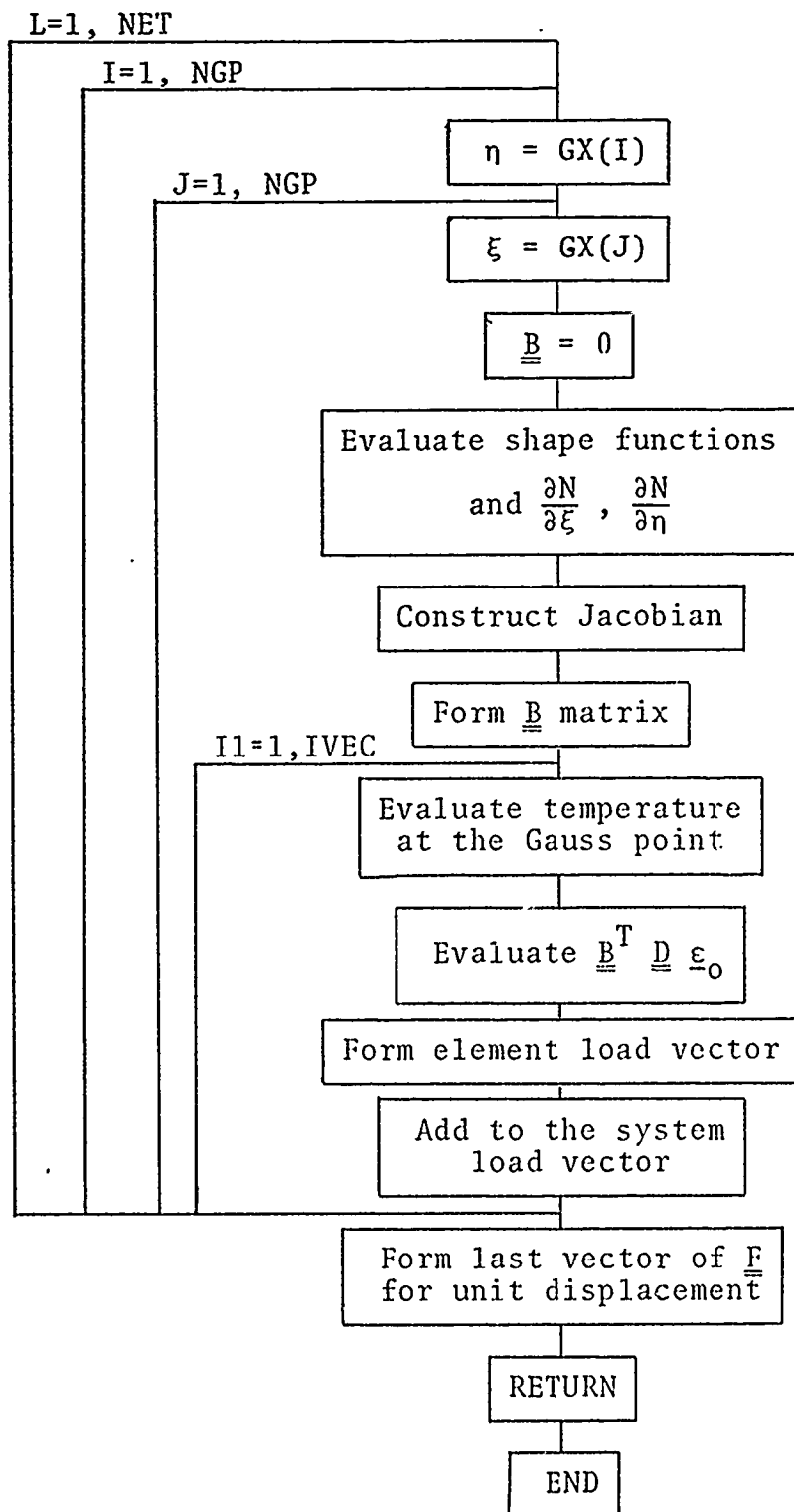


Figure .25. Functional Flow Chart of FØRMF.

11. SUBROUTINE PRESS

If there is any pressure acting on the system, the pressure load vector is calculated in this subroutine and this vector is placed in the column next to centrifugal load vector, if any, otherwise it will fill the position designated for centrifugal load vector in F.

12. SUBROUTINE DISPL

The displacement vectors are found in this subroutine. Since the stiffness matrix is already decomposed in banded form, then the displacement vectors one after another are obtained by back and forward substitution. The principle of superposition is applied here and finally the axial force, if any, is corrected for the plane-end boundary condition.

13. SUBROUTINE STRESS

For every problem this subroutine is called by the main program once to evaluate the stresses at the points $(\eta=\pm 1, \xi=\pm \frac{1}{\sqrt{3}})$ of each element. The transient stresses are calculated and printed for each displacement vector. For each problem the maximum mean stress and the maximum octahedral shearing stress, together with the corresponding times and locations, are printed.

14. LIMITATIONS

In the process of the development of the program discussed so far, it has been intended that all of the calculations and storage of data occur in the computer without

using any external devices such as disks or magnetic tapes so as to be able to solve any sizable problem in relatively short time.

For this reason the following limitations (Table VI) are set forth which give an overall size of 450K bytes to the program.

TABLE VI
MAXIMUM VALUES FOR PROGRAM PARAMETERS

NBAND	Half band-width of the system stiffness matrix	66
NMAT	Total number of different materials	5
NET	Total number of elements	40
NNT	Total number of nodes	149
NCFØT	Total number of nodes in contact with outside fluid	37
NCFIT	Total number of nodes in contact with inside fluid	37
NNRE	Total number of nodes of right-hand end	37
NNLT	Total number of nodes constrained against any longitudinal motion	37
NNRT	Total number of nodes constrained against any radial motion	37
IVEC	Total number of temperature vectors stored for thermal stresses	20
NRAMPØ	Total number of ramps for outside flow	15
NRAMPI	Total number of ramps for inside flow	15
NPNT	Total number of pressure nodes	37

LIST OF REFERENCES

1. Zienkiewicz, O. C., The Finite Element Method in Engineering Science. McGraw-Hill, 1971.
2. Arpaci, Vedat S., Conduction Heat Transfer, Addison-Wesley Publishing Company, 1966.
3. Collatz, Lothar, Numerical Treatment of Differential Equations. Third edition. Springer-Verlag, New York Inc., 1966.
4. Private communication from Bruce M. Irons to Robert E. Newton.
5. Chapman, J., Heat Transfer, Second edition. MacMillan Company, 1967.
6. Timoshenko, S. and Goodier, J. N., Theory of Elasticity, Second Edition, McGraw-Hill, 1951.
7. Fung, Y. C. Foundations of Solid Mechanics. Prentice-Hall, 1965.
8. Irons, B. M., International Journal for Numerical Methods in Engineering, Vol. 2, No. 1, p. 5-35, January-March 1970.
9. Boley, Bruno A. and Weiner, Jerome H., Theory of Thermal Stresses. John Wiley and Sons, Inc., 1962.