S	differential operator over boundary
S	surface
S_1	surface having Type 1 boundary condition
S_2	surface having Type 2 boundary condition
T	temperature
$\frac{\partial T}{\partial n}$	temperature gradient normal to surface
$T(s)$	temperature over the surface
T_∞	ambient temperature
T_1	temperature at time t_n
T_2	temperature at time $t_n + \theta \Delta t$
T_3	temperature at time t_{n+1}
t	time
Δt	length of time increment
$u(x)$	exact solution to function over domain
$\hat{u}(x)$	piecewise approximation to $u(x)$
$\hat{u}_i$	nodal value of $u(x)$
V	volume
ΔV	voltage difference (volts/ m^3)
w_r	weighting function for weighted residual method
x, y, z	global coordinates
ϵ	emissivity
$e(x)$	error between approximate and exact solution
λ_n	Eigenvalue
$\phi_i(x)$	basis function
σ	Stefan-Boltzman constant
Θ	weighting function for time integration
ξ, η, ζ	local coordinates

CANADIAN THESES ON MICROFICHE

I.S.B.N.

THESES CANADIENNES SUR MICROFICHE

National Library of Canada
Collections Development Branch

Canadian Theses on
Microfiche Service

Ottawa, Canada
K1A 0N4

Bibliothèque nationale du Canada
Direction du développement des collections

Service des thèses canadiennes
sur microfiche

NOTICE

The quality of this microfiche is heavily dependent upon the quality of the original thesis submitted for microfilming. Every effort has been made to ensure the highest quality of reproduction possible.

If pages are missing, contact the university which granted the degree.

Some pages may have indistinct print especially if the original pages were typed with a poor typewriter ribbon or if the university sent us a poor photocopy.

Previously copyrighted materials (journal articles, published tests, etc.) are not filmed.

Reproduction in full or in part of this film is governed by the Canadian Copyright Act, R.S.C. 1970, c. C-30. Please read the authorization forms which accompany this thesis.

THIS DISSERTATION
HAS BEEN MICROFILMED
EXACTLY AS RECEIVED

AVIS

La qualité de cette microfiche dépend grandement de la qualité de la thèse soumise au microfilmage. Nous avons tout fait pour assurer une qualité supérieure de reproduction.

S'il manque des pages, veuillez communiquer avec l'université qui a conféré le grade.

La qualité d'impression de certaines pages peut laisser à désirer, surtout si les pages originales ont été dactylographiées à l'aide d'un ruban usé ou si l'université nous a fait parvenir une photocopie de mauvaise qualité.

Les documents qui font déjà l'objet d'un droit d'auteur (articles de revue, examens publiés, etc.) ne sont pas microfilmés.

La reproduction, même partielle, de ce microfilm est soumise à la Loi canadienne sur le droit d'auteur, SRC 1970, c. C-30. Veuillez prendre connaissance des formules d'autorisation qui accompagnent cette thèse.

LA THÈSE A ÉTÉ
MICROFILMÉE TELLE QUE
NOUS L'AVONS REÇUE

National Library
of CanadaBibliothèque nationale
du Canada

Canadian Theses Division Division des thèses canadiennes

Ottawa, Canada
K1A 0N4

60017

PERMISSION TO MICROFILM — AUTORISATION DE MICROFILMER

• Please print or type — Écrire en lettres moulées ou dactylographier

Full Name of Author — Nom complet de l'auteur

LORNA ANNE BERESFORD

Date of Birth — Date de naissance

29 MARCH 1958

Country of Birth — Lieu de naissance

CANADA

Permanent Address — Résidence fixe

1203 - 716 THE WEST MALL
ETOBICOKE, ONT
M9C 4X6

Title of Thesis — Titre de la thèse

FINITE ELEMENT ANALYSIS OF THREE
DIMENSIONAL TRANSIENT HEAT FLOW IN SOLIDS

University — Université

CARLETON UNIVERSITY

Degree for which thesis was presented — Grade pour lequel cette thèse fut présentée

MASTER OF MATERIALS ENGINEERING

Year this degree conferred — Année d'obtention de ce grade

1983

Name of Supervisor — Nom du directeur de thèse

J. A. GOLDAK

Permission is hereby granted to the NATIONAL LIBRARY OF
CANADA to microfilm this thesis and to lend or sell copies of
the filmThe author reserves other publication rights, and neither the
thesis nor extensive extracts from it may be printed or other-
wise reproduced without the author's written permission.L'autorisation est, par la présente, accordée à la BIBLIOTHÈ-
QUE NATIONALE DU CANADA de microfilmer cette thèse et de
prêter ou de vendre des exemplaires du film.L'auteur se réserve les autres droits de publication, ni la thèse
ni de longs extraits de celle-ci ne doivent être imprimés ou
autrement reproduits sans l'autorisation écrite de l'auteur

Date

19/01/83

Signature

L. Beresford

FINITE ELEMENT ANALYSIS OF THREE DIMENSIONAL, TRANSIENT HEAT FLOW IN SOLIDS

by

© **Lorna Anne Beresford, B.Eng (Mech.)**

A thesis submitted to the Faculty of Graduate Studies in partial
fulfilment of the requirements for the degree of Master of Materials
Engineering.

Department of Mechanical and Aeronautical Engineering

Carleton University

Ottawa, Ontario

December 27, 1982

The undersigned recommend to the Faculty of Graduate Studies acceptance of the thesis "Finite Element Analysis of Three Dimensional, Transient Heat Flow in Solids" submitted by Lorna Anne Beresford, B.Eng. (Mech.) in partial fulfilment of the requirements for the degree of Master of Materials Engineering.

Thesis Supervisor

Chairman, Department of Mechanical
and Aeronautical Engineering

Carleton University

January 18, 1983

Abstract

A finite element program is presented which provides an approximate solution to three dimensional problems described by the quasi-harmonic equation, with particular attention paid to transient heat flow in solids. The theory associated with formulating these problems is outlined, identifying various boundary conditions which can exist along with the corresponding finite element formulation. The fortran subroutine FT3D, part of the ASGARD finite element package is discussed in detail and several test problems are presented to illustrate the capabilities of the program.

Acknowledgements

I would like to thank my advisor, Dr. John A. Goldak for his assistance and encouragement during preparation of this thesis. The assistance of Ms. Carol Morgan and Mr. Warren Hik who shared their programming expertise, and Mr. Alex Golob who aided in the preparation of the figures throughout the text, was also greatly appreciated.

Table of Contents

	Page
Chapter 1 Introduction	1
Chapter 2 Theory	2
2.1 The Quasi-harmonic Equation	2
2.2 Solving the Transient Field Problem	4
2.2.1 The Galerkin Weighted Residual Method	4
2.2.2 The Galerkin Finite Element Method	5
2.3 The 3-D Field Transient Element	5
Chapter 3 Finite Element Implementation	8
3.1 Finite Element Solution in the Time Domain	8
3.2 Components of the Basic element Equation	9
3.3 Program Implementation	9
3.3.1 Point Loading	10
3.3.2 Surface Loading	10
3.3.3 Volume Integrals	11
3.3.4 Final Assembly	13
3.4 Error Checking	13
Chapter 4 Test Problems	16
4.1 Test Problem Requirements	16
4.2 Test Problems for FT3D	16
4.2.1 Conduction in a Bar	16
4.2.2 Distributed Surface Loads	17
4.2.3 Iteration	19
4.2.4 Coupling 2D and 3D	21
Chapter 5 Conclusions	28
References	29
Appendix I Numerical Integration	30
Product Rules	30
Non-Product Rules	31

Choosing Integration Schemes	32
Appendix II Program Documentation	33
Appendix III Sample Input and Output Files	43
Appendix IV Source Code for FT3D	53

List of Figures

2.1 Domain Ω , and Boundaries S_1 and S_2	3
2.2 The 20 Node Brick	6
2.3 The Unit Brick	6
3.1 Flowchart for FT3D	15
4.1 Discretization for Conduction in a Bar	22
4.2 Temperature Distribution for Conduction in a Bar	23
4.3 Discretization for Convection on one Face of a Block	24
4.4 Temperature Distribution for Convection on a Block ($t=0.1s$)	25
4.5 Discretization for Radiation on one Face of a Block	26
4.6 Temperature Distribution for Radiation on a Block	27
4.7 Discretization for Transition from 3D to 2D	28

List of Tables

2.1 Physical Situations Described by the Quasi-harmonic Equation	2
2.2 Shape Functions for the 20-Node Brick	7
3.1 Weighting Functions for Integration in the Time Domain	9
4.1 Temperature vs. Distance at Time $t = 0.1$ (for $\Delta t = 0.1$)	19
4.2 Temperature vs. Distance for various solutions to radiation problem	20
A1-1 Two and Three Dimensional Gaussian Integration	30
A1-2 Numerical Integration Schemes for Quadrilaterals and Bricks (Product Rules)	31
A1-3 Numerical Integration Schemes for Quadrilaterals and Bricks (Non-Product Rules)	33

Nomenclature

a_i	coefficient on basis function
α	thermal diffusivity
A	intercept of curve for linearly temperature dependent conductivity
B	slope of curve for linearly temperature dependent conductivity
B	matrix of global derivatives of shape functions
C	capacitance matrix
c_p	volumetric specific heat (J/m^3C)
D	conduction matrix
EFLOAD	effective load vector
ELSTIF	element stiffness matrix
EXLOAD1	load vector from flux and volume source at time t_n
EXLOAD2	load vector from convection and radiation at time $t_{n+\theta\Delta t}$
EXLOAD3	load vector from flux and volume source at time t_{n+1}
F	external load vector
$f(x)$	function over surface
$g(x)$	function over domain
H	Biot number
h	surface heat transfer coefficient
J	Jacobian matrix
J_{ij}	entry in i^{th} row j^{th} column of Jacobian matrix
K	stiffness matrix
k	isotropic thermal conductivity
k_x, k_y, k_z	thermal conductivity in x,y,z, direction
L	length
$\mathcal{L}$	differential operator over domain
N_i	shape function for the i^{th} node
Q	volume heat source (J/m^3s)
q	surface flux (J/m^2s)
r	number of weighting functions
R	resistivity (Ωm)

S	differential operator over boundary
S	surface
S_1	surface having Type 1 boundary condition
S_2	surface having Type 2 boundary condition
T	temperature
$\frac{\partial T}{\partial n}$	temperature gradient normal to surface
$T(s)$	temperature over the surface
T_∞	ambient temperature
T_1	temperature at time t_n
T_2	temperature at time $t_n + \Delta t$
T_3	temperature at time t_{n+1}
t	time
Δt	length of time increment
$u(x)$	exact solution to function over domain
$\hat{u}(x)$	piecewise approximation to $u(x)$
$\hat{u}_i$	nodal value of $u(x)$
V	volume
ΔV	voltage difference (volts/ m^2)
w_r	weighting function for weighted residual method
x, y, z	global coordinates
ϵ	emissivity
$e(x)$	error between approximate and exact solution
λ_n	Eigenvalue
$\phi_i(x)$	basis function
σ	Stefan-Boltzman constant
Θ	weighting function for time integration
ξ, η, ζ	local coordinates

CHAPTER 1 : INTRODUCTION

An understanding of heat flow in solids is of particular importance in the analysis of welds. Distortion and cracking in and around welds continues to be a major area of concern. To better understand and control these problems requires an increased knowledge of the stresses in cooling welds. Since these stresses are caused by a temperature gradient within the material, this leads us to the study of the thermal history of the weld as our starting point for analysis. Currently, most of the existing knowledge of transient temperature distributions in welds comes from experimental work.

The analytic solutions which do exist rely heavily on simplifying assumptions. These assumptions frequently take the form of a limitation of the problem so that a plane temperature model is sufficient. This is in response to a belief that a three-dimensional transient model for temperature and stress distribution would be 'too unwieldy for a direct practical application' [Rosenthal] or 'would lead to prohibitive costs using today's computers and computational techniques.' [Andersson]

When attempting to analyse a weld in a plate, a two-dimensional model can either ignore the heat flow through the thickness of the plate as in the work by Masubuchi [Andersson], and Fujita and Nomoto [Andersson] or it can consider a plane section perpendicular to the weld which will not consider heat flow along the weld. This second technique, based on Rosenthal's Theory of Moving Sources of Heat and the quasi-stationary state provides fairly good correlation between analytic results and experimental data in weld sections sufficiently removed from the material boundaries.

These techniques are of little use however, in analysing run-on and run-off tabs for example where the problems of distortion and cracking are particularly severe. The two-dimensional models are also insufficient when attempting to analyse the thermal history of an electron-beam weld [Chakravarti]. There are also many areas apart from welding where a fully three-dimensional transient model is desirable, for example in the cooling of castings.

In an attempt to address this problem, a three-dimensional, transient, finite element model of heat flow in solids has been developed. While by necessity, the three-dimensional model is considerably more expensive to run than a two-dimensional analysis, the costs are far from prohibitive as is frequently claimed.

CHAPTER 2 : THEORY

2.1 THE QUASI-HARMONIC EQUATION

The subroutine FT3D solves the basic quasi-harmonic equation in three dimensions for transient or time dependent situations. This equation

$$\frac{\partial}{\partial x} \left(k_x \frac{\partial T}{\partial x} \right) + \frac{\partial}{\partial y} \left(k_y \frac{\partial T}{\partial y} \right) + \frac{\partial}{\partial z} \left(k_z \frac{\partial T}{\partial z} \right) + Q = c \frac{\partial T}{\partial t} \quad (2.1)$$

describes a class of physical situations known as field transient problems. The notation used throughout this discussion and in the code itself is that associated with a heat flow problem. However it can just as readily be used to describe any of the situations found in Table 2.1 [Hinton].

TABLE 2.1 Physical Situations Described by the Quasi-harmonic Equation

Physical Problem	Unknown (T)	k_x, k_y, k_z	Q
Gas diffusion	Concentration	Diffusivity	
Groundwater seepage	Pressure Head	Permeability	
Compressible flow	Stream function Velocity potential	Density	
Magnetic Field	Magnetic Potential Reluctivity	Current density	
Torsion	Stress Function Warping Function	(shear modulus) ⁻¹ shear modulus	Twist
Reynolds Film Lubrication	Pressure	$\frac{(\text{Film thickness})^3}{\text{viscosity}}$	Lubricant Supply
Electric Field	Electric potential	Conductivity	

The quasi-harmonic equation describes the situation within a domain. On the boundary surrounding this domain, two types of boundary conditions may exist.

Fig. 2.1 Domain Ω , and boundaries S_1 and S_2 .

Type 1. The Essential or Dirichlet boundary condition requires that the value of the temperature be specified on the boundary.

$$T = T_1 \quad \text{on } S_1 \quad (2.2)$$

Type 2. The Natural or second kind of boundary condition can be thought of as a boundary loading condition, where the temperature gradient normal to the surface must be specified. This type of boundary condition can be further broken down into three cases.

Case a) The Neumann boundary condition describes a surface at which the temperature gradient normal to the surface is zero, i.e. a perfectly insulated surface.

$$\frac{\partial T}{\partial n} = 0 \quad \text{on } S_2 \quad (2.3)$$

Case b) The flux boundary condition exists when a known finite temperature gradient is prescribed over a surface indicating that an amount of heat q flows into the body per unit area.

$$-k \frac{\partial T}{\partial n} = q_0 \quad \text{on } S_2 \quad (2.4)$$

Case c) The case in which a normal temperature gradient is specified on a surface is the convection or radiation boundary condition described by

$$-k \frac{\partial T}{\partial n} = h(T - T_{\infty}) \quad \text{on } S_2 \quad (2.5)$$

For convection, h is simply the convective heat transfer coefficient, however, in the case of radiation, h is a linearized coefficient obtained from

$$h = \epsilon\sigma(T^2 + T_\infty^2)(T + T_\infty) \quad (2.6)$$

Substituting this equation for h into Eq. 2.5 gives the more familiar

$$-k \frac{\partial T}{\partial n} = \epsilon\sigma(T^4 - T_\infty^4) \quad (2.7)$$

describing the surface temperature gradient.

2.2 SOLVING THE TRANSIENT FIELD PROBLEM

One approach to solving the transient field problem of the type described by Eqns. 2.1-2.7 involves finding a variational formulation, and the minimization of the functional obtained. This can be a difficult procedure, and in some cases the variational formulation may not exist. A trial function approximation to the field problem can be employed if the variational formulation exists, using a method such as the Ritz method. This numerical solution however, is still dependent on the existence of a suitable functional.

When solving an engineering problem, the existence of a variational formulation is not of prime importance. A numerical solution can be obtained without the need for a variational formulation using a trial function approximation and one of the weighted residual methods such as collocation, least squares or the Galerkin weighted residual method.

2.2.1 THE GALERKIN WEIGHTED RESIDUAL METHOD

The Galerkin weighted residual method is popular with Finite Element users because it produces the same solution as the variational method when a variational formulation exists, and the best solution when no functional can be found. The quasi-harmonic equation describing the transient field problem comes from a family of equations having the general form

$$L[u(x)] = g(x) \quad \text{in } \Omega \quad (2.8)$$

$$S[u(x)] = f(x) \quad \text{on } S$$

where $u(x)$ is the exact solution ($x = (x_1, x_2, x_3)$ may be a vector). A weighted residual formulation solves an approximate solution where $u(x)$ is replaced by a series of polynomials.

$$\hat{u}(x) = a_1\phi_1(x) + a_2\phi_2(x) + \dots + a_p\phi_p(x) \quad (2.9)$$

Substituting this approximation into Eq. 2.8(a) gives an error

$$\epsilon(x) = L[\hat{u}(x)] - g(x) \quad (2.10)$$

The best approximation will minimize the error, i.e. the solution which will satisfy the equation

$$\int_{\Omega} \epsilon(x) \omega_r(x) d\Omega = 0 \quad , \quad r = 1, 2, \dots, p \quad (2.11)$$

The choice of test or weighting functions w , distinguishes the Galerkin weighted residual method from other techniques. The weighting functions in the Galerkin method are the basis functions $\phi_i(x)$ found in Eq. 2.9.

2.2.2 THE GALERKIN FINITE ELEMENT METHOD

The Galerkin finite element method is an adaptation of the more general Galerkin weighted residual method. The classical trial function methods such as the Ritz, collocation, least squares, and Galerkin method, can all be adapted for use with finite element techniques. The difference between a weighted residual method and its finite element counterpart is in the definition of the trial functions. The classical trial function techniques attempt to define each trial function over the entire domain while the finite element technique utilizes trial functions that are only defined within each element i.e. a piece-wise polynomial approximation.

In the Galerkin finite element method the basis or trial functions, ϕ_i are often called shape functions, and the nodal values $\hat{u}_i$ of the unknown $\hat{u}(x)$ are the Fourier coefficients. This results in an approximate solution of the form

$$\hat{u}(x) = \sum_{i=1}^P N_i(x) \hat{u}_i \quad (2.12)$$

The integral which must be solved to minimize the error now becomes

$$\int_{\Omega} (L(\sum_{i=1}^P N_i(x) \hat{u}_i) - g(x)) N_i(x) d\Omega = 0 \quad (2.13)$$

2.3 THE THREE DIMENSIONAL FIELD TRANSIENT ELEMENT

The subroutine which will be discussed in detail in Chapter 3 was developed to solve a three dimensional transient field problem of the type governed by the quasi-harmonic equation. The element type chosen for use in this subroutine was the 20 node, curvilinear, isoparametric brick shown in Fig. 2.2.

Fig. 2.2 The 20 node brick.

This curvilinear brick is transformed to the unit brick shown in Fig. 2.3 which allows all computations to be done over a unit volume or unit area.

Fig. 2.3 The unit brick.

This feature along with the fact that we have an isoparametric brick, together with the use of the Galerkin method, reduces the requirement for shape functions to one set in three dimensions for volume integrals and a second set in two dimensions for surface integrals. The shape functions can be found in Table 2.2 [Segerlind].

TABLE 2.2 Shape Functions for a 20 Node Brick

	Corner Node	Midedge Node
Surface 2D	$\frac{1}{4}(1+S)(1+T)(S+T-1)$	$\frac{1}{2}(1-S^2)(1+T)$
Volume 3D	$\frac{1}{4}(1-R^2)(1+S)(1+T)$	$\frac{1}{8}(1+R)(1+S)(1+T)(R+S+T-2)$
Volume 3D	$\frac{1}{4}(1-S^2)(1+R)(1+T)$	
Volume 3D	$\frac{1}{4}(1-T^2)(1+R)(1+S)$	

where $S = \xi/\xi_i$, $T = \eta/\eta_i$, $R = \zeta/\zeta_i$. The local coordinates of the Gauss point are (ξ, η, ζ) and the local coordinates of the i^{th} node are (ξ_i, η_i, ζ_i) .

CHAPTER 3 : FINITE ELEMENT FORMULATION

The finite element method is a numerical procedure for obtaining an approximate solution to a system of differential equations. This is done by taking some continuous quantity, temperature in the case of a heat transfer problem, and replacing it by a series of piecewise continuous functions each defined over a subdomain within the total domain. A weighted residual method such as Galerkin's Method is used to solve an integral form of the differential equation within each subdomain or element. This produces an element stiffness matrix and applied load vector which can then be assembled into a global system of equations, and used to solve for nodal values. The equation which is obtained at the element level has the form

$$C \frac{dT}{dt} + K T + F = 0 \quad (3.1)$$

This equation is solved by the Finite Element method using elements defined in the time domain [Zienkiewicz].

3.1 FINITE ELEMENT SOLUTION IN THE TIME DOMAIN

The Finite Element method is applied to the transient solution as a trial function discretization of a series of finite time domains. Solving the problem over such a series of consecutive time increments or time domains with new initial conditions leads to a recursive solution of the type which can be derived through finite difference or Runge-Kutta procedures. In the Finite Element formulation we define the shape functions in terms of local variables as follows, noting from Eq. 3.1 that only first order derivatives are required.

$$\begin{aligned} N_n &= 1 - \Theta & \frac{dN_n}{dt} &= -\frac{1}{\Delta t} \\ N_{n+1} &= \Theta & \frac{dN_{n+1}}{dt} &= \frac{1}{\Delta t} \end{aligned} \quad (3.2a)$$

where

$$\Theta = \frac{t}{\Delta t}, \quad 0 \leq \Theta \leq 1 \quad (3.2b)$$

This allows us to rewrite Eq. 3.1 as

$$\left(\frac{C}{\Delta t} + K \Theta \right) T_{n+1} + \left(\frac{-C}{\Delta t} + K(1 - \Theta) \right) T_n + F_{n+1} \Theta + F_n(1 - \Theta) = 0 \quad (3.3)$$

which is solved to give T_{n+1} . The solution is an implicit, explicit, Galerkin, or Crank-Nicholson formulation depending on the value of Θ as shown below. Hughes points out the importance of evaluating the terms in the stiffness matrix (K and C), and the convection and radiation load vector at the intermediate temperature $T = \Theta T_{n+1} + (1 - \Theta) T_n$. The

contributions from point flux loads, distributed surface flux loads, and distributed volume sources are evaluated at T_n for EXLOAD1 and at T_{n+1} for EXLOAD3 [Hughes].

TABLE 3.1 WEIGHTING FUNCTIONS

	Θ
Implicit	1
Explicit	0
Galerkin	2/3
Crank-Nicholson	0.5

3.2 COMPONENTS OF THE BASIC ELEMENT EQUATION

The element conduction matrix K depends on the geometry of the problem and contains a contribution from any convection or radiation boundary conditions present. It is identical to the element stiffness matrix for a steady-state or equilibrium problem, and can be expressed as

$$K = \int_V B^T DB dV + \int_{S_2} h N^T N dS \quad (3.4)$$

The "transient" component of the problem is accommodated by the element capacitance matrix C , which is obtained by solving

$$C = \int_V c_p N^T N dV \quad (3.5)$$

The term c_p contains contributions from the volumetric specific heat and the time step.

The external load vector is composed of terms arising from the various applied boundary conditions which can be applied as point, surface, or volume loads. These loads are assembled into the external load vector through the equation

$$F = Q_i + \int_V Q N^T dV + \int_{S_2} q N^T dS + \int_{S_2} h T_\infty N^T dS \quad (3.6)$$

The point and volume loads are applied flux, either a heat source or sink, but the surface loading can represent either a distributed flux, or convection or radiation on a surface. When radiation loading is being considered the radiation term is linearized to a heat transfer coefficient equivalent to the convective heat transfer coefficient.

3.3 PROGRAM IMPLEMENTATION

Following the format described in previous sections for isolating various components of the basic element equation (Eq. 3.1), the subroutine can be broken down into five sections as follows:

1. Housekeeping and initializing.
2. Point Loading
3. Surface Loading
4. Volume Integrals
5. Final assembly.

These five sections follow one another as seen in Fig. 3-1, the flowchart for FT3D. This approach first considers all types of boundary conditions, i.e. the surface integrals, and then evaluates the stiffness matrix entries, i.e. the volume integrals.

There are some features of this program which are worth noting before examining the details of various sections of code. Numerical integration is used throughout the program for both two dimensional and three dimensional integration. The integration scheme is chosen by the user and can vary from element to element. For the brick element, two schemes must be specified, one for surface integrals over the faces, and one for volume integrals. Details of the various integration schemes can be found in Appendix 1. Many of the matrices encountered in this program are symmetrical and this feature is exploited to improve the efficiency of the code. Symmetric matrices are stored as vectors in upper right triangular form.

The first section of the code does the 'housekeeping' for the problem. This refers to the identification of the element, increment, and iteration of current concern and the retrieval of the information required for describing the situation. This includes extracting material properties for the element from a large property array which contains material properties for all the element types in the problem, initializing local variables, and reading any previous values of nodal loads or temperatures from a keyed file. The treatment of these loads and temperatures is dependent on whether the subroutine is being called for the first time, for a new increment, or for a new iteration. If nodal temperatures have been specified at a global level then these values are assumed to be constant for all increments, as are the nodal loads. However, the element subroutine FT3D is only concerned with element level loads.

3.3.1 POINT LOADING

The simplest element level loading condition from a computational viewpoint is point loading. For this loading case, the user specifies the actual values of thermal point loads (J/s) which must be applied at the nodes. These load values are read directly from the input file and stored in the external load vector.

3.3.2 SURFACE LOADING

Two types of surface loads are considered in the program; distributed surface flux and convection or radiation on a surface. For both the surface loading cases, the loaded face must first be identified and the area computed so that an integration can be performed over the surface and equivalent nodal values for the distributed load can be obtained. The user specifies the face being loaded by listing the nodes on that face in the input file. To be consistent with the overall numbering scheme in the program, these nodes must be listed

in order starting at any corner on the face and travelling counterclockwise around the face if one is looking at the face from outside the brick.

To integrate over the surface and find equivalent nodal values, shape functions and a Jacobian matrix must be evaluated for the transformation from a unit square in (ξ, η) space to the curvilinear surface of the face in (x, y, z) space. This process is complicated by the transformation from a three dimensional surface at the global level to a two dimensional local surface. The shape functions for the surface are the same as those for an eight node quadrilateral, however the Jacobian requires terms from all three global dimensions. This results in a 2x3 matrix where the two rows are the vectors in the ξ and η directions. The entries in the Jacobian would be computed as shown below.

$$\begin{aligned} J_{11} &= \sum_{i=1}^8 \frac{\partial N_i}{\partial \xi} x_i & J_{12} &= \sum_{i=1}^8 \frac{\partial N_i}{\partial \xi} y_i & J_{13} &= \sum_{i=1}^8 \frac{\partial N_i}{\partial \xi} z_i \\ J_{21} &= \sum_{i=1}^8 \frac{\partial N_i}{\partial \eta} x_i & J_{22} &= \sum_{i=1}^8 \frac{\partial N_i}{\partial \eta} y_i & J_{23} &= \sum_{i=1}^8 \frac{\partial N_i}{\partial \eta} z_i \end{aligned} \quad (3.7)$$

Once shape functions and the Jacobian have been evaluated, the type of loading on the face is identified. If a distributed flux is present, the total flux input is summed and equivalent nodal values are computed using the surface shape functions. These nodal values are then added to the external load vector for the element along with any point flux loads that may have been present.

If the loading on the surface is convection or radiation then a surface heat transfer coefficient must be evaluated. For convection this is simply h , the convective heat transfer coefficient specified by the user, however the radiation term must be linearized using the following equation.

$$h = \epsilon \sigma (T(s)^2 + T_\infty^2)(T(s) + T_\infty) \quad (3.8)$$

The contribution from convection and radiation to the conductivity matrix is calculated by evaluating the second term in Eq. 3.4

$$\int_{S_1} h N^T N dS$$

where the shape functions used are the shape functions for the surface. The load vector also receives a contribution at this point as the last term in Eq. 3.6 is evaluated

$$\int_{S_1} h T_\infty N^T dS$$

also with the surface shape functions.

3.3.3 VOLUME INTEGRALS

The final contribution to the external load vector comes from a volume load, either a heat source or sink. This load is assumed to apply throughout the entire volume of the element and can be specified either as a single uniform value, or as a varying quantity

described by nodal values. Equivalent nodal values are obtained by integrating over the volume of the element and using the three dimensional shape functions. This corresponds to the second term in Eq. 3.6.

$$\int_V Q N^T dV$$

This heat source can represent the heat generated by resistive losses when a voltage difference is applied to a metal sample. This voltage difference is specified by the user in the property list. The temperature dependent resistivity is linearly interpolated from a table specified by the user. These two values are then used to compute the total volume heat generated for that element using the equation

$$Q = \frac{\Delta V^2}{R} \quad (3.9)$$

Once the contributions to the external load vector have all been evaluated, then the terms in the conduction and capacitance matrices must be computed. The first term in Eq. 3.4

$$\int_V B^T D B dV$$

is the remaining contribution to the conduction matrix K, which must be evaluated. The D matrix in this case is the thermal conductivity of the material, which can be specified in three ways.

1. Three constant values corresponding to the three global axes can be specified by the user allowing a different value in each direction but assuming this value to be independent of temperature. (This option must be used when specifying a voltage difference across the element.)
2. A temperature dependent value for conductivity, capacitance and enthalpy, valid in all directions, can be computed at a Gauss point by interpolating for nodal values using nodal temperatures.
3. A temperature dependent value for the conductivity at a Gauss point can be computed using the Gauss point temperature for the equation

$$K(x, y, z) = A + B \times T(x, y, z) \quad (3.10)$$

where the user specifies A and B.

The D matrix obtained from any of these cases is a 3x3 diagonal matrix which is then pre- and post-multiplied by B^T and B. The B matrix contains the global derivatives of the shape functions as shown below.

$$[B] = \begin{bmatrix} \frac{\partial N_1}{\partial x} & \frac{\partial N_2}{\partial x} & \dots & \frac{\partial N_{20}}{\partial x} \\ \frac{\partial N_1}{\partial y} & \frac{\partial N_2}{\partial y} & \dots & \frac{\partial N_{20}}{\partial y} \\ \frac{\partial N_1}{\partial z} & \frac{\partial N_2}{\partial z} & \dots & \frac{\partial N_{20}}{\partial z} \end{bmatrix}$$

The final term to be evaluated before assembly of the stiffness matrix and effective load vector is the capacitance matrix. The entries in the capacitance matrix come from evaluating Eq. 3.5 which can also be expressed as

$$C = \int_V \frac{c_p}{\Delta t} N^T N dV \quad (3.11)$$

3.3.4 FINAL ASSEMBLY

Up to this point both the conduction and capacitance matrices have been stored in upper right triangular form, however at this stage, the lower left triangle in each is filled by symmetry to simplify the final assembly. The stiffness matrix and effective load vector submitted to the solver are assembled using the following equations which take into account the finite element formulation in the time domain.

$$EFLOAD(i) = \sum_{j=1}^{20} T1_j (C(j,i) - (1 - \Theta) K(j,i)) \\ - \Theta EXLOAD3(i) - (1 - \Theta) EXLOAD1(i) - EXLOAD2(i) \quad (3.12)$$

$$ELSTIF(i,j) = \Theta K(i,j) + C(i,j) \quad (3.13)$$

Actually the upper right portion of the stiffness matrix is stored by columns as a vector.

3.4 ERROR CHECKING

In addition to the extensive error checking done at the global level, the subroutine FT3D performs many error checks on the user input element level loads. Separate load data must be presented for each element in order of increasing element number, and for each time increment during which that element is loaded, in increasing increment number. In addition, if distributed surface loads are present, the loaded faces must be identified by the global node numbers of the nodes on that face. While this method of specifying element level loads allows for a great deal of flexibility in the applied loads, it also introduces a high risk of error in the input file.

The error checking features present, are designed to help the user pinpoint the source of the problem as quickly as possible. If an element has element level loads, the first loading line encountered in the input file is a series of flags identifying the type of loading present, the element number and the increment number for which it applies. A check is made to ensure that both the element number and increment number are in the correct order. If they are found to be out of order, an error message is output which identifies the element and increment number expected by the program and the element and increment number provided by the input file.

If the element has distributed surface loads then two checks are made on the input; first that the element number is correct, and that the nodes identifying the face actually lie on a face within that element. If the element number is out of order, the error message will tell the user at which increment the error was encountered, and will give both the expected and input element numbers. If the node numbers in the input do not correspond to a face on that element, then this will be highlighted to the user along with the increment number and element number at which the problem was encountered.

An error message will also be output if the element number is found to be out of order for volume loads, giving the increment number and both the expected and incorrect element number as with the distributed loading case. After any of the error messages have been output, the program execution will halt.

Fig. 3.1 FLOWCHART for FT3D

CHAPTER 4 : TEST PROBLEMS

Numerical problem solving techniques such as the Finite Element Method are usually employed when the physical problem is such that an analytical solution is difficult or even impossible to obtain. In this type of situation the user must have a high degree of confidence in the results obtained for example, from a particular Finite Element Program. This confidence can only be obtained through rigorous testing of the program. Once the program capabilities have been demonstrated by solving simpler problems for which analytic solutions can be readily obtained, it can then be applied to more complex situations.

4.1 TEST PROBLEM REQUIREMENTS

The test problems were selected for their ability to both test and demonstrate the program capabilities. Three guidelines were followed in the problem selection to ensure that they did perform this function.

1. The problem's ability to isolate a small section of code was extremely valuable in the testing and debugging phase of the program development
2. The ability to demonstrate a single capability of the program (which is not always the same as 1.) provides a clear indication to a user of how that particular type of problem or loading condition can be handled by the program.
3. The existence of an analytic solution to the problem was necessary to allow for comparisons and error analysis of the program results.

While it was recognized that not all test problems selected would be able to fulfill the requirements of the above guidelines, this did provide an excellent starting point when considering test problems.

4.2 TEST PROBLEMS FOR FTAD

The logical starting point for testing a program is a problem that will use only the 'core' section of the code, that is, code which must be used in every problem. Once it has been established that this core section of the code is working, attention can then be turned to problems which involve a single type of boundary condition. This allows us to isolate the section of code associated with each type of boundary condition in turn.

4.2.1 CONDUCTION IN A BAR

The first test problems considered, involved one dimensional conduction in a bar. Two brick elements were used to approximate the bar (Fig. 4.1) which had material properties

$$k_x = k_y = k_z = 1$$

and an initial temperature $T_0 = 0$. This geometry provided the basis for a series of five test problems with varying boundary conditions.

Case a) The first test problem was a steady-state conduction problem with the essential boundary condition (prescribed temperature) on two faces,

$$\begin{aligned} T &= 0 & \text{at} & \quad y = 0 \\ T &= 200 & \text{at} & \quad y = 2 \end{aligned}$$

and the first type of natural boundary condition (perfect insulation) on the remaining faces. When this is solved as a static problem, done by letting $c_p = 0$ to remove the contribution of the capacitance matrix, it tests global to local coordinate transformation, shape function subroutines, and the calculation and assembly of entries in both the element conduction matrix and global stiffness matrix. An implicit formulation must be used when modelling a static problem in this manner due to oscillations in the Galerkin and Crank-Nicholson formulations. These oscillations are particularly evident when the problem has any natural boundary conditions prescribed. The solution to this problem was as expected a linear curve (Fig. 4.2) which obeys the equation

$$q = -k \frac{\partial T}{\partial x} \quad 4.1$$

Case b) The problem was solved as a transient problem with four time steps of $\Delta t = 0.5s$, and specific heat $c_p = 1.0$. This illustrates the time stepping capability of the program and checks the calculation and assembly of terms in the capacitance matrix. The solution in this case is a series of curves approaching the static solution from below (Fig. 4.2).

Case c) The static problem solved in Case (a) has as a 'result' the nodal values of the flux necessary to satisfy Eq. 4.1. The problem can be reformulated so that these flux values are applied as a boundary condition. The boundary conditions become

$$\begin{aligned} T &= 0 & \text{at} & \quad y = 0 \\ q_{\text{total}} &= 100 & \text{at} & \quad y = 2. \end{aligned}$$

These point loads can be applied at global level to check that assignments to the global load vector are being handled correctly. When applied as point loads at element level, the calculation and assembly of flux terms into the element load vector is checked. This problem also illustrates that the element level load vector is being correctly added to the global load vector. The solution in this case is the same linear curve found in case (a).

4.2.2 DISTRIBUTED SURFACE LOADS

The flux loading in the previous two problems was applied in the form of point loads which could be added directly to the load vector. The program also has the ability to handle distributed surface loads as illustrated by the next two cases.

Case d) The simplest distributed surface load is uniform convection. To illustrate this loading case, a block was modeled using five twenty node brick elements as shown in Fig. 4.3 [Burns]. The block had boundary conditions

$$T = 1.0 \quad \text{at} \quad y = 0$$

$$h = 0.6 \quad \text{at} \quad y = 1.0$$

$$T_{\infty} = 0.0$$

with perfect insulation on all remaining faces. The block had material properties

$$k_x = k_y = k_z = 1.0; \quad c_p = 1.0$$

and an initial temperature

$$T_0 = 1.0$$

The exact solution to the problem is given by

$$T(z, t) = T_0 - (T_0 - T_{\infty}) \frac{H}{1 + H} \frac{z}{L} + 2H(T_0 - T_{\infty}) \sum_{n=1}^{\infty} \frac{\sin \lambda_n L \sin \lambda_n z}{(\lambda_n L)^2 + H \sin^2 \lambda_n L} \exp -\lambda_n^2 a t$$

where

$$H = \frac{h}{k} = \text{Biot number}$$

$$a = \frac{k}{c_p} = \text{thermal diffusivity}$$

and λ_n is obtained from

$$\lambda_n L \cot(\lambda_n L) = -H$$

The finite element analysis was done with Galerkin, implicit and Crank-Nicholson formulation using ten equal time increments of $\Delta t = 0.1s$. These solutions are compared to the exact solution for $t = 0.1$ sec. The results can be found in Table 4.1 and Fig. 4.4. The error in Table 4.1 was computed from

$$\%error = \frac{\text{Exact solution} - \text{F.E.M. solution}}{\text{Exact solution}} \times 100$$

The average error for the finite element solution is -0.087% for the Galerkin, -0.765% for the implicit and 1.575% for the Crank- Nicholson formulation. Clearly, the Galerkin formulation provides the best approximation to the exact solution. This error could be reduced by taking a smaller time step, however the user must determine whether the increase in accuracy is worth the increase in cost.

TABLE 4.1 Temperature vs. Distance at Time $t = 0.1$ (for $\Delta t = 0.1$)

Distance	Temperature				%Error		
	Exact	Galerkin	Implicit	Crank-N.	Galerkin	Implicit	Crank-N.
0.0	1.0000	1.0000	1.0000	1.0000			
0.1	0.9965	0.9967	0.9957	0.9975	-0.0201	0.0803	-0.1004
0.2	0.9930	0.9928	0.9909	0.9945	0.0201	0.2115	-0.1511
0.3	0.9878	0.9879	0.9852	0.9904	-0.5062	0.2632	-0.2632
0.4	0.9795	0.9812	0.9780	0.9843	-0.1736	0.1531	-0.4900
0.5	0.9687	0.9715	0.9686	0.9750	-0.2890	0.0103	-0.6504
0.6	0.9520	0.9577	0.9560	0.9606	-0.5987	-0.4202	-0.9034
0.7	0.9304	0.9374	0.9391	0.9383	-0.7524	-0.9351	-0.8491
0.8	0.8995	0.9075	0.9159	0.9034	-0.8894	-1.8232	-0.4336
0.9	0.8626	0.8636	0.8843	0.8489	-0.1159	-2.5157	1.5882
1.0	0.8191	0.7990	0.8410	0.7636	2.4539	-2.6737	6.7757

Case e) The static problem in case (a) can also be formulated with boundary conditions

$$T = 0 \quad \text{at} \quad y = 0$$

$$q = 100 \quad \text{at} \quad y = 2$$

where q is a uniformly distributed surface flux. This problem tests the program's ability to transform nodal values of a distributed load to equivalent nodal loads. The approximate solution obtained in this case is once again the linear curve in Fig. 4.2, showing that the finite element approximation is exact for this problem.

4.2.3 ITERATION

Case f) The iterative capabilities are best illustrated by a radiation problem as radiation is highly non-linear. The test problem uses two twenty-node brick elements to model a bar with radiation on one end and a uniform volume heat source through the entire volume (see Fig. 4.5) [Burns]. The problem can be stated mathematically as follows

$$\frac{d}{dy} \left(k \frac{dT}{dy} \right) + Q = 0 \quad 0 \leq y \leq L \quad 4.2$$

where

$$k(x) = A + (B \times T(x))$$

The boundary conditions for the problem are

$$k \frac{dT}{dy} = \sigma \epsilon (T^4 - T_\infty^4) \quad \text{at} \quad y = 0$$

$$T = T_L \quad \text{at} \quad y = L$$

The numerical values used in this case were

$$L = 1.0 \quad \sigma\epsilon = 0.0025$$

$$A = 1.0 \quad T_{\infty} = 4.0$$

$$B = -0.2 \quad T_L = 1.0$$

$$Q = 0.2$$

The exact solution is

$$T = 2 - y$$

Table 4.2 compares the finite element approximation with the exact solution. It can be seen that the finite element approximation with an implicit formulation converges to the exact solution. The results obtained with a Galerkin and Crank-Nicolson solution are included merely to illustrate the error introduced by using either of these formulations to model a static problem.

TABLE 4.2 Temperature vs. Distance for various solutions to radiation problem

Solution	Iteration	Temperature				
		x=0.00	x=0.25	x=0.50	x=0.75	x=1.00
Exact		2.000	1.750	1.500	1.250	1.000
Implicit	0	0.000	0.000	0.000	0.000	0.000
	1	1.500	1.394	1.275	1.144	1.000
	2	1.854	1.655	1.446	1.227	1.000
	3	1.956	1.722	1.484	1.243	1.000
	4	1.987	1.742	1.495	1.248	1.000
	5	1.996	1.747	1.499	1.249	1.000
	6	1.999	1.749	1.500	1.250	1.000
	7	2.000	1.750	1.500	1.250	1.000
Final Solution	7	2.000	1.750	1.500	1.250	1.000
Galerkin (Final solution)	8	2.335	1.996	1.661	1.329	1.000
Crank-Nicholson (Final solution)	7	2.671	2.243	1.823	1.409	1.000

4.2.4 COUPLING 3-D WITH 2-D

Situations are often encountered which require a three-dimensional analysis over a small portion of the domain, while a two-dimensional analysis is sufficient outside of this region. In these cases, the most efficient analysis would allow the variation in the number of dimensions as required. The final test problem illustrates how a transition can be made from three dimensions to two dimensions using a twenty node brick and an eight node quadrilateral.

Case g) Steady-state, one dimensional conduction in a bar is modelled using two twenty node brick elements, and two eight node quadrilaterals as seen in Fig. 4.7. The transition from three dimensions to two dimensions is achieved using the sectorial symmetry feature of the program to create 'master-slave' pairs [see Appendix 2] of nodes on the face of the brick with the same x and y coordinates. The slave nodes can then be ignored by the adjoining quadrilateral element.

The material properties of the bar being modeled are

$$k_x = k_y = k_z = 1.0$$

with $c_p = 0.0$ to simulate a static problem, and imposed boundary conditions

$$T = 1.0 \quad \text{at} \quad y = 0.0$$

$$T = 0.0 \quad \text{at} \quad y = 4.0$$

and

$$T_{\infty} = 0.0$$

$$T_0 = 0.0.$$

The finite element approximation in this case produces an exact solution; the temperature at nodes 5, 22, and 34 which have y coordinates $y = 1.0$, $y = 2.0$, and $y = 3.0$ being 0.75, 0.50, and 0.25 respectively.

Fig. 4.1 Discretization for Conduction
in a bar

Fig. 4.2 Temperature Distribution for Conduction in a Bar

Fig. 4.3 Discretization for Convection
on one face of a block

Fig. 4.4 Temperature Distribution for Convection on a Block ($t=0.1s$)

Fig. 4.5 Discretization for Radiation
on one face of a block

Fig. 4.6 Temperature Distribution for Radiation on a Block

Fig. 4.7 Discretization for Transition
from 3D to 2D

CHAPTER 5 : CONCLUSIONS

- i) A subroutine, FT3D, was written with the following capabilities:
 - a) Three dimensional transient heat transfer
 - b) Nonlinear thermal conductivity and specific heat
 - c) Nonlinear boundary conditions.
- ii) The boundary conditions which can be applied are:
 - a) Prescribed temperature
 - b) Prescribed point or distributed surface flux
 - c) Prescribed distributed volume source
 - d) Convection
 - e) Radiation.
- iii) Time integration is performed using a two point scheme in which Crank-Nicholson, Galerkin, and Implicit are special cases.
- iv) A sectorial symmetry capability was developed to permit coupling of three and two dimensional analysis.
- v) The program achieved the performance predicted by the theory. This was verified by a number of test problems.
- vi) This subroutine provides the basic capability needed to perform an accurate thermal analysis of a number of important welding problems.

REFERENCES

B.A.B. Andersson, 'Thermal Stresses in a Submerged-Arc Welded Joint Considering Phase Transformations', Transactions of the A.S.M.E., Vol 100, pp. 356-362, October 1978.

P.J. Burns, TACO2D-A Finite Element Heat Transfer Code, Lawrence Livermore Laboratories, UC1D-17980, Rev. 2, June 1982.

A.P. Chakravarti, Modelling of Welds and Heat Sources using Finite Element Method, Masters Thesis, Carleton University, 1982.

E. Hinton, D.R.J. Owen, An Introduction to Finite Element Computations, Pineridge Press Ltd., Swansea, U.K. 1980.

T.J.R. Hughes, 'Unconditionally Stable Algorithms for Nonlinear Heat Conduction', Computer Methods in Applied Mechanics and Engineering 10, 135-139, 1977.

B. Irons, A. Sohrab, Techniques of Finite Element Analysis, Ellis Horwood Ltd., Chichester England, 1980.

G.E. Myers, Analytical Methods in Conduction Heat Transfer, McGraw-Hill, Toronto, 1971.

D. Rosenthal, 'The Theory of Moving Sources of Heat and Its Application to Metal Treatments', Transactions of the A.S.M.E., Vol. 100, pp.356-362, October 1978.

L.J. Segerlind, Applied Finite Element Analysis, J. Wiley and Sons, Inc., Toronto, 1976.

O.C. Zienkiewicz, The Finite Element Method, McGraw-Hill, 1977.

APPENDIX I NUMERICAL INTEGRATION

The explicit evaluation of integrals can be a difficult procedure allowing a high possibility of error. In order to overcome this difficulty, numerical integration techniques are used frequently in the finite element method. Numerical integration is a technique which allows us to evaluate an integral such as

$$\int_{-1}^1 f(\xi) d\xi \quad (i)$$

by evaluating the integral at a series of sampling points, multiplying the results for each point by a weight then summing to find the total. Represented in equation form, the integral becomes

$$\sum_{i=1}^n w_i f(\xi_i) \quad (ii)$$

Using the Gauss-Legendre quadrature rules the sampling points are selected to obtain the best accuracy. Using this technique, a polynomial of degree $(2n - 1)$ can be integrated exactly using n sampling points. Extending this technique to two and three dimensions poses no significant difficulties. The single integral in the one dimensional case, simply becomes a double or triple integral as required. Product Gauss rules are then used in the equation form as seen in Table A1-1.

TABLE A1-1 Two and Three Dimensional Gaussian Integration

<i>Number of Dimensions</i>	<i>Integral</i>	<i>Equation</i>
2	$\int_{-1}^1 \int_{-1}^1 f(\xi, \eta) d\xi d\eta$	$\sum_{i=1}^n \sum_{j=1}^m w_i w_j f(\xi_i, \eta_j)$
3	$\int_{-1}^1 \int_{-1}^1 \int_{-1}^1 f(\xi, \eta, \zeta) d\xi d\eta d\zeta$	$\sum_{i=1}^n \sum_{j=1}^m \sum_{k=1}^p w_i w_j w_k f(\xi_i, \eta_j, \zeta_k)$

The weights and sampling points for the product rules described above, can be found in Table A1-2 along with the scheme. The scheme is an identifying number used to select a particular integration rule.

TABLE A1-2 Numerical Integration Schemes For Quadrilaterals And Bricks (Product Rules)

Number of Dimensions	Scheme	Quadrature Rule	Sampling	Points	Weight		
2	6	4 point (2X2)	± 0.5774	± 0.5774	1.0000		
	8	9 point (3x3)	± 0.7745	± 0.7745	0.3088		
			± 0.7745	0.0	0.4938		
			0.0	± 0.7745	0.4938		
			0.0	0.0	0.7901		
	9	16 point (4x4)	± 0.8611	± 0.8611	0.1210		
			± 0.3400	± 0.8611	0.2269		
			± 0.8611	± 0.3400	0.2269		
			± 0.3400	± 0.3400	0.4529		
	10	25 point (5x5)	± 0.9062	± 0.9062	0.0561		
			± 0.5385	± 0.9062	0.1134		
			± 0.9062	± 0.5385	0.1134		
			± 0.9062	0.0	0.1348		
			0.0	± 0.9062	0.1348		
			± 0.5385	± 0.5385	0.2291		
			± 0.5385	0.0	0.2723		
			0.0	± 0.5385	0.2723		
			0.0	0.0	0.3226		
	3	2	8 point (2x2x2) complete cubic	± 0.5774	± 0.5774	± 0.5774	1.0000

NON-PRODUCT RULES

Although the product-Gauss rules will always provide a solution, they are not always the most efficient solution. A series of non-product rules were developed which provide the same accuracy as the product rules, but with greater efficiency [Irons]. The non-product rules reduce the double and triple summations in Table A1-1 to the single summation

$$\sum_{i=1}^n w_i f(\xi_i, \eta_i) \quad \text{or} \quad \sum_{i=1}^n w_i f(\xi_i, \eta_i, \zeta_i) \quad (\text{iii})$$

The weights and sampling points for the non-product rules are given in Table A1-3.

TABLE A1-3 Numerical Integration Schemes for Quadrilaterals and Bricks (Non-Product Rules)

Number of Dimensions	Scheme	Quadrature Rule	Sampling	Points		Weight
			ξ	η	ζ	
2	7	5 point	0.0000	0.0000		
			± 0.5923	± 0.5923		0.95
3	3	14 point (complete quintic)	± 0.75	± 0.75	± 0.75	0.33
			± 0.79	0.00	0.00	0.88
			0.00	± 0.79	0.00	0.88
			0.00	0.00	± 0.79	0.88
	4	27 point (complete heptic)	± 1.10	± 1.10	0.00	0.032
			± 1.10	0.00	± 1.1	0.032
			0.00	± 1.1	± 1.1	0.032
			± 0.65	± 0.65	± 0.65	0.47
			± 0.84	0.00	0.00	0.49
			0.00	± 0.84	0.00	0.49
			0.00	0.00	± 0.84	0.49
			0.00	0.00	0.00	0.78

CHOOSING INTEGRATION SCHEMES

The product-Gauss rules are recommended for two-dimensional integrations, however the non-product rules are preferred in three dimensions. Scheme 6 is the most efficient rule for an eight node quadrilateral, however it is susceptible to mechanisms. Should mechanisms occur, then scheme 7 can be used to overcome this problem with nearly the same efficiency.

In three dimensions, scheme 2 is preferred in most cases. With some boundary conditions scheme 2 may not provide enough integration points to suppress mechanism, in which case scheme 3 should be used.

APPENDIX II PROGRAM DOCUMENTATION

Summary

After a concise review of the fundamental laws of heat transfer this report presents the theoretical formulation for a finite element analysis, documents a Fortran subroutine listing, presents the results of a number of test problems to verify the code and evaluate the performance. This report is intended to be a user's guide to the program ASGAR.

Heat transfer by conduction, convection, and radiation is considered. Nonlinear analysis by the method of successive approximations allows any parameter to be a function of time, space or material. A two point time integration scheme is used with theta differencing. Crank-Nicholson, Galerkin, and implicit schemes are used frequently. Explicit integration is not currently an option.

Curvilinear isoparametric elements are used with a wide range of numerical integration schemes. These include 4, 7, 8 and 9 node quadrilaterals; 3 and 6 node triangles for two dimensional analysis. The 20 node brick is the work horse for three dimensional analysis. In one dimension 2, 3, 4 and 5 node elements are available. For additional detail see DOC1.WASIRON.

There is no restriction on the number or type of elements that can be used in any problem.

For each type of element the user must input a property table for each distinct set of properties. Each element could have a unique property table. In addition the program has a data base with temperature dependent thermal properties for 0.23% carbon steel, 304 stainless steel, and water. Additional materials can be added to the property table.

A conversational input program with extensive error checking and diagnostics assists the user in preparing and editing input data files.

Finally a graphics package can draw the mesh, number the nodes and elements, contour temperatures, draw a surface representation of the temperature. The discretization of the continuum into elements with edges and nodes is vaguely referred to as a mesh. It does resemble a fishnet or spiders web.

An elastic plastic analysis program is being developed to accept the temperature fields from the heat flow analysis for a thermal stress analysis.

Finally this code can be used for a wide variety of potential problems in fluid dynamics, electrostatics, magnetic fields, torsion, lubrication and seepage where the problem can be formulated in terms of a scalar potential.

The code does not deal with conduction-advection problems, i.e. conduction in a moving fluid. Nor does the program currently permit view factors or radiation scattering.

HOW TO RUN A JOB

There are three main steps to performing a transient heat flow analysis.

1. Preprocessing: creation of an input file. It often includes plotting a mesh with node numbers and element numbers.
2. Processing: generation of an output file using your input file and the ASGAR

finite element analysis program.

3. Postprocessing: the printing, plotting and interpretation of the results for presentation.

PREPROCESSING

You have the option of preparing your input file using the EDIT processor with the information given below or using a conversational input program by typing:

IXEQ ASGARD_TALK3_XEQ.WASIRON

PREPARING AN INPUT DATA FILE

The input data file for a transient heat flow problem contains two distinct sections. The first section of the input file deals with global parameters and is independent of the element type and number or dimensions in the problem. Element dependent information is found in the second section of the input file.

GLOBAL LEVEL SECTION

First Line GLOBAL PARAMETERS

NELZ, NFIX, NLOAD, NEXTIF, NROTATE, NSECSM, NEWRHS, MAXRHS, ITERAT, TITLELEN, TEMPLEN, NCORD, NALGOR, MINCRE, MITERA, THETA, DTIME, TOLERA, FIXSTEP

NELZ	the total number of elements in this problem.
NFIX	the total number of nodes with prescribed temperatures
NLOAD	the total number of nodes with prescribed thermal loads(J/S)
NEXTIF	the total number of nodes with prescribed conductances to zero temperature.
NROTATE	the number of nodes with rotated or inclined boundary conditions.
NSECSM	number of nodal pairs with sectorial symmetry. See sectorial symmetry.
NEWRHS	number of right hand sides in this problem (default is 1). See Jennings's 'Matrix Computations...' page 101

MAXRHS	the maximum number of right hand sides expected in any future resolution of the problem (default = 1).
ITERAT	a flag = 1 for iterative or transient problems.
TITLELEN	the number of lines occupied by title and abstract in the input file. Users are advised to include full details, including the reasons for doing the problem. There is no limit on the number of lines for title and description. For example, if 6 lines of the input file are used for the title and description, then TITLELEN = 6.
TEMPLN	set to 10 000. It is used for dynamic memory management. Most users may safely set this parameter to 10 000 and forget it.
NCORD	the number of parameters per nodal point. The coordinates x, y, z, temperature, perhaps thickness, cross-sectional area are examples. The user could specify the thermal properties at virtually every node with minor changes in coding. Default is the no. of dimensions in the problem.
NALGOR	integer to define the algorithm; 24 for static analysis; 25 for transient non-linear analysis.
MINCRE	the maximum no. of increments in this problem (default=1).
MITERA	maximum number of iterations in any time step (default=1).
THETA	a real number ranging from 0.5 to 1.0 that defines the time integration scheme; 1.0 is implicit; 0.5 is Crank-Nicolson; 2/3 is Galerkin. See Zienkiewics, chapter 21.
DTIME	magnitude of the first time increment in seconds.
TOLERA	a convergence parameter. If the maximum temperature change at any node is less than TOLERA on the first iteration, a new time increment is taken. If TOLERA is exceeded, iteration continues until MITERA is reached or until the maximum change between iterations is less than TOLERA/100.
FIXSTEP	the number of time increments with fixed DTIME.

Next TITLELEN Lines are title and description of problem

NOTE: The title and description must contain exactly TITLELEN lines.

Next Line Element topology - repeat one line for each element

NEL, LTYPE, LPOP, LNODS

NEL the element numbers in sequence with no missing elements

LTYPE an integer specifying the element type, eg. 11 for 8 node quadrilateral for transient analysis.

LPOP the property list number for this element

LNODS the element nodes listed in the correct sequence, counter clockwise for 2D elements. See element library for 3D elements in DOC1.WASIRON

Next Line Nodal Coordinates

NODE, X-coord, Y-coord, Z-coord

NODE the global node number in any order except the largest number must be last. Midedge nodes may be omitted if the edge is straight.

X-coord x coordinate of this node

Y-coord y coordinate of this node if y coordinate exists

Z-coord z coordinate of this node if z coordinate exists

Next Line Prescribed temperatures NFIX lines

SEQUENCE_No., NODE_No., 1, NODAL_TEMPERATURE

Nodes may be entered in any order. The 1 means the one and only degree of freedom at this node is prescribed.

Next Line Prescribed fluxes NLOAD lines

SEQUENCE_No., NODE_No., NODAL_FLUX in Joules/second

Prescribed fluxes can be entered as element level nodal fluxes, which is often more convenient in transient problems with time dependent fluxes.

Next Line Prescribed conductances. NEXTIF lines

SEQUENCE_No., NODE_No., CONDUCTANCE in J/s C

An easy replacement for a 2 node bar element with one node prescribed to zero temperature. Never been used to my knowledge.

Next Line Rotated or inclined boundary conditions.

SEQUENCE_No., NODE_No., ANGLES OF ROTATION

Next Line Sectorial Symmetry NSECSM lines

SEQUENCE_No., MASTER_NODE, SLAVE_NODE

If a sector of the problem domain and symmetry operations can be identified that creates the complete problem by symmetry operations such as rotation or translation on the sector, dramatic reductions in the cost of data preparation and computing are possible.

Each symmetrically equivalent pair of nodes is designated as a master slave pair. See Iron's book, chapter 20 for details on sectorial symmetry. The transition from 20 node bricks to 8 node quads or 8 node quads to 1D bar is best handled with sectorial symmetry. When using sectorial symmetry to reduce the number of dimensions in a problem, the first elements in the problem have the highest number of dimensions.

ELEMENT LEVEL SECTION

Any element loaded in any increment must have the data following the property table. For each increment data is presented only for loaded elements. For each load increment the data must be in order of increasing element number and loaded increments must be in increasing increment number.

For 2-D case :

Next Line The property list

SEQUENCE_No., SCHEME, AMBT, DMOD1, DMOD2, H, NTYPE, THICK, TEMPER, NSCONV, CP, MATNO, EMIS

SCHEME	a number identifying the Gaussian integration scheme. See DOC1.WASIRON, Appendix 1 for scheme description.
AMBT	the ambient temperature for convection or radiation
DMOD1	the thermal conductivity in the x direction
DMOD2	the thermal conductivity in the y direction
H	the heat transfer coefficient for convection
NTYPE	=1 for planar symmetry and 2 for axisymmetry
THICK	the thickness for a planar problem

TEMPER	the initial temperature, ie the temperature at time = 0
NSCONV	the number of surfaces, not edges, with convective loading
CP	the volumetric specific heat
MATNO	= 1 for 0.23% C steel, = 2 for 304 SS, = 3 for water. The thermal properties are taken from a data base in ASGARD.
EMIS	the emissivity for Stefan-Boltzmann radiation 1 .GE. EMIS .GT. 0

ELEMENT LEVEL LOADS

Next Line

NEL, NEDGE, NQSURF, NPOIN, INCRE

NEL	the element number
NEDGE	the number of loaded edges in this element
NQSURF	the number of surfaces with flux loading
NPOIN	the number of nodes with flux loads (J/s)
INCRE	this data applies to increment number INCRE

Next line if NPOIN .GT. 0

NODE_1, FLUX_1, NODE_2, FLUX_2.....NODE_LAST, FLUX_LAST

NODE_1	is the global node number of the node with FLUX_1 Any number of the nodes may have point loads. They can be entered in any order. Of course the node must belong to this element.
--------	--

Next Line if NEDGE .GT. 0

NEL, IQEDGE, HECONV, EMISV, TAMBT, (NODPRS(NOD), NOD=1,4) QEDGE(NOD),
NOD=1,4)

NEL	the element number
IQEDGE	= 1 if this edge has flux loads, otherwise = 0.

HECONV	the convective heat transfer coefficient ($J/m^2 \cdot m \cdot s \cdot ^\circ C$)
EMISV	the emissivity along this edge 0 .GT. EMISV .GT. 0.
TAMBT	the ambient temperature along this edge.
NODPRS	the global node numbers along this edge in CCW order
QEDGE	the nodal values of the flux distribution.

Repeat the above line for each edge with convective, radiation or distributed flux loads in this element.

Next Line if NQSURF.GT.0

NEL, (NQSURF(NOD), NOD = 1, LNODZ)

NEL	the element number
NQSURF	the value of the flux distribution at this node NOD
NOD	the local node numbers as listed in LNODS for this element.

The SI units for NQSURF are joules/ $m^2 \cdot m \cdot s$

For 3-D case :

Next Line Property List - one line for each property table

SEQUENCE_No., VSCHEME, DMOD1, DMOD2, DMOD3, TEMPER, CP, SCHEME, MATNO, AA, BB, DELTAV

SCHEME	a number identifying the surface (2D) integration scheme. See DOC1.WASIRON, Appendix 1 for scheme description.
VSCHEME	a number identifying the scheme for volume (3D) integration
DMOD1	the thermal conductivity in the x direction
DMOD2	the thermal conductivity in the y direction
DMOD3	the thermal conductivity in the z direction
TEMPER	the initial temperature, ie the temperature at time = 0
CP	the volumetric specific heat

MATNO = a number identifying a material whose properties are taken from a data base in ASGARD.
 1 = 0.23%C Steel - temperature dependent conductivity
 2 = 304 SS - temperature dependent conductivity
 3 = water - temperature dependent conductivity
 100 = linearly specified temperature dependent conductivity $CONDG = AA + BB \cdot TG2 = DMOD1 = DMOD2 = DMOD3$
 101 = 0.23%C Steel - temperature dependent resistivity internal heat Q generated from applied ΔTAV

ELEMENT LEVEL LOADS

Next Line - flags for element level loads

NEL, NSURF, NQVOL, NPOINT, INCRE

NEL the element number for which the flags apply
NSURF the number of surfaces which are loaded
NQVOL 0 for no volume source over the element, 1 for constant volume source, 20 for varying volume source over element.
NPOINT the number of nodes with flux loads (J/s)
INCRE this data applies to increment number INCRE

Next NPOINT lines - use a separate line for each node

NODE, QNOD

NODE is the global node number of the node with point load
QNOD Any number of the nodes may have point loads. They can be entered in any order. Of course the node must belong to this element.

Next Line if NSURF.GT.0 - use one line for each face

NEL, IQSURF, HECONV, EMIS, TAMBT, NODPRS(8), QSURF(8)

NEL the element number
IQSURF = 1 if this surface has flux loading
HECONV the convective heat transfer coefficient ($J/m^2 \cdot s \cdot ^\circ C$)

NODPRS the nodes on this surface (numbered counter-clockwise) around the face
 looking from outside the brick
QSURF the value of the distributed flux at this node NODPR

Next Line if NQVOL = 1 - constant volume source over entire element

QVOL(1)

QVOL(1) the volume source over the entire element

Next Line if QVOL = 20 - varying volume source over element

QVOL(NOD); NOD = 1, LNODS

QVOL(NOD) the volume source at this node nodes must be given in same order as topology
 lines

These lines of element level loading must be repeated in the same order for each
increment.

PROCESSING

The computer must know the name of your input file (for purposes of illustration we will call it fid_IN), the name you wish to have attached to your output file, eg fid_OUT, and the name of the program you wish to execute ASGARD.WASIRON in this case. The above steps are most conveniently done by building yet another file that could be called fid_XEQ that might read as follows:

```
!SET 5 fid_IN,FUN=IN
!DELETE fid_OUT
!SET 6 fid_OUT,CTG=YES,EXIST=NEWFILE
!ASGARD.WASIRON
```

FUN=IN confirms that it is an input file. Deleting your output file before you start avoids appending your results to the end of the file if it exists. This is your option. CTG=YES will preserve any part of the output file that is created even if the job aborts because of errors - a useful feature.

To process or execute the job, simply type:

```
!XEQ fid_XEQ
```

POSTPROCESSING

It is usual to begin with the following command to determine how many lines are in the output file.

```
!L fid_OUT
```

Often one goes into EDIT to check the last 20 lines of the file for errors and to see that the job completed successfully before printing a hard copy listing.

PRINTING a hard copy listing of fid_OUT.

```
!C fid_OUT TO LPØME351
```

For long listings you can save money and show consideration for other users by having long listings printed after 11 pm. To do this use:

```
!XEQ PRINT.WASIRON
```

It will ask you the file name, etc.

PLOTTING RESULTS

You may plot the mesh, temperature contours, or a surface representation of the temperature using GNU. See DOC_GRAPHICS.WASIRON

APPENDIX III SAMPLE INPUT AND OUTPUT FILES

```

4 11 0 0 0 5 1 1 1 2 10000 7 25 1 1 .666667 10 100 1
A 4 element problem to test the use of sectorial
symmetry for reducing a 3D problem to 2D.
1 27 1 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
2 27 1 7 6 5 21 22 23 24 25 12 11 26 27 19 18 17 28 29 30 31 32
3 11 2 24 23 22 33 34 35 36 37
4 11 2 36 35 34 33 39 40 41 42
1 0 0 0
3 1 0 0
5 1 1 0
7 0 1 0
13 0 0 1
15 1 0 1
17 1 1 1
19 0 1 1
23 1 2 0
24 0 2 0
29 1 2 1
31 0 2 1
34 1 3
36 0 3
39 1 4
41 0 4
42 0 3.5
1 1 1 1
2 2 1 1
3 3 1 1
4 9 1 1
5 10 1 1
6 13 1 1
7 14 1 1
8 15 1 1
9 39 1 0
10 40 1 0
11 41 1 0
1 22 26
2 24 27
3 22 29
4 23 30
5 24 31
1 2 1 1 1 0 0 6 0
2 6 0 1 1 0 1 1 0 0 0 0 0

```


A 4 element problem to test the use of sectorial
symmetry for reducing a 3D problem to 2D.

11:47:12 12/14/82
This is version D00 of ASGARD
Machine run on is Honeywell CP6
Run unit used is ASGARD.WASGARD
Input file : LORNA-THESIS-TEST6-IN.PN094571
Output file : LORNA-THESIS-TEST6-OUT.WASIRON

LENGTH OF VECTOR OF FLOATING WORDS = LENVEC = 10000
NUMBER OF INTEGER WORDS PER FLOATING WORD = INTEG = 2

*** WE HAVE A NEW JOB, SO NEWJOB = 1

IT IS NOT A RE-SOLUTION, SO IFCOEF = 1

NUMBER OF ELEMENTS = NELZ = 4
NUMBER OF NODES WITH SOME FIXED VALUES = NFIX = 11
NUMBER OF NODES WITH ADDITIONAL LOADS = NLOAD = 0
NUMBER OF NODES WITH ADDITIONAL STIFFNESSES = NEXTIF = 0
THE NUMBER OF NODES WITH ROTATED DOF = 0
NUMBER OF NODES RELABELLED FOR SECTORIAL SYMMETRY = NSECSM = 5
NUMBER OF RIGHT HAND SIDES = NEWRHS = 1
MAXIMUM R.H.S. ENVISAGED IN RE-SOLUTIONS = MAXRHS = 1
IS IT AN ITERATIVE PROBLEM. ITERAT = 1
NUMBER OF INCREMENTS WITH A TIME STEP FIXED, FIXSTEP = 1
THETA for explicit, implicit, Galerkin, Crank-Nicholson = .667
The time increment/step is initially = 10.000000
The maximum temperature change/step at any node is 100.000000

NALGOR=25; time marching solution to nonlinear
quasiharmonic equation using Crank-Nicholson, Galerkin,
implicit, or explicit recurrence schemes.

NUMBER OF ELEMENT TYPES NOW IMPLEMENTED = MAXTYP = 61

ELEMENT NUMBER	ELEMENT TYPE	PROPERTY = LPROP	NODE NUMBERS = LNODS
1	27	1	1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
2	27	1	7 8 9 21 22 23 24 25 12 11 26 27 19 18 17 28 29 30 31 32
3	11	2	24 23 22 33 34 35 36 37
4	11	2	36 35 34 38 39 40 41 42

MAXIMUM NODE NUMBER = NODMAX = 42
 MAXIMUM NODES PER ELEMENT = LNOMAX = 20
 MAXIMUM DEGREES OF FREEDOM PER NODE = NDFMAX = 1
 NUMBER OF DIMENSIONS, 2 OR 3, = NDIM = 3
 THE NUMBER OF PARAMETERS PER NODE = NCORD = 7
 NUMBER OF PROPERTIES, E.G. THICKNESS, DENSITY = IPROP = 16
 NUMBER OF SETS OF PROPERTIES AVAILABLE = JPROP = 2
 MAXIMUM NUMBER OF STRESSES AT A POINT = MAXTRS = 1

NODE	X	Y	Z
1	.0000	.0000	.0000
3	1.0000	.0000	.0000
5	1.0000	1.0000	.0000
7	.0000	1.0000	.0000
13	.0000	.0000	1.0000
15	1.0000	.0000	1.0000
17	1.0000	1.0000	1.0000
19	.0000	1.0000	1.0000
22	1.0000	2.0000	.0000
24	.0000	2.0000	.0000
29	1.0000	2.0000	1.0000
31	.0000	2.0000	1.0000
34	1.0000	3.0000	.0000
36	.0000	3.0000	.0000
39	1.0000	4.0000	.0000
41	.0000	4.0000	.0000
42	.0000	3.5000	.0000

RHS	FIXING NUMBER	NODE (NODFIX)	CODE (KODFIX)	FIXED VALUES (VFIX)
1	1	1	1	1.0000000

1	2	2	1	1.0000000
1	3	3	1	1.0000000
1	4	9	1	1.0000000
1	5	10	1	1.0000000
1	6	13	1	1.0000000
1	7	14	1	1.0000000
1	8	15	1	1.0000000
1	9	39	1	.0000000
1	10	40	1	.0000000
1	11	41	1	.0000000

Sectorial symmetry:

Master	Slave
22	26
24	27
22	29
23	30
24	31

NUMBER ELEMENT PROPERTIES = VPROP

1	.2000000E+01	.1000000E+01	.1000000E+01	.1000000E+01	.0000000E+00	.0000000E+00
	.6000000E+01					
	.0000000E+00	.0000000E+00	.0000000E+00	.0000000E+00	.0000000E+00	.0000000E+00
	.0000000E+00					
	.0000000E+00	.0000000E+00				
2	.5000000E+01	.0000000E+00	.1000000E+01	.1000000E+01	.0000000E+00	.1000000E+01
	.1000000E+01					
	.0000000E+00	.0000000E+00	.0000000E+00	.0000000E+00	.0000000E+00	.0000000E+00
	.0000000E+00					
	.0000000E+00	.0000000E+00				

DEGREES OF FREEDOM AT NODES OF ELEMENT OF TYPE 11 - NDF = 11111111

DEGREES OF FREEDOM AT NODES OF ELEMENT OF TYPE 27 - NDF = 1111111111111111
1 1 1 1

NODE 22 IS DUPLICATED 3 TIMES IN ELEMENT 2
 NODE 23 IS DUPLICATED 2 TIMES IN ELEMENT 2
 NODE 24 IS DUPLICATED 3 TIMES IN ELEMENT 2

** NODE 26 NEVER APPEARS IN LNODS-element nodes list

** NODE 27 NEVER APPEARS IN LNODS-element nodes list

** NODE 29 NEVER APPEARS IN LNODS-element nodes list

** NODE 30 NEVER APPEARS IN LNODS-element nodes list

** NODE 31 NEVER APPEARS IN LNODS-element nodes list

*** LIST OF 12 NONFATAL DATA ERRORS ***

ERROR 83 ASSOCIATED NUMBER 5

Detected in routine MATRON :
repeated nodes in an element might cause trouble. See section 20. 2 of the text.

ERROR 84 ASSOCIATED NUMBER 5

Detected in routine MATRON :
a node number is not used. This may be intentional.

ERROR 85 ASSOCIATED NUMBER 2

Detected in routine MATRON :
a node number which is not used has nonzero coordinates.

CREATE ELEMENT FILE

Time= 10.00000000 time increment = 10.00000000 Increment number = 1 iteration number
= 1

Flux input = .00000000E+00

ELEMENT 1 OF TYPE 27, WITH PROPERTY TABLE NUMBER 1 AND NODES 1 2 3 4 5
6 7 8

9 10 11 12 13 14 15 16 17 18 19 20

VOLUME SCHEME = 2

X CONDUCTION COEFFICIENT = 1.00000

Y CONDUCTION COEFFICIENT = 1.00000

Z CONDUCTION COEFFICIENT = 1.00000

SURFACE SCHEME = 6

INITIAL TEMPERATURE = .00000

SPECIFIC HEAT = .0000000

MATERIAL NUMBER = 0

AA - NODE	X	Y	Z
1	.0000000	.0000000	.0000000
2	.5000000	.0000000	.0000000
3	1.0000000	.0000000	.0000000
4	1.0000000	.5000000	.0000000
5	1.0000000	1.0000000	.0000000
6	.5000000	1.0000000	.0000000
7	.0000000	1.0000000	.0000000
8	.0000000	.5000000	.0000000
9	.0000000	.0000000	.5000000
10	1.0000000	.0000000	.5000000
11	1.0000000	1.0000000	.5000000
12	.0000000	1.0000000	.5000000
13	.0000000	.0000000	1.0000000
14	.5000000	.0000000	1.0000000
15	1.0000000	.0000000	1.0000000
16	1.0000000	.5000000	1.0000000
17	1.0000000	1.0000000	1.0000000
18	.5000000	1.0000000	1.0000000
19	.0000000	1.0000000	1.0000000
20	.0000000	.5000000	1.0000000

ELEMENT 2 OF TYPE 27, WITH PROPERTY TABLE NUMBER 1 AND NODES 7 6 5 21 22
23 24 25

12 11 26 27 19 18 17 28 29 30 31 32

VOLUME SCHEME - 2
 X CONDUCTION COEFFICIENT - 1.00000
 Y CONDUCTION COEFFICIENT - 1.00000
 Z CONDUCTION COEFFICIENT - 1.00000
 SURFACE SCHEME - 6
 INITIAL TEMPERATURE - .00000
 SPECIFIC HEAT - .0000000
 MATERIAL NUMBER - 0

AA - NODE	X	Y	Z
7	.0000000	1.0000000	.0000000
6	.5000000	1.0000000	.0000000
5	1.0000000	1.0000000	.0000000
21	1.0000000	1.5000000	.0000000
22	1.0000000	2.0000000	.0000000
23	.5000000	2.0000000	.0000000
24	.0000000	2.0000000	.0000000

25	.0000000	1.5000000	.0000000
12	.0000000	1.0000000	.5000000
11	1.0000000	1.0000000	.5000000
26	1.0000000	2.0000000	.5000000
27	.0000000	2.0000000	.5000000
19	.0000000	1.0000000	1.0000000
18	.5000000	1.0000000	1.0000000
17	1.0000000	1.0000000	1.0000000
28	1.0000000	1.5000000	1.0000000
29	1.0000000	2.0000000	1.0000000
30	.5000000	2.0000000	1.0000000
31	.0000000	2.0000000	1.0000000
32	.0000000	1.5000000	1.0000000

ELEMENT 3 OF TYPE 11, WITH PROPERTY TABLE NUMBER 2 AND NODES 24 23 22 33
34 35 36 37

SCHEME— 6
 AMBIENT TEMPERATURE — .00000
 X CONDUCTION COEFFICIENT — 1.00000
 Y CONDUCTION COEFFICIENT — 1.00000
 CONVECTION COEFFICIENT (H) — .00000
 NTYPE— 1
 THICKNESS — 1.00000
 TEMPERATURE — .00000
 SURFACES WITH CONVECTION — 0
 SPECIFIC HEAT — .0000000
 MATERIAL NUMBER — 0
 EMISIVITY (surface) — .00000

Node No.	24	23	22	33	34	35	36	37
X-Coord.	.00000	.50000	1.00000	1.00000	1.00000	.50000	.00000	.00000
Y-Coord.	2.00000	2.00000	2.00000	2.50000	3.00000	3.00000	3.00000	2.50000
Z-Coord.	.00000	.00000	.00000	.00000	.00000	.00000	.00000	.00000

ELEMENT 4 OF TYPE 11, WITH PROPERTY TABLE NUMBER 2 AND NODES 36 35 34 38
39 40 41 42

SCHEME— 6
 AMBIENT TEMPERATURE — .00000
 X CONDUCTION COEFFICIENT — 1.00000
 Y CONDUCTION COEFFICIENT — 1.00000
 CONVECTION COEFFICIENT (H) — .00000
 NTYPE— 1
 THICKNESS — 1.00000

TEMPERATURE -- .00000
 SURFACES WITH CONVECTION -- 0
 SPECIFIC HEAT -- .0000000
 MATERIAL NUMBER -- 0
 EMISIVITY (surface) -- .00000

Node No.	36	35	34	38	39	40	41	42
X-Coord.	.00000	.50000	1.00000	1.00000	1.00000	.50000	.00000	.00000
Y-Coord.	3.00000	3.00000	3.00000	3.50000	4.00000	4.00000	4.00000	3.50000
Z-Coord.	.00000	.00000	.00000	.00000	.00000	.00000	.00000	.00000

OUTPUT FOR ELEMENT NO. 4

For right hand side no. 1

Node No.	36	35	34	38	39	40	41	42
0	0	0	0	0	0	0	0	
0	0	0		0				
Tempera.	.250	.250	.250	.125	.000	.000	.000	.125

OUTPUT FOR ELEMENT NO. 3

For right hand side no. 1

Node No.	24	23	22	33	34	35	36	37
0	0	0	0	0	0	0	0	
0	0	0		0				
Tempera.	.500	.500	.500	.375	.250	.250	.250	.375

OUTPUT FOR ELEMENT NO. 2

7	1	.750000E+00
6	1	.750000E+00
5	1	.750000E+00
21	1	.625000E+00
22	1	.500000E+00
23	1	.500000E+00
24	1	.500000E+00
25	1	.625000E+00
12	1	.750000E+00
11	1	.750000E+00
22	1	.500000E+00
24	1	.500000E+00
19	1	.750000E+00
18	1	.750000E+00

17	1	.750000E+00
28	1	.025000E+00
22	1	.500000E+00
23	1	.500000E+00
24	1	.500000E+00
32	1	.025000E+00

For right hand side no. 1

Node No.	7	6	5	21	22	23	24	25
12	11	22	24	19	18	17	28	
22		23	24	32				

OUTPUT FOR ELEMENT NO. 1

1	1	.100000E+01
2	1	.100000E+01
3	1	.100000E+01
4	1	.875000E+00
5	1	.750000E+00
6	1	.750000E+00
7	1	.750000E+00
8	1	.875000E+00
9	1	.100000E+01
10	1	.100000E+01
11	1	.750000E+00
12	1	.750000E+00
13	1	.100000E+01
14	1	.100000E+01
15	1	.100000E+01
16	1	.875000E+00
17	1	.750000E+00
18	1	.750000E+00
19	1	.750000E+00
20	1	.875000E+00

For right hand side no. 1

Node No.	1	2	3	4	5	6	7	8
9	10	11	12	13	14	15	16	
17		18	19	20				

RIGHT HAND SIDE 1

NODE CODE REACTIONS DUE TO PRESCRIBED VARIABLES

1	1	-.13888800E-01
2	1	.55555583E-01
3	1	-.13888800E-01
9	1	.55555583E-01
10	1	.55555583E-01
13	1	-.13888800E-01
14	1	.55555583E-01
15	1	-.13888800E-01
39	1	-.27777792E-01
40	1	-.11111117E+00
41	1	-.27777792E-01

*** RMS ERROR IN STRAIN ENERGY DUE TO ROUND OFF = .00000 PERCENT.

STRAIN ENERGY IS .083333 POTENTIAL ENERGY IS .083333 DIAGONAL
ENERGY IS 15.514635

APPENDIX IV SOURCE CODE FOR FT3D

```

SUBROUTINE FT3D(COORD, EFLOAD, ELSTIF, IPROP, JPROP, LCOEF, LNODS,
1      LNODZ, LNOMAX, LPOP, LVABZ, LVMAX, MAXTRS, NDIM,
2      NEL, NELZ, NEWRHS, NODMAX, NSTRES, STRESM, VPROP, NUSTEP,
3      DTIME, TIME, FACTOR, FITERA, TFACT, KRESL, NALGOR, NINCRE, NITERA,
4      THETA, LTYPE, FLUXIN)

```

PURPOSE:

To compute the effective stiffness and load vector for FRONT for the 20 node isoparametric brick element for the transient quasiharmonic equation. Notation favors heat flow problems but fluid flow, electric field, magnetic field and lubrication problems are equally appropriate. Numerical integration is used throughout

REAL

```
1      STEFAN, TWOPI
```

INTEGER

```
1      MXGAUS, NODL, NODL3, NODLSQ, NODSRF
```

```
PARAMETER (NODL=20, MXGAUS=27, NODL3=NODL*3, NODLSQ=NODL*NODL,
```

```
1      TWOPI=6.283185308, STEFAN=5.735E-8, NODSRF=8)
```

INTEGER

```

1      EDGE(20,2), FITERA, I, IEDGE, IEFLUX, IGAUS, INCRE, INODE,
2      IODEG, IPROP, IQEDGE, IQSURF, ISCONV, ITERM, J, J1, JDEL,
3      JDIAG, JGAUS, JNODE, JPROP, K, KNODE, KOUNT, KRESL, L, LCOEF,
4      LNODS(LNOMAX, NELZ), LNODZ, LNOMAX, LPOP, LTYPE, LVABZ, LVMAX,
5      M, MATNO, MAXTRS, MIDNOD, N, NALGOR, NBARLO, ND, NDIM, NDOFN,
6      NECONV, NEDGE, NEL, NELZ, NEWRHS, NGAUS, NINCRE, NITERA, NOD,
7      NODE, NODEG, NODMAX, NODPRS(NODL), NPOINT, NQEDGE, NQSURF,
8      NSCONV, NST, NST2, NSTRES, NSURF, NUSTEP, SCHEME, TEMP,
9      LOCNOD(8), FACID, ELEMFACE(8,8), POSELSTIF(8,8,8), NQVOL,
A      KNOD, PLACE, NOGAUS, VSCHEME, ERRNO

```

REAL

```

1      AMBT, AREA, B(3,NODL), BV(NODL3), C, CAP(NODL,NODL),
2      CAPV(NODLSQ), COND(NODL,NODL), CONDG, CONDN(NODL),
3      CONDV(NODLSQ), COORD(NODMAX,NDIM), CP, CP2, CPN(NODL),
4      DB(3), DENSTY, DER(NODL3), DERG(NODL,3), DERL(3,NODL), DMOD1,

```

•
•
•

•

1

1 0,3,0,5,0,7,0,1,13,15,17,19,0,15,0,17,0,19,0,13 /

• •

[illegible]

•
•
•
•
•

- INODE: a node number either local or global
- ITERAT: a flag for iterative solutions
- ITERM:
- J, J1, JDEL, JDIAG, JGAUS, JNODE: are counters
-
- K, KNODE, KOUNT: are counters
- KRESL: a flag for reforming the stiffness matrix.
- L: a counter
- LCOEF: the maximum storage required for the stiffness matrix stored*
as the vector ELSTIF. Only the upper right triangular portion
- LNODS: column NEL lists the global node numbers for element NO. NEL
- LNODZ: the number of nodes in this element. Could be from 4 up to 12
- LNO MAX: the greatest number of nodes for any element in this problem
- LPOP: the material number for this element; e.g. VPROP(2,LPOP)
- LVABZ: the number of degrees of freedom in this element
- LVMAX: the greatest number of DOF for an element in this problem
- M
- MATNO: number for specific materials for NTHALPY, etc. 0.23%C steel=1
- MAXTRS:
- NALGOR: id number for the algorithm; 2 for Newton-Raphson
- NBARLO: The number of Gauss points in this elements
- ND: a counter for the number of dimensions
- NDM: the number of dimensions for this element
- NDOFN: the number of degree's of freedom per node for this element
- NEDGE: the number of edges with external loads
- NEFLUX: the number of edges with flux loads
- NEL: the number of the element now being processed
- NELZ: the total number of elements in this problem
- NEWRHS: the number of right hand sides processed simultaneously
- NGAUS: the number of Gauss points in this integration scheme
- NINCRE: the current increment number
- NITERA: the number iterations in this time step
- NPOINT: the number of point loads applied per element
- NOD: a node number, usually local
- NODE: a node number
- NODEG: the number of nodes along an edge in this element
- NODMAX: the largest global node number used in this problem
- NODPRS: the global node numbers of an edge with prescribed loads
- NSURF: the number of surfaces with external loads
- NSCONV: the number of surfaces with convective losses
- NQSURF: the number of surfaces with external flux loads
- NST, NST2: counters
- NUSTEP: a flag for a new time step. NUSTEP=0 if TOLERA exceeded
- SCHEME; the scheme number from which the number of Gauss points may be chosen
- TEMP: a temporary value

.....

•

• DICTIONARY for real variables •

•

.....

• AA: first term in equation for temperature dependant conductivity
 • in isotropic material ($DMOD = AA + BB*TG2$)

• AMBT: ambient temperature •

• AREA: the determinant of the jacobian at this point •

• B: the gradient matrix; the derivative of P w/r to x & y •

• BB: second term in equation for temperature dependant conductivity

• BV: B stored as a vector to reduce cost of addressing •

• C: the product of CP times P(NOD) is required for CAPV •

• CAP: the thermal capacitance matrix •

• CAPV: CAP stored as a vector •

• COND: the conductivity matrix •

• CONDV: COND stored as a vector •

• CONDG: the conductivity at a Gauss point •

• COORD: the global coordinates of all nodes •

• CP: the heat capacity at a Gauss point •

• DB: the product of D times B •

• DENSITY: the density of this material •

• DERG: the derivatives of the shape functions w/r to global coord •

• DERL: the shape function derivatives w/r to local coordinates •

• DMOD1: the thermal conductivity in the x direction •

• DMOD2: the thermal conductivity in the y direction •

• DTIME: the time increment for this step •

• DVOLUM: the 'volume' times the Gaussian weight factors at this point •

• EFLOAD: the effective element load vector submitted to the solver •

• ELSTIF: the effective element stiffness matrix sent to the solver •
 • in vector form as upper right triangular. •

• ELXYZT: the global coordinate list for this element; T can be thickness •

• EMIS: emissivity for radiation boundary conditions on a surface •

• EMISIV: emissivity for radiation boundary conditions on an edge •

• EXLOAD1: the external load vector for step P+1 •

• EXLOAD2: the external load vector for step P+2 •

• EXLOAD3: the external load vector for step P+3 •

• FACTOR: the proportion of total load applied in this increment •

• GASH: a temporary storage location •

• GFLUX: the summed and interpolated flux at a Gauss point •

• H: the convective heat transfer coefficient from VPROP •

• HECONV: the convective heat transfer coefficient for an edge •

• H1: the nodal enthalpy values for step P+1. A vector. •

• H2: the nodal enthalpy values for step P+2. A vector. •

• HG1: enthalpy at the Gauss point for step P+1 •

- * HG2: enthalpy at the Gauss point for step P+2 *
- * HRAD: 'heat transfer' coefficient for radiation *
- * JAC: the jacobian for the transformation from local to global *
- * JINV: the inverse of JAC *
- * P: the shape functions or interpolating polynomials *
- * PEFLOAD: the effective load vector from previous iteration *
- * PEXLOAD2: EXLOAD2 from the previous iteration *
- * POIN: the global coordinates of this Gauss point *
- * QEDGE: the nodal values of a distributed flux. *
- * QFLUX: *
- * QSURF: the nodal values of a surface flux distribution. *
- * STEFAN: Stefan-Boltzman constant *
- * T1: the nodal temperatures for step P+1 *
- * T2: the nodal temperatures for step P+2 *
- * T3: the nodal temperatures for step P+3 *
- * TAMBT: ambient temperature for a particular edge *
- * TAMB: ambient temperature in degrees Kelvin *
- * TEMPER: the initial temperature read from the property list, VPROP *
- * TFACTOR: sum of FACTOR to this step; is the total load factor *
- * TG1: the temperature at this Gauss point at step P+1 *
- * TG2: the temperature at this Gauss point at step P+1 *
- * TGABS: temperature at Gauss point in degrees Kelvin *
- * THETA: a factor for explicit, implicit, Galerkin, etc. *
- * THETA1: 1-THETA *
- * THICK: the thickness of this element *
- * TIME: the total time to this step *
- * VPROP: a list of element properties. It depends on element type *
- * PXVOL: $P(NOD)*DVOLUM$ used to reduce the number of multiplies *
- * WEIGP: the vector of Gaussian weight factors *
- * WSIDE: $NSCONV*H*P(NOD)*DVOLUM$ - surface convection term for COND *
- * XGAUS: the local coordinates of the Gauss points *
- * YGAUS: the local y coordinate of the Gauss points *
- * XITA: the local coordinates of the Gauss points *

*** Zero the capacitance matrix or vector. Also the conductance matrix

IF (DTIME.EQ.0.) STOP'DTIME CANNOT BE ZERO IN FTSD'

DO 10 I = 1,NODLSQ,1

CONDV(I) = 0.0

CAPV (I) = 0.0

10 CONTINUE

DO 20 NOD=1,NODZ,1

```

        EXLOAD2(NOD)=0.0
        EXLOAD3(NOD)=0.0
20      CONTINUE
***    extract all relevant material properties from VPROP(I,LPOP)
.
        THETA1 = 1.0 - THETA
        VSCHEME =VPROP(1,LPOP)
        DMOD1=VPROP(2,LPOP)
        DMOD2=VPROP(3,LPOP)
        DMOD3=VPROP(4,LPOP)
***    if isotropic, the user may neglect to input DMOD2 and DMOD3. So default to DMOD1
        IF (DMOD2. EQ .0.0) DMOD2=DMOD1
        IF (DMOD3. EQ .0.0) DMOD3=DMOD1
        TEMPER = VPROP(5,LPOP)
        CP=VPROP(6,LPOP)
        SCHEME = VPROP(7,LPOP)
        MATNO=VPROP(8,LPOP)
        AA = VPROP(9,LPOP)
        BB = VPROP(10,LPOP)
        DELTAV = VPROP(11,LPOP)
        IF (NITERA.EQ.1.AND.NINCRE.EQ.1)WRITE (6,940) VSCHEME, DMOD1,
1      DMOD2, DMOD3, SCHEME, TEMPER,CP,MATNO,AA,BB,DELTAV
        IF ((NITERA. GT .1). AND (NUSTEP. EQ .0)) WRITE (6,900)
.
***    Now the weights and distances for gaussian integration
.
.
.
        The global coordinates for the nodes in this element are collected into
        ELXYZT. Coordinates not input as data default to 5.5E+55 in NURSE.
        For midedge nodes, MIDSID interpolates linearly for the coordinates.
        For corner nodes MIDSID declares a fatal error and stops.
.
.
        The first step is different,NITERA=1 and NINCRE=1 for the
        first step and equal to 1 for all subsequent steps.
        for the first step initialize nodal temperatures with VPROP(8,LPOP)
.
        IF (NITERA.EQ.1.AND.NINCRE.EQ.1) THEN
.
*On the first entry, interpolate for any unassigned midedge nodal
*coordinates. An unassigned coordinate for a corner node is a fatal
*error.
.
        MIDNOD = 12
        CALL MIDSID(COORD,LNOMAX,LNODS,LNODZ,3,NEL,NELZ,ELXYZT,NDIM,
1      NODMAX,MIDNOD)

```

```

DO 30 NOD = 1, LNODZ
  T3(NOD) = TEMPER
  T2(NOD) = TEMPER
  T1(NOD) = TEMPER
30  CONTINUE
.
.
**  For the first step, zero the previous external load vectors
.
DO 40 NOD = 1, LNODZ, 1
  EXLOAD1(NOD) = 0.0
40  CONTINUE

ELSE
.
.
.
DO 50 NOD = 1, LNODZ
  NODE = IABS(LNODS(NOD, NEL))
  DO 45 I = 1, NDIM
    ELXYZT(NOD, I) = COORD(NODE, I)
45  CONTINUE
50  CONTINUE
.
.
.
Read the historical values of temperature and external load
in preparation for a new step, if NUSTEP = 1. In any case T3
and EXLOAD2 are meaningless here and read only to avoid read
errors.
.
IF (NITERA .EQ. 1) THEN
  READ (8, KEY=NEL) L, (T3(I), I=1, L), (T2(I), I=1, L), (T1(I), I=1, L),
& (EXLOAD1(I), I=1, L)
  DO 60 NOD = 1, LNODZ
    T2(NOD) = T1(NOD)
60  CONTINUE
.
ELSE
.
.
or iterate last step, if NUSTEP = 0.
.
  READ (8, KEY=NEL) L, (T3(I), I=1, L), (T2(I), I=1, L), (T1(I), I=1, L),
& (EXLOADS(I), I=1, L), (PEXLOAD2(I), I=1, L), (EXLOAD1(I),
& I=1, L), (PEFLOAD(I), I=1, L), GASH
  DO 70 NOD = 1, LNODZ, 1
    EXLOAD2(NOD) = 0.0
70  T2(NOD) = THETA * T3(NOD) + THETA1 * T1(NOD)
  WRITE (8, 910) TIME, NEL, ((IABS(LNODS(I, NEL)), T2(I), I=1,
1    LNODZ))

```



```

      IF (L.NE.LVABZ) CALL DOCTOR
      ENDIF
    END IF

    *
    *
    *
    * If NEL=1 & INCRE=NINCRE this element has element level loads
    *
    *   On the first entry, load data must be read from DCB '5
    *   which is the main input file. If this load data is for
    *   this element (NEL) then we write it to DCB 9 for subsequent
    *   iterations.
    *
    IF (NITERA.EQ.1) THEN
      READ (UNIT=5,END=270, FMT=930) I, NSURF, NQVOL,NPOINT,INCRE
      IF (I.LT. NEL AND INCRE.LE. NINCRE)
1      CALL LOADERROR (1,I,INCRE,NEL,NINCRE)
      IF (INCRE.LT. NINCRE AND. I.LE. NEL)
1      CALL LOADERROR (1,I,INCRE,NEL,NINCRE)
      IF (I.EQ. NEL AND INCRE.EQ.NINCRE) THEN
        WRITE (UNIT=9) I,NSURF,NQVOL,NPOINT,INCRE
      ELSE
        BACKSPACE 5
        NQVOL=0
        GO TO 280
      END IF
    ELSE
      READ (UNIT=9,END=270) I,NSURF,NQVOL,NPOINT,INCRE
      IF (I.EQ.NEL AND INCRE.EQ.NINCRE) THEN
        DO NOTHING
      ELSE
        BACKSPACE 9
        GO TO 280
      ENDIF
    END IF

    *
    * Add nodal loads to external load vector
    *
    IF (NPOINT.EQ.0) GO TO 110
    IF (NITERA.EQ.1) THEN
      DO 72 I = 1,NPOINT
72  READ (UNIT=5,FMT=930)(NODPRS(I),QNOD(I))
      DO 100 NOD=1,NPOINT
      DO 80 N=1,LNODZ

```

```

      IF (NODPRS(NOD).EQ.(LABS(LNODS(N,NEL))))GO TO 90
80      CONTINUE
90      EXLOAD3(N)=EXLOAD3(N)-QNOD(NOD)
100     CONTINUE
      END IF
110     CONTINUE
      *
      IF (NSURF. EQ .0) GO TO 280
      *
      *
      *           LOOP OVER EACH LOADED FACE
      *
      DO 260 ISURF=1,NSURF
      *
      *   Read element number, flags for edge convection and fluxes, the
      *   face node numbers and the flux amplitudes at the nodes.
      *
      *
      IF (NITERA.EQ.1) THEN
        READ (6,930) I, IQSURF, HECONV, EMISIV, TAMBT,(NODPRS(NOD),
1          NOD=1,8), (QSURF(NOD), NOD=1,8)
      IF (I. NE. NEL) CALL LOADERERROR (2,I,INCR,NEL,NINCR)
        WRITE (6,920) I,(NODPRS(NOD),NOD=1,8),EMISIV,TAMBT,HECONV,
1          IQSURF,(QSURF(NOD),NOD=1,8)
        WRITE (UNIT=9) I, IQSURF, HECONV, EMISIV, TAMBT,(NODPRS(NOD)
1          , NOD=1,8), (QSURF(NOD),NOD=1,8)
      ELSE
        READ (UNIT=9) I, IQSURF, HECONV, EMISIV, TAMBT,(NODPRS(NOD),
1          NOD=1,8), (QSURF(NOD), NOD=1,8)
        WRITE (6,920) I,(NODPRS(NOD),NOD=1,8),EMISIV,TAMBT,HECONV,
1          IQSURF, (QSURF(NOD),NOD=1,8)
      *   IF (I. NE. NEL) CALL DOCTOR ('Load cards shuffled')
      *   END IF
      *
      *
      *   Identify local node numbers for input nodes on face.
      *
      DO 130 INODE = 1,8,1
        DO 135 KNODE = 1,20,1
          IF (NODPRS(INODE) .EQ. LABS(LNODS(KNODE,NEL)))
            ILOCNOD(INODE) = KNODE
135      CONTINUE
130      CONTINUE
      *
      *
      *

```

```

.....
*
*   enter loop for numerical integration along this edge for contributions
*   to the stiffness matrix and the external load vector
*
.....
*
*
.....
*
*** contributions from distributed surface flux to the external load vector
*
.....
*
CALL FACFID(LOCNOD,FACXYZ,QSURF,MVDFLUX,FACID,ELEMFACE,
1   ELXYZT,NODPRS,NDIM,LINCRE,NEL,NINCRE)
CALL GAUSS2D(SCHEME,NOGAUS,XGAUS,YGAUS,WEIGP)
DO 250 IGAUS = 1,NOGAUS
    XITA(1) = XGAUS(IGAUS)
    XITA(2) = YGAUS(IGAUS)
CALL DAREA(NEL,XITA,P,DERL,AREA,JACV,FACXYZ)

    AREA = AREA*WEIGP(IGAUS)
    IF (IQSURF .EQ. 0) GO TO 200
    GFLUX=0.0
    DO 180 NOD=1,NODSRF
        GFLUX=GFLUX+MVDFLUX(NOD)*P(NOD)
180    CONTINUE

*
*   Add equivalent nodal values of distributed surface loads to
*   external load vector.
*       I=0
    DO 190 I = 1,NODSRF,I
        N = LOCNOD(I)
        EXLOADS(N) = EXLOADS(N) - P(I)*GFLUX*AREA
190    CONTINUE

*
.....
*
*   convection or radiation loading section
*
.....
200    IF ((HECONV .EQ. 0). AND. (EMISIV. EQ. 0.0)) GO TO 250

```

```

*
*   For convection and radiation, calculate the contributions
*   to the stiffness matrix, COND, and the load vector, EXLOAD.
*   Because it's symmetric, only the upper right triangular portion
*   is stored. To minimise addressing costs, it is stored as a
*   vector, CONDV, that runs down the columns from 1 to 20 stopping
*   at the diagonal. JDIAG is the number of entries in this column
*
*
*   Linearise the radiation term to a "heat transfer coefficient"
*   See Myers, "Analytic Methods in Conduction Heat Transfer" page 303
*   Interpolate for the Gauss point temperature
      NOD=0
      TG2=0.0
      DO 210 NOD = 1,8,1
        N = LOCNOD(NOD)
210      TG2 = TG2 + T2(N)*P(NOD)
*
*   Use absolute temperatures
*
      TAMB = TAMB1
      TGABS = TG2
      H=HECONV+EMISIV*STEFAN*(TGABS*TGABS+TAMB*TAMB)*(TGABS+TAMB)
*
      DO 230 NOD = 1,8,1
        PXVOL(NOD) = P(NOD)*AREA
        J = LOCNOD(NOD)
        DO 220 K = 1,8,1
          I = LOCNOD(K)
          IF (I .LT. J) GO TO 220
          PLACE = 20*(I-1)+J
          CONDV(PLACE) = CONDV(PLACE) + P(K)*PXVOL(NOD)*H
220        CONTINUE
230      CONTINUE
*
***   convection and radiation adds to the external load vector
*
      NOD = 0
      DO 240 NOD = 1,8,1
        K = LOCNOD(NOD)
        EXLOAD2(K) = EXLOAD2(K) - H*TAMB*PXVOL(NOD)
240      CONTINUE
250      CONTINUE
260      CONTINUE
      GO TO 280

```

```

270   NQVOL=0
280   CONTINUE
.
.....
.
**** NOW PERFORM ALL VOLUME INTEGRALS OVER THE CURRENT ELEMENT
.
.....
.
      IF ((MATNO. GT .0). AND. (MATNO .LT. 100)) THEN
        CALL NTHALPY (H1, LNODZ, MATNO, T1, CPN, CONDN)
        CALL NTHALPY (H2, LNODZ, MATNO, T2, CPN, CONDN)
      END IF
.
.
      IF (NQVOL .EQ. 0) GO TO 300
      IF (NITERA.EQ.1) THEN
        READ (5,930) I, (QVOL(NOD), NOD=1, NQVOL)
        IF (NEL. NE .1) CALL LOADERROR (3, I, INCRE, NEL, NINCRE)
      ENDIF
.
300   CONTINUE
.
.   Now the weights and distances for Gaussian integration
CALL GAUSS3D(VSCHEME, NGAUS, XGAUS, YGAUS, ZGAUS, WEIGP)
DO 400 IGAUS=1, NGAUS
  XITA(1)=XGAUS(IGAUS)
  XITA(2)=YGAUS(IGAUS)
  XITA(3)=ZGAUS(IGAUS)
.
.   evaluate shape functions, their derivatives, the determinant of
.   the jacobian, is COMMON /CBRIK/.
.
      CALL BRICK(LNODZ, LVABZ, LNOMAX, NEL, NELZ, XITA,
1      P, ELXYZT, DERG, VOLUME)
      DVOLUM = VOLUME*WEIGP(IGAUS)
.
.....
.   GET THERMAL PROPERTIES
.   .....
      IF (MATNO. GT .0) THEN
.
.   Interpolate for temperature dependent conductivity, capacitance
.   and enthalpy at the Gauss point from their nodal values

```

• that were computed in NTHALPY from the nodal temperatures.
 • An alternative is to interpolate only the Gauss point
 • temperature from the nodal values, and then compute the
 • thermal properties from the Gauss point temperatures with
 • TH PROP. The latter is probably cheaper but less accurate.
 •

```

      HG1=0.0
      HG2=0.0
      TG1=0.0
      TG2=0.0
      IF (MATNO .LT. 100) THEN
        CONDG = 0.0
        CP = 0.0
        DO 310 NOD=1, LNODZ
          TG1=TG1+T1(NOD)*P(NOD)
          TG2=TG2+T2(NOD)*P(NOD)
          HG1=HG1+H1(NOD)*P(NOD)
          HG2=HG2+H2(NOD)*P(NOD)
          CONDG = CONDG + CONDN(NOD)*P(NOD)
          CP = CP + CPN(NOD)*P(NOD)
310      CONTINUE
        CALL TH PROP (MATNO, TG1, TG2, CP, CONDG)
        IF (ABS(TG2-TG1). GT..01) CP=(HG2-HG1)/(TG2-TG1)
        ELSE
          DO 320 NOD = 1, LNODZ
            TG2 = TG2 + T2(NOD)*P(NOD)
320      CONTINUE
        Evaluate linearly temperature dependant value for conductivity
        for MATNO = 100

        IF (MATNO .EQ. 100) THEN
          CONDG = AA + BB*TG2
          ELSE

        Evaluate volume flux from resistive heating for MATNO > 100

        CALL QVOLCALC(MATNO, TG2, DELTAV, QVOL)
        DMOD1X = DMOD1
        DMOD2X = DMOD2
        DMOD3X = DMOD3
        ENDIF
        ENDIF
        DMOD1X=CONDG
        DMOD2X=CONDG
  
```

```

      DMOD3X=CONDG
    ELSE
      DMOD1X=DMOD1
      DMOD2X=DMOD2
      DMOD3X=DMOD3

```

```

    END IF

```

```

    .....

```

```

    **** CREATE MODULUS MATRIX, ALREADY MULTIPLIED BY ITS INTEGRATING FACTOR

```

```

      DMOD1X = DMOD1X * DVOLUM
      DMOD2X = DMOD2X * DVOLUM
      DMOD3X = DMOD3X*DVOLUM
      CP2=CP*DVOLUM/DTIME

```

```

    COMPUTE EQUIVALENT LOAD AT GAUSS POINT FOR DISTRIBUTED VOLUME FLUX

```

```

      IF (NQVOL .EQ. 0) THEN
        GO TO 350
      ELSE
        IF (NQVOL .EQ. LNODZ) THEN
          GFLUX = 0
          DO 340 NOD=1, NQVOL

```

```

          GFLUX=GFLUX+QVOL(NOD)*P(NOD)
340      CONTINUE
          ELSE
            IF (NQVOL .EQ. 1) GFLUX = QVOL(1)
            ENDDIF
            ENDDIF

```

```

    * The gradient matrix B was equivalenced to DERG, the global
    * derivatives of the shape functions, P. Next DB = D times B.
    * Then B D B is computed column by column to minimize multiplication

```

```

350      NST = 0
      DO 390 NOD=1, LNODZ
        PXVOL(NOD)=P(NOD)*DVOLUM

        B(1,NOD) = DERG(NOD,1)
        B(2,NOD) = DERG(NOD,2)
        B(3,NOD) = DERG(NOD,3)
        DB(1) = DERG(NOD,1)*DMOD1X
        DB(2) = DERG(NOD,2)*DMOD2X

```

```

      DB(3) = DERG(NOD,3)*DMOD3X
      C=CP2*P(NOD)
      COMPUTE THE NEXT COLUMN OF THE CONDUCTIVITY AND CAPACITANCE MATRIX

      JDEL = 0
      DO 370 K=1,NOD
        GASH=BV(JDEL+1)*DB(1) + BV(JDEL+2)*DB(2)
        +BV(JDEL+3)*DB(3)
        CONDV(NST+K)=CONDV(NST+K)+GASH
360      CAPV(NST+K) = CAPV(NST+K) + C*P(K)
        JDEL = JDEL + 3
370      CONTINUE

      *** contribution to nodal loads from volume flux

      IF((NQVOL .GT. 0).AND.(NITERA.EQ.1))EXLOAD3(NOD)=
1      EXLOAD3(NOD)-GFLUX*PXVOL(NOD)

380      NST = NST + NODL
390      CONTINUE

400      CONTINUE

      The upper right portions of CAP and COND are complete. Now
      Fill lower left triangle of COND and CAP by symmetry
      The full matrix is used only to simplify coding for ELSTIF below

      DO 410 I=1,LNODZ
        K=I+1
        DO 410 J=K,LNODZ
          CAP(J,I) = CAP(I,J)
410      COND(J,I) = COND(I,J)

      The effective load vector sums contributions from the current
      external load vector (EXLOAD2); the previous external load vector
      and a contribution from the conductivity (COND) and capacitance
      (CAP) matrices. EFLOAD is sent to FRONT. Theta is a weight factor
      appropriate for explicit, implicit, Crank-Nicholson, Galerkin, etc.

```



```

      IF (NITERA.EQ.1) THEN
        NST=0
        DO 430 I=1,LNODZ
          GASH=0.0
          DO 420 J=1,LNODZ
            GASH=GASH+T1(J)*(CAPV(NST+J)-THETA1*CONDV(NST+J))
420          CONTINUE
          EFLOAD(I)= GASH-THETA*EXLOAD3(I)-THETA1*EXLOAD1(I)- EXLOAD2(I)
          IF (NITERA.EQ.1)FLUXIN=FLUXIN+EXLOAD3(I)*DTIME
430          NST=NST+NODL
          ELSE
            DO 435 I=1,LNODZ
              EFLOAD(I)=PEFLOAD(I)+PEXLOAD2(I)-
                EXLOAD2(I)
435            CONTINUE
          END IF
        .
        . store data needed for next iteration, increment or restart
        . on a random access disc file keyed to the element number
          L=LNODZ
          WRITE (8,KEY=NEL) L, (T3(I),I=1,L), (T2(I),I=1,L), (T1(I), I=1,L),
            & (EXLOAD3(I),I=1,L), (EXLOAD2(I), I=1,L), (EXLOAD1(I),
            & I=1,L), (EFLOAD(I), I=1,L), THETA "
        .
        . form ELSTIF, an upper right triangular matrix stored as a vector.
        . ELSTIF is checked for zero or negative diagonal entries in ELFILE
        . before being sent to FRONT the solver.
        .
          NST=0
          NST2=0
          JDIAG=0
          DO 450 I =1,LNODZ
            JDIAG=JDIAG+1
            DO 440 K=1,JDIAG
440            ELSTIF(NST+K)=THETA*CONDV(NST2+K)+CAPV(NST2+K)
              NST=NST+JDIAG
              NST2=NST2+NODL
450            CONTINUE
          .
          RETURN
900          FORMAT ("WARNING* Tolerance was exceeded in previous
            1          'step.' )
910          FORMAT ('TEMPERATURES AT TIME=F10.8,FOR ELEMENT 'I3,/,
            1          ' NODE',10X,'T2',/,50(15,1X,F10.5,/))

```

```

920  FORMAT ('ELEMENT NO. = ',I3,
1      ' ,LIST OF LOADED FACES AND APPLIED LOADS-'
2      ',8(1X,I3),/ ',EMISIVITY RADIATION BOUNDARY CONDITION '
3      'ON THIS FACE= ',F10.5,/,'AMBIENT TEMP FOR THIS FACE= ',
4      'F10.5,/
5      'HEAT TRANSFER COEFFICIENT = ',F10.5,/
6      'FLAG FOR FLUX LOADING =',I2,/
7      'NODAL VALUES FOR FLUX DISTRIBUTION-',4(1X,F18.6),/)
930  FORMAT(30G)
940  FORMAT('VOLUME SCHEME = ',I3,/
2      'X CONDUCTION COEFFICIENT = ',F10.5,/
3      'Y CONDUCTION COEFFICIENT = ',F10.5,/
4      'Z CONDUCTION COEFFICIENT = ',F10.5,/
B      'SURFACE SCHEME = ',I3,/
6      'INITIAL TEMPERATURE = ',F10.5,/
8      'SPECIFIC HEAT = ',F18.7,/
9      'MATERIAL NUMBER = ',I2,/
A      'AA = ',F10.5,/
C      'BB = ',F10.5,/
D      'VOLTAGE GRADIENT = ',F10.5,/

```

END

*Subroutine to identify the nodes on a given face.

```

SUBROUTINE FACFID (LOGNOD,FACXYZ,QSURF,MVDFLUX,FACID,ELEMFAC,
1      ELXYZT,NODPRS,NDIM,LINCRE,NEL,NINCRE)

```

```

.....
*   This subroutine identifies the loaded face by computing a
*   number - FACID, then shifts the surface load vector - QSURF
*   to match the internal order of the nodes on the face - stored
*   as row FACID in ELEMFAC.
.....

```

DICTIONARY FOR VARIABLES

Real Variables

FACXYZ : the global coordinates (x,y,z) for the nodes on the face.

QSURF : the input nodal values of the distributed load

MVDFLUX: the shifted values of QSURF

Integer Variables

* ELEMFACE : an array identifying the local node numbers
 * for each face.
 * LOCNOD : local node numbers on a face in the order which
 * user input loads.
 * FACID : number identifying the loaded face;
 * corresponds to a row number in ELEMFACE.
 *

.....

```

      REAL
1FACXYZ(8,3),QSURF(8),MVDFLUX(8),ELXYZT(20,3),NODPRS(20)

      INTEGER
1IROW,ICOL,LOCNOD(8),ELEMFACE(8,8),KORECT,ECOL,FACID,
  2       KNOD,NDIM,L

      LOGICAL
1HERE

      DO 30 IROW = 1,8,1
        DO 20 ICOL = 1,8,1
          IF (LOCNOD(1) .EQ. ELEMFACE(IROW,ICOL)) THEN
            *       KORECT = 1
            DO 10 C = 2,8,1
              HERE = .FALSE.
              DO 5 ECOL = 1,8,1
                IF (LOCNOD(C) .EQ. ELEMFACE(IROW,ECOL)) THEN
                  HERE = .TRUE.
                  KORECT = KORECT + 1
                  GO TO 8
                ENDIF
              CONTINUE
            5       IF (KORECT .GE. 8) GO TO 40
            8       IF (.NOT. HERE) GO TO 20
            10       CONTINUE
            10       ENDIF
            20       CONTINUE
            30       CONTINUE

          40       FACID = IROW
          IF (KORECT .LT. 8) CALL LOADERROR (4,LINCRE,NEL,NINCRE)
          IF (ECOL .GE. 4) THEN
            ECOL = ECOL - 3
          ELSE
            ECOL = ECOL + 5
          
```

```

      ENDIF
*ECOL now identifies position in ROW of ELEMFACE that user picked
*as initial node.
      DO 50 L = 1,8,1
        MVDFLUX(ECOL) = QSURF(L)
        ECOL = ECOL + 1
        IF (ECOL .GT. 8) ECOL = ECOL - 8
*      Identify coordinates of nodes on face.

        KNOD = ELEMFACE(FACID,L)
        DO 155 COL = 1,NDIM,1
          FACXYZ(L,COL) = ELXYZT(KNOD,COL)
155      CONTINUE
        LOCNOD(L) = KNOD
50      CONTINUE
*MVDFLUX now contains the input distributed flux in the order
*which nodes are found in ELEMFACE.
      RETURN
      END
      SUBROUTINE FACLOD(NEL,XITA,P,DERL,AREA,JAC,ELXYZ)
.....
*      Compute two rows of the Jacobian. They are the vectors in the
*      XITA(1) and XITA(2) direction. Their cross product is the
*      vector area needed for area integrals over the face of a 20 node
*      brick. This is needed for surface loads such as flux or
*      convective thermal loads.
*
*      The algorithm:
*      1. Compute the basis functions and their derivatives at this Gauss
*      point XITA.
*      2. Compute two rows of Jacobian JAC(2,3). The entries are the
*      derivatives. DXDR DYDR DZDR = JAC(1,1) JAC(1,2) JAC(1,3)
*      DXDS DYDS DZDS = JAC(2,1) JAC(2,2) JAC(2,3)
*
*      = JV(1)   JV(3)   JV(5)
*      = JV(2)   JV(4)   JV(6)
.....
*      Dictionary for variables
.....
*
*      Real variables
*
*      AREA:      determinant of the Jacobian at this point
*      DERL:      the shape function derivatives w/r to local coordinates
*      ELXYZ:      the global coordinate list for the nodes on this face

```

* JAC: the Jacobian for the transformation from local
 * to global coordinates
 * SUM: cumulative value of each entry in the Jacobian
 * P: the shape functions or interpolating polynomials
 * XITA: the local coordinates of the Gauss points (R,S)
 * S,T,S2,T2,SS,TT,ST,SST,STT,ST2:
 * the shape function factors.

Integer variables

* COLUMN: column of the Jacobian
 * ELPOS: position pointer
 * NEL: number of elements in the entire problem
 * NODE: local node numbers for this face
 * ROW: row of the Jacobian

.....
 IMPLICIT LOGICAL (A-Z,R)

REAL

1 AREA, DERL(2,8), ELXYZ(24), JAC(2,3), SUM,
 2 P(8), S, S2, SS, SST, ST, ST2, STT, T, T2, TT, XITA(2)

INTEGER

1 COLUMN, ELPOS, NEL, NODE, ROW

S=XITA(1)

T=XITA(2)

Evaluate shape function factors

S2=S*2.0

T2=T*2.0

SS=S*S

TT=T*T

ST=S*T

SST=SS*T

STT=S*TT

ST2=ST*2.0

The basis functions for the 8 node quad at the point (R,S)

$P(1) = (-1 + ST + SS + TT - SST - STT) * .25$

$P(2) = (1.0 - T - SS + SST) / 2.0$

$P(3) = (-1.0 - ST + SS + TT - SST + STT) / 4.0$

$P(4) = (1.0 + 2 - TT - STT) / 2.0$

```

P(5) = (-1.0 + ST + SS + TT + SST + STT) / 4.0
P(6) = (1.0 + T - SS - SST) / 2.0
P(7) = (-1.0 - ST + SS + TT + SST - STT) * .25
P(8) = (1.0 - S - TT + STT) * .50

```

The derivatives of the basis functions w/r to R.

```

DERL(1,1) = (T + S2 - ST2 - TT) * .25
DERL(1,2) = -S + ST
DERL(1,3) = (-T + S2 - ST2 + TT) / 4.0
DERL(1,4) = (1.0 - TT) / 2.0
DERL(1,5) = (T + S2 + ST2 + TT) / 4.0
DERL(1,6) = -S - ST
DERL(1,7) = (-T + S2 + ST2 - TT) * .25
DERL(1,8) = (-1.0 + TT) * .50

```

The derivatives of the basis functions w/r to S.

```

DERL(2,1) = (S + T2 - SS - ST2) * .25
DERL(2,2) = (1.0 + SS) / 2.0
DERL(2,3) = (-S + T2 - SS + ST2) / 4.0
DERL(2,4) = -T - ST
DERL(2,5) = (S + T2 + SS + ST2) / 4.0
DERL(2,6) = (1.0 - SS) / 2.0
DERL(2,7) = (-S + T2 + SS - ST2) * .25
DERL(2,8) = -T + ST

```

.....

Compute two rows of the Jacobian. They are the vectors in the XITA(1) and XITA(2) direction. Their cross product is the vector area needed for area integrals over the face of a brick. This is needed for surface loads such as flux or convective thermal loads.

.....

```

DO 30 ROW = 1, 2
  ELPOS = 0
  DO 20 COLUMN = 1, 3
    SUM = 0.
    DO 10 NODE = 1, 8
      ELPOS = ELPOS + 1
      SUM = SUM + DERL(ROW, NODE) * ELXYZ(ELPOS)
    CONTINUE
  JAC(ROW, COLUMN) = SUM
  CONTINUE
CONTINUE
RETURN

```

```

      END
      SUBROUTINE DAREA(NEL,XITA,P,DERL,AREA,JACV,ELXYZ)
      .....
      *   This subroutine computes the cross product of the
      *   vectors in the XITA(1) and XITA(2) directions. This
      *   value is the vector area needed for area integrals
      *   over the face of a brick. This is needed for surface
      *   loads such as flux or convective thermal loads.
      .....
      *   JAC(1,1) JAC(1,2) JAC(1,3) = JACV(1) JACV(3) JACV(5)
      *   JAC(2,1) JAC(2,2) JAC(2,3) = JACV(2) JACV(4) JACV(6)
      .....
      REAL XITA(2), P(8), DERL(2,8), AREA, ELXYZ(24), JACV(6)
      *
      INTEGER NEL
      *
      CALL FACLOD(NEL,XITA,P,DERL,AREA,JACV,ELXYZ)
      *
      AREA=SQRT((JACV(3)*JACV(6)-JACV(4)*JACV(5))**2
      1      +(JACV(5)*JACV(2)-JACV(1)*JACV(6))**2
      2      +(JACV(1)*JACV(4)-JACV(3)*JACV(2))**2)
      RETURN
      END
      *
      *
      SUBROUTINE QVOLCALC (MATNO,TG2,DELTAV,QVOL)
      *
      *   Given the Gauss point temperature and the voltage gradient,
      *   calculate the resistive heat generated. Resistivity is
      *   stored in a matrix by MATNO and temperature. eg. R(1,3)
      *   is the resistivity for material 101 at 300 deg. C.
      .....
      *
      *   DICTIONARY OF VARIABLES
      *
      *   DELTAV : the voltage difference across the element
      *   I      : TG2 rounded down to nearest multiple of 100
      *   K      : MATNO / 100; to find row in R.
      *   MATNO  : material identification number (101 for 1025 steel)
      *   N      : the Gauss point temperature divided by 100
      *   QVOL   : the heat generated
      *   R      : the matrix of resistivity values for interpolation
      *   RES    : the interpolated value for resistivity
      *   TG2    : the interpolated Gauss point temperature
      *

```

.....

REAL

1 TG2, DELTAV, QVOL, R(1,0:13), RES

INTEGER

1 MATNO, K, N, I

Resistivity at given temp for 0.23% Carbon steel (MATNO = 101)

DATA R/0.153, 0.219, 0.292, 0.394, 0.487, 0.647, 0.758,
1 0.925, 1.094, 1.136, 1.167, 1.194, 1.219, 1.239/

Linear Interpolation for R

K = MATNO - 100

N = TG2/100.0

I = N*100

RES = R(K,N) + (R(K,N+1) - R(K,N+K)) * (TG2-I)/100

QVOL = DELTAV*DELTAV/RES

RETURN

END

SUBROUTINE LOADERROR (ERRNO,I,INCRE,NEL,NINCRE)

PURPOSE : to identify location of error in element level
loading and output appropriate error message.

INTEGER

1 ERRNO,I,INCRE,NEL,NINCRE

.....

DICTIONARY FOR INTEGER VARIABLES

ERRNO : number identifying READ statement at which error
occured. 1 - flags for, element level loading

2 - distributed surface loading

3 - volume loading

1 : element number specified in input file


```

*   INCRE : increment number specified in input file
*
*   NEL   : actual element number
*
*   NINCRE: actual increment number
*
* .....
*
*       GO TO (10,20,30,40) ERRNO
*
10  WRITE (6,110) NEL,NINCRE,I,INCRE
    STOP
*
20  WRITE (6,120) INCRE, NEL,I
    STOP
*
30  WRITE (6,130) INCRE,NEL,I
    STOP
*
40  WRITE (6,140) NEL,NINCRE
    STOP
*
110 FORMAT('Error encountered reading flags for element level',/,
1      'loads. Element number or increment number out of',/,
2      'order.',/,
3      'Expected element  ',I4,' increment  ',I4,/,
4      'Read   element  ',I4,' increment  ',I4,/)
120 FORMAT('Error encountered reading values for distributed surface'
1      ',/,loads. Element number out of order for increment ',
2      'I4,/,Expected element  ',I4,'Read element  ',I4,/)
130 FORMAT('Error encountered reading values for distributed volume',
1      ',/,loads. Element number out of order for increment ',
2      'I4,/,Expected element  ',I4,' Read element  ',I4,/)
140 FORMAT('Nodes for distributed surface load do not lie on a face.'
1      ',/,For element number ',I4,' increment number ',I4,/)
END

```

END

0	6	H	0	9	H	8	4
---	---	---	---	---	---	---	---

FIN