NISTIR 88-3868

A Discrete Thermal Analysis Method (DTAM) for Building Energy Simulation with DTAM1 Users Manual

James Axley

Department of Architecture
Cornell University
Ithaca, New York 14853

October 1988

Prepared for
National Institute of Standards and Technology
U.S. Department of Energy

NISTIR 88-3868

A Discrete Thermal Analysis Method (DTAM) for Building Energy Simulation with DTAM1 Users Manual

James Axley

Department of Architecture
Cornell University
Ithaca, New York 14853

October 1988

National Bureau of Standards became the National Institute of Standards and Technology on August 23, 1988, when the Omnibus Trade and Competitiveness Act was signed. NIST retains all NBS functions. Its new programs will encourage improved use of technology by U.S. industry.

Prepared for
National Institute of Standards and Technology
U.S. Department of Energy

U.S. DEPARTMENT OF COMMERCE
C. William Verity, Secretary

NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY
Ernest Ambler, Director

ABSTRACT

This document includes a report that describes the theoretical basis of the program DTAM1 and a users manual for the program.

DTAM1 is a general purpose building energy simulation program that was developed to demonstrate an approach to building energy simulation based upon discrete analysis techniques, including, but not limited to, the Finite Element Method, used in other fields of physical simulation. It is the product of a first phase of development of *Discrete Thermal Element Analysis Techniques for Building Energy Simulation* that are expected to provide a means to unify existing building energy simulation theory.

DTAM1 provides a library of *discrete thermal elements*, that may be assembled to model thermal systems idealized to have constant material and heat transfer properties (i.e., linear idealizations), including:

- 1D two-node thermal resistance elements
- single-node lumped capacitance elements
- two-node fluid flow loop element
- 1D two-to-four node isoparametric conduction Finite Elements
- 2D four-node isoparametric conduction Finite Elements (planar and axisymmetric)

Equations defining a variable node mean radiant temperature element are also presented in the report.

Steady state and transient analysis capabilities are included. Temperature, heat flow rate, and convective boundary conditions may be modeled and system temperature variables may be constrained to be equal so that mixed assemblages of 1D and 2D elements may be employed.

KEY WORDS: building energy simulation, building dynamics, computer simulation techniques, discrete analysis techniques, discrete thermal elements, dynamic simulation, finite element analysis, DTAM1

CONTENTS

ABSTRACT 1
ACKNOWLEDGEMENTS 5

REPORT

1. Introduction 6

2. Theoretical Basis 6

2.1 Thermal Model 6
2.2 System Degrees of Freedom 7
2.3 General Form of Element Equations 7
2.4 System Equations 9
2.5 Boundary Conditions 11
2.6 Solution of System Equations 12
2.6.1 Steady Excitation
2.6.2 Harmonic Excitation
2.6.3 General Excitation
2.7 Specific Element Equations 14
2.7.1 Resistance Elements
2.7.2 Lumped Capacitance Elements
2.7.3 Flow Loop Elements
2.7.4 Transient Conduction Finite Elements
2.7.5 Variable-Node, Mean Radiant Temperature Element

3. Summary and Conclusions 24

4. References 26

DTAM1 USERS MANUAL

I. General Instructions 28

II. Preparation of Input File 30

A. Problem Initiation 32

B. Constraint Data Group 32

C. Coordinate Data Group 33

D. Element Data Groups 33

1) Resistance Elements
2) Lumped Capacitance Elements
3) Fluid Flow Loop Element
4) Isoparametric 1D Two To Four-node Conduction Finite Elements
5) Isoparametric 2D Four-node Conduction Finite Elements

E. Boundary Conditions Data Group 37

F. Convection Boundary Conditions Data Group 37

G. Solution Control 38

1) Steady State Excitation/Response
2) Harmonic Excitation/Response
3) General Excitation/Response - Predictor-Corrector Integration

H. Output Report Control 41

Appendix A - Free-Field Input Conventions of the CAL-SAP Software Development System

Appendix B - Example Problems

B1 Response of a Semi-infinite Slab to a Step Change of Heat Flux
B2 Residential Building Example
B3 Analysis of the Huron Building Thermal Bridge

Appendix C - Program Structure

Appendix D - FORTRAN 77 Source Code

ACKNOWLEDGEMENTS

Most of the development of the program DTAM1 was completed during the summer of 1985 at the National Bureau of Standards, Gaithersburg, Maryland. During that time the author was employed as a visiting researcher with the Thermal Analysis Group of the Building Physics Division. Dr. Kusuda, the Chief of the Building Physics Division, generously offered support for the development of the program and provided direction in a most adept and timely manner. Dr. Richard Grot and George Walton of the Thermal Analysis Group critically reviewed the progress of development of the program on a day-to-day basis. They provided invaluable insight that has led to a generalization and extension of the methods presented in this document currently under development. Daniel Clark of the Building Equipment Division introduced the author to the HVACSIM+ program and the theoretical basis of the HVAC component models it contains. This effort was also supported by the U.S. Department of Energy through an Interagency Agreement with Lawrence Berkeley Laboratory.

REPORT

1. Introduction

DTAM1 is a general purpose building energy simulation program that was developed to demonstrate an approach to building energy simulation based upon discrete analysis techniques, including, but not limited to, the Finite Element Method, used in other fields of physical system simulation. It is the product of a first phase of development of *Discrete Thermal Element Analysis Techniques for Building Energy Simulation* that are expected to provide a means to unify existing building energy simulation theory. Portions of the program are based on the Finite Element conduction analysis program, *HEAT* ,written by Robert Taylor, Department of Civil Engineering, U. C. , Berkeley [1] . DTAM1 was developed using the CAL-SAP library of subroutines developed by Ed Wilson, Department of Civil Engineering, U.C., Berkeley [2] .

DTAM1 provides a library of *discrete thermal elements*, that may be assembled to model thermal systems idealized to have constant material and heat transfer properties (i.e., linear idealizations), including:

- 1D two-node thermal resistance elements
- single-node lumped capacitance elements
- two-node fluid flow loop elements
- 1D two-to-four node isoparametric conduction Finite Elements
- 2D four-node isoparametric conduction Finite Elements (planar and axisymmetric)

Equations defining a variable node mean radiant temperature element are also presented in the report.

Static and dynamic analysis capabilities are included. Temperature, heat flow rate, and convective boundary conditions may be modeled and system temperature variables may be constrained to be equal so that mixed assemblages of 1D and 2D elements may be employed.

2. Theoretical Basis

2.1 Thermal Model

A building thermal *system* may be modeled as an assemblage of *discrete thermal elements* connected at discrete *system nodes*. Typically, these nodes will be associated with discrete points in the physical system, although this is not, in general, necessary. It will be shown that *system equations*, governing the behavior of the building thermal system, may be *assembled* from *element equations* that govern the behavior of individual elements.

A hypothetical example is illustrated below. Here, a simple house is idealized to model dynamic thermal exchanges between the house and its environment using resistance elements, lumped capacitance elements, and 1D and 2D conduction finite elements. Idealization of a thermal system as an assemblage of discrete thermal elements is facilitated by the use of diagrams such as the one illustrated below. The process of mathematical assembly of discrete element equations is analogous to the graphical assembly of element icons used in such diagrams and provides the analyst with an intuitive aide to the process of thermal idealization.

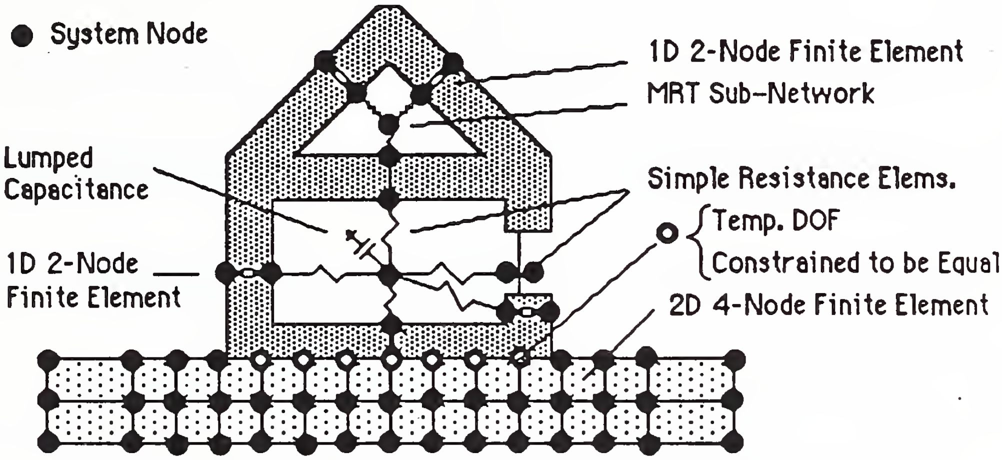


Fig. 1 Hypothetical Example of a Thermal Model for a Residence

2.2 System Degrees of Freedom

With each system node we associate a temperature degree of freedom (DOF), T_i , and a direct heat flow rate DOF, Q_i (i.e., heat flow rate directly from a source into the system node). These variables define the state of the system, at any time, and will be referred to as the *system degrees of freedom* . The system DOFs may be represented as vectors for symbolic mathematical manipulation:

$$\{T\} = \{ T_1 , T_2 , T_3 , \ldots , T_n \}^T$$; the *system temperature vector* (2.1)

$$\{Q\} = \{ Q_1 , Q_2 , Q_3 , \ldots , Q_n \}^T$$; the *system direct heat flow rate vector* (2.2)

for a system with n system nodes. The system temperature vector will be treated as the principal dependent variable.

2.3 General Form of Element Equations

One or more nodes may be associated with each element. With each element node, i, we associate an element temperature DOF, T^e_i, and net heat flow rate DOF, $q^e_{net\text{-}i}$ (i.e., heat flow rate from the element node into the element):

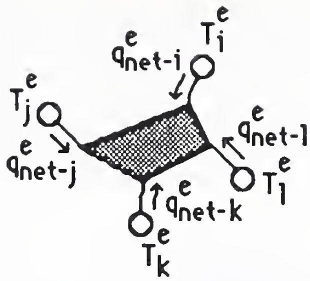


Fig. 2 Element DOFs

These element DOFs may also be represented as vectors for symbolic mathematical manipulation:

$\{T^e\} = \{ T^e{}_1 , T^e{}_2 , T^e{}_3 , \ldots , T^e{}_m \}^T$; the *element temperature vector* (2.3)

$\{q^e{}_{net}\} = \{ q^e{}_{net-1} , q^e{}_{net-2} , \ldots , q^e{}_{net-m} \}^T$; the *element net heat flow rate vector* (2.4)

for an element with m nodes.

In general, we attempt to describe the overall energy balance of an element by equations of the form:

$$\{q^e{}_{net}\} = [k^e]\{T^e\} + [c^e]\{dT^e/dt\} - \{q^e\} \tag{2.5}$$

where:

$[k^e]$ = matrix of element conductance transfer coefficients
= the *element conductance matrix*
= $[k^e(t, T^e)]$, in general

$[c^e]$ = matrix of element thermal capacity coefficients; the *element capacitance matrix*
= $[c^e(t,T^e)]$, in general

$\{q^e\}$ = vector of element-generated heat flow rates; the *element internal generation rate vector*

For elements employed in the program DTAM1, the element conductance matrices, $[k^e]$, will be positive semidefinite and the element capacitance matrices, $[c^e]$, will be positive definite.

Physically, the terms on the right hand side of equation (2.5) will often represent heat flow rate into the element, at each node, by conduction, convection and/or radiation, heat flow into storage, and heat flow generated internally within the element, respectively.

Other forms of element equations could have been considered, but equations of this form embrace a large variety of thermal models and lead to systems of equations that have been extensively studied and for which many numerical solution strategies have been published. (Other forms based upon conduction transfer functions, transmission matrix formulations, and equations associated with fluid flow heat transfer phenomena have been formulated and are presently under study.)

2.4 System Equations

Requiring conservation of energy at each system node we demand:

$$\left\{ \text{(direct heat flow rate)} = \sum_{\substack{\text{connected} \\ \text{elements}}} \text{(net heat flow rate into element)} \right\}_{\substack{\text{system} \\ \text{node}}} \qquad (2.6)$$

or, for an arbitrary system node, n, with connected elements "a", "b", "c", ... :

$$Q_n = q^a{}_{net\text{-}k} + q^b{}_{net\text{-}l} + q^c{}_{net\text{-}i} + ... \qquad (2.7)$$

where k, l, i, ... are the element nodes of elements a, b, c, ... , respectively, that are associated with the system node n.

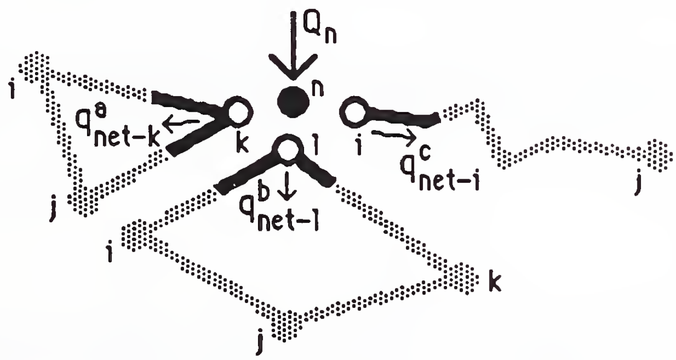


Fig. 3 Conservation of Energy at System Node

If individual element equations, of the form of equation (2.5), are substituted directly into the equilibrium relation of equation (2.7) a system of equations that relate system direct heat flow rate DOFs, {Q} , to the collection of element temperature DOFs, $\{T^a\}$, $\{T^b\}$, $\{T^c\}$, ... would result. As each element temperature DOF is associated with a specific system temperature DOF it is possible to reform these equations into the so-called *system equations* , that have the form:

$$[\,K\,]\{\,T\,\} + [\,C\,]\{dT/dt\} = \{\,E\,\} \qquad (2.8)$$

where;

[K] = matrix of system conductance transfer coefficients; the *system conductance matrix*

[C] = matrix of system thermal capacity coefficients; the *system capacitance matrix*

{ E } = vector of source/sink flow rates; the *system excitation vector*

For elements employed in the program DTAM1, the system conductance matrix, [K], will be positive semidefinite and the system capacitance matrix, [C], will be positive definite.

The process of forming the system equations from the element equations is known as the *element assembly process* . It may be represented formally by first establishing the one-to-one correspondence between each element's DOFs and the system DOFs through simple Boolean Transformations of the form:

$$\{T^e\} = [\, B^e] \{T\} \tag{2.9a}$$

or, equivalently, as the correspondence is one-to-one:

$$\{T\} = [\, B^e]^T \{T^e\} \tag{2.9b}$$

where:

[B^e] = the transformation matrix for element "e" ; a Boolean transformation matrix consisting of zeros and ones as the element DOFs are either equal to a system DOF (i.e., 1) or not (i.e., 0).

The same correspondence exists between each element's net heat flow rate DOFs and the contribution of that element to the equilibrium at each of the system nodes, represented, here, by $\{Q^e\}$, as:

$$\{Q^e\} = [\, B^e]^T \{q^e{}_{net}\} \tag{2.10}$$

Rewriting equation (2.7) in terms of the element contributions, $\{Q^e\}$, for all system nodes:

$$\{Q\} = \{Q^a\} + \{Q^b\} + \{Q^c\} + \ldots \tag{2.11a}$$

or

$$\{Q\} = \sum_{a,\, b,\, c,\, \ldots} \{Q^e\} \tag{2.11b}$$

and substituting the transformation expressions (2.9a), (2.10), and (2.5) we obtain equation (2.8), above, with:

$$[K] = [B^a]^T[k^a][B^a] + [B^b]^T[k^b][B^b] + [B^c]^T[k^c][B^c] + \dots \qquad (2.12)$$

$$[C] = [B^a]^T[c^a][B^a] + [B^b]^T[c^b][B^b] + [B^c]^T[c^c][B^c] + \dots \qquad (2.13)$$

$$\{E\} = \{Q\} + \{[B^a]^T\{q^a\} + [B^b]^T\{q^b\} + [B^c]^T\{q^c\} + \dots \qquad (2.14)$$

The formal representation of the assembly process, equations (2.12) , (2.13) , and (2.14), is useful for theoretical consideration. Practically, however, the system matrices may be assembled more directly and efficiently using the so-called "LM Algorithm" [3] . This algorithm is used to assemble element equations to form the system equations in the program DTAM1.

2.5 Boundary Conditions

Some nodal temperatures, $\{T_p\}$, may be prescribed (e.g., outside air temperature), while the rest, $\{T_f\}$ may be considered variable or free. At those nodes where nodal temperatures are variable, direct heat flow rate (or in the case of some of the elements, convective flow rates) may be prescribed. Equation (2.8) , then , may be partitioned as:

$$\begin{bmatrix} K_{ff} & K_{fp} \\ K_{pf} & K_{pp} \end{bmatrix} \begin{Bmatrix} T_f \\ T_p \end{Bmatrix} + \begin{bmatrix} C_{ff} & C_{fp} \\ C_{pf} & C_{pp} \end{bmatrix} \begin{Bmatrix} \dfrac{dT_f}{dt} \\ \dfrac{dT_p}{dt} \end{Bmatrix} = \begin{Bmatrix} E_f \\ E_p \end{Bmatrix} \qquad (2.15)$$

(Note: for *lumped mass* idealizations $[C_{fp}] = [C_{pf}] = [0]$.)

The first equation of equations (2.15) provides the governing equation of the free response of the system:

$$[K_{ff}]\{T_f\} + [C_{ff}]\{dT_f/dt\} = \{E_f\} - [K_{fp}]\{T_p\} - [C_{fp}]\{dT_p/dt\} \qquad (2.16)$$

The right hand side of this equation defines an *effective excitation*, that includes the effect of prescribed temperatures on system response.

Equation (2.16) may be solved by a variety of methods and the solution for $\{T_f\}$ may then be substituted into the second equation above:

$$\{E_p\} = [K_{pf}]\{T_f\} + [K_{pp}]\{T_p\} + [C_{pf}]\{dT_f/dt\} + [C_{pp}]\{dT_p/dt\} \qquad (2.17)$$

to determine the excitation quantities, $\{E_p\}$, needed to maintain the prescribed temperatures, $\{T_p\}$.

A prescribed nodal temperature, T_i , may also be imposed numerically by scaling the diagonal terms corresponding to the prescribed temperature DOF of either the system conductance matrix - term K_{ii} - and/or the system capacitance matrix - term C_{ii} - by a large value and setting the corresponding excitation term, E_i , to the product of the prescribed temperature and the appropriate combination of K_{ii} and C_{ii} (depending upon the solution type employed, steady, harmonic, or predictor-corrector integration) . This method is simple, effective, and easily implemented. It is, therefore, employed in DTAM1.

2.6 Solution of System Equations

The solution of equation (2.8) for {T(t)} defines the *system response* to a given *excitation*, {E(t)}. Response analysis may be classified by the nature of the excitation, the nature of the system (i.e., linear or nonlinear; constant property or nonconstant property), and the type and characteristics of the analytical or numerical method used to obtain the solution. Here we shall limit consideration to linear systems with constant properties, subjected to either steady excitation, harmonic excitation, or any general excitation that may be approximated by a piece-wise linear function.

2.6.1 Steady Excitation

For linear systems with constant properties driven by a steady excitation the response of the system will, eventually, come to a steady state (i.e., {dT/dt} = 0) given by the solution of:

$$[K]\{T\} = \{E\} \qquad (2.18)$$

2.6.2 Harmonic Excitation

For linear systems with constant properties driven by a steady harmonic excitation of the form:

$$\{E\} = \mathrm{Re}(\{E^*\}\, e^{i\omega t}) \; ; \; i = \sqrt{-1} \qquad (2.19a)$$

where;

{E*} = the complex excitation vector (i.e., the excitation at each DOF j, E^*_j, is represented in terms of amplitude or modulus, E'_j, and phase angle or argument , ϕ, as;

$$E^*_j = E'_j\, e^{i\phi} = E'_j\,(\cos(\phi) + i \sin(\phi)) \qquad (2.19b)$$

ω = circular frequency of excitation [=] radians/time

the response of the system will, eventually, come to a steady harmonic response:

$$\{T\} = \mathrm{Re}(\{T^*\}\, e^{i\omega t}) \qquad (2.20)$$

given by the solution of:

$$[K^*]\{T^*\} = \{E^*\} \qquad (2.21a)$$

where:

$[K^*] = [K + i\omega C]$ (2.21b)

$\{T^*\}$ = complex temperature response

= defined in terms of amplitude, T'_j, and phase angle ϕ, as:

$$T^*_j = T'_j e^{i\phi} = T'_j (\cos(\phi) + i \sin(\phi)) \qquad (2.21c)$$

2.6.3 General Excitation

A finite difference scheme for the approximate integration of the semi-discrete equation (2.8) may be developed by dividing time domain into discrete steps:

$t_{n+1} = t_n + \delta t$; $n = 0,1,2,3 \ldots$ (2.22)

t_0 = initial time

where:

δt = integration time step (often constant but may be variable)

demanding the satisfaction of equation (2.8) at each of these steps:

$$[K]\{T\}_{n+1} + [C]\{dT/dt\}_{n+1} = \{E\}_{n+1} \qquad (2.23)$$

where:

$\{T\}_{n+1} \equiv \{T(t_{n+1})\}$

$\{dT/dt\}_{n+1} \equiv \{dT(t_{n+1})/dt\}$

$\{E\}_{n+1} \equiv \{E(t_{n+1})\}$

and substituting into this equation, (2.23), the consistent difference approximation represented by:

$$\{T\}_{n+1} \approx \{T\}_n + (1-\alpha)\delta t\{dT/dt\}_n + \alpha\delta t\{dT/dt\}_{n+1} \qquad (2.24)$$

where:

$0 \le \alpha \le 1$

$\alpha = 0$ corresponds to the *Forward Difference* scheme

$\alpha = 1/2$ corresponds to the *Crank-Nicholson* scheme

$\alpha = 2/3$ corresponds to the *Galerkin* scheme

$\alpha = 1$ corresponds to the *Backward Difference* scheme

a general implicit finite difference scheme is formulated:

$$[\ \alpha\delta t[K] + [C]\ ]\{dT/dt\}_{n+1} \approx \{E\}_{n+1} - [K]\{\ \{T\}_n + (1-\alpha)\delta t\{dT/dt\}_n\ \} \qquad (2.25a)$$

or, equivalently:

$$[\ [K] + (1/\alpha\delta t)[C]\ ]\{T\}_{n+1} \approx \{E\}_{n+1} + (1/\alpha\delta t)[C]\{T\}_n + (1-\alpha)\delta t[C]\{dT/dt\}_n \qquad (2.25b)$$

Computationally it is strategic to implement this general finite difference scheme, equation (2.25), as a three step predictor-corrector algorithm:

$$\{T'\}_{n+1} = \{T\}_n + (1-\alpha)\delta t(dTdt\}_n \qquad \text{; predictor step} \qquad (2.26a)$$

$$[\ \alpha\delta t[K] + [C]\ ]\{dT/dt\}_{n+1} \approx \{E\}_{n+1} - [K]\{T'\}_n \qquad \text{; (i.e., equation (2.25a))} \qquad (2.26b)$$

$$\{T\}_{n+1} = \{T'\}_{n+1} + \alpha\delta t\ \{dT/dt\}_{n+1} \qquad \text{; corrector step} \qquad (2.26c)$$

It should be noted that this algorithm is self-starting. Given initial conditions, $\{T(t_0)\}$, equation (2.23) may be solved to obtain an estimate of the initial rate of change of nodal temperatures, $\{dT(t_0)/dt\}$, and the first predictor step, equation (2.26a), may then be computed.

This predictor-corrector scheme has been analyzed by Taylor [1] and Huebner [5] and a more general predictor-multicorrector scheme that includes this *implicit* scheme has been analyzed by Hughes [6]. For $\alpha \geq 1/2$ this scheme leads to an unconditionally stable (approximate) solution; for $\alpha \geq 3/4$ (approximately) leads to an unconditionally stable non-oscillatory solution; beyond this Taylor makes some recommendations regarding selection of α and step size, δt, to limit error while minimizing computational effort. In the program DTAM1 the default value of α is set to 0.75, and may be reset by the user, and an estimate of the time step needed to limit error is reported (for given initial conditions) using a method developed by Taylor [1].

2.7 Specific Element Equations

This section presents the element equations for:

- 1D two-node thermal resistance elements,
- single-node lumped capacitance elements,
- two-node fluid flow loop elements,
- 1D two-to-four node isoparametric conduction Finite Elements,
- 2D four-node isoparametric conduction Finite Elements (planar and axisymmetric), and
- variable-node mean radiant temperature element.

2.7.1 Resistance Elements

Resistance elements may be developed directly from fundamental heat transfer theory. A 2-node resistance element results that is defined by an element conductance matrix of the general form:

$$[k^e] = (A/R)\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \qquad (2.27)$$

where:

A = the area available for heat transfer
R = the element resistance

Fig. 4 2-Node Resistance Element

2.7.2 Lumped Capacitance Element

Lumped capacitance elements may be developed directly from fundamental heat transfer theory. A 1-node capacitance element results that is defined by an element capacitance matrix of the general form:

$$[c^e] = (mc)\,[1] \qquad (2.28)$$

where;

m = mass of element
c = specific heat capacity

Fig. 5 Lumped Capacitance Element

2.7.3 Flow Loop Elements

Energy transferred by convective flow may be modeled with flow-related elements. Flow-related element equations are best formulated in terms of nodal temperatures and the net rate of enthalpy change, $h^e_{net\text{-}i}$, (i.e., enthalpy transported from the node, i, into the element). For the simple case of incompressible fluid flow from a single node to another, ignoring fluid capacitance effects, the element equation takes the form:

$$\begin{Bmatrix} h^e_{net\text{-}1} \\ h^e_{net\text{-}2} \end{Bmatrix} = (w_{1\text{-}2}\, C_P) \begin{bmatrix} 1 & 0 \\ -1 & 0 \end{bmatrix} \begin{Bmatrix} T^e_1 \\ T^e_2 \end{Bmatrix} \qquad (2.29)$$

where:

$w_{1\text{-}2}$ = the mass flow rate from node 1 to node 2

C_P = the specific heat capacity at constant pressure of the fluid

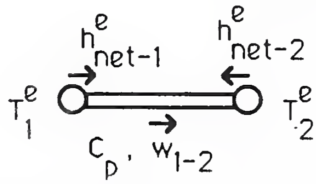


Fig. 6 2-Node Simple Flow Element

In principal, element equations relating rate of enthalpy change to nodal temperatures may assembled, along with nonflow related elements, to form system equations governing coupled flow and nonflow heat transfer. Flow related element equations are, however, characteristically nonsymmetric, as in the case presented above. The program DTAM1 is organized to take advantage of the property of symmetry of nonflow elements and, therefore, even this simple flow related element may not be added to its library of elements. A subassemblage of two simple flow elements, creating a flow loop, however, results in a symmetric system of equations of the form:

$$\begin{Bmatrix} q^e_{net\text{-}1} \\ q^e_{net\text{-}2} \end{Bmatrix} = \begin{Bmatrix} h^{e1}_{net\text{-}1} + h^{e2}_{net\text{-}1} \\ h^{e1}_{net\text{-}2} + h^{e2}_{net\text{-}2} \end{Bmatrix} = (w\, C_P) \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \begin{Bmatrix} T^e_1 \\ T^e_2 \end{Bmatrix} \qquad (2.30)$$

where:

w = the mass flow rate from node 1 to node 2 and return, (here we have used superscripts e1 and e2 indicate the two simple flow elements e1 and e2 of the subassembly)

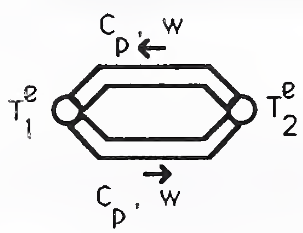


Fig. 7 2-Node Flow Loop Element

The flow loop element equations are seen to be similar in form to the simple resistance element equations and, therefore, flow loops may be modeled with resistance elements by setting the quotient (A/R) equal to the product of (wC_p), as is commonly done in thermal network analysis. This approach to formulating flow loop equations, based upon a consideration of rate of enthalpy change, leads to a entire class of new thermal elements that is currently under investigation.

2.7.4 Transient Conduction Finite Elements

The solution of transient heat conduction problems using the finite element method has been discussed by several authors [1, 3, 5, 7, 8, 9, & 10]. Over a given region, Ω , bounded by a surface, Γ , of a solid continuum, the spacial and temporal variation of temperature, T(x,y,z,t) , due to conduction is governed by the so-called transient heat conduction equation:

$$\frac{\partial}{\partial x}(k_x \frac{\partial T}{\partial x}) + \frac{\partial}{\partial y}(k_y \frac{\partial T}{\partial y}) + \frac{\partial}{\partial z}(k_z \frac{\partial T}{\partial z}) = \rho c \frac{\partial T}{\partial t} + q^*(x,y,z,t) \quad ; \text{ in } \Omega \tag{2.31a}$$

with temperature boundary conditions specified on part of the boundary, Γ_1 :

$$T = T_P(x, y, z, t{=}0) \quad ; \text{ on } \Gamma_1 , \; t > 0 \tag{2.31b}$$

direct heat flow and/or convective heat flow specified on part of the boundary, Γ_2 :

$$k_x \frac{\partial T}{\partial x} n_x + k_y \frac{\partial T}{\partial y} n_y + k_z \frac{\partial T}{\partial z} n_z = q(x,y,z,t) + h(x,y,z,t)\,(T_\infty - T) \quad ; \text{ on } \Gamma_2 , t > 0 \tag{2.31c}$$

and initial conditions:

$$T = T_0(x, y, z, t{=}0) \quad ; \text{ in } \Omega \tag{2.31d}$$

where;

k_x , k_y , k_z	= the principal conductivities for a thermally anisotropic continuum
ρ	= the density
c	= the specific heat capacity
q^*	= the internal heat generation rate per unit volume
T_P	= the prescribed temperature boundary condition
n_x , n_y , n_z	= the direction cosines of a normal to the surface at (x, y, z)
q	= the prescribed heat flow rate boundary condition on Γ_2
h	= the convective heat transfer coefficient on Γ_2
T_∞	= the temperature of the external environment that drives the convective heat flow

In the finite element method a solution for the temperature field, T(x, y, z, t), is approximated by assumed functions that are defined over a small, but finite, part of the region (i.e., the finite element). These functions have the general form:

$$T(x, y, z, t) = \{N(x, y, z)\}\{T^e(t)\} \tag{2.32}$$

where the spacial functions, $\{N(x, y, z)\}$, - the so-call *shape functions* - are selected to satisfy continuity requirements across inter-element boundaries. The spacial variation of the temperature field is thus approximated in a piecewise manner, using the shape functions, and the temporal variation is captured by the variation of discrete values of temperature, $\{T^e\}$, that correspond to the element temperature DOFs discussed above.

Using the approximation represented by equation (2.32) and the Galerkin variation of the method of weighted residuals, the transient heat conduction equation is transformed to equations of the form of (2.8) that may be assembled, in the manner discussed above, from element equations of the form of equation (2.5) where, now:

– the element conductance matrix coefficients for elements in Ω and on Γ_1 are:

$$k_{ij}^e = \int_{\Omega^e} \left(k_x \frac{\partial N_i}{\partial x}\frac{\partial N_j}{\partial x} + k_y \frac{\partial N_i}{\partial y}\frac{\partial N_j}{\partial y} + k_z \frac{\partial N_i}{\partial z}\frac{\partial N_j}{\partial z} \right) d\Omega^e \tag{2.33a}$$

– the element conductance matrix coefficients for elements on Γ_2 are:

$$k_{ij}^e = \int_{\Omega^e} \left(k_x \frac{\partial N_i}{\partial x}\frac{\partial N_j}{\partial x} + k_y \frac{\partial N_i}{\partial y}\frac{\partial N_j}{\partial y} + k_z \frac{\partial N_i}{\partial z}\frac{\partial N_j}{\partial z} \right) d\Omega^e + \int_{\Gamma_2} h\, N_i N_j d\Gamma_2 \tag{2.33b}$$

– the element capacitance matrix coefficients for all elements are:

$$c_{ij}^e = \int_{\Omega^e} \rho\, c\, N_i N_j d\Omega^e \tag{2.33c}$$

– the element internal generation rate vector components are:

$$q_i^e = \int_{\Omega^e} q^{\circ}\, N_i d\Omega^e \tag{2.33d}$$

– the system direct heat flow rate vector is augmented by the components:

$$Q_i^e = \int_{\Gamma_2} (\, q\, N_i + h\, T_\infty N_i)\, d\Gamma_2 \tag{2.33e}$$

where Q_i^e , the element contribution to the system direct heat flow rate at element node i, includes two terms - a direct surface gain term and a convective gain term, respectively.

A practically unlimited variety of specific conduction elements may be formulated using these equations; no restrictions on the number of nodes per element have been made and the compatibility requirements placed upon admissible shape functions is not very restrictive. Furthermore, even after decisions regarding the number of nodes per element and the choice of shape functions has been made the analyst may choose to employ analytical or various numerical integration strategies to evaluate element matrices thereby generating subclasses of element types.

Elements having distorted geometries and/or having nonuniform node spacings may be conveniently formulated by mapping the *global* (i.e., actual or physical) coordinate geometry (x,y) to an undistorted *local* geometry (r,s) and evaluating the integral expressions, above, in the undistorted geometry space using relatively simple shape functions defined in the undistorted geometry space. If the mapping from the global geometry to the local geometry is defined using the same shape functions used to approximate the temperature field (i.e., in equation (2.32)) then the element formulation is said to be *isoparametric*. (An introduction to isoparametric element formulation may be found in [8] pp.193-230 and the practical implementation of the approach is presented in [11].)

Two finite elements are provided in the program DTAM1; a variable 2 to 4-node isoparametric one-dimensional conduction finite element and a 4-node isoparametric two-dimensional finite element.

<u>Two to Four-Node One-dimensional Isoparametric Elements</u>

A variable two to four-node isoparametric conduction finite element may be formulated for one-dimensional heat transfer by using the following shape functions, $\{N\} = \{N_1, N_2, N_3, N_4\}$, defined relative to a biunit element in local coordinate r as (see [3] page 199):

	Include only if node 3 is present	Include only if node 3 and 4 are present	
$N_1 = \frac{1}{2}(1 - r)$	$-\frac{1}{2}(1 - r^2)$	$+\frac{1}{16}(-9r^3 + r^2 + 9r - 1)$	(2.34)
$N_2 = \frac{1}{2}(1 + r)$	$-\frac{1}{2}(1 - r^2)$	$+\frac{1}{16}(9r^3 + r^2 - 9r - 1)$	
$N_3 = (1 - r^2)$		$+\frac{1}{16}(27r^3 + 7r^2 - 27r - 7)$	
$N_4 = \frac{1}{16}(-27r^3 - 9r^2 + 27r + 9)$			

where;

r = the local coordinate defined as indicated below:

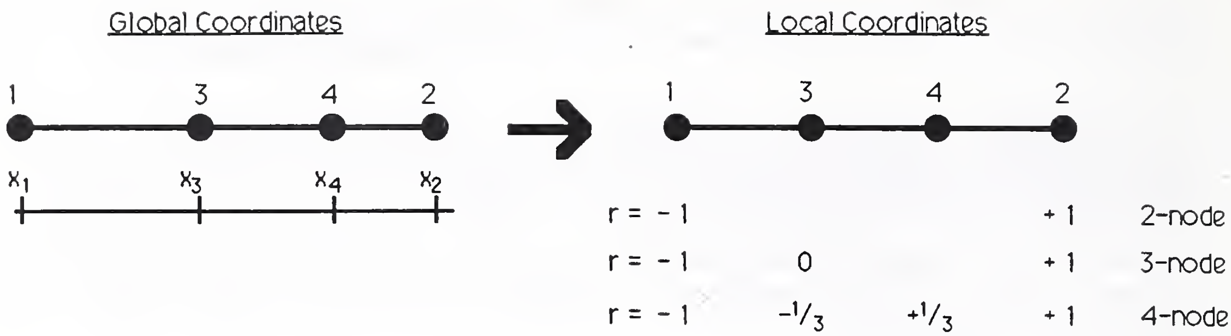


Fig. 8 Two to Four-Node Element Coordinate Systems

In the isoparametric formulation the element global coordinate, x, is related to the element local coordinate, r, through a transformation defined by the shape functions given above:

$$x = \{N\}\{x\} \tag{2.35}$$

where;

$\{N\}$	$= \{N_1 , N_2\}$ for two-node elements
	$= \{N_1 , N_2 , N_3\}$ for three-node elements
	$= \{N_1 , N_2 , N_3 , N_4\}$ for four-node elements
$\{x\}$	$= \{x_1 , x_2\}$ for two-node elements
	$= \{x_i , x_2 , x_3\}$ for three-node elements
	$= \{x_1 , x_2 , x_3 , x_4\}$ for four-node elements

The integral equations (2.33) may then be rewritten in terms of the local coordinates as:

$$k_{ij}^{e} = Ak \int_{x_1}^{x_2} \frac{dN_i}{dx} \frac{dN_j}{dx} \, dx = Ak \int_{-1}^{+1} \frac{dN_i}{dr} \frac{dN_j}{dr} \left(\frac{1}{J}\right) dr \tag{2.36a}$$

$$c_{ij}^{e} = A \rho c \int_{x_1}^{x_2} N_i N_j \, dx = A \rho c \int_{-1}^{+1} N_i N_j J \, dr \tag{2.36b}$$

$$q_i^{e} = q^* \int_{x_1}^{x_2} N_i \, dx = q^* \int_{-1}^{+1} N_i J \, dr \tag{2.36c}$$

where:

A	= the area available for heat transfer
k	$= k_x$; the element conductivity
J	$= dx/dr = d/dr\{N\}\{x\}$; the *Jacobian* of the coordinate transformation.

It is convenient to evaluate these integrals numerically. Gauss quadrature may be used to exactly evaluate these integrals, but the resulting equations will tend to be "stiff" and, as a result, will tend to demonstrate artificial oscillatory behavior [12,13]. Hughes has shown that this problem can be mitigated through approximate evaluation of the capacitance matrix using appropriate quadrature rules [6]. A diagonal or lumped capacitance matrix, which tends to inhibit artificial oscillatory behavior at the expense of accuracy, will result if quadrature points for the evaluation of the capacitance matrix, (2.36b), are selected to be equal to the element nodal coordinates in the local coordinate system.

For the two-node element the element matrices include:

– the element conductance transfer matrix:

$$[k^e] = (Ak/L)\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \qquad (2.37)$$

– the element thermal capacitance matrix:

$$[c^e] = (A\rho cL)\begin{bmatrix} (1/2 - r) & r \\ r & (1/2 - r) \end{bmatrix} \qquad (2.38)$$

where:

r = 0 corresponds to a *lumped* capacitance model that results from using the quadrature points (-1 , +1)
= 1/12 corresponds to a *higher order* capacitance model that results from using the quadrature points ($-\sqrt{2/3}$, $+\sqrt{2/3}$)
= 1/6 corresponds to a *consistent* capacitance model that results from using Gauss quadrature (i.e., quadrature points $-1/\sqrt{3}$, $+1/\sqrt{3}$) to exactly evaluate equation (2.36b)

and

– the element internally generated heat flow rate vector (e.g., a radiant floor) is:

$$\{q^e\} = q^* L/2 \begin{Bmatrix} 1 \\ 1 \end{Bmatrix} \qquad (2.39)$$

It may be shown that the higher order capacitance matrix will minimize error in the evaluation of temperature [6].

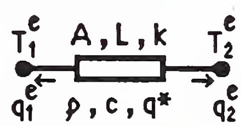


Fig. 9 One-Dimensional Two-Node Conduction Finite Element

When used with the implicit numerical integration scheme provided in the program

DTAM1, the higher order capacitance model, r = 1/12, results in the greatest accuracy for a given time step. The choice of r=0 results in equations identical to those associated with the finite difference method solutions.

Four-Node Two-Dimensional Isoparametric Element

Element matrices for a four-node two-dimensional isoparametric element may be evaluated in a similar manner as that presented above for the one-dimensional isoparametric element. The program DTAM1 provides planar and axisymmetric elements that are evaluated using Gauss integration of integral expressions corresponding to equations (2.33). As Gauss integration results in the exact evaluation of these integrals these elements will result in "stiff" equations, which tend to demonstrate artificial oscillatory behavior in regions where the temperature field has rapidly changing and pronounced gradients.

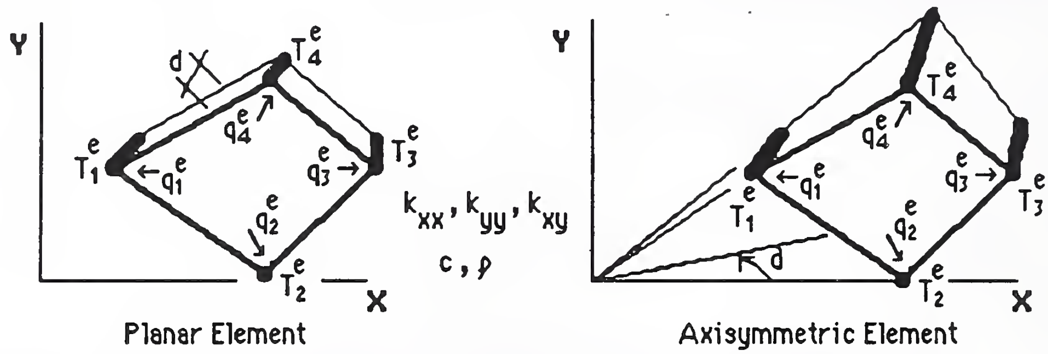


Fig. 10 Four-Node Isoparametric Finite Element

2.7.5 Variable Node Mean Radiant Temperature Elements

A variable node nonlinear radiative element may be derived directly from Joseph Carroll's mean radiant temperature network method [14, 15, 16] that has a single node for each surface considered and an additional node:

> "The fictitious Mean Radiant Temperature node in the center of the network [that] acts as a clearinghouse for all radiative interchanges."

The surface nodes are linked to the MRT node by simple resistance elements whose resistances are defined, nonlinearly, by Carroll's method. The element conductance transfer matrix for this variable node radiative element will have the form:

$$[k^e] = h_b \begin{bmatrix} \sum(A_i F'_i) & (-A_1 F'_1) & (-A_2 F'_2) & \cdots & (-A_n F'_n) \\ (-A_1 F'_1) & (A_1 F'_1) & 0 & \cdots & 0 \\ (-A_2 F'_2) & 0 & (A_2 F'_2) & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ (-A_n F'_n) & 0 & 0 & \cdots & (A_n F'_n) \end{bmatrix} \qquad (2.40)$$

where:
The element DOF are numbered i=0,1,2,3 ... n with DOF i=0 corresponding to the MRT node.

h_b $= 4\sigma T^3$

σ = Stefan-Boltzmann Constant

T = the average absolute temperature of surface

A_i = area available for radiative exchange from surface i

F'_i = radiant interchange factor between surface i and the MRT node

$= 1/(\,1/F_i + (1-\varepsilon_i)/\varepsilon_i)$

$F_i = 1/(\,1 - A_i F_i / \sum(A_i F_i)\,)$; determined iteratively

ε_i = emissivity of surface i

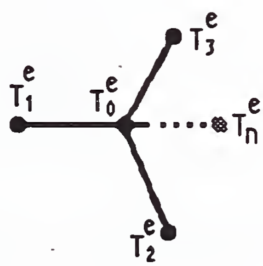


Fig. 11 Variable Node MRT Element

3. Summary and Conclusions

An approach to building energy simulation has been presented that is based upon discrete analysis techniques used in other fields of physical system simulation wherein system behavior is governed by an overriding equilibrium principle (e.g., mass, momentum, or energy conservation) that is used to define an assembly process that allows the formation of system equations from individual element equations. The element equations are developed to model the behavior of discrete regions (e.g., finite elements) or discrete processes within the system. In the present context the governing equilibrium principle is the conservation of energy and element equations have been presented that model conductive, convective, and radiative energy transport and that account for thermal storage processes in building systems.

The discrete thermal analysis theory and its practical implementation in the program DTAM1 has been presented to provide a first demonstration of this very general approach. It has been shown that element equations may be developed from very basic considerations (e.g., the 1D thermal resistance element and the simple capacitance element), from Finite Element approximations to governing partial differential equations (e.g., the 1D and 2D isoparametric conduction Finite Elements), and from more heuristic, ad hoc physical arguments (i.e., the variable-node MRT element). Although the discussion was limited to element formulations with constant thermal properties the extension to nonconstant and nonlinear thermal properties is straightforward. Consequently, a practically unlimited variety of elements could, conceivably be developed following the examples presented in this report. In particular the nonsymmetric flow element discussed in section 2.7.3 should lead, quite naturally, to the development of a variety of HVAC component elements.

One may also approach the formation of system equations based upon either conduction response function theory or transmission matrix theory from an element assembly approach and, thus, consider the development of *response function elements* and/or *transmission matrix elements*. In these cases, however, the system equations that result will be algebraic, rather than differential, the extension to nonlinear situations is not so straightforward, and mixed assemblies of elements based upon the approach presented here with those based upon these heat transfer theories will only be possible in special cases. Nevertheless, by taking this approach to the formulation of these approaches one is able to bring the full range of building energy simulation possibilities under one unifying theory.

Steady state, steady periodic, and transient analysis options for thermal systems with constant thermal properties (i.e., linear thermal systems) have been discussed. Additional solution options may also be considered for linear thermal systems and a variety of techniques exists for the analysis of thermal system with nonconstant and/or nonlinear thermal properties; Table 1 outlines some of these possibilities.

From a theoretical point of view the discrete thermal analysis method serves to unify existing building energy simulation theory and place it on a rigorous base. In so doing the building energy simulation community will be in a better position to:

- compare advantages and disadvantages of existing building energy simulation methods,

- make use of the enormous volume of research literature presently available in the general area of discrete analysis techniques (e.g., the Finite Element and related numerical methods), and
- move on to the more demanding task of simulating coupled thermal, flow, and HVAC system behavior for indoor air quality analysis.

From a computational point of view the discrete thermal analysis method, being based upon distinct formal operations (e.g., formation of element matrices, assembly of element matrices, solution of system equations, & post evaluation of element results) using a library of thermal elements, leads to highly modular program structures and could be the basis of a higher level programming language for building energy simulation program development.

From a practical point of view the discrete thermal analysis method allows an analyst to make use of all the "tools" that have emerged from building energy simulation research community, to create detailed or simple idealization – as suits the purpose – that may readily be modified as design evolves (i.e., through the addition, modification, or deletion of elements), and, for a given idealization, to consider simple to complex models of excitation idealization as appropriate to the stage of design or level of analytical inquiry.

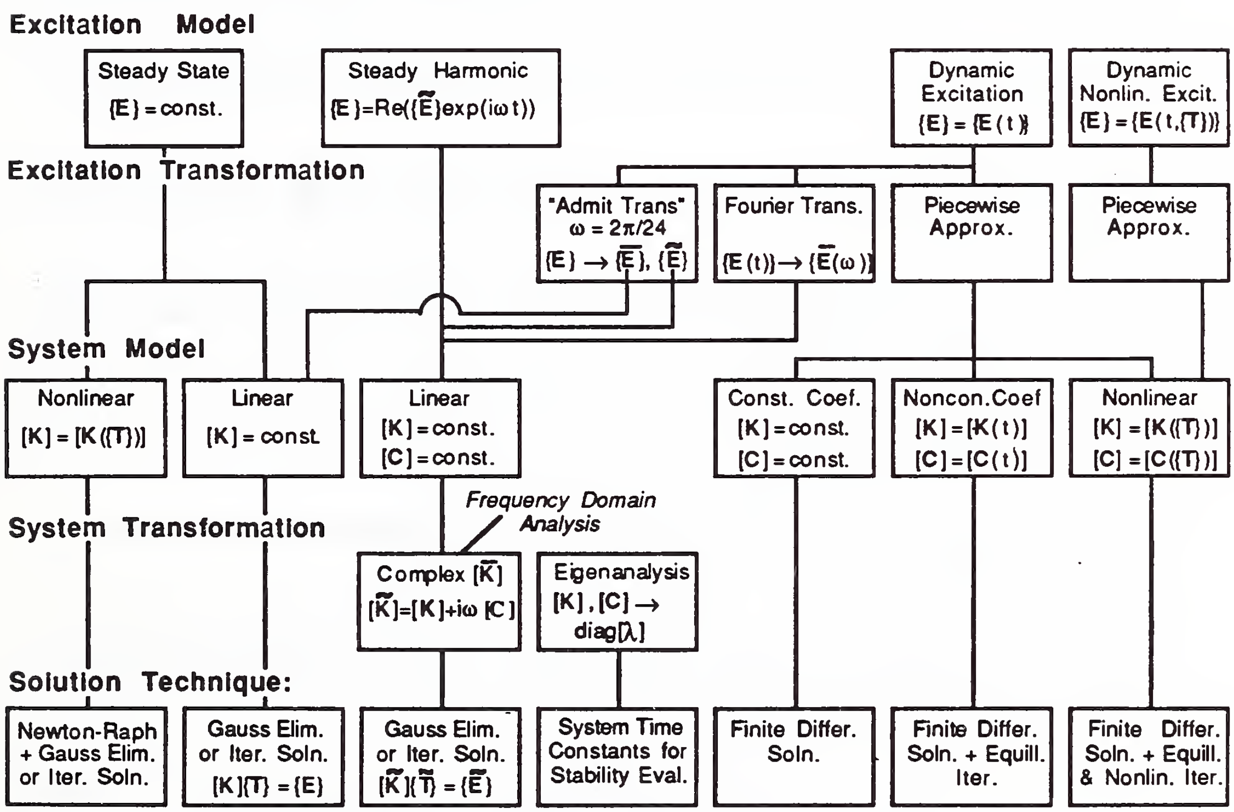


Table 1 Solution Options for The Governing Discrete Thermal Analysis System Equations

$[K]\{T\} + [C]\{dT/dt\} = \{E\}$

4. References

[1] Taylor, Robert L., "HEAT", A Finite Element Computer Program for Heat-Conduction Analysis, Report 75-1, Prepared for: Civil Engineering Laboratory, Naval Construction Battalion Center, Port Hueneme, California, May 1975

[2] Wilson, E. L. & Hoit, M. I., "A Computer Adaptive Language for the Development of Structural Analysis Programs", Computers & Structures Vol.19, No.3, pp 321-338, 1984

[3] Bathe,K.J., Finite Element Procedures in Engineering Analysis, Prentice-Hall, Inc. Englewood Cliffs, New Jersey, 1982

[4] Parlett, B., "Roundoff Error in the Solution of Finite Element Systems," Proceeding, U.S.-Germany Symposium, Formulation and Computational Procedure in Finite Element Analysis, M.I.T., August, 1976

[5] Huebner, K. H., The Finite Element for Engineers, Second Edition, John Wiley & Sons, New York, 1982

[6] Hughes, T.J., "Analysis of Some Fully-Discrete Algorithms for the One-Dimensional Heat Equation," International Journal for Numerical Methods in Engineering, Vol. 21, 1985

[7] Zienkiewicz, O.C., The Finite Element Method, Third Edition, McGraw-Hill Book Co., London, 1977

[8] Zienkiewicz, O.C. & Morgan, K., Finite Elements and Approximation, John Wiley & Sons, New York, 1983

[9] Cook, R. D., Concepts and Applications of Finite Element Analysis, Second Edition, John Wiley & Sons, New York

[10] Desai, C.S., Elementary Finite Element Method, Prentice-Hall, Englewood Cliffs, New Jersey, 1979; pp. 108-120

[11] Akin, J.E., Application and Implementation of Finite Element Methods, Academic Press, London, 1982

[12] Emery, A.F. & Sugihara, K. "A Comparison of Some of the Thermal Characteristics of Finite-Element and Finite Difference Calculations of Transient Problems," Numerical Heat Transfer, Vol 2, pp.97-113, 1979

[13] Schreyer, Howard L., "Nonlinear Finite-Element Heat Conduction Analysis with Direct Implicit Time Integration," Numerical Heat Transfer, Vol 4, pp.377-391, 1981

[14] Carroll, J.A., "An MRT Method of Computing Radiant Energy Exchange in Rooms," Proceedings of the 2nd Systems Simulation & Economic Analysis Conference, San

Diego, 1980, pp. 343-348

[15] Carroll, J.A., "A Comparison of Radiant Interchange Algorithms", Proceedings of 3rd Systems Simulation and Economic Analysis Conference, Reno, 1981

[16] Carroll, J.A. & Clinton, J.R., "A Thermal Network of a Passive Solar House", Proceedings of 5th National Passive Solar Conference, Amherst, 1980

[17] Hoit, M.I., "New Computer Programming Techniques for Structural Engineering," Ph.D. Dissertation, SESM, Dept. of Civil Engineering, Univ.Ca., Berkeley, 1983

DTAM1 USERS MANUAL

I. General Instructions

Thermal analysis of a building system, using DTAM1, involves three basic steps:

Step 1: Idealization of the Building System and Excitation

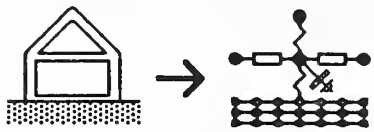


Fig. I.1 Idealization of the Building System and Excitation

Idealization of the building thermal system involves definition of a coordinate system, discretization of the system as an assemblage of appropriate thermal elements connected at system nodes, identification of boundary conditions, numbering of system nodes optimally (i.e., to minimize the bandwidth of system equations), and estimation of the bandwidth of the system equations (i.e., the largest coupled DOF number difference plus 1). Selection of appropriate thermal elements involves some skill and considerable knowledge of the characteristics of building thermal behavior and the numerical and behavioral characteristics of specific thermal elements.

The excitation - specified temperatures, direct nodal heat flow rates, and external environmental temperatures associated with convection - may be modeled to be steady, to vary harmonically, or defined in terms of arbitrary piecewise-linear time histories. For the latter case initial conditions of nodal temperatures will have to also be specified. (For those problems where these initial conditions are not known apriori the analyst may use a false start-up period to attempt to determine initial conditions.)

Step 2: Preparation of Input Command/Data File

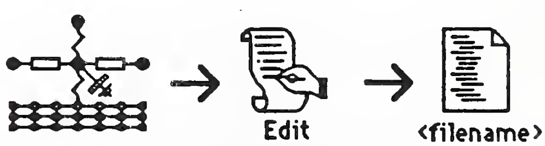


Fig. I.2 Preparation of Input Command/Data File

The program DTAM1 is, in essence, a command processor. The program reads an ASCII text file that contains commands and associated data, collected together in distinct data groups, that define the thermal idealization. The command/data input file may be prepared with any available ASCII text editing program and given a file name, <filename>, specified by the user. The <filename> must, however, consist of 8 or less alphanumeric characters and can not include an extension (i.e., characters separated from the filename by a period, ".").

Step 3: Execution of DTAM1

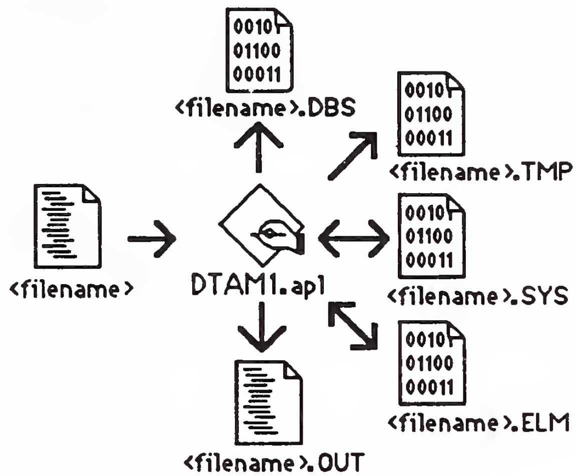


Fig. I.3 Execution of DTAM1

DTAM1 is then executed. It will request the name of the command/data input file and proceed to form element and system arrays and compute the solution to the posed problem. DTAM1 reads the ASCII input file and creates a single ASCII (i.e., printable) output file and up to four binary files:

Files Read

<filename> an ASCII input file specified by the user that contains problem data,

Files Created

<filename>.OUT a printable ASCII output file that contains analysis results,

<filename>.DBS a binary file containing system arrays that may be read by CAL-80 program segments for further processing of system arrays,

<filename>.TMP a binary file containing the computed time histories of nodal temperatures,

<filename>.PRE a binary file used for temporary disk storage of system arrays used by the predictor-corrector integration subroutines in DTAM1, and

<filename>.ELM a binary file used for temporary disk storage of element arrays.

The results of an analysis, <filename>.OUT, may be conveniently reviewed using an ASCII editor and, from the editor, portions or all of the results may be printed out.

II. Preparation of Input File

Input data is defined in terms of data groups that are delimited by a data group separator label, that may be thought to be a command, and an end flag. With the exception of the problem initiation data group, data groups may be ordered as the user sees fit. Command/data input files follow the free-field input conventions of the CAL-SAP software development system (see Appendix A).

The specific data groups needed to be included in a command/data input file will depend upon the details of the thermal idealization, the nature of the excitation, and the type of analysis to be computed. For the dynamic analysis, using the predictor-corrector integration option, an input file will have the following general form (where nn is a data value):

Column	1
Problem Initiation:	**DTAM1** <problem labeling information> **NODE=nn BAND=nn** **SAVE**
Constraint Data Group:	**CONSTRAIN** nn,nn,nn M=nn . . . **END**
Coordinate Data Group:	**COORDINATES** nn X=nn Y=nn Z=nn nn X=nn Y=nn Z=nn GEN=nn,nn,nn . . . **END**
Element Data Groups: Resistance Elements:	 **RESIST** nn I=nn,nn R=nn A=nn nn I=nn,nn R=nn A=nn GEN=nn . . . **END**
Capacitance Elements:	**CAPACI** nn I=nn M=nn C=nn nn I=nn M=nn C=nn GEN=nn . . . **END**
Isoparametric 1D 2-Node Elements:	**ISO1D** nn I=nn,nn K=nn P=nn C=nn A=nn nn I=nn,nn K=nn P=nn C=nn A=nn GEN=nn . . . **END**
Isoparametric 2D 4-Node Elements:	**ISO2D4** nn I=nn,nn,nn,nn K=nn,nn,nn C=nn P=nn nn I=nn,nn,nn,nn K=nn,nn,nn C=nn P=nn . . . **END**
Boundary Conditions Group:	**BOUNDARY COND**

```
nn,nn,nn BC=nn
nn,nn,nn BC=nn
...
END
```

Convection Boundary Conditions Group:

```
CONVECTION
nn I=nn,nn H=nn
nn I=nn,nn H=nn GEN=nn
...
END
```

Solution Control
- Predict-Correct. Integ.
Solution Control Data:

```
PREDICT
TIME=nn,nn,nn ALPHA=nn NEFN=nn
nn,nn,nn T=nn Q=nn TEX=nn
nn,nn,nn T=nn Q=nn TEX=nn
...
END
```

Initial Conditions Group:

```
INITIAL COND
nn,nn,nn T=nn
nn,nn,nn T=nn
...
END
```

Excitation Function Data:

```
EXCITATION
T=nn EFN=nn,nn,nn, ...
T=nn EFN=nn,nn,nn, ...
...
END
```

Report Control Data:

```
REPORT
WRITE PINT=nn
nn,nn,nn
...
END
```

Details are given on the following pages for each data group.

A. Problem Initiation

An analysis problem must be initiated by the following lines of data:

DTAM1
<problem labeling information> :
NODE=I1   BAND=I2
[SAVE]

where:

I1 = the number of nodes

I2 = the (half) bandwidth; the largest DOF number difference between coupled nodes plus 1

If the keyword SAVE is found at the beginning of the fourth line, system arrays will be written to the binary file <filename>.DBS that may be read by CAL-80 program segments for further processing. CAL-80 [16] is a high-level language developed to be used as an educational/research tool that facilitates basic matrix algebraic operations and numerical analysis methods (e.g., equation solving, eigenanalysis, fast fourier transformation, etc.). CAL-80 may be obtained from the National Information Service for Earthquake Engineering, NISEE/Computer Applications, Davis Hall, University of California, Berkeley, California 94720, (415) 642-5113.

B. Constraint Data Group

System temperature DOFs may be constrained to be equal (e.g., see Fig. 1) by the following lines of data:

CONSTRAIN
I1,I2,I3   M=I4
...
END

where:

I1, I2, I3 = first node, last node, node number increment of a group of *slave* nodes constrained to be equal to the same *master* node; I3=1 default

I4 = the *master* node to which the other, *slave* , nodes are constrained

Constraining system temperature DOF provides one means of using one-dimensional elements in conjunction with two-dimensional elements. If not done with care, constraining may have the effect of destroying the compactness of system arrays and thus result in unnecessary computational effort in solving the system equations.

C. Coordinate Data Group

Coordinate data is required for some, but not all, elements. It is defined by the following lines of data:

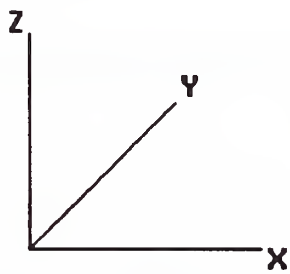
```
COORDINATES
I1  X=R1  Y=R2  Z=R3  GEN=I2,I3,I4
...
END
```

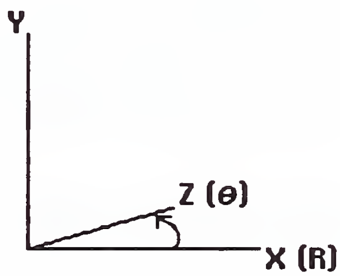
where;

I1	= the node number
R1	= the x coordinate
R2	= the y coordinate
R3	= the z coordinate
I2,I3,I4	= first node, last node, node number increment; I4=1 default

If the generation data is given, nodal coordinates are generated at equal intervals along a straight line between the first and last nodes, I2 & I3, incrementing node numbers by the generation increment, I4. Nodal coordinates not given will be assumed to be zero. Coordinate data is not needed for all elements; see specific element input description below for details.

Either rectangular and cylindrical coordinate data may be defined but right hand coordinate systems must be used:





D. Element Data Groups

Element data is entered by element type. Details for each element type is given below. Each element type follows, however, the general form:

```
<element name>
I1  I=I2,I3, ... GEN=I4  <element properties>  [O=C1]
. . .
END
```

where:

I1 = the element number
I2,I3,... = the element node numbers
I4 = generation increment (default = 1)
C1 = **Y** if element nodal heat flow rate results are to reported (i.e., output)
= **N** if element nodal heat flow rate results are not to be reported; **N** default.

Element data must be supplied in numerical order. Omitted element data is automatically generated by incrementing the preceding node numbers by the current generation increment. Properties of generated elements will be set equal to that of the current element.

***** The O=Y option is presently not implemented. *****

1) Resistance Elements

Conventional 2-node resistance element equations may be added to the system equations with the following lines of data:

RESIST
I1 I=I2,I3 GEN=I4 R=R1 A=R2
. . .
END

where:

I1 = the element number
I2, I3 = the element node numbers
I4 = generation increment (default = 1)
R1 = the element resistance
R2 = the area available for heat transfer

2) Lumped Capacitance Elements

Conventional 1-node capacitance element equations (i.e., lumped capacitance) may be added to the system equations with the following lines of data:

CAPACI
I1 I=I2 GEN=I3 M=R1 C=R2
. . .
END

where;

I1 = the element number
I2 = the element node number
I3 = generation increment (default = 1)

R1 = the thermal mass
R2 = the specific heat capacity of the thermal mass

3) Fluid Flow Loop Elements

Elements equations that model heat transfer via a fluid flow loop between any two nodes may be added to the system equations with the following lines of data:

```
FLOWLOOP
I1  I=I2,I3  GEN=I4  W=R1  C=R2
. . .
END
```

where:

I1	= the element number
I2, I3	= the element node numbers
I4	= generation increment (default = 1)
R1	= the mass flow rate from first node to second & second node to first
R2	= the fluid specific heat capacity (at constant pressure)

4) Isoparametric 1D Two to Four-node Conduction Finite Elements

Isoparametric 1D two to four-node conduction finite element equations may be added to the system equations with the following lines of data:

```
ISO1D
I1  I=I2,I3,[I4,I5]  GEN=I6  X=R1,R2,[R3,R4]  K=R5  P=R6  C=R7  A=R8  [Q=R9]
[R=...]
. . .
END
```

where:

I1	= the element number	
I2,I3	= the element end node numbers (for node number order see sketch below)	
I4,I5	= the intermediate nodes (for node number order see sketch below):	
	if I3 ≤ 0	a two-node linear element will be generated
	if I3 > 0 & I4 ≤ 0	a three-node quadratic element will be generated
	if I3 > 0 & I4 > 0	a four-node cubic element will be generated
I6	= generation increment (default = 1)	
R1,R2	= the element end-node coordinates (i.e., along the length of the element)	
R3,R4	= the element intermediate node coordinates	
R5	= the element conductivity	
R6	= the element weight density	
R7	= the element specific heat capacity	
R8	= the area available for heat transfer	

R9 = the internal heat generation rate per unit length of element

R=... = **GAUSS** or = **R10, R11, [R12,R13]**

- if **GAUSS** Gauss quadrature will be used to evaluate the element capacitance matrix
- **R10, R11, [R12,R13]** are quadrature points to be used to evaluate element capacitance matrix; the number of quadrature points used will be equal to the number of element nodes (this assures exact integration when Gauss quadrature points are used); the following quadrature points will result in a lumped capacitance modeling:

Element		R10	R11	R12	R13
2-Node Linear	-1.00	1.00			
3-Node Quadratic	-1.00	0.00	1.00		default
4-Node Cubic	-1.00	-1/3	+1/3	1.00	default

- for 2-node linear elements the quadrature points $R = -\sqrt{2/3}, +\sqrt{2/3}$ will minimize error; these quadrature points are the default quadrature points for the 2-node linear element.

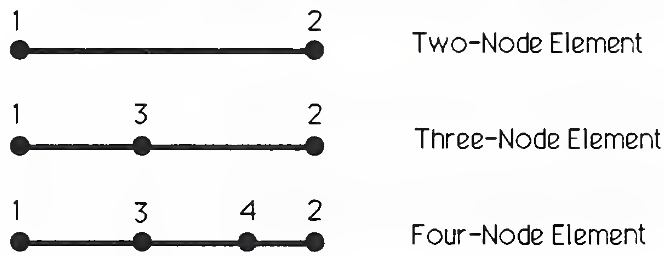


Fig. II.2 One-Dimensional Isoparametric Element Node Numbering

NOTE:

a) One may implement, in effect, a 1D finite difference model, corresponding to a linear difference approximation over the interval of the element length, by using lumped capacitance two-node elements.

b) When using 3-node quadratic and 4-node cubic elements lumped capacitance modeling may provide more reliable results than Gauss quadrature capacitance modeling (see Example B1 discussed in Appendix B).

5) Isoparametric 2D Four-node Conduction Finite Elements

Isoparametric 2D 4-node conduction finite element equations may be added to the system equations with the following lines of data:

ISO2D4

I1 I=I2,I3,I4,I5 GEN=I6 K=R1,R2,R3 C=R4 P=R5 [Q=R6] [D=R7] [REG=C1]

. . .
END

where;

I1 = the element number

I2,I3,I4,I5 = the element node numbers ordered counterclockwise around the element

R1,R2,R3 = the components of the conductivity tensor, K_{xx}, K_{yy}, K_{xy}; for homogeneous, isotropic materials $K_{xx}=K_{yy}=k$, $K_{xy}=0$; for anisotropic materials aligned to the chosen coordinate system $k_{xx} = k_x$, $k_{yy} = k_y$, $k_{xy} = 0$

R4 = the specific heat capacity

R5 = the weight density

R6 = the internal heat generation per unit volume: R6=0.0 default

R7 = the thickness (planar regions) or subtended angle (axisymmetric regions) of the element: R7=1.0 default

C1 = **PLN** for planar regions or **AXI** for axisymmetric regions: **PLN** default.

Triangular elements are possible; they are identified by repeating any two nodal point numbers (e.g., I,J,K,K).

E. Boundary Conditions Data Group

Boundary condition flags are established with the following lines of data:

BOUNDARY CONDITIONS
I1,I2,I3 BC=C1
. . .
END

where:

I1,I2,I3 = the first node, last node, node number increment of a series of nodes with identical boundary conditions

C1 = **Q** for heat flow rate prescribed boundaries; **T** for temperature prescribed boundaries; **Q** default

The heat flow rate or the temperature - but not both - may be specified at each node to establish boundary conditions of prescribed temperature or heat flow rate. These conditions are specified with as many lines as needed.

If this data group is omitted all nodes will be assumed to be heat flow rate DOF.

F. Convection Boundary Conditions Data Group

Convection boundary conditions are established by the following lines of data:

CONVECTION BOUNDARY CONDITIONS

I1 I=I2,I3 GEN=I4 H=R1 [D=R2] [REG=C1]

. . .

END

where:

- I1 = the convective surface segment number
- I2,I3 = the surface segment end nodes
- I4 = the generation increment: I4=1 default
- R1 = the convective transfer coefficient
- R2 = the the thickness (planar regions) or subtended angle (axisymmetric regions) of the element: R2=1.0 default
- C1 = **PLN** for planar regions or **AXI** for axisymmetric regions: **PLN** default.

Convective boundary conditions may be specified for surface segments defined by pairs of nodes that have been specified to be part of the Q boundary. For these nodes, then, it is possible to account for both general direct heat flow excitation and convective excitation (e.g., direct solar-to-surface gain and convective air-to-surface gain). These conditions are specified with as many lines as needed.

(Convective boundary conditions would reasonably only be associated with subassemblages of two-dimesional conduction elements. Furthermore, one would reasonably only specify plane convective transport for subassemblages of plane elements and axisymmetric convective transport for subassemblages of axisymmetric elements. No attempt is made, however, to check the consistency of the model.)

G. Solution Control

Data groups needed to control the solution procedure and define the excitation are explained below.

1) Steady State Excitation/Response

The response of the system to steady excitation may be computed by including the following lines of data:

STEADY

I1,I2,I3 T=R1 Q=R2 TEX=R3

. . .

END

where:

- I1,I2,I3 = the first node number, last node number, and node number increment of a series of nodes with identical excitation conditions
- R1 = the specified temperature: R1=0.0 default
- R2 = the specified heat flow rate: R2=0.0 default
- R3 = the specified convective fluid temperature: R3=0.0 default.

2) Harmonic Excitation/Response

The response of the system to steady harmonic excitation may be computed by including the following lines of data:

```
HARMONIC
F=R1
I1,I2,I3   T=R2,R3  Q=R4,R5  TEX=R6,R7
. . .
END
```

where:

R1 = the frequency of the harmonic excitation [=] cycles/time

I1,I2,I3 = the first node number, last node number, and node number increment of a series of nodes with identical excitation conditions

Harmonic Temperature Excitation:

R2 = Amplitude of Excitation: R2 = 0.0 default

R3 = Time Lag: R3 = 0.0 default

Harmonic Direct (External) Flow Rate Excitation:

R4 = Amplitude of Excitation: R4 = 0.0 default

R5 = Time Lag: R5 = 0.0 default

Harmonic External (Convective Fluid) Temperature Excitation:

R6 = Amplitude of Excitation: R6 = 0.0 default

R7 = Time Lag: R7 = 0.0 default

note; Frequency = (circular frequency)/2π

Lag is defined in terms of the phase angle or argument of the complex representations of both excitation components and response as:

Lag = (-argument)/(2π (frequency))

(See discussion in section 2.6.2 of the REPORT, above, for clarification.)

3) General Excitation/Response - Predictor-Corrector Integration

The response of the system, including transients, to a general dynamic excitation that is defined in terms of piecewise linear functions may be computed by including the following lines of data:

Solution Control Data:

```
PREDICTOR-CORRECTOR
TIME=R1,R2,R3  ALPHA=R4  NEFN=I1
I2,I3,I4  T=I5  Q=I6  TEX=I7
```

. . .
END

where:

R1,R2,R3	= beginning time, ending time, time step increment for dynamic analysis
R4	= integration parameter; $0.0 \leq R4 \leq 1.0$; R4=0.75 default; instability may result for R4 < 0.5 .
I1	= number of excitation functions that will be defined.
I2,I3,I4	= the first node number, last node number, and node number increment of a series of nodes with identical excitation conditions
I5	= the excitation function number for specified temperature
I6	= the excitation function number for specified external flow rates
I7	= the excitation function number for specified external fluid temperature for convective boundary conditions

Initial Conditions Data:

INITIAL CONDITIONS
I1,I2,I3 T=R1
. . .
END

or

INITIAL CONDITIONS
READ=C1
END

where:

I1,I2,I3	= the first node, last node, node number increment of a series of nodes with identical initial temperature conditions
R1	= the initial temperature condition: R1=0.0 default
C1	= the filename of file to read initial conditions from.

Initial temperature conditions are specified by as many lines as needed. If this data group is omitted initial conditions of temperature at all nodes will be assumed to be zero.

If the READ=C1 option is used instead, initial conditions will be read from the last record of <filename>.TMP in the form (TIME, (T(N), N=1,NNOD)) where NNOD is the number of nodes in the system and the value TIME is ignored (see **REPORT** below). This option facilitates restarting a problem to continue computation or as a means to get an estimate of initial conditions that are not known apriori.

Excitation Function Data:

EXCITATION
T=R1 EFN=R2,R3,R4, ...
. . .
END

where:

R1	= time
R1,R2,R3, ...	= excitation function values for time=R1 for excitation functions 1, 2, 3,

... respectively

H. Output Report Control

The following lines of data control output reporting:

REPORT
[WRITE] PINT=I1
I2,I3,I4
. . .
END

where:

If the keyword **WRITE** is included the time and computed temperature solution for all nodes will be written to a binary file <filename>.TMP as a series of records of the form (TIME, (T(N),N=1,NNOD)), where NNOD is the number of nodes in the system

I1 = the temperature solution "print" output interval (the solution will be written to a printable ASCII file <filename>.OUT; used by predictor-corrector integration procedure only - otherwise ignored

I2,I3,I4 = the first node number, last node number, and node number increment of a series of nodes selected for printable output.

The **WRITE** option is useful, when used with the **READ** option used to specify initial conditions (above) to restart analysis.

Appendix A - Free-Field Input Conventions of the CAL-SAP Software Development System

A "C" in column 1 of any line will cause the line to be echoed as a comment on the console.

A backslash "\" at the end of information on the first line will allow the next line to be interpreted as a continuation of the first line: therefore, a 160 character record is possible.

A colon ":" indicates the end of information on any line. Information entered to the right of the colon is ignored by the program. Therefore, it can be used to annotate an input file.

Data may be identified by an identifier of the form "<identifier=" (e.g. NUM=). Data not identified by an identifier must be placed first in a line. If fewer data items are supplied than required the missing data will be assumed to be zero or blank as appropriate.

Decimal points are not needed for real data. Scientific notation formats of the form E± nn (e.g. 5.5 E-15) may be used. Simple arithmetic expressions may be used using the standard operators +, -, *, and / . The order of evaluation is sequential from left to right - unlike FORTRAN.

Appendix B - Example Problems

B1 Response of a Semi-Infinite Slab to a Step Change of Surface Heat Flux

To demonstrate the use and characteristics of the one- and two-dimensional isoparametric conduction finite elements the temperature response of a semi-infinite slab, T(x,t), to a step change of surface flux, Q(0,t) = 10.0 Btu/sec ; for t > 0 , was considered. All material properties were assigned unit values;

Conductivity = k = 1.0 Btu/sec-in-°F

Specific Heat = c = 1.0 Btu-in/sec^2-lb-°F

Mass Density = ρ = 1.0 sec^2-lb/in^4

and initial conditions were assumed to be zero, T(x,0) = 0.0 .

The exact solution to this problem is given by[1];

$$T(x,t) = \frac{20}{k}\left\{\left(\frac{\kappa t}{\pi}\right)^{1/2} \exp(-x^2/4\kappa t) - \left(\frac{x}{2}\right) \operatorname{erfc}(x/2\sqrt{\kappa t})\right\}$$

where;

$\kappa = k/\rho c$

Four different models of this problem were considered, each representing a 6 inch deep solid with a perfectly insulated boundary at the 6 inch depth. These models are illustrated below in Figure B1. It may be seen that these models are similar in that an even spacial discretization at 0.5" intervals is used in all models. Variants of each of the three models employing one-dimensional isoparametric element assemblages based on different capacity modeling strategies were also investigated bringing the total number of models studied to eight;

Model A1: 12 Two-node 1D isoparametric elements with "consistent" (i.e., exact evaluation of the element capacitance matrices using two-point Gauss quadrature),

Model A2: 12 Two-node 1D isoparametric elements with "lumped" element capacitance matrices (i.e., using two-point quadrature with quadrature points of (-1.0, +1.0)),

Model A3: 12 Two-node 1D isoparametric elements with "optimal" evaluation of the element capacitance matrices (i.e., using two-point quadrature with quadrature points of $(-\sqrt{2/3}, +\sqrt{2/3})$),

Model B1: 6 Three-node 1D isoparametric elements with "consistent" (i.e., exact)

---

[1] Carslaw, H.S. & Jaeger, J.C., Conduction of Heat in Solids, 2nd Ed. Oxford University Press, 1969.

evaluation of the element capacitance matrices (i.e., using three-point Gauss quadrature),

Model B2: 6 Three-node 1D isoparametric elements with "lumped" element capacitance matrices (i.e., using two-point quadrature with quadrature points of (-1.0, 0.0, +1.0)),

Model C1: 4 Four-node 1D isoparametric elements with "consistent" (i.e., exact) evaluation of the element capacitance matrices (i.e., using four-point Gauss quadrature),

Model C2: 4 Four-node 1D isoparametric elements with "lumped" element capacitance matrices (i.e., using two-point quadrature with quadrature points of (-1.0, -1/3, +1/3, +1.0)),

Model D: 12 Four-node 2D isoparametric elements with "consistent" (i.e., exact evaluation of the element capacitance matrices using four-point Gauss quadrature).

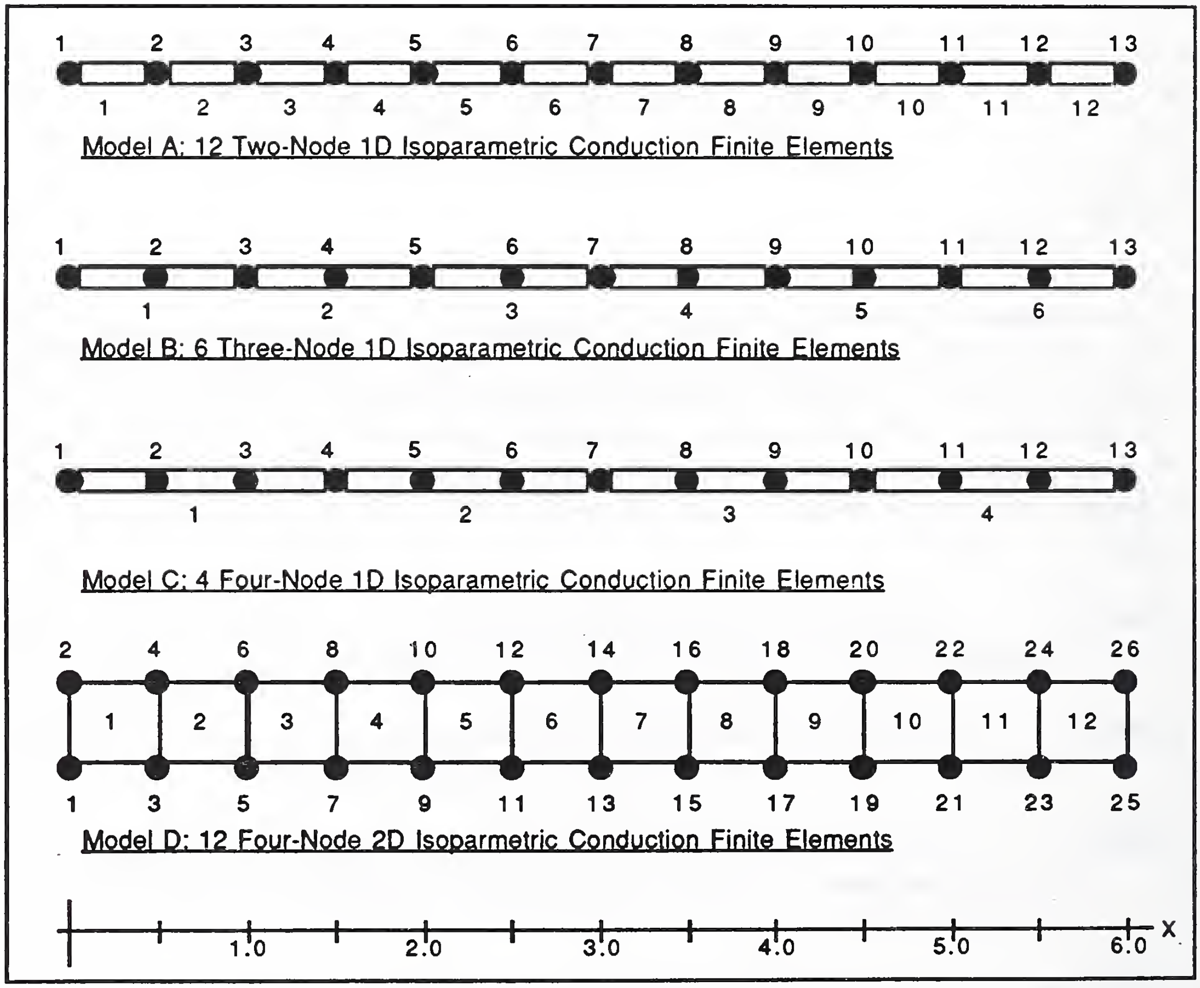


Fig. B1 Thermal Models Used to Study The Response of a Semi-Infinite Slab

Temperature distributions were computed for t=0.02 sec and t=0.10 sec. The results of Model A1 and Model D , the consistent 1D and 2D isoparametric elements repectively, were practically identical in this case of 1D heat transfer. This is to be expected since identical shape functions and numerical integration schemes were used for these cases. Results of analysis for the seven one-dimensional element assemblages are plotted below.

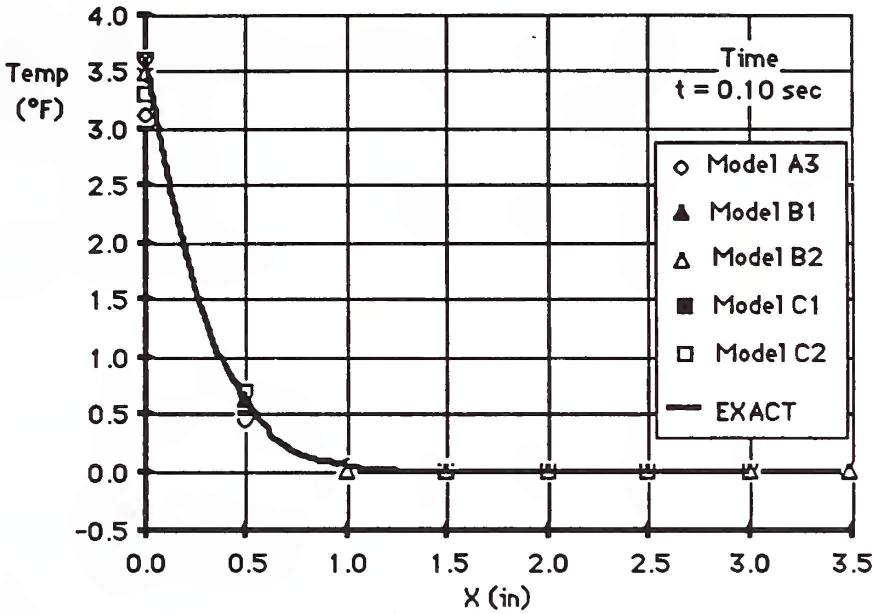


Fig. B2 Temperature Profiles After a Long Time Interval

The results of all modelings are very close after long time intervals, converging to the exact solution. It may be seen, however, that the higher order elements (i.e., the four-node and three-node elements) provide marginally better approximations of the temperature profiles. Although it appears that the use of "consistent" capacitance modeling for these higher order elements provides slightly better approximations of the temperature profiles than the "lumped" approximations the difference is slight and the "lumped" capacitance modeling results in a diagonal system capacitance matrix, a fact that can be used to advantage in transient solution algorithms.

For short time intervals the two-node element assemblages, Models A1-A3, tend to predict results that overshoot the exact solution, as may be seen in Fig. B3. Comparing the results of the three capacitance modeling alternatives reveals that the "consistent" capacitance modeling provides the best estimation of surface temperature but overshoots at the second node the most, the "optimal" capacitance modeling provides minimal overall error in the estimation of temperature but reveals a slight overshoot at the second node, and the "lumped" capacitance modeling suffers no overshoot at the second node but proves least accurate in modeling the surface temperature. The lumped capacitance modeling results in a model that is mathematically equivalent to a finite difference model and as such provides a useful comparison between finite-element and finite-difference analysis of this particular problem.

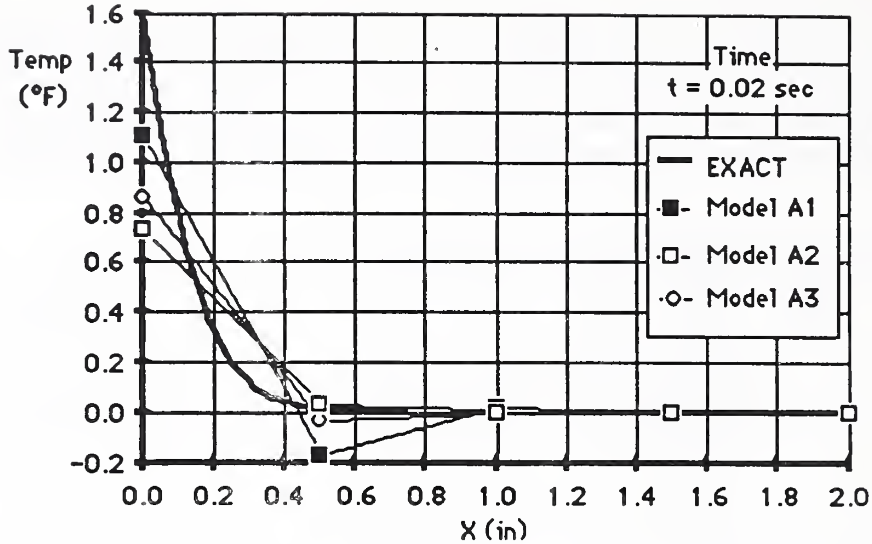


Fig. B3 Temperature Profiles After a Short Time Interval for Models A1-A3

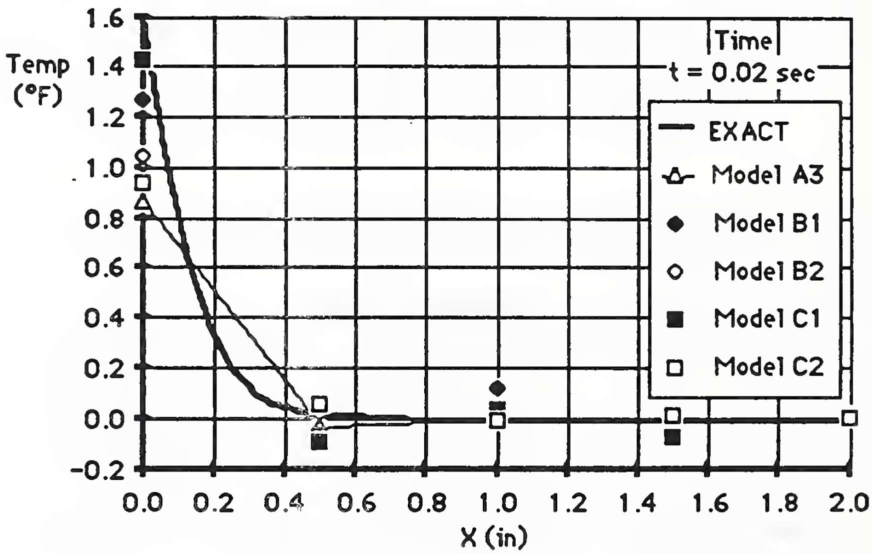


Fig. B4 Temperature Profiles After a Short Time Interval for Model A and Models B & C

Finally, Fig. B4 provides a comparison of the "optimal" two-node element assemblage with the three-node assemblages, Models B1 & B2, and the four-node assemblages, Models C1 & C2. The higher order elements, again, provide better estimates of surface temperature than the lower order elements with the "consistent" higher order elements out-performing the "lumped" higher order elements. The "lumped" higher order elements, Models B2 and C2, on the other hand, tend to exhibit less overshoot and again lead to diagonal system capacitance matrices that can result in substantial savings in memory and solution time.

The command/data input file for Model B, using Gauss quadrature for capacitance modeling is listed below. The output results file for this problem is also listed.

Model B-Gauss Command/Data Input File

```
DTAM1:
Semi Infinite Slab: MODEL B-Gauss
NODE=13 BAND=3:
ISO1D:
1  I=1,3,2         X=0.0,1.0,0.5 K=1.0 P=1.0 C=1.0 A=1.0 R=GAUSS:
6  I=11,13,12 GEN=2 X=0.0,1.0,0.5 K=1.0 P=1.0 C=1.0 A=1.0 R=GAUSS:
END:
PREDICT:
TIME=0,0.1,0.005 ALPHA=0.75  NEFN=1:
1  Q=1:
END:
EXCITATION:
T=0   EFN=10.0:
T=1.0 EFN=10.0:
END:
REPORT:
PINT=4:
1,7,1
END:
```

Model B-Gauss Output Results File

```
------------------------------------------------------------------------
| DTAM1:   Building Energy Simulation for Linear Systems               |
--------------------------------------------------------------Ver-6-86--
                                           Jim Axley - Cornell & NBS
                                                  MTOT:      50000
 ==== PROBLEM LABEL

 SEMI INFINITE SLAB

 ==== PROBLEM CONTROL VARIABLES

      Number of nodes .............    13
      Maximum probable bandwidth ...    3

 ==== EQUALITY CONSTRAINT CONDITIONS

   -- NOTE: Constraint condition data not found.
            No DOFs will be constrained to be equal.

 ==== NODAL   COORDINATES

   -- NOTE: Coordinate data not found.

 ==== ELEMENT DATA

   == ISOPARAMETRIC 1D 2 TO 4-NODE CONDUCTION ELEMENTS

      Kx = Conductivity     P = Density      C = Capacity
      A  = Area             Q = Int. Heat
```

```
Elem Node Coords.    Kx         P         C         A         Q'   Quad. Points
   1    1  .00      1.0       1.0       1.0       1.0       .00   GAUSS
        3 1.0
        2 .50
   2    3  .00      1.0       1.0       1.0       1.0       .00   GAUSS
        5 1.0
        4 .50
   3    5  .00      1.0       1.0       1.0       1.0       .00   GAUSS
        7 1.0
        6 .50
   4    7  .00      1.0       1.0       1.0       1.0       .00   GAUSS
        9 1.0
        8 .50
   5    9  .00      1.0       1.0       1.0       1.0       .00   GAUSS
       11 1.0
       10 .50
   6   11  .00      1.0       1.0       1.0       1.0       .00   GAUSS
       13 1.0
       12 .50

  -- NOTE: Needed bandwidth .....      3
            Requested bandwidth ..     3

  -- NOTE: Time to form equations:      4 seconds.

==== BOUNDARY.CONDITIONS AND EQUATION NUMBERS

  -- NOTE: Boundary condition data not found.
             All nodes assumed to be flux-prescribed nodes.
             Equation numbers = node numbers.

==== CONVECTIVE BOUNDARY CONDITIONS

  -- NOTE: Convection boundary data not found.

==== SOLUTION

  == DYNAMIC SOLUTION: PREDICTOR-CORRECTOR INTEGRATION

  -- SOLUTION CONTROL INFORMATION
     Start time .......................   .000
     End time .........................   .100
     Time step increment ..............  0.500E-02
     Integration parameter, alpha .....   .750
     Number of excitation functions ...      1

  -- EXCITATION FUNCTION NUMBERS

     Node            Temp        Ext.Flux       Ext.Temp.
       1                            1              0

  -- INITIAL CONDITIONS

  -- NOTE: Initial condition data not found.
```

```
          Initial temperatures assumed to be zero.

-- TIME STEP

-- NOTE: Estimated time step to limit error to approx. 5.00% is: 0.277E-02
          Specified time step is: 0.500E-02

-- RESPONSE

   Time:    0.200E-01
Node    Temp.    Node    Temp.    Node    Temp.    Node    Temp.    Node    Temp.
   1    1.27        2 -0.760E-01    3    .117        4 -0.516E-02    5   0.932E-02
   6 -0.101E-03    7   0.486E-03

   Time:    0.400E-01
Node    Temp.    Node    Temp.    Node    Temp.    Node    Temp.    Node    Temp.
   1    2.09        2   0.167E-01    3    .102        4   0.905E-02    5 -0.131E-03
   6  0.142E-02    7 -0.863E-03

   Time:    0.600E-01
Node    Temp.    Node    Temp.    Node    Temp.    Node    Temp.    Node    Temp.
   1    2.68        2    .178        3  0.678E-01    4   0.188E-01    5 -0.447E-02
   6  0.126E-02    7 -0.641E-03

   Time:    0.800E-01
Node    Temp.    Node    Temp.    Node    Temp.    Node    Temp.    Node    Temp.
   1    3.16        2    .366        3  0.472E-01    4   0.221E-01    5 -0.313E-02
   6  0.412E-03    7   0.101E-03

   Time:     .100
Node    Temp.    Node    Temp.    Node    Temp.    Node    Temp.    Node    Temp.
   1    3.56        2    .564        3  0.476E-01    4   0.216E-01    5   0.853E-03
   6 -0.211E-03    7   0.611E-03

-- NOTE: Solution time:      9 seconds.
```

B2 Residential Building Example

A section of a hypothetical town house and a corresponding thermal idealization is illustrated below.

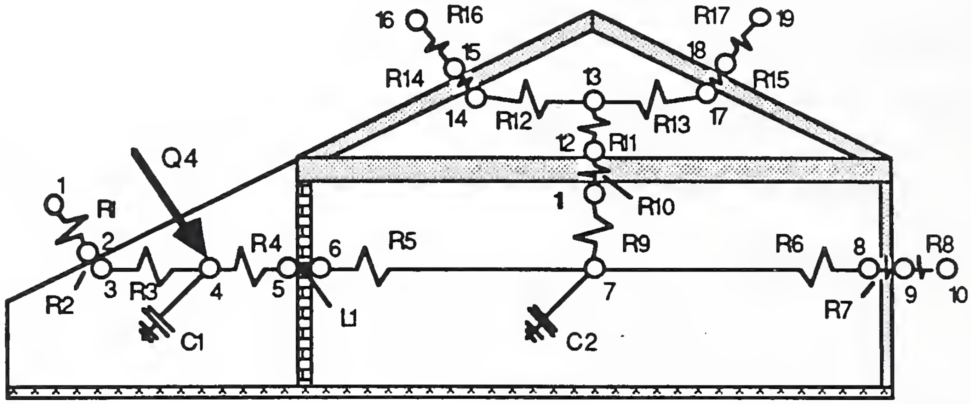


Fig B5 Hypothetical Town House and Thermal Idealization

It may be seen that the model is primarily a R/C network model with a single linear one-dimensional two-node isoparametric finite element, L1, used to model the dynamics of the mass wall.

The command/input file to compute both steady and harmonic indoor air temperature responses for specified conditions of outside air temperature and solar gain is listed below;

```
DTAM1:
Townhouse Model:
NODE=19 BAND=5:
RESIST: Consistant Units of Btu, ft, hr
1    I=1,2    A=17 R=1/1.5  :R1   Air Film at Sunspace
2    I=2,3    A=17 R=0.9    :R2   Sunspace Glass
3    I=3,4    A=17 R=1/1.5  :R3   Air Film at Sunspace
4    I=4,5    A=8  R=1/1.5  :R4   Air Film at Mass Wall
5    I=6,7    A=8  R=1/1.5  :R5   Air Film at Mass Wall
6    I=7,8    A=8  R=1/1.5  :R6   Air Film at N. Wall
7    I=8,9    A=8  R=11     :R7   N. Wall
8    I=9,10   A=8  R=1/1.5  :R8   Air Film at N. Wall
9    I=7,11   A=24 R=1/1.1  :R9   Air Film at Ceiling
10   I=11,12  A=24 R=11     :R10  Ceiling
11   I=12,13  A=24 R=1/1.6  :R11  Air Film at Ceiling
12   I=13,14  A=13 R=1/1.3  :R12  Air Film at Roof
13   I=13,17  A=13 R=1/1.3  :R13  Air Film at Roof
14   I=14,15  A=13 R=11     :R14  Roof
15   I=17,18  A=13 R=11     :R15  Roof
16   I=15,16  A=13 R=1/1.6  :R16  Air Film at Roof
17   I=18,19  A=13 R=1/1.6  :R17  Air Film at Roof
END:
CAPACI: Consistant Units of Btu, ft, hr
```

```
1 I=4    M=150*12/3   C=0.16   :C1 Sunspace Quickmass (4" conc. floor)
2 I=7    M=150*24/3   C=0.16   :C1 Room Quickmass (4" conc. floor)
END:
ISO1D: Consistant Units of Btu, ft, hr
1 I=5,6 X=0.0,8.0/12.0 K=1.0 P=150 C=0.2 A=8 :L1  8" Concrete Wall
END:
BOUNDARY:
1    BC=T     :Outside air temp to be specified
10   BC=T     :Outside air temp to be specified
16   BC=T     :Outside air temp to be specified
19   BC=T     :Outside air temp to be specified
END:
STEADY:
1    T=7.5              :Outside air temp - Steady Component
10   T=7.5              :Outside air temp - Steady Component
16   T=7.5              :Outside air temp - Steady Component
19   T=7.5              :Outside air temp - Steady Component
4    Q=200*13/3.1416    :Sunspace Solar Gain - Steady Component
END
HARMONIC:
F=1.0/24
1    T=10,10            :Outside air temp - Harmonic Component
10   T=10,10            :Outside air temp - Harmonic Component
16   T=10,10            :Outside air temp - Harmonic Component
19   T=10,10            :Outside air temp - Harmonic Component
4    Q=200*13*0.5,6     :Sunspace Solar Gain
END
REPORT:
PINT=1:
1,19,1
END:
```

B3 Analysis of the Huron Building Thermal Bridge

In this example, assemblages of two-dimensional isoparametric conduction elements and simple resistance elements are used to model the details of heat transfer in an exterior masonry wall section, at the intersection of a concrete floor slab, Figure B6, of a recently constructed federal office building in Huron, South Dakota, studied earlier by Grot et. al.[2].

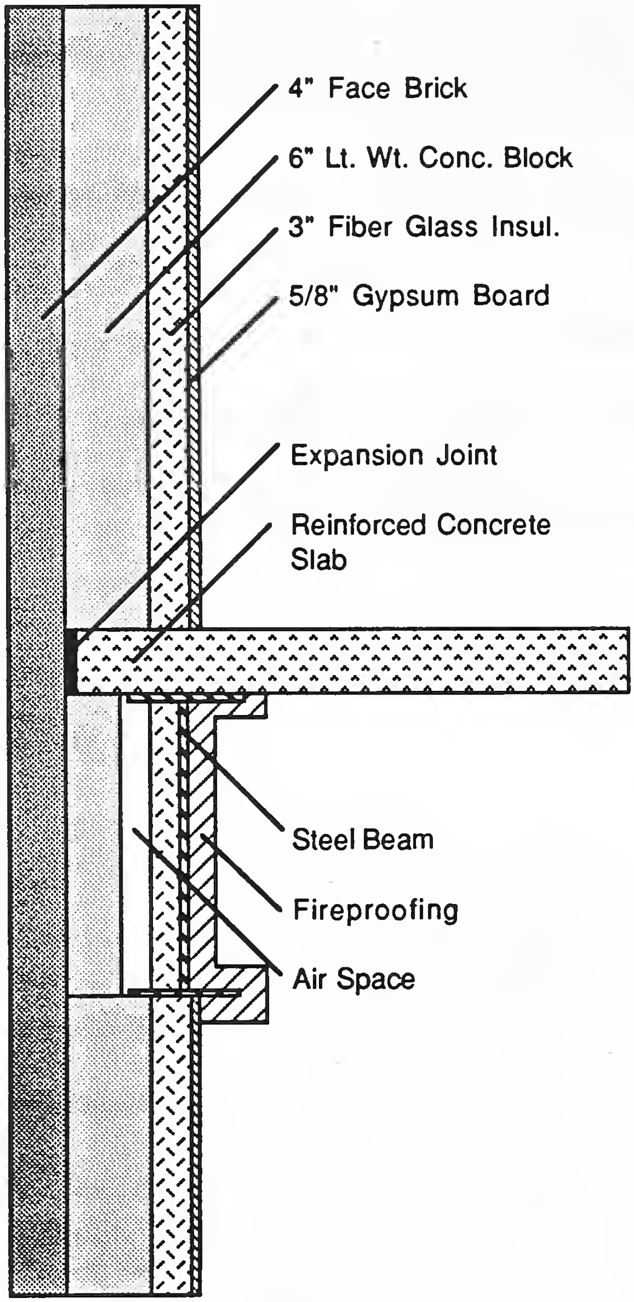


Fig. B6 Huron Building Wall Section

[2] Grot, R.A., Childs, K.W., Fang, J.B., & Courville, G.E., "The Measurement and Quantification of Thermal Bridges in Four Office Buildings", ASHRAE Trans. 1985, V. 91, Pt. 1, CH-85-11 No. 4

Field thermographic measurements of this building revealed a thermal bridge (i.e., highly conductive path through the wall construction) due, evidently, to the penetration of the floor slab that acted to significantly increase building heat loss. Analysis of heat transfer in the vicinty of this thermal bridge is complicated by the complex two-dimensional nature of the geometry and the air space between the steel beam and the outer wall.

Two models of the wall section were considered. In the first model, Model 1, an assemblage of two-dimensional isoparametric elements was used to model the simpler construction from the center of the concrete slab upward, Fig. B8; heat transfer was assumed to be symmetric about the slab centerline. In the second model, Model 2, an assemblage of two-dimensional isoparametric elements were used to model, in detail, the geometry and discontinuous material properties of the actual construction and a subassembly of simple resistance elements was used to model convective/radiative heat transfer in the air space adjacent to the steel beam, Fig. B9. The response of both models to a steady state temperature drop of 30.5°F, from inside-air to outside-air, and a steady excitation of a harmonic variation of outside air of a 12.5°F amplitude and 24 hour period were computed.

The response of both models was similar. Representative results, for the temperature field through the construction, are shown in Fig. B7. It is seen that the steady state response dominates the behavior of the construction and the thermal path provided by the slab is evident. Surface fluxes, calculated from the computed results closely agreed with measured behavior and other analytical results.

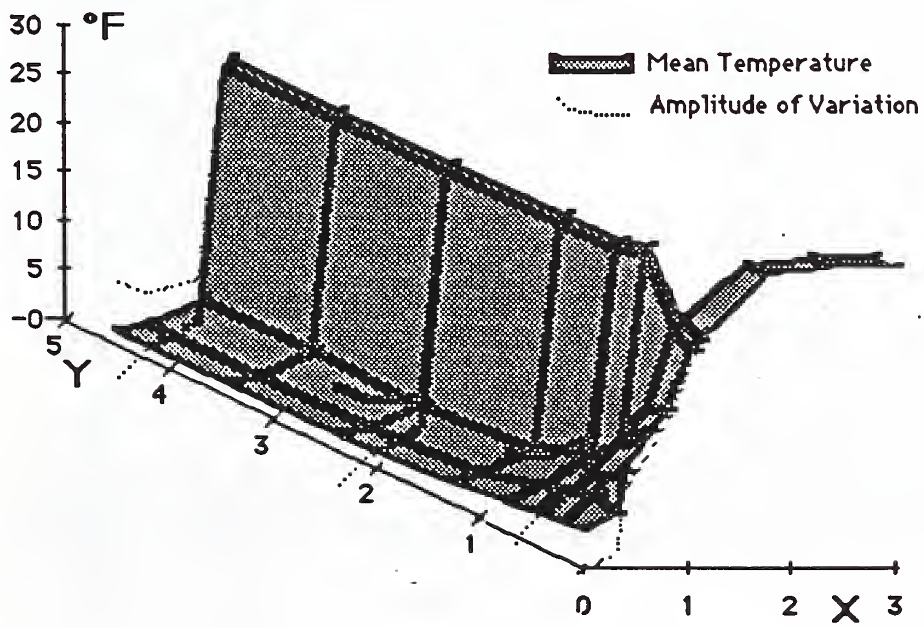
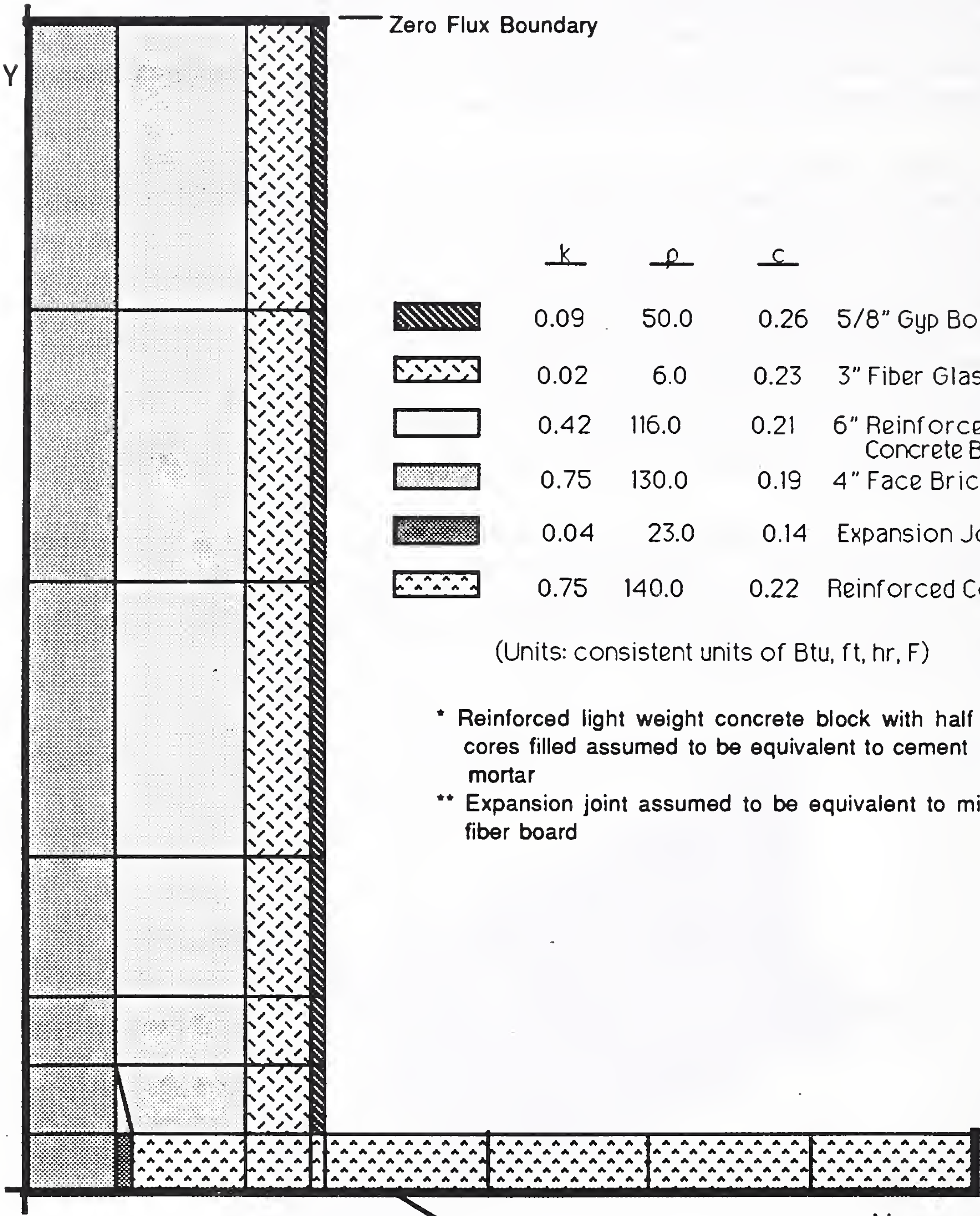


Fig. B7 Computed Temperature Field for Model 1

Model 1 employed 34 elements connected at 46 nodes and Model 2 employed 105 elements connected at 139 nodes. Computation to form and assemble the element equations for Model 2 took 73 seconds and solution for both the steady state and steady harmonic excitation took an additional 48 seconds using a Macintosh microcomputer with a MC 68000 microprocessor and 512K of central memory.



	k	ρ	c	
(Gyp Board pattern)	0.09	50.0	0.26	5/8" Gyp Board
(Fiber Glass Board pattern)	0.02	6.0	0.23	3" Fiber Glass Board
(Concrete Block pattern)	0.42	116.0	0.21	6" Reinforced Lt.Wt. Concrete Block*
(Face Brick pattern)	0.75	130.0	0.19	4" Face Brick
(Expansion Joint pattern)	0.04	23.0	0.14	Expansion Joint**
(Reinforced Concrete pattern)	0.75	140.0	0.22	Reinforced Concrete

(Units: consistent units of Btu, ft, hr, F)

* Reinforced light weight concrete block with half of cores filled assumed to be equivalent to cement mortar

** Expansion joint assumed to be equivalent to mineral fiber board

Fig. B8 Huron Building Wall Section Model 1

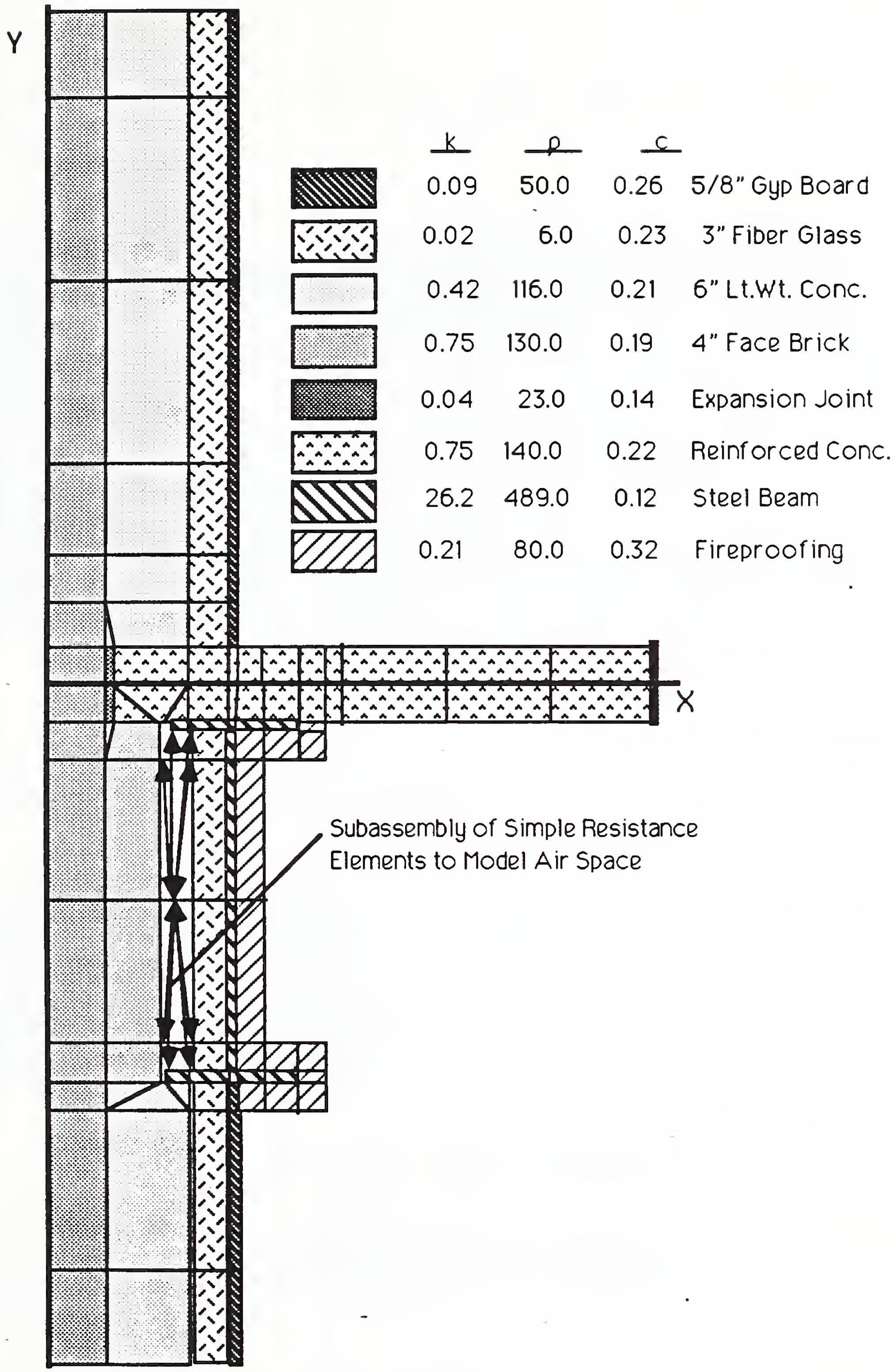


Fig. B9 Huron Building Wall Section Model 2

Appendix C - Program Structure

The program DTAM1 is structured in a manner similar to a command processor. Structurally it was developed as a collection of *trees* of subroutines linked to the main program through *roots* .

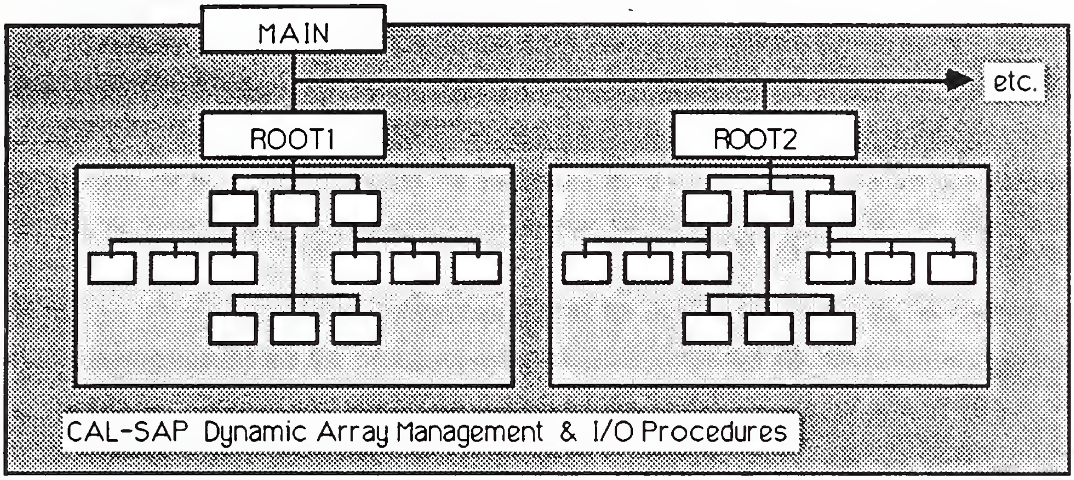


Fig. C1 Generic Trees Structure

Each tree was developed to be largely independent, dynamically defining arrays and seeking only the input data that it alone requires.

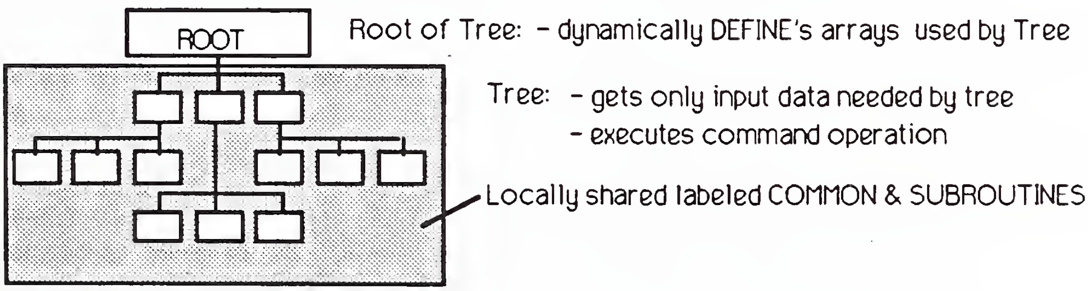


Fig. C2 Generic Tree of Subroutines

This independence allows incremental processing of input data and reporting of results so that a single error in the command/data input data file will not result in program termination. It is also hoped that the hierarchical structure will help to facilitate future program development efforts.

The CAL-SAP development software subroutines [2] are used by all trees for dynamic array management and input data interpretation. The CAL-SAP dynamic array management is accomplished by storing array values and a directory to those values in a single blank common vector IA(MTOT) as indicated below in Fig. C3. One may change the array capacity of the program by simply resetting the variable MTOT and redimensioning IA(MTOT) in the main program declaration.

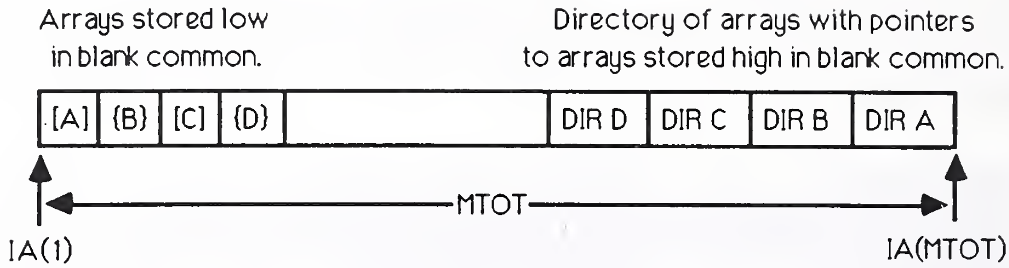


Fig. C3 CAL-SAP Dynamic Array Management

Specifically, the main program DTAM1 calls six trees of subroutines;

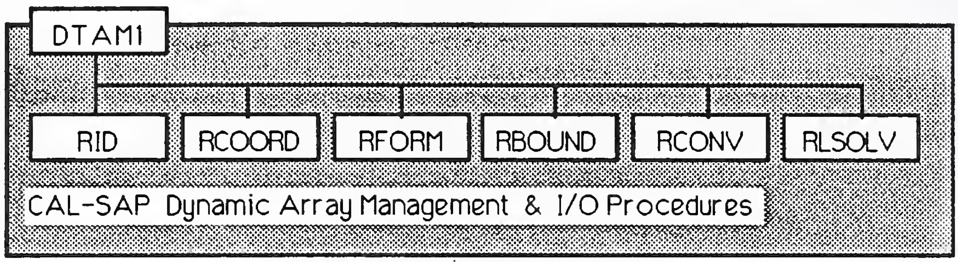


Fig. C4 Structure of DTAM1

RID is the root to the ID Tree that establishes node-number to equation-number correspondance, accounting for any constraint input; it creates an ID array.

RCOORD is the root to the Coordinate Tree that inteprets coordinate input data and generates nodal coordinates; it forms the XYZ array.

RFORM is the root to the FORM Tree that forms element arrays and assembles them to form system arrays; it forms the system conductance matrix, K(NEQN,MBAND), the system capacitance matrix, C(NEQN,MBAND), and the internal generation heat flow rate vector, EO(NEQN) - where NEQN=the number of equations and MBAND=the system (half) bandwidth.

RBOUND is the root to the BOUND Tree that establishes boundary conditions by modifying the ID array.

RCONV is the root to the CONV Tree that establishes convective boundary conditions by modifying the system conductance matrix and stores convective coefficients in array CH(NNOD), where NNOD = the number of nodes in the system.

RLSOLV is the root to the LSOLV Tree that interprets solution specification and excitation data and implements solution by calling STEADY, HARMON, and/or PREDIC to affect steady state, steady harmonic, or predictor-corrector solution options.

APPENDIX D
FORTRAN 77 SOURCE CODE

The FORTRAN77 source code for DTAM1 is listed below.

```
      PROGRAM DTAM1
C*******************************************************************
C--PRO:DTAM1 - A BUILDING ENERGY SIMULATION PROGRAM BASED ON A
C      DISCRETE THERMAL ANALYSIS METHOD FOR LINEAR IDEALIZATIONS
C
C----- DEVELOPED BY JAMES AXLEY, CORNELL UNIVERSITY
C                  NBS BUILDING PHYSICS DIV, 1985-86
C      USING;
C       A) CAL-SAP LIBRARY OF SUBROUTINES DEVELOPED BY ED WILSON,
C          U.C. BERKELEY
C       B) MacFortran V2.0 & MicroSoft Fortran V2.1 COMPILERS
C          ANSI FORTRAN 77 STANDARD
C          *INCLUDING;
C          1. USE OF SAVE /commonname/ TO RETAIN DEFINITION OF
C          LOCAL, SUBROUTINE VARIABLES AND SOME NAMELIST VARIABLES
C          *EXCEPT;
C          1. TYPE '...': MacFortran EXTENSION "TYPE" USED FOR
C          PROMPTS TO ALLOW IN-LINE RESPONSES
C*******************************************************************
C
      IMPLICIT REAL*8 (A-H,O-Z)
C
C**** CAL-SAP: DATA & COMMON STORAGE ********************************
C
      CHARACTER FIN*1,EXT*1
      COMMON MTOT,NP,IA(50000)
      COMMON /DBSYS/ NUMA,NEXT,IDIR,IP(3)
      COMMON /IOLIST/ NTM,NTR,NIN,NOT,ND1,ND2,ND3,ND4
      COMMON /PARC/ FIN(12),EXT(80)
C
C---- D I C T I O N A R Y   O F   V A R I A B L E S -------------------
C
C     VARIABLE      DESCRIPTION-----------------------------------------
C     MTOT          SIZE OF BLANK COMMON VECTOR IA
C     NP            CURRENT DATA TYPE: 1=INTEGER; 2=REAL; 3=LOGICAL
C     IA(MTOT)      BLANK COMMON VECTOR
C     NUMA          NUMBER OF ARRAYS IN BLANK COMMON DATA BASE
C     NEXT          NEXT AVAILABLE STORAGE LOCATION IN BLANK COMMON
C     IDIR          START OF DIRECTORY IN BLANK COMMON
C     IP(3)         NUMBER OF LOGICALS IN EACH DATA TYPE
C     NTM           LOGICAL UNIT NUMBER FOR TERMINAL/SCREEN
C     NTR           LOGICAL UNIT NUMBER FOR TERMINAL/KEYBOARD
C     NIN           LOGICAL UNIT NUMBER FOR INPUT DATA FILE
C     NOT           LOGICAL UNIT NUMBER FOR OUTPUT DATA FILE
C     ND1 thru ND4  LOGICAL UNIT NUMBERS FOR GENERAL USE
C     FIN(12)       INPUT DATA FILE NAME
C     EXT(80)       FILE EXTENSION NAME FOR GENERAL USE
C*******************************************************************
C
C**** LINEAR: DATA & COMMON STORAGE *********************************
C
      LOGICAL*1 ERR
      CHARACTER YESNO*1, SAVE*4
C
C---- D I C T I O N A R Y   O F   V A R I A B L E S -------------------
C
C     VARIABLE      DESCRIPTION-----------------------------------------
C     ERR           DO-WHILE TERMINATOR FLAG
C     YESNO         CONTINUATION FLAG
C     NNOD          NUMBER OF NODES IN SYSTEM
C     NEQN          NUMBER OF (SYSTEM) EQUATIONS
C     MBAN          (HALF) BANDWIDTH OF SYSTEM EQUATIONS
C
C     P O I N T E R S   T O   B L A N K   C O M M O N   L O C A T I O N S
C
C     MPXYZ         XYZ(NNOD,3) : COORDINATE ARRAY
C     MPC           C(NEQN,MBAN): CAPACITY MATRIX
C     MPK           K(NEQN,MBAN): CONDUCTANCE MATRIX
C     MPEO          EO(NNOD)    : INTERNALLY GENERATED EXCITATION
VECTOR
C     MPT           T(NEQN)     : NODAL TEMPERATURE VECTOR
C     MPID          ID(NNOD)    : NODAL EQUATION NUMBERS B. C. FLAG
ARRAY
C                               : EQTN NO. = ABS(ID)
C                               : NEG. ID = TEMPERATURE PRESCRIBED
NODE
C                               : POS. ID = FLUX PRESCRIBED NODE
C     MPCH           CH(NNOD)   : COEFS TO COMPUTE EFFECT. CONVECT.
FLUX
C     MPNPR          NPR(NNOD)  : OUTPUT PRINT CONTROL ARRAY
C
C
C*******************************************************************
C
C--1.0 INITIALIZE INTERNAL VARIABLES
C
C----CAL-SAP: DATABASE
   10 MTOT = 50000
      NUMA = 0
      NEXT = 1
      IDIR = MTOT
      IP(1) = 4
      IP(2) = 8
      IP(3) = 1
C----CAL-SAP: LOGICAL UNIT NUMBERS
      NTM = 9
      NTR = 9
      NIN = 10
      NOT = 11
      ND1 = 12
      ND2 = 13
      ND3 = 14
      ND4 = 15
C----LINEAR: INTERNAL VARIABLES
      ERR=.FALSE.
C
C--2.0 WRITE BANNER & OPEN INPUT AND OUTPUT DATA FILES
C
      WRITE(NTM,2200) MTOT
      CALL IFILE
      CALL POPEN('OUT ')
      WRITE(NOT,2200) MTOT
 2200 FORMAT(//
     .' ---------------------------------------------------------------
     .---------------------'/
     .' | DTAM1:   Building Energy Simulation for Linear Systems
     .                    |'/
     .' ---------------------------------------------------------------
     .----------Ver-6-86--'/
     .55X,'Jim Axley - Cornell & NBS'/
     .65X,'MTOT:'I10)
C
C--3.0 WHILE ERR=.FALSE.
C
C---3.1 GET PROB LABEL & CONTROL INFORMATION
C
      CALL FINDN('DTAM1',5,KEY)
      IF(KEY.EQ.1) THEN
        WRITE(NTM,2310)
WRITE(NOT,2310)
GO TO 999
      ENDIF
 2310 FORMAT(' **** ERROR: Separator "DTAM1" not found.')

      CALL FREETY
      WRITE(NTM,2312)
      WRITE(NOT,2312)
 2312 FORMAT(/' ==== PROBLEM LABEL'/)
      CALL FREE
      CALL FREEPT

      WRITE(NTM,2316)
      WRITE(NOT,2316)
 2316 FORMAT(/' ==== PROBLEM CONTROL VARIABLES')
      NNOD=0
      MBAN=0
      CALL FREE
      CALL FREETY

      CALL FREEI('E',NNOD,1)
        IF(NNOD.LE.0) THEN
WRITE(NTM,2314)
WRITE(NOT,2314)
ERR=.TRUE.
      ENDIF
 2314 FORMAT(' **** ERROR: Number of nodes must be greater than 0.')

      CALL FREEI('D',MBAN,1)
        IF(MBAN.LE.0) MBAN = 5 + INT(SQRT(REAL(NNDO)))


      WRITE(NOT,2318) NNOD,MBAN
 2318 FORMAT(/
     .'       Number of nodes ..............',I5,/
     .'       Maximum probable bandwidth ...',I5)


      CALL FREE
      CALL FREEH(' ',SAVE,4,1)

      IF(ERR) GO TO 999
C
C---3.2 ESTABLISH EQUATION NUMBERS
C
      CALL RID(MPID,NNOD,NEQN,ERR)
      IF(ERR) GO TO 999
C
C---3.3 GET & GENERATE NODAL COORDINATES
C
      CALL RCOORD(MPXYZ,NNOD,ERR)
      IF(ERR) GO TO 999
C
C---3.4 FORM ELEMENT ARRAYS AND ADD TO SYSTEM ARRAYS
C
        CALL TIME(ITIME1)
      CALL RFORM(MPID,MPXYZ,MPC,MPK,MPEO,NNOD,NEQN,MBAN,ERR)
        CALL TIME(ITIME2)

WRITE(NTM,2340) (ITIME2-ITIME1)

WRITE(NOT,2340) (ITIME2-ITIME1)
 2340   FORMAT(/'    -- NOTE: Time to form equations: ',I5,' seconds.')
```

```
      IF(ERR) GO TO 999
C
C---3.5 GET BOUNDARY CONDITION FLAGS
C
      CALL RBCOND(MPID,NNOD,ERR)
      IF(ERR) GO TO 999
C
C---3.6 MODIFY K(NEQN,MBAN) FOR CONVECTIVE TRANSFER
C
      CALL RCONV(MPID,MPXYZ,MPK,MPCH,NNOD,NEQN,MBAN,ERR)
      IF(ERR) GO TO 999
C
C---3.7 SAVE SYSTEM ARRAY DATABASE
C
      IF(SAVE.EQ.'SAVE') CALL SAVE
C
C---3.8 SOLVE SYSTEM OF EQUATIONS
C
      CALL RLSOLV(MPID,MPK,MPC,MPEO,MPCH,NNOD,NEQN,MBAN,ERR)
C
C--4.0 CLOSE INPUT AND OUTPUT DATA FILES
C
 999  CALL FCLOSE(NOT)
      CALL FCLOSE(NIN)
C
C--5.0 SOLICIT ANOTHER PROBLEM
C
      TYPE 2500
      READ(NTM,2510) YESNO
      IF((YESNO.EQ.'Y').OR.(YESNO.EQ.'y')) GO TO 10
 2500 FORMAT(/'   ** Do you want to consider another problem? (Y/N) ')
 2510 FORMAT(1A1)

      STOP
      END

C*****************************************************************
C TREE: ID - TREE OF SUBROUTINES TO ESTABLISH EQUATION NUMBERS
C*****************************************************************
C-----------------------------------------------------------------RID
      SUBROUTINE RID(MPID,NNOD,NEQN,ERR)
C--SUB:RID - ROOT TO ESTABLISB THE NODE NUMBER-EQUATION NUMBER
C            CORRESPONDANCE
C *** PRESENTLY IMPOSES EQUALITY CONSTRAINTS AND NUMBERS EQUATIONS
C *** IN NODE NUMBER ORDER
C *** TO BE USED SUBSEQUENTLY TO:
C ***       1. OPTIMIZE EQUATION NUMBERING TO MINIMIZE BANDWIDTH
C ***       2. ALLOW USER REDEFINITION OF NUMBERING
C
C---- CAL-SAP: DATA & COMMON STORAGE
C
      COMMON MTOT,NP,IA(1)
      COMMON /DBSYS/ NUMA,NEXT,IDIR,IP(3)
      COMMON /IOLIST/ NTM,NTR,NIN,NOT,ND1,ND2,ND3,ND4
C
C---- COORD: DATA & COMMON STORAGE
C
      LOGICAL*1 ERR

      SAVE /DBSYS/,/IOLIST/
C
C--0.0 WRITE HEADER
C
      WRITE(NTM,2000)
      WRITE(NOT,2000)
 2000 FORMAT(/' ==== EQUALITY CONSTRAINT CONDITIONS'/)

C
C--1.0 DEFINE ID ARRAY & INITIALIZE
C
      CALL DELETE('ID  ')
      CALL DEFINI('ID  ',MPID,NNOD,1)
      CALL ZEROI(IA(MPID),NNOD,1)
C
C---2.0 FIND SEPARATOR <CONSTR>AIN; NUMBER IN NODE ORDER IF NOT FOUND
C
      CALL FINDN('CONSTR',6,KEY)
      IF(KEY.EQ.1) THEN
        WRITE(NTM,2100)
        WRITE(NOT,2100)

NEQN = NNOD

DO 20 N=MPID,MPID+NNOD
   20   IA(N) = N-MPID+1
        RETURN
      ENDIF
 2100 FORMAT('   -- NOTE: Constraint condition data not found.'/
     .'              No DOFs will be constrained to be equal.')
      CALL FREETY

C
C--3.0 CALL ID TO DO THE WORK
C
      CALL ID(IA(MPID),NNOD,NEQN,ERR)

      RETURN
      END

C------------------------------------------------------------------ID
      SUBROUTINE ID(ID,NNOD,NEQN,ERR)
C-SUB:ID - READS AND GENERATES EQUALITY CONSTRAINT DATA
C          AND NUMBERS DOFS VIA NODE ID ARRAY
C
      IMPLICIT REAL*8 (A-B,O-Z)
C
C---- CAL-SAP: DATA & COMMON STORAGE
C
      COMMON /IOLIST/ NTM,NTR,NIN,NOT,ND1,ND2,ND3,ND4
C
C---- ID: DATA & COMMON STORAGE
C
      INTEGER MSTR, MNODE, MDOF
      CHARACTER ENDFLAG*3, BC*1
      DIMENSION ID(NNOD), IJK(3)
      LOGICAL*1 ERR

      SAVE /IOLIST/

C
C--1.0 READ CONSTRAINT DATA AND GENERATE CONSTRAINT FLAGS
C
   10 CALL FREE
      CALL FREETY
C---- CHECK FOR "END"
      CALL FREEH(' ',ENDFLAG,3,1)
      IF(ENDFLAG.EQ.'END') GO TO 20
C---- GET MASTER NODE
      CALL FREEI('M',MSTR,1)
      IF(MSTR.LE.O.OR.MSTR.GT.NNOD) THEN
        WRITE(NTM,2100) MSTR

WRITE(NOT,2100) MSTR

ERR = .TRUE.
      ENDIF
 2100 FORMAT(' **** ERROR: Master node',I5,' is out of range.')
C---- GET SLAVE NODE GENERATION DATA
      CALL FREEI(' ',IJK(1),3)
      IF(IJK(2).EQ.0) IJK(2)=IJK(1)
      IF(IJK(3).EQ.0) IJK(3)=1
      WRITE(NOT,2110) MSTR,IJK
 2110 FORMAT('     Master Node = ',I3,' Slave Nodes = ',I3,' to ',I3,
     +' step ',I3)
      DO 12 N=IJK(1),IJK(2),IJK(3)
        IF(N.LE.0.OR.N.GT.NNOD) THEN
          WRITE(NTM,2120) N
          WRITE(NOT,2120) N
          ERR=.TRUE.
          GO TO 10
        ENDIF
   12 ID(N) = -MSTR
      GO TO 10
 2120 FORMAT(' **** ERROR: (Generated) Node',I5,' is out of range.')

C
C--2.0 NUMBER UNCONSTRAINED DOF (ID=0)
C
   20 IF(ERR) RETURN
      NEQN = 0
      DO 22 N=1,NNOD
      IF(ID(N).EQ.0) THEN
        NEQN = NEQN + 1

ID(N) = NEQN
      ENDIF
   22 CONTINUE
C
C--3.0 NUMBER CONSTRAINED DOF (ID=NEG) TO MASTER NODE (MNODE) DOF
(MDOF)
C
      DO 30 N=1,NNOD
      IF(ID(N).LT.0) THEN
        MNODE=ABS(ID(N))

MDOF = ID(MNODE)

IF(MDOF.LT.O) THEN

  WRITE(NTM,2300) N,MNODE

  WRITE(NOT,2300) N,MNODE

  ERR = .TRUE.

ELSE

  ID(N) = MDOF

ENDIF
      ENDIF
   30 CONTINUE
 2300 FORMAT(' **** ERROR: Slave node ',I5,' is constrained to
     + slave node ',I5,/
     +'              Multilevel constraints are not permitted.')

      RETURN
      END

C*****************************************************************
C TREE: COORD - TREE OF SUBROUTINES TO FORM XYZ COORDINATE ARRAY
C*****************************************************************
C--------------------------------------------------------------RCOORD
      SUBROUTINE RCOORD(MPXYZ,NNOD,ERR)
C--SUB: RCOORD - ROOT SUBROUTINE TO COORD
C        COORD  - READS COORDINATE INFORMATION AND FORMS X & Y ARRAYS
C
```

```
C---- D I C T I O N A R Y   O F   V A R I A B L E S -------------------
C
C     VARIABLE      DESCRIPTION--------------------------------------
C VARIABLES PASSED TO SUBROUTINE
C     ERR           DO-WHILE TERMINATOR FLAG
C     NNOD          NUMBER OF NODES IN SYSTEM
C     MPXYZ         POINTER TO XYZ(NNOD) IN BLANK COMMON
C     XYZ(NNOD)     COORDINATE ARRAY (ORDERED BY NODE NUMBER)
C-----------------------------------------------------------------------
C---- CAL-SAP: DATA & COMMON STORAGE
C
      COMMON MTOT,NP,IA(1)
      COMMON /DBSYS/ NUMA,NEXT,IDIR,IP(3)
      COMMON /IOLIST/ NTM,NTR,NIN,NOT,ND1,ND2,ND3,ND4
C
C---- COORD: DATA & COMMON STORAGE
C
      LOGICAL*1 ERR

      SAVE /DBSYS/,/IOLIST/
C
C--0.0 WRITE HEADER
C
      WRITE(NTM,2000)
      WRITE(NOT,2000)
 2000 FORMAT(/' ==== NODAL  COORDINATES')

C
C--1.0 FIND SEPARATOR <COOR>DINATES
C
      CALL FINDN('COORD',5,KEY)
      IF(KEY.EQ.1) THEN
        WRITE(NTM,2100)

WRITE(NOT,2100)

RETURN
      ENDIF
      CALL FREETY
C
C--2.0 DEFINE ARRAY
C
      CALL DELETE('XYZ ')
      CALL DEFINE('XYZ ',MPXYZ,NNOD,3)
C
C--3.0 INITIALIZE ARRAYS
C
      CALL ZEROR(IA(MPXYZ),NNOD,3)
C
C--4.0 CALL COORD TO DO THE WORK
C
      CALL COORD(IA(MPXYZ),NNOD,ERR)

      RETURN

 2100 FORMAT(/'   -- NOTE: Coordinate data not found.')

      END
C-----------------------------------------------------------------
COORD
      SUBROUTINE COORD(XYZ,NNOD,ERR)
C-----------------------------------------------
C     SUBROUTINE TO READ AND GENERATE COORDINATES
C     DEVELOPED BY MARC HOIT - U.C. BERKELEY
C-----------------------------------------------
C
      IMPLICIT REAL*8 (A-H,O-Z)
C
      CHARACTER ENDFLAG*3
      DIMENSION XYZ(NNOD,3),NN(3)
      COMMON /IOLIST/ NTM,NTR,NIN,NOT,ND1,ND2,ND3,ND4
      SAVE /IOLIST/
C
      WRITE(NOT,1000)
C-----INITIALIZATION -----
      X = 0.0
      Y = 0.0
      Z = 0.0
      NODE = 0
C-----READ LINE OF COORDINATE INFORMATION -----
        DO 300 I=1,NNOD
        CALL FREE

CALL FREETY

CALL FREEB(' ',ENDFLAG,3,1)

IF(ENDFLAG.EQ.'END') GO TO 400
        CALL FREEI(' ',NODE,1)
          IF(NODE.EQ.0) GO TO 400
        CALL FREER('X',X,1)
        CALL FREER('Y',Y,1)
        CALL FREER('Z',Z,1)
C-----STORE COORDINATES -----
        XYZ(NODE,1) = X
        XYZ(NODE,2) = Y
        XYZ(NODE,3) = Z
C-----READ AND SET GENERATION PARAMETERS -----
          NN(1) = 0
          NN(3) = 0
          CALL FREEI('N',NN,3)
          N1 = NN(1)
            IF(N1.EQ.0) GO TO 300
        N2 = NN(2)
        N3 = NN(3)
          IF(N1.EQ.N2) GO TO 300
          IF(N2.EQ.0) GO TO 300
          IF(N2.EQ.N1) GO TO 300
          IF(N3.EQ.0) N3 = 1
          IF(N1.GT.NNOD) N1 = NNOD
          IF(N2.GT.NNOD) N2 = NNOD
        NDIF = (N2-N1)/N3
        DIF = DBLE(NDIF)
        XDIF = ( XYZ(N2,1) - XYZ(N1,1) )/DIF
        YDIF = ( XYZ(N2,2) - XYZ(N1,2) )/DIF
        ZDIF = ( XYZ(N2,3) - XYZ(N1,3) )/DIF
C-----GENERATE ADDITIONAL COORDINATES -----
        XX = XYZ(N1,1)
        YY = XYZ(N1,2)
        ZZ = XYZ(N1,3)
        N1 = N1 + N3
        N2 = N2 - N3
          DO 200 J=N1,N2,N3
          XX = XX + XDIF
          XYZ(J,1) = XX
          YY = YY + YDIF
          XYZ(J,2) = YY
          ZZ = ZZ + ZDIF
          XYZ(J,3) = ZZ
  200     CONTINUE
  300   CONTINUE
C-----PRINT FINAL COORDINATES -----
  400 DO 500 I=1,NNOD
      X = XYZ(I,1)
      Y = XYZ(I,2)
      Z = XYZ(I,3)
      WRITE(NOT,2000) I,X,Y,Z
  500 CONTINUE
C
  900 RETURN
C----------------------- FORMAT SPECIFICATIONS ----------------------
C
 1000 FORMAT(/
     1 5X,' NODE         X-COORD.        Y-COORD.        Z-COORD.')
C
 2000 FORMAT(7X,I3,9X,G10.3,4X,G10.3,4X,G10.3)
C
      END

C*******************************************************************
C TREE: FORM - TREE OF SUBROUTINES TO FORM AND ASSEMBLE ELEMENT EQTNS
C*******************************************************************
C-------------------------------------------------------------RFORM
      SUBROUTINE RFORM(MPID,MPXYZ,MPC,MPK,MPEO,NNOD,NEQN,MBAN,ERR)
C--SUB: RFORM  - ROOT SUBROUTINE TO ALL ELEMENT SUBOUTINES THAT
C                FORM & ASSEMBLES ELEMENT ARRAYS
C
C---- D I C T I O N A R Y   O F   V A R I A B L E S ----------------
C
C     VARIABLE      DESCRIPTION--------------------------------------
C VARIABLES PASSED TO SUBROUTINE
C     ERR           DO-WHILE TERMINATOR FLAG
C     NNOD          NUMBER OF NODES IN SYSTEM
C     NEQN          NUMBER OF (SYSTEM) EQUATIONS
C     MBAN          (HALF) BANDWIDTH OF SYSTEM EQUATIONS
C     MPEO          POINTER TO EO(NNOD) IN BLANK COMMON
C     MPK           POINTER TO K(NEQN,MBAN) IN BLANK COMMON
C     MPC           POINTER TO C(NEQN,MBAN) IN BLANK COMMON
C     MPXYZ         POINTER TO XYZ(NNOD,3) IN BLANK COMMON
C     MPID          POINTER TO ID(NNOD) IN BLANK COMMON
C     EO(NNOD)      INITIAL EXCITATION VECTOR
C     C(NEQN,MBAN)  SYSTEM CAPACITY MATRIX (COMPACT FORM)
C     K(NEQN,MBAN)  SYSTEM CONDUCTANCE TRANSFER MATRIX (COMPACT FORM)
C     ID(NNOD)      EQUATION NUMBER/B.C. FLAG ARRAY;
C                   ABS(ID(N)) = EQUATION NUMBER OF NODE N
C                   SIGN(ID(N)) = -1 = TEMPERATURE PRESCRIBED NODE
C                   SIGN(ID(N)) = +1 = FLUX PRESCRIBED NODE
C     XYZ(NNOD,3)   COORDINATE ARRAY (ORDERED BY NODE NUMBER)
C-------------------------------------------------------------------
C
C---- CAL-SAP: DATA & COMMON STORAGE
C
      COMMON MTOT,MP,IA(1)
      COMMON /DBSYS/ NUMA,NEXT,IDIR,IP(3)
      COMMON /IOLIST/ NTM,NTR,NIN,NOT,ND1,ND2,ND3,ND4
C
C---- RFORM: DATA & COMMON STORAGE
C
      LOGICAL*1 ERR, NOELEM

      SAVE /DBSYS/,/IOLIST/
C
C--0.0 WRITE HEADER
C
      WRITE(NTM,2000)
      WRITE(NOT,2000)
 2000 FORMAT(/' ==== ELEMENT DATA')

C
C--1.0 DEFINE ARRAYS
C
      CALL DELETE('C   ')
      CALL DELETE('K   ')
      CALL DELETE('EO  ')
      CALL DEFINE('EO  ',MPEO,NNOD,1)
      CALL DEFINE('K   ',MPK,NEQN,MBAN)
      CALL DEFINE('C   ',MPC,NEQN,MBAN)
```

```
C
C--2.0 INITIALIZE ARRAYS
C
      CALL ZEROR(IA(MPEO),NNOD,1)
      CALL ZEROR(IA(MPK),NEQN,MBAN)
      CALL ZEROR(IA(MPC),NEQN,MBAN)
C
C--3.0 FIND ELEMENTS
C
      NOELEM =.TRUE.
      MAXBAN = 0
C
C--3.1 FIND SEPARATOR "RESIST"
C
      CALL FINDN('RESIST',6,KEY)
      IF(KEY.EQ.0) THEN
        NOELEM=.FALSE.

CALL FREETY

CALL RESIST(IA(MPID),IA(MPK),NNOD,NEQN,MBAN,MAXBAN,'RES',ERR)
      ENDIF
C
C--3.2 FIND SEPARATOR "FLOWLO"
C
      CALL FINDN('FLOWLO',6,KEY)
      IF(KEY.EQ.0) THEN
        NOELEM=.FALSE.

CALL FREETY

CALL RESIST(IA(MPID),IA(MPK),NNOD,NEQN,MBAN,MAXBAN,'FLO',ERR)
      ENDIF

C
C--3.3 FIND SEPARATOR "CAPACI"
C
      CALL FINDN('CAPACI',6,KEY)
      IF(KEY.EQ.0) THEN
        NOELEM=.FALSE.

CALL FREETY

CALL CAPACI(IA(MPID),IA(MPC),NNOD,NEQN,MBAN,ERR)
      ENDIF
C
C--3.4 FIND SEPARATOR "ISO1D"
C
      CALL FINDN('ISO1D',5,KEY)
      IF(KEY.EQ.0) THEN
        NOELEM=.FALSE.

CALL FREETY

CALL ISO1D(IA(MPID),IA(MPXYZ),IA(MPC),IA(MPK),IA(MPEO),
     . NNOD,NEQN,MBAN,MAXBAN,ERR)
      ENDIF
C
C--3.5 FIND SEPARATOR "ISO2D4"
C
      CALL FINDN('ISO2D4',6,KEY)
      IF(KEY.EQ.0) THEN
        NOELEM=.FALSE.

CALL FREETY

CALL ISO2D4(IA(MPID),IA(MPXYZ),IA(MPC),IA(MPK),IA(MPEO),
     . NNOD,NEQN,MBAN,MAXBAN,ERR)
      ENDIF
C
C--3.6 FIND SEPARATOR "VNMRT"
C
      CALL FINDN('VNMRT',5,KEY)
      IF(KEY.EQ.0) THEN
        NOELEM=.FALSE.

CALL FREETY

CALL VNMRT(IA(MPID),IA(MPK),NNOD,NEQN,MBAN,MAXBAN,ERR)
      ENDIF
C
C--4.0 IF NO ELEMENTS ...
C
      IF(NOELEM) THEN
        WRITE(NTM,2400)

WRITE(NOT,2400)

ERR=.TRUE.
      ENDIF
C
C--5.0 REPORT BANDWIDTH NEEDED
C
      WRITE(NTM,2500) MAXBAN,MBAN
      WRITE(NOT,2500) MAXBAN,MBAN

      RETURN

 2400 FORMAT(' **** ERROR: No element data found.')
 2500 FORMAT(/'   -- NOTE: Needed bandwidth .....',I5,/
     .'              Requested bandwidth ..',I5)

      END
C-----------------------------------------------------------------

RESIST
      SUBROUTINE RESIST(ID,K,NNOD,NEQN,MBAN,MAXBAN,TYPE,ERR)
C-SUB:RESIST  - FORMS & ASSEMBLES RESISTANCE OR FLOWLOOP ELEMENTS
C
      IMPLICIT REAL*8 (A-H,O-Z)

C
C
C---- CAL-SAP: DATA & COMMON STORAGE
C
      COMMON /IOLIST/ NTM,NTR,NIN,NOT,ND1,ND2,ND3,ND4
C
C---- RESIST: DATA & COMMON STORAGE
C
      REAL*8 K(NEQN,MBAN)
      INTEGER ID(NNOD), LMNEW(2), LMOLD(2)
      LOGICAL*1 ERR
      CHARACTER ENDFLAG*3,TYPE*3

      SAVE NNEW,NOLD,LMNEW,LMOLD,R,A,/IOLIST/

      NDOF = 2

C
C--1.0 WRITE ELEMENT HEADER
C
      IF(TYPE.EQ.'RES') THEN
        WRITE(NOT,2100)
      ELSEIF(TYPE.EQ.'FLO') THEN
        WRITE(NOT,2110)
      ENDIF

 2100 FORMAT(
     ./'   == RESISTANCE ELEMENTS'//
     .'      Elem     I-Node     J-Node    Resist.       Area')
 2110 FORMAT(
     ./'   == FLOW-LOOP ELEMENTS'//
     .'      Elem     I-Node     J-Node     Flow Rate    Capac.')
C
C--2.0 GET FIRST ELEMENT, FORM & ADD ELEMENT TO SYS
C
      INCR = 0
      R = 0.0
      A = 0.0
      CALL FREE
      CALL FREETY
      CALL FREEI(' ',NOLD,1)
      CALL FREEI('I',LMOLD(1),2)
      IF(TYPE.EQ.'RES') THEN
        CALL FREER('R',R,1)
        CALL FREER('A',A,1)
      ELSEIF(TYPE.EQ.'FLO') THEN
        CALL FREER('W',R,1)
        CALL FREER('C',A,1)
      ENDIF


      CALL RESIS0(ID,NOLD,LMOLD,K,R,A,NNOD,NEQN,MBAN,MAXBAN,TYPE,ERR)

C
C--3.0 GET NEXT LINE OF DATA
C
   30 CALL FREE
      CALL FREETY
C---- CHECK FOR "END"
      CALL FREEH(' ',ENDFLAG,3,1)
      IF(ENDFLAG.EQ.'END') THEN
        RETURN
      ENDIF
C---- GET NEW ELEMENT INFORMATION
      CALL FREEI(' ',NNEW,1)
      CALL FREEI('I',LMNEW(1),2)
      CALL FREEI('N',INCR,1)
      IF(INCR.EQ.0) INCR=1
      IF(TYPE.EQ.'RES') THEN
        CALL FREER('R',R,1)
        CALL FREER('A',A,1)
      ELSEIF(TYPE.EQ.'FLO') THEN
        CALL FREER('W',R,1)
        CALL FREER('C',A,1)
      ENDIF
C---- CHECK NUMERICAL ORDER
      IF(NNEW.LE.NOLD) THEN
        WRITE(NTM,2300) NNEW

WRITE(NOT,2300) NNEW

ERR=.TRUE.

RETURN
      ENDIF
C---- GENERATE MISSING ELEMENTS
      IF(NNEW.GT.NOLD+1) THEN
        DO 34 N=NOLD+1,NNEW-1,1
         DO 32 I=1,NDOF
   32    LMOLD(I) = LMOLD(I) + INCR
   34   CALL RESIS0(ID,N,LMOLD,K,R,A,NNOD,NEQN,MBAN,MAXBAN,TYPE,ERR)
      ENDIF
C---- DO NEW ELEMENT
      NOLD = NNEW
      DO 36 I=1,NDOF
   36 LMOLD(I) = LMNEW(I)
      CALL RESIS0(ID,NOLD,LMOLD,K,R,A,NNOD,NEQN,MBAN,MAXBAN,TYPE,ERR)
```

```
      GO TO 30

 2300 FORMAT(' **** ERRDR: Element number ',I5,' is out of order.')

      END

C---------------------------------------------------------------RESIS0
      SUBROUTINE
RESIS0(ID,NELM,LM,K,R,A,NNOD,NEQN,MBAN,MAXBAN,TYPE,ERR)
C-SUB:RESIS0 - REPORTS ELEM INFORMATION. CHECKS BANDWIDTH,
C               FORMS ELEM ARRAYS & ADDS THEM TO SYS ARRAYS
C
      IMPLICIT REAL*8 (A-B,O-Z)
C
C
C---- CAL-SAP: DATA & COMMON STORAGE
C
      COMMON /IOLIST/ NTM,NTR,NIN,NOT,ND1,ND2,ND3,ND4
C
C---- RESIS0: DATA & COMMON STORAGE
C
      REAL*8 K(NEQN,MBAN), S(2,2)
      INTEGER ID(NNOD), LM(2)
      LOGICAL*1 ERR
      CHARACTER TYPE*3
      SAVE /IOLIST/
C
C---- REPORT ELEMENT INFORMATION TO OUTPUT DATA FILE
C
      WRITE(NOT,2000) NELM,LM(1),LM(2),R,A
C
C---- NODE ERROR TRAP
C
      DO 200 N=1,2
      NN=LM(N)
      IF(NN.LE.0.OR.NN.GT.NNOD) THEN
        WRITE(NTM,2010) NN

WRITE(NOT,2010) NN
        ERR=.TRUE.

RETURN
  200 ENDIF
C
C---- DETERMINE ELEMENT BANDWIDTH
C
      CALL MBAND(ID,LM,2,NNOD,MBAN,MAXBAN,ERR)
C
C---- FORM ELEMENT ARRAYS
C
      IF(TYPE.EQ.'FLO') THEN
        A = R*A

R = 1.0
      ENDIF

      CALL RESIS(S,R,A)
C
C---- ADD ELEMENT CONTRIBUTION TO SYSTEM ARRAYS
C
      CALL ADDCM(ID,LM,S,K,2,2,NNOD,NEQN,MBAN)


      RETURN

 2000 FORMAT(3(5X,I5),4X,2G10.3)
 2010 FORMAT(' **** ERROR: (Generated) Node ',I5,' is out of range.')

      END

C----------------------------------------------------------------RESIS
      SUBROUTINE RESIS(XK,R,A)
C---SUB:RESIS - 2-NODE CONVENTIONAL RESISTANCE ELEM
C
      IMPLICIT REAL*8 (A-B,O-Z)
C
      DIMENSION XK(2,2)
C----- FORM 2X2 CONDUCTANCE MATRIX
      XK(1,1) = A/R
      XK(1,2) = -A/R
      XK(2,1) = -A/R
      XK(2,2) = A/R
      RETURN
      END
C-----------------------------------------------------------------
CAPACI
      SUBROUTINE CAPACI(ID,C,NNOD,NEQN,MBAN,ERR)
C-SUB:CAPACI  - FORMS & ASSEMBLES CAPACITANCE ELEMENTS
C
      IMPLICIT REAL*8 (A-H,O-Z)
C
C
C---- CAL-SAP: DATA & COMMON STORAGE
C
      COMMON /IDLIST/ NTM,NTR,NIN,NOT,ND1,ND2,ND3,ND4
C
C---- CAPACI: DATA & COMMON STORAGE
C
      REAL*8 C(NEQN,MBAN), MASS, CAP
      INTEGER ID(NNOD), LMNEW, LMOLD
      LOGICAL*1 ERR
      CHARACTER ENDFLAG*3

      SAVE NNEW,NDLD,LMNEW,LMOLD,MASS,CAP,/IDLIST/
      NDOF = 1
C
C--1.0 WRITE ELEMENT HEADER
C
      WRITE(NOT,2100)
 2100 FORMAT(
     ./'   == CAPACITANCE ELEMENTS'//
     .'      Elem    I-Node      Mass    Capac.')
C
C--2.0 GET FIRST ELEMENT, FORM & ADD ELEMENT TO SYS
C
      INCR = 0
      MASS = 0.0
      CAP = 0.0
      CALL FREE
      CALL FREETY
      CALL FREEI(' ',NOLD,1)
      CALL FREEI('I',LMOLD,1)
      CALL FREER('M',MASS,1)
      CALL FREER('C',CAP,1)

      CALL CAPAC0(ID,NDLD,LMOLD,C,MASS,CAP,NNOD,NEQN,MBAN,ERR)

C
C--3.0 GET NEXT LINE DF DATA
C
   30 CALL FREE
      CALL FREETY
C---- CHECK FOR "END"
      CALL FREEH(' ',ENDFLAG,3,1)
      IF(ENDFLAG.EQ.'END') THEN
        RETURN
      ENDIF
C---- GET NEW ELEMENT INFORMATION
      CALL FREEI(' ',NNEW,1)
      CALL FREEI('I',LMNEW,1)
      CALL FREEI('N',INCR,1)
      IF(INCR.EQ.0) INCR=1
      CALL FREER('M',MASS,1)
      CALL FREER('C',CAP,1)
C---- CHECK NUMERICAL ORDER
      IF(NNEW.LE.NOLD) THEN
        WRITE(NTM,2300) NNEW

WRITE(NDT,2300) NNEW

ERR=.TRUE.

RETURN
      ENDIF
C---- GENERATE MISSING ELEMENTS
      IF(NNEW.GT.NOLD+1) THEN
        DO 34 N=NOLD+1,NNEW-1,1
         LMOLD = LMOLD + INCR
   34   CALL CAPAC0(ID,N,LMOLD,C,MASS,CAP,NNOD,NEQN,MBAN,ERR)
      ENDIF
C---- DO NEW ELEMENT
      NOLD = NNEW
      LMOLD = LMNEW
      CALL CAPAC0(ID,NOLD,LMOLD,C,MASS,CAP,NNOD,NEQN,MBAN,ERR)

      GO TO 30

 2300 FORMAT(' **** ERROR: Element number ',I5,' is out of order.')

      END

C---------------------------------------------------------------CAPAC0
      SUBROUTINE CAPAC0(ID,NELM,LM,C,MASS,CAP,NNOD,NEQN,MBAN,ERR)
C-SUB:CAPAC0 - REPORTS ELEM INFORMATION,
C               FORMS ELEM ARRAYS & ADDS TBEM TO SYS ARRAYS
C
      IMPLICIT REAL*8 (A-B,O-Z)
C
C
C---- CAL-SAP: DATA & COMMON STORAGE
C
      COMMON /IOLIST/ NTM,NTR,NIN,NOT,ND1,ND2,ND3,ND4
C
C---- FORM: DATA & COMMON STORAGE
C
      REAL*8 C(NEQN,MBAN), MASS, CAP, MC
      INTEGER ID(NNOD), LM ,NELM
      LOGICAL*1 ERR
      SAVE /IOLIST/
C
C---- REPORT ELEMENT INFORMATION TO OUTPUT DATA FILE
C
      WRITE(NOT,2000) NELM,LM,MASS,CAP
C
C---- NODE ERROR TRAP
C
      NN=LM
      IF(NN.LE.0.OR.NN.GT.NNOD) THEN
        WRITE(NTM,2010) NN

WRITE(NOT,2010) NN
        ERR=.TRUE.

RETURN
      ENDIF
```

```
C
C---- FORM ELEMENT ARRAY
C
      MC = MASS * CAP
C
C---- ADD ELEMENT CONTRIBUTION TO SYSTEM ARRAYS
C
      CALL ADDCM(ID,LM,MC,C,1,1,NNOD,NEQN,MBAN)


      RETURN

 2000 FORMAT(2(5X,I5),4X,2G10.3)
 2010 FORMAT(' **** ERROR: (Generated) Node ',I5,' is out of range.')

      END

C-----------------------------------------------------------------
1SO1D
      SUBROUTINE ISO1D(ID,XYZ,C,K,EO,NNOD,NEQN,MBAN,MAXBAN,ERR)
C-SUB:ISO1D  - FORMS & ASSEMBLES ISOPARAM.1D 2-4-NODE FINITE ELEMENTS
C
      IMPLICIT REAL*8 (A-H,O-Z)
C
C
C---- CAL-SAP: DATA & COMMON STORAGE
C
      COMMON /IOLIST/ NTM,NTR,NIN,NOT,ND1,ND2,ND3,ND4
C
C---- ISO1D: DATA & COMMON STORAGE
C
C---- D I C T I O N A R Y   O F   V A R I A B L E S -------------------
C
C     VARIABLE       DESCRIPTION---------------------------------------
C     PK             ELEMENT CONDUCTIVITY
C     PA             AREA AVAILABLE FOR HEAT TRANSFER
C     PP             ELEMENT DENSITY
C     PC             ELEMENT CAPACITY
C     PQ             ELEMENT INTERNAL HEAT GENERATION RATE/LENGTH
C     PX(4)          ELEMENT NODE LOCAL COORDINATES
C     PR(4)          QUAD POINTS -1.0 TO 1.0
C*********************************************************************

      REAL*8 K(NEQN,MBAN),C(NEQN,MBAN),XYZ(NNOD,3),EO(NNOD),
     + PK,PA,PP,PC,PQ,PX(4),PR(4)
      INTEGER ID(NNOD), LMNEW(4), LMOLD(4)
      LOGICAL*1 ERR, GAUSS
      CHARACTER ENDFLAG*3, QUADFLAG*5

      SAVE NNEW,NOLD,LMNEW,LMOLD,PK,PA,PP,PC,PQ,PX,PR,GAUSS,/IOLIST/

C
C--1.0 WRITE ELEMENT HEADER
C
      WRITE(NOT,2100)
 2100 FORMAT(
     ./'    == ISOPARAMETRIC 1D 2 TO 4-NODE CONDUCTION ELEMENTS'//
     .'       Kx = Conductivity     P = Density      C = Capacity'/
     .'       A  = Area             Q = Int. Heat'//
     .' Elem Node Coords.    Kx        P        C        A        Q'
     .'  Quad. Points')
C
C--2.0 GET FIRST ELEMENT, FORM & ADD ELEMENT TO SYS
C
      INCR = 0
      DO 10 I=1,4
   10 PX(I) = 0.0
      PK = 0.0
      PP = 0.0
      PC = 0.0
      PA = 0.0
      PQ = 0.0
      CALL FREE
      CALL FREETY
      CALL FREEI(' ',NOLD,1)
      DO 15 I=1,4
   15 LMOLD(I) = 0
      CALL FREEI('I',LMOLD(1),4)
      NDOF = 4
      IF(LMOLD(4).LE.0) NDOF=3
      IF(LMOLD(3).LE.0) NDOF=2
      CALL FREER('X',PX(1),4)
      CALL FREER('K',PK,1)
      CALL FREER('P',PP,1)
      CALL FREER('C',PC,1)
      CALL FREER('A',PA,1)
      CALL FREER('Q',PQ,1)
      QUADFLAG = '12345'
      CALL FREEH('R',QUADFLAG,5,1)
      GAUSS = .FALSE.
      IF(QUADFLAG.EQ.'GAUSS') GAUSS = .TRUE.
      IF(.NOT.GAUSS) THEN
C     SET DEFAULT QUAD POINTS

IF(NDOF.EQ.2) THEN

  PR(1) = - SQRT(2.0D0/3.0D0)

  PR(2) = - PR(1)

ELSEIF(NDOF.EQ.3) THEN

  PR(1) = -1.0D0

  PR(2) = 0.0D0

  PR(3) = 1.0D0

ELSEIF(NDOF.EQ.4) THEN

  PR(1) = -1.0D0

  PR(2) = -1.0D0/3.0D0

  PR(3) = 1.0D0/3.0D0

  PR(4) = 1.0D0

ENDIF
C      GET QUAD POINTS FOR EVALUATION OF [C]
         CALL FREER('R',PR(1),NDOF)

DO 20 I=1,NDOF

IF(ABS(PR(I)).GT.1.0D0) THEN

  WRITE(NTM,2200)

  WRITE(NOT,2200)
   20   ENDIF
      ENDIF

 2200 FORMAT(' **** WARNING: Quadrature points should normally
     + be in the range of -1.0 to +1.0')

      CALL ISO1D0(ID,NOLD,LMOLD,C,K,EO,PX,PK,PA,PP,PC,PQ,PR,
     +NDOF,GAUSS,NNOD,NEQN,MBAN,MAXBAN,ERR)

C
C--3.0 GET NEXT LINE OF DATA
C
   30 CALL FREE
      CALL FREETY
C---- CHECK FOR "END"
      CALL FREEH(' ',ENDFLAG,3,1)
      IF(ENDFLAG.EQ.'END') THEN
        RETURN
      ENDIF
C---- GET NEW ELEMENT INFORMATION
      CALL FREEI(' ',NNEW,1)
      DO 35 I=1,4
   35 LMNEW(I) = 0
      CALL FREEI('I',LMNEW(1),4)
      NDOF = 4
      IF(LMNEW(4).LE.0) NDOF=3
      IF(LMNEW(3).LE.0) NDOF=2
      CALL FREEI('N',INCR,1)
      IF(INCR.EQ.0) INCR=1
      CALL FREER('X',PX(1),4)
      CALL FREER('K',PK,1)
      CALL FREER('P',PP,1)
      CALL FREER('C',PC,1)
      CALL FREER('A',PA,1)
      CALL FREER('Q',PQ,1)
      QUADFLAG = '12345'
      CALL FREEH('R',QUADFLAG,5,1)
      GAUSS = .FALSE.
      IF(QUADFLAG.EQ.'GAUSS') GAUSS = .TRUE.
      IF(.NOT.GAUSS) THEN
C     SET DEFAULT QUAD POINTS

IF(NDOF.EQ.2) THEN

  PR(1) = - SQRT(2.0D0/3.0D0)

  PR(2) = - PR(1)

ELSEIF(NDOF.EQ.3)  THEN

  PR(1) = -1.0D0

  PR(2) = 0.0D0

  PR(3) = 1.0D0

ELSEIF(NDOF.EQ.4) THEN

  PR(1) = -1.0D0

  PR(2) = -1.0D0/3.0D0

  PR(3) = 1.0D0/3.0D0

  PR(4) = 1.0D0

ENDIF
C      GET QUAD POINTS FOR EVALUATION OF [C]
         CALL FREER('R',PR(1),NDOF)

DO 40 I=1,NDOF

IF(ABS(PR(I)).GT.1.0D0) THEN

  WRITE(NTM,2200)

  WRITE(NOT,2200)
   40   ENDIF
```

```
      ENDIF

C---- CHECK NUMERICAL ORDER
      IF(NNEW.LE.NOLD) THEN
        WRITE(NTM,2300) NNEW

WRITE(NOT,2300) NNEW

ERR=.TRUE.

RETURN
      ENDIF
C---- GENERATE MISSING ELEMENTS
      IF(NNEW.GT.NOLD+1) THEN
        DO 50 N=NOLD+1,NNEW-1,1
         DO 45 I=1,NDOF
   45    LMOLD(I) = LMOLD(I) + INCR
   50   CALL ISO1D0(ID,N,LMOLD,C,K,EO,PX,PK,PA,PP,PC,PQ,PR,
     +  NDOF,GAUSS,NNOD,NEQN,MBAN,MAXBAN,ERR)
      ENDIF
C---- DO NEW ELEMENT
      NOLD = NNEW
      DO 55 I=1,NDOF
   55 LMOLD(I) = LMNEW(I)
      CALL ISO1D0(ID,NOLD,LMOLD,C,K,EO,PX,PK,PA,PP,PC,PQ,PR,
     +NDOF,GAUSS,NNOD,NEQN,MBAN,MAXBAN,ERR)

      GO TO 30

 2300 FORMAT(' **** ERROR: Element number ',I5,' is out of order.')

      END

C----------------------------------------------------------------ISO1D0
      SUBROUTINE ISO1D0(ID,NELM,LM,C,K,EO,PX,PK,PA,PP,PC,PQ,PR,
     +NDOF,GAUSS,NNOD,NEQN,MBAN,MAXBAN,ERR)
C-SUB:ISO1D0 - REPORTS ELEM INFORMATION, CHECKS BANDWIDTH,
C              FORMS ELEM ARRAYS & ADDS THEM TO SYS ARRAYS
C
      IMPLICIT REAL*8 (A-H,O-Z)
C
C
C---- CAL-SAP: DATA & COMMON STORAGE
C
      COMMON /IOLIST/ NTM,NTR,NIN,NOT,ND1,ND2,ND3,ND4
C
C----ISO1D0: DATA & COMMON STORAGE
C
      REAL*8 C(NEQN,MBAN),K(NEQN,MBAN),EO(NNOD),CE(4,4),KE(4,4),
     +EE(4),PX(4),PR(4)
      INTEGER ID(NNOD), LM(4)
      LOGICAL*1 ERR,GAUSS
      SAVE /IOLIST/
C
C---- REPORT ELEMENT INFORMATION TO OUTPUT DATA FILE
C
      IF(GAUSS) THEN
        WRITE(NOT,2000) NELM,LM(1),PX(1),PK,PP,PC,PA,PQ

WRITE(NOT,2005) (LM(I),PX(I),I=2,NDOF)
      ELSE
        WRITE(NOT,2010) NELM,LM(1),PX(1),PK,PP,PC,PA,PQ,
     +  (PR(I),I=1,NDOF)

WRITE(NOT,2005) (LM(I),PX(I),I=2,NDOF)
      ENDIF
 2000 FORMAT(2I5,6G8.2E1,'GAUSS')
 2005 FORMAT(5X,I5,G8.2)
 2010 FORMAT(2I5,6G8.2E1,4F6.3)
C
C---- NODE ERROR TRAP
C
      DO 200 N=1,NDOF
      NN=LM(N)
      IF(NN.LE.0.OR.NN.GT.NNOD) THEN
        WRITE(NTM,2020) NN

WRITE(NOT,2020) NN
        ERR=.TRUE.

RETURN
  200 ENDIF

 2020 FORMAT(' **** ERROR: (Generated) Node ',I5,' is out of range.')
C
C---- DETERMINE ELEMENT BANDWIDTH
C
      CALL MBAND(ID,LM,NDOF,NNOD,MBAN,MAXBAN,ERR)
C
C---- FORM ELEMENT ARRAYS
C
      CALL ISO241(KE,CE,EE,PK,PA,PP,PC,PQ,PX,NDOF,PR,GAUSS)
C
C---- ADD ELEMENT CONTRIBUTION TO SYSTEM ARRAYS
C
      CALL ADDCM(ID,LM,KE,K,NDOF,4,NNOD,NEQN,MBAN)
      CALL ADDCM(ID,LM,CE,C,NDOF,4,NNOD,NEQN,MBAN)

      DO 300 N=1,NDOF
  300 EO(LM(N)) = EE(N)


      RETURN
      END
```

```
C-----------------------------------------------------------------
ISO241
      SUBROUTINE ISO241(KE,CE,EE,PK,PA,PP,PC,PQ,PX,NELNOD,RPT,GAUSS)
C--SUB:ISO241 - 2 TO 4 NODE ISOPARAMETRIC CONDUCTION FINITE ELEMENT
C
C     NODE NUMBERING   1---3---4---2
C
C---- D I C T I O N A R Y   O F   V A R I A B L E S -----------------
C
C     VARIABLE            DESCRIPTION
C DUMMY VARIABLES
C     NELNOD              NUMBER OF ELEMENT NODES [2 TO 4]
C     NIP                 NUMBER OF INTEGRATION POINTS
C     KE(NELNOD,NELNOD)   ELEMENT CONDUCTANCE MATRIX
C     CE(NELNOD,NELNOD)   ELEMENT CAPACITANCE MATRIX
C     EE(NELNOD)          ELEMENT INTERNAL GENERATION RATE VECTOR
C     PK                  CONDUCTIVITY
C     PA                  AREA AVAILABLE FOR HEAT TRANSFER
C     PP                  WEIGHT DENSITY
C     PC                  HEAT CAPACITY
C     PQ                  INTERNAL GENERATION RATE PER UNIT VOL.
C     PX(NELNOD)          (GLOBAL) COORDINATES OF ELEM. NODAL POINTS
C     RPT(4)

 QUAD. POINTS USED TO EVALUATE [C]
C     GAUSS

 QUADRATURE METHOD TO EVALUATE [C]
C


 = .TRUE. USE GAUSS QUADRATURE
C



 = .FALSE. USE GIVEN QUAD POINTS
C INTERNAL VARIABLES
C     QPT(4)
GAUSS QUAD. POINTS
C     QWT(4)
GAUSS QUAD. WEIGHTS
C     H(4)
ELEMENT SHAPE FUNCTION VALUES
C     DEL(4)
ELEMENT LOCAL DERIVATIVE VALUES
C     R                  LOCAL COORDINATE
C     JACOB              JACOBIAN
C
      IMPLICIT REAL*8 (A-H,O-Z)
      REAL*8 KE(4,4),CE(4,4),EE(4),PX(4), RPT(4)
      REAL*8 JACOB, QPT(4), QWT(4), H(4), DEL(4)
      LOGICAL*1 GAUSS

C-1.0 ZERO ELEMENT ARRAYS

      DO 10 I=1,4
      EE(I) = 0.0D0
      DO 10 J=1,4
      KE(I,J) = 0.0D0
      CE(I,J) = 0.0D0
   10 CONTINUE

C-2.0 EVALUATE CONDUCTANCE MATRIX AND INTERNAL GENERATION RATE VECTOR
C-2.1 GET GAUSS QUAD POINTS AND WEIGHTS
C     NOTE: USE NIP=NELNOD-1 FOR EXACT INTEGRATION OF [KE] & {EE}
      NIP = NELNOD - 1
      CALL GAUSCO(NIP,QPT,QWT)
      DO 30 IP=1,NIP
C-2.2  GET SHAPE FUNCTION VALUES AT QUAD POINT
        CALL SHP241(QPT(IP),H,NELNOD)
C-2.3  GET LOCAL DERIVATIVE VALUES AT QUAD POINT
        CALL DER241(QPT(IP),DEL,NELNOD)
C-2.4  COMPUTE JACOBIAN AT THE POINT (NOTE: FOR 1D JACOBIAN IS 1X1)
        CALL JACOB1(NELNOD,DEL,PX,JACOB)
C-2.5  ACCUMULATE QUADRATURE CONTRIBUTION TO ELEMENT MATRICES
       DO 20 I=1,NELNOD
C      ELEMENT INTERNAL GENERATION RATE VECTOR
        EE(I) = EE(I) + QWT(IP)*PQ*H(I)*JACOB
       DO 20 J=1,NELNOD
C      ELEMENT CONDUCTANCE MATRIX
        KE(I,J) = KE(I,J) + QWT(IP)*DEL(I)*PK*DEL(J)/JACOB
   20  CONTINUE
   30 CONTINUE

C-3.0 EVALUATE CAPACITANCE MATRIX
C-3.1 GET QUAD POINTS AND WEIGHTS
C     NOTE: USE NIP=NELNOD FOR EXACT INTEGRATION OF [C]
      NIP = NELNOD
C     USE GAUSS QUADRATURE IF SPECIFIED
C     EVALUATE WEIGHT COEFFICIENTS IF QUAD POINTS ARE SPECIFIED
        IF(GAUSS)       CALL GAUSCO(NIP,RPT,QWT)
        IF(.NOT.GAUSS) CALL QUADCO(NIP,QWT,RPT)
      DO 50 IP=1,NIP
C-3.2  GET SHAPE FUNCTION VALUES AT QUAD POINT
        CALL SHP241(RPT(IP),H,NELNOD)
C-3.3  GET LOCAL DERIVATIVE VALUES AT QUAD POINT
        CALL DER241(RPT(IP),DEL,NELNOD)
C-3.4  COMPUTE JACOBIAN AT THE POINT (NOTE: FOR 1D JACOBIAN IS 1X1)
        CALL JACOB1(NELNOD,DEL,PX,JACOB)
C-3.5  ACCUMULATE QUADRATURE CONTRIBUTION TO ELEMENT MATRICES
       DO 40 I=1,NELNOD
       DO 40 J=1,NELNOD
        CE(I,J) = CE(I,J) + QWT(IP)*H(I)*PP*PC*H(J)*JACOB
```

```
   40 CONTINUE
   50 CONTINUE

      RETURN
      END

C-----------------------------------------------------------------------
SHP241
      SUBROUTINE SHP241(R,H,NELNOD)
C-SUB:SHP24L - DETERMINES THE VALUE OF 2-4 NODE ISOPARAMETRIC 1D
C     SHAPE FUNCTIONS, H, AT LOCAL COORDINATE R
C
C     NODE NUMBERING  1---3---4---2
C
C---- D I C T I O N A R Y  O F  V A R I A B L E S ------------------
C
C     VARIABLE      DESCRIPTION
C     R             LOCAL COORDINATE -1 TO +1
C     H(2 TO 4)     VALUE OF SHAPE FUNCTIONS
C     NELNOD        NUMBER OF ELEMENT NODES [2 TO 4]
C
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION H(NELNOD)

        H(1) = 0.5*(1.0 - R)
        H(2) = 0.5*(1.0 + R)
      IF(NELNOD.GE.3) THEN
        H(1) = H(1) - 0.5*(1.0 - R*R)

H(2) = H(2) - 0.5*(1.0 - R*R)

H(3) = (1.0 - R*R)
      ENDIF
      IF(NELNOD.EQ.4) THEN
        H(1) = H(1) + (-9.0*R*R*R + R*R + 9.0*R - 1.0)/16.0

H(2) = H(2) + (9.0*R*R*R + R*R - 9.0*R - 1.0)/16.0

H(3) = H(3) + (27.0*R*R*R + 7.0*R*R - 27.0*R - 7.0)/16.0

H(4) = (-27.0*R*R*R - 9.0*R*R + 27.0*R + 9.0)/16.0
      ENDIF
      RETURN
      END

C-----------------------------------------------------------------------
DER241
      SUBROUTINE DER241(R,DEL,NELNOD)
C-SUB:DER241 - DETERMINES THE VALUE OF LOCAL DERIVATIVES, DEL, OF
C     2-4 NODE ISOPARAMETRIC 1D SHAPE FUNCTIONS AT LOCAL COORDINATE R
C
C     NODE NUMBERING  1---3---4---2
C
C---- D I C T I O N A R Y  O F  V A R I A B L E S ------------------
C
C     VARIABLE      DESCRIPTION
C     R             LOCAL COORDINATE -1 TO +1
C     DEL(2 TO 4)   VALUE OF LOCAL DERIVATIVE OF SHAPE FUNCTIONS
C     NELNOD        NUMBER OF ELEMENT NODES [2 TO 4]
C
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION DEL(NELNOD)

        DEL(1) = -0.5
        DEL(2) = 0.5
      IF(NELNOD.GE.3) THEN
        DEL(1) = DEL(1) + R

DEL(2) = DEL(2) + R

DEL(3) = -2.0*R
      ENDIF
      IF(NELNOD.EQ.4) THEN
        DEL(1) = DEL(1) + (-27.0*R*R + 2.0*R + 9.0)/16.0

DEL(2) = DEL(2) + (27.0*R*R + 2.0*R - 9.0)/16.0

DEL(3) = DEL(3) + (81.0*R*R + 14.0*R - 27.0)/16.0

DEL(4) = (-81*R*R - 18*R + 27.0)/16.0
      ENDIF
      RETURN
      END

C-----------------------------------------------------------------------
JACOB1
      SUBROUTINE JACOB1(NELNOD,DEL,XCOORD,JACOB)
C-SUB:JACOB1 - COMPUTES THE JACOBIAN, JACOB, FOR 2-4 NODE ISOPARA.
C     ELEM. USING VALUES OF LOCAL DERIVATIVES OF SHAPE FUNCTION IN DEL
C
C      J = dx/dr = d/dr{H1 H2 H3 H4}{X1 X2 X3 X4}T  ; dHi/dr = DELi
C
C     NODE NUMBERING  1---3---4---2
C
C---- D I C T I O N A R Y  O F  V A R I A B L E S ------------------
C
C     VARIABLE       DESCRIPTION
C     NELNOD         NUMBER OF ELEMENT NODES [2 TO 4]
C     DEL(2 TO 4)    VALUE OF LOCAL DERIVATIVE OF SHAPE FUNCTIONS
C     XCOORD(2 TO 4) (GLOBAL) COORDINATES OF NODES
C     JAC
C
      IMPLICIT REAL*8 (A-H,O-Z)
      REAL*8 JACOB
      DIMENSION DEL(NELNDD), XCOORD(NELNOD)
      JACOB = 0.0
      DO 10 N=1,NELNOD
   10 JACOB = JACOB + DEL(N)*XCOORD(N)
      RETURN
      END

C-----------------------------------------------------------------------
GAUSCO
      SUBROUTINE GAUSCO(N,A,W)
C--SUB:GAUSCO - GAUSS QUADRATURE ABSCISSAE AND WEIGHT COEFFICIENTS
C
C---- D I C T I O N A R Y  O F  V A R I A B L E S ------------------
C
C     VARIABLE             DESCRIPTION
C     N                    NUMBER OF QUADRATURE POINTS
C     A(N)                 ABSCISSAE
C     W(N)                 WEIGHTS
C
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION A(N), W(N)
      IF(N.EQ.1) THEN
        A(1) = 0.0D0

W(1) = 2.0D0
      ELSEIF(N.EQ.2) THEN
        A(1) = -1.0D0/SQRT(3.0D0)

A(2) = -A(1)

W(1) = 1.0D0

W(2) = 1.0D0
      ELSEIF(N.EQ.3) THEN
        A(1) = -SQRT(3.0/5.0)

A(2) = 0.0D0

A(3) = -A(1)

W(1) = 5.0D0/9.0D0

W(2) = 8.0D0/9.0D0

W(3) = W(1)
      ELSEIF(N.EQ.4) THEN
        A(1) = -0.861136311594053D0

A(2) = -0.339981043584856D0

A(3) = -A(2)

A(4) = -A(1)

W(1) = 0.347854845137454D0

W(2) = 0.652145154862546D0

W(3) = W(2)

W(4) = W(1)
      ELSE
        PAUSE ' **** ERROR:SUB:GAUSCO: Gauss coefs. not available.'
      ENDIF
      RETURN
      END

C-----------------------------------------------------------------------
QUADCO
      SUBROUTINE QUADCO(N,W,R)
C--SUB:QUADCO - EVALUATES WEIGHT COEFS. GIVEN QUAD POINTS
C
C---- D I C T I O N A R Y  O F  V A R I A B L E S ------------------
C
C     VARIABLE             DESCRIPTION
C INPUT
C     N                    NUMBER OF QUADRATURE POINTS
C     R(N)                 SPECIFIED QUAD POINTS
C OUTPUT
C     W(N)                 WEIGHT COEFFICIENTS
C LOCAL
C     C(4,4)

 QUADRATURE RULE EQUATION COEFFICIENTS
C
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION W(4), R(4), C(4,4)

      DO 10 I=1,N
      W(I) = (1.0D0 -(-1.0D0)**I)/DBLE(I)
      DO 10 J=1,N
   10 C(I,J) = R(J)**(I-1)

      CALL CROUT(C,W,N,4,1,0)

      RETURN
      END

C------------------------------------------------------------------ISD2D4
      SUBROUTINE ISO2D4(ID,XYZ,C,K,EO,NNOD,NEQN,MBAN,MAXBAN,ERR)
C-SUB:ISO2D4 - FORMS AND ASSEMBLES ISOPARAMETRIC 2D 4-NODE ELEMENTS
C
      IMPLICIT REAL*8 (A-H,O-Z)
```

```
C
C---- CAL-SAP: DATA & COMMON STORAGE
C
      COMMON /IOLIST/ NTM,NTR,NIN,NOT,NDI,ND2,ND3,ND4
C
C---- RESIST: DATA & COMMON STORAGE
C
      REAL*8 K(NEQN,MBAN),C(NEQN,MBAN),XYZ(NNOD,3),EO(NNOD),PK(3)
      INTEGER ID(NNOD), LMNEW(4), LMOLD(4)
      LOGICAL*1 ERR
      CHARACTER ENDFLAG*3, REG*3
      SAVE LMNEW,LMOLD,NNEW,NOLD,PK,PC,PP,PQ,PD,REG,/IOLIST/

      NDOF = 4

C
C--1.0 WRITE ELEMENT HEADER
C
      WRITE(NOT,2100)
 2100 FORMAT(
     ./'    == ISOPARAMETRIC 2D 4-NODE CONDUCTION ELEMENTS'//
     .'       C = Capacity    P = Density   Q = Internal Heat Rate'/
     .'       D = Thickness for REG=PLN; Subtended angle for REG=AXI'//
     .'       Elem   I  J  K  L  Kxx      Kyy      Kxy        C
     .        P        Q        D      REG')
C
C--2.0 GET FIRST ELEMENT, FORM & ADD ELEMENT TO SYS
C
      PQ = 0.0
      PD = 1.0
      REG = 'PLN'
      INCR = 1
      CALL FREE
      CALL FREETY
      CALL FREEI(' ',NOLD,1)
      CALL FREEI('I',LMOLD(1),4)
      CALL FREER('K',PK(1),3)
      CALL FREER('C',PC,1)
      CALL FREER('P',PP,1)
      CALL FREER('Q',PQ,I)
      CALL FREER('D',PD,1)
      CALL FREEH('G',REG,3,I)

      CALL ISO2D0(ID,XYZ,NOLD,LMOLD,K,C,EO,PK,PC,PP,PQ,PD,REG,
     .NNOD,NEQN,MBAN,MAXBAN,ERR)
C
C--3.0 GET NEXT LINE OF DATA
C
   30 CALL FREE
      CALL FREETY
C---- CHECK FOR "END"
      CALL FREEH(' ',ENDFLAG,3,I)
      IF(ENDFLAG.EQ.'END') THEN
        RETURN
      ENDIF
C---- GET NEW ELEMENT INFORMATION
      CALL FREEI(' ',NNEW,1)
      CALL FREEI('I',LMNEW(1),4)
      CALL FREEI('N',INCR,I)
      IF(INCR.EQ.0) INCR=I
      CALL FREER('K',PK(1),3)
      CALL FREER('C',PC,1)
      CALL FREER('P',PP,I)
      CALL FREER('Q',PQ,1)
      CALL FREER('D',PD,1)
      CALL FREEH('G',REG,3,I)
C---- CHECK NUMERICAL ORDER
      IF(NNEW.LE.NOLD) THEN
        WRITE(NTM,2300) NNEW

WRITE(NOT,2300) NNEW

ERR=.TRUE.

RETURN
      ENDIF
C---- GENERATE MISSING ELEMENTS
      IF(NNEW.GT.NOLD+I) THEN
        DO 34 N=NOLD+1,NNEW-1,1
          DO 32 I=1,NDOF
   32     LMOLD(I) = LMOLD(I) + INCR
   34   CALL ISO2D0(ID,XYZ,N,LMOLD,K,C,EO,PK,PC,PP,PQ,PD,REG,
     .NNOD,NEQN,MBAN,MAXBAN,ERR)
      ENDIF
C---- DO NEW ELEMENT
      NOLD = NNEW
      DO 36 I=1,NDOF
   36 LMOLD(I) = LMNEW(I)
      CALL ISO2D0(ID,XYZ,NOLD,LMOLD,K,C,EO,PK,PC,PP,PQ,PD,REG,
     .NNOD,NEQN,MBAN,MAXBAN,ERR)

      GO TO 30

 2300 FORMAT(' **** ERROR: Element number ',I5,' is out of order.')

      END

C-----------------------------------------------------------------ISO2D0
      SUBROUTINE ISO2D0(ID,XYZ,NELM,LM,K,C,EO,PK,PC,PP,PQ,PD,REG,
     .NNOD,NEQN,MBAN,MAXBAN,ERR)
C-SUB:ISO2D0 - REPORTS ELEM INFORMATION, CHECKS BANDWIDTH,
C              FORMS ELEM ARRAYS & ADDS THEM TO SYS ARRAYS
C
      IMPLICIT REAL*8 (A-H,O-Z)
C
C
C---- CAL-SAP: DATA & COMMON STORAGE
C
      COMMON /IOLIST/ NTM,NTR,NIN,NOT,ND1,ND2,ND3,ND4
C
C---- FORM: DATA & COMMON STORAGE
C
      REAL*8 XYZ(NNOD,3),K(NEQN,MBAN),C(NEQN,MBAN),EO(NNOD),
     .PK(3),KE(4,4),CE(4,4),EE(4),XY(2,4)
      INTEGER ID(NNOD), LM(4)
      LOGICAL*1 ERR
      CHARACTER REG*3
      SAVE/IOLIST/
C
C---- REPORT ELEMENT INFORMATION TO OUTPUT DATA FILE
C
      WRITE(NOT,2000) NELM,LM,PK,PC,PP,PQ,PD,REG
 2000 FORMAT(6X,I4,1X,4I3,1X,7G8.2,A3)
C
C---- NODE ERROR TRAP
C
      DO 200 N=1,4
      NN=LM(N)
      IF(NN.LE.0.OR.NN.GT.NNOD) THEN
        WRITE(NTM,2010) NN

WRITE(NOT,2010) NN
        ERR=.TRUE.

RETURN
  200 ENDIF
C
C---- DETERMINE ELEMENT BANDWIDTH
C
      CALL MBAND(ID,LM,4,NNOD,MBAN,MAXBAN,ERR)
C
C---- FORM ELEMENT COORD SUB-ARRAY XY(2,4)
C
      DO 300 I=1,2
      DO 300 J=1,4
  300 XY(I,J) = XYZ(LM(J),I)
C
C---- FORM ELEMENT ARRAYS
C
      CALL ISO2D(KE,CE,EE,PK,PC,PP,PQ,PD,REG,XY)
C
C---- ADD ELEMENT CONTRIBUTION TO SYSTEM ARRAYS
C
      CALL ADDCM(ID,LM,KE,K,4,4,NNOD,NEQN,MBAN)
      CALL ADDCM(ID,LM,CE,C,4,4,NNOD,NEQN,MBAN)

      DO 400 N=1,4
  400 EO(LM(N)) = EE(N)


      RETURN

 2010 FORMAT(' **** ERROR: (Generated) Node ',I5,' is out of range.')

      END
C------------------------------------------------------------------ISO2D
      SUBROUTINE ISO2D(KE,CE,EE,PK,PC,PP,PQ,PD,REG,XY)
C-SUB:ISO2D - 2D ISOPARMETRIC 4-NODE CONDUCTION FINITE ELEMENT
C            MODIFICATION OF SUBROUTINE DEVELOPED BY R. TAYLOR
C            U.C. BERKELEY FOR PROGRAM *HEAT*
C
C---- D I C T I O N A R Y   O F   V A R I A B L E S --------------------
C
C     VARIABLE     DESCRIPTION---------------------------------------
C     KE(4,4)      ELEMENT CONDUCTANCE MATRIX
C     CE(4,4)      ELEMENT CAPACITANCE MATRIX
C     EE(4)        ELEMENT INTERNAL GENERATED FLUX VECTOR
C     PK(3)        ELEMENT CONDUCTIVITY TENSOR: Kxx, Kyy, Kxy
C     PC           ELEMENT SPECIFIC HEAT CAPACITY
C     PP           ELEMENT DENSITY
C     PQ           VOLUMETRIC INTERNAL HEAT GENERATION RATE
C     PD           ELEMENT THICKNESS/SUBTENDED ANGLE
C     REG          ELEMENT TYPE: 'PLN' OR 'AXI'
C     XY(2,4)      ELEMENT NODAL COORDINATES
C-----------------------------------------------------------------------
      IMPLICIT REAL*8 (A-H,O-Z)
      CHARACTER REG*3
      REAL*8 SG(4),TG(4),KE(4,4),CE(4,4),EE(4),XY(2,4),PK(3)
      COMMON /ISODAT/SH(3,4),DV
      SAVE /ISODAT/
      DATA SG/1.,1.,-1.,-1./,TG/-1.,I.,1.,-1./
      G = 1.0/SQRT(3.0)
      DO 100 I=1,4
      EE(I) = 0.0
      DO 100 J=1,4
      CE(I,J) = 0.0
  100 KE(I,J) = 0.0
      PCPP=PC*PP
      DO 103 L=1,4
      CALL ISOSHP(SG(L)*G,TG(L)*G,XY)
      RR = 0.0
      DV = DV*PD
      DO I01 I=1,4
  101 RR = RR + XY(1,I)*SH(3,I)
      IF(REG.EQ.'AXI') DV = DV*RR
      DO 102 J=1,4
      SHJ = SH(3,J)*DV
```

```
      EE(J) = EE(J) + SHJ*PQ
      SBJ = SHJ*PCPP
      A1 = (PK(1)*SB(1,J) + PK(3)*SB(2,J))*DV
      A2 = (PK(3)*SH(1,J) + PK(2)*SB(2,J))*DV
      DO 102 I=1,4
      KE(I,J) = KE(I,J) + A1*SB(1,I) + A2*SB(2,I)
  102 CE(I,J) = CE(I,J) + SHJ*SB(3,I)
  103 CONTINUE
      RETURN
      END
C--------------------------------------------------------------ISOSHP
      SUBROUTINE ISOSHP(SS,TT,XY)
C-SUB:ISOSBP - SHAPE FUNCTION SUBROUTINE FOR ISO2D
      IMPLICIT REAL*8 (A-B,O-Z)
      REAL*8 SX(2,2),XS(2,2),R(4),T(4),XY(2,4)
      COMMON /ISODAT/SB(3,4),DV
      SAVE /ISODAT/
      DATA R/-0.5,0.5,0.5,-0.5/,T/-0.5,-0.5,0.5,0.5/
      DO 100 I=1,4
      SB(3,I) = (0.5+R(I)*SS)*(0.5+T(I)*TT)
      SB(1,I) = R(I)*(0.5+T(I)*TT)
  100 SB(2,I) = T(I)*(0.5+R(I)*SS)
      DO 101 I=1,2
      DO I0I J=1,2
      XS(I,J)=0.0
      DO 101 K=1,4
  101 XS(I,J) = XS(I,J) + XY(I,K)*SB(J,K)
      DV = XS(1,1)*XS(2,2) - XS(1,2)*XS(2,1)
      SX(1,1) = XS(2,2)/DV
      SX(2,2) = XS(1,1)/DV
      SX(1,2) = -XS(1,2)/DV
      SX(2,1) = -XS(2,1)/DV
      DO 102 K=1,4
      CC = SB(1,K)*SX(1,1) + SB(2,K)*SX(2,1)
      SH(2,K) = SB(1,K)*SX(1,2) + SB(2,K)*SX(2,2)
  102 SB(1,K) = CC
      RETURN
      END

C----------------------------------------------------------------VNMRT
      SUBROUTINE VNMRT(ID,K,NNOD,NEQN,MBAN,MAXBAN,ERR)
C-SUB:VNMRT - FORMS AND ASSEMBLES VARIABLE NODE MRT ELEMENTS
C *** PRESENTLY A STUB
      RETURN
      END

C----------------------------------------------------------------MBAND
      SUBROUTINE MBAND(ID,LM,NDOF,NNOD,MBAN,MAXBAN,ERR)
C-SUB: MBAND - DETERMINES ELEMENT BAND NIDTH
C
C---- CAL-SAP: DATA & COMMON STORAGE
C
      COMMON /IOLIST/ NTM,NTR,NIN,NOT,ND1,ND2,ND3,ND4
C
C---- MBAND: DATA
C
      INTEGER ID(NNOD),LM(NDOF)

      MBANEL = 0
      DO I0 I=1,NDOF
      II = ABS(ID(LM(I)))
      DO I0 J=1,NDOF
      JJ = ABS(ID(LM(J)))
   10 MBANEL = MAX(MBANEL, ABS(II-JJ+1))

      IF(MBANEL.GT.MBAN) THEN
        WRITE(NTM,2000) MBANEL,MBAN

WRITE(NOT,2000) MBANEL,MBAN

ERR=.TRUE.
      ENDIF

      MAXBAN = MAX(MAXBAN,MBANEL)

      RETURN

 2000 FORMAT(' **** ERROR: Bandwidth exceeded for current element.'/
     .'             Bandwidth required :',I5/
     .'             Bandwidth available:',I5)

      END

C---------------------------------------------------------------ADDCM
      SUBROUTINE ADDCM(ID,LM,ELM,SYS,NDOF,NSIZE,NNOD,NEQN,MBAN)
C-SUB: ADDCM - ADDS ELEMENT MATRIX TO
C             COMPACT FORM OF SYSTEM MATRIX

      REAL*8 ELM(NSIZE,NSIZE), SYS(NEQN,MBAN)
      INTEGER ID(NNOD),LM(NSIZE)

      DO 20 I=1,NDOF
      II = ABS(ID(LM(I)))
      DO I0 J=1,NDOF
      JJ = ABS(ID(LM(J))) - II +1
      IF(JJ.LE.0) GO TO 10
      SYS(II,JJ) = SYS(II,JJ) + ELM(I,J)
   I0 CONTINUE
   20 CONTINUE

      RETURN
      END

C*******************************************************************
C TREE: BCOND - TREE OF SUBROUTINES TO ESTABLISH BOUNDARY CONDITIONS
C*******************************************************************
C---------------------------------------------------------------RBCOND
      SUBROUTINE RBCOND(MPID,NNOD,ERR)
C-SUB:RBCOND - ROOT SUBROUTINE TO BCOND
C     BCOND  - ESTABLISHES BOUNDARY CONDITION FLAGS BY MODIFYING
ID(NNOD)
C
C---- D I C T I O N A R Y   O F   V A R I A B L E S ------------------
C
C     VARIABLE      DESCRIPTION--------------------------------------
C VARIABLES PASSED TO SUBROUTINE
C     ERR           DO-WBILE TERMINATOR FLAG
C     NNOD          NUMBER OF NODES IN SYSTEM
C     MBAN          (HALF) BANDWIDTB OF SYSTEM EQUATIONS
C     MPID          POINTER TO ID(NNOD) IN BLANK COMMON
C     ID(NNOD)      EQUATION NUMBER/B.C. FLAG ARRAY;
C                   ABS(ID(N)) = EQUATION NUMBER OF NODE N
C                   SIGN(ID(N)) = -1 = TEMPERATURE PRESCRIBED NODE
C                   SIGN(ID(N)) = +1 = FLUX PRESCRIBED NODE
C--------------------------------------------------------------------
-
C
C---- CAL-SAP: DATA & COMMON STORAGE
C
      COMMON MTOT,NP,IA(1)
      COMMON /DBSYS/ NUMA,NEXT,IDIR,IP(3)
      COMMON /IOLIST/ NTM,NTR,NIN,NOT,ND1,ND2,ND3,ND4
C
C---- RBCOND: DATA & COMMON STORAGE
C
      LOGICAL*1 ERR

      SAVE /DBSYS/,/IOLIST/
C
C--0.0 WRITE HEADER
C
      WRITE(NTM,2000)
      WRITE(NOT,2000)
 2000 FORMAT(/' ==== BOUNDARY CONDITIONS AND EQUATION NUMBERS')

C
C--1.0 FIND SEPARATOR "BOUNDARY"
C
      CALL FINDN('BOUNDARY',8,KEY)
      IF(KEY.EQ.1) THEN
        WRITE(NTM,2100)
        WRITE(NOT,2100)
        RETURN
      ENDIF
      CALL FREETY
C
C--2.0 CALL BCOND TO DO THE WORK
C
      CALL BCOND(IA(MPID),NNOD,ERR)

      RETURN

 2100 FORMAT(/'   -- NOTE: Boundary condition data not found.'/
     .'              All nodes assumed to be flux-prescribed nodes.'/
     .'              Equation numbers = node numbers.')

      END
C------------------------------------------------------------------
BCOND
      SUBROUTINE BCOND(ID,NNOD,ERR)
C-SUB:BCOND - READS AND GENERATES BOUNDARY CONDITION FLAGS
C
      IMPLICIT REAL*8 (A-B,O-Z)
C
C---- CAL-SAP: DATA & COMMON STORAGE
C
      COMMON /IOLIST/ NTM,NTR,NIN,NOT,ND1,ND2,ND3,ND4
C
C---- BCOND: DATA & COMMON STORAGE
C
      CHARACTER ENDFLAG*3, BC*1
      DIMENSION ID(NNOD), IJK(3)
      LOGICAL*1 ERR

      SAVE /IOLIST/

C
C--1.0 WRITE BOUNDARY CONDITIONS HEADER
C
      WRITE(NOT,2100)
 2100 FORMAT(/
     .6X'Negative Eqtn-# = part of temperature-prescribed boundary.'/
     .6x'Positive Eqtn-# = part of flux-prescribed boundary.'//
     .6X,'Node   Eqtn-#  Node   Eqtn-#  Node   Eqtn-#  Node   Eqtn-#
     . Node   Eqtn-# ')
C
C--2.0 INTERPRET LINE OF DATA
C
   20 CALL FREE
      CALL FREETY
C---- CHECK FOR "END"
      CALL FREEH(' ',ENDFLAG,3,1)
      IF(ENDFLAG.EQ.'END') GO TO 30
C---- DO IT
      CALL FREEH('C',BC,1,1)
      IF(BC.EQ.'Q'.OR.BC.EQ.'T') THEN
        ISIGN = 1
```

```
        IF(HC.EQ.'T') ISIGN = -I
        CALL FREEI(' ',IJK(I),3)
        IF(IJK(2).EQ.0) IJK(2)=IJK(I)
        IF(IJK(3).EQ.0) IJK(3)=I
        DO 24 N=IJK(1),IJK(2),IJK(3)
        IF(N.LE.0.OR.N.GT.NNOD) THEN
          WRITE(NTM,2200) N
          WRITE(NOT,2200) N
          ERR=.TRUE.
          GO TO 20
        ENDIF

NTDOF = ABS(ID(N))
C---- SET TDOF TO NEG. VALUE FOR ALL NODES CONSTRAINED EQUAL
C---- TO AVOID PROBLEMS OF MULTIPLE BC SPECIFICATION USE ABS()
C---- SO LAST REDUNDANT BC SPECIFICATION GOVERNS

DO 24 NC=1,NNOD

  NCDOF=ABS(ID(NC))

  IF(NCDOF.EQ.NTDOF) ID(NC)=ISIGN*NCDOF
   24   CONTINUE
      ELSE
        WRITE(NTM,2210) BC
        WRITE(NOT,2210) BC
        ERR=.TRUE.
      ENDIF
      GO TO 20
C
C--3.0 REPORT BOUNDARY CONDITIONS IF NO ERROR ENCOUNTERED
C
   30 IF(ERR) RETURN
      WRITE(NOT,2300) (N,ID(N),N=I,NNOD)

      RETURN

 2200 FORMAT(' **** ERROR: (Generated) Node',I5,' is out of range.')
 2210 FORMAT(' **** ERROR: Houndary condition ',AI,' not available.')
 2300 FORMAT((4X,5(I6,IX,I6,2X)))

      END

C*****************************************************************
C TREE: CONV - TREE OF SUBROUTINES TO ESTABLISH CONVECTIVE B.C.s
C*****************************************************************
C--------------------------------------------------------------RCONV
      SUHROUTINE RCONV(MPID,MPXYZ,MPK,MPCH,NNOD,NEQN,MBAN,ERR)
C-SUH:RCONV - ROOT SUBROUTINE TO CONV
C     CONV  - ESTABLISHES CONVECTIVE HOUNDARY CONDITIONS
C
C
C---- D I C T I O N A R Y   O F   V A R I A H L E S -------------------
C
C     VARIAHLE       DESCRIPTION--------------------------------------
C VARIABLES PASSED TO SUBROUTINE
C     ERR            DO-WHILE TERMINATOR FLAG
C     NNOD           NUMBER OF NODES IN SYSTEM
C     NEQN           NUMBER OF (SYSTEM) EQUATIONS
C     MBAN           (HALF) BANDWIDTH OF SYSTEM EQUATIONS
C     MPCH           POINTER TO CH(NNOD) IN BLANK COMMON
C     MPK            POINTER TO K(NEQN) IN HLANK COMMON
C     MPXYZ          POINTER TO XYZ(NHOD,3) IN HLANK COMMON
C     MPID           POINTER TO ID(NNOD) IN BLANK COMMON
C     CH(NNOD)       COEFS. FOR COMPUTING EFFECTIVE CONVECTIVE
EXCITATION
C     K(NEQN,MBAN)   SYSTEM CONDUCTANCE TRANSFER MATRIX (COMPACT FORM)
C     ID(NNOD)       EQUATION NUMBER/B.C. FLAG ARRAY;
C     XYZ(NNOD,3)      COORDINATE ARRAY (ORDERED HY NODE NUMBER)
C                    ABS(ID(N)) = EQUATION NUMBER OF NODE N
C                    SIGN(ID(N)) = -1 = TEMPERATURE PRESCRIHED NODE
C                    SIGN(ID(N)) = +I = FLUX PRESCRIBED NODE
C-------------------------------------------------------------------
C---- CAL-SAP: DATA & COMMON STORAGE
C
      COMMON MTOT,NP,IA(1)
      COMMON /DHSYS/ NUMA,NEXT,IDIR,IP(3)
      COMMON /IOLIST/ NTM,NTR,NIN,NOT,ND1,ND2,ND3,ND4
C
C---- ICOND: DATA & COMMON STORAGE
C
      LOGICAL*1 ERR

      SAVE /DBSYS/,/IOLIST/
C
C--0.0 WRITE HEADER
C
      WRITE(NTM,2000)
      WRITE(NOT,2000)
 2000 FORMAT(/' ==== CONVECTIVE BOUNDARY CONDITIONS')

C
C--I.0 DEFINE ARRAY
C
      CALL DELETE('CH  ')
      CALL DEFINE('CH  ',MPCH,NNOD,1)
C
C--2.0 INITIALIZE ARRAYS
C
      CALL ZEROR(IA(MPCH),NNOD,1)
C
C--3.0 FIND SEPARATOR "CONVECT"
C
      CALL FINDN('CONVECT',7,KEY)
      IF(KEY.EQ.I) THEN
```

```
        WRITE(NTM,2300)
        WRITE(NOT,2300)
        RETURN
      ENDIF
      CALL FREETY
C
C--4.0 CALL CONV TO DO THE WORK
C
      CALL
CONV(IA(MPID),IA(MPXYZ),IA(MPK),IA(MPCH),NNOD,NEQN,MBAN,ERR)

      RETURN

 2300 FORMAT(/'   -- NOTE: Convection boundary data not found.')

      END

C-----------------------------------------------------------------CONV
      SUBROUTINE CONV(ID,XYZ,K,CH,NNOD,NEQN,MBAN,ERR)
C-SUH:CONV  - READS AND GENERATES CONVECTION BOUNDARY DATA
C
      IMPLICIT REAL*8 (A-H,O-Z)
C
C
C---- CAL-SAP: DATA & COMMON STORAGE
C
      COMMON /IOLIST/ NTM,NTR,NIN,NOT,NDI,ND2,ND3,ND4
C
C---- CONV: DATA & COMMON STORAGE
C
      REAL*8 K(NEQN,MBAN), CH(NNOD), XYZ(NNOD,3)
      INTEGER ID(NNOD), LMNEW(2), LMOLD(2)
      LOGICAL*1 ERR
      CHARACTER ENDFLAG*3, REG*3

      SAVE

      NDOF = 2

C
C--I.0 WRITE CONVECTION B.C. HEADER
C
      WRITE(NOT,2100)
 2100 FORMAT(/
     .6X'D = Thickness for REG=PLN; Subtended angle for REG=AXI.'//
     .'      Surf    I     J      H-Coef        D        REG')
C
C--2.0 GET FIRST SURFACE SEGMENT, FORM & MODIFY SYSTEM K
C
      CALL FREE
      CALL FREETY
      CALL FREEI(' ',NOLD,1)
      CALL FREEI('I',LMOLD(I),2)
      CALL FREER('H',COEF,I)
      D=1.0
      CALL FREER('D',D,1)
      REG = 'PLN'
      CALL FREEH('G',REG,3,I)

      CALL CONVO(ID,XYZ,NOLD,LMOLD,K,CH,COEF,REG,D,NNOD,NEQN,MHAN,ERR)

C
C--3.0 GET NEXT LINE OF DATA
C
   30 CALL FREE
      CALL FREETY
C---- CHECK FOR "END"
      CALL FREEH(' ',ENDFLAG,3,1)
      IF(ENDFLAG.EQ.'END') THEN
        RETURN
      ENDIF
C---- GET NEW ELEMENT INFORMATION
      CALL FREEI(' ',NNEW,I)
      CALL FREEI('I',LMNEW(1),2)
      INCR = 1
      CALL FREEI('N',INCR,1)
      CALL FREER('H',COEF,1)
      D=1.0
      CALL FREER('D',D,1)
      REG = 'PLN'
      CALL FREEH('G',REG,3,1)
C---- CHECK NUMERICAL ORDER
      IF(NNEW.LE.NOLD) THEN
        WRITE(NTM,2300) NNEW
        WRITE(NOT,2300) NNEW
        ERR=.TRUE.
        RETURN
      ENDIF
C---- GENERATE MISSING SURFACE SEGMENTS
      DO 34 N=NOLD+1,NNEW-1,I
        DO 32 I=1,NDOF
   32   LMOLD(I) = LMOLD(I) + INCR
   34 CALL CONVO(ID,XYZ,N,LMOLD,K,CH,COEF,REG,D,NNOD,NEQN,MBAN,ERR)
C---- DO NEW SURFACE SEGMENT
      NOLD = NNEN
      DO 36 I=1,NDOF
   36 LMOLD(I) = LMNEW(I)
      CALL CONVO(ID,XYZ,NOLD,LMOLD,K.CH,COEF,REG,D,NNOD,NEQN,MBAN,ERR)

      GO TO 30

 2300 FORMAT(' **** ERROR: Surface segment number',I5,' out of
order.')
```

```
      END

C-----------------------------------------------------------------CONV0
      SUBROUTINE CONV0(ID,XYZ,NSEG,LM,K,CH,COEF,REG,D,NNOD,NEQN,
     +MBAN,ERR)
C-SUB:CONV0 - REPORTS CONVECTIVE SURFACE SEGMENT INFORMATION,
C            FORMS COEFS. CH(NNOD), FOR COMPUTING EFFECT.CONV.EXCIT.
C            MODIFIES K MATRIX FOR CONVECTION BOUNDARY CONDITIONS
C
      IMPLICIT REAL*8 (A-H,O-Z)
C
C---- CAL-SAP: DATA & COMMON STORAGE
C
      COMMON /IOLIST/ NTM,NTR,NIN,NOT,ND1,ND2,ND3,ND4
C
C---- FORM: DATA & COMMON STORAGE
C
      REAL*8 K(NEQN,MBAN), CH(NNOD), XYZ(NNOD,3), S(2,2)
      INTEGER ID(NNOD), LM(2)
      LOGICAL*1 ERR
      CHARACTER REG*3

      SAVE
C
C--1.0 REPORT ELEMENT INFORMATION TO OUTPUT DATA FILE
C
      WRITE(NOT,2000) NSEG,LM(1),LM(2),COEF,D,REG
C
C--2.0 COMPUTE COEFFICIENT,CH, FOR COMPUTING EFFECTIVE CONVECTIVE FLUX
C
C---- ERROR TRAP FOR REGION TYPE
      IF(REG.EQ.'PLN'.OR.REG.EQ.'AXI') THEN
        CONTINUE
      ELSE
        WRITE(NTM,2200) REG
        WRITE(NOT,2200) REG
        ERR=.TRUE.
        RETURN
      ENDIF
C---- ERROR TRAP FOR OUT-OF-RANGE NODES
      DO 200 N=1,2
      NN = LM(N)
      IF(NN.LE.0.OR.NN.GT.NNOD) THEN
        WRITE(NTM,2210) NN
        WRITE(NOT,2210) NN
        ERR=.TRUE.
        RETURN
C---- ERROR TRAP FOR TEMP-PRESCRIBED NODES
      ELSEIF(ID(NN).LE.0) THEN
        WRITE(NTM,2220) NN

WRITE(NOT,2220) NN

ERR=.TRUE.

RETURN
      ENDIF
  200 CONTINUE
      N1 = LM(1)
      N2 = LM(2)
C---- COMPUTE SURFACE SEGMENT LENGTH
      XL = SQRT((XYZ(N1,1)-XYZ(N2,1))**2 +
     .      (XYZ(N1,2)-XYZ(N2,2))**2)
C---- COMPUTE CH
      IF(REG.EQ.'PLN') THEN
        CH(N1) = CH(N1) + D*COEF*XL/2.0
        CH(N2) = CH(N2) + D*COEF*XL/2.0
      ELSEIF(REG.EQ.'AXI') THEN
        CH(N1) = CH(N1)+(D*COEF*XL/6.0)*(2.0*XYZ(N1,1) + XYZ(N2,1))
        CH(N2) = CH(N2)+(D*COEF*XL/6.0)*(XYZ(N1,1) + 2.0*XYZ(N2,1))
      ENDIF
C
C--3.0 MODIFY CONDUCTANCE TRANSFER MATRIX, K
C
C---- COMPUTE CONVECTIVE CONTRIBUTION TO K
      IF(REG.EQ.'PLN') THEN
        S(1,1) = D*COEF*XL/3.0
        S(1,2) = D*COEF*XL/6.0
        S(2,1) = S(1,2)
        S(2,2) = S(1,1)
      ELSEIF(REG.EQ.'AXI') THEN
        S(1,1) = (D*COEF*XL/4.0)*(XYZ(N1,1) + XYZ(N2,1)/3.0)
        S(1,2) = (D*COEF*XL/12.0)*(XYZ(N1,1) + XYZ(N2,1))
        S(2,1) = S(1,2)
        S(2,2) = (D*COEF*XL/4.0)*(XYZ(N1,1)/3 + XYZ(N2,1))
      ENDIF
C---- ADD CONVECTIVE CONTRIBUTION TO SYSTEM ARRAYS

      CALL ADDCM(ID,LM,S,K,2,2,NNOD,NEQN,MBAN)

      RETURN
 2000 FORMAT(6X,I4,2I5,5X,2G10.3,3X,A3)
 2200 FORMAT(' **** ERROR: Region ',A3,' is not defined.')
 2210 FORMAT(' **** ERROR: (Generated) Node',I5,' is out of range.')
 2220 FORMAT(' **** ERROR: (Generated) Node',I5,
     .' is not part of flux-prescribed boundary')

      END

C*********************************************************************
C TREE: LSOLV - TREE OF SUBROUTINES TO SOLVE LINEAR SYSTEM EQTNS
C*********************************************************************
C------------------------------------------------------------------
```

```
RLSOLV
      SUBROUTINE RLSOLV(MPID,MPK,MPC,MPEO,MPCH,NNOD,NEQN,MBAN,ERR)
C--SUB: RLSOLV - ROOT SUBROUTINE TO ALL LINEAR SOLVER SUBROUTINES
C
C---- D I C T I O N A R Y   O F   V A R I A B L E S ------------------
C
C     VARIABLE      DESCRIPTION---------------------------------------
C
C     ERR           DO-WHILE TERMINATOR FLAG
C     NNOD          NUMBER OF NODES IN SYSTEM
C     NEQN          NUMBER OF (SYSTEM) EQUATIONS
C     MBAN          (HALF) BANDWIDTH OF SYSTEM EQUATIONS
C     IWRT          WRITE-SOLUTION-TO-FILE FLAG (1=DO IT)
C     IPRT          OUTPUT PRINT INTERVAL
C     NPRT          NUMBER OF NODES SELECTED FOR SOLUTION OUTPUT PRINT
C     MPNPR          POINTER TO NPR(NPRT) IN BLANK COMMON
C     MPCH          POINTER TO CH(NNOD) IN BLANK COMMON
C     MPEO          POINTER TO EO(NNOD) IN BLANK COMMON
C     MPC           POINTER TO C(NEQN,MBAN) IN BLANK COMMON
C     MPK           POINTER TO K(NEQN,MBAN) IN BLANK COMMON
C     MPID          POINTER TO ID(NNOD) IN BLANK COMMON
C     NPR(NPRT)      LIST OF NODES SELECTED FOR SOLUTION OUTPUT PRINT
C     CH(NNOD)      COEFS. FOR COMPUTING EFFECTIVE CONVECTIVE
EXCITATION
C     EO(NNOD)      INITIAL EXCITATION VECTOR (STORED BY NODE)
C     C(NEQN,MBAN)  SYSTEM CAPACITY MATRIX (COMPACT FORM)
C     K(NEQN,MBAN)  SYSTEM CONDUCTANCE TRANSFER MATRIX (COMPACT FORM)
C     ID(NNOD)      EQUATION NUMBER/B.C. FLAG ARRAY;
C                   ABS(ID(N)) = EQUATION NUMBER OF NODE N
C                   SIGN(ID(N)) = -1 = TEMPERATURE PRESCRIBED NODE
C                   SIGN(ID(N)) = +1 = FLUX PRESCRIBED NODE
C------------------------------------------------------------------
--
C---- CAL-SAP: DATA & COMMON STORAGE
C
      COMMON MTOT,NP,IA(1)
      COMMON /DBSYS/ NUMA,NEXT,IDIR,IP(3)
      COMMON /IOLIST/ NTM,NTR,NIN,NOT,ND1,ND2,ND3,ND4
C
C---- RFORM: DATA & COMMON STORAGE
C
      LOGICAL*1 ERR, NOSOLV
      INTEGER ITIME1,ITIME2

      SAVE /DBSYS/,/IOLIST/
C
C--0.0 WRITE SOLUTION HEADER & WRITE SYSTEM MATRICES TO <filename>.SYS
C
      WRITE(NTM,2100)
      WRITE(NOT,2100)
 2100 FORMAT(/,' ==== SOLUTION')
      CALL SYSAVE(IA(MPK),IA(MPC),NEQN,MBAN)
      NOSOLV = .TRUE.
C
C--1.0 DEFINE E(NEQN)
C

CALL DELETE('E   ')

CALL DEFINE('E   ',MPE,NEQN,1)

C
C--2.0 GET REPORT CONTROL INFORMATION
C
      CALL REPORT(MPNPR,NPRT,IPRT,IWRT,NNOD,ERR)

      IF(ERR) RETURN
C
C--3.0 LOOK FOR SEPARATOR "STEADY"
C
      CALL FINDN('STEADY',6,KEY)
      IF(KEY.EQ.0) THEN
        NOSOLV=.FALSE.
        CALL ZEROR(IA(MPE),NEQN,1)

CALL TIME(ITIME1)

CALL STEADY(IA(MPID),IA(MPK),IA(MPE),IA(MPEO),IA(MPCH),
     +  IA(MPNPR),NPRT,IWRT,NNOD,NEQN,MBAN,ERR)
        CALL TIME(ITIME2)

WRITE(NTM,2300) (ITIME2-ITIME1)

WRITE(NOT,2300) (ITIME2-ITIME1)
      ENDIF

 2300 FORMAT(/'   -- NOTE: Solution time: ',I5,' seconds.')

C
C--4.0 LOOK FOR SEPARATOR "HARMON"
C
      CALL FINDN('HARMON',6,KEY)
      IF(KEY.EQ.0) THEN
        NOSOLV=.FALSE.
        CALL ZEROR(IA(MPE),NEQN,1)

CALL TIME(ITIME1)

CALL HARMON(IA(MPID),IA(MPK),IA(MPC),IA(MPE),IA(MPEO),IA(MPCH),
     .  IA(MPNPR),IA(MPK),IA(MPE),NPRT,IWRT,NNOD,NEQN,MBAN,ERR)
        CALL TIME(ITIME2)

WRITE(NTM,2300) (ITIME2-ITIME1)
```

```
WRITE(NOT,2300) (ITIME2-ITIME1)
      ENDIF
C
C--5.0 LOOK FOR SEPARATOR "PREDICT"
C
      CALL FINDN('PREDICT',7,KEY)
      IF(KEY.EQ.0) THEN
        NOSOLV=.FALSE.
        CALL ZEROR(IA(MPE),NEQN,1)

CALL TIME(ITIME1)

CALL PREDIC(IA(MPID),IA(MPK),IA(MPC),IA(MPE),IA(MPEO),IA(MPCH),
     .  IA(MPNPR),NPRT,IPRT,IWRT,NNOD,NEQN,MBAN,ERR)
        CALL TIME(ITIME2)

WRITE(NTM,2300) (ITIME2-ITIME1)

WRITE(NOT,2300) (ITIME2-ITIME1)
      ENDIF
C
C--6.0 REPORT NO SOLUTION REQUEST ...
C
      IF(NOSOLV) THEN
        WRITE(NTM,2600)
        WRITE(NOT,2600)
      ENDIF

      CALL FCLOSE(ND1)

      RETURN

 2600 FORMAT(/'   -- NOTE: No solution control data found.')

      END

C---------------------------------------------------------------- SYSAVE
      SUBROUTINE SYSAVE(K,C,NEQN,MBAN)
C-SUB:SYSAVE - WRITES [K] & [C] TO <filename>.SYS
C
C---- CAL-SAP: DATA & COMMON STORAGE
C
      COMMON /IOLIST/NTM,NTR,NIN,NOT,ND1,ND2,ND3,ND4
C
C---- SYSAVE: DATA
C
      REAL*8 K(NEQN,MBAN), C(NEQN,MBAN)
C
      CALL FOPEN(ND1,'SYS','NEW')
      WRITE(ND1) ((K(N,M),N=1,NEQN),M=1,MBAN),
     +((C(N,M),N=1,NEQN),M=1,MBAN)

      RETURN
      END

C----------------------------------------------------------------STEADY
      SUBROUTINE STEADY(ID,K,E,EO,CH,NPR,NPRT,IWRT,NNOD,NEQN,MBAN,ERR)
C-SUB: STEADY - FORMS {E} OF [K]{T} = {E}
C                CALL SOLVER AND REPORTS RESULTS
C                SOLUTION {T} WRITTEN OVER {E}
      IMPLICIT REAL*8 (A-H,O-Z)
C
C---- CAL-SAP: DATA & COMMON STORAGE
C
      COMMON /IOLIST/NTM,NTR,NIN,NOT,ND1,ND2,ND3,ND4
C
C---- STEADY: DATA & COMMON STORAGE
C
      REAL*8 K(NEQN,MBAN),E(NEQN),EO(NNOD),CH(NNOD)
      INTEGER ID(NNOD), NPR(NPRT), IJK(3)
      LOGICAL*1 ERR, TDOF
      CHARACTER ENDFLAG*3

      SAVE /IOLIST/
C
C--1.0 WRITE HEADER
C
      WRITE(NTM,2100)
      WRITE(NOT,2100)
 2100 FORMAT(/'   == STEADY STATE SOLUTION')
      CALL FREETY
C
C--2.0 SCALE [K] FOR TEMP-PRESCRIBED NODES
C
      DO 20 I=1,NEQN
   20 IF(TDOF(I,ID,NNOD)) K(I,1) = K(I,1)*1.0E+15
C
C--3.0 GET T, Q, AND/OR TEX & FORM {E}
C
      WRITE(NTM,2300)
      WRITE(NOT,2300)
      WRITE(NOT,2302)
   30 CALL FREE
      CALL FREETY
C---- CHECK FOR "END"
      CALL FREEH(' ',ENDFLAG,3,1)
      IF(ENDFLAG.EQ.'END') GO TO 40
C---- INTERPRET LINE AND GENERATE DATA
      Q = 0.0
      CALL FREER('Q',Q,1)
      T = 0.0
      CALL FREER('T',T,1)
      TEX = 0.0
      CALL FREER('X',TEX,1)
      CALL FREEI(' ',IJK(1),3)
      IF(IJK(2).EQ.0) IJK(2)=IJK(1)
      IF(IJK(3).EQ.0) IJK(3)=1
      DO 32 N=IJK(1),IJK(2),IJK(3)
      IF(N.LE.0.OR.N.GT.NNOD) THEN
        WRITE(NTM,2310) N
        WRITE(NOT,2310) N
        ERR=.TRUE.
        GO TO 30
      ENDIF
      NN = ID(N)
      NNN = ABS(NN)
C---- TEMPERATURE-PRESCRIBED NODES
      IF(NN.LT.0) THEN
        WRITE(NOT,2320) N,T
        E(NNN) = T*K(NNN,1)
C---- FLUX-PRESCRIBED NODES
      ELSEIF(NN.GT.0) THEN
        WRITE(NOT,2330) N,EO(N),Q,TEX
        E(NNN) = E(NNN) + EO(N) + Q + TEX*CH(N)
      ENDIF
   32 CONTINUE
      GO TO 30
C
C--4.0 SOLVE
C
   40 IF(ERR) RETURN
      CALL SYMBC(K,E,NEQN,MBAN,NEQN,1)
C
C--5.0 REPORT SOLUTION {T}
C
C---- WRITE TO <FILENAME>.TMP OPTION
      IF(IWRT.EQ.1) THEN
        TIME = 0.0
        CALL FOPEN(ND1,'TMP ','NEW')
        WRITE(ND1) TIME, (E(N),N=1,NEQN)
        CALL FCLOSE(ND1)
      ENDIF
C---- PRINTABLE OUTPUT
      WRITE(NTM,2500)
      WRITE(NTM,2510) (NPR(N),E(ABS(ID(NPR(N)))),N=1,NPRT)
      WRITE(NOT,2500)
      WRITE(NOT,2510) (NPR(N),E(ABS(ID(NPR(N)))),N=1,NPRT)

      RETURN

 2300 FORMAT(/'   -- EXCITATION')
 2302 FORMAT(/
     .6X'Node          Temp.          Int.Flux          Ext.Flux
     .          Ext.Temp.')
 2310 FORMAT(' **** ERROR: (Generated) Node',I5,' is out of range.')
 2320 FORMAT(6X,I4,9X,G10.3)
 2330 FORMAT(6X,I4,19X,3(5X,G10.3))
 2500 FORMAT(/'   -- RESPONSE'//
     .6X,'Node    Temp.    Node    Temp.    Node    Temp.    Node
     .    Temp.    Node    Temp.')
 2510 FORMAT((6X,5(I4,G11.3)))

      END

C----------------------------------------------------------------HARMON
      SUBROUTINE HARMON(ID,K,C,E,EO,CH,NPR,KSTAR,ESTAR,
     .NPRT,IWRT,NNOD,NEQN,MBAN,ERR)
C-SUB: HARMON - FORMS [K*]{T*} = {E*}
C                 CALL SOLVER AND REPORTS RESULTS
C        COMPLEX WRITTEN OVER DOUBLE PRECISION STORAGE CELLS:
C            [K*] WRITTEN OVER [K]
C            {E*} WRITTEN OVER {E}
C
      IMPLICIT REAL*8 (A-H,O-Z)
C
C---- CAL-SAP: DATA & COMMON STORAGE
C
      COMMON /IOLIST/NTM,NTR,NIN,NOT,ND1,ND2;ND3,ND4
C
C---- HARMON: DATA & COMMON STORAGE
C

      REAL*8 K(NEQN,MBAN),C(NEQN,MBAN),E(NEQN),EO(NNOD),CH(NNOD),
     +Q(2),T(2),TEX(2)
      INTEGER ID(NNOD), NPR(NPRT), IJK(3)
      COMPLEX KSTAR(NEQN,MBAN), ESTAR(NEQN), QSTAR, TSTAR, TXSTAR, Z
      LOGICAL*1 ERR, TDOF
      CHARACTER ENDFLAG*3
      REAL ZLAG

      SAVE /IOLIST/

      DATA PI/3.141592654/


C
C--1.0 WRITE HEADER
C
      WRITE(NTM,2100)
      WRITE(NOT,2100)
 2100 FORMAT(/'   == STEADY HARMONIC SOLUTION')
      CALL FREETY
C
C--2.0 GET FREQUENCY
C
      CALL FREE
      CALL FREETY
      W = 0.0
```

```
      CALL FREER('F',W,1)
      WRITE(NOT,2200) W
 2200 FORMAT(/'      Frequency of excitation:',G10.3,' cycles/time')
      W = W*2.0*PI
C
C--3.0 GET [K] & [C], FORM [K*], & SCALE [K*] FOR TEMP-PRESCRIBED
NODES
C
      REWIND ND1
      READ(ND1) ((K(N,M),N=1,NEQN),M=1,MBAN),
     .((C(N,M),N=1,NEQN),M=1,MBAN)
      DO 30 I=1,NEQN
      DO 30 J=1,MBAN
   30 KSTAR(I,J) = CMPLX(K(I,J), W*C(I,J))

      DO 32 I=1,NEQN
   32 IF(TDOF(I,ID,NNOD)) KSTAR(I,1) = KSTAR(I,1)*(1.0E+15, 0.0)
C
C--4.0 GET T, Q, AND/OR TEX & FORM (E)
C
      WRITE(NTM,2400)
      WRITE(NOT,2400)
      WRITE(NOT,2402)
   40 CALL FREE
      CALL FREETY
C---- CHECK FOR "END"
      CALL FREEH(' ',ENDFLAG,3,1)
      IF(ENDFLAG.EQ.'END') GO TO 50
C---- INTERPRET LINE AND GENERATE DATA
C     CONVERT: Z = MAG*COS(-LAG*W) + i MAG*SIN(-LAG*W)
      Q(1) = 0.0
      Q(2) = 0.0
      CALL FREER('Q',Q,2)
      QSTAR = CMPLX(Q(1)*COS(-Q(2)*W),Q(1)*SIN(-Q(2)*W))
      T(1) = 0.0
      T(2) = 0.0
      CALL FREER('T',T,2)
      TSTAR = CMPLX(T(1)*COS(-T(2)*W),T(1)*SIN(-T(2)*W))
      TEX(1) = 0.0
      TEX(2) = 0.0
      CALL FREER('X',TEX,2)
      TXSTAR = CMPLX(TEX(1)*COS(-TEX(2)*W),TEX(1)*SIN(-TEX(2)*W))
      CALL FREEI(' ',IJK(1),3)
      IF(IJK(2).EQ.0) IJK(2)=IJK(1)
      IF(IJK(3).EQ.0) IJK(3)=1
      DO 42 N=IJK(1),IJK(2),IJK(3)
      IF(N.LE.0.OR.N.GT.NNOD) THEN
        WRITE(NTM,2410) N
        WRITE(NOT,2410) N
        ERR=.TRUE.
        GO TO 40
      ENDIF
      NN = ID(N)
      NNN = ABS(NN)
C---- TEMPERATURE-PRESCRIBED NODES
      IF(NN.LT.0) THEN
        WRITE(NOT,2420) N,T(1),T(2)
        ESTAR(NNN) = TSTAR*KSTAR(NNN,1)
C---- FLUX-PRESCRIBED NODES
      ELSEIF(NN.GT.0) THEN
        WRITE(NOT,2430) N,EO(N),Q(1),Q(2),TEX(1),TEX(2)
        ESTAR(NNN) = ESTAR(NNN) + CMPLX(EO(N)) + QSTAR + CH(N)*TXSTAR
      ENDIF
   42 CONTINUE
      GO TO 40
C
C--5.0 SOLVE
C
   50 IF(ERR) RETURN
      CALL SYMBCX(KSTAR,ESTAR,NEQN,MBAN,NEQN,1)
C
C--6.0 REPORT SOLUTION (T)
C
C---- WRITE TO <FILENAME>.TMP OPTION
      IF(IWRT.EQ.1) THEN
        TIME = 0.0
        CALL FOPEN(ND1,'TMP ','NEW')
        WRITE(ND1) TIME, (ESTAR(N),N=1,NEQN)
        CALL FCLOSE(ND1)
      ENDIF
C---- PRINTABLE OUTPUT
      WRITE(NOT,2600)
      WRITE(NOT,2610) (NPR(N),ABS(ESTAR(ABS(ID(NPR(N))))),
     .ZLAG(ESTAR(ABS(ID(NPR(N)))),W), N=1,NPRT)
      WRITE(NTM,2600)
      WRITE(NTM,2610) (NPR(N),ABS(ESTAR(ABS(ID(NPR(N))))),
     .ZLAG(ESTAR(ABS(ID(NPR(N)))),W), N=1,NPRT)

      RETURN

 2400 FORMAT(/'  -- EXCITATION')
 2402 FORMAT(/
     .23X,'Temperature          Ext. Flux            Ext. Temp'/
     .6X,'Node  Int.Flux   Amp.       Lag        Amp.        Lag
     .      Amp.       Lag')
 2410 FORMAT(' **** ERROR: (Generated) Node',I5,' is out of range.')
 2420 FORMAT(6X,I4,10X,2G10.3)
 2430 FORMAT(6X,I4,G10.3,20X,4G10.3)
 2600 FORMAT(/'  -- RESPONSE'//
     .13X'Temperature                Temperature
          Temperature'/
     .6X'Node   Amp.       Lag    Node   Amp.       Lag
     .     Node   Amp.       Lag')
 2610 FORMAT((6X,3(I4,2G10.3)))

      END
C-----------------------------------------------------------------ZLAG
      FUNCTION ZLAG(Z,W)
C-FUN:ZLAG - COMPUTES LAG=ARG/FREQ OF COMPLEX NUMBER
      COMPLEX Z
      REAL*8 W
      REAL ZLAG,PI,IM,RE
      DATA PI/3.141592654/
      IM = AIMAG(Z)
      RE = REAL(Z)

      IF((ABS(RE).LT.1.0E-6).OR.(W.LT.1.0D-6)) THEN
        ZLAG = 0.0
      ELSE
        ZLAG = -ATAN(IM/RE)
        IF(IM.GT.0.0.AND.RE.GT.0.0) ZLAG = 2.0*PI+ZLAG
        IF(RE.LT.0.0) ZLAG = PI+ZLAG
        ZLAG = ZLAG/REAL(W)
      ENDIF
      RETURN
      END

C-----------------------------------------------------------------
PREDIC
      SUBROUTINE PREDIC(ID,K,C,E,EO,CH,NPR,NPRT,IPRT,IWRT,
     +NNOD,NEQN,MBAN,ERR)
C-SUB: PREDIC - PREDICTOR-CORRECTOR 1ST O.D.E. EQUATION SOLVER
C              BASED ON SOLVER IN *HEAT* BY R.L.TAYLOR - U.C.
BERKELEY
C     SOLVES EQUATION;
C
C     [K](T) + [C](dT/dt) = (E(t))
C
C     FOR GENERAL EXCITATION, (E(t)), DEFINED BY A SET OF PIECEWISE
C     LINEAR FUNCTIONS.
C     METHOD BASED ON DIFFERENCE APPROXIMATION;
C
C     (T)n+1 = (T)n + (1-a)DT(dT/dt)n + (a)DT(dT/dt)n+1
C
C     WHERE;   a = "alpha", an integration parameter
C                = 0 corresponds to Forward Difference method
C                = 1 corresponds to Backward Difference method
C                = 1/2 corresponds to Crank-Nicholson method
(unstable)
C             DT = time step increment
C
C---- D I C T I O N A R Y   O F   V A R I A B L E S -------------------
C     INTERNAL TO SUBROUTINE
C
C     VARIABLE      DESCRIPTION----------------------------------------
C     NEFN
NUMBER OF EXCITATION FUNCTIONS DEFINED
C     MPIEFN
IEFN(NNOD,2)
: EXCITATION FUNCTION NUMBER ARRAY
C     MPEFN
EFN(NEFN,2)
: EXCITATION FUNCTION DATA
C                   E(NEQN)

: CURRENT (E) (ORDERED BY EQTN #)
C     MPT
T(NEQN)

: CURRENT (T) (ORDERED BY EQTN #)
C     MPTD
TD(NEQN)
: CURRENT (dT/dt) (ORDERED BY EQTN #)
C     MPTDD
TDD(NEQN)
: INITIAL (d/dt(dT/dt)) TO EST TIME STEP
C     IEFN(NNOD,2)  LIST OF FUNCTION NUMBERS (ORDERED BY NODE NUMBER)
C                   IEFN(I,1) = TEMP OR EXT.FLUX FUNCTION NUMBER
C                   IEFN(I,2) = CONVECTIVE FLUID TEMP. FUNCTION NUMBER
C     EFN(NEFN,2)   EXCITATION FUNCTION DATA
C                   EFN(I,1) = OLD VALUE
C                   EFN(I,2) = NEW VALUE
C     TOLD, TNEW    EXCITATION FUNCTION DATA TIMES
C     TIME(1)       START TIME
C     TIME(2)       END TIME
C     TIME(3)       TIME INCREMENT
C----------------------------------------------------------------------
-

      IMPLICIT REAL*8 (A-H,O-Z)
C
C---- CAL-SAP: DATA & COMMON STORAGE
C
      COMMON MTOT,NP,IA(1)
      COMMON /DBSYS/ NUMA,NEXT,IDIR,IP(3)
      COMMON /IOLIST/ NTM,NTR,NIN,NOT,ND1,ND2,ND3,ND4

C
C---- PREDIC: DATA & COMMON STORAGE
C
      REAL*8 K(NEQN,MBAN),C(NEQN,MBAN),E(NEQN),EO(NNOD),CH(NNOD)
      INTEGER ID(NNOD), NPR(NPRT)
      CHARACTER ENDFLAG*3
      LOGICAL*1 ERR

      COMMON /LIST/ TIME(3),ALPHA,TOLD,TNEW
      SAVE /DBSYS/,/IOLIST/,/LIST/
```

```
C
C---- WRITE HEADER
C
      WRITE(NOT,2000)
 2000 FORMAT(/'    == DYNAMIC SOLUTION: PREDICTOR-CORRECTOR
INTEGRATION')

C
C--1.0 GET SOLUTION CONTROL INFORMATION
C
      CALL FREE
      CALL FREETY
      CALL FREER('E',TIME(1),3)
      ALPHA = 0.75
      CALL FREER('A',ALPHA,1)
      IF(ALPHA.LT.0.0.OR.ALPHA.GT.1.0) THEN
        WRITE(NTM,2100)

WRITE(NOT,2100)

ERR=.TRUE.

RETURN
 2100 FORMAT(' **** ERROR: Alpha must be in range 0.0 to 1.0.')
      ENDIF
      NEFN = 0
      CALL FREEI('N',NEFN,1)
      IF(NEFN.LE.0) THEN
        WRITE(NTM,2110)

WRITE(NOT,2110)

ERR=.TRUE.

RETURN
 2110 FORMAT(' **** ERROR: One or more excitation functions
     . must be defined.')
      ENDIF
C---- REPORT CONTROL INFORMATION
      WRITE(NOT,2120) TIME,ALPHA,NEFN
 2120 FORMAT(/'   -- SOLUTION CONTROL INFORMATION'/
     .'       Start time ....................... ',G10.3/
     .'       End time ......................... ',G10.3/
     .'       Time step increment .............. ',G10.3/
     .'       Integration parameter, alpha ..... ',G10.3/
     .'       Number of excitation functions ... ',I6)
C
C--2.0 DEFINE ARRAYS
C
      CALL DELETE('IEFN')
      CALL DELETE('EFN ')
      CALL DELETE('T   ')
      CALL DELETE('TD  ')
      CALL DELETE('TDD ')
      CALL DEFINE('TDD ',MPTDD,NEQN,1)
      CALL DEFINE('TD  ',MPTD,NEQN,1)
      CALL DEFINE('T   ',MPT,NEQN,1)
      CALL DEFINE('EFN ',MPEFN,NEFN,2)
      CALL DEFINI('IEFN',MPIEFN,NNOD,2)
      NSTOR = (IDIR-NEXT-20)
      IF(NSTOR.LT.0) THEN
        WRITE(NTM,2200) -NSTOR

WRITE(NOT,2200) -NSTOR

ERR=.TRUE.

RETURN
 2200 FORMAT(' **** ERROR: Insufficient storage to form arrays for
     . solution.'/
     .'                    Increase IA(MTOT) by ',I6)
      ENDIF
C---- INITIALIZE
      CALL ZEROI(IA(MPIEFN),NNOD,2)
      CALL ZEROR(IA(MPEFN),NEFN,2)
      CALL ZEROR(IA(MPT),NEQN,1)
      CALL ZEROR(IA(MPTD),NEQN,1)
      CALL ZEROR(IA(MPTDD),NEQN,1)
C
C--3.0 GET EXCITATION CONTROL INFORMATION
C
      CALL ECNTRL(ID,IA(MPIEFN),NNOD,ERR)
      IF(ERR) RETURN
C
C--4.0 GET INITIAL CONDITIONS
C
C---4.1 WRITE HEADER
C
      WRITE(NTM,2400)
      WRITE(NOT,2400)
 2400 FORMAT(/'   -- INITIAL CONDITIONS')
C
C---4.2 FIND SEPARATOR "INITIAL"
C
      CALL FINDN('INITIAL',7,KEY)
      IF(KEY.EQ.1) THEN
        WRITE(NTM,2410)

WRITE(NOT,2410)
 2410 FORMAT(/'   -- NOTE: Initial condition data not found.'/
     .'              Initial temperatures assumed to be zero.')
        GO TO 50
      ENDIF
      CALL FREETY
C
C---4.3 CALL ICOND TO DO THE WORK
C
      CALL ICOND(ID,IA(MPT),NNOD,NEQN,ERR)
C
C--5.0 & READ [K] & [C] FROM <filename>.SYS
C
   50 REWIND ND1
      READ(ND1) ((K(N,M),N=1,NEQN),M=1,MBAN),
     .((C(N,M),N=1,NEQN),M=1,MBAN)
C
C--6.0 INITIALIZE EXCITATION FUNCTION VALUES
C
      CALL FINDN('EXCIT',5,KEY)
      IF(KEY.EQ.1) THEN
        WRITE(NTM,2600)
        WRITE(NOT,2600)
 2600 FORMAT(' **** ERROR: Separator "EXCIT" not found.')
        ERR=.TRUE.
        RETURN
      ENDIF

      DO 60 I=1,2
      CALL GETEFN(IA(MPEFN),TOLD,TNEW,NEFN,ENDFLAG)
      IF(ENDFLAG.EQ.'END') THEN
        WRITE(NTM,2610)

WRITE(NOT,2610)

RETURN
   60 ENDIF
 2610 FORMAT(' **** ERROR: Insufficient excitation data: at least
     + two values must be given for excitation functions.')
C
C--7.0 CALL PREDI TO DO THE WORK
C
      CALL PREDI(ID,K,C,EO,IA(MPT),CH,NPR,IA(MPIEFN),IA(MPEFN),E,
     .IA(MPTD),IA(MPTDD),NPRT,IPRT,IWRT,NEFN,NNOD,NEQN,MBAN,ERR)

      RETURN
      END

C-----------------------------------------------------------------------
ICOND
      SUBROUTINE ICOND(ID,T,NNOD,NEQN,ERR)
C-SUB:ICOND - READS AND GENERATES INITIAL CONDITIONS
C
      IMPLICIT REAL*8 (A-H,O-Z)
C
C---- CAL-SAP: DATA & COMMON STORAGE
C
      COMMON /IOLIST/ NTM,NTR,NIN,NOT,ND1,ND2,ND3,ND4
      COMMON /PARC/ FIN(12),EXT(3)
C
C---- ICOND: DATA & COMMON STORAGE
C
      REAL*8 T(NEQN)
      INTEGER ID(NNOD), IJK(3)
      CHARACTER ENDFLAG*3, READ*5, FN(12)*1
      LOGICAL*1 ERR

      SAVE /IOLIST/,/PARC/

C
C--1.0 CHECK TO SEE IF READ-FROM-DISK OPTION IS REQUESTED
C
      CALL FREE
      CALL FREEH(' ',READ,5,1)
      IF(READ.EQ.'READ=') THEN
        CALL FREETY
C----   GET FILENAME  AND CREATE EXTENSION
C --    STORE CURRENT FILENAME TEMPORARILY
        DO 10 I=1,12
   10   FN(I) = FIN(I)
        CALL FREEH('D',FIN,1,12)
C----   OPEN FILE <filename>.TMP, MOVE TO EOF, BACKSPACE ONE RECORD
        CALL FOPEN(ND1,'TMP ','OLD')
   12   READ(ND1, END=14)
        GO TO 12
   14   BACKSPACE ND1
C----   READ INITIAL CONDITIONS FROM LAST RECORD
        READ(ND1) TIME,(T(N),N=1,NEQN)
        CALL FCLOSE(ND1)
        WRITE(NOT,2100) (FIN(N),N=1,12),'.',(EXT(N),N=1,3)

WRITE(NOT,2200)
        WRITE(NOT,2400) (N,T(ABS(ID(N))),N=1,NNOD)
C --    RESTORE CURRENT FILENAME
        DO 16 I=1,12
   16   FIN(I) = FN(I)
        RETURN
      ENDIF
C
C--2.0 WRITE INITIAL CONDITIONS HEADER
C
      WRITE(NOT,2200)
C
C--3.0 INTERPRET LINE OF DATA
C
   30 CALL FREETY
C---- CHECK FOR "END"
      CALL FREEH(' ',ENDFLAG,3,1)
      IF(ENDFLAG.EQ.'END') GO TO 40
```

```
C---- DO IT
      TEMP = 0.0
      CALL FREER('T',TEMP,1)
      CALL FREEI(' ',IJK(1),3)
      IF(IJK(2).EQ.0) IJK(2)=IJK(1)
      IF(IJK(3).EQ.0) IJK(3)=1
      DO 34 N=IJK(1),IJK(2),IJK(3)
      IF(N.LE.0.OR.N.GT.NNOD) THEN

WRITE(NTM,2300) N

WRITE(NOT,2300) N

ERR=.TRUE.

GO TO 36
      ENDIF
   34 T(ABS(ID(N))) = TEMP
C---- GET NEXT LINE OF DATA
   36 CALL FREE
      GO TO 30
C
C--4.0 REPORT INITIAL CONDITIONS IF NO ERROR ENCOUNTERED
C
   40 IF(ERR) RETURN
      WRITE(NOT,2400) (N,T(ABS(ID(N))),N=1,NNOD)

      RETURN

 2100 FORMAT(/'   -- NOTE: Initial conditions read from last record
of:'
     .16A1)
 2200 FORMAT(/
     .6X,'Node    Temp.    Node    Temp.    Node    Temp.    Node
     .    Temp.    Node    Temp.')
 2300 FORMAT(' **** ERROR: (Generated) Node ',I5,' is out of range.')
 2400 FORMAT((6X,5(I4,G11.3)))

      END

C----------------------------------------------------------------PREDI
      SUBROUTINE PREDI(ID,K,C,EO,T,CH,NPR,IEFN,EFN,E,TD,TDD,
     +NPRT,IPRT,IWRT,NEFN,NNOD,NEQN,MBAN,ERR)
C-SUB:PREDI - THE KERNEL OF PREDIC

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

C
C---- CAL-SAP: DATA & COMMON STORAGE
C
      COMMON /IOLIST/NTM,NTR,NIN,NOT,ND1,ND2,ND3,ND4
C
C---- LISTP: DATA & COMMON STORAGE
C
      REAL*8 K(NEQN,MBAN),C(NEQN,MBAN),EO(NNOD),T(NEQN),CH(NNOD),
     .EFN(NEFN,2),E(NEQN),TD(NEQN),TDD(NEQN)
      INTEGER ID(NNOD), NPR(NPRT), IEFN(NNOD,2)
      LOGICAL*1 ERR, SKIP, TDOF

      COMMON /LIST/ TIME(3),ALPHA,TOLD,TNEW
      SAVE /IOLIST/,/LIST/

      WRITE(NOT,2000)
      WRITE(NTM,2000)
 2000 FORMAT(/'   -- TIME STEP')

C
C--1.0 COMPUTE INITIAL TEMPERATURE RATES: {dT(0)/dt}
C
C---1.1 FOR TEMP-DOF: SCALE [C]=[C]*1E15 OR SET [C]=1E15 FOR
C       MASSLESS NODES
C
      SKIP = .FALSE.
      DO 10 N=1,NEQN
      IF(TDOF(N,ID,NNOD)) THEN
        C(N,1) = C(N,1)*1.0E15

IF(C(N,1).EQ.0.0) C(N,1) = 1.0E15
      ENDIF
   10 CONTINUE
C
C---   CHECK FOR STEP CHANGE IN SPECIFIED INITIAL TEMP
C
      DO 12 N=1,NNOD
      NN = ID(N)
      IF(NN.LT.O) THEN
        NNN = ABS(NN)

NFNT = IEFN(N,1)
        TEMP = 0.0

IF(NFNT.NE.O) TEMP = EFN(NFNT,1)
        IF(T(NNN).NE.TEMP) THEN

  WRITE(NTM,2100)

  WRITE(NOT,2100)

  SKIP = .TRUE.
        ENDIF
      ENDIF
   12 CONTINUE
 2100 FORMAT(/'   -- NOTE: Unable to estimate time step for initial
     + step change in specified nodal temperature(s).')
```

```
C
C---1.2 FORM {E(t=0)}
C
      TM = 0.0
      CALL EXCIT(ID,C,T,TD,E,CH,EO,IEFN,EFN,TM,NPRT,IPRT,IWRT,
     +NEFN,NNOD,NEQN,MBAN,ERR)
      IF(ERR) RETURN

C
C---1.3 FORM RHS {E}-[K]{T} FOR FLUX-DOF, (dT/dt)*DIAG[C] FOR TEMP-DOF
C
      CALL RHS(ID,T,TD,E,K,C,NNOD,NEQN,MBAN)
C
C---1.4 SOLVE [C]{dT/dt} = {E}
C
      CALL SYMBC(C,TD,NEQN,MBAN,NEQN,1)
      IF(SKIP) GO TO 30
C
C--2.0 COMPUTE TIMESTEP CHECK
C
C---2.1 COMPUTE INITIAL RATE OF TEMP RATES: d/dt{dT(0)/dt}
C
      CALL RHS(ID,TD,TDD,TDD,K,C,NNOD,NEQN,MBAN)
      CALL SYMBC(C,TDD,NEQN,MBAN,NEQN,2)
C
C---2.2 COMPUTE NORMS: ||{T(0)}||, ||{dT(0)/dt}||, ||d/dt{dT(0)/dt}||
C
      TN = 0.0
      TDN = 0.0
      TDDN = 0.0
      DO 22 N=1,NEQN
      TN = TN + T(N)**2
      TDN = TDN + TD(N)**2
   22 TDDN = TDDN + TDD(N)**2
      TN = SQRT(TN)
      TDN = SQRT(TDN)
      TDDN = SQRT(TDDN)
C
C---2.3 EVALUATE TAYLORS EXPRESSION FOR TIME STEP ESTIMATE
C
      B = 0.05
      IF(TDDN.NE.0.0) THEN
        DTEST = (B*TDN + SQRT(B*B*TDN*TDN + 2.0*B*TN*TDDN))/TDDN
        WRITE(NTM,2200) B*100.0, DTEST,TIME(3)
        WRITE(NOT,2200) B*100.0, DTEST,TIME(3)
 2200 FORMAT(/'   -- NOTE: Estimated time step to limit error to
     . approx.',F5.2,'% is:',G10.3,/
     .'             Specified time step is:',G10.3)
      ELSE
        WRITE(NTM,2210)

WRITE(NOT,2210)
 2210 FORMAT(/'   -- NOTE: Unable to estimate time step to limit
     . error for the given system.')
      ENDIF
C
C--3.0 READ [C] & [K] FROM DISK
C
   30 REWIND ND1
      READ(ND1) ((K(N,M),N=1,NEQN),M=1,MBAN),
     .((C(N,M),N=1,NEQN),M=1,MBAN)
C
C--4.0 FORM LHS: [[C] + aDT[K]) SCALING TEMP-DOF DIAG TERMS
C
      ADT = ALPHA*TIME(3)
      DTA = TIME(3) - ADT
      DO 40 N=1,NEQN
      DO 40 M=1,MBAN
   40 C(N,M) = C(N,M) + ADT*K(N,M)
      DO 42 N=1,NEQN
   42 IF(TDOF(N,ID,NNOD)) C(N,1) = C(N,1)*1.0E15
C
C--5.0 TIME STEP THRU SOLUTION
C
      WRITE(NOT,2500)
      WRITE(NTM,2500)
 2500 FORMAT(/'   -- RESPONSE')

      IOP = 1
      ISTEP = 0
      DO 500 TM=TIME(1)+TIME(3),TIME(2),TIME(3)
      ISTEP = ISTEP + 1
C
C---5.1 FORM {E}
C
      CALL EXCIT(ID,C,T,TD,E,CH,EO,IEFN,EFN,TM,NPRT,IPRT,IWRT,
     +NEFN,NNOD,NEQN,MBAN,ERR)
      IF(ERR) RETURN

C
C---5.1 PARTIAL UPDATE OF T: {T} = {T} + (1-a)DT{dT/dt}
C
      DO 51 N=1,NEQN
   51 T(N) = T(N) + DTA*TD(N)
C
C---5.3 FORM RHS:{E}-[K]{T} FOR FLUX-DOF, {dT/dt}*DIAG[C] FOR TEMP-DOF
C
      CALL RHS(ID,T,TD,E,K,C,NNOD,NEQN,MBAN)
C
C---5.4 SOLVE FOR {dT/dt}
C
      CALL SYMBC(C,TD,NEQN,MBAN,NEQN,IOP)
      IOP = 2
```

```
C
C---5.5 COMPLETE UPDATE OF T: {T} = {T} + aDT{dT/dt}
C
      DO 55 N=1,NEQN
   55 T(N) = T(N) + ADT*TD(N)
C
C---5.6 REPORT RESULTS
C
      IF(MOD(ISTEP,IPRT).EQ.0) THEN
C---- WRITE TO <FILENAME>.TMP OPTION
        IF(IWRT.EQ.1) THEN
          CALL FOPEN(ND1,'TMP ','NEW')
          WRITE(ND1) TM, (T(N),N=1,NEQN)
          CALL FCLOSE(ND1)
        ENDIF
C---- PRINTABLE OUTPUT
        WRITE(NOT,2510) TM

WRITE(NOT,2520)
        WRITE(NOT,2530) (NPR(N),T(ABS(ID(NPR(N)))),N=1,NPRT)
        WRITE(NTM,2510) TM

WRITE(NTM,2520)
        WRITE(NTM,2530) (NPR(N),T(ABS(ID(NPR(N)))),N=1,NPRT)
      ENDIF

  500 CONTINUE
      RETURN
 2510 FORMAT(/'       Time: ',G11.3)
 2520 FORMAT(
     .6X,'Node    Temp.    Node    Temp.    Node    Temp.    Node
     . Temp.    Node    Temp.')
 2530 FORMAT((6X,5(I4,G11.3)))
      END

C---------------------------------------------------------------ECNTRL
      SUBROUTINE ECNTRL(ID,IEFN,NNOD,ERR)
C-SUB: ECNTRL - FORMS EXCITATION FUNCTION NUMBER ARRAY IEFN(NNOD,2)
C
C---- CAL-SAP: DATA & COMMON STORAGE
C
      COMMON /IOLIST/NTM,NTR,NIN,NOT,ND1,ND2,ND3,ND4
C
C---- ECNTRL: DATA & COMMON STORAGE
C
      INTEGER ID(NNOD),IEFN(NNOD,2),IJK(3)
      LOGICAL*1 ERR
      CHARACTER ENDFLAG*3

      SAVE /IOLIST/

      WRITE(NOT,2000)
 2000 FORMAT(/'   -- EXCITATION FUNCTION NUMBERS'//
     .6X'Node             Temp        Ext.Flux        Ext.Temp.')
   10 CALL FREE
      CALL FREETY
C---- CHECK FOR "END"
      CALL FREEH(' ',ENDFLAG,3,1)
      IF(ENDFLAG.EQ.'END') RETURN
C---- INTERPRET LINE AND GENERATE DATA
      IQ = 0
      CALL FREEI('Q',IQ,1)
      IT = 0
      CALL FREEI('T',IT,1)
      ITEX = 0
      CALL FREEI('X',ITEX,1)
      CALL FREEI(' ',IJK(1),3)
      IF(IJK(2).EQ.0) IJK(2)=IJK(1)
      IF(IJK(3).EQ.0) IJK(3)=1
      DO 12 N=IJK(1),IJK(2),IJK(3)
      IF(N.LE.0.OR.N.GT.NNOD) THEN
        WRITE(NTM,2010) N
        WRITE(NOT,2010) N
        ERR=.TRUE.
        GO TO 10
 2010 FORMAT(' **** ERROR: (Generated) Node',I5,' is out of range.')
      ENDIF
      NN = ID(N)
      NNN = ABS(NN)
C---- TEMPERATURE-PRESCRIBED NODES
      IF(NN.LT.0) THEN
        WRITE(NOT,2020) N,IT

IEFN(N,1) = IT
 2020 FORMAT(6X,I4,10X,I5)
C---- FLUX-PRESCRIBED NODES
      ELSEIF(NN.GT.0) THEN
        WRITE(NOT,2030) N,IQ,ITEX
        IEFN(N,1) = IQ

IEFN(N,2) = ITEX
 2030 FORMAT(6X,I4,20X,I5,10X,I5)
      ENDIF
   12 CONTINUE
      GO TO 10

      END
C---------------------------------------------------------------GETEFN
      SUBROUTINE GETEFN(EFN,TOLD,TNEW,NEFN,ENDFLAG)
C-SUB:GETEFN - READ INPUT FILE TO UPDATE EXCITATION FUNCTION DATA

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

      REAL*8 EFN(NEFN,2)
```

```
      CHARACTER ENDFLAG*3

C
C--1.0 UPDATE OLD VALUES
C
      TOLD = TNEW
      DO 10 I=1,NEFN
   10 EFN(I,1) = EFN(I,2)
C
C--2.0 READ NEW VALUES
C
      CALL FREE
      CALL FREETY
C---- CHECK FOR "END"
      CALL FREEH(' ',ENDFLAG,3,1)
      IF(ENDFLAG.EQ.'END') RETURN
      CALL FREER('T',TNEW,1)
      CALL FREER('N',EFN(1,2),NEFN)

      RETURN
      END
C---------------------------------------------------------------RHS
      SUBROUTINE RHS(ID,T,TD,E,K,C,NNOD,NEQN,MBAN)
C-SUB:RHS - FORMS RHS OF [C]{dT/dt} = {E*} = {E} - [K]{T};
C
C          {E*(t)} = {E(t)}-[K]{T(t)}       FOR FLUX-DOF
C          {E*(t)} = {dT(t)/dt}*DIAG OF [C]     FOR TEMP-DOF
C
C
C          {E*} IS WRITTEN OVER {TD}
C-----------------------------------------------------------------

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

      REAL*8 T(NEQN),TD(NEQN),E(NEQN),K(NEQN,MBAN),C(NEQN,MBAN)
      INTEGER ID(NNOD)
      LOGICAL*1 TDOF

      DO 20 N=1,NEQN
C---- SCALE BY DIAGONAL FOR TEMP PRESCRIBED NODES
      IF(TDOF(N,ID,NNOD)) THEN
        TD(N) = TD(N)*C(N,1)
C---- FORM [E]-[K]{T} WHERE [K] IS IN COMPACT STORAGE
      ELSE
        TEMP = E(N) - K(N,1)*T(N)

KK=N

        LL=N

DO 10 I=2,MBAN

KK = KK-1

IF(KK.GT.0) TEMP = TEMP - K(KK,I)*T(KK)

LL = LL+1

IF(LL.LE.NEQN) TEMP = TEMP - K(N,I)*T(LL)
   10   CONTINUE
        TD(N) = TEMP
      ENDIF
   20 CONTINUE
      RETURN
      END

C-------------------------------------------------------------------EXCIT
      SUBROUTINE EXCIT(ID,C,T,TD,E,CH,EO,IEFN,EFN,TM,NPRT,IPRT,IWRT,
     +NEFN,NNOD,NEQN,MBAN,ERR)
C--SUB: EXCIT - FOR t=TIME;
C      FOR FLUX-DOF COMPUTES {E(t)};
C
C      {E(t)} = {EO} + {Q(t)} + {CH}T(TEX(t))
C
C      FOR TEMP-DOF COMPUTES {dT/dt(t)}
C
C      {dT(t)/dt} = (1/Dt){T(t) - T(t-Dt)}
C
C      FROM EXCITATION FUNCTION DATA STORED AND READ INTO EFN(NEFN,2)
C      AND LIST OF FUNCTION NUMBERS
C-------------------------------------------------------------------

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

C
C---- CAL-SAP: DATA & COMMON STORAGE
C
      COMMON /IOLIST/NTM,NTR,NIN,NOT,ND1,ND2,ND3,ND4
C
C---- EXCIT: DATA & COMMON STORAGE
C
      REAL*8 C(NEQN,MBAN),T(NEQN),TD(NEQN),CH(NNOD),EO(NNOD),E(NEQN),
     +EFN(NEFN,2)
      INTEGER ID(NNOD), IEFN(NNOD,2)
      LOGICAL*1 ERR
      CHARACTER ENDFLAG*3

      COMMON /LIST/ TIME(3),ALPHA,TOLD,TNEW
      SAVE /IOLIST/,/LIST/

C
C--1.0 UPDATE EXCITATION FUNCTION DATA IF NEEDED
C
   10 IF(TM.GT.TNEW) THEN
```

```
        CALL GETEFN(EFN,TOLD,TNEW,NEFN,ENOFLAG)

 IF(ENDFLAG.EQ.'ENO') THEN

  WRITE(NTM,2100)

  WRITE(NOT,2100)
 2100     FORMAT(' **** ERROR: Insufficient excitation data.')

  ERR = .TRUE.
          RETURN

ENDIF

GO TO 10
      ELSEIF(TM.LT.TOLD) THEN
        WRITE(NTM,2110) TM

WRITE(NOT,2110) TM
 2110 FORMAT(' **** ERROR: Excitation data not defined for current
     . time:',G10.3)
        ERR=.TRUE.

RETURN
      ENDIF
C---- INTERPOLATION FRACTION
      XT = (TM-TOLD)/(TNEW-TOLD)
C
C--2.0 FORM (E) OR (dT/dt)
C
      CALL ZEROR(E,NEQN,1)

      DO 20 N=1,NNOD
      NN = ID(N)
      NNN = ABS(NN)
C---- FLUX-DOF: SUM INT. FLUX + EXT. FLUX + EFFECT. CONVECTIVE FLUX
      IF(NN.GT.0) THEN
        NFNQ = IEFN(NNN,1)

NFNC = IEFN(NNN,2)

Q = 0.0

IF(NFNQ.NE.0) Q = XT*(EFN(NFNQ,2)-EFN(NFNQ,1)) + EFN(NFNQ,1)

CONVE = 0.0

IF(NFNC.NE.0) CONVE = CH(N)*(XT*(EFN(NFNC,2) - EFN(NFNC,1))
     +    + EFN(NFNC,1))
        E(NNN) = E(NNN) + EO(N) + Q + CONVE
C---- TEMP-DOF: COMPUTE (dT/dt) FROM PRESCRIBEO TEMP
      ELSEIF(NN.LT.0) THEN
        NFNT = IEFN(NNN,1)

TEMP = 0.0

IF(NFNT.NE.O) TEMP = XT*(EFN(NFNT,2)-EFN(NFNT,1)) + EFN(NFNT,1)

TO(NNN) = (TEMP - T(NNN))/TIME(3)
      ENOIF
   20 CONTINUE

      RETURN
      ENO

C-----------------------------------------------------------------REPORT
      SUBROUTINE REPORT(MPNPR,NPRT,IPRT,IWRT,NNOD,ERR)
C-SUB: REPORT - READS REPORT CONTROL INFORMATION
C                FORMS REPORT CONTROL ARRAY NPR(NPRT)
C
C---- CAL-SAP: DATA & COMMON STORAGE
C
      CHARACTER EXT*1, FIN*1
      COMMON MTOT,NP,IA(1)
      COMMON /DBSYS/ NUMA,NEXT,IDIR,IP(3)
      COMMON /IOLIST/ NTM,NTR,NIN,NOT,ND1,ND2,ND3,ND4
      COMMON /PARC/ FIN(12),EXT(80)
      SAVE /DBSYS/,/IOLIST/,/PARC/
C
C---- REPORT: DATA & COMMON STORAGE
C
      INTEGER IJK(3)
      LOGICAL*1 ERR
      CHARACTER ENDFLAG*3, WRITEFLAG*5

      SAVE /IOLIST/
C
C--0.0 WRITE HEADER
C
      WRITE(NTM,2000)
 2000 FORMAT(/'    -- REPORT CONTROL')

C
C--1.0 FIND "REPORT"
C
      CALL FINDN('REPORT',6,KEY)
      IF(KEY.EQ.1) THEN
        WRITE(NTM,2100)
        WRITE(NOT,2100)
        ERR=.TRUE.
        RETURN
      ENOIF
      CALL FREETY
C
```

```
C--2.0 GET PRINT INTERVAL, WRITE OPTION
C
      CALL FREE
      CALL FREETY
      CALL FREEH(' ',WRITEFLAG,5,1)
      IWRT=0
      IF(WRITEFLAG.EQ.'WRITE') THEN
        IWRT = 1
        WRITE(NTM,2200)
        WRITE(NOT,2200)
      ENDIF

      IPRT=1
      CALL FREEI('T',IPRT,1)
C
C--3.0 FORM PRINT CONTROL ARRAY
C
C---- DEFINE ARRAY NPR(1,1) TO GET POINTER & CREATE DIRECTORY ENTRY
      CALL DELETE('NPR ')
      CALL DEFINE('NPR ',MPNPR,1,1)
C---- GENERATE PRINT CONTROL ARRAY WHILE BLANK COMMON SPACE AVAILABLE
      NPRT=0
   30 CALL FREE
      CALL FREETY
C---- CHECK FOR "END"
      CALL FREEH(' ',ENDFLAG,3,1)
      IF(ENDFLAG.EQ.'END') GO TO 36
C---- DO IT
      CALL FREEI(' ',IJK(1),3)
      IF(IJK(2).EQ.0) IJK(2)=IJK(1)
      IF(IJK(3).EQ.0) IJK(3)=1
      DO 34 N=IJK(1),IJK(2),IJK(3)
      IF(N.LE.0.OR.N.GT.NNOD) THEN
        WRITE(NTM,2300) N
        WRITE(NOT,2300) N
        ERR=.TRUE.
        GO TO 30
      ENDIF
      NN = MPNPR+N-1
C---- CHECK BALNK COMMON SPACE
      IF(NN.GT.IOIR) THEN
        WRITE(NTM,2310)

WRITE(NOT,2310)

ERR=.TRUE.

RETURN
      ENDIF
      NPRT = NPRT + 1
   34 IA(NN) = N
      GO TO 30
C---- REOEFINE DIRECTORY FOR ACTUAL SIZE OF NPR ARRAY
   36 CALL DELETE('NPR ')
      CALL DEFINE('NPR ',MPNPR,NPRT,1)

      RETURN
 2100 FORMAT(' **** ERROR: No report control information found.')
 2200 FORMAT(
     .'    -- Solution will be written to binary file <filename>.TMP
     . in form (TIME,T(N),N=1,NNOD).')
 2300 FORMAT(' **** ERROR: (Generated) Node,'I5' selected for
printable
     . output is out of range.')
 2310 FORMAT(
     .' **** ERROR: Insufficient storage to form print control array.'
     .'             Increase IA(MTOT) by ',I6)

      END
C-------------------------------------------------------------------
TDOF
      FUNCTION TDOF(NEQ,IO,NNOD)
C-FUN:TDOF - DETERMINES IF EQUATION NUMBER NEQ IS A TEMP DOF
      LOGICAL*1 TDOF
      INTEGER IO(NNOD)
      TDOF = .FALSE.
      DO 10 N=1,NNOD
       IF((IO(N).LT.0).AND.(ABS(ID(N)).EQ.NEQ)) THEN
         TDOF = .TRUE.
         RETURN
       ENDIF
   10 CONTINUE
      RETURN
      END

C*******************************************************************
C    C A L  -  S A P   LIBRARY EXTENSIONS
C*******************************************************************

C-----------------------------------------------------------------FREETY
      SUBROUTINE FREETY
C--SUB:FREETY - TYPE COMMAND LINE TO SCREEN
      CHARACTER*1 LINE
      COMMON /ILINE/ II
      COMMON /CLINE/ LINE(160)
      COMMON /IOLIST/ NTM,NTR,NIN,NOT,ND1,ND2,ND3,NO4
      SAVE /ILINE/,/CLINE/,/IOLIST/

      WRITE(NTM,2000) (LINE(I),I=1,II)
      RETURN

 2000 FORMAT (1X,80A1)
      ENO
```

```
C---------------------------------------------------------------FINDN
      SUBROUTINE FINDN(SEP,NC,KEY)
      CHARACTER*1 LINE,SEP(NC)
      COMMON /CLINE/ LINE(160)
      COMMON /ILINE/ II
      COMMON /IOLIST/ NTM,NTR,NIN,NOT,ND1,ND2,ND3,ND4
      SAVE /CLINE/,/ILINE/,/IOLIST/
C---- FIND SEPARATOR OF NC CHARACTERS IN INPUT FILE
      KEY=0
      REWIND NIN
   50 CONTINUE
      READ(NIN,1000,ERR=200,END=200) (LINE(I),I=1,NC)
      II = NC
      CALL UPPER
      DO 60 N=1,NC
   60 IF(SEP(N).NE.LINE(N)) GO TO 50
      GO TO 900

  200 KEY=1
  900 RETURN

 1000 FORMAT (80A1)
      END
C-----------------------------------------------------------------SAVE
      SUBROUTINE SAVE
C--SUB:SAVE - WRITES INCORE DATA BASE TO <FILENAME>.DBS
      CHARACTER EXT*1, FIN*1
      COMMON MTOT,NP,IA(10000)
      COMMON /DBSYS/ NUMA,NEXT,IDIR,IP(3)
      COMMON /IOLIST/ NTM,NTR,NIN,NOT,ND1,ND2,ND3,ND4
      COMMON /PARC/ FIN(12),EXT(80)
      SAVE /DBSYS/,/IOLIST/,/PARC/

      CALL FOPEN(ND1,'DBS ','NEW')
      WRITE(ND1) NUMA,NEXT,IDIR,MTOT,IP
      WRITE(ND1) (IA(I),I=1,NEXT),(IA(J),J=IDIR,MTOT)
      CALL FCLOSE(ND1)

      RETURN
      END
C--------------------------------------------------------------ZEROI
      SUBROUTINE ZEROI(IA,NR,NC)
      DIMENSION IA(NR,NC)
      DO 10 I=1,NR
      DO 10 J=1,NC
      IA(I,J) = 0
   10 CONTINUE
      RETURN
      END
C--------------------------------------------------------------ZEROR
      SUBROUTINE ZEROR(A,NR,NC)
      REAL*8 A(NR,NC)
      DO 10 I=1,NR
      DO 10 J=1,NC
      A(I,J) = 0.0
   10 CONTINUE
      RETURN
      END
C--------------------------------------------------------------SYMBC
      SUBROUTINE SYMBC(C,B,MAX,MB,ND,IOP)
C--SUB:SYMBC - SYMMETRIC COMPACT-FORM EQUATION SOLVER
C  SOLVES SYMMETRIC BANDED EQUATIONS;
C
C                 [C] {X} = {B}
C
C  IN-CORE WITB [C] STORED IN COMPACT FORM C(ND,MB).
C  SOLUTION X(ND) IS WRITTEN OVER B(ND).
C
C      MAX = MAXIMUM DOF FOR PARTIAL L*D*U DECOMPOSITION
C      MB  = (HALF) BANDWIDTH
C      ND  = NUMBER OF DEGREES OF FREEDOM
C      IOP = 1 : COMPLETE SOLVE
C      IOP = 2 : RESOLVE
C

      IMPLICIT REAL*8 (A-B,O-Z)
      DIMENSION C(ND,MB), B(ND)

C-I--PERFORM THE L*D*U DECOMPOSITION OF C AND FORM Y
      NBM = MB-1
      DO 300 M=2,MAX
      N = M-1
      IF(C(N,1).EQ.0.0) GO TO 300
      WN = B(N)
      NNB = MIN((N+NBM),MAX)
      KJ = NNB - N + 1
      II = 1
      DO 200 I = M,NNB
      II = II + 1
      IF(C(N,II).EQ.0.0) GO TO 200
      GIN = C(N,II)/C(N,1)
      B(I) = B(I) - GIN*WN
      IF(IOP.EQ.2) GO TO 200
      J = 0
      DO 100 K=II,KJ
      J = J + 1
 100  C(I,J) = C(I,J) - C(N,K)*GIN
 200  CONTINUE
 300  CONTINUE

C-2--PERFORM BACKSUBSTITUTION

      M = MAX
 400  IF(C(M,1).NE.0.0) B(M) = B(M)/C(M,1)
      M = M - 1
      IF(M.LE.0) GO TO 600
      KJ = MIN(MB,MAX-M+1)
      J = M
      DO 500 N=2,KJ
      J = J + 1
 500  B(M) = B(M) - C(M,N)*B(J)
      GO TO 400
 600  RETURN
      END
C------------------------------------------------------------SYMBCX
      SUBROUTINE SYMBCX(C,B,MAX,MB,ND,IOP)
C--SUB:SYMBCX - SYMMETRIC COMPLEX COMPACT-FORM EQUATION SOLVER
C  SOLVES SYMMETRIC BANDED COMPLEX EQUATIONS;
C
C                 [C*] {X*} = {B*}
C
C  IN-CORE WITB [C*] STORED IN COMPACT FORM C(ND,MB).
C  SOLUTION X*(ND) IS WRITTEN OVER B*(ND).
C
C      MAX = MAXIMUM DOF FOR PARTIAL L*D*U DECOMPOSITION
C      MB  = (HALF) BANDWIDTB
C      ND  = NUMBER OF DEGREES OF FREEDOM
C      IOP = 1 : COMPLETE SOLVE
C      IOP = 2 : RESOLVE
C

      IMPLICIT COMPLEX (A-B,O-Z)
      DIMENSION C(ND,MB), B(ND)

C-1--PERFORM THE L*D*U DECOMPOSITION OF C AND FORM Y
      NBM = MB-1
      DO 300 M=2,MAX
      N = M-1
      IF(C(N,1).EQ.(0.0,0.0)) GO TO 300
      WN = B(N)
      NNB = MIN((N+NBM),MAX)
      KJ = NNB - N + 1
      II = 1
      DO 200 I = M,NNB
      II = II + 1
      IF(C(N,II).EQ.(0.0,0.0)) GO TO 200
      GIN = C(N,II)/C(N,1)
      B(I) = B(I) - GIN*WN
      IF(IOP.EQ.2) GO TO 200
      J = 0
      DO 100 K=II,KJ
      J = J + 1
 100  C(I,J) = C(I,J) - C(N,K)*GIN
 200  CONTINUE
 300  CONTINUE

C-2--PERFORM BACKSUBSTITUTION

      M = MAX
 400  IF(C(M,1).NE.(0.0,0.0)) B(M) = B(M)/C(M,1)
      M = M - 1
      IF(M.LE.0) GO TO 600
      KJ = MIN(MB,MAX-M+1)
      J = M
      DO 500 N=2,KJ
      J = J + 1
 500  B(M) = B(M) - C(M,N)*B(J)
      GO TO 400
 600  RETURN
      END

C-----------------------------------------------------------------
CROUT
      SUBROUTINE CROUT(A,B,N,ND,LD,KEY)
C-----------------------------------------------
C     GENERAL EQUATION SOLVER - M.I. BOIT 1/17/83
C     SYMMETRIC AND NON-SYMMETRIC
C
C     KEY = 0 TRIANGULARIZE AND SOLVE
C     KEY = 1 TRIANGULARIZE ONLY
C     KEY = 2 FORWARD AND BACK SUBSTITUTION ONLY
C-----------------------------------------------
      IMPLICIT REAL*8 (A-B, O-Z)
      DIMENSION A(ND,ND),B(ND,LD)
C
      II = 1
      IF(A(1,1).NE.0.0) GO TO 10
      PAUSE ' **** ERROR:SUB:CROUT: Attempt to solve singular system.'
      GO TO 900
   10 IM = 1
      IF(KEY.EQ.2) GO TO 200
      IF(N.EQ.1) GO TO 200
      DO 140 J=2,N
      JM = J - 1
      A(J,I) = A(J,1)/A(1,1)
      IF(J.EQ.2) GO TO 130
      DO 120 I=2,JM
      IM = I - 1
      DO 100 K=1,IM
C-----FORM L(I,J) & U(I,J)
      A(J,I) = A(J,I) - A(J,K)*A(K,I)
  100 A(I,J) = A(I,J) - A(I,K)*A(K,J)
      D = A(I,I)
C-----CHECK IF A ZERO IS ON DIAGONAL
      IF(D.NE.0.0) GO TO 120
      PAUSE ' **** ERROR:SUB:CROUT: Attempt to solve singular system.'
```

```
      GO TO 900
  120 A(J,I) = A(J,I)/D
  130 DO 140 K=1,JM
  140 A(J,J) = A(J,J) - A(J,K)*A(K,J)
C-----FORWARD AND BACK SUBSTITUTE
  200 IF(KEY.EQ.1) GO TO 900
      DO 400 L=1,LD
C-----FORM Y(I,L)
      IF(N.EQ.1) GO TO 300
      DO 210 I=2,N
      IM = I - 1
      DO 210 K=1,IM
  210 B(I,L) = B(I,L) - A(I,K)*B(K,L)
C-----FORM X(I,L)
  300 B(N,L) = B(N,L)/A(N,N)
      IF(N.EQ.1) GO TO 400
      IM = N - 1
      DO 330 I=2,N
      JM = IM + 1
      DO 310 J = JM,N
  310 B(IM,L) = B(IM,L) - A(IM,J)*B(J,L)
  320 B(IM,L) = B(IM,L)/A(IM,IM)
  330 IM = IM - 1
C
  400 CONTINUE
  900 RETURN
      END
```

NBS-114A (REV. 2-8C)

U.S. DEPT. OF COMM. **BIBLIOGRAPHIC DATA SHEET** *(See instructions)*	1. PUBLICATION OR REPORT NO. NISTIR 88-3868	2. Performing Organ. Report No.	3. Publication Date OCTOBER 1988

4. TITLE AND SUBTITLE

A Discrete Thermal Analysis Method (DTAM) for Building Energy Simulation with DTAM1 Users Manual

5. AUTHOR(S)

James Axley

6. PERFORMING ORGANIZATION *(If joint or other than NBS, see instructions)*

NATIONAL BUREAU OF STANDARDS
DEPARTMENT OF COMMERCE
WASHINGTON, D.C. 20234

7. Contract/Grant No.

8. Type of Report & Period Covered

9. SPONSORING ORGANIZATION NAME AND COMPLETE ADDRESS *(Street, City, State, ZIP)*

National Institute of Standards and Technology
Gaithersburg, MD 20899
U.S. Department of Energy
Washington, D.C.

10. SUPPLEMENTARY NOTES

☐ Document describes a computer program; SF-185, FIPS Software Summary, is attached.

11. ABSTRACT *(A 200-word or less factual summary of most significant information. If document includes a significant bibliography or literature survey, mention it here)*

This document includes a report that describes the theoretical basis of the program DTAM1 and a users manual for the program. DTAM1 is a general purpose building energy simulation based upon discrete analysis techniques, including, but not limited to, the Finite Element Method, used in orther fields of physical simulation. It is the product of a first phase of development of Discrete Thermal Element Analysis Techniques for Building Energy Simulation that are expected to provide a means to unify existing building energy simulation theory. DTAM1 provides a library of discrete thermal elements, that may be assembled to model thermal systems idealized to have constant material and heat transfer properties (i.e., linear idealizations), including:

- 1D two-node thermal resistance elements
- single-node lumped capacitance elements
- two-node fluid flow loop element
- 1D two-to-four node isoparametric conduction Finite Elements
- 2D four-node isoparametric conduction Finite Elements (planar and axisymmetric)

Equations defining a variable node mean radiant temperature element are also presented in the report. Steady state and transient analysis capabilities are included. Temperature, heat flow rate, and convective boundary conditions may be modeled and s-stem temperature variables may be constrained to be equal so that mixed assemblages of 1D and 2D elements may be employed.

12. KEY WORDS *(Six to twelve entries; alphabetical order; capitalize only proper names; and separate key words by semicolons)*

building energy simulation, building dynamics, computer simulation techniques, discrete analysis techniques, discrete thermal elements, dynamic simulation, finite element analysis, DTAM1.

13. AVAILABILITY

☒ Unlimited
☐ For Official Distribution. Do Not Release to NTIS
☐ Order From Superintendent of Documents, U.S. Government Printing Office, Washington, D.C. 20402.
☒ Order From National Technical Information Service (NTIS), Springfield, VA. 22161

14. NO. OF PRINTED PAGES

79

15. Price

$13.95