2

WSRC-TR-92-501

FINITE DIFFERENCE PROGRAM FOR CALCULATING HYDRIDE BED WALL TEMPERATURE PROFILES (U)

by J. E. Klein

Westinghouse Savannah River Company
Savannah River Site
Aiken, South Carolina 29808

Other Authors:

This paper was prepared in connection with work done under Contract No. DE-AC09-89SR18035 with the U. S. Department of Energy.

WESTINGHOUSE SAVANNAH RIVER COMPANY
INTER-OFFICE MEMORANDUM

WSRC-TR-92-501

October 29, 1992

To: J.R. Knight, 773-A

From: J.E. Klein, 773-A JK

FINITE DIFFERENCE PROGRAM FOR CALCULATING HYDRIDE BED WALL TEMPERATURE PROFILES (U)

SUMMARY

A QuickBASIC finite difference program was written for calculating one dimensional temperature profiles in up to two media with flat, cylindrical, or spherical geometries. The development of the program was motivated by the need to calculate maximum temperature differences across the walls of the Tritium metal hydrides beds for thermal fatigue analysis.

INTRODUCTION

This purpose of this report is to document the equations and the computer program used to calculate transient wall temperatures in stainless steel hydride vessels. The development of the computer code was motivated by the need to calculate maximum temperature differences across the walls of the hydrides beds in the Tritium Facility for thermal fatigue analysis.

A QuickBASIC finite difference program was written for calculating one dimensional temperature profiles in up to two media with flat, cylindrical, or spherical geometries. The two materials in contact with one another may have concentric geometries (e.g. a cylinder in a cylinder) or contacting geometries (e.g. a cylinder in contact with a sphere). This program was written so analysis on systems other than the hydride beds may be performed.

T. Motyka 11/5/92
Derivative Classifier

DISCUSSION

Heat Conduction Difference Equations

Heat conduction equation for material "i", with constant thermal conductivity, k_i,

$$\rho_i C_{pi} \frac{\partial T_i}{\partial t} = k_i \nabla^2 T = k_i \left[\frac{\partial^2 T_i}{\partial r_i^2} + \frac{\Omega_i}{r_i} \frac{\partial T_i}{\partial r_i} \right] \quad (1)$$

where Ω_i = 0, 1, or 2 for planar, cylindrical, or spherical geometries, respectively.

For finite difference approximations, let $T_{i,j}^n$ be the temperature of material i, at the jth spacial node and nth time step for $j = 0,\ldots,M_i$ and $n = 0,\ldots,N$. The following finite difference approximations were used to approximate the derivatives:

$$\frac{\partial T_i}{\partial t} \approx \frac{T_{i,j}^{n+1} - T_{i,j}^n}{\Delta t} \quad (2)$$

$$\frac{\partial T_i}{\partial r_i} \approx \frac{T_{i,j+1}^n - T_{i,j-1}^n}{2\Delta r_i} \quad (3)$$

$$\frac{\partial^2 T_i}{\partial r_i^2} \approx \frac{T_{i,j-1}^n - 2T_{i,j}^n + T_{i,j+1}^n}{(\Delta r_i)^2} \quad (4)$$

Using the Crank-Nicolson method with these finite difference expressions, the differential equation becomes

$$\begin{aligned} &[-\gamma_i + \delta_i]\, T_{i,j-1}^{n+1} + [2 + 2\gamma_i]\, T_{i,j}^{n+1} + [-\gamma_i - \delta_i]\, T_{i,j+1}^{n+1} \\ &= [\gamma_i - \delta_i]\, T_{i,j-1}^n + [2 - 2\gamma_i]\, T_{i,j}^n + [\gamma_i + \delta_i]\, T_{i,j+1}^n \end{aligned} \quad (5)$$

where

$$\alpha_i = \frac{k_i}{\rho_i C_{pi}}, \quad \gamma = \frac{\alpha_i \Delta t}{(\Delta r_i)^2}, \quad \delta_i = \frac{\alpha_i \Delta t \Omega_i}{2 r_i \Delta r_i} \quad (6)$$

At the internal boundary (j=0), the boundary condition is given by

$$-k_i \frac{\partial T_i}{\partial r_i} + h_{i,in} T_i = f_{i,in}, \quad at\ r_i = r_{i,in} \tag{7}$$

and at the external boundary, ($j=M_i$, $r_{i,ex} > r_{i,in}$), the boundary condition is given by

$$k_i \frac{\partial T_i}{\partial r_i} + h_{i,ex} T_i = f_{i,ex}, \quad at\ r_i = r_{i,ex} \tag{8}$$

where $f_{i,b}$ is the heat transfer function at the boundary of material i. For constant, convective heat loss, $f_{i,b} = h_{i,b}T_{\infty b}$ where $T_{\infty b}$ is the bulk temperature for convective heat transport.

To satisfy the finite difference equation at the boundaries, fictitious temperature values $T_{i,-1}$ and $T_{i,Mi+1}$ where defined. Using the boundary condition at $r_i = r_{i,in}$,

$$T^n_{i,-1} = \frac{-2\Delta r_i h_{i,in}}{k_i} T^n_{i,0} + T^n_{i,1} + \frac{2\Delta r_i}{k_i} f^n_{i,in} \tag{9}$$

Similarly, using the boundary condition at $r_i = r_{i,ex}$,

$$T^n_{i,M+1} = T^n_{i,M-1} + \frac{-2\Delta r_i h_{i,ex}}{k_i} T^n_{i,M_i} + \frac{2\Delta r_i}{k_i} f^n_{i,ex} \tag{10}$$

Matrix System for a Single Material

System of Equations

The finite difference expression and these boundary conditions form the following system of equations:

$$\boldsymbol{A}\boldsymbol{T}^{n+1} = \boldsymbol{B}\boldsymbol{T}^{n} + \boldsymbol{C} = \boldsymbol{D} \tag{11}$$

where

$$
\mathbf{A} = \begin{bmatrix}
2+2\gamma_i\beta_{i,in}-2\delta_i\lambda_{i,in} & -2\gamma_i & 0 & \cdots & 0 & 0 \\
-\gamma_i+\delta_i & 2+2\gamma_i & -\gamma_i-\delta_i & \cdots & 0 & 0 \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
0 & 0 & \cdots & -\gamma_i+\delta_i & 2+2\gamma_i & -\gamma_i-\delta_i \\
0 & 0 & \cdots & 0 & -2\gamma_i & 2+2\gamma_i\beta_{i,ex}+2\delta_i\lambda_{i,ex}
\end{bmatrix} \tag{12}
$$

$$
\mathbf{B} = \begin{bmatrix}
2-2\gamma_i\beta_{i,in}+2\delta_i\lambda_{i,in} & 2\gamma_i & 0 & \cdots & 0 & 0 \\
\gamma_i-\delta_i & 2-2\gamma_i & \gamma_i+\delta_i & \cdots & 0 & 0 \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
0 & 0 & \cdots & \gamma_i-\delta_i & 2-2\gamma_i & \gamma_i+\delta_i \\
0 & 0 & \cdots & 0 & 2\gamma_i & 2-2\gamma_i\beta_{i,ex}-2\delta_i\lambda_{i,ex}
\end{bmatrix} \tag{13}
$$

$$
\mathbf{T}^{n+1} = \begin{bmatrix} T_{i,0}^{n+1} \\ T_{i,1}^{n+1} \\ \cdot \\ \cdot \\ \cdot \\ T_{i,M_i-1}^{n+1} \\ T_{i,M_i}^{n+1} \end{bmatrix}, \quad
\mathbf{T}^{n} = \begin{bmatrix} T_{i,0}^{n} \\ T_{i,1}^{n} \\ \cdot \\ \cdot \\ \cdot \\ T_{i,M_i-1}^{n} \\ T_{i,M_i}^{n} \end{bmatrix}, \quad
\mathbf{C} = \begin{bmatrix} 4(\gamma_i-\delta_i)\mu_{i,in} \\ 0 \\ \cdot \\ \cdot \\ \cdot \\ 0 \\ 4(\gamma_i+\delta_i)\mu_{i,ex} \end{bmatrix} \tag{14}
$$

The system of equations are solved as follows. At time zero (n=0), the initial temperature distribution in the material is known, $\mathbf{T}^n$, (n=0) so vector **D** can be calculated using $\mathbf{D} = \mathbf{B}\mathbf{T}^n + \mathbf{C}$. The temperature at the next time step, $\mathbf{T}^{n+1}$, is obtained by solving the system of equation $\mathbf{A}\mathbf{T}^{n+1} = \mathbf{D}$. These steps are repeated for each time step. The matrix system is similar to that derived by Özişik[1] for a rectangular geometry.

System Modifications for Different Boundary Conditions

Seven different boundary conditions can be applied at each interior and each exterior boundary. These options and how they affect the system of equations to solve are discussed below.

Case 1. Constant Surface Temperature

For a constant internal surface temperature at $r_i = r_{i,in}$, element $a_{0,0}$ is set equal to 1 and $a_{0,1}$ is set equal to 0, $b_{0,0}$ is set equal to 1 and $b_{0,1}$ is set equal to 0, and $c_0 = 0$. Similarly for a constant external surface temperature at $r_i = r_{i,ex}$, element $a_{Mi,M-1i}$ is set equal to 1 and $a_{Mi,Mi}$ is set equal to 0, $b_{Mi,M-1i}$ is set equal to 0 and $b_{Mi,Mi}$ is set equal to 1, and $c_{Mi} = 0$. The surface temperatures are set to their respective values at time = 0 and are constant throughout the computation.

Case 2. Perfectly Insulated Boundary

For a perfectly insulated surface, the terms $h_{i,b}$ and $f_{i,b}$ are set to zero. This gives $\lambda_{i,b} = 0$, $\beta_{i,b} = 1$, and $\mu_{i,b} = 0$ for the appropriate boundary.

Case 3. Constant Boundary Heat Flux

For a constant heat (energy) flux at a boundary, $h_{i,b}$ is set to zero. The term $f_{i,b}$ is set to a positive value for energy going into the material and to a negative value for energy leaving the material.

Case 4. Constant Convective Heat Flux

For a constant convective heat flux at a boundary, $f_{i,b}$ is set equal to $h_{i,b}T_{\infty,b}$. This gives the familiar convective heat loss expression which can be seen by examining the boundary conditions.

Case 5. Variable Surface Temperature

For a surface temperatures as a function of time, the matrix elements of **A** and **B** are modified as was done in the case for a constant surface temperature (case 1). Instead of a surface temperatures being constant, the surface temperature as a function of time is entered into a subroutine of the program and the temperature values updated at each time step.

Case 6. Variable Surface Heat Flux

Parameters for the system are treated as they were for the case of constant heat flux for a surface. In this case, $f_{i,b}$ is a function of time, positive for energy going into the material, negative value for energy leaving the material, and is updated at each time step.

Case 7. Variable Convective Heat Flux

For a variable convective heat flux at a boundary, $f_{i,b}$ is set equal to $h_{i,b}T_{\infty,b}$, where $T_{\infty,b}$ is a function of time. The value of $T_{\infty,b}$ is updated at each time step.

Matrix Systems for Composite (Two) Materials

Equations for Concentric Geometries

For two concentric geometries, such as cylinder 1 inside and in thermal contact with cylinder 2, the boundary condition is

$$-k_1 A_{1,ex}\frac{\partial T_1}{\partial r_1} = -k_2 A_{2,in}\frac{\partial T_2}{\partial r_2} \qquad (15)$$

Since these two cylinders are concentric with one another, $A_{1,ex} = A_{2,in}$. Use a backward-difference operator for the derivative for material 1 and a forward-difference operator for the derivative in material 2, the finite-difference equation is

$$-\epsilon_{1,ex}T_{1,M_1} + \epsilon_{1,ex}T_{1,M_1-1} = -\epsilon_{2,in}T_{2,1} + \epsilon_{2,in}T_{2,0} \qquad (16)$$

Using the assumptions that the points of contact are in thermal equilibrium, $T_{1,M1} = T_{2,0}$, we have

$$+ \epsilon_{1,ex} T_{1,M_1-1} - (\epsilon_{1,ex} + \epsilon_{2,in}) T_{1,M_1} + \epsilon_{2,in} T_{2,1} = 0 \tag{17}$$

Averaging the equation at time-steps n and n+1 for the Crank-Nicolson method yields

$$\begin{aligned} &\epsilon_{1,ex} T^{n+1}_{1,M_1-1} - (\epsilon_{1,ex} + \epsilon_{2,in}) T^{n+1}_{1,M_1} + \epsilon_{2,in} T^{n+1}_{2,1} \\ &= -\epsilon_{1,ex} T^{n}_{1,M_1-1} + (\epsilon_{1,ex} + \epsilon_{2,in}) T^{n}_{1,M_1} - \epsilon_{2,in} T^{n}_{2,1} \end{aligned} \tag{18}$$

The system of equations to solve are similar to those derived for a single material except at the boundary where the two materials come in contact with one another. For the case of concentric materials, the matrices **A, B, C, $\mathbf{T}^n$, and $\mathbf{T}^{n+1}$**, become

$$\mathbf{A} = \begin{bmatrix} 2+2\gamma_1\beta_{1,in}-2\delta_1\lambda_{1,in} & -2\gamma_1 & 0 & \cdots & 0 & 0 \\ -\gamma_1+\delta_1 & 2+2\gamma_1 & -\gamma_1-\delta_1 & \cdots & 0 & 0 \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \epsilon_{1,ex} & -\epsilon_{1,ex}-\epsilon_{2,in} & \epsilon_{2,in} & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ 0 & 0 & \cdots & -\gamma_2+\delta_2 & 2+2\gamma_2 & -\gamma_2-\delta_2 \\ 0 & 0 & \cdots & 0 & -2\gamma_2 & 2+2\gamma_2\beta_{2,ex}+2\delta_2\lambda_{2,ex} \end{bmatrix}$$

(19)

$$\mathbf{B} = \begin{bmatrix} 2-2\gamma_1\beta_{1,in}+2\delta_1\lambda_{1,in} & 2\gamma_1 & 0 & \cdots & 0 & 0 \\ \gamma_1-\delta_1 & 2-2\gamma_1 & \gamma_1+\delta_1 & \cdots & 0 & 0 \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & -\epsilon_{1,ex} & \epsilon_{1,ex}+\epsilon_{2,in} & -\epsilon_{2,in} & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ 0 & 0 & \cdots & \gamma_2-\delta_2 & 2-2\gamma_2 & \gamma_2+\delta_2 \\ 0 & 0 & \cdots & 0 & 2\gamma_2 & 2-2\gamma_2\beta_{2,ex}-2\delta_2\lambda_{2,ex} \end{bmatrix} \tag{20}$$

$$\boldsymbol{T}^{n+1} = \begin{bmatrix} T^{n+1}_{1,0} \\ T^{n+1}_{1,1} \\ \cdot \\ T^{n+1}_{1,M_1-1} \\ T^{n+1}_{1,M_1} \\ T^{n+1}_{2,0} \\ \cdot \\ T^{n+1}_{2,M_2-1} \\ T^{n+1}_{2,M_2} \end{bmatrix}, \quad \boldsymbol{T}^{n} = \begin{bmatrix} T^{n}_{1,0} \\ T^{n}_{1,1} \\ \cdot \\ T^{n}_{1,M_1-1} \\ T^{n}_{1,M_1} \\ T^{n}_{2,0} \\ \cdot \\ T^{n}_{2,M_2-1} \\ T^{n}_{2,M_2} \end{bmatrix}, \quad \boldsymbol{C} = \begin{bmatrix} 4(\gamma_1-\delta_1)\mu_{1,in} \\ 0 \\ \cdot \\ 0 \\ 0 \\ 0 \\ \cdot \\ 0 \\ 4(\gamma_2+\delta_2)\mu_{2,ex} \end{bmatrix} \tag{21}$$

Contacting Geometries

For two geometries touching one another, such as part of the external surface of cylinder 1 in thermal contact with the external surface of cylinder 2, the boundary condition is

$$-k_1 A_{1,ex}\frac{\partial T_1}{\partial r_1} = +k_2 A_{2,ex}\frac{\partial T_2}{\partial r_2} \tag{22}$$

where it is implied that the areas represent the thermal contact areas for the materials. Since these two cylinders are not concentric, in general we have $A_{1,ex} \neq A_{2,ex}$. Use a backward-difference operator for the derivative for material 1 and material 2, the finite-difference equation is

$$-\epsilon_{1,ex}T_{1,M_1} + \epsilon_{1,ex}T_{1,M_1-1} = +\ \epsilon_{2,ex}T_{2,M_2} - \epsilon_{2,ex}T_{2,M_2-1} \tag{23}$$

Using the assumptions that the points of contact are in thermal equilibrium, $T_{1,M1} = T_{2,M2}$, we have

$$+\ \epsilon_{1,ex}T_{1,M_1-1} - (\epsilon_{1,ex} + \epsilon_{2,ex})\,T_{1,M_1} + \epsilon_{2,ex}T_{2,M_2-1} = 0 \tag{24}$$

Again, averaging the equation at time-steps n and n+1 for the Crank-Nicolson method yields

$$\begin{aligned} &\epsilon_{1,ex}T^{n+1}_{1,M_1-1} - (\epsilon_{1,ex} + \epsilon_{2,ex})\,T^{n+1}_{1,M_1} + \epsilon_{2,ex}T^{n+1}_{2,M_2-1} \\ &= -\epsilon_{1,ex}T^{n}_{1,M_1-1} + (\epsilon_{1,ex} + \epsilon_{2,ex})\,T^{n}_{1,M_1} - \epsilon_{2,ex}T^{n}_{2,M_2-1} \end{aligned} \tag{25}$$

The system of equations to solve are similar to those derived for contacting geometries: the major difference is the changing of the subscripts at the boundaries of material 2. For this case, the matrices **A, B, C, T^n, and T^{n+1}**, become

$$\mathbf{A}=\begin{bmatrix} 2+2\gamma_1\beta_{1,in}-2\delta_1\lambda_{1,in} & -2\gamma_1 & 0 & \cdots & 0 & 0 \\ -\gamma_1+\delta_1 & 2+2\gamma_1 & -\gamma_1-\delta_1 & \cdots & 0 & 0 \\ . & . & . & . & . & . \\ . & . & \epsilon_{1,ex} & -\epsilon_{1,ex}-\epsilon_{2,ex} & \epsilon_{2,ex} & . \\ . & . & . & . & . & . \\ 0 & 0 & \cdots & -\gamma_2+\delta_2 & 2+2\gamma_2 & -\gamma_2-\delta_2 \\ 0 & 0 & \cdots & 0 & -2\gamma_2 & 2+2\gamma_2\beta_{2,in}+2\delta_2\lambda_{2,in} \end{bmatrix} \quad (26)$$

$$\mathbf{B}=\begin{bmatrix} 2-2\gamma_1\beta_{1,in}+2\delta_1\lambda_{1,in} & 2\gamma_1 & 0 & \cdots & 0 & 0 \\ \gamma_1-\delta_1 & 2-2\gamma_1 & \gamma_1+\delta_1 & \cdots & 0 & 0 \\ . & . & . & . & . & . \\ . & . & -\epsilon_{1,ex} & \epsilon_{1,ex}+\epsilon_{2,ex} & -\epsilon_{2,ex} & . \\ . & . & . & . & . & . \\ 0 & 0 & \cdots & \gamma_2-\delta_2 & 2-2\gamma_2 & \gamma_2+\delta_2 \\ 0 & 0 & \cdots & 0 & 2\gamma_2 & 2-2\gamma_2\beta_{2,in}-2\delta_2\lambda_{2,in} \end{bmatrix} \quad (27)$$

$$\mathbf{T}^{n+1}=\begin{bmatrix} T_{1,0}^{n+1} \\ T_{1,1}^{n+1} \\ . \\ T_{1,M_1-1}^{n+1} \\ T_{1,M_1}^{n+1} \\ T_{2,M_2-1}^{n+1} \\ . \\ T_{2,1}^{n+1} \\ T_{2,0}^{n+1} \end{bmatrix}, \quad \mathbf{T}^{n}=\begin{bmatrix} T_{1,0}^{n} \\ T_{1,1}^{n} \\ . \\ T_{1,M_1-1}^{n} \\ T_{1,M_1}^{n} \\ T_{2,M_2-1}^{n} \\ . \\ T_{2,1}^{n} \\ T_{2,0}^{n} \end{bmatrix}, \quad \mathbf{C}=\begin{bmatrix} 4(\gamma_1-\delta_1)\mu_{1,in} \\ 0 \\ . \\ 0 \\ 0 \\ 0 \\ . \\ 0 \\ 4(\gamma_2+\delta_2)\mu_{2,in} \end{bmatrix} \quad (28)$$

Computer Program Testing

Steady-State Testing

To the test program's ability to calculate a steady-state temperature profile for a material, the initial temperature of a material was set to 150°C. Next, boundary conditions were applied such that the boundaries temperature values should be constant at 150°C. The program was started and run to see how accurately the program would maintain this initial temperature.

For example, a cylinder had its internal surface temperature set to 150°C and had constant, convective heat loss at its external boundary to a bulk temperature of 150°C. The program was run knowing that the temperature at all grid points should start at 150°C and maintain this temperature as the program marches through time steps.

The program was run with all 7×7 combinations of boundary conditions to see how accurately the program would maintain this initial temperature profile of 150°C. Testing the program in this manner gave temperature errors of less than 0.0015°C and gave confidence that the algorithm would give correct steady-state temperature profiles.

Transient Testing

The program was tested on the conduction equation in rectangular coordinates for a single material with the following boundary conditions:

$$\frac{\partial T_i}{\partial r_i} = 0, \quad at\ r_{i,in} = 0, \qquad \frac{\partial T_i}{\partial r_i} + 7\,T_i = 0, \quad at\ r_{i,ex} = 1\ cm \tag{29}$$

At time zero, the material was at 150°C, and the solution compared to the analytical solution for the problem. After 20 time steps, temperature errors between the analytical and numerical solution of up to 0.04 °C were obtained, but decreased as additional time steps were taken. The accuracy of the numerical technique is considered excellent. Transient testing along with the steady-state testing satisfactorily demonstrates the ability of the program to calculate transient temperature profiles.

CONCLUSIONS

Equations were derived and a computer program was written to solve simultaneous finite difference heat conduction equations for the transient analysis of one-dimensional heat conduction using the Crank-Nicolson algorithm. The program written is an interactive, versatile program which can analyze planar, cylindrical, or spherical geometries. A total of 7x7 combinations of boundary conditions can be chosen for the thermal analysis of a single material. A total of 2x7x7 different analyses can be performed on a two material system: the materials may be concentric with one another or have their external surfaces in thermal contact with one another.

The program has been tested on its ability to calculate steady-state and transient temperatures. For the steady-state testing performed, errors in calculating steady-state temperatures were less than 0.001% of the known value. Comparison of the result from the numerical method versus the results obtained from the analytical solution show errors well below the accuracy required for thermal fatigue analysis.

REFERENCES

1. M. Necati Özişik. *Heat Conduction*. p. 502. John Wiley & Sons, Inc., New York (1980).

NOTATION

Subscripts

i = material index, = 1 or 2

b = boundary index, = <u>in</u> for interior boundary, <u>ex</u> for exterior boundary

Symbols

$a_{p,q}$ = element of matrix **A**, p = 0,...,M and q = 0,...,M

A = matrix of coefficients, defined in text

$A_{i,b}$ = heat transfer area for material i at the boundary, m^2

$b_{p,q}$ = element of matrix **B**, p = 0,...,M and q = 0,...,M

B = matrix of coefficients, defined in text

c_p = element of vector **C**, p = 0,...,M

C = vector, defined in text

C_{pi} = heat capacity for material i, cal/g-°C

D = vector, = $\mathbf{BT}^n + \mathbf{C}$

$f_{i,b}$ = thermal flux for material i at boundary b, watts/m^2

$h_{i,b}$ = heat transfer coefficient for material i at boundary b, watts/m^2-°C

j = spacial parameter index, 0,...,M_i

k_i = thermal conductivity for material i, watts/m-°C

M_i = number of spacial grid points for material i

n = time step index, 0,...,N

N = number of time steps

r_i = spacial distance for material i, cm

$r_{i,b}$ = radius of material i at boundary b (I.R. and O.R. for the material), cm

t = time, seconds

$T_{i,j}^n$ = temperature of material i, at spacial node j, at time step n

$T_{\infty,b}$ = bulk media convective heat transfer temperature at boundary b, °C

Greek

α_i = thermal diffusivity for material i, = k_i/ρ_i-C_{pi}, cm^2/sec

$\beta_{i,b}$ = $1 + \lambda_{i,b}$

γ_i = dimensionless time for material i, = $\alpha_i \Delta t/(\Delta r_i)^2$

δ_i = $\alpha_i \Delta t \Omega_i / (2 r_i \Delta r_i)$

Δr_i = radial coordinate spacing for material i, cm

Δt = time step size, sec

$\epsilon_{i,b}$ = $k_i A_{i,b} / \Delta r_i$

$\lambda_{i,b}$ = $\Delta r_i h_{i,b} / k_i$

$\mu_{i,b}$ = $\Delta r_i / k_i (f_{i,b}^{n+1} + f_{i,b}^{n}) / 2$

ρ_i = density for material i, g/cm^3

Ω_i = geometric parameter for material i: = 0 for planar, 1 for cylindrical, or 2 for spherical geometry

APPENDIX A. COMPUTER PROGRAM LISTING

```
DECLARE SUB BASICINPUT (APPL!, GEOM!(), CONFIG!, ICOND%(),
BCIN!(), BCOUT!(), MTLTYPE%()) DECLARE SUB BCMENU (I%, J%, BC!)
DECLARE SUB BCONDITION (M%(), TIN!(), TOUT!(), HIN!(), HOUT!(),
FIN!(), FOUT!(), TOLD!()) DECLARE SUB BCUPDATE (I%, M%(), T!,
TOLD!())
DECLARE SUB DATAFROM (IZINPUT%, ZINPUT$)
DECLARE SUB DATATO (IZOUTPUT%, ZOUTPUT$)
DECLARE FUNCTION FFIN! (I%, TIME!)
DECLARE FUNCTION FFOUT! (I%, TIME!)
DECLARE FUNCTION FICOND! (I%, X!)
DECLARE FUNCTION FSAMPLE1! (T!, X!)
DECLARE FUNCTION FSAMPLE2! (T!, X!)
DECLARE FUNCTION FTIN! (T!)
DECLARE FUNCTION FTOUT! (T!)
DECLARE SUB ICONDITION (APPL!, ICOND%(), M%(), MTOT%, R!(),
TOLD!()) DECLARE SUB MATRIXA (MTOT%, AA!(), AB!(), AC!())
DECLARE SUB MATRIXB (MTOT%, BA!(), BB!(), BC!())
DECLARE SUB MATTERMS (GAMMA!(), EPS!(), DELTA!(), LAMIN!(),
LAMOUT!(), BETAIN!(), BETAOUT!()) DECLARE SUB MTLPROP (APPL!,
MTLTYPE%(), K!(), ALPHA!())
DECLARE SUB PAUSE ()
DECLARE SUB SIZE (APPL!, GEOM!(), CONFIG!, THICK!(), RIN!(),
ROUT!(), M%(), DR!(), R!()) DECLARE SUB SOLVE (M%(), MTOT%,
TNEW!(), AA!(), AB!(), AC!(), D!()) DECLARE SUB TIMEPARMS (DR!(),
ALPHA!(), DT!, N!, TMAX!)
DECLARE SUB VECTORC (MTOT%, TIME!, C!())
DECLARE SUB VECTORD (M%(), TOLD!(), BA!(), BB!(), BC!(), C!(),
D!())

'TCOND.BAS..................7/92
'FINITE-DIFFERENCE HEAT CONDUCTION PROGRAM DEVELOPED AT:
'SAVANNAH RIVER TECHNOLOGY CENTER, AIKEN SC 29808 (BY JE KLEIN)


DEFINT I, M

M1 = 20       'NUMBER OF SPACIAL GRID POINTS FOR MATERIAL 1
M2 = 5       'NUMBER OF SPACIAL GRID POINTS FOR MATERIAL 2

DEBUG = 0

MMAX = M1 + M2

DIM M(2)
DIM GEOM(2), ICOND(2), BCIN(2), BCOUT(2), MTLTYPE(2)
DIM RIN(2), ROUT(2), THICK(2), DR(2), R(2, MMAX)
DIM TOLD(MMAX), TNEW(MMAX), TZERO(2)
DIM TIN(2), TOUT(2), HIN(2), HOUT(2), FIN(2), FOUT(2)
```

```
DIM K(2), ALPHA(2), GAMMA(2), DELTA(MMAX), LAMIN(2), LAMOUT(2)
DIM BETAIN(2), BETAOUT(2), AREA(2), EPS(2)
DIM AA(MMAX), AB(MMAX), AC(MMAX)
DIM BA(MMAX), BB(MMAX), BC(MMAX)
DIM C(MMAX), D(MMAX)

'****************************************************************
*********** 'CORE PROGRAM
CONST PI = 3.141592654#

M(0) = 0
M(1) = M1
M(2) = M2

CLS
CALL DATAFROM(IZINPUT, ZINPUT$)
'
'BASIC INPUT TO PROGRAM
CALL BASICINPUT(APPL, GEOM(), CONFIG, ICOND(), BCIN(), BCOUT(),
MTLTYPE()) '
'SET MAXIMUM SIZE OF MATRIX SYSTEM
IF (APPL = 1) THEN
   MTOT = M(1)
ELSE
   MTOT = M(1) + M(2)
END IF
'
'PHYSICAL SIZE OF MATERIAL(S)
CALL SIZE(APPL, GEOM(), CONFIG, THICK(), RIN(), ROUT(), M(),
DR(), R()) '
'PHYSICAL PROPERTIES OF MATERIAL(S)
CALL MTLPROP(APPL, MTLTYPE(), K(), ALPHA())
'
'SET INITIAL TEMPERATURE OF MATERIAL(S)
CALL ICONDITION(APPL, ICOND(), M(), MTOT, R(), TOLD())

'SET BOUNDARY CONDITIONS
CALL BCONDITION(M(), TIN(), TOUT(), HIN(), HOUT(), FIN(), FOUT(),
TOLD())

'SET TIME-STEP FOR PROBLEM
CALL TIMEPARMS(DR(), ALPHA(), DT, N, TMAX)

'CALCULATE CONSTANT TERMS IN MATRICIES
CALL MATTERMS(GAMMA(), EPS(), DELTA(), LAMIN(), LAMOUT(),
BETAIN(), BETAOUT())

'CALCULATE MAXTRIX CORRESPONDING WITH VECTOR[TOLD]
CALL MATRIXB(MTOT, BA(), BB(), BC())

'CALCULATE MAXTRIX CORRESPONDING WITH VECTOR[TOLD]
CALL MATRIXA(MTOT, AA(), AB(), AC())
```

```
CALL DATATO(IZOUTPUT, ZOUTPUT$)

'CALCULATE NUMBER OF TIME STEPS
N = INT(N) + 1
IF N < 20 THEN N = 20
DELTAMAX = 0!

IF (M(1) > 10) THEN PSTEP1 = INT(M(1) / 10)

FOR TSTEP = 1 TO N - 1
   TIME = TSTEP * DT
   IF (TSTEP > 0) THEN TIMEOLD = (TSTEP - 1) * DT

   CALL VECTORC(MTOT, TIME, C())

   'UPDATE B.C. IF TEMP. IS FUNCTION OF TIME
     I = 1                    'FOR MATERIAL 1 INSIDE B.C.
     IF (BCIN(I) = 5) THEN
       CALL BCUPDATE(I, M(), TIMEOLD, TOLD())
     END IF
     I = APPL                 'FOR MATERIAL 'APPL' OUTSIDE B.C.
IF (BCOUT(I) = 5) THEN
       CALL BCUPDATE(I, M(), TIMEOLD, TOLD())
     END IF

   IF (DEBUG = 2) THEN
     PRINT "TOLD"
     FOR J = 0 TO M(1)
       PRINT TOLD(J)
     NEXT J
     CALL PAUSE
     IF (APPL = 2) THEN
        FOR J = M(1) TO M(2)
          PRINT TOLD(J)
        NEXT J
     END IF

   END IF

   'CALCULATE VECTOR D: VECTOR[D] = MATRIX[B] * VECTOR[TOLD] +
VECTOR[C]    CALL VECTORD(M(), TOLD(), BA(), BB(), BC(), C(),
D())

   'CALCULATE TEMPERATURES AT NEW TIME: SOLVE FOR VECTOR[TNEW]
'MATRIX[A]* VECTOR(TNEW) = VECTOR[D] = MATRIX[B] * VECTOR[TOLD] +
VECTOR[C]    CALL SOLVE(M(), MTOT, TNEW(), AA(), AB(), AC(), D())

   CLS
   DELTAT = TNEW(M(1)) - TNEW(0)
   IF (ABS(DELTAT) > ABS(DELTAMAX)) THEN
      DELTAMAX = DELTAT
      TIMEDTMAX = TIME
```

```
   END IF
PRINT "TIME = "; TIME; " DELTAT (C) = "; DELTAT; "DTmax = ";
DELTAMAX; "AT T = "; TIMEDTMAX '   FOR J = 0 TO M(1) STEP PSTEP1
   FOR J = 0 TO M(1) STEP 2
     SELECT CASE ISAMPLE
       CASE 1
         TANAL = FSAMPLE1(TIME, R(1, J))
       CASE 2
         TANAL = FSAMPLE2(TIME, R(1, J))
     END SELECT
     IF (ISAMPLE <> 0) THEN
       PRINT J, R(1, J), TNEW(J), TANAL
     ELSE
       PRINT J, R(1, J), TNEW(J)
     END IF
   NEXT J
   IF (APPL = 2) THEN
      PRINT ""
      FOR J = M(1) TO MTOT STEP 2
          PRINT J, R(2, J - M(1)), TNEW(J)
      NEXT J
   END IF
'CALL PAUSE
IF (DEBUG = 2) THEN CALL PAUSE
'PRINT TSTEP; TIME; TNEW(1, 0); TNEW(1, M/4); TNEW(1, M/2);
TNEW(1, M*3/4); TNEW(1, M)

'SET TEMPERATURE VALUES FOR NEXT TIME STEP
    FOR J = 0 TO MTOT
      TOLD(J) = TNEW(J)
    NEXT J

NEXT TSTEP


END
'*****************************************************************
*********
'*****************************************************************
*********


'OUTPUT OF
DATA...........................................................
PRINT SPC(7); "Program TCOND.BAS..................."; DATE$;
CHR$(13) PRINT SPC(7); "Identification:"; DENT$
PRINT SPC(7); "Material:":

SELECT CASE GEOM(1)
  CASE 0
    PRINT SPC(7); "Plane Sheet"
    PRINT SPC(10); "Wall thickness (in.)="; THICK(1)
```

```
  CASE 1
    PRINT SPC(7); "Cylinder"
    PRINT SPC(10); "Inner radius (in.)="; RIN(1), "Outer radius
(in.)="; ROUT(1)   CASE 2
    PRINT SPC(7); "Sphere"
    PRINT SPC(10); "Inner radius (in.)="; RIN(1), "Outer radius
(in.)="; ROUT(1) END SELECT
'*****************************************************************
********* '                              START INPUT DATA HERE
'*****************************************************************
*********

DEFINT J
SUB BASICINPUT (APPL, GEQM(), CONFIG, ICOND(), BCIN(), BCOUT(),
MTLTYPE()) SHARED ZINPUT$, DENT$, ISAMPLE
'FUNDEMENTAL INPUT FOR PROGRAM
CLS

SELECT CASE ZINPUT$

        CASE IS = "KYBD:"

ISAMPE = 0
INPUT "COMPARE TO ANALYTICAL SOLUTION? (Y/N): ", A$
IF (A$ = "y" OR A$ = "Y") THEN
   INPUT "ENTER SAMPLE PROBLEM NUMBER ", ISAMPLE
END IF
CLS

PRINT "PROGRAM TCOND.BAS FOR DIFFUSION CALCULATIONS  "; DATE$;
CHR$(13) SELECT CASE ISAMPLE
   CASE 0
INPUT "Identification name and/or number for analysis"; DENT$:
PRINT "" PRINT "            APPLICATION DESIRED"
PRINT "1. Single Material Conduction (Default)"
PRINT "2. Coupled Material Conduction"
'PRINT "3. ": PRINT ""
INPUT "OPTION"; APPL: PRINT ""
IF (APPL = 0) THEN APPL = 1
   CASE 1
APPL = 1
   CASE 2
APPL = 1
END SELECT

CONFIG = 0
IF (APPL = 2) THEN
   PRINT "            COUPLED CONDITION APPLICATION"
   PRINT "1. Concentric Layers (Default)"
   PRINT "2. Contacting Geometrys "
   'PRINT "3. ": PRINT ""
   INPUT "OPTION"; CONFIG: PRINT ""
```

```
   IF (CONFIG = 0) THEN CONFIG = 1
END IF

CLS
SELECT CASE ISAMPLE
   CASE 0
FOR I = 1 TO APPL
   PRINT "GEOMETRY FOR MATERIAL "; I: PRINT ""
   PRINT "1. Plane sheet"
   PRINT "2. Cylinder (Default)"
   PRINT "3. Sphere"
   INPUT "OPTION"; GEOM(I): PRINT "": PRINT ""
   IF (GEOM(I) = 0) THEN GEOM(I) = 2

   IF (CONFIG = 1) THEN
      GEOM(2) = GEOM(1)
      I = I + 1
   END IF
NEXT I
CLS
   CASE 1
GEOM(1) = 1
   CASE 2
GEOM(1) = 1
END SELECT


'SET PARAMETERS DESCRIBING INITIAL CONDITIONS
SELECT CASE ISAMPLE
   CASE 0
FOR I = 1 TO APPL
   PRINT "INITIAL CONDITION FOR MATERIAL "; I: PRINT ""
   PRINT "1. Constant Temperature Profile (Default)"
   PRINT "2. Variable Temperature Profile (User Entered)"
   PRINT "3. Variable Temperature Profile (Function Generated)"
INPUT "OPTION"; ICOND(I): PRINT "": PRINT ""
   IF (ICOND(I) = 0) THEN ICOND(I) = 1
NEXT I
CLS
   CASE 1
ICOND(1) = 3
   CASE 2
ICOND(1) = 1
END SELECT


'SET PARAMETERS DESCRIBING BOUNDARY CONDITIONS
SELECT CASE ISAMPLE
   CASE 0
FOR I = 1 TO APPL
   IF (CONFIG = 0) THEN
     CALL BCMENU(1, 1, BCIN(1))
```

```
        PRINT "": PRINT ""
        CALL BCMENU(1, 2, BCOUT(1))
      ELSE
         IF (I = 1) THEN
            J = 1
            CALL BCMENU(I, J, BCIN(I))
            PRINT ""
            PRINT "Material 1 in Conductive Contact with Material 2"
           PRINT ""
            BCOUT(I) = 8
         ELSE
            J = 2
            BCIN(I) = 8
            CALL BCMENU(I, J, BCOUT(I))
         END IF
      END IF
   NEXT I

   CLS

      CASE 1
   BCIN(1) = 1
   BCOUT(1) = 1
      CASE 2
   BCIN(1) = 2
   BCOUT(1) = 4
   END SELECT


   'SET PARAMETERS DESCRIBING MATERIAL PROPERTIES
   SELECT CASE ISAMPLE
      CASE 0
   FOR I = 1 TO APPL
      PRINT "MATERIAL "; I; " OPTIONS"
      PRINT "1. 304L (default)"
      PRINT "2. 21-6-9"
      PRINT "3. Copper alloys"
      PRINT "4. Aluminum alloys"
      PRINT "5. UNITY VALUES (For testing program)"
      PRINT "6. Other"
      INPUT "OPTION"; MTLTYPE(I)
      IF (MTLTYPE(I) = 0) THEN MTLTYPE(I) = 1
      PRINT ""
   NEXT I
   CLS
      CASE 1
   MTLTYPE(1) = 5
      CASE 2
   MTLTYPE(1) = 5
      END SELECT

           CASE IS = "PRGM"
```

```
READ DENT$

        CASE ELSE
INPUT #1, DENT$

END SELECT

END SUB

SUB BCMENU (I, J, BC)
'MENU FOR INPUT OF BOUNDARY CONDITIONS

IF (J = 1) THEN
   'DEFAULT MENU FOR INSIDE B.C. (EDIT AS NEEDED)
   PRINT SPC(10); "INSIDE BOUNDARY CONDITION FOR MATERIAL "; I
PRINT " "                                    'VARIABLE/FUNCTION
TO USE    IF (BC <> 8) THEN
      PRINT "1. Constant Temperature (Default)"    'TIN
      PRINT "2. Insulated Surface"                 '
      PRINT "3. Constant Energy Flux"              'QIN
      PRINT "4. Constant Convective Energy Flux"   'HIN,TIN
PRINT "5. Variable Surface Temperature"      'FTIN
      PRINT "6. Variable Surface Energy Flux"      'FFIN
      PRINT "7. Variable Convective Flux"          'HIN,FTIN
INPUT "OPTION"; BC
      IF (BC = 0) THEN BC = 1
    ELSE
      PRINT "Internal Boundary Condition for Material"; I; ":
Conductive"    END IF
ELSE
   'DEFAULT MENU FOR OUTSIDE B.C. (EDIT AS NEEDED)
   PRINT SPC(10); "OUTSIDE BOUNDARY CONDITION FOR MATERIAL "; I
PRINT " "                                    'VARIABLE/FUNCTION
TO USE    IF (BC <> 8) THEN
      PRINT "1. Constant Temperature (Default)"    'TOUT
      PRINT "2. Insulated Surface"                 '
      PRINT "3. Constant Energy Flux"              'QOUT
      PRINT "4. Constant Convective Energy Flux"   'HOUT,TOUT
 PRINT "5. Variable Surface Temperature"      'FTOUT
      PRINT "6. Variable Surface Energy Flux"      'FFOUT
      PRINT "7. Variable Convective Flux"          'HOUT,FTOUT
  INPUT "OPTION"; BC
      IF (BC = 0) THEN BC = 1
    ELSE
      PRINT "External Boundary Condition for Material"; I; ":
Conductive"    END IF
END IF

END SUB

SUB BCONDITION (M(), TIN(), TOUT(), HIN(), HOUT(), FIN(), FOUT(),
TOLD()) SHARED APPL, BCIN(), BCOUT(), ISAMPLE, CONFIG, AREA()
```

```
SELECT CASE ISAMPLE
   CASE 0
CLS
FOR I = 1 TO APPL
   PRINT SPC(10); "BOUNDARY CONDITIONS FOR MATERIAL"; I
   PRINT ""
   FOR J = 1 TO 2
      IF (J = 1) THEN
         PRINT "INTERNAL BOUNDARY CONDITION"
         TEST = BCIN(I)
      ELSE
         PRINT "EXTERNAL BOUNDARY CONDITION"
         TEST = BCOUT(I)
      END IF
         SELECT CASE TEST
      CASE IS = 1 ' CONSTANT TEMPERATURE
      IF (J = 1) THEN
         INPUT "Enter Constant INTERNAL Temperature (C)"; TIN(I)
       TOLD(0) = TIN(I)
      ELSE
         INPUT "Enter Constant EXTERNAL Temperature (C)"; TOUT(I)
        TOLD(M(I)) = TOUT(I)
      END IF

      CASE IS = 2 ' INSULATED BOUNDARY
      IF (J = 1) THEN
         PRINT "Internal boundary insulated"
         HIN(I) = 0!
         FIN(I) = 0!
      ELSE
         PRINT "External boundary insulated"
         HOUT(I) = 0!
         FOUT(I) = 0!
      END IF

      CASE IS = 3 ' CONSTANT HEAT FLUX
      IF (J = 1) THEN
         HIN(I) = 0!
         PRINT "ENTER CONSTANT INTERNAL ENERGY FLUX (WATTS/CM^2)"
        INPUT "(+ FOR HEAT INTO MATERIAL, - FOR HEAT OUT OF
MATERIAL) "; FIN(I)       ELSE
         HOUT(I) = 0!
         PRINT "ENTER CONSTANT EXTERNAL ENERGY FLUX (WATTS/CM^2)"
        INPUT "(+ FOR HEAT INTO MATERIAL, - FOR HEAT OUT OF
MATERIAL) "; FOUT(I)       END IF

      CASE IS = 4 ' CONVECTIVE HEAT FLUX
      IF (J = 1) THEN
         INPUT "ENTER CONVECTIVE HEAT TRANSFER COEFFICIENT
(WATTS/M^2-C) "; HIN(I)          'CONVERT FROM (WATTS/M^2-C) TO
(WATTS/CM^2-C)
         HIN(I) = HIN(I) / 10000!
```

```
          INPUT "ENTER CONSTANT CONVECTIVE 'BULK' TEMPERATURE
(C)"; TIN(I)            'FIN(I) = HIN(I) * TIN(I)     '(WATTS/CM^2)
       ELSE
          INPUT "ENTER CONVECTIVE HEAT TRANSFER COEFFICIENT
(WATTS/M^2-C) "; HOUT(I)            'CONVERT FROM (WATTS/M^2-C) TO
(WATTS/CM^2-C)
          HOUT(I) = HOUT(I) / 10000!
          INPUT "ENTER CONSTANT CONVECTIVE 'BULK' TEMPERATURE
(C)"; TOUT(I)           'FOUT(I) = HOUT(I) * TOUT(I)
'(WATTS/CM^2)
       END IF

       CASE IS = 5
       IF (J = 1) THEN
          PRINT "Internal surface temperature as a function of
time must be"          PRINT "entered into subroutine FTIN.
VERIFY BEFORE CONTINUING"          CALL PAUSE
          TOLD(0) = FTIN(0!)
       ELSE
          PRINT "External surface temperature as a function of
time must be"          PRINT "entered into subroutine FTOUT.
VERIFY BEFORE CONTINUING"          CALL PAUSE
          TOLD(M(I)) = FTOUT(0!)
       END IF

       CASE IS = 6
       IF (J = 1) THEN
          PRINT "Internal surface ENERGY flux as a function of
time must be"          PRINT "entered into subroutine FFIN.
VERIFY BEFORE CONTINUING"          CALL PAUSE
       ELSE
          PRINT "External surface ENERGY flux as a function of
time must be"          PRINT "entered into subroutine FFOUT.
VERIFY BEFORE CONTINUING"          CALL PAUSE
       END IF

       CASE IS = 7
       IF (J = 1) THEN
          INPUT "ENTER CONVECTIVE HEAT TRANSFER COEFFICIENT
(WATTS/M^2-C) "; HIN(I)            'CONVERT FROM (WATTS/M^2-C) TO
(WATTS/CM^2-C)
          HIN(I) = HIN(I) / 10000!
          PRINT "Internal convective temperature as a function of
time must be"          PRINT "entered into subroutine FTIN.
VERIFY BEFORE CONTINUING"          CALL PAUSE
       ELSE
          INPUT "ENTER CONVECTIVE HEAT TRANSFER COEFFICIENT
(WATTS/M^2-C) "; HOUT(I)            'CONVERT FROM (WATTS/M^2-C) TO
(WATTS/CM^2-C)
          HOUT(I) = HOUT(I) / 10000!
          PRINT "Internal surface temperature as a function of
time must be"          PRINT "entered into subroutine FTOUT.
```

```
VERIFY BEFORE CONTINUING"          CALL PAUSE
      END IF

      CASE IS = 8
         PRINT "Coupled Problem Chosen - B.C. fixed"
         IF (I = 1) THEN
            AREA(1) = 1!
            IF (CONFIG = 2) THEN
               INPUT "Enter EXTERNAL conductive area for material
1 (CM^2/CM)"; AREA(1)              END IF
         ELSE
            AREA(2) = 1!
            IF (CONFIG = 2) THEN
               INPUT "Enter EXTERNAL conductive area for material
2 (CM^2/CM)"; AREA(2)              END IF
         END IF
         PRINT ""
      END SELECT
      PRINT ""
  NEXT J
NEXT I

   CASE 1
         TOLD(0) = 0!
         TOLD(M(1)) = 0!

   CASE 2
         HIN(I) = 0!
         FIN(I) = 0!
         HOUT(1) = 70000!
         HOUT(1) = HOUT(1) / 10000!
         TOUT(1) = 0
         FOUT(1) = HOUT(1) * TOUT(1)  '(WATTS/CM^2)

END SELECT
END SUB

DEFSNG J
SUB BCUPDATE (I, M(), TIME, TEMP())
SHARED BCIN(), BCOUT()
'UPDATE B.C. TEMP. WHEN TEMP. IS A FUNCTION OF TIME

IF (BCIN(1) = 5) THEN
   TEMP(0) = FTIN(TIME)
END IF


IF (BCOUT(I) = 5) THEN
   IF (I = 1) THEN
      TEMP(M(I)) = FTOUT(TIME)
   ELSE
      TEMP(M(I)) = FTOUT(TIME)
```

```
   END IF
END IF

END SUB

DEFINT J
SUB DATAFROM (IZINPUT, ZINPUT$)
'
'DETERMINE IF DATA IS FROM KEYBOARD, PROGRAM DATA STATEMENTS, OR
DATA FILE '
    PRINT "PROGRAM INPUT DATA FROM"
    PRINT " 1 - KEYBOARD (** DEFAULT **) "
    PRINT " 2 - PROGRAM DATA STATEMENTS  "
    PRINT " 3 - DATA FILE                "
    INPUT "ENTER CHOICE (1-3) :", IZINPUT
IF IZINPUT = 0 THEN IZINPUT = 1

SELECT CASE IZINPUT
        CASE 1
                ZINPUT$ = "KYBD:"
        CASE 2
                ZINPUT$ = "PRGM"
        CASE 3
                PRINT " "
                INPUT "ENTER INPUT DATA FILE NAME: ", ZINPUT$
           OPEN ZINPUT$ FOR INPUT AS #1
END SELECT
    PRINT " "
    PRINT " "

END SUB

DEFSTR Z
SUB DATATO (IZOUTPUT, ZOUTPUT$)
'
'DETERMINE IF OUTPUT IS TO SCREEN, PRINTER, OR DATAFILE
'
'
    PRINT "PROGRAM OUTPUT TO:           "
    PRINT " 0 - SCREEN (** DEFAULT **)  "
    PRINT " 1 - LPT1 (DEFAULT PRINTER)  "
    PRINT " 2 - LPT2                    "
    PRINT " 3 - LPT3                    "
    PRINT " 4 - COM1                    "
    PRINT " 5 - COM2                    "
    PRINT " 6 - DATA FILE               "
    INPUT "ENTER CHOICE (0-6) :", IZOUTPUT
'    IF ((IZOUTPUT < 0) OR (IZOUTPUT > 6)) THEN GOTO 205

SELECT CASE IZOUTPUT
        CASE 0
                ZOUTPUT$ = "SCRN:"
```

```
        CASE 1
                ZOUTPUT$ = "LPT1:"
        CASE 2
                ZOUTPUT$ = "LPT2:"
        CASE 3
                ZOUTPUT$ = "LPT3:"
        CASE 4
                ZOUTPUT$ = "COM1:"
        CASE 5
                ZOUTPUT$ = "COM2:"
        CASE 6
                PRINT " "
                INPUT "ENTER OUTPUT DATA FILE NAME: ", ZOUTPUT$
END SELECT

                OPEN ZOUTPUT$ FOR OUTPUT AS #2
                PRINT " "
                PRINT " "

END SUB

DEFINT K-L, N
DEFSNG Z
SUB DESCRIPT
'RIN       = INSIDE RADIUS
'ACTD     = ACTIVATION ENERGY FOR D2 DIFFUSION IN METAL
(CAL/MOLE) 'ACTS     = ACTIVATION ENERGY FOR HYDROGEN ISOTOPE
SOLUBILITY IN '          METALS (INDEPENDENT OF ISOTOPE)
(CAL/MOLE)
'APPL     = APPLICATION
'          = 0: LIFE STORAGE
'          = 1: STANDARD RECLAMATION
'          = 2: MULTIPLE-EXPOSURE RECLAMATION
'          = 3: FULL CAPABILITIES
'AREA     = PERMEATION SURFACE AREA (CM^2)
'AREACONF = SURFACE AREA OF DIAPHRAM FOR CONFINED VOLUME (CM^2)
'ROUT       = OUTSIDE RADIUS
'CONTAM   = CONTAMINATION RATE (CURIES/YEAR)
'DEIN()   = DETERIUM FILL DATA (ATM)
'DIFF     = DIFFUSIVITY IN METAL (CM^2/SEC) = PRED * EXP(-ACTD /
R1 / TEMP(KL)) 'GEOM     = GEOMETRY
'          = 0: PLANE SHEET
'          = 1: CYLINDER
'          = 2: SPHERE
'GHYD     = GRAMS HYDRIDE
'GTYPE    = GEOM + 1 TYPE PROBLEM ??????
'HYIN()   = HYDROGEN FILL DATA (ATM)
'ISO      = ISOTOPE TYPE
'          = 0: D&T
'          = 1: H
'          = 2: D
'          = 3: T
```

```
'KLMAX   = NUMBER OF EXPOSURE/OFFGASSING STEPS
'MTLTYPE = MATERIAL TYPE
'          = 0: 304L
'          = 1: 21-6-9
'          = 2: COPPER ALLOYS
'          = 3: ALUMINUM ALLOYS
'N         = NUMBER OF TIME STEPS
'PERM      = PERMEATION RATE (CC/CM/CM/SEC)
'PRED      = PRE-EXPONENTIAL FOR ARRHENIUS FORM OF DIFFUSIVITY
(CM^2/SEC) 'PRES     = PRE-EXPONENTIAL FOR ARRHENIUS FORM OF
SOLUBILITY BASED ON ISOTOPE '          PARTIAL PRESSURE ([CC
ISOTOPE]/[CC-METAL]/[ATM^0.5]) 'R1       = IDEAL GAS CONSTANT
(CAL/MOLE-K)
'TEMP()    = TEMPERATURE (C OR K)
'THICK     = MATERIAL THICKNESS
'THOURS    = NUMBER OF HOURS PER DAY AT PERMEATION TEMPERATURE
(HOURS/DAY) 'TINPUT  = 0: GRAMS
'          = 1: ATMOSPHERES
'TINSIDE = B.C. ON INSIDE OF CONTAINER
'          = 0: TRITIUM IN GAS BOTTLE
'          = 1: TRITIUM IN A SOLID STORAGE?
'          = 2: TRITIUM FLOW THROUGH INSIDE STRUCTURE

'TOUT      = B.C. TYPE ON OUTSIDE OF CONTAINER
'          = 0: CONC. OUTSIDE 0
'          = 1: CONC. OUTSIDE CONSTANT
'          = 2: CONFINED EXTERNAL VOLUME
'          = 3: DIFFUSION FROM INSIDE AND OUTSIDE
'TRIN()    = TRITIUM FILL DATA (ATM)
'U(0)      = INTERIOR SURFACE CONCENTRATION (CC DISSOCIATED
ISOTOPE/CC METAL) '            IF BASED ON ISOTOPE PARTIAL
PRESSURE,
'            U(O) = SQR(PARTIAL PRESSURE) * PREXPS * EXP(-ACTS / R1
/ TEMP(KL)) 'VOL      = VOLUME OF VESSEL (CC)
END SUB

DEFSNG J-N
FUNCTION FFIN (I, TIME)
SHARED BCIN(), FIN(), HIN(), TIN()
'SET INTERNAL B.C. "FORCING FUNCTION"

SELECT CASE BCIN(I)
   CASE IS = 1        'CONSTANT SURFACE TEMP.
      PRINT "ERROR IN FFIN: BCIN = 1": END
   CASE IS = 2        'INSULATED SURFACE
      FFIN = 0!
   CASE IS = 3        'CONSTANT ENERGY FLUX = FIN(I)
      FFIN = FIN(I)
   CASE IS = 4        'CONSTANT CONVECTIVE HEAT LOSS
      FFIN = HIN(I) * TIN(I)
   CASE IS = 5        'SURFACE TEMP. A FUNCTION OF TIME IN SUB
"FTIN"       PRINT "ERROR IN FFIN: BCIN = 5": END
```

```
   CASE IS = 6       'ENTER DESIRED FLUX AS A FUNCTION OF TIME
(WATTS/CM^2)       FFIN = -9999!
   CASE IS = 7      ''BULK' CONVECTION TEMP. AS A FUNCT OF TIME IN
SUB "FTIN"       FFIN = HIN(I) * FTIN(TIME)
   CASE IS = 8
      PRINT "ERROR IN FFIN: BCIN = 8": END
END SELECT

END FUNCTION

FUNCTION FFOUT (I, TIME)
SHARED BCOUT(), FOUT(), HOUT(), TOUT(), RIN(), ROUT(), MTOT,
TOLD() 'SET EXTERNAL B.C. "FORCING FUNCTION"

SELECT CASE BCOUT(I)
   CASE IS = 1       'CONSTANT SURFACE TEMP.
      FFOUT = 0!
   CASE IS = 2       'INSULATED SURFACE
      FFOUT = 0!
   CASE IS = 3       'CONSTANT ENERGY FLUX = FOUT(I)
      FFOUT = FOUT(I)
   CASE IS = 4       'CONSTANT CONVECTIVE HEAT LOSS
      FFOUT = HOUT(I) * TOUT(I)
   CASE IS = 5       'SURFACE TEMP. A FUNCTION OF TIME (C)
      FFOUT = -9999!
   CASE IS = 6       'ENTER DESIRED FLUX AS A FUNCTION OF TIME
(WATTS/CM^2)       N = 9
      Q = 2.5      '.408             'LIQUID FLOW, L/MIN
      A = PI * 2 * RIN(2) * (PI * 2 * ROUT(1) * 1.25 * N)    'COIL
AREA, CM^2       HEATVAP = 199.1         'J/G
      DEN = .808              'G/CM^3
      CP = .25                'CAL/G-C
      TN2 = -196              'C
      M = Q * 1000 * DEN / 60!      'MASS FLOW (G/SEC)
      QLOAD = M * (HEATVAP + CP / .23901 * (TOLD(MTOT) - TN2))
'HEAT LOAD (WATTS)       F = -QLOAD / A                 '
HEAT FLUX (WATTS/CM^2)       FFOUT = F
'      PRINT "FFOUT = "; F

   CASE IS = 7   'BULK' CONVECTION TEMP. AS A FUNCT OF TIME IN
SUB "FTOUT"       FFOUT = HOUT(I) * FTOUT(TIME)
   CASE IS = 8
      PRINT "ERROR IN FFOUT: BCOUT = 8": END
END SELECT

END FUNCTION

DEFINT M
FUNCTION FICOND (I, X)
SHARED THICK()
'ENTER INITIAL CONDITION FUNCTION AS A FUNCTION OF POSTION
'(ADDITIONAL VARIABLES NEEDED CAN BE ADDED TO SHARE STATEMENT)
```

```
IF (I = 1) THEN
   FICOND = 100! * SIN(4 * PI * X / THICK(1))
ELSE
   FICOND = 0!
END IF

END FUNCTION

FUNCTION FSAMPLE1 (T, X)
SHARED THICK(), ALPHA()

FSAMPLE1 = 100! * EXP(-16! * PI ^ 2 * ALPHA(1) * T / THICK(1) ^
2) * SIN(4! * PI * X / THICK(1))

END FUNCTION

FUNCTION FSAMPLE2 (T, X)
SHARED APPL, ALPHA(), HOUT(), THICK(), TZERO()
DIM B(6)
'ANALYTICAL SOLUTION FOR PROBLEM IN RECTANGULAR COORDINATES '
DT/DX=0 AT X=0, DT/DX+H2T = 0 AT X=L, T=T0 AT T = 0
'(K IS ASSUMED TO BE 1, THICK = 0.3937 IN = 1 CM)
'FIRST SIX ROOTS ARE FOR SOLUTION WHEN L = .3937 IN (1CM) AND '
H2 =7E+4 WATTS/M^2-C

B(1) = 1.3766
B(2) = 4.1746
B(3) = 7.064
B(4) = 10.0339
B(5) = 13.0584
B(6) = 16.1177
SUM = 0
FOR I = 1 TO 6
  COEF = (B(I) ^ 2 + HOUT(1) ^ 2) / (THICK(1) * (B(I) ^ 2 +
HOUT(1) ^ 2) + HOUT(1))    FUNCS = SIN(B(I) * THICK(1)) / B(I) *
COS(B(I) * X)
  TERM = EXP(-ALPHA(1) * B(I) ^ 2 * T) * COEF * FUNCS
  SUM = SUM + TERM
  IF (DEBUG = 1) THEN
    PRINT "FSAMPLE2 OUTPUT"
    PRINT "COEF = "; COEF
    PRINT "TERM = "; TERM
    PRINT "SUM = "; SUM
    CALL PAUSE
  END IF
NEXT I
IF (ABS(TERM) / ABS(SUM) < .01) THEN
  FSAMPLE2 = 2 * TZERO(APPL) * SUM
ELSE
  'OUTPUT ZERO IF SERIES DOES NOT CONVERGE
  FSAMPLE2 = 0
END IF
```

```
END FUNCTION

FUNCTION FTIN (TIME)
SHARED HIN(), TZERO(), TOLD(), RIN()
'ENTER INTERNAL BOUNDARY CONDITION AS A FUNCTION OF TIME
'ADD VARIABLES NEED TO SHARE STATEMENT

I = 1
THEATA = 4! * HIN(I) / 4.2 / (RIN(I) * 2!) / .8 / .1 * .23901
FTIN = (TZERO(I) - TOLD(0)) * EXP(-THEATA * TIME) + TOLD(0) END
FUNCTION

FUNCTION FTOUT (TIME)
SHARED ISAMPLE, TZERO(), TOLD(), MTOT
'ENTER EXTERNAL BOUNDARY CONDITION AS A FUNCTION OF TIME
'ADD VARIABLES NEED TO SHARE STATEMENT

'FTOUT = (TZERO(1) - -200) * EXP(-.29 * TIME) + -200
TN2 = -196                                 'C

FTOUT = (TOLD(MTOT) + TN2) / 2!

END FUNCTION

DEFINT J
SUB ICONDITION (APPL, ICOND(), M(), MTOT, R(), TOLD())
SHARED ZINPUT$, THICK(), TZERO(), DEBUG
CLS

SELECT CASE ZINPUT$

        CASE IS = "KYBD:"
FOR I = 1 TO APPL
   SELECT CASE ICOND(I)

   CASE IS = 1
     PRINT "INTIAL CONDITION FOR MATERIAL"; I
     INPUT "Enter Initial Material Temperature (C)"; TZERO(I)
IF (I = 1) THEN
        FOR J = 0 TO M(1)
          TOLD(J) = TZERO(I)
        NEXT J
     ELSE
        FOR J = 1 TO M(2)
          TOLD(M(1) + J) = TZERO(I)
        NEXT J
     END IF

   CASE IS = 2
     PRINT "REGION IS DIVIDED INTO"; M(I); "SECTIONS"
     PRINT "ENTER TEMPERATURE AT EACH LOCATION": PRINT ""
     IF (I = 1) THEN
```

```
            FOR J = 0 TO M(1)
              PRINT "J = "; J; ",  R (CM) = "; R(1, J)
              INPUT "TEMPERATURE (C) AT R = "; TOLD(J)
              PRINT " "
            NEXT J
         ELSE
            FOR J = 1 TO M(2)
              PRINT "J = "; J; ",  R (CM) = "; R(2, J)
              INPUT "TEMPERATURE (C) AT R = "; TOLD(M(1) + J)
              PRINT " "
            NEXT J
         END IF

       CASE IS = 3
         IF (I = 1) THEN
            FOR J = 0 TO M(1)
              TOLD(J) = FICOND(1, R(1, J))
            NEXT J
         ELSE
            FOR J = 1 TO M(2)
              TOLD(M(1) + J) = FICOND(1, R(2, J))
            NEXT J
         END IF

       END SELECT

       PRINT "": PRINT ""

   NEXT I
   CLS
           CASE IS = "PRGM"
   READ DENT$

           CASE ELSE
   INPUT #1, DENT$

   END SELECT


   IF (DEBUG = 1) THEN
      FOR I = 1 TO APPL
         PRINT "INITIAL CONDITIONS"
            FOR J = M(I - 1) TO M(I - 1) + M(I)
              PRINT "J = "; J; ",  R (CM) = "; R(I, J - M(I - 1)); "
   TOLD = "; TOLD(J)           NEXT J
            CALL PAUSE
            CLS
      NEXT I
   END IF

   END SUB
```

```
SUB MATRIXA (MTOT, AA(), AB(), AC())
SHARED APPL, M(), BCIN(), BCOUT(), DEBUG, EPS()
SHARED GAMMA(), DELTA(), LAMIN(), LAMOUT(), BETAIN(), BETAOUT()
'SET-UP MATRIX OF COEFFICIENTS FOR KNOWN TEMPERATURES

'FOR J=0
IF (BCIN(1) = 1 OR BCIN(1) = 5) THEN
   AB(0) = 1!
   AC(0) = 0!
ELSE
   AB(0) = 2! * (1! + GAMMA(1) * BETAIN(1) - DELTA(0) * LAMIN(1))
  AC(0) = -2! * GAMMA(1)
END IF
FOR J = 1 TO M(1) - 1
   AA(J) = -GAMMA(1) + DELTA(J)
   AB(J) = 2! * (1! + GAMMA(1))
   AC(J) = -GAMMA(1) - DELTA(J)
NEXT J
IF (APPL = 1) THEN
   'FOR J=M(1)
   IF (BCOUT(1) = 1 OR BCOUT(1) = 5) THEN
      AA(M(1)) = 0!
      AB(M(1)) = 1!
   ELSE
     AA(M(1)) = -2! * GAMMA(1)
     AB(M(1)) = 2! * (1! + GAMMA(1) * BETAOUT(1) + DELTA(M(1)) *
LAMOUT(1))    END IF
ELSE
   'FOR J=M(1)
      AA(M(1)) = EPS(1)
      AB(M(1)) = -(EPS(1) + EPS(2))
      AC(M(1)) = EPS(2)
   FOR J = M(1) + 1 TO MTOT - 1
      AA(J) = -GAMMA(2) + DELTA(J)
      AB(J) = 2! * (1! + GAMMA(2))
      AC(J) = -GAMMA(2) - DELTA(J)
   NEXT J
   'FOR J=MTOT
   IF (BCOUT(2) = 1 OR BCOUT(2) = 5) THEN
      AA(MTOT) = 0!
      AB(MTOT) = 1!
   ELSE
     AA(MTOT) = -2! * GAMMA(2)
     AB(MTOT) = 2! * (1! + GAMMA(2) * BETAOUT(2) + DELTA(MTOT) *
LAMOUT(2))    END IF
END IF

IF (DEBUG = 1) THEN
      PRINT "AA,AB,AC VECTORS, BETAOUT(1)="; BETAOUT(1)
      FOR J = 0 TO M(1)
         PRINT J, AA(J), AB(J), AC(J)
       NEXT J
```

```
         CALL PAUSE
      IF (APPL = 2) THEN
         PRINT "AA,AB,AC VECTORS, BETAOUT(2)="; BETAOUT(2)
         FOR J = M(1) TO MTOT
            PRINT J, AA(J), AB(J), AC(J)
          NEXT J
          CALL PAUSE
      END IF
   END IF
   END SUB

   SUB MATRIXB (MTOT, BA(), BB(), BC())
   SHARED APPL, M(), BCIN(), BCOUT(), DEBUG, EPS()
   SHARED GAMMA(), DELTA(), LAMIN(), LAMOUT(), BETAIN(), BETAOUT()
   'SET-UP MATRIX OF COEFFICIENTS FOR KNOWN TEMPERATURES

   'FOR J=0
   IF (BCIN(1) = 1 OR BCIN(1) = 5) THEN
      BB(0) = 1!
      BC(0) = 0!
   ELSE
     BB(0) = 2! * (1! - GAMMA(1) * BETAIN(1) + DELTA(0) * LAMIN(1))
   BC(0) = 2! * GAMMA(1)
   END IF

   FOR J = 1 TO M(1) - 1
      BA(J) = GAMMA(1) - DELTA(J)
      BB(J) = 2! * (1! - GAMMA(1))
      BC(J) = GAMMA(1) + DELTA(J)
   NEXT J

   IF (APPL = 1) THEN
      'FOR J=M(1)
      IF (BCOUT(1) = 1 OR BCOUT(1) = 5) THEN
         BA(M(1)) = 0!
         BB(M(1)) = 1!
      ELSE
        BA(M(1)) = 2! * GAMMA(1)
        BB(M(1)) = 2! * (1! - GAMMA(1) * BETAOUT(1) - DELTA(M(1)) *
   LAMOUT(1))    END IF
   ELSE
      'FOR J=M(1)
         BA(M(1)) = -EPS(1)
         BB(M(1)) = EPS(1) + EPS(2)
         BC(M(1)) = -EPS(2)
      FOR J = M(1) + 1 TO MTOT - 1
         BA(J) = GAMMA(2) - DELTA(J)
         BB(J) = 2! * (1! - GAMMA(2))
         BC(J) = GAMMA(2) + DELTA(J)
      NEXT J
      'FOR J=MTOT
      IF (BCOUT(2) = 1 OR BCOUT(2) = 5) THEN
```

```
      BA(MTOT) = 0!
      BB(MTOT) = 1!
   ELSE
     BA(MTOT) = 2! * GAMMA(2)
     BB(MTOT) = 2! * (1! - GAMMA(2) * BETAOUT(2) - DELTA(MTOT) *
LAMOUT(2))    END IF
END IF

IF (DEBUG = 1) THEN
      PRINT "BA,BB,BC VECTORS, BETAOUT(1)="; BETAOUT(1)
      FOR J = 0 TO M(1)
         PRINT J, BA(J), BB(J), BC(J)
       NEXT J
       CALL PAUSE
   IF (APPL = 2) THEN
      PRINT "BA,BB,BC VECTORS, BETAOUT(2)="; BETAOUT(2)
      FOR J = M(1) TO MTOT
         PRINT J, BA(J), BB(J), BC(J)
       NEXT J
       CALL PAUSE
   END IF
END IF

END SUB

SUB MATTERMS (GAMMA(), EPS(), DELTA(), LAMIN(), LAMOUT(),
BETAIN(), BETAOUT()) SHARED APPL, M(), MTOT, GEOM(), AREA(),
ALPHA(), DT, HIN(), HOUT(), K() SHARED R(), DR(), DEBUG


FOR I = 1 TO APPL
   GAMMA(I) = ALPHA(I) * DT / DR(I) ^ 2
   LAMIN(I) = DR(I) * HIN(I) / K(I)
   LAMOUT(I) = DR(I) * HOUT(I) / K(I)
   BETAIN(I) = 1! + LAMIN(I)
   BETAOUT(I) = 1! + LAMOUT(I)

   IF (I = 1) THEN
      EPS(1) = 0!
      EPS(2) = 0!
   ELSE
      EPS(1) = K(1) * AREA(1) / DR(1)
      EPS(2) = K(2) * AREA(2) / DR(2)
   END IF

   SELECT CASE GEOM(I)
     CASE 1
       OMEGA = 0!
     CASE 2
       OMEGA = 1!
     CASE 3
       OMEGA = 2!
```

```
    END SELECT

    IF (I = 1) THEN
       FOR J = 0 TO M(1)
         IF (OMEGA = 0!) THEN
           DELTA(J) = 0!
         ELSE
           DELTA(J) = OMEGA * ALPHA(1) * DT / R(1, J) / DR(1)
   END IF
       NEXT J
    ELSE
       FOR J = M(1) TO MTOT
         IF (OMEGA = 0!) THEN
           DELTA(J) = 0!
         ELSE
           DELTA(J) = OMEGA * ALPHA(2) * DT / R(2, J - M(1)) /
DR(2)           END IF
       NEXT J
    END IF

NEXT I

IF (DEBUG = 1) THEN
    PRINT "MTOT = "; MTOT
  FOR I = 1 TO APPL
    PRINT "ALPHA   = "; ALPHA(I)
    PRINT "GAMMA   = "; GAMMA(I)
    PRINT "LAMIN   ="; LAMIN(I)
    PRINT "LAMOUT  ="; LAMOUT(I)
    PRINT "BETAIN  ="; BETAIN(I)
    PRINT "BETAOUT ="; BETAOUT(I)
    PRINT "EPSILON ="; EPS(I)
    CALL PAUSE
    PRINT ""
    PRINT "DELTA TERMS"
    FOR J = 0 TO M(1)
      PRINT "J = "; J; " R = "; R(1, J); " DELTA = "; DELTA(J)
NEXT J
    CALL PAUSE
    FOR J = M(1) TO MTOT
      PRINT "J = "; J; " R = "; R(2, J - M(1)); " DELTA = ";
DELTA(J)     NEXT J
    CALL PAUSE
  NEXT I
END IF

END SUB

SUB MTLPROP (APPL, MTLTYPE(), K(), ALPHA())
SHARED DEBUG
'DEFINE PROPERTIES FOR MATERIALS SECLECTED
' K(I) = THERMAL CONDUCTIVITY      (WATTS/M-C)
```

```
' DENSITY = MATERIAL DENSITY        (G/CM^3)
' HEATCAP = MATERIAL HEAT CAPACITY (CAL/G-C)

FOR I = 1 TO APPL
      SELECT CASE MTLTYPE(I)
      CASE 1
      K(I) = 16.26              '9.4 BTU/HR-FT^2-F/FT
      DENSITY = 7.83            '7.83 G/CM^3
      HEATCAP = .12             '.12 CAL/G-C FOR STEEL

      CASE 2
      K(I) = 0
      DENSITY = 0
      HEATCAP = 0

      CASE 3    'COPPER
      K(I) = 398
      DENSITY = 8.92
      HEATCAP = .0924         '5.44+0.001462*T(K) (CAL/MOLE-C) /
63.57(G/MOLE)
      CASE 4   'ALUMINUM
      K(I) = 273
      DENSITY = 2.7
      HEATCAP = .214          '4.80+0.00322*T(K) (CAL/MOLE-C) /
26.97(G/MOLE)
      CASE 6
      INPUT "INPUT THERMAL CONDUCTIVITY (WATTS/M-C) "; K(I)
INPUT "INPUT MATERIAL DENSITY (G/CM^3) "; DENSITY
      INPUT "MATERIAL HEAT CAPACITY (CAL/G-C) "; HEATCAP

      END SELECT

      'CONVERT MATERIAL PROPERTIES
      '  K FROM (WATTS/M-C) TO (WATTS/CM-C)
      '  HEATCAP FROM (CAL/G-C) TO (WATT-SEC/G-C)
      '  100 CM/M
      '  0.23901 CAL/WATT-SEC
      K(I) = K(I) / 100!
      HEATCAP = HEATCAP / .23901

      SELECT CASE MTLTYPE(I)
        CASE 5
        K(I) = 1!
        DENSITY = 1!
        HEATCAP = 1!
     END SELECT

    'COMPUTE ALPHA = K/RHO-CP (CM^2/SEC)
    ALPHA(I) = K(I) / DENSITY / HEATCAP

    IF (DEBUG = 1) THEN
      PRINT "MATERIAL "; I
```

```
        PRINT "K = "; K(I)
        PRINT "DEN="; DENSITY
        PRINT "CP ="; HEATCAP
        PRINT "ALPHA = "; ALPHA(I)
        PRINT ""
        CALL PAUSE
      END IF

NEXT I
END SUB

DEFINT K-L, N
SUB PAUSE
PRINT "PRESS ANY KEY TO CONTINUE PROGRAM"
DO
LOOP WHILE INKEY$ = ""
     PRINT " "

END SUB

SUB PCALC (GTYPE, A(), B(), C(), D(), X(), Y(), P(), U())
SHARED M
'subroutine for X,Y,P and U
calculations......................... X(M - 1) = (D(M - 1) + A(M
- 1) * P(M)) / B(M - 1)
Y(M - 1) = C(M - 1) / B(M - 1)
FOR I = M - 2 TO 1 STEP -1
   X(I) = (D(I) + A(I) * X(I + 1)) / (B(I) - A(I) * Y(I + 1))
Y(I) = C(I) / (B(I) - A(I) * Y(I + 1))
NEXT I
FOR I = 1 TO M - 1
   P(I) = X(I) + Y(I) * P(I - 1)
   U(I) = P(I) + U(I)
NEXT I

END SUB

DEFSNG K-L, N
SUB SIZE (APPL, GEOM(), CONFIG, THICK(), RIN(), ROUT(), M(),
DR(), R()) SHARED ZINPUT$, ISAMPLE
DIM THICK0(2), RIN0(2), ROUT0(2)

THICK0(1) = 1
RIN0(1) = .805
ROUT0(1) = .95

THICK0(2) = 1
RIN0(2) = .1575
ROUT0(2) = .1875


SELECT CASE ZINPUT$
```

```
        CASE IS = "KYBD:"
SELECT CASE ISAMPLE
   CASE 0
PRINT "         DIMENSIONS OF STRUCTURE 1"
IF GEOM(1) = 1 THEN
   PRINT "Input material 1 wall Thickness (in.) (Default =";
THICK0(1); " inch)"    INPUT ""; THICK(1): PRINT ""
   IF (THICK(1) = 0) THEN THICK(1) = THICK0(1)
ELSE
   PRINT "Input material 1 Inner Radius (in.) (Default = ";
RIN0(1); " in)"    INPUT ""; RIN(1): PRINT ""
   IF (RIN(1) = 0) THEN RIN(1) = RIN0(1)

   PRINT "Input material 1 Outer Radius (in.) (Default = ";
ROUT0(1); " in)"    INPUT ""; ROUT(1)
   IF (ROUT(1) = 0) THEN ROUT(1) = ROUT0(1)
END IF

IF (APPL > 1) THEN
   PRINT ""
   PRINT "         DIMENSIONS OF STRUCTURE 2"
   IF GEOM(2) = 1 THEN
      PRINT "Input material 2 wall Thickness (in.) (Default =";
THICK0(2); " inch)"       INPUT ""; THICK(2): PRINT ""
      IF (THICK(2) = 0) THEN THICK(2) = THICK0(2)
   ELSE
      IF (CONFIG = 1) THEN
         RIN(2) = ROUT(1)
         PRINT ""
         PRINT "Inner Radius for material 2 set equal to Outer
Radius of material 1"          PRINT "(Concentric geometry
chosen)"
         PRINT ""
      ELSE
         PRINT "Input material 2 Inner Radius (in.) (Default = ";
RIN0(2); " in)"           INPUT ""; RIN(2): PRINT ""
         IF (RIN(2) = 0) THEN RIN(2) = RIN0(2)
      END IF

      PRINT "Input material 2 Outer Radius (in.) (Default = ";
ROUT0(2); " in)"       INPUT ""; ROUT(2)
      IF (ROUT(2) = 0) THEN ROUT(2) = ROUT0(2)
   END IF
END IF
PRINT "": PRINT ""
   CASE 1
RIN(1) = 0!
THICK(1) = .3937
   CASE 2
RIN(1) = 0!
THICK(1) = .3937
END SELECT
```

```
            CASE IS = "PRGM"
READ THICK
READ RIN, ROUT

            CASE ELSE
INPUT #1, THICK
INPUT #1, RIN, ROUT

END SELECT


'geometric information
FOR I = 1 TO APPL
   IF GEOM(I) = 2 OR GEOM(I) = 3 THEN
     THICK(I) = ROUT(I) - RIN(I)
   ELSE
     RIN(I) = 0!
   END IF
NEXT I

'CONVERT DIMENSIONS TO CM
FOR I = 1 TO APPL
   THICK(I) = THICK(I) * 2.54
   RIN(I) = RIN(I) * 2.54
   ROUT(I) = ROUT(I) * 2.54
   DR(I) = THICK(I) / M(I)
   IF (I = 1) THEN
      FOR J = 0 TO M(I)
         R(I, J) = RIN(I) + J * DR(I)
      NEXT J
   ELSE
      IF (CONFIG = 1) THEN
         FOR J = 0 TO M(I)
            R(I, J) = RIN(I) + J * DR(I)
         NEXT J
      ELSE
         FOR J = 0 TO M(I)
            R(I, J) = ROUT(I) - J * DR(I)
         NEXT J
      END IF
   END IF
NEXT I
'
END SUB

DEFSNG J
SUB SOLVE (M(), MTOT, TNEW(), AA(), AB(), AC(), D())
SHARED APPL, DEBUG
DIM WORK(100)     'WORK SPACE VECTOR

'FOR I = 1 TO APPL
```

```
   IF (AB(0) = 0) THEN PRINT "ERROR 1 IN SOLVE ROUTINE": END
BET = AB(0)
   TNEW(0) = D(0) / BET

   FOR J = 1 TO MTOT
      WORK(J) = AC(J - 1) / BET
      BET = AB(J) - AA(J) * WORK(J)
      IF (BET = 0) THEN PRINT "ERROR 2 IN SOLVE ROUTINE": END
 TNEW(J) = (D(J) - AA(J) * TNEW(J - 1)) / BET
   NEXT J
   FOR J = MTOT - 1 TO 0 STEP -1
      TNEW(J) = TNEW(J) - WORK(J + 1) * TNEW(J + 1)
   NEXT J
'NEXT I

IF (DEBUG = 2) THEN
  PRINT "TNEW VALUES"
  FOR J = 0 TO M
    PRINT J, TNEW(J)
  NEXT J
  CALL PAUSE
  IF (APPL = 2) THEN
     PRINT "TNEW VALUES"
     FOR J = 0 TO M
       PRINT J, TNEW(J)
     NEXT J
     CALL PAUSE
   END IF
END IF

END SUB

DEFINT J
SUB TIMEPARMS (DR(), ALPHA(), DT, N, TMAX)
SHARED ZINPUT$, APPL
'subroutine for time step and totol time ....................

SELECT CASE ZINPUT$
        CASE IS = "KYBD:"

DTMIN = 2! * DR(1) ^ 2 / ALPHA(1)
IF (APPL > 1) THEN
  DT = 2! * DR(2) ^ 2 / ALPHA(2)
  IF (DT < DTMIN) THEN DTMIN = DT
END IF

CLS
PRINT "TIME STEP AND TOTAL TIME"
PRINT ""
PRINT "MAXIMUM ESTIMATED TIME STEP TO USE (SEC) = "; DTMIN
INPUT "Enter time step (sec) ", DT
IF (DT = 0) THEN DT = DTMIN
```

```
PRINT ""
INPUT "Enter total analysis time (minutes) (Default = 1 minute)";
TMAX IF (TMAX = 0) THEN TMAX = 1!

PRINT ""
N = TMAX * 60! / DT
PRINT "NUMBER OF TIME STEPS = "; N

INPUT "CHANGE TIME STEPSIZE? (Y/N) ", A$
IF (A$ = "y" OR A$ = "Y") THEN
   INPUT "Enter time step (sec) ", DT
END IF
PRINT "": PRINT ""

        CASE IS = "PRGM"
READ DT
READ THOURS

        CASE ELSE
INPUT #1, THOURS

END SELECT


'SCALE TIME

'CONVERT TIME TO SECONDS
TMAX = TMAX * 60

END SUB

SUB VECTORC (MTOT, TIME, C())
SHARED APPL, M(), BCIN(), BCOUT(), DEBUG
SHARED GAMMA(), DELTA(), DT, DR(), HIN(), HOUT(), K()

I = 1                         ' FOR INSIDE B.C. OF MATERIAL 1
   SELECT CASE BCIN(I)

   CASE IS = 1, 2, 5
     VMUIN = 0!

   CASE IS = 3, 4, 6, 7
     VMUIN = DR(I) / K(I) * (FFIN(I, TIME) + FFIN(I, TIME + DT))
/ 2!
   END SELECT

I = APPL                      'FOR EXTERNAL B.C.
   SELECT CASE BCOUT(I)

   CASE IS = 1, 2, 5
     VMUOUT = 0!
```

```
    CASE IS = 3, 4, 6, 7
      VMUOUT = DR(I) / K(I) * (FFOUT(I, TIME) + FFOUT(I, TIME +
DT)) / 2!

    END SELECT

'FOR J=0
C(0) = 4! * (GAMMA(1) - DELTA(0)) * VMUIN

FOR J = 1 TO MTOT - 1
   C(J) = 0!
NEXT J

I = APPL
'FOR MTOT
C(MTOT) = 4! * (GAMMA(I) + DELTA(MTOT)) * VMUOUT



IF (DEBUG = 1) THEN
  PRINT "C VECTOR, VMU = "; VMUIN; " VMUOUT ="; VMUOUT
  FOR J = 0 TO M(1)
    PRINT J, C(J)
  NEXT J
  CALL PAUSE
  IF (APPL > 1) THEN
  PRINT "C VECTOR"
  FOR J = M(1) + 1 TO MTOT
    PRINT J, C(J)
  NEXT J
  CALL PAUSE

  END IF
END IF
END SUB

DEFSNG J
SUB VECTORD (M(), TOLD(), BA(), BB(), BC(), C(), D())
SHARED APPL, DEBUG, MTOT
'CALCULATE VECTOR D: VEC[D] = MAT[B] * VEC[TOLD] + VEC[C]

'FOR J = 0
D(0) = BB(0) * TOLD(0) + BC(0) * TOLD(1) + C(0)

IF (APPL = 1) THEN
   I = 1
ELSE
   I = 2
END IF

FOR J = 1 TO MTOT - 1
   D(J) = BA(J) * TOLD(J - 1) + BB(J) * TOLD(J) + BC(J) * TOLD(J
```

```
+ 1) + C(J) NEXT J

'FOR J = MTOT
D(MTOT) = BA(MTOT) * TOLD(MTOT - 1) + BB(MTOT) * TOLD(MTOT) +
C(MTOT)


IF (DEBUG = 1) THEN
  PRINT "D VECTOR"
  FOR J = 0 TO M(1)
    PRINT J, D(J)
  NEXT J
  CALL PAUSE
  IF (APPL > 1) THEN
     PRINT "D VECTOR"
     FOR J = M(1) TO MTOT
       PRINT J, D(J)
     NEXT J
     CALL PAUSE
   END IF
END IF

END SUB
```

END

DATE FILMED

10 15 93