THERMAL AND FLOW ANALYSIS SUBROUTINES
FOR THE SINDA-VERSION 9 COMPUTER ROUTINE

REPORT NO. 00.1582
24 September 1973

(NASA-CR-134121) THERMAL AND FLOW ANALYSIS SUBROUTINES FOR THE SINDA-VERSION 9 COMPUTER ROUTINE (LTV Aerospace Corp.)
274 p HC $15.75

CSCL 20D

Unclas

VOUGHT SYSTEMS DIVISION
LTV AEROSPACE CORPORATION
THERMAL AND FLOW ANALYSIS SUBROUTINES
FOR THE SINDA-VERSION 9 COMPUTER ROUTINE

Report No. 00.1582

24 September 1973

Submitted by

Vought Systems Division
LTV Aerospace Corporation
P.O. Box 5907
Dallas, Texas

To

National Aeronautics and Space Administration
Lyndon B. Johnson Space Center
Houston, Texas

Prepared by:

A. Oren
D. R. Williams

Approved by:

R. J. French, Supervisor
Environmental Control/
Life Support Systems
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>SUMMARY AND INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td>2.0</td>
<td>DISCUSSION OF METHODS</td>
<td>2</td>
</tr>
<tr>
<td>2.1</td>
<td>Thermal Analysis Features</td>
<td>2</td>
</tr>
<tr>
<td>2.1.1</td>
<td>Convection Conductors</td>
<td>2</td>
</tr>
<tr>
<td>2.1.2</td>
<td>Flow Conductors</td>
<td>4</td>
</tr>
<tr>
<td>2.1.3</td>
<td>Heat Exchanger Analysis</td>
<td>5</td>
</tr>
<tr>
<td>2.1.4</td>
<td>Inline Heater Analysis</td>
<td>8</td>
</tr>
<tr>
<td>2.1.5</td>
<td>Cabin Analysis</td>
<td>8</td>
</tr>
<tr>
<td>2.1.6</td>
<td>Radiation Interchange Analysis</td>
<td>14</td>
</tr>
<tr>
<td>2.2</td>
<td>Fluid Flow Analysis</td>
<td>25</td>
</tr>
<tr>
<td>2.2.1</td>
<td>Overall Flow Model Description</td>
<td>25</td>
</tr>
<tr>
<td>2.2.2</td>
<td>Tube Conductor Determination</td>
<td>29</td>
</tr>
<tr>
<td>2.2.3</td>
<td>Valve Analysis</td>
<td>31</td>
</tr>
<tr>
<td>2.2.4</td>
<td>Pressure-Flow Network Solution</td>
<td>37</td>
</tr>
<tr>
<td>2.2.5</td>
<td>Pump and System Pressure-Flow Matching</td>
<td>40</td>
</tr>
<tr>
<td>3.0</td>
<td>MODIFICATIONS TO SINDA SUBROUTINES</td>
<td>45</td>
</tr>
<tr>
<td>3.1</td>
<td>Preprocessor Modifications</td>
<td>45</td>
</tr>
<tr>
<td>3.2</td>
<td>Execution Routine Modifications</td>
<td>45</td>
</tr>
<tr>
<td>4.0</td>
<td>USER SUBROUTINES</td>
<td>46</td>
</tr>
<tr>
<td>5.0</td>
<td>SAMPLE PROBLEM</td>
<td>91</td>
</tr>
<tr>
<td>6.0</td>
<td>REFERENCES</td>
<td>94</td>
</tr>
</tbody>
</table>

## APPENDICES

<table>
<thead>
<tr>
<th>Appendix</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>SUBROUTINE LISTINGS</td>
<td>A-1</td>
</tr>
<tr>
<td>B</td>
<td>USERS DESCRIPTION FOR PLOTA</td>
<td>B-1</td>
</tr>
<tr>
<td>Figure</td>
<td>Title</td>
<td>Page</td>
</tr>
<tr>
<td>--------</td>
<td>----------------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>1</td>
<td>Illustration of Method Used to Determine Specular Surface Reflected View Factors</td>
<td>16</td>
</tr>
<tr>
<td>2</td>
<td>Flow System Schematic</td>
<td>26</td>
</tr>
<tr>
<td>3</td>
<td>Main Network and Subnetworks</td>
<td>28</td>
</tr>
<tr>
<td>4</td>
<td>Friction Factor vs Reynolds Number</td>
<td>32</td>
</tr>
<tr>
<td>5</td>
<td>Rate Limited Valve Operation</td>
<td>34</td>
</tr>
<tr>
<td>6</td>
<td>System/Pump Curve Solution</td>
<td>41</td>
</tr>
<tr>
<td>7</td>
<td>Flow Charts of PFCS and NTWRK</td>
<td>80</td>
</tr>
<tr>
<td>8</td>
<td>Flow Charts of NTWRK1 and NTWRKKN</td>
<td>81</td>
</tr>
<tr>
<td>9</td>
<td>Flow Chart of FLOCAL</td>
<td>82</td>
</tr>
<tr>
<td>10</td>
<td>Fluid Model of the Sample Problem</td>
<td>92</td>
</tr>
<tr>
<td>11</td>
<td>Structure Model for the Sample Problem</td>
<td>93</td>
</tr>
<tr>
<td>12</td>
<td>Radiator Temperatures For Sample Problem</td>
<td>148</td>
</tr>
<tr>
<td>13</td>
<td>System Temperatures For Sample Problem</td>
<td>149</td>
</tr>
<tr>
<td>14</td>
<td>System Pressures For Sample Problem</td>
<td>150</td>
</tr>
<tr>
<td>15</td>
<td>Radiator Flow Rates For Sample Problem</td>
<td>151</td>
</tr>
<tr>
<td>16</td>
<td>Pump Pressures For Sample Problem</td>
<td>152</td>
</tr>
<tr>
<td>17</td>
<td>System Pressures For Sample Problem</td>
<td>153</td>
</tr>
</tbody>
</table>
## LIST OF TABLES

<table>
<thead>
<tr>
<th>#</th>
<th>Table Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>New SINDA User Subroutines</td>
<td>47</td>
</tr>
<tr>
<td>2</td>
<td>Value of GC For Various Problem Units</td>
<td>79</td>
</tr>
<tr>
<td>3</td>
<td>Output For Sample Problem</td>
<td>94</td>
</tr>
<tr>
<td>B-1</td>
<td>Correspondence Between Fortran Unit No. &amp; I/O Device</td>
<td>B-9</td>
</tr>
</tbody>
</table>
1.0 SUMMARY AND INTRODUCTION

During the past decade extensive capabilities for combined thermal and fluid flow transient analysis was developed at the Vought Systems Division (VSD) of LTV Aerospace Corporation. The capabilities included (1) a pressure/flow solution for a general flow network (integrated with the finite difference temperature solution) including general valve analyses, orifice and pump analysis packages, (2) a number of special thermal analysis options including heat exchanger analysis, cavity radiant interchange analysis, cabin analysis, etc. and (3) a number of input/output capabilities such as automatic plotting, interrupt and restart, etc. These capabilities were included in a general purpose thermal analysis routine, MOTAR\textsuperscript{1*}, developed by VSD for NASA-JSC.

The objective of the effort described by this report was to incorporate these fluid flow analysis, special thermal analysis and input/output capabilities of the MOTAR routine into the SINDA\textsuperscript{7} routine which was developed by the TRW Corporation. This effort was performed under contract NAS9-6807 for NASA-JSC. All the capabilities were added in the form of user subroutines so that they may be added to different versions of SINDA with a minimum of programmer effort.

Two modifications were made to the existing subroutines of SINDA/VERSION 8 to incorporate the above subroutines. These were:

1. A modification to the preprocessor to permit actual values of array numbers, conductor numbers, node numbers or constant numbers supplied as array data to be converted to relative numbers.

2. Modifications to execution subroutine CNFAST to make it compatible with the radiant interchange user subroutine, RADIR.

This modified version of SINDA has been designated SINDA/VERSION 9.

A detailed discussion of the methods used for the capabilities added is presented in Section 2.0. The modifications for the SINDA subroutines are described in Section 3.0. User subroutines are described in Section 4.0, and a sample problem is given in Section 5.0. All subroutines added or modified are listed in Appendix A.

* Superscripts refer to references in Section 6.0
2.0 DISCUSSION OF METHODS

SINDA user subroutines were developed to incorporate the MOTAR routine capabilities for fluid/pressure analysis, thermal analysis of a flowing fluid and enclosure analysis into the SINDA routine. The analytical methods for these capabilities are described in this section. Thermal analysis features such as those methods required for analysis of a flowing fluid and those required for enclosure radiation analysis are described in Section 2.1. Pressure-flow analysis methods are described in Section 2.2.

2.1 Thermal Analysis Features

The calculation methods for (1) convection and flow thermal conductors for flow in a tube, (2) heat exchanger thermal performance, (3) inline heater thermal performance, (4) cabin thermal and mass balance, and (5) enclosure radiation thermal performance have been added to the SINDA library of user subroutines. The methods used are based on those from the MOTAR computer routine and are described in detail in the following sections.

2.1.1 Convection Conductors

Three user subroutines were prepared for the SINDA library to give to the user the capability of analyzing convection heat transfer for flow in a tube. These subroutines and their functions are:

- **CONV1** - Calculates heat transfer coefficient using relationships for convection in a flowing tube
- **CONV2** - Calculates the heat transfer coefficient using the Stanton number obtained from interpolating a curve
- **CONV3** - Interpolates a curve of heat transfer coefficients vs flowrates for the coefficient

The value of the convection conductor, $G_{ij}$, between a fluid lump and tube lump is given by the following relation for all three of the above routines:

$$G_{ij} = hA$$

where

- $h$ - the convection heat transfer coefficient
- $A$ - the convection area
CONVI1 uses one of several methods for determining the heat transfer coefficient, \( h \), for flowing fluid in a tube depending on the flow regime. The flow regime is assumed to be laminar when the Reynolds number is 2000 or less. For this regime the convection heat transfer coefficient is calculated by:

\[
h = \frac{k}{D} \left[ 3.66 \cdot F1 + \frac{0.0155 \cdot F2}{\frac{1}{R ep} \cdot \frac{X}{D}} \right]
\]

Equation (2) is a curve fit obtained by VMSC to approximate the Gratz solution\(^2\) to flow in a tube for values of \( \frac{X}{D} \frac{1}{R epPr} \) greater than 0.001.

The convection heat transfer coefficient for flow in a tube in the transition flow regime \((2000 < Re < 6400)\) is approximated in CONVI1 by the following relation:

\[
h = \frac{K}{D} \left[ 0.116 \left( Re^{2/3} - 125 \right) \frac{1}{Pr} \right]^{1/3}
\]

This relation was derived by Hausen\(^3\) and holds only for fully developed flow.

---

\(^1\) k = thermal conductivity
\(^2\) D = hydraulic diameter to flow
\(^3\) X = distance from tube entrance
\(^4\) Re = Reynolds number
\(^5\) \( \frac{m}{\mu} \) = flow rate of fluid
\(^6\) \( \mu \) = viscosity of fluid
\(^7\) P = wetted perimeter of fluid flow passage
\(^8\) F1 = An input factor for modifying fully developed flow
\(^9\) F2 = An input factor for modifying developing flow
The relation used in CONV1 to determine $h$ for turbulent flow ($Re > 6400$) is the following:

$$h = 0.023 \frac{K}{D} (Re)^{0.8} (Pr)^{1/3}$$

(4)

CONV2 supplies a more general option for determining the convection heat transfer coefficient. A curve of $St(Pr)^{2/3}$ vs Reynolds No. is interpolated to obtain the value of $St(Pr)^{2/3}$. That is,

$$St(Pr)^{2/3} = F(Re)$$

(5)

Where $St$ = Stanton number

$$St = Nu = \frac{Nu}{Re Pr}$$

$$St = \frac{h}{CpV}$$

$V$ = Average fluid velocity

$F(Re)$ = An arbitrary function of Reynonds number which the user can input as a table

The heat transfer coefficient is calculated by

$$h = K \frac{F(Re)}{D} Re(Pr)^{1/3}$$

(6)

In CONV3, the convection heat transfer coefficient is obtained by direct interpolation of a curve of heat transfer coefficient vs flowrate.

2.1.2 Flow Conductors

A method for calculating the value of flow conductors is required when analyzing a problem with fluid flowing in a tube. The flow conductor is a one way conductor from node i to node j and is calculated by

$$G_{ij} = \dot{W} C_{pi}$$

(7)

where:

$G_{ij}$ = the conductance from the upstream lump

$\dot{W}$ = the mass flow rate in the tube

$C_{pi}$ = the fluid specific heat for lump i
Two user subroutines FLOCN1 and FLOCN2, were prepared to calculate the values for the flow conductors. Both subroutines reference the flowrate array and an array containing conductor identification information. FLOCN1 assumes the specific heat is a function of temperature whereas; FLOCN2 assumes a constant value for specific heat.

2.1.3 Heat Exchanger Analysis

Four subroutines have been written to facilitate the thermal analysis of systems containing heat exchangers. These are HXCNT for analysis of counterflow heat exchangers, HXPAR for parallel flow heat exchangers, HXCROS for cross flow heat exchangers and HXEFF for any heat exchanger with an input effectiveness. These subroutines calculate the outlet temperatures of two sides based upon the inlet temperatures and heat exchanger effectiveness. The relations used for calculating effectiveness are described below.

2.1.3.1 Counterflow Heat Exchanger

Subroutine HXCNT calculates the heat exchanger effectiveness using the relation from Reference 3 for counterflow heat exchangers. That is,

\[
\epsilon = \frac{1 - e^{-\frac{UA(s)}{(MC)_s} \left\{ 1 - \frac{(MC)_s}{(MC)_l} \right\}}}{1 - \frac{(MC)_s}{(MC)_l} e^{-\frac{UA(s)}{(MC)_s} \left\{ 1 - \frac{(MC)_s}{(MC)_l} \right\}}}
\]

Where

- \( \epsilon \) = effectiveness
- \( UA \) = overall effectiveness
- \( (MC)_s \) = mass, specific heat product for the side with the smallest MC
- \( (MC)_l \) = mass, specific heat product for the side with the largest MC

The limiting cases for this relation are:

1. When \( (MC)_s / (MC)_l = 0 \),
   \[
   \epsilon = 1 - e^{-\frac{UA(s)}{(MC)_s}}
   \]
2. When \( (MC)_s / (MC)_l = 1 \),
   \[
   \epsilon = \frac{UA(s)}{1 + \frac{UA(s)}{(MC)_s}} = \frac{UA(s)}{(MC)_s + UA(s)}
   \]
Using the effectiveness as calculated by the above method, the outlet temperatures are calculated as follows:

1. For the side with the smallest MC, \((MC)_s\):

\[
T_{out_s} = T_{in_s} - \epsilon (T_{in_s} - T_{in_l}) \tag{9}
\]

2. The outlet temperature for the side with the large MC is then calculated by

\[
T_{out_l} = \frac{(MC)_s}{(MC)_l} (T_{in_s} - T_{out_s}) + T_{in_l} \tag{10}
\]

2.1.3.2 Parallel Flow Heat Exchanger

Subroutine HXPAR calculates the heat exchanger effectiveness using the relation for parallel flow heat exchangers which is:

\[
\epsilon = \frac{UA}{(MC)_s} \left[ \frac{1+(MC)_s}{(MC)_l} \right] \tag{11}
\]

The limiting cases are

1. When \((MC)_s/(MC)_l = 0,\)

\[
\epsilon = 1 - e^{-UA/(MC)_s}
\]

2. When \((MC)_s/(MC)_l = 1,\)

\[
\epsilon = \frac{-2 \cdot UA}{(MC)_s} \frac{1 - e}{2.0}
\]

The heat exchanger outlet temperatures are then calculated using equations 9 and 10.

2.1.3.3 Cross Flow Heat Exchanger

Subroutine HXCROS calculates the effectiveness for cross flow heat exchangers using one of the four relations below depending upon mixing of the streams.
Both Streams Unmixed

\[
\epsilon = 1 - e^{-1 - \frac{1}{\eta}}
\]  

Where \( \eta = \frac{(MC)_s}{(MC)_l} \) 0.22

Both Streams Mixed

\[
\epsilon = \frac{UA}{(MC)_s} - \frac{UA}{1-e} + \frac{UA}{1-e} 
\]  

Stream (MC)_s Unmixed

\[
\epsilon = 1 - e^{-1 - \frac{UA}{(MC)_s}} \left[ 1 - e^{-\frac{UA}{(MC)_l}} \right] 
\]  

Stream (MC)_l Unmixed

\[
\epsilon = 1 - e^{-\frac{UA}{(MC)_s}} \left[ 1 - e^{-\frac{UA}{(MC)_l}} \right] 
\]

The heat exchanger outlet temperatures are calculated using equations (9) and (10):

2.1.3.4 User Supplied Effectiveness

Subroutine HXEFF was written to perform heat exchanger thermal analysis with a user supplied effectiveness. The effectiveness may either be supplied as a constant or as an array number which gives the effectiveness as a bivariant function of the flowrates on the two sides. The outlet temperatures
are then calculated using equations (9) and (10).

2.1.4 Inline Heater Analysis

Provisions for the analysis of a fluid heater have been included in SINDA with subroutine HEATER. This subroutine simulates an electrical heater with a control system which turns the heater on when a specified sensor lump drops below a set value and turns the heater off when the specified sensor lump rises above another set value. When the heater is on an input quantity of heat is added to the heater node.

2.1.5 Cabin Analysis

A subroutine has been written for use with SINDA which will give the user the ability to perform thermal analyses on cabin air systems including condensation on the walls and a vapor mass balance. The cabin heat transfer and condensation analysis involves the two-component flow of a condensible vapor and a non-condensable gas, with condensation of the vapor occurring on surfaces in contact with the fluid. Two problems of this nature have been studied extensively.

1. Condensation on, or evaporation from, a surface over which a free stream of fluid is passing. In this case, for relatively low mass transfer rates, the fluid properties can be assumed to be constant.

2. Dehumidification of a confined fluid stream by a bank of tubes. In this case there is a marked change in the temperature and vapor content of the fluid, and the detailed deposition of the condensate is not of primary interest. This type of analysis is usually handled on an overall basis similar to heat exchanges effectiveness calculations.

The following additional assumptions have been made with respect to the cabin atmospheric conditions.

1. The heat of circulation in the cabin is sufficiently high that the temperature and humidity are effectively the same throughout the cabin.

2. The velocity at all points where heat transfer and/or condensation can occur is known, and is proportional to the total mass flow rate in the cabin.
These assumptions make it possible to calculate the heat and vapor balance in the cabin for the entire volume as a unit, and to solve the heat transfer and condensation equations at each node independently of the other nodes.

Cabin humidity can be determined from an overall vapor balance in the cabin. The total vapor in the cabin at the end of an iteration is:

\[ W_v = W_{v_{i-1}} + W_{v_{in}} - W_{v_{out}} - \sum W_L \]

Where
- \( W_{v_{i-1}} \) = mass of vapor in cabin at end of iteration \( i-1 \)
- \( W_v \) = mass of vapor in cabin at start of iteration \( i \)
- \( W_{v_{in}} \) = mass of vapor flowing into cabin during iteration \( i \)
- \( W_{v_{out}} \) = mass of vapor flowing out of cabin during iteration \( i \)
- \( \sum W_L \) = mass of vapor condensed during iteration \( i-1 \)

\( W_{v_{in}} \) is determined from the known conditions of the gas flowing into the cabin.

\[ W_{v_{in}} = m_{in} \left[ \frac{\psi_{in}}{1 + \psi_{in}} \right] \]

Where
- \( m_{in} \) = mass flow rate into cabin
- \( \psi_{in} \) = specific humidity of gas flowing into cabin
- \( t \) = time increment

It is assumed that an equal volume of gas is flowing out of the cabin. Then,

\[ W_{v_{out}} = \dot{m}_{out} \left[ \frac{\psi_c}{1 + \psi_c} \right] \]

Where
- \( \psi_c \) = specific humidity in the cabin (at the end of the previous iteration)

and

\[ \dot{m}_{out} = \dot{m}_{in} \left[ \rho_c / \rho_{in} \right] \]

Where
- \( \rho_c \) = cabin density
- \( \rho_{in} \) = density of gas flowing into cabin

The condensation term \( \sum W_L \) is determined from the calculations for the individual nodes as described below. The properties of the cabin atmosphere are determined from the calculated value of \( W_v \). The vapor pressure
in the cabin is

\[ P_v = \frac{W_v}{V_c} \frac{R_v}{T_c} \]

Where
- \( V_c \) = cabin volume
- \( R_v \) = gas constant
- \( T_c \) = temperature of cabin gas
- \( P_v \) = vapor pressure

Assuming that the cabin pressure \( P_c \) is a constant, the gas partial pressure \( P_a \) is:

\[ P_a = P_c - P_v \]

and

\[ W_a = \frac{P_a}{R_a T_c} \]

Where \( W_a \) = mass of non-condensible gas in the cabin.

Now the new value of specific humidity in the cabin can be determined by

\[ \psi_c = \frac{W_v}{W_a} \]

The properties of the atmosphere can now be determined by

\[ \mu_c = \frac{X \mu g + \psi_c \mu v}{X + \psi c} \]

\[ C_{pc} = \frac{C_{pg} + \psi c C_{pv}}{1 + \psi c} \]

\[ k_c = \frac{X \kappa g + \psi c \kappa v}{X + \psi c} \]

\[ \rho c = \frac{W_v + W_s}{V_c} \]

Where
- \( \mu \) = viscosity
- \( C_p \) = specific heat
- \( k \) = thermal conductivity
- \( X \) = molecular weight ratio, \( \frac{M_v}{M_g} \)
and all values are evaluated at \( T_{c_{i-1}} \). Cabin temperature \( T_c \) can be determined by a heat balance on the cabin atmosphere.

\[
T_c = T_{c_{i-1}} + \frac{\dot{m} \ln c_T (T_{in} - T_{c_{i-1}}) - \Sigma Q_L}{(W_v + W_A) C_p_c}
\]

Where
- \( T_{c_{i-1}} \) = \( T_c \) after previous iteration
- \( T_{in} \) = temperature of gas flowing into cabin
- \( \Sigma Q_L \) = net heat loss to cabin lumps

The heat transfer between the cabin atmosphere and the tube and structure lumps in the cabin is defined by:

\[
Q_{Li} = h A_Li [T_c - T_{Li}] \Delta r
\]

Where
- \( h \) = heat transfer coefficient
- \( A_Li \) = heat transfer area of lump
- \( T_{Li} \) = temperature of tube lump
- \( \Delta r \) = time increment

Using the Colburn-Chilton heat transfer-mass transfer analogy, the condensation (or evaporation) at the tube lump is determined by:

\[
\Delta W_{Li} = K_m A_Li (P_v - P_{wi}) \Delta r
\]

Where
- \( W_{Li} \) = condensation on wall, lb.
- \( K_m \) = mass transfer coefficient
- \( P_{wi} \) = vapor pressure at \( T_{Li} \)

The latent heat addition to the lump due to this condensation is

\[
\Delta Q_{\lambda} = \Delta W_{Li} \lambda
\]

Where \( \lambda \) = latent heat of vaporization

The vapor pressure \( P_{wi} \) can be determined by a relationship derived from the Clausius-Clapeyron equation and the perfect gas law (Appendix K of Reference 4).
The second option assumes flat plate flow for cabin wall lumps. In this case the heat transfer coefficient, for laminar flow, varies along the plate. Hence, direction of gas flow and the location of an assumed leading edge must be assumed. The equation for flat plates from Reference 3 is:

\[ N_u = 0.332 \frac{Re^{0.5}}{Pr^{1/3}} \]

where the Nusselt and Reynold's numbers are local values and are defined by the distance \( X \) from the assumed leading edge. For a wall lump of length \( L_1 \) which is located a distance \( L_{10} \) from the assumed leading edge, the

\[ P_{w1} = P_0 \left( 1 - e^{-\frac{\lambda}{R_g (T_1 - T_0)}} \right) \]

Where \( P_0 \) is known vapor pressure at a reference temperature \( T_0 \).

Three methods are available for determining mass and heat transfer coefficient. For tube lumps the equations from Reference 3 for gas flowing normal to the tube axis was assumed. Three different equations are used depending on the value of the Reynolds's number.

\[ \begin{align*}
    Nu &= 0.43 + 0.533 (Re)^{0.618} (Pr)^{0.31} \quad 4000 < Re < 40000 \\
    Nu &= 0.43 + 0.193 (Re)^{0.805} (Pr)^{0.31} \quad 40000 < Re < 400000
\end{align*} \]

These equations were derived for an air-vapor mixture, but should be relatively accurate for other similar gases. The Nusselt and Reynolds's numbers in the equations are defined using the tube diameter for the characteristic dimension, and the velocity in the Reynolds's number is input at each lump and ratioed to the total cabin atmosphere flow rate.

\[ \frac{\dot{W}_c}{W_{co}} = \frac{\dot{W}_c}{W_{co}} \]

Where \( \dot{W}_{co} \) = nominal cabin atmosphere circulation rate

\( \dot{W}_1 \) = velocity at lump at \( W_{co} \)

\( \dot{W}_c \) = circulation rate at time of calculation
average Nusselt number can be defined as:

\[ \text{Nu} = 0.664 \quad \text{Pr}^{1/3} \left[ (\text{Re}_1)^{0.5} - (\text{Re}_0)^{0.5} \right] \]

Where \( \text{Nu} \) is defined by \( L_i \),
\( \text{Re}_0 \) is defined by \( L_{i0} \),
\( \text{Re}_1 \) is defined by \( L_{i0} + L_i \)

The third option is a direct user input for convective heat transfer coefficient. For the determination of mass transfer coefficients, the same equations which were used for heat transfer coefficient can be used with the Sherwood number substituted for Nusselt number and Schmidt number for Prandtl number. However, if the diffusion coefficient for the cabin is approximately equal to thermal diffusivity, the Sherwood number is equal to the Nusselt number and the mass transfer coefficient can be determined directly from the heat transfer coefficient. That is:

\[ \text{Sh} = \text{Nu} \]

\[ \frac{K_m \rho C_p RT g x}{D} = \frac{h_x}{k} \]

If \( D \approx \alpha \) then

\[ K_m = \frac{h D}{\alpha \rho C_p RT g} \]

\[ K_m \approx \frac{h}{C_p P_c} \]

Equation (16) is the Lewis relationship (Reference 3). For a mixture of oxygen and water vapor characteristic values are .866 for the diffusion coefficient, \( D \), and .879 for thermal diffusivity, \( \alpha \), so the relationship should be valid.

For cabin tube and wall lumps the values for \( \Delta Q_{Li} \) and \( \Delta Q_{\lambda i} \) are added to the basic heat balance equation for these lumps. Values for \( \Delta Q_{Li} \)
are summed for all participating lumps for input to the cabin atmosphere heat balance. Values for $\Delta W_L$ are also summed for all lumps for cabin humidity balance, and the value for total water condensed on each lump $W_L$ is maintained.

If the rate of evaporation or condensation is high it would be possible for the cabin humidity to change significantly during a single iteration. This could lead, for example, to overestimating condensation by assuming that the humidity is constant in the calculation. A test of the approximate vapor pressure in the cabin at the end of the iteration is made, and the condensation or evaporation at any lump is reduced, if the sign of the $\Delta W_L$ term is changed. A value $W_V'$ is calculated by:

$$W_V' = W_{V-1} - \sum W_L$$

and

$$P_V' = \frac{W_V'}{144 V_c} R V T g$$

Then for each lump if

$$\frac{P_V' - P_{W_i}}{P_V - P_{W_i}} < 0$$

a new value of $\Delta W_L$ is calculated by:

$$\Delta W_L = \Delta W_L \left[ \frac{P_V - P_{W_i}}{P_V - P_V'} \right]$$

The new values of $\Delta W_L$ are now again summed for the new value of $\sum \Delta W_L$ for establishing cabin humidity for the next iteration. A test is also made to assure that $W_V'$ is never less than zero.

2.1.6 Radiation Interchange Analysis

Capabilities have been incorporated into subroutines for use with SINDA to facilitate the analysis of radiation heat transfer in an enclosure. The capabilities include the ability to:
(1) Analyze diffuse and/or specular infrared radiation in an enclosure
(2) Analyze diffuse and/or specular radiation from an external source for as many wave bands as desired
(3) Consolidate several temperature nodes into a single surface to improve computational efficiency

A radiation surface is defined as a group of temperature nodes which may be assumed to have identical radiating properties, angle factors and interchange factors.

The subroutines account for the net radiation heat transfer between a number of surfaces due to emitted radiation from each surface, reflected radiation from each surface, and radiation from any number of incident sources. The reflection of the energy originally emitted by another surface or from an external source may be either diffuse, specular, or any combination of the two.

2.1.6.1 Emitted Radiation In A Cavity

The radiosity of a surface is defined as the flux of infrared radiation leaving that surface with a diffuse distribution (according to Lambert's Law). That energy leaving a surface which has been reflected in a specular manner does not contribute to the radiosity of that surface. The incident infrared radiosity is denoted by the symbol H. The reflectance \((1 - \varepsilon)\) of a surface is separated into two components, the diffuse reflectance \((\rho)\), and the specular reflectance \((\rho^S)\). Here \(\varepsilon\) is the emittance of the surface and is equivalent to the absorptance for long wavelength radiation. With the angle factors \((F_{ij})\) defined in the normal way, there exist similar angle factors which relate the geometrical ability of surface i to radiate to surface j by means of a mirror-like reflection from specular surface k. Reference to Figure 1 indicates the method of imagery which will enable the calculation of these reflected angle factors. Here the angle factor to surface j is identical with the angle factor to the image of surface j. Also the angle factor is limited by the ability of surface i to "see" through the "window" of surface k. With the specular surface angle factors so defined, an interchange factor \(E_{ij}\) is defined similarly to reference 5 as follows:

\[
E_{ij} = \sum_k \rho_k^S F_{ij}(k) + \sum_k \sum_l (\rho_k^S) (\rho_l^S) F_{ij(k,l)} + \ldots \quad (17)
\]
**Figure 1** Illustration of method used to determine specular surface reflected view factors.
Here $F_{ij}(k)$ is the angle factor from $i$ to $j$ as seen in the specular surface
$k$, $F_{ij}(k,1)$ is the angle factor from $i$ to $j$ as seen in the double specular
reflection from $k$ and $1$. There are an infinite number of possible combinations
of these multi-reflections. It is evident that the interchange factors account for
the specularly reflected radiant flux from the reflecting surface. This
portion of total leaving flux is not a component of the radiosity of that
surface. The radiosity may be written

$$B_i = \epsilon_i \sigma T_i^4 + \rho_i H_i,$$

and, for $ns$ surfaces,

$$H_i = \frac{1}{At} \sum_{j=1}^{ns} B_j A_j E_{ji}$$

(19)

Now the interchange factors obey the reciprocity relation

$$A_i E_{ij} = A_j E_{ji}$$

(20)

So,

$$H_i = \sum_j B_j E_{ij}$$

(21)

Substitution into the equation for $B$ results in

$$\sum_j \left( \delta_{ij} - \rho_i E_{ij} \right) B_j = \epsilon_i \sigma T_i^4$$

(22)

This equation represents a set of linear, simultaneous, inhomogeneous
algebraic equations for the unknowns ($B_j$). The symbol $\delta_{ij}$ is the Kronecker
delta function which is 1 when $i = j$ and is 0 when $i \neq j$.

Note that the coefficients of $B_j$ in equation (22) do not form a
symmetric coefficient matrix since the off diagonal terms contain $-\rho_i E_{ij}$.
This equation can be made symmetric by multiplying each equation by $A_j/\rho_i$. 

17
This gives
\[ \sum_{j} \left( \frac{b_{ij} A_{i}}{\rho_{i}} - E_{ij} A_{i} \right) B_{i} = \frac{\epsilon_{i} A_{i}}{\rho_{i}} \sigma T_{i}^{4} \quad i = 1, ns \]  

\[ (23) \]

Written in matrix form this equation is
\[ E B = T \]  

\[ (24) \]

Where \( E \) is a symmetric coefficient matrix. The solution is
\[ B = E^{-1} T = \left[ e_{ij}^{-1} \right] T \]  

\[ (25) \]

or
\[ B_{i} = \sum_{j=1}^{ns} e_{ij}^{-1} \frac{\epsilon_{i} A_{i}}{\rho_{j}} \sigma T_{j}^{4} \]  

\[ (26) \]

The net heat transfer rate absorbed by surface \( i \) is given by
\[ Q_{i} = A_{i} \epsilon_{i} \left[ H_{i} - \sigma T_{i}^{4} \right] \]  

\[ (27) \]

Where \( H_{i} \) is given from equation (18) as
\[ H_{i} = \frac{1}{\rho_{i}} \left[ B_{i} - \epsilon_{i} \sigma T_{i}^{4} \right] \]

Substituting in for \( H_{i} \) gives
\[ Q_{i} = A_{i} \epsilon_{i} \left\{ \frac{1}{\rho_{i}} \left[ B_{i} - \epsilon_{i} \sigma T_{i}^{4} \right] - \sigma T_{i}^{4} \right\} \]

\[ = \frac{A_{i} \epsilon_{i}}{\rho_{i}} \left\{ B_{i} - \left[ \rho_{i} + \epsilon_{i} \right] \sigma T_{i}^{4} \right\} \]  

\[ (28) \]
Substituting in for $B_i$ from equation (26) into equation (28) gives

$$Q_i = \frac{A_i \varepsilon_i}{\rho_i} \left\{ \sum_{j=1}^{n_s} \frac{e_{ij}^{-1} \varepsilon_j A_j \sigma T_j^4}{\rho_j} \left[ \frac{\rho_i + \varepsilon_j}{\rho_i} \right] \sigma T_i^4 \right\}$$

$$= \frac{A_i \varepsilon_i}{\rho_i} \left\{ \sum_{j=1}^{n_s} \frac{e_{ij}^{-1} \varepsilon_j A_j}{\rho_j} \sigma T_j^4 \left[ \frac{\rho_i + \varepsilon_j}{\rho_i} \right] \sigma T_i^4 \right\} (29)$$

Since, in steady state, $Q_i = 0$, and $T_i^4 = T_j^4$ for all $i$ and $j$ we can conclude that

$$\rho_i + \varepsilon_i - \frac{e_{ij}^{-1} \varepsilon_j A_j}{\rho_i} = \sum_{j=1}^{n_s} \frac{e_{ij}^{-1} \varepsilon_j A_j}{\rho_j}$$

Making the above substitution in equation (29) gives

$$Q_i = \sum_{j=1}^{n_s} \sigma \frac{\varepsilon_i \varepsilon_j A_i A_j e_{ij}^{-1}}{\rho_i \rho_j} \left[ T_j^4 - T_i^4 \right] (30)$$

If we define $\mathcal{F}$ as

$$\mathcal{F}_{ij} = \frac{\varepsilon_i \varepsilon_j A_i A_j e_{ij}^{-1}}{\rho_i \rho_j} (31)$$

$$\mathcal{F}_{ij} = \frac{\varepsilon_i \varepsilon_j A_i A_j}{\rho_i \rho_j} \left[ e_{ij}^{-1} - \rho_i / A_i \right] (32)$$

Then

$$Q_i = \sum_{j=1}^{n_s} \sigma \mathcal{F}_{ij} A_i \left[ T_j^4 - T_i^4 \right] (32)$$

This equation gives the heat flux between surfaces. However, each surface can contain several nodes. The heat absorbed by for each node is determined by:
\[ Q_n = \frac{A_n}{A_i} \sum_{j=1}^{ns} \sigma \mathcal{F}_{ij} A_i \left[ T_j^4 - T_n^4 \right] \]  

(33)

Where \( n \) = the node number on surface \( i \)

Prior to each iteration, the temperature of the surfaces are determined by

\[ T_i^4 = \frac{\sum_{n=1}^{nn} A_n T_n^4}{\sum_{n=1}^{nn} A_n} = \frac{\sum_{n=1}^{nn} A_n T_n^4}{A_i} \]  

(34)

Where \( nn \) = the number of nodes on surface \( i \)

Since the heat transfer rate given by equation (33) depends on the node temperature, stability considerations must be taken into account. This is handled by storing the following relation into the array containing the sum of the conductors used for time increment calculation

\[ \text{CON}_n = 4 \frac{A_n}{A_i} \sigma T_n^3 \sum_{j=1}^{nc} \mathcal{F} A_{ij} \]  

(35)

Subroutine RADIR makes the calculations necessary to obtain \( Q_n \) given by equation (33) and \( \text{CON}_n \) given by equation (35). The following is a summary of the calculations:

A. The following are performed the first time through RADIR:
   1. From the user input values of \( E_{ij}, A_i, \) and \( p_i \), the \( E \) matrix given by equation (24) is formed. Only half of the symetric matrix is stored to save space.
   2. The \( E \) matrix is inverted in its own space to get \( E^{-1} \) with elements \( e_{ij}^{-1} \)
   3. The \( \mathcal{F} A_{ij} \) values are determined from equation (31) and stored in the surface connections data.

B. The following calculations are performed on each temperature iterations:
   1. The temperature of each surface is calculated by equation (34).
   2. The heat absorbed for each node is determined using equation (33) and is added to the \( Q \) array.
The routine utilizes data used for obtaining $\mathcal{F} A_{ij}$ in step A as working space for step B, thus, maximizing space utilization.

2.1.6.2 Radiation From External Source

As with the internally generated radiation, the solar (or any other external source radiation) interchange factor is defined by

$$E_{ij}^* = F_{ij} + \sum_k \rho_k^s \mathcal{F}_{ij}(k) + \sum_k \sum_l \rho_k^s \rho_l^s \mathcal{F}_{ij}(k, l) + \ldots$$

Where $\rho_k^s$ is the solar specular reflectance of surface $K$

$F_{ij}(k)$ is the angle factor from $i$ to $j$ as seen in the specular surface $K$.

$F_{ij}(K, l)$ is the angle factor from $i$ to $j$ as seen in a double specular reflection from $j$ to $l$ to $k$ back to $i$.

The interchange factors as defined above accounts for the specularly flux reflected from the surface. Thus, since the specular component of the flux is assumed to go directly from surface $i$ to surface $j$ by the interchange factor, $E_{ij}$, this portion of the total flux is not a component of the radiosity for the intermediate surfaces ($k$ and $l$ above). The radiosity of surface $i$ is given by

$$B_i^* = \rho_i^* H_i$$

Where $B_i^*$ is the radiosity (energy leaving)

$H_i^*$ is the incident energy

$\rho_i^*$ is the diffuse reflectance

The energy incident upon a surface is given by

$$H_i = \sum_{j=1}^{ns} B_j^* E_{ij}^* + S_i$$

Where $S_i$ is the energy directly incident on surface $i$ from an external source.
Substituting equation (36) into (37), multiplying by \( \frac{A_i}{\rho^*_i} \) and simplifying gives the following relation for the radiosity

\[
\begin{bmatrix}
\frac{A_i}{\rho^*_i} - E^*_i A_i
\end{bmatrix} B^*_i = \sum_{j=1}^{n} E^*_i A_i B^*_j = S_i A_i \quad i=1,n
\]

(38)

This set of \( n \) equations can be written in matrix form as

\[
E^* B^* = S
\]

(39)

Note that the equations are written so that \( E^* \) is a symmetric matrix, which has the solution for \( B^* \)

\[
B^* = E^{*-1} S \quad \text{or} \quad B_i = \sum_{j=1}^{n} [e^*_{ij}]^{-1} S_j A_j
\]

(40)

Where \([e^*_{ij}]^{-1}\) is the \( ij \)th element of the inverse of the \( E^* \) matrix.

The heat flux absorbed by the \( i \)th surface is given by

\[
\frac{Q^*_i}{A_i} = \alpha H_i
\]

(41)

But from equation (36)

\[
H_i = \frac{B_i}{\rho^*_i}
\]

(42)

Combining equations (40), (41), and (42) gives

\[
Q^*_i = \sum_{j=1}^{n} e^*_{ij}^{-1} \frac{\alpha_i}{A_i} A_j S_j
\]

(43)

If we define

\[
\sigma^*_{ij} = e^*_{ij}^{-1} \frac{\alpha_i}{A_i} A_j
\]

(44)

Then the absorbed heat flux is given by

\[
Q^*_i = \sum_{j=1}^{n} \sigma^*_{ij} A_i S_j
\]

(45)
Equation (45) gives the heat absorbed by each surface. However, each surface may contain several temperature nodes. The absorbed heat for each node is given by:

\[
Q_n^* = \frac{A_n}{A_i} Q_i^*
\]  

(46)

Where \( A_n \) is the area of the node.

Subroutine RADSOL was written to make necessary calculations to obtain \( Q_n^* \) given by equation (46). The following is a summary of the calculations:

A. The following calculations are made the first time through RADSOL:

1. From the user input values of \( E_{ij}^* \), \( \rho_i^* \), and \( A_i \), the \( E^* \) matrix given by equation (39) is formed. Only one half is stored since \( E^* \) is symmetric.
2. The \( E^* \) matrix is inverted in its own space to get \( E^{*\text{-}1} \) with elements, \( e_{ij} \).
3. The \( \mathcal{F}_{ij}^* A_i \) values are determined from equation (44) and stored in the surface connections data.

B. The following calculations are performed on each temperature iteration:

1. The heat flux absorbed by each node is calculated by

\[
Q_i^* = \frac{1}{A_i} \sum_{J=1}^{n} \mathcal{F}_{ij}^* A_i S_i
\]

2. The net heat absorbed by this wavelength radiation is calculated for each temperature node on each surface by

\[
Q_n^* = A_n \frac{Q_i^*}{A_i}
\]

This quantity of absorbed heat is added to the \( Q \) array for node \( n \).
Note that the user may specify subroutine RADSOL for an many bands of radiation from an external source as desired. A single call is required for each band.
2.2 Fluid Flow Analysis

Subroutine PFCS was written as a SINDA user subroutine to provide the ability to perform fluid pressure/flow analysis for flow of an incompressible fluid in tubes. The fluid flow analysis of PFCS is integrated with the thermal analysis capability so that the temperature dependence of properties is included in the pressure balances. PFCS is called from the VARIABLES 2 user logic block.

PFCS performs a pressure-flow balance on a general flow network including the following effects:

1. Friction pressure drop
2. Orifices and fitting type pressure losses
3. Valves
4. Pumps
5. Incoming flow sources at any pressure point in the system

The user describes the flow model to the subroutine by supplying the tube network connections and information concerning fluid properties, flow geometry, temperature model lumps, orifices, valves and pumps. Using this information, the subroutine determines the flow distribution required to satisfy (1) the conservation of mass at each node point and (2) equal pressure drops across tubes in parallel. The model used to describe the flow system and the analytical methods for determining the solution are described below.

2.2.1 Overall Flow Model Description

A flow problem may be analyzed with PFCS, simultaneously with a thermal analysis, so that the flow solution is continually updated based on the thermal conditions. To perform a flow analysis, the user must input a mathematical model of the flow system. The flow system is assumed to consist of a set of interconnected tubes such as the example shown in Figure 2 which consists of two radiator panels, each containing four tubes and connected so that they flow in parallel.

For clarity the following definitions are made at this point:
FIGURE 2 FLOW SYSTEM SCHEMATIC

Tube Numbers

Pressure Nodes

FIGURE 2 FLOW SYSTEM SCHEMATIC
(1) A tube is any single length of pipe between two pressure nodes. A tube "contains" fluid temperature nodes and may contain as many of these as required.

(2) A pressure node is located at each end of a tube. As many tubes as desired may be connected at a node junction and a node must exist at the junctions of two flow pipes.

We must make a mathematical model to describe the fluid flow information to the computer. The information required consists of:

(1) Identification of the pressure node numbers
(2) Identification of the tube numbers and the two pressure nodes connected by tube
(3) The fluid temperature nodes contained in each tube
(4) The flow geometry for each temperature fluid nodes
(5) The number of "head losses" for items such as orifices
(6) Fluid property information
(7) Valve connections and characteristics
(8) Pump characteristics

To build a flow mathematical model, a schematic of the flow system is needed. As shown in Figure 2, the pressure nodes and tubes may be superimposed on the schematic. It is also helpful to impose the fluid temperature lump numbers for each tube.

To facilitate speedy analysis on a general flow problem, provisions have been made for the user to divide the flow system network into subnetwork elements. For example, the flow system shown in Figure 2 could be divided as shown in Figure 3. Tubes 23 and 24 are added in the main network as shown in 3(a) to replace subnetwork elements 1 and 2. The subnetwork elements 1 and 2 which are shown in Figures 3(b) and 3(c) are then input as separate network elements. This type of subdivision allows the solution to be obtained by solving two sets of 6 simultaneous equations and one set of 8 equations rather than the original set of 16 simultaneous equations. This type of subdivision has been found to enhance the solution speed and accuracy for problems with a large number of nodes.

In summary, the pressure/flow solution is obtained by the following sequence:
a) Main Flow Network

b) Subnetwork No. 1

c) Subnetwork No. 2

FIGURE 3 MAIN NETWORK AND SUBNETWORKS

28
(1) The flow resistance is obtained for each fluid temperature lump in each tube including the effects of friction, orifices, and fitting type losses.

(2) The flow conductor valve is obtained for each tube by summing all the resistances of the fluid lumps in the tube, adding the valve and user supplied resistance to the sum, and inverting the resistance.

(3) A set of simultaneous equations is set-up and solved for each main system and subnetwork to obtain the pressures.

(4) The flow rates are then calculated.

A detail discussion of each element in the above sequence is described in the following subsections.

2.2.2 Tube Conductor Determination

The value of the flow conductor is determined for each tube by first calculating the flow resistance for each temperature fluid lump contained in the tube, summing these resistances up to obtain the flow resistance of the tube and inverting the tube resistance to get the conductance. Flow conductance is defined by the relationship

\[ \dot{w}_{ij} = GF_{ij} [P_i - P_j] \]

Where \( \dot{w}_{ij} \) = flow rate between pressure nodes \( i \) and \( j \)
\( GF_{ij} \) = flow conductance between nodes \( i \) and \( j \)
\( P_i \) = pressure at pressure node \( i \)
\( P_j \) = pressure at pressure node \( j \)

The flow resistance for each lump is then

\[ R_k = \frac{1}{GF} = \frac{\Delta P_k}{\dot{w}} \]

Where \( R_k \) = flow resistance for lump \( k \)
\( \Delta P_k \) = pressure drop for lump \( k \)

But \( \Delta P_k \) is given by

\[ \Delta P_k = \left( f_k \cdot ffc \cdot \frac{L_k}{D_k} + K \right) \frac{W^2}{2g_c \rho_k A^2} \]
Where \( f_k \) = the friction factor for lump \( k \)
\( \text{ffc} \) = the friction factor coefficient
\( L_k \) = the lump length for lump \( k \)
\( D \) = the lump hydraulic diameter for lump \( k \)
\( K \) = the dynamic head losses for lump \( k \)
\( \dot{W} \) = the flow rate
\( g_c \) = the gravitational constant
\( \rho_k \) = the fluid density for lump \( k \)
\( A \) = the flow area

The flow resistance is then given by

\[
R_k = \left( f_k \text{ffc} \frac{L_k}{D_k} + K \right) \frac{\dot{W}}{2 g_c \rho_k A^2}
\]  

(49)

Two options are available for obtaining the friction factor, \( f_k \). These are
(1) internal calculations for all flow regimes and (2) internal calculation for laminar flow and obtained from a table of \( f \) vs \( \text{Re} \) (where \( \text{Re} \) is the Reynold's number) for transition and turbulent flow. For the first option the internal calculations for the three flow regimes are:

**Laminar Regime:** \( \text{Re}_k \leq 2000 \).

\[
f_k = \frac{64}{\text{Re}_k}
\]  

(50)

Where \( f_k \) = friction factor for lump \( k \)
\( \text{Re}_k \) = Reynolds number for lump \( k \)

**Transition Regime:** \( 2000 < \text{Re}_k < 4000 \)

\[
f_k = 0.2086082052 - 0.1868265324 \left( \frac{\text{Re}_k}{1000} \right) \\
+ 0.06236703785 \left( \frac{\text{Re}_k}{1000} \right)^2 - 0.0065545818 \left( \frac{\text{Re}_k}{1000} \right)^3
\]  

(51)
Turbulent Regime: \( \text{Re}_k \geq 4000 \)

\[
f_k = \frac{316}{(\text{Re}_k)^{.25}} \tag{52}
\]

Equation (51) for the transition regime is a curve fit between the laminar and turbulent regimes which was derived to match the two curves in a continuous manner. It is merely an arbitrary curve in this undefined region. A curve of the friction factor vs Reynold's number given by the above relations is shown in Figure 4.

The second option for friction factor uses equation (50) for the laminar regime and a user input curve of \( f_k \) vs \( \text{Re} \) for the other regimes. The options available for input of the dynamic head loss, \( K \), include (1) an input constant or (2) a tabulated curve of \( K \) vs \( \text{Re} \).

To obtain the conductance for each tube, the flow resistances for all the lumps in the tube are added and then inverted, giving

\[
G_{F1j} = \frac{1}{\sum_k R_k} \tag{53}
\]

### 2.2.3 Valve Analysis

Provisions have been included in subroutine 'PFCS for valves to be included in the flow balance. The valve pressure drop is characterized by the following equation for each side of the valve;

\[
\Delta P = E \left( \frac{\dot{\omega}}{X} \right)^2
\]

where \( \Delta P \) = valve pressure drop  
\( E \) = valve pressure drop factor (user input)  
\( \dot{\omega} \) = flowrate through the side of the valve under consideration  
\( X \) = the fraction of the valve opening ( \( X = 0 \) indicates valve closed; \( X = 1.0 \) indicates valve full open)
FIGURE 4  Friction Factor vs Reynolds Number

Re \times 10^{-3} \text{ (curve 1)}

Re \times 10^{-4} \text{ (curve 2)}
Three basic types of valves are available in PFCS which give different characteristics for the dynamics of the valve position $X$. These types are:

1. Rate Limited
2. Polynomial
3. Shut-off

A number of variations are available for each valve type. For instance, each of the above may be either one sided or two sided. If a valve is two sided, the valve position of side 2, $X_2$, is related to that of side one by

$$X_2 = 1.0 - X_1$$

If the valve is one sided, either side one or side two may be used. Provisions are included for a valve time constant to be included with the polynomial valve.

The methods used to obtain the valve positions for each of the three methods are described below.

2.2.3.1 Rate Limited Valve

The valve position for the rate limited valve is obtained by an approximate integration of the valve rate of movement, $\dot{X}$. $\dot{X}$ depends on the temperature difference between the valve control set point temperature and the sensor temperature as shown in Figure 5. With this characteristic, the valve has no movement as long as the valve temperature error, $\Delta T$, is within the dead band. Outside the dead band, the velocity of the valve increases linerarly as the error increases to a maximum rate, $\dot{X}_{\text{max}}$. The dead band, rate of velocity increase, $d\dot{X}/d(\Delta T)$, and the maximum velocity are controlled by user input.

The relations used to obtain the valve positions are as follows:

$$X_{i+1} = X_i + (\dot{X}_{i+1}) (\Delta r)$$  \hspace{1cm} (54)

Where

$X_{i+1}$ = valve position at iteration $i+1$
$X_i$ = valve position at iteration $i$
$\dot{X}_{i+1}$ = valve velocity at iteration $i+1$
$\Delta r$ = the problem time increment
Valve Position, $X = \dot{X}_0 + \dot{X}$ (Time Increment) $0 \leq x < 1.0$

Figure 5  Rate Limited Valve Operation
The valve position is limited by

\[ X_{\text{min}} \leq X^{i+1} \leq X_{\text{max}} \]

Where \( X_{\text{min}} \) and \( X_{\text{max}} \) are input limits on the valve position.

The valve velocity, \( \dot{X}^{i+1} \), in equation (54) is given by:

\[ \dot{X}^{i+1} = 0 \text{ if } |T_{\text{sen}} - T_{\text{set}}| \leq T_{\text{db}} \]

Where

- \( T_{\text{sen}} \) = Sensor lump temperature
- \( T_{\text{set}} \) = Set point temperature
- \( T_{\text{db}} \) = Valve dead band temperature

\[ \dot{X}^{i+1} = \frac{d}{dT} [T_{\text{sen}} - T_{\text{set}} - T_{\text{db}}] \text{ if } T_{\text{sen}} > T_{\text{set}} + T_{\text{db}} \]

\[ \dot{X}^{i+1} = \frac{d}{dT} [T_{\text{sen}} - T_{\text{set}} + T_{\text{db}}] \text{ if } T_{\text{sen}} < T_{\text{set}} - T_{\text{db}} \]

The valve velocity is limited by

\[ \dot{X}_{\text{min}} \leq \dot{X}^{i+1} \leq \dot{X}_{\text{max}} \]

After the valve position for side 1 is obtained from equation (54), the side 2 position is obtained from \( X_{2} = 1.0 - X_{1} \)

2.2.3.2 Polynomial Valve

The polynomial valve determines the steady state valve position as a 4th degree polynomial function of the temperature error between the sensor lump and the set point. A valve time constant is then applied to determine how far between the previous position and the new steady state position the valve will move. The steady state position, \( X_{SS} \), is given by

\[ X_{SS} = A_{0} + A_{1} \Delta T + A_{2} \Delta T^{2} + A_{3} \Delta T^{3} + A_{4} \Delta T^{4} \]

Where
- \( \Delta T = T_{\text{sen}} - T_{\text{set}} \)
- \( T_{\text{sen}} \) = the sensor lump temperature
- \( T_{\text{set}} \) = the set point temperature
- \( A_{0}, A_{1}, A_{2}, A_{3}, A_{4} \) = input constants

35
The valve position, $X_{i+1}$, is then determined by

$$X_{i+1} = X_{ss} + (X_i - X_{ss}) e^{-\Delta r/r_c}$$

(55)

Where $X_{i+1}$ = valve position at iteration $i+1$

$X_i$ = valve position at iteration $i$

$\Delta r$ = problem time increment

$r_c$ = valve time constant

The valve position for side 2 is given by

$$X_2 = 1.0 - X_1$$

where $X_1$ is given by equation (55)

Note that this valve combines the capabilities of the polynomial valve and the proportioning valve described in Reference 6. If one desires to eliminate the effect of the time constant (and thus, give the valve an instantaneous response), a value for $r_c$ should be input which is small compared to the time increment, $\Delta r$. Also, either a constant value or a temperature lump number may be specified for the set point to permit use of the valve for proportioning between two sides.

2.2.3.3 Shut-off Valve

For side 1 of a shut-off valve the valve position decreases from $X_{\text{max}}$ to $X_{\text{min}}$ when the temperature of the sensor lump drops below the specified "off" temperature, $T_{\text{off}}$, and increased from $X_{\text{min}}$ to $X_{\text{max}}$ when the sensor lump exceeds a second specified temperature, $T_{\text{on}}$. $T_{\text{on}}$ must be greater than $T_{\text{off}}$. Side 2 works in reverse of side 1. The valve position increased from $X_{\text{min}}$ to $X_{\text{max}}$ when the sensor temperature drops below the specified $T_{\text{on}}$ and decreases from $X_{\text{max}}$ to $X_{\text{min}}$ when the sensor lump increases above the off temperature, $T_{\text{off}}$. For side 2, $T_{\text{off}}$ must be greater than $T_{\text{on}}$. Note that, if the shut-off valve is a two sided valve with both sides active, the valve is a switching valve.

2.2.3.4 Valve Flow Resistance Calculations

The valve pressure drop on side one is assumed to be given by:
\[ \Delta P = E \left( \frac{\dot{W}}{X} \right)^2 \]  

(56)

Where \( E \) is an input constant

\( \dot{W} \) is the flow through one side of the valve

\( X \) is the valve position (fraction of total possible distance)

Since flow resistance is \( \Delta P/\dot{W} \), the valve flow resistance is given by

\[ R_v = \frac{E \dot{W}}{X^2} \]  

(57)

This value of flow resistance is calculated and added to the other flow resistances of the tube prior to performing the operation in equation (53) to find the value of the flow conductor for the tube.

Valves may be either one way or two way - i.e., be one tube or two tubes at the outlet. If only one tube exists on the valve outlet the flow resistance is calculated using equation (57) above. If a second tube exists, the resistance on side 2 is given by

\[ R_{v2} = \frac{E_2 \dot{W}_2}{(1 - X)^2} \]  

(58)

2.2.4 Pressure-Flow Network Solution

As previously stated, the user may subdivide a system flow network into a main network and subnetwork elements. The elements which are subnetworks to the main network may also contain subnetwork elements but the subdivision can go no lower than two levels.

After the flow conductor values have been obtained by the methods described in Sections 2.2.2 and 2.2.3 a set of simultaneous equations are set up and solved for the main system and for each subnetwork. The subnetwork elements are all solved first and then, their equivalent flow conductor value is calculated. The value is inserted in the main system network and the system solution is obtained. The procedure is repeated until the problem is balanced.

A set of simultaneous equations are obtained by conservation of mass at each pressure node for each network and subnetwork. For any node \( i \) the conservation equation can be written as follows:
\[ \sum W_{\text{out}} - \sum W_{\text{in}} = 0 \quad (59) \]

Let \( W_{\text{in}} = W_1 \)

and \( \sum W_{\text{out}} = \sum_{j=1}^{nc} G F_{ij} [ P_j - P_i ] \)

Then equation (59) becomes

\[ \sum_{j=1}^{n} G F_{ij} [ P_j - P_i ] - W_1 = 0 \quad i=1,n \quad (60) \]

Where \( G F_{ij} \) = flow conductor between pressure nodes \( i \) and \( j \)

\( P_i \) = pressure at node \( i \)

\( P_j \) = pressure at node \( j \)

\( W_i \) = flow rate added at node \( i \)

\( n \) = number of pressure nodes in the subnetwork

The above equation is a set of \( n \) simultaneous equations for \( P \) array.

Pressure in the system or subsystem may be set at a specified level but the last (outlet) node must be specified. Equation (60) may be written in matrix form as:

\[ GP = C \quad (61) \]

Where

\[ G = \begin{bmatrix}
\sum G F_{ij} & -G F_{12} & -G F_{13} \\
-G F_{21} & \sum G F_{2j} & -G F_{23} \\
& & \\
& & \\
& & \\
-G F_{n-1,1} & -G F_{n-1,2} & -G F_{n-1,j}
\end{bmatrix} \]
\[
P = \begin{bmatrix}
P_1 \\
P_2 \\
\vdots \\
\vdots \\
P_{n-1}
\end{bmatrix}
\]

\[
C = \begin{bmatrix}
W_1 + GF_{1n} P_n \\
W_2 + GF_{2n} P_n \\
\vdots \\
\vdots \\
W_{n-1} + GF_{n-1n} P_n
\end{bmatrix}
\]

\(P_n\) is the specified pressure. The above equations are solved for pressures at each point in the system and flow rates are then calculated for each tube by:

\[
\dot{W}_{ij} = GF_{ij} (P_i - P_j) \quad (62)
\]

Since the coefficient matrix given by equation (61) is symmetric and positive definite the efficient square root or Symmetric Cholesky method was programmed to obtain the solution. This method is more accurate and faster than any other methods studied for this application.

Since the flow conductors are functions of the flow rate, the set of equations given by (61) are solved numerous times on each temperature iteration with a net set of GF\(_{ij}\) values for each solution. The iteration
process continues until the change in the flowrates is within some user specified tolerance before proceeding to the next iteration.

2.2.5 Pump and System Pressure - Flow Matching

Concurrent with iterating the system flow equation to solution on each temperature iteration, the overall system pressure drop and flowrate must be matched to a pump characteristic. Several types of pump characteristics are available to the user as options. These are (1) system flow rate specified as a constant, (2) system flowrate specified as a known function of time, (3) pressure drop specified as a function of the flowrate in a tabulated form and (4) pressure drop specified as a function of flowrate with a fourth degree polynomial curve.

The first two options require no balancing of the pump with the system. Balancing is required for options (3) and (4) and iterative procedures have been devised to obtain the solution of the pump curve to the system characteristics with as few passes as possible through the system pressure/flow balancing loop for these options. The procedures used for these options are described below.

2.2.5.1 Tabulated Pump Curve Solution

The matching of a tabulated pump pressure rise/flow characteristic to the system pressure drop/flow characteristic is accomplished by the following procedure. See Figure 6 to aid in understanding the procedure.

Step 1: The initial flowrate, $W_1$, at the system inlet is established either from user input on the first iteration or the system flow of the previous iteration for subsequent iterations.

Step 2: Using $W_1$, a solution to the flow network is obtained using the methods described in Sections 2.2.2, 2.2.3 and 2.2.4. Following this solution, $\Delta P_1$ is available establishing point 1 on the true system characteristic curve shown in Figure 6.

Step 3: Obtain an equation for the straight line approximation of the system characteristic (line 0, 1 for the first pass, line 1, 2 for the second pass, etc.)

$$\Delta P_S = C W_S + D$$
FIGURE 6 SYSTEM/PUMP CURVE SOLUTION
where \( C = \frac{\Delta P_1 - \Delta P_0}{W_1 - W_0} \)

\[ D = \Delta P_0 - \frac{\Delta P_1 - \Delta P_0}{W_1 - W_0} W_0 \]

\( \Delta P_S, W_S \) are the system pressure drop and flowrate values given by the approximate equation

\( \Delta P_1, W_1 \) are the latest values for system pressure drop and corresponding system flowrate

\( \Delta P_0, W_0 \) are the values for system pressure drop and corresponding system flowrate for the previous pass (These values are zero for the first pass)

Step 4: Obtain the equation of the line connecting points \( a_1 \) and \( b_1 \) which is an approximation of the pump characteristic.

(1) Two points are determined on the pump characteristic curve:

(a) Interpolate the tabulated characteristic at \( W_1 \) to obtain \( \Delta P_{al} \) (See Figure 6) to locate point \( a_1 \) at \( W_1, \Delta P_{al} \). If \( W_1 \) is greater than \( W_{max} \), set \( W_1 \) equal to \( W_{max} \) and \( \Delta P_{al} \) equal to zero.

(b) Reverse interpolate the tabulated characteristic at \( \Delta P_{1} \) to obtain \( W_{bl} \) to locate point \( b_1 \) on the curve. If \( \Delta P_{1} \) is greater than \( \Delta P_{max} \), \( \Delta P_{1} \) is set to \( \Delta P_{max} \) and \( W_{bl} \) is set to zero.

(2) Determine the coefficients \( A \) and \( B \) for the equation

\[ \Delta P_p = AW_p + B \]

where \( A = \frac{\Delta P_{1} - \Delta P_{al}}{W_{bl} - W_1} \)

\[ B = \Delta P_{al} - \frac{\Delta P_{1} - \Delta P_{al}}{W_{bl} - W_1} \]
\( \Delta P_p, W_p \) are the pump pressure rise and flowrate as given by the approximation.

Step 5: Solve the approximate equations obtained in Steps 3 and 4 to obtain an approximate solution to the system characteristic and the pump characteristic (Point N) as follows:

\[
W_N = \frac{D - B}{A - C}
\]

\( \Delta P_N = AW_3 + B \)

Step 6: Check the tolerance below where \( W_{N-1} \) is the previous \( W_N \) (\( W_1 \) for the first time through):

\[
\frac{W_N - W_{N-1}}{W_{N-1}} < 0.01
\]

(1) If the above inequality equation is not satisfied repeat steps 4 through 6 substituting \( W_N \) for \( W_1 \) and \( \Delta P_N \) for \( \Delta P_1 \)

(2) If the inequality is satisfied the point \( S_1 \) (Figure 6) has been located. Continue with step 7. The final flowrate is \( W_2 \).

Step 7: Check the following tolerance

\[
\frac{W_2 - W_1}{W_1} < TOL^*
\]

(1) If the above inequality equation is not satisfied, repeat steps 2 through 7 using the value of \( W_2 \) for \( W_1 \).

(2) If the inequality is satisfied, \( W_2 \) is the solution flowrate.

* \( TOL \) is the input pressure solution tolerance described on page 74.
2.2.5.2 Polynomial Pump Curve Solution

When the user describes the pump curve with a polynomial curve fit, the pump characteristic is described by the relation

$$\Delta P_p = A_0 + A_1 W + A_2 W^2 + A_3 W^3 + A_4 W^4$$

When this option is used, the procedure for matching the pump characteristic to the system characteristic is identical to that described in Section 2.2.5.1 for the tabulated pump characteristic except Steps 4 and 5 are replaced with the following:

Step 4: Obtain the coefficients of the 4th order equation to be solved.

Since:

$$\Delta P_p - \Delta P_s = 0$$

$$\Delta P_s = C W_s + D$$ (C and D are obtained from Step 3)

$$\Delta P_p = A_0 + A_1 W_p + A_2 W_p^2 + A_3 W_p^3 + A_4 W_p^4$$

The solution occurs when

$$\Delta P_s = \Delta P_p$$

Then the equation for \( \dot{W}_N \) is

$$(A_0-D) + (A_1-C) \dot{W}_N + A_2 \dot{W}_N^2 + A_3 \dot{W}_N^3 + A_4 \dot{W}_N^4 = 0$$

Step 5: Solve the equation for \( \dot{W}_N \) using the Newton-Raphson Method of solution for a fourth order polynomial.

The remaining steps are identical to that given in Section 2.2.5.1.
3.0 MODIFICATIONS TO SINDA SUBROUTINES

3.1 Preprocessor Modifications

Subroutine IMBED was written to convert actual conductor numbers, node numbers, array numbers or constant numbers which are input in the array data to their relative location in the G, T, A or K arrays respectively. The number to be converted in the array data is entered with an * followed by G, T, A, or K depending on the type of relative location desired. For instance, an array with the following input

```
```

would be changed so that the location of A10 in the A array would replace *A10 in the A12+1 location. The relative node number of actual node 5 would replace the *T5 and the relative conductor number of actual conductor number 101 would be placed in A12+3. The converted array might read

```
12, 102, 3, 21, END
```

Where A10 is located at location 102 in the A array, actual node 5 has relative number 3, and actual conductor 101 has relative conductor 21.

Subroutine IMBED is called from CODERD. Listings of IMBED, CODERD, and a modified overlay are supplied in Appendix A.

3.2 User Subroutine Modifications

A modification to the SINDA execution subroutine, CNFAST, was required to make it compatible with the radiant interchange subroutine, RADIR. This minor modification was required because the manner in which temperatures were calculated when the problem time increment is larger than the maximum convergence criterion for a given node was not compatible with the manner in which the convergence criteria information was carried over to CNFAST. A listing of the modified version of CNFAST is supplied in Appendix A.

A modification was made to user subroutine TPRINT to print temperatures in increasing order of actual node numbers. This modification increased the dynamic storage requirement to two locations for each node in the network.
4.0 USER SUBROUTINES

The capabilities described in Section 2.0 are available to the SINDA user through user subroutines which were added to the existing SINDA library. This section presents a description and user input requirements for each subroutine. Table 1 summarizes the subroutines and the page that each user description is found.

The subroutine inputs rely heavily upon the capability to convert from actual array, constants, node, and conductor numbers to relative numbers in the array data. To use this capability the user may supply an actual array number, node number or conductor number by preceding the actual number with *A, *K, *T, or *G respectively. This causes the preprocessor to replace the entry with the relative number. Consider the example for array number 2 shown below.


In this example, following the preprocessor phase, *A14 will be replaced by the location in the A array of the Array No. 14 data, *T5 will be replaced by the relative node number for actual node No. 5, and *G7 will be replaced by the relative conductor number for actual conductor No. 7. This feature is used extensively for the input to user subroutines described below and is described in more detail in Section 3.1.
### TABLE 1
NEW SINDA USER SUBROUTINES

<table>
<thead>
<tr>
<th>Category</th>
<th>Subroutine</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>A. Convection Conductor Calculation Subroutines</td>
<td>CONV1</td>
<td>48</td>
</tr>
<tr>
<td></td>
<td>CONV2</td>
<td>50</td>
</tr>
<tr>
<td></td>
<td>CONV3</td>
<td>52</td>
</tr>
<tr>
<td>B. Flow Conductor Calculation Subroutines</td>
<td>FLOCN1</td>
<td>53</td>
</tr>
<tr>
<td></td>
<td>FLOCN2</td>
<td>53</td>
</tr>
<tr>
<td>C. Time Variant Conductor Calculation Subroutines</td>
<td>CONDT1</td>
<td>55</td>
</tr>
<tr>
<td></td>
<td>CONDT2</td>
<td>55</td>
</tr>
<tr>
<td>D. Enclosure Radiation Analysis Subroutines</td>
<td>RADIR</td>
<td>57</td>
</tr>
<tr>
<td></td>
<td>RADSOL</td>
<td>59</td>
</tr>
<tr>
<td>E. Heat Exchanger and Heater Analysis Subroutines</td>
<td>HXEFF</td>
<td>61</td>
</tr>
<tr>
<td></td>
<td>HXCNT</td>
<td>62</td>
</tr>
<tr>
<td></td>
<td>HXCROS</td>
<td>63</td>
</tr>
<tr>
<td></td>
<td>HXPAR</td>
<td>65</td>
</tr>
<tr>
<td></td>
<td>HEATER</td>
<td>66</td>
</tr>
<tr>
<td>G. Cabin Analysis Subroutines</td>
<td>CABIN</td>
<td>67</td>
</tr>
<tr>
<td>H. Pressure/Flow Balance Subroutine</td>
<td>PFCS</td>
<td>71</td>
</tr>
<tr>
<td>I. Input/Output Subroutines</td>
<td>HSTRY</td>
<td>83</td>
</tr>
<tr>
<td></td>
<td>NEWTMP</td>
<td>85</td>
</tr>
<tr>
<td></td>
<td>FLPRNT</td>
<td>86</td>
</tr>
<tr>
<td></td>
<td>GENOUT</td>
<td>86</td>
</tr>
<tr>
<td></td>
<td>GENI</td>
<td>86</td>
</tr>
<tr>
<td></td>
<td>GENR</td>
<td>86</td>
</tr>
<tr>
<td></td>
<td>FLUX</td>
<td>87</td>
</tr>
<tr>
<td>J. Elapsed Computer Time Comparison and Termination Routine</td>
<td>TIMCHK</td>
<td>89</td>
</tr>
<tr>
<td>K. Interpolation</td>
<td>REVPO</td>
<td>90</td>
</tr>
</tbody>
</table>
SUBROUTINE NAME: CONV1

PURPOSE:

CONV1 calculates conductor values using the relations for convection heat transfer for flow in a tube. The relations used to obtain the film coefficient, \( h \), are given by equations 2 thru 4 in section 2.1.1, depending on the flow regime (laminar, transitional or turbulent flow). \( h \) is then multiplied by the input area for heat transfer to obtain the conductor value. The conductor value is stored in the input conductor location. Any number of conductors may be calculated with a single call. The flowrate array and the fluid properties data are addressed to tie the convection calculations with the pressure-flow solution. The first argument, AFLOW, is the first argument in the PFCS routine and identifies the flowrate array and fluid type data array needed by CONV1. APR is an array which references the specific heat, density, viscosity and thermal conductivity array. ADAT supplies other information needed.

RESTRICTIONS:

Should be called in the VARIABLES 1 block so that \( hA \) values are obtained every iteration.

CALLING SEQUENCE:

CONV1 (AFLOW, APR, ADAT)

where:

AFLOW is the first argument of the PFCS call and is of the following format:

AFLOW(IC), AW, AP, AGF, AVP, AIFR, AFT, AFR, APD, END

AW - array of flowrates per tube in the system
AP - array of pressures per pressure node in the system
AGF - array of pressure conductors per tube
AVP - array of valve positions for all valves in the system
AIFR - array of imposed flowrate per node
AFT - array of fluid type data
AFR - array of user added flow resistance per tube
APD - array for output of pressure drops per tube
APR - is the second argument of CONV1 and is also an array in the PFCS data input which contains fluid properties.

It is of the following format: APR(IC), CP, RO, MU, KT, GC, END
CP - is a doublet temperature dependent specific curve when input with the *A format
- is a constant specific heat valve if input as a real constant

RO - is a doublet temperature dependent density curve when supplied using *A
- is a constant density valve when supplied as a real constant

MU - is a doublet temperature dependent viscosity curve when supplied using *A
- is a constant viscosity valve when supplied as a real constant

KT - is a doublet temperature dependent thermal conductivity curve when supplied using *A
- is a constant conductivity valve when supplied as a real constant

GC - is the gravitational constant in the problem under consideration

ADAT - is an argument to CONVI which contains convection information. It has the following format:
ADAT(IC), NG1, AHT1, ITUBE1, NFL1, ITYPE1, X1, F11, F21
   ... ... ... ... ... ... ... ...
NGn, AHTn, ITUBEn, NFLn, ITYPEn, Xn, F1n, F2n
END

NGi - is actual conductor number of the ith set of data
AHT - is the area for heat transfer
ITUBE is the tube number for obtaining flowrate
NFL - is the actual fluid lump number
ITYPE is the fluid lump type number
X - is the entry length
F1 - is the laminar fully developed coefficient
F2 - is the laminar entry length coefficient
SUBROUTINE NAME: CONV2

PURPOSE:

CONV2 calculates convection conductor values from a user input curve of Stanton number (St) vs Reynolds number (Re). The film heat transfer coefficient is obtained by (1) interpolating a curve of St·Pr^{2/3} versus Re to obtain St·Pr^{2/3} and (2) using the relation \( h = \frac{k}{D} \left( St \cdot Pr^{2/3} \right) \cdot Re \cdot Pr^{1/3} \) to obtain \( h \). \( h \) is multiplied by the heat transfer area to obtain the conductor values which is stored in the proper conductor location. The flow data used by subroutine PFCS is referenced by the arguments AFLOW and APR to obtain flowrate, type data and fluid property data thus tying the convection and flow analysis together.

RESTRICTIONS:

Should be called in the VARIABLES 1 block so that hA values are calculated on each iteration prior to the temperature calculation.

CALLING SEQUENCE:

CONV2(AFLOW, APR, ADAT)

where:

- AFLOW - is the first argument of the PFCS call and is of the following format:
  AFLOW(IC), AW, APR, AGF, AVR, AIFR, AFT, AFR, APD, END

- AW - array of flowrates per tube in the system

- APR - array of pressures per pressure node in the system

- AGF - array of pressure conductors per tube

- AVF - array of valve positions for all valves in the system

- AIFR - array of imposed flowrate per node

- AFT - array of fluid type data

- AFR - array of user added flow resistance per tube

- APD - array for output of pressure drops per tube

- APR - is an array in the PFCS data input which contains fluid properties.
  It is of the following format:
  APR(IC), CP, RO, MU, KT, GC, END

- CP - is a doublet temperature dependent specific curve when input with the *A

- is a constant specific heat valve if input as a real constant
RO - is a doublet temperature dependent density curve when supplied using *A
- is a constant density value when supplied as a real constant

MU - is a doublet temperature dependent viscosity curve when supplied using *A
- is a constant viscosity value when supplied as a real constant

KT - is a doublet temperature dependent thermal conductivity curve when supplied using *A
- is a constant conductivity value when supplied as a real constant

GC - is the gravitational constant in the problem under consideration

ADAT - is an argument to CONV1 which contains convection information.
It has the following format:

ADAT(IC), NG1, AHT1, ITUBE1, NFL1, ITYPE1, AHST1
    :    :    :    :    :    :    :
    :    :    :    :    :    :
NGn, AHTn, ITUBE_n, NFL_n, ITYPE_n, AHST_n
END

NG1 - is conductor number of the 1st set of data

AHT - is the area for heat transfer

ITUBE - is the tube number for obtaining flowrate

NFL - is the fluid lump number

ITYPE - is the fluid lump type number

AHST - is a doublet curve of ST(PR)^2/3 vs Re
SUBROUTINE NAME: CONV3

PURPOSE:

CONV3 calculates convection conductor values by interpolating a user supplied curve of heat transfer coefficient, h, versus tube flowrate. The conductor is then obtained by multiplying h times the area, A. A large number of conductors may be processed with a single call to CONV3. The flow data used by subroutine PFCS is referenced by the argument AFLOW to obtain flowrate and type data.

RESTRICTIONS:

CONV3 should be called from the VARIABLES 1 block to obtain updated hA values on each iteration.

CALLING SEQUENCE:

CONV3 (AFLOW, ADAT)

where:

AFLOW - is the first argument of the PFCS call and is of the following format:

AFLOW(IC), AW, APR, AGF, AVR, AIFR, AFT, AFR, APD, END

AW - array of flowrates per tube in the system

APR - array of pressures per pressure node in the system

AGF - array of pressure conductors per tube

AVP - array of valve positions for all valves in the system

AIFR - array of imposed flowrate per node

AFT - array of fluid type data

AFR - array of user added flow resistance per tube

APD - array for output of pressure drops per tube

ADAT - is an argument to CONV1 which contains convection information.

It has the following format:

ADAT(IC), NGI, AHTI, ITUBEI, AHWI

...:

NGN, AHTN, ITUBEN, AHWN, END

NGi - is conductor number of the ith set of data

AHT - is the area for heat transfer

ITUBE - is the tube number for obtaining flowrate

AHW - is a doublet array of heat transfer coefficient vs flowrate
SUBROUTINE NAME: FLOCN1 or FLOCN2

PURPOSE:

Subroutine FLOCN1 and FLOCN2 calculate thermal conductor values required for thermal characterization of fluid flowing down a tube. The conductor values are obtained by multiplying the tube flowrates times the specific heat for each of the conductors identified in the ADAT array. Both subroutines reference the flowrate array, AW, and FLOCN1 references the ACP array which gives the specific heat vs temperature relationship. FLOCN2 assumes a constant value for specific heat.

The conductor values referenced in the ADAT array must also be supplied in the CONDUCTOR DATA block as one-way conductors with the proper connections identified. Any dummy value may be supplied for the initial flow conductor values since these values will be replaced following the first call to FLOCN1 or FLOCN2. These subroutines are called from VARIABLES 1.

RESTRICTIONS:

Must be called from VARIABLES 1.

CALLING SEQUENCES:

FLOCN1 (AW, ACP, ADAT1) or FLOCN2 (AW, CP, ADAT2)

where

AW - is the array of flowrates per tube also referenced in subroutine PFCS
ACP - is a doublet array of specific heat versus temperature
Cp - is a constant value of specific heat
ADAT1 - is the array which identifies the conductor, the corresponding upstream lump and the tube number for each conductor. It is of the format:

ADAT1, NG1, UPL1, ITUBE1
   NG2, UPL2, ITUBE2
   . . .
   NGn, UPLn, ITUBEn, END
ADAT2 - is the array which identifies the conductor, number and the tube number for each conductor. It is of the format:
ADAT2, NG1, ITUBE1, NG2, ITUBE2, --, NGn, ITUBEn, END

NGi - is the ith conductor number (the *G notation is used)

UPLi - is the upstream lump number for conductor number NGi
(The *T notation is used)

ITUBEi - is the tube number which contains the flow for the ith conductor $\dot{\omega}$. Cp product. For flow splitting or mixing junctions, ITUBEi should be the number of the connect tube containing the smallest amount of flow. For example, for a splitting junction the flow conductor which crosses the junction should contain the downstream tube. For a mixing junction ITUBE should be the upstream tube.
SUBROUTINE NAME: CONDT1 or CONDT2

PURPOSE:
Subroutines CONDT1 and CONDT2 calculate the value of conductance values as a product of a time variant argument $W(t)$ and a temperature variant or constant argument $C_p$. The value of $C_p$ is temperature dependent with CONDT1 and is constant for CONDT2. The subroutines were written primarily for the purpose of evaluating flow conductors for the case of flowrate a given function of time. However, they may be used for other time variant conductor applications.

RESTRICTIONS:
Should be called from VARIABLES 1.

CALLING SEQUENCE:
CONDT1 (ADAT1) or CONDT2 (ADAT2)
where ADAT1 is of the form
\[
\begin{align*}
\text{ADAT1, NG1, NLT1, ATIME1, ATEMP1} \\
\text{NG2, NLT2, ATIME2, ATEMP2} \\
\text{\ldots} \\
\text{NGn, NLTn, ATIMEn, ATMEPn, END}
\end{align*}
\]

and ADAT2 is of the form:
\[
\begin{align*}
\text{ADAT2, NG1, ATIME1, CP1} \\
\text{NG2, ATIME2, CP2} \\
\text{\ldots} \\
\text{NGn, ATIMEn, CPn, NED}
\end{align*}
\]

The following definitions apply to the above.

$NG_i$ - the conductor number of the $i$th conductor addressed in ADAT1 or ADAT2. The $*G$ notation should be used.

$NLT_i$ - the lump whose temperature will be used to interpolate the $ATEMP_i$ array to obtain the $C_p$ constant. The $*T$ notation should be used.

$ATIME_i$ - the time variant array for determining the value of $W(t)$ for the $i$th conductor. The $*A$ notation should be used.
ATEMP_i - the temperature variant array which is interpolated with
the temperature of lump NLT_i to obtain Cp in CONDT1. The
conductor is calculated as the value of W(t)*Cp. The
*A notation should be used.

Cp - the constant value which will be multiplied by W(t) from
the ATIME_i array to obtain the conductor value for CONDT2.
SUBROUTINE NAME: RADIR

PURPOSE:

RADIR calculates the script-F values for infrared radiation heat transfer within an enclosure and uses these values to obtain the heat transfer during the problem. Several temperature nodes may be combined on a single surface for radiation heat transfer purposes. Also, the user may analyze problems with specular, diffuse or combinations of specular and diffuse radiation. See Section 2.1.6.1 for definitions and detailed description of methods.

RADIR calculates the script-F values on the initial call. This is performed by the procedure outlined in Section 2.1.6.1, Equations 23, 25 and 31. These values replace the EFT values in the SC array for future use. The heat flux values are then calculated on all iterations by:

1. Calculating the temperature of each surface using equation 34
2. Calculating the absorbed heat for each node by the relation of equation 33

The value given by equation 35 is added to the conductor sum for each node so that the proper convergence time increment may be obtained. As many enclosures as desired may be analyzed by each enclosure but each enclosure requires a different call to RADIR. RADIR must be called in VARIABLES 1.

RESTRICTIONS:

Must be called from VARIABLES 1 Surface nodes must be boundary nodes

CALLING SEQUENCE:

RADIR (A(IC)).

Where A is of the following format:

A(IC),SN,SE,SR,SC,NA,SP,END

SN,SE,SR,SC,NA, and SP are actual array numbers input using the *A procedure and are of the following formats:

SN(IC),n,SN1,SA1,NN1,SN2,SA2,NN2,..............SNn,SA,n,NNn,END
SE(IC),SE1,SE2------SEn,END
SR(IC),SR1,SR2------SRn,END
SC(IC),SNF1,SNT1,EFT1,SNF2,SNT2,EFT2,---SNFm,SNTm,EFTm,END
NA(IC),NNO(1,1),AN(1,1),NNO(1,2),AN(1,2)---NNO(1,NN1),AN(1,NN1)
    NNO(2,1),AN(2,1),NNO(2,2),AN(2,2)---NNO(2,NN2),AN(2,NN2)
    
    NNO(n,1),AN(n,1),NNO(n,2),AN(n,2)---NNO(n,NNn),AN(n,NNn),END
SP(IC),SPACE,NSPACE,END
The following definitions apply in the above calling sequence:

- **A**: Array identification for the array which identifies the other arrays containing the data
- **SN**: Array number for the array containing surface numbers and areas
- **SE**: Array number for the array containing the surface emissivities
- **SR**: Array number for the array containing the surface reflectivities
- **SC**: Array number for the array containing the surface connections data
- **NA**: Array number for the array containing the temperature node numbers and areas
- **SP**: Array number for the array containing the space which is used for obtaining script FA values and for subsequent temperature calculations
- **n**: The number of surfaces
- **SN1,SN2,...,SNn**: Node number for surfaces - must be boundary nodes
- **SA1,SA2,...,SAN**: Total area for each surface
- **NN1,NN2,...,NNn**: Number of temperature nodes on each surface
- **SE1,SE2,...,SEN**: Emissivity values for each surface
- **SR1,SR2,...,SRn**: Diffuse reflectivity values for each surface
- **SNF1,SNT1,EFT1**: Connections data: Surface number from, surface number to, E value from SNF1 to SNT1, etc.
- **NNO(X,Y)**: Temperature node numbers on surfaces; Node number Y on surface X
- **AN(X,Y)**: Area of node Y on surface X
- **NSPACE**: Number of spaces needed to store script-FA values - NSPACE must be an integer values of $n \times n(n+1)/2$
- **m**: The number of surface connections
SUBROUTINE NAME: RADSOL

PURPOSE:

RADSOL calculates a pseudo script-F for radiation from an external source entering an enclosure and uses these values to calculate the net heat transfer to each node due to the entering source. A number of temperature nodes may be combined on a single surface for radiation purposes. Also, problems with specular, diffuse, or combinations of specular and diffuse radiation may be analyzed. Section 2.1.6.2 should be consulted for definitions and descriptions of methods.

RADSOL calculates the pseudo script-F values on the initial call. This is performed by equations 38, 40, and 44 of section 2.1.6.2. The values are stored in the EFT values of the SC array supplied by the user. The heat flux values are then calculated on each iteration by equations 45 and 46.

The user may analyze as many enclosures as desired by supplying a call statement for each enclosure. Also, a user may analyze several wave length bands by supplying a call to RADSOL for each wave length band.

RESTRICTIONS:

Must be called from VARIABLES I; Surface nodes must be boundary nodes

CALLING SEQUENCE:

RADSOL (A(IC))

Where the A array is of the following format:

A(IC),SN,SE,SR,HT,SC,NA,SP,END

SN,SE,SR,HT,SC,NA, and SP are actual array numbers input using the *A procedure and are of the following formats:

SN(IC), n,SN1,SA1,NN1,SN2,SA2,NN2,------Snn,SAan,NNn,END
SE(IC),SET1,SE2,------SETn,END
SR(IC),SR1,SR2,------SRn,END
HT(IC),SHT1,SHT2,------SHTn,END
SC(IC),SNF1,SNT1,EFT1,SNF2,SNT2,EFT2,------SNFm,SNTm,EFTm,END
NA(IC),NNO(1,1),AN(1,1),NNO(1,2),AN(1,2),--NNO(1,NN1),AN(1,NN1), NNO(2,1),AN(2,1),NNO(2,2),AN(2,2),--NNO(2,NN2),AN(2,NN2),
.......NNO(n,1),AN(n,1),NNO(n,2),AN(n,2),--NNO(n,NNn),AN(n,NNn),END
SP(IC),SPACE,NSPACE,END

59
The following definitions apply in the above calling sequence

A
Array identification for the array which identifies the other arrays containing the data

SN
Array number for the array containing surface numbers and areas

SE
Array number for the array containing the surface emissivities

SR
Array number for the array containing the surface reflectivities

HT
Array number for the array containing the incident heat curves or constant heat flux values

SC
Array number for the array containing the surface connections data

NA
Array number for the array containing the temperature node numbers and areas

SP
Array number for the array containing the space which is used for obtaining script values and for subsequent temperature calculations

SN1,SN2,...SNn
Node number for surfaces; must be boundary nodes

SA1,SA2,...SAN
Total area for each surface

N1,NN2,...NNn
Number of temperature nodes on each surface

SE1,SE2,...SEn
Emissivity values for each surface

SR1,SR2,...SRn
Diffuse reflectivity values for each surface

SHT1,SHT2,...SHTn
Incident heat flow on surfaces; may identify curves containing incident values vs time

SNF1,SNT1,EFT1
Connections data: Surface number from surface number to, E value from SNF1 to SNT1, etc.

NNO(X,Y)
Temperature node numbers on surfaces: Node number Y on surface X

AN(X,Y)
Area of node Y on surface X

NSPACE
Number of spaces needed to store script-FA values - NSPACE must be an integer values of n/2(n+1)
SUBROUTINE NAME: HXEFF

PURPOSE:
This subroutine obtains the heat exchanger effectiveness either from a user constant or from a biariant curve of effectiveness versus the flow rates on the two sides. The effectiveness thus obtained is used with the supplied flow rates, inlet temperatures and fluid properties to calculate the outlet temperatures using the methods described in Section 2.1.3.4. The user may specify a constant effectiveness by supplying a real number or may reference and array number to specify the effectiveness as a biariant function of the two flow rates. The user also supplies flow rates, specific heat values, inlet temperatures and a location for the outlet temperatures for each of the two sides. The flow rate array may be referenced to obtain flow rates and the temperature array may be used for temperatures. The specific heat values may be supplied as a temperature dependent curve or a constant value may be supplied.

RESTRICTIONS:
HXEFF should be called in the VARIABLES 1 block. The value for EFF, the first argument must never be zero. Tout1 and Tout2 must be boundary nodes.

CALLING SEQUENCE:
HXEFF(EFF,W1,W2,CP1,CP2,TIN1,TIN2,TOUT1,TOUT2)

Where
EFF - is (1) the effectiveness if real, (2) a curve number of a biariant curve of effectiveness versus W1 and W2 if an array
W1,W2 - are the flow rates for side 1 and 2 respectively. May reference the flow rate array, AW+I where I is the tube number
CP1,CP2 - are the specific heat value for side 1 and side 2 fluid respectively. Constant values may be input or arrays may be used for temperature dependent properties
TIN1,TIN2 - are inlet lump temperatures - Usually T(IN1) and T(IN2) where IN1 and IN2 are the inlet lumps on side 1 and side 2
TOUT1,TOUT2 - are the outlet lump temperature locations sides 1 and 2 where the calculated values will be stored Must be boundary nodes
SUBROUTINE NAME: HXCNT

PURPOSE:

This subroutine calculates the heat exchanger effectiveness using the relation described in Section 2.1.3.1 for a counter flow type exchanger. The value of UA used in the calculations may be specified as a constant by supplying a real number or it may be specified as a bivariant function of the two flow rates by referencing an array number. The user also supplies flow rates, specific heat values, inlet temperatures and a location for the outlet temperatures for each of the two sides. The flow rate array may be referenced to obtain flow rates and the temperature array may be used for temperatures. The specific heat values may be supplied as a temperature dependent curve or a constant value may be supplied.

RESTRICTIONS:

HXCNT should be called in the VARIABLES 1 block. The value for UA, the first argument must never be zero. Tout1 and Tout2 must be boundary nodes.

CALLING SEQUENCE:

HXCNT(UA,W1,W2,CP1,CP2,TIN1,TIN2,TOUT1,TOUT2)

Where

UA is (1) the heat exchanger conductance if real, (2) a curve number of a bivariant curve of conductance versus W1 and W2 if an array

W1,W2 are the flow rates for side 1 and side 2 respectively. May reference the flow rate array, AW+1 where is the tube number

CP1,CP2 are the specific heat values for side 1 and 2 fluid respectively. Constant values may be input or arrays may be used for temperature dependent properties

TOUT1-TOUT2 are the outlet lump temperature locations (sides 1 and 2) where the calculated values will be stored Must be boundary nodes
SUBROUTINE NAME: HXCROS

PURPOSE:

This subroutine calculates the heat exchanger effectiveness using the relations described in Section 2.1.3.3 for a cross flow type exchanger. The value of UA used in the calculations may be specified as a constant by supplying a real number or it may be specified as a bivariant function of the two flow rates by referencing an array number. Any one of the following four types of cross flow exchangers may be analyzed (see Section 2.1.3.3 for the relations):

1) Both streams unmixed
2) Both streams mixed
3) Stream with smallest MCp product unmixed
4) Stream with largest MCp product unmixed

The type is specified by the last argument in the call statement. The user also supplies flow rates, specific heat values, inlet temperatures and a location for the outlet temperatures for both sides. The flow rate array may be referenced to obtain flow rates and the temperature array may be used for temperatures. The specific heat values may be supplied as a temperature dependent curve or a constant value may be supplied.

RESTRICTIONS:

HXCROS should be called in the VARIABLES 1 block. The value for UA, the first argument must never be zero. Tout1 and Tout2 must be boundary nodes.

CALLING SEQUENCE:

HXCROS(UA,W1,W2,CP1,CP2,TIN1,TIN2,TOUT1,TOUT2,K)

Where

UA is (1) the heat exchanger conductance if real, (2) a curve number of a bivariant curve of conductance versus W1 and W2 if an array.

W1,W2 are the flow rates for side 1 and side 2 respectively. May reference the flow rate array, AW+I where I is the tube number

CP1,CP2 are the specific heat values for side 1 and side 2 fluid respectively. Constant values may be input or arrays may be used for temperature dependent properties

TIN1,TIN2 are inlet lump temperatures - Usually T(IN1) and T(IN2) where IN1 and IN2 are the inlet lumps on side 1 and side 2

TOUT1,TOUT2 are the outlet lump temperature locations(sides 1 & 2) where the calculated values will be stored. Must be boundary nodes
K is the code specifying type of cross flow exchanger:

Both streams unmixed: \( K=1 \)
Both streams mixed: \( K=2 \)
Stream with small WCP Unmixed: \( K=3 \)
Stream with large WCP Unmixed: \( K=4 \)
SUBROUTINE NAME: HXPAR

PURPOSE:

This subroutine calculates the heat exchanger effectiveness using the relations described in Section 2.1.3.2 for a parallel flow type exchanger. The value of UA used in the calculations may be specified as a constant by supplying a real number or it may be specified as a bivariant function of the two flow rates by referencing an array. The user also supplies flow rates, specific heat values, inlet temperatures and a location for the outlet temperatures for each of the two sides. The flow rate array may be referenced to obtain flow rates and the temperature array may be used for temperatures. The specific heat values may be supplied as a temperature dependent curve or a constant value may be supplied.

RESTRICTIONS:

HXPAR should be called in the VARIABLES 1 block. The value for UA, the first argument must never be zero. Tout1 and Tout2 must be boundary temperatures.

CALLING SEQUENCE:

HXPAR(UA,W1,W2,CP1,CP2,TIN1,TIN2, Tout1,Tout2)

Where

UA is (1) the heat exchanger conductance if real, (2) a curve number of a bivariant curve of conductance versus W1 and W2 if an array.

W1,W2 are the flow rates for side 1 and 2 respectively. May reference the flow rate array,AW+I where I is the tube number.

CP1,CP2 are the specific heat values for side 1 and side 2 fluid respectively. Constant values may be input or arrays may be used for temperatures dependent curves.

TIN1,TIN2 are inlet lump temperatures - Usually T(IN1) and T(IN2) where IN1 and IN2 are the inlet lumps on side 1 and side 2.

TOUT1,TOUT2 are the outlet lump temperature locations (sides 1 and 2) where the calculated values will be stored (should be boundary temperatures).
SUBROUTINE HEATER

PURPOSE:

This subroutine simulates an electrical heater with a control system which turns the heater on when the sensor lump temperature falls below the "heater on" temperature TON, and turns the heater off when the sensor lump rises above the heater off temperature, TOFF. When the heater is on, the input Q value is added to the Q location specified by the user. When the heater is off, no heat is added.

RESTRICTIONS:

HEATER must be called in the VARIABLES 1 block.

CALLING SEQUENCE:

HEATER(Q,QHT,KODE,TSEN,TON,TOFF)

Where

TSEN is the sensed temperature
TON is the heater on temperature
TOFF is the heater off temperature
QHT is the heater heat rate
Q is the location for storing the heat

KODE is an integer variable set by HEATER
= 1 if the heater was "on" at last call
= 0 if the heater was "off" at the last call
(User sets KODE for first call)
SUBROUTINE NAME:  
CABIN

PURPOSE:

This subroutine performs a thermal and mass balance on a cabin air system. The cabin air is assumed to be a two component gas mixture with one condensible component and one noncondensible component. The cabin air is assumed to be well mixed so that the temperature and specific humidity are constant throughout. The cabin may contain any number of entering streams each with different temperature and humidity conditions. The cabin air may transfer heat to any number of nodes in its surroundings with the heat transfer coefficient obtained by one of the three options:

1. User input coefficient
2. Relations for flow over a flat plot
3. Relations for flow over a tube bundle

The relations describing the second and third options are given in Section 2.1.5. The mass transfer coefficient for determining the rate of condensation or evaporation is determined by the Lewis relation which relates the mass transfer coefficient directly to the convection heat transfer coefficient. By the Lewis Relation, if the diffusion coefficient is approximately equal to the thermal diffusivity, the Sherwood number is approximately equal to the Nusselt number, thus giving a direct relation. (See Section 2.1.5 for details). Mass and heat transfer rates are determined at each node that interfaces the cabin gas as well as entering and exiting streams and a new cabin gas temperature and humidity is determined each iteration based upon the heat and mass balance. An account is kept of the condensate on the walls when condensation occurs but the condensate is assumed to remain stationary and not flow to other wall nodes.

Limits are applied when necessary to prevent more condensation than the vapor existing under severe transient condition and to prevent evaporation of more liquid than exists at each wall lump.

As many cabins as desired may be analyzed in a given problem, but each must contain separate input information.

RESTRICTIONS:

CABIN must be called in VARIABLES 1.

CALLING SEQUENCE:

CABIN(A(IC) TC, TC, K1, K2)

The following definitions apply to the above calling sequence:

A is an array containing arrays numbers which contain cabin input information
The cabin gas temperature which must be a boundary node

Storage locations needed by CABIN

The A array has the following format where the *A procedure is used:

\[ A(IC),IF,PR,CN,H,FP,TB,SP,END \]

Where

- **IF** identifies an array containing the entering flow rate information. The format of the array is:
  \[ IF(IC),NS,FR_1,PSI,TE_1,FR_2,PSI_2,TE_2,\ldots,FR_{ns},PSI_{ns},TE_{ns} \]

- **PR** identifies an array identifying array numbers for property values. The format of the array is:
  \[ PR(IC),NFLC,NMUO,NMUV,NCPV,NKO,NKV,NLAT \]

- **CN** identifies an array containing pertinent constants. The format of the array is:
  \[ CN(IC),RA,RV,VC,PC,XC,WV,PSIC,PO,TO,CONV \]

- **H** identifies an array containing node numbers and convection heat transfer coefficient values for nodes surrounding the cabin gas. The format of the array is:
  \[ H(IC),LN_1,HA_1,LN_2,HA_2,\ldots,LN_{n1},HA_{n1} \]

- **FP** identifies an array containing node numbers and information to permit calculation of convection coefficients for flat plates. The format is:
  \[ FP(IC),LN_1,XX_1,XI_1,Al_1,VIWO_1,LN_2,XX_2,XI_2,Al_2,VIWO_2,\ldots,LN_{n2},XX_{n2},XI_{n2},Al_{n2},VIWO_{n2} \]

- **TB** identifies an array containing node numbers and information to permit calculation of convection coefficients for tube bundles. The format is:
  \[ TB(IC),LN_1,DI_1,Al_1,VIWO_1,LN_2,DI_2,Al_2,VIWO_2,\ldots,LN_{n3},DI_{n3},Al_{n3},VIWO_{n3} \]

- **SP** identifies an array which contains working space equal to or greater than three times the sum of the number of nodes with input heat transfer coefficients plus the number using flat plot relations plus the number using tube bundles.

The following symbol definitions apply in the above:
NS  Number of incoming streams
FR\textsubscript{i}  Entering flow rate for stream i
PSI\textsubscript{i}  Specific humidity for entering stream i
TE\textsubscript{i}  Temperature of entering stream i
NFLC  Curve number for circulation flow rate vs time
NMUO  Curve number for noncondensible viscosity vs temperature
NMUV  Curve number for condensible viscosity vs temperature
NCPO  Curve number for noncondensible specific heat vs temperature
NCPV  Curve number for condensible specific heat vs temperature
NKO  Curve number for noncondensible thermal conduction vs temperature
NKV  Curve number for condensible thermal conduction vs temperature
NLAT  Curve number for latent heat of condensible vs temperature
RA  Gas constant for non-condensible component
RV  Gas constant for condensible component
VC  Cabin volume
PC  Cabin Pressure
XC  Molecular weight ratio, \(M_v/M_o\)
WV  Initial vapor weight in cabin
PSIC  Initial specific humidity for cabin
LN\textsubscript{i}  Cabin wall lump
HA  Heat transfer coefficient times area
n1  Number of wall lumps which have input HA values
n2  Number of wall lumps which have HA calculated by flat plate relations
n3  Number of wall lumps which have HA calculated by tube bundle relations
<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_{xi}$</td>
<td>Distance from leading edge for flat plate heating for ith flat plate node</td>
</tr>
<tr>
<td>$XI_{i}$</td>
<td>Length of flat plate in flow direction for ith flat plate node</td>
</tr>
<tr>
<td>$AI_{i}$</td>
<td>Heat transfer area for flat plate or tube node</td>
</tr>
<tr>
<td>$DI_{i}$</td>
<td>Tube outside diameter for tubes in the bundle for ith tube node</td>
</tr>
<tr>
<td>$VIWO$</td>
<td>Ratio of velocity at the lump to the circulation flow rate</td>
</tr>
<tr>
<td>$To$</td>
<td>The reference temperature to be used for estimating the saturation pressure of the condensible component. Should be near the range of saturation temperature expected</td>
</tr>
<tr>
<td>$Po$</td>
<td>The saturation pressure at $To$ for the condensible component</td>
</tr>
<tr>
<td>CONV</td>
<td>Conversion factor to make the quantity $XLAM/Rv/To$ dimensionless where $XLAM$ is the latent heat of vaporization and $Rv$ is the gas constant for the vapor. If $XLAM$ is BTU/lb, $Rv$ is FT-LB/$°R$ and $To$ is $°R$, $CONV=778$.</td>
</tr>
</tbody>
</table>
SUBROUTINE NAME: PFCS

PURPOSE:
Subroutine PFCS determines the flow distribution in a set of general parallel/series fluid flow tubes so that the pressure drop values between any parallel flow paths are equal and flow is conserved. The following effects are included in the pressure drop calculations:

1. pipe flow friction
2. orifices and fittings
3. valves

The effect of temperature dependent properties are included in the calculations. The properties are evaluated at the temperature of each fluid lump in each tube in evaluating the flow resistance when setting up the equations to be solved.

A balance is made between the flow/pressure drop characteristics of the system and the flow/pressure rise of a pump for each system concurrent with the system pressure flow solution to obtain the incoming system flowrate. A detailed discussion of the equations and techniques used are described in Section 2.2.

General flow charts of PFCS and supporting subroutines are shown in Fig. 7,8, & 9.

RESTRICTIONS:
Must be called from VARIABLES 2. The system of units used for the thermal and flow problems should be consistent.

CALLING SEQUENCE: PFCS (AFLOW, ADAT, NAME)

where AFLOW - is an array which references other arrays for flowrates, pressures, flow conductors, valve positions, imposed flowrates, fluid type data, user added flow resistances and pressure drops. It is of the following format where the *A conversion feature described in Section 3.1 is used to reference arrays.

AFLOW (IC), AW, APN, AGF, AVP, AIFR, AFT, AFR, APD, END

ADAT - is an array which identifies other arrays containing fluid property values, parameters needed for the pressure/flow solution numerical technique, the flow system network, valve data, pump data and a check outprint code. The integer count should be addressed. The format of ADAT is as follows where
the *A format is used to address array values:
ADAT(IC), APR, ASOL, ANET, AVLS, AP, KOP, END

NAME - is an array containing the name of the network (it may also be supplied as a Hollerith using the H format). Nine 6 character words should be used.

AFLOW ARGUMENTS

AW - is the array number of an array containing flowrates per tube for each system. The integer count must be addressed and it must contain the number of spaces exactly equal to the number of tubes in the system.

APN - is an array number for an array containing the pressures for each pressure node in the system. On input the user need only set up the space. The integer count must be addressed and it must contain the number of spaces exactly equal to the number of pressure nodes in the system.

AGF - is an array number for an array containing the flow conductors for each tube in the system. The user needs only to setup the space on input which must be exactly equal to the number of tubes in the flow system.

AVP - is an array number for an array containing valve positions in order of valve numbers. The integer count must be addressed and the number of input valves must be exactly equal to the number of valves. The user supplies the initial valve positions in this array.

AIFR - is the array location of an array of imposed flow sources for each pressure node. The integer count must be addressed and the array must contain the number of spaces exactly equal to the number of pressure nodes in the flow system.

AFT - is the array location of an array which contains fluid lump type data. The AFT array is of the following format:
AFT, \( WP_1, CSA_1, FLL_1, MFF_1, NHL_1, FFC_1 \)
\( \vdots \)
\( WP_{nt}, CSA_{nt}, FLL_{nt}, MFF_{nt}, NHL_{nt}, FFC_{nt} \)
END

nt - is the number of types
WP_i - is the wetted perimeter of fluid type i
CSA_i - is the cross sectional area for fluid type i
FLL_i - is the fluid lump length for fluid type i
MFF_i - is the curve of friction factor vs Reynolds number for Reynolds number greater than 2000 when greater than 0.

- is a key to use internal calculations methods for friction factor when MFF = 0.

NHL_i - is the number of head losses for type i when real
- is the number of an array of head losses vs Reynolds number when an integer

FFC_i - is a user input constant to be multiplied times the friction factor to modify it for type i.

AFR - is an array number of an array containing user added flow resistances for the tubes. This can be used to include the effects of changes in flow altitudes or the effects of valve types not available in the valve package or other known flow resistances. The integer count must be addressed and the number of values in the array must be exactly equal to the number of tubes in the flow system.

APD - is an array number of an array which will contain pressure drop values for all tubes in the flow system following a call to PFCS. The array is strictly for output purposes. The integer count must be addressed and the number of array valves must be exactly equal to the number of tubes in the system.

ADAT ARGUMENTS

APR - is an array identifying the fluid properties data and GC, the gravitational constant. It is of the following format: APR, CP, RO, MU, KT, GC, END

CP, RO, MU, KT, are the values of fluid specific heat, density, viscosity, and thermal conductivity respectively or the appropriate array reference (using the *A format). The value is constant for any of the properties if a real number is supplied. The integer count must be referenced when variable properties are used.
GC  - is the gravitational constant. Table 2 gives the value for various system of units.

ASOL - is an array number of an array containing various numerical solution parameters needed by PFCS. ASOL is of the following format:

ASOL (IC), TOL, MXPASS, EPS, FRDF, END

TOL  - is the solution tolerance on rate of change of flowrates from one pass to the next. The fraction of change must be within TOL for all tubes in any system or subsystem before a solution is reached. TOL must be greater than 0. A typical value is 0.001

MXPASS - is the maximum number of passes permitted in the balancing loop of PFCS to obtain a pressure-flow solution on any given iteration. This value should always be greater than 20 with a typical value of 100.

EPS  - Not used but a space must be supplied.

FRDF  - is the flowrate damping factor used to accelerate the rate of convergence for the iterative solution to the set of non-linear equations. This value should generally be between 0.5 and 1.0. Values of 0.5 to 0.7 have been found best for most turbulent flow problems.

ANET - is the array number of an array which identifies the tube connections, pressure nodes connected and fluid lumps contained in the tube. The format of ANET is as follows were the *A format is used for AD, APNPS, and AVL:

ANET, NNAME, APNPS, AVL
TUBE1, NFRM1, NTO1, KD1, AD1
TUBE2, NFRM2, NTO2, KD2, AD2
... ...
TUBEn, NFRMn, NTON, KDN, ADn, END
NNAME  - is a six character name to be used for identifying the network in output statements, etc.

APNPS  - is an array number (referenced using *A) of an array identifying nodes with specified pressures and is of the form
           APNPS, NSPRI, NSPR2, - - - - NSPRn, END

NSPRI  - is the ith pressure node with specified value

TUBEi  - is the tube number of the ith connection

NFRMi  - is the "from" pressure node for the ith connection

NTOi  - is the "to" pressure node for the ith connection

KDi  - is an integer code to identify the type of conductor for the ith connection. (See ADi below)

ADi  - is the data to be used for calculating the conductor value for the ith connection.

If KDi < 0, the conductor value is the equivalent conductance of a subnetwork described by array ADi. ADi is then of identical format to ANET.

If KDi > 0, the conductor is obtained by the normal pressure drop equations and array ADi fluid lumps, fluid lump types and tube lumps that are contained in the tube. The form of ADi is

ADi, NFLMP1, ITYPE1, NTLMP1
       NFLMP2, ITYPE2, NTLMP2
       :       :       :
       NFLMPn, ITYPEn, NTLMPn, END

Where NFLMPi is ith temperature lump contained in the tube, ITYPEi is the NFLMPi fluid lump type, and NTLMP is the tube lump containing NFLMPi.

If KDi = 0, the conductor calculation is not made, allowing the user to supply the pressure conductance value. ADi is not used.
AVLS - is the array location (identified in array ADAT using the *A format) of an array which identifies the array location of the valve data for all the valves in the system.

AVL - is the array location (identified in array ANET using the *A format) of an array which identifies the array locations of the valve data for the valves in the network or subnetwork described by ANET.

AVLS and AVL are of the following format:

```
AVLS or AVL, AVLV1, AVLV2, - - - - AVLVNV, END
```

Where AVLVi is the array number (using *A) of an array which contains the valve data for the ith valve in AVLS or AVL. The format for the valve data arrays, AVLVi, is one of three forms depending on the valve type (rate limited, polynomial, or switching).

The format for a rate limited valve is:

```
AVLV, NV, NTS1, NTS2, MODE, XMIN1, XMAX1, E, TSEN1, TSEN2, DB, RF, RL, END
```

The format for a polynomial valve is:

```
AVLV, NV, NTS1, NTS2, MODE, XMIN1, XMAX1, E, TSEN1, TSEN2, AO, A1, A2, A3, A4, A5, VTC, END
```

The format for a switching valve is:

```
AVLV, NV, NTS1, NTS2, MODE, XMIN1, XMAX1, E, NSEN, T1, T2, END
```

The following definitions apply for the above arrays:

- **NV** - Valve number
- **NTS1** - Tube number connected to side 1 of the valve
- **NTS2** - Tube number connected to side 2 of the valve
- **MODE** - Operating mode: 1 - operating; 0 - not operating
- **XMIN1** - Side 1 minimum position; side 2 maximum position is \(1.0 - XMIN1\)
- **XMAX1** - Side 1 maximum position; side 2 minimum position is \(1.0 - XMAX1\)
- **E** - The valve geometric factor relating pressure drop through the valve by

\[
\Delta P = E(\text{flowrate}/\text{valve position})^2
\]
TSEN1 - Sensor lump for side 1 or set point for side 2; If TSEN1 is an integer, it identifies the side 1 sensor lump to be controlled to (a) the set point for side 1 or (b) the sensor lump for side 2 (TSEN2). If the variable is input as a real number it represents a set point to which the side 2 sensor lump will be controlled.

TSEN2 - Sensor lump for side 2 or set point for side 1; If TSEN2 is an integer, it identifies the side 2 sensor lump to be controlled to (a) the set point for side 2 or (b) the sensor lump for side 1 (TSEN1). If the variable is input as a real number it represents a set point to which the side 1 sensor lump will be controlled.

AO, A1, A2, A3, A4, A5 - Polynomial curve fit coefficients for a curve fit of the steady state valve position vs sensed temperature error for side 1:

\[ X_{ISS} = A0 + A1 \cdot \Delta T + A2 \cdot \Delta T^2 + A3 \cdot \Delta T^3 + A4 \cdot \Delta T^4 + A5 \cdot \Delta T^5 \]

DB - Dead band for the rate limited valve, degrees of temperature (See Figure 5).

RF - Rate factor, the rate of change of valve velocity to sensed temperature error (dx/d(\Delta T)) as shown on Figure 5.

RL - Rate limit, the maximum valve velocity, \( \dot{x}_{\text{max}} \) (See Figure 5).

VTC - Valve time constant as described in Section 2.2.3.2. If a valve is desired with no time lag, a time constant which is very small compared to the problem time increment should be input. (VTC must be greater than zero).

NSEN - Sensor lump for switching valve

T1 - Side 1 off temperature or side 2 on temperature for switching valve

T2 - Side 2 off temperature or side 1 on temperature for switching valve

AP - is the array number of an array containing the pump data for the system specified in the ADAT array using the *A nomenclature. The format of the AP array is different for different types of pumps. If flowrate is a function of time the format is (where AW is supplied using *A):
If the flowrate is obtained using a tabulated pump curve the format is:
(\text{where ADP is supplied with } *A)
\text{AP, NPI, NPO, ADP, END}

If the flowrate is obtained using a polynomial pump curve, the format is:
\text{AP, NPI, NPO, AO, A1, A2, A3, A4, END}

The following definitions apply in the above arrays:

\text{NPI} \quad \text{- System inlet pressure node}
\text{AW} \quad \text{- Tabulated curve of flowrate vs time}
\text{NPO} \quad \text{- System outlet pressure node}
\text{ADP} \quad \text{- Tabulated pump curve giving pressure rise as a function of flowrate}
\text{AO, A1, A2, A3, A4} \quad \text{- Polynomial curve fit constants for flowrate as a function of pressure rise. i.e.,}
\quad \dot{w} = AO + A1 \cdot \Delta P + A2 \cdot \Delta P^2 + A3 \cdot \Delta P^4 + A4 \cdot \Delta P^4
\text{KOP} \quad \text{- is an integer code for checkout print from subroutine PFCS.}
\quad \text{If KOP} = 1 \text{ a checkout print will be obtained. If KOP} = 0 \text{ a print will not be obtained.}

\text{DYNAMIC STORAGE REQUIREMENTS:}

\text{Dynamic storage required for PFCS is } \frac{1}{2}(NPRN^2 + 6\times NPRN + 12), \text{ where NPRN is the maximum of the number of pressure nodes in any network.}
<table>
<thead>
<tr>
<th>UNITS</th>
<th>GC</th>
</tr>
</thead>
<tbody>
<tr>
<td>LB</td>
<td>386.1</td>
</tr>
<tr>
<td>Min</td>
<td>1.390×10^6</td>
</tr>
<tr>
<td>Hr</td>
<td>5.004×10^9</td>
</tr>
<tr>
<td>Sec</td>
<td>32.174</td>
</tr>
<tr>
<td>Min</td>
<td>1.1583×10^5</td>
</tr>
<tr>
<td>Hr</td>
<td>4.1696×10^8</td>
</tr>
<tr>
<td>Sec</td>
<td>10.725</td>
</tr>
<tr>
<td>Min</td>
<td>3.861×10^4</td>
</tr>
<tr>
<td>Hr</td>
<td>1.3899×10^8</td>
</tr>
<tr>
<td>Sec</td>
<td>1.0</td>
</tr>
<tr>
<td>Min</td>
<td>3600.</td>
</tr>
<tr>
<td>Hr</td>
<td>1.296×10^7</td>
</tr>
<tr>
<td>Sec</td>
<td>1×10^{-2}</td>
</tr>
<tr>
<td>Min</td>
<td>36</td>
</tr>
<tr>
<td>Hr</td>
<td>1.296×10^5</td>
</tr>
<tr>
<td>Sec</td>
<td>1.0</td>
</tr>
<tr>
<td>Min</td>
<td>3600.</td>
</tr>
<tr>
<td>Hr</td>
<td>1.296×10^7</td>
</tr>
</tbody>
</table>
FIGURE 7 FLOW CHARTS OF PFCS AND NTWRK
FIGURE 8 FLOW CHARTS OF NTWRK1 AND NTWRKN
FLOBAL

CALCULATE VALVE RESISTANCES

SET UP SPACE FOR MATRIX, RIGHT HAND SIDE, ETC.

ASSEMBLE COEFFICIENT MATRIX

STORE IMPOSED FLOW RATES IN RHS

STORE INLET FLOW RATE IN RHS

REDUCE MATRIX AND ADJUST RHS

CALL SYMSOL FOR PRESSURES

A

CALCULATE NEW FLOW RATES

RETURN

FIGURE 9 FLOW CHART OF FLOBAL
SUBROUTINE NAME: HSTRY

PURPOSE:

Subroutine HSTRY stores the problem time, the pressures of all pressure nodes, the valve positions for all valves, the flowrates for all tubes, and the temperatures of all temperature nodes at an input interval on a magnetic tape (the history tape) mounted on Unit T. The number of records written on the history tape is the number of history intervals plus two. The first record contains a title, an integer count of the number of items to be written for each of the four categories (pressures, valve positions, flowrates and temperatures), and the actual node numbers in order of the relative numbers. The second thru the next-to-last records contain the history records with one for each time point and the last record is the same as the next-to-last except the time is negative. The arguments to HSTRY are the pressure array, PR, the valve position array, VP, the flow rate array, W, and the history tape writing interval, TINC.

The format for the history tape is as follows:

Record No. 1
Title (Written Internally) in 12A6 format, 0, 0, 0, 0, 0, 0, No. of pressure nodes, number of valve positions, 0, 0, 0, number of tubes, 0, 0, number of nodes, actual node numbers in increasing order of relative node numbers.

Record No. 2
Initial problem time, pressures, valve positions, flowrates, node temperatures

Record No. 3
Second history time, pressures, valve positions, flowrates, node temperatures

Record No. N+1 (Where N = number of history time slices to be written)
Last history time, pressures, valve positions, flowrates, node temperatures
Record No. N+2
Same as last record except time is negative

RESTRICTIONS:
Should be called in VARIABLES 2. An output history tape should be mounted on Unit T. Subroutine TMCHK must be in VARIABLES 2 prior to the call to Subroutine HSTRY if TIMCHK is called in the problem.

If the backup feature is used in VARIABLES 2, the call to subroutine HSTRY should not be made until the last pass to avoid nonincreasing time records or invalid data. For example:

```
BCD 3VARIABLES 2

F IF (T(i6) .LT. TMAX) BACKUP = 1.

F IF (BACKUP .GT. 0.) GO TO 10
HSTRY (A1, A2, A3, .01)
F 10 CONTINUE
END
```

CALLING SEQUENCE:
```
HSTRY(PR(IC), VP(IC), W(IC), TINC)
```

PR - is the pressure (or pressure drop) array
VP - is the valve position array
W - is the flowrate array
TINC - is the time interval for plotting
SUBROUTINE NAME: NEWTMP

PURPOSE:
Subroutine NEWTMP will read the node temperatures, flowrates, pressures and valve positions at time TMPTIM from the history tape assigned to Unit U generated by subroutine HSTRY for a previous run on Unit T to initiate a problem at these conditions. The pressure array, PR, valve position array, VP, flow rate array, W, and time to read the tape, TMPTIM, are arguments. The subroutine should be called in the execution block prior to the call to the temperature solution subroutine.

RESTRICTIONS:
Must be called in the EXECUTION block prior to the call to the appropriate temperature solution subroutine. The history tape must be assigned on Unit U.

CALLING SEQUENCE:
NEWTMP(PR(IC), VP(IC), W(IC), TMPTIM)

PR - is the pressure array
VP - is the valve position array
W - is the flowrate array
TMPTIM - is the time to read the values of PR, VP, W and temperatures from the U tape
SUBROUTINE NAME: FLPRNT

PURPOSE:

Subroutine FLPRNT will write the values of the DATA array of real numbers at 10 to a line. The array is labeled by the variable input HEAD which contains 9 six character alpha numeric words. The array location of every tenth value in the array is identified to the right of the appropriate line. FLPRNT was written primarily for the output of flowrates, pressures, pressure drops, and valve positions obtained from PFCS but may be used for the output of any real array.

RESTRICTIONS:

Should be called from OUTPUT. The array must be real.

CALLING SEQUENCE:

FLPRNT(DATA(IC), HEAD(DV))

SUBROUTINE NAMES: GENOUT, GENI OR GENR

PURPOSE:

These subroutines print out arrays of numbers 10 to a line. GENOUT prints either real numbers, integer or both. GENI and GENR print integers and real number arrays respectively. The integers are written in an I9 format and the real numbers in an E12.4 format.

RESTRICTIONS:

GENI writes arrays of integers only. GENR writes arrays of real numbers only.

CALLING SEQUENCE:

GENOUT (A, ISTRT, ISTP, 'NAME')
GENI (A, ISTRT, ISTP, 'NAME')
GENR (A, ISTRT, ISTP, 'NAME')

where A - is the array location
ISTRT - is the first value in A being written
ISTP - is the last value in A being written
'NAME' - is a title of 22 Hollerith words for identification
SUBROUTINE NAME: FLUX

PURPOSE:
Subroutine FLUX permits doublet time variant curve values stored on magnetic tape unit NFLXTP to be read into NCRV arrays starting at array DATA when the mission time exceeds DQTIME. The flux tape must be generated prior to the run using a GE routine LTVFTP. This routine generates the flux tape in the following format:

Record No. 1
First Read Time

Record No. 2
Number of points on first curve (Integer), first curve independent variables, first curve dependent variables, number of points on second curve, second curve independent variables, second curve dependent variables, etc. for all curves.

Record No. 3
Second Read Time

Record No. 4
Same as Record No. 2 except with new values

Record No. 5
Third Read Time
Etc. until all blocks of data are on tape.

Subroutine FLUX writes the values from the appropriate NFLXTP record into the arrays defined by DATA and NCRV in the proper doublet array format. Flux values should be input into the heat flux arrays (DATA1---DATA_NCRV) initially if the user doesn't want the values to be read from the tape at the start of the problem. The value of QTIME should initially be the value of the time the first read is desired.

RESTRICTIONS:
The following restrictions apply:

(1) The initial block of curve data must be input on cards or data
(2) Particular curves must have the same number of points on each block of data read in as were input on cards initially.

(3) Each curve may have a different number of points.

(4) The first point on each curve in each block of data must be the same as the last point on that curve in the previous block of data.

(5) All incident heat curves must be in a single block by themselves.

**CALLING SEQUENCE:**

\[
\text{FLUX}(\text{NFLXTP}, \text{DATA}, \text{NCRV}, \text{DQTIME}, \text{QTIME})
\]

where

- \text{NFLXTP} - logical unit to which the flux tape is assigned. Must be supplied by a user constant.
- \text{DATA} - starting location (IC) for flux curves
- \text{NCRV} - number of flux curves to be updated from the flux tape
- \text{DQTIME} - time scale shift for flux curves. \text{DQTIME} is added to each independent value for each flux curve read from NFLXTP.
- \text{QTIME} - the last point on the latest set of flux curves read from NFLXTP. (\text{QTIME} = \text{FLXTIM} + \text{DQTIME}, where \text{FLXTIM} is the time read from the flux tape) must be supplied by user constant.
SUBROUTINE NAME: TIMCHK

PURPOSE:

Subroutine TIMCHK compares the elapsed computer time against the requested computer time, RTIME, and terminates the run if RTIME is exceeded by the elapsed time. If the second argument, KODE, is non-zero an output of computer time used will be printed out on each call to TIMCHK. Thus, a call to TIMCHK in VARIABLES 2 should normally be with KODE=0. If the output of computer time used is desired, TIMCHK should be called from OUTPUT with KODE ≠ 0. The most desirable procedure is to supply two calls to TIMCHK: (1) a call in VARIABLES 2 with KODE = 0 and (2) a call in OUTPUT with KODE ≠ 0.

RESTRICTIONS:

KODE should zero when called from VARIABLES 1 or 2.

CALLING SEQUENCE:

TIMCHK (RTIME, KODE)

where RTIME = maximum computer time requested
KODE = print code: = 0, computer time used is not printed out ≠ 0, computer time used is printed out on each call to TIMCHK
SUBROUTINE NAME: REVPOL

PURPOSE:
This subroutine performs single variable linear interpolation on a doublet array of \( X,Y \) pairs in the same manner as \( D1DEG1 \) except in reverse order. The array is interpolated in reverse order to obtain the value of independent variable, \( X \), which corresponds to the input dependent variable, \( Y \).

RESTRICTIONS:
All values must be floating point numbers.

CALLING SEQUENCE:
\[
\text{REVPOL} \ (Y,A(IC),X)
\]
where 
- \( Y \) - input value of dependent variable
- \( A \) - Doublet array of \( X,Y \) pairs
- \( X \) - output value of independent variable
5.0 SAMPLE PROBLEM

A sample problem was prepared for the SINDA routine to demonstrate the input and output for a typical thermal/flow analysis problem. A schematic of the problem is shown in Figure 10. The problem consists of 8 two dimensional radiator panels, each modeled by two flow paths (one for the main panel of 11 tubes and one for the prime bypass tube). Contained in the system are a pump, a bypass valve (valve No. 1) and a stagnation valve between the two flow paths. The heat load to the radiator system comes through a counter flow heat exchanger which has a controlled inlet temperature of 40°F. The fluid is Freon 21 in the radiator system and water on the cooled side of the heat exchanger. The nodal subdivision for the fluid system is shown in Figure 10. The structure nodal subdivision is shown in Figure 11.

The output for the run is presented in Table 3. A selected few items were plotted using the plot package described in Appendix B. These are presented in Figures 12 thru 17.
xx - Fluid temperature lumps
X - Pressure Nodes
□ - Tube numbers
vx - Valve numbers
FIGURE 11 STRUCTURE MODEL FOR THE SAMPLE PROBLEM
| ASG G = 505962 | 25 Feb 73 | 19:19:46 |
| ASG Z = 505743 | 25 Feb 73 | 19:19:46 |
| ASG T = 502293 | 25 Feb 73 | 19:19:46 |
| ASG 0,3,4,5,6 | 25 Feb 73 | 19:19:46 |
STARTING ADDRESS 014000
CORE LIMITS 014000 047745 050366 163771 163772 163777

PREPRO/CODE
0 050366-050425
1 014000-014454

PROTS /CLEC4
0 050426-050432
1 014455-014461
2 050433-050450

RIDING /CLEC5
1 015460-015531
2 050451-050501

MTABS /CODE
0 050502-050569

MPATS /CLEC4
1 015532-015617
2 050561-050665

NCNRTS/RECS
1 016470-016794
2 050666-050759

NOTICG/RECS
1 016715-017369
2 050760-051020

PACKS/CODE
1 017365-017430

DEPTH /******
0 051021-051026

RFYE5 /RECS
1 017431-017453

RDFV5 /CLEC4
0 051027-051213
SAMPLE PROBLEM FOR SINDA VERSION 9

WINPS/RECS
0 051214-051216
1 017649-020571
2 051217-051259

WINDES/RECS
1 020572-020727
2 051255-051292

WINPS/RECS
0 051305-051305
1 020730-021235
2 051306-051402

WINDES/RECS
0 051403-051572
1 021236-021700

NFOUTS/RECS
1 021701-022132
2 051573-051579

NBUFF/R23
1 022133-022155
2 051575-052605

HOMED/RLECS
1 022156-022190

RUNL/R CODE
0 052608-052659
1 022191-022233

RCSTS/RECS
1 022234-022234

REZS/R CODE
0 052651-052705
1 022355-022365

TITLE/R CODE
0 052706-053006
1 022366-022445

CREDLE/******
0 053007-054023

TAPE/******
0 054024-054035

BLENFS/******
0 054036-055105

TABCON/******
0 055106-055375

DATA/******
0 055376-055417
SAMPLE PROBLEM FOR SINDA VERSION 9

LOGIC /*****
 0 05420-05519
 1 05520-05526
 2 055515-055526

SRDCOM/******
 0 055227-056130

BUCKET/******
 0 056131-154500

POINT /******/
 0 154501-154575

CHECK/******
 0 154576-154747

FLAGS /******/
 0 154750-154752

JPS /******/
 0 154753-154753

CIPAGE/******
 0 154754-157333

RFIPS/PLECS
 1 022496-022707
 2 157334-157334

SEARCH/ CODE
 0 157335-157351
 1 022710-022774

MaTOF/ CODE
 1 022876-023331

BLKCD/ CODE
 0 157352-157360
 1 023434-024327

WATBLK/ CODE
 1 024330-024341

STIFB/ CODE
 0 157361-157433
 1 024342-024371

FINDCM/ CODE
 0 157444-157717
 1 024732-024723

SQUEEZ/ CODE
 0 157720-157752
 1 024724-025015

98
SAMPLE PROBLEM FOR SINDA VERSION 9

SREADC/CODE
5 157743-160035
1 025014-029350

ZREADC/CODE
5 160036-160055
1 025501-025501

RDLES/CODE
5 160056-160070
1 025502-025610

MERRS/CODE
5 160071-160132
1 025641-025631

NTRAM/CODE
5 160133-160137
1 025632-025729

EURAN/CODE
5 160138-160149
1 025725-026004

**EDIT** /CODE
5 160149-161054
1 026005-027374

**MODUR** /CODE
5 160345-160354
1 027065-027043

BUFFAP/CODE
5 160355-160377
1 027064-027223

BUFFACF/CODE
5 160378-160394
1 027214-027306

TOCENT/CODE
5 160327-160350
1 027307-027314

FINDOR/CODE
5 160351-160357
1 027355-027613

PPCH/CODE
5 160365-160372
1 027614-027641

ZILUS/CODE
5 160373-160452
1 027642-027746

TODCF/CODE
5 160451-160542

DATE 250273 PAGE 4
SAMPLE PROBLEM FOR SINDA VERSION 9

DATE 250273  PAGE  5

TOCRIT/COCE
0  160463-160571
1  030036-030151

ROSET /CODE
0  160572-160601
3  030152-030223

MURRT/CODE
0  160602-160646
1  030224-030310

EDITIN/COCE
0  160647-160683
1  030311-030430

KCC601/COCE
0  160684-160705

ETOD6 /CODE
1  030431-030465

CLOCK /CODE
0  160706-160710
1  030466-030466

NEWMOD/COCE
0  160711-161040
1  030547-030547

UPDAT2/COCE
0  161041-161130
1  032046-032046

UPD11/COCE
0  161131-161217
1  032047-032606

MWRT6/COCE
0  161218-161275
1  033614-034043

COPPAS/COCE
0  161276-161415
1  034044-034275

SINDA/COCE
0  161416-161644
1  034276-034533

GREAD /CODE
0  161665-161743
1  034534-035074

STATISH/COCE
0  161744-162125
SAMPLE PROBLEM FOR SIMOA VERSION 9
DATE 250273 PAGE 4

SETFLD/CODE
G 162226-162245
I 035614-035779

MOVE /CODE
G 162246-162269
I 035775-036127

NCOL /CODE
G 162265-162271
I 036130-036150

NCH /CODE
G 162272-162302
I 036151-026212

BUFFLK/******
G 162303-163276

PONERW/******
G 163277-163373

RURCAD/******
G 163374-163400

**GENLINK/CODE
G 163401-160267
I 026005-026316

**PSEUDO/ CODE
G 160145-160271
I 026005-030131

**PCSZ /CODE
G 160272-160304
I 030132-030177

**CODE /CODE
G 160145-160426
I 026005-030400

**EMDED /CODE
G 160427-160529
I 030411-031107

**DATARD /CODE
G 160531-161407
I 031110-035727

**ERAMES/ CODE
G 161408-162757
I 035729-024611

**CONVRT/ CODE
G 162758-162785
I 036612-036737
SAMPLE PROBLEM FOR SINDA VERSION 9

**TYPE/CODE**

G 163016-163036
I 047966-037959

**DATA/CODE**

G 163037-163053
I 037156-047045

**RELAY/CODE**

G 163159-163205
I 040146-041226

**WIRE/CODE**

G 163206-163227
I 041227-042064

**INRPT/CODE**

G 163230-163246
I 042065-042442

**SWAP/CODE**

G 163247-163266
I 042443-042954

**INCORE/CODE**

G 163267-163316
I 043255-043350

**SETFMT/CODE**

G 163317-163346
I 043351-043937

**GENX/CODE**

G 163417-163473
I 043546-044347

**MINDS/CODE**

G 163474-163562
I 044150-047235

**CONS /CODE**

G 163474-163613
I 044150-047745

**RESUP/CODE**

G 165145-165223
I 026613-026133

**SINDA/CODE**

G 165224-165113
I 026035-025516

**STGFA/CODE**

G 165137-165224
I 025917-031257

**LPINT/CODE**
SAMPLE PROBLEM FOR SINDA VERSION 9
0 161227-161497
1 031210-031712
**SPLIT /CODE
0 160149-160170
1 026605-026770
**SKIP /CODE
0 160171-160216
1 026771-027301

END OF ALLOCATION 1103 0039A 09099
A description of the modifications to each version of SINDA is contained on comment cards in subroutine NEWS.
SAMPLE PROBLEM FOR SINDA VERSION 9

*SINDA

MODEL C 2

BEGIN SINDA PREPROCESSOR, VERSION 8, AT 19:15:03

DATE 250273 PAGE 10
**SAMPLE PROBLEM FOR SINDA VERSION 9**

**BCD THERMAL LPCS**

**BCD 6 SAMPLE PROBLEM FOR SINDA VERSION 9**

**END**

**BCD NODE DATA**

REM NODE, NOD, INC, TI, REM, I, CONST

<table>
<thead>
<tr>
<th>NODE</th>
<th>AX</th>
<th>BX</th>
<th>CX</th>
<th>DX</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>6</td>
<td>70</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>6</td>
<td>70</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>6</td>
<td>70</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>6</td>
<td>70</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>6</td>
<td>70</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>6</td>
<td>70</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>6</td>
<td>70</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>6</td>
<td>70</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>0</td>
<td>6</td>
<td>70</td>
<td>0</td>
</tr>
</tbody>
</table>

**FLUID LPCS**

<table>
<thead>
<tr>
<th>NODE</th>
<th>AX</th>
<th>BX</th>
<th>CX</th>
<th>DX</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>0</td>
<td>6</td>
<td>70</td>
<td>0</td>
</tr>
<tr>
<td>101</td>
<td>0</td>
<td>6</td>
<td>70</td>
<td>0</td>
</tr>
<tr>
<td>102</td>
<td>0</td>
<td>6</td>
<td>70</td>
<td>0</td>
</tr>
<tr>
<td>103</td>
<td>0</td>
<td>6</td>
<td>70</td>
<td>0</td>
</tr>
<tr>
<td>104</td>
<td>0</td>
<td>6</td>
<td>70</td>
<td>0</td>
</tr>
<tr>
<td>105</td>
<td>0</td>
<td>6</td>
<td>70</td>
<td>0</td>
</tr>
<tr>
<td>106</td>
<td>0</td>
<td>6</td>
<td>70</td>
<td>0</td>
</tr>
<tr>
<td>107</td>
<td>0</td>
<td>6</td>
<td>70</td>
<td>0</td>
</tr>
<tr>
<td>108</td>
<td>0</td>
<td>6</td>
<td>70</td>
<td>0</td>
</tr>
<tr>
<td>109</td>
<td>0</td>
<td>6</td>
<td>70</td>
<td>0</td>
</tr>
</tbody>
</table>

**TUBE LPCS**

<table>
<thead>
<tr>
<th>NODE</th>
<th>AX</th>
<th>BX</th>
<th>CX</th>
<th>DX</th>
</tr>
</thead>
<tbody>
<tr>
<td>110</td>
<td>0</td>
<td>6</td>
<td>70</td>
<td>0</td>
</tr>
<tr>
<td>111</td>
<td>0</td>
<td>6</td>
<td>70</td>
<td>0</td>
</tr>
<tr>
<td>112</td>
<td>0</td>
<td>6</td>
<td>70</td>
<td>0</td>
</tr>
<tr>
<td>113</td>
<td>0</td>
<td>6</td>
<td>70</td>
<td>0</td>
</tr>
<tr>
<td>114</td>
<td>0</td>
<td>6</td>
<td>70</td>
<td>0</td>
</tr>
<tr>
<td>115</td>
<td>0</td>
<td>6</td>
<td>70</td>
<td>0</td>
</tr>
<tr>
<td>116</td>
<td>0</td>
<td>6</td>
<td>70</td>
<td>0</td>
</tr>
<tr>
<td>117</td>
<td>0</td>
<td>6</td>
<td>70</td>
<td>0</td>
</tr>
</tbody>
</table>

**SAMPLE PROBLEM FOR SINDA VERSION 9**

**DATE 250273 PAGE 11**

106
SAMPLE PROBLEM FOR SINDA VERSION 9

SIV 299 .70 . 1.20 .
SIV 300 .70 . 0.199 .
SIV 301 .70 . 0.199 .
SIV 302 .70 . 0.199 .
SIV 303 .70 . 0.199 .
SIV 304 .70 . 0.199 .
SIV 305 .70 . 0.199 .
SIV 306 .70 . 0.199 .
SIV 307 .70 . 0.199 .
SIV 308 .70 . 0.199 .
SIV 309 .70 . 0.199 .
SIV 310 .70 . 0.199 .
SIV 311 .70 . 0.199 .
SIV 312 .70 . 0.199 .
SIV 313 .70 . 0.199 .
SIV 314 .70 . 0.199 .
SIV 315 .70 . 0.199 .
SIV 316 .70 . 0.199 .
SIV 317 .70 . 0.199 .

END

RELATIVE NODE NUMBERS

ACTUAL NODE NUMBERS

NODE ANALYSIS... DIFFUSION = 234, ARITHMETIC = 5, BOUNDARY = 2, TOTAL = 239
SAMPLE PROBLEM FOR SINDA VERSION 9

SIT 210, A15, 16.270 s
SIT 211, A15, 16.270 s
SIT 212, A15, 16.270 s
SIT 213, A15, 16.270 s
SIT 214, A15, 16.270 s
SIT 215, A15, 16.270 s
SIT 216, A15, 16.270 s
SIT 217, A15, 16.270 s
SIT 218, A15, 16.270 s
SIT 219, A15, 16.270 s
SIT 220, A15, 16.270 s
SIT 221, A15, 16.270 s
SIT 222, A15, 16.270 s
SIT 223, A15, 16.270 s
SIT 224, A15, 16.270 s
SIT 225, A15, 16.270 s
SIT 226, A15, 16.270 s
SIT 227, A15, 16.270 s
SIT 228, A15, 16.270 s
SIT 229, A15, 16.270 s
SIT 230, A15, 16.270 s
SIT 231, A15, 16.270 s
SIT 232, A15, 16.270 s
SIT 233, A15, 16.270 s
SIT 234, A15, 16.270 s
SIT 235, A15, 16.270 s
SIT 236, A15, 16.270 s
SIT 237, A15, 16.270 s
SIT 238, A15, 16.270 s
SIT 239, A15, 16.270 s
SIT 240, A15, 16.270 s
SIT 241, A15, 16.270 s
SIT 242, A15, 16.270 s
SIT 243, A15, 16.270 s
SIT 244, A15, 16.270 s
SIT 245, A15, 16.270 s
SIT 246, A15, 16.270 s
SIT 247, A15, 16.270 s
SIT 248, A15, 0.431 s
SIT 249, A15, 0.431 s
SIT 250, A15, 0.431 s
SIT 251, A15, 0.431 s
SIT 252, A15, 0.431 s
SIT 253, A15, 0.431 s
SIT 254, A15, 0.431 s
SIT 255, A15, 0.431 s
SIT 256, A15, 0.431 s
SIT 257, A15, 0.431 s
SIT 258, A15, 0.431 s
SIT 259, A15, 0.431 s
SIT 260, A15, 0.431 s
SIT 261, A15, 0.431 s
SIT 262, A15, 0.431 s
SIT 263, A15, 0.431 s
SIT 264, A15, 0.431 s
SIT 265, A15, 0.431 s
SIT 266, A15, 0.431 s
SIT 267, A15, 0.431 s
SIT 268, A15, 0.431 s
SIT 269, A15, 0.431 s
SIT 270, A15, 0.431 s
SIT 271, A15, 0.431 s
SIT 272, A15, 0.431 s
SIT 273, A15, 0.431 s
SIT 274, A15, 0.431 s
SIT 275, A15, 0.431 s
SIT 276, A15, 0.431 s
SIT 277, A15, 0.431 s
SIT 278, A15, 0.431 s
SIT 279, A15, 0.431 s
SIT 280, A15, 0.431 s
SIT 281, A15, 0.431 s
SIT 282, A15, 0.431 s
SIT 283, A15, 0.431 s
SIT 284, A15, 0.431 s
SIT 285, A15, 0.431 s
SIT 286, A15, 0.431 s
SIT 287, A15, 0.431 s
SIT 288, A15, 0.431 s
SIT 289, A15, 0.431 s
SIT 290, A15, 0.431 s
SIT 291, A15, 0.431 s
SIT 292, A15, 0.431 s
SIT 293, A15, 0.431 s
SIT 294, A15, 0.431 s
SIT 295, A15, 0.431 s

108
Flow conductors identified in array 17, page 114

*Comments added to listing of input data to clarify input of flow systems
SAMPLE PROBLEM FOR SINDA VERSION 9

<table>
<thead>
<tr>
<th>GEN</th>
<th>201, 117, 1, 1, 1, 201, 1, 100.</th>
</tr>
</thead>
<tbody>
<tr>
<td>REM</td>
<td>NC NOG 16 RA ENA NA NID</td>
</tr>
<tr>
<td>GEN</td>
<td>-901, 8, 1, 202, 6, 400, 0, 2.59E-8</td>
</tr>
<tr>
<td>GEN</td>
<td>-909, 8, 1, 203, 6, 400, 0, 2.59E-8</td>
</tr>
<tr>
<td>GEN</td>
<td>-917, 6, 1, 204, 6, 400, 0, 2.59E-8</td>
</tr>
<tr>
<td>GEN</td>
<td>-927, 6, 1, 205, 6, 400, 0, 2.59E-8</td>
</tr>
<tr>
<td>GEN</td>
<td>-941, 8, 1, 251, 6, 400, 0, 0.64E-9</td>
</tr>
<tr>
<td>GEN</td>
<td>-947, 8, 1, 252, 6, 400, 0, 0.64E-9</td>
</tr>
<tr>
<td>GEN</td>
<td>-957, 8, 1, 253, 6, 400, 0, 0.64E-9</td>
</tr>
</tbody>
</table>

END

CONVECTION - Connection conductor identified in array 16, page 112.

RADIATION

RELATIVE CONDUCTOR NUMBERS

<table>
<thead>
<tr>
<th>1</th>
<th>THRU</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>THRU</td>
<td>20</td>
</tr>
<tr>
<td>11</td>
<td>THRU</td>
<td>30</td>
</tr>
<tr>
<td>31</td>
<td>THRU</td>
<td>40</td>
</tr>
<tr>
<td>41</td>
<td>THRU</td>
<td>50</td>
</tr>
<tr>
<td>51</td>
<td>THRU</td>
<td>60</td>
</tr>
<tr>
<td>61</td>
<td>THRU</td>
<td>70</td>
</tr>
<tr>
<td>71</td>
<td>THRU</td>
<td>80</td>
</tr>
<tr>
<td>81</td>
<td>THRU</td>
<td>90</td>
</tr>
<tr>
<td>91</td>
<td>THRU</td>
<td>100</td>
</tr>
<tr>
<td>101</td>
<td>THRU</td>
<td>110</td>
</tr>
<tr>
<td>111</td>
<td>THRU</td>
<td>120</td>
</tr>
<tr>
<td>121</td>
<td>THRU</td>
<td>130</td>
</tr>
<tr>
<td>131</td>
<td>THRU</td>
<td>140</td>
</tr>
<tr>
<td>141</td>
<td>THRU</td>
<td>150</td>
</tr>
<tr>
<td>151</td>
<td>THRU</td>
<td>160</td>
</tr>
<tr>
<td>161</td>
<td>THRU</td>
<td>170</td>
</tr>
<tr>
<td>171</td>
<td>THRU</td>
<td>180</td>
</tr>
<tr>
<td>181</td>
<td>THRU</td>
<td>190</td>
</tr>
<tr>
<td>191</td>
<td>THRU</td>
<td>200</td>
</tr>
<tr>
<td>201</td>
<td>THRU</td>
<td>210</td>
</tr>
<tr>
<td>211</td>
<td>THRU</td>
<td>220</td>
</tr>
<tr>
<td>221</td>
<td>THRU</td>
<td>230</td>
</tr>
<tr>
<td>231</td>
<td>THRU</td>
<td>240</td>
</tr>
<tr>
<td>241</td>
<td>THRU</td>
<td>250</td>
</tr>
<tr>
<td>251</td>
<td>THRU</td>
<td>260</td>
</tr>
<tr>
<td>261</td>
<td>THRU</td>
<td>270</td>
</tr>
<tr>
<td>271</td>
<td>THRU</td>
<td>280</td>
</tr>
<tr>
<td>281</td>
<td>THRU</td>
<td>290</td>
</tr>
<tr>
<td>291</td>
<td>THRU</td>
<td>300</td>
</tr>
<tr>
<td>301</td>
<td>THRU</td>
<td>310</td>
</tr>
<tr>
<td>311</td>
<td>THRU</td>
<td>320</td>
</tr>
</tbody>
</table>

ACTUAL CONDUCTOR NUMBERS

<table>
<thead>
<tr>
<th>1</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>20</td>
<td>11</td>
<td>12</td>
<td>13</td>
<td>14</td>
<td>15</td>
<td>16</td>
<td>17</td>
<td>18</td>
<td>19</td>
</tr>
<tr>
<td>12</td>
<td>30</td>
<td>21</td>
<td>22</td>
<td>23</td>
<td>24</td>
<td>25</td>
<td>26</td>
<td>27</td>
<td>28</td>
<td>29</td>
</tr>
<tr>
<td>13</td>
<td>40</td>
<td>31</td>
<td>32</td>
<td>33</td>
<td>34</td>
<td>35</td>
<td>36</td>
<td>37</td>
<td>38</td>
<td>39</td>
</tr>
<tr>
<td>14</td>
<td>50</td>
<td>41</td>
<td>42</td>
<td>43</td>
<td>44</td>
<td>45</td>
<td>46</td>
<td>47</td>
<td>48</td>
<td>49</td>
</tr>
<tr>
<td>15</td>
<td>60</td>
<td>51</td>
<td>52</td>
<td>53</td>
<td>54</td>
<td>55</td>
<td>56</td>
<td>57</td>
<td>58</td>
<td>59</td>
</tr>
<tr>
<td>16</td>
<td>70</td>
<td>61</td>
<td>62</td>
<td>63</td>
<td>64</td>
<td>65</td>
<td>66</td>
<td>67</td>
<td>68</td>
<td>69</td>
</tr>
<tr>
<td>17</td>
<td>80</td>
<td>71</td>
<td>72</td>
<td>73</td>
<td>74</td>
<td>75</td>
<td>76</td>
<td>77</td>
<td>78</td>
<td>79</td>
</tr>
<tr>
<td>18</td>
<td>90</td>
<td>81</td>
<td>82</td>
<td>83</td>
<td>84</td>
<td>85</td>
<td>86</td>
<td>87</td>
<td>88</td>
<td>89</td>
</tr>
<tr>
<td>19</td>
<td>100</td>
<td>91</td>
<td>92</td>
<td>93</td>
<td>94</td>
<td>95</td>
<td>96</td>
<td>97</td>
<td>98</td>
<td>99</td>
</tr>
<tr>
<td>20</td>
<td>110</td>
<td>101</td>
<td>102</td>
<td>103</td>
<td>104</td>
<td>105</td>
<td>106</td>
<td>107</td>
<td>108</td>
<td>109</td>
</tr>
<tr>
<td>21</td>
<td>120</td>
<td>111</td>
<td>112</td>
<td>113</td>
<td>114</td>
<td>115</td>
<td>116</td>
<td>117</td>
<td>118</td>
<td>119</td>
</tr>
<tr>
<td>22</td>
<td>130</td>
<td>121</td>
<td>122</td>
<td>123</td>
<td>124</td>
<td>125</td>
<td>126</td>
<td>127</td>
<td>128</td>
<td>129</td>
</tr>
<tr>
<td>23</td>
<td>140</td>
<td>131</td>
<td>140</td>
<td>160</td>
<td>161</td>
<td>170</td>
<td>171</td>
<td>180</td>
<td>190</td>
<td>200</td>
</tr>
<tr>
<td>24</td>
<td>150</td>
<td>210</td>
<td>211</td>
<td>212</td>
<td>213</td>
<td>214</td>
<td>215</td>
<td>216</td>
<td>217</td>
<td>218</td>
</tr>
<tr>
<td>25</td>
<td>160</td>
<td>220</td>
<td>221</td>
<td>222</td>
<td>223</td>
<td>224</td>
<td>225</td>
<td>226</td>
<td>227</td>
<td>228</td>
</tr>
<tr>
<td>26</td>
<td>170</td>
<td>230</td>
<td>231</td>
<td>232</td>
<td>233</td>
<td>234</td>
<td>235</td>
<td>236</td>
<td>237</td>
<td>238</td>
</tr>
<tr>
<td>27</td>
<td>180</td>
<td>240</td>
<td>241</td>
<td>242</td>
<td>243</td>
<td>244</td>
<td>245</td>
<td>246</td>
<td>247</td>
<td>248</td>
</tr>
<tr>
<td>28</td>
<td>190</td>
<td>250</td>
<td>251</td>
<td>252</td>
<td>253</td>
<td>254</td>
<td>255</td>
<td>256</td>
<td>257</td>
<td>258</td>
</tr>
<tr>
<td>29</td>
<td>200</td>
<td>260</td>
<td>261</td>
<td>262</td>
<td>263</td>
<td>264</td>
<td>265</td>
<td>266</td>
<td>267</td>
<td>268</td>
</tr>
<tr>
<td>30</td>
<td>210</td>
<td>270</td>
<td>271</td>
<td>272</td>
<td>273</td>
<td>274</td>
<td>275</td>
<td>276</td>
<td>277</td>
<td>278</td>
</tr>
<tr>
<td>31</td>
<td>220</td>
<td>280</td>
<td>281</td>
<td>282</td>
<td>283</td>
<td>284</td>
<td>285</td>
<td>286</td>
<td>287</td>
<td>288</td>
</tr>
<tr>
<td>32</td>
<td>230</td>
<td>290</td>
<td>291</td>
<td>292</td>
<td>293</td>
<td>294</td>
<td>295</td>
<td>296</td>
<td>297</td>
<td>298</td>
</tr>
<tr>
<td>33</td>
<td>240</td>
<td>300</td>
<td>301</td>
<td>302</td>
<td>303</td>
<td>304</td>
<td>305</td>
<td>306</td>
<td>307</td>
<td>308</td>
</tr>
<tr>
<td>34</td>
<td>250</td>
<td>310</td>
<td>311</td>
<td>312</td>
<td>313</td>
<td>314</td>
<td>315</td>
<td>316</td>
<td>317</td>
<td>318</td>
</tr>
<tr>
<td>35</td>
<td>260</td>
<td>320</td>
<td>321</td>
<td>322</td>
<td>323</td>
<td>324</td>
<td>325</td>
<td>326</td>
<td>327</td>
<td>328</td>
</tr>
<tr>
<td>36</td>
<td>270</td>
<td>330</td>
<td>331</td>
<td>332</td>
<td>333</td>
<td>334</td>
<td>335</td>
<td>336</td>
<td>337</td>
<td>338</td>
</tr>
<tr>
<td>37</td>
<td>280</td>
<td>340</td>
<td>341</td>
<td>342</td>
<td>343</td>
<td>344</td>
<td>345</td>
<td>346</td>
<td>347</td>
<td>348</td>
</tr>
<tr>
<td>38</td>
<td>290</td>
<td>350</td>
<td>351</td>
<td>352</td>
<td>353</td>
<td>354</td>
<td>355</td>
<td>356</td>
<td>357</td>
<td>358</td>
</tr>
<tr>
<td>39</td>
<td>300</td>
<td>360</td>
<td>361</td>
<td>362</td>
<td>363</td>
<td>364</td>
<td>365</td>
<td>366</td>
<td>367</td>
<td>368</td>
</tr>
</tbody>
</table>

110
SAMPLE PROBLEM FOR SINDA VERSION 9

CONDUCTOR ANALYSIS... LINEAR = 248, RADIATION = 69, TOTAL = 312, CONNECTIONS = 312

BCD CONSTANTS DATA

TIMEEND = 3.0
DIPTIL = 0.01
NOCTP = 100
ALPHA = 0.01
ALPHA = 0.01
OUTPUT = 1.0

END

CONSTANTS ANALYSIS... CSER = 1, ADED = 46 0 196, TOTAL = 131

BCD ARRAY DATA

1 FRIENE-21 SPECIFIC HEAT

-400. -223 -210. -223 -217. 3.723
-212. -210. -223 -217. 3.723
-110. .220 -60. .231 0. .237
-10. .244 -90. .235 120. 234
190. .274 -150. .280 100. 2395
246. .315

END

2 FRIENE-21 DENSITY

-400. 110. -211. 110. -217. 110.
-212. 110. -211. 110. -160. 104.
-110. 99.25 -60. 95. 0. 91.5
40. 88.5 90. 84.2 120. 81.0
140. 80.1 150. 79.9 180. 76.
246. 69.

END

3 FRIENE-21 DENSITY TIMES SPECIFIC HEAT

-400. 24.53 -211. 24.53 -217. 409.93
-212. 409.93 -211. 24.53 -160. 23.30
-110. 22.23 -60. 22.18 0. 21.39
-10. 21.59 -90. 21.39 120. 21.60
140. 21.95 -150. 22.37 180. 22.42
246. 21.73

END

4 ALUMINUM SPECIFIC HEAT

-400. .092 -300. .124 -200. .152
-100. .175 -0. .192 -100. .204
200. .214

END

5 FRIENE-21 VISCOSITY

-400. 19.1 -212. 19.1 -211. 19.1
-200. 18.5 -200. 16.55 -200. 14.75
-200. 13.7 -194. 11.5 -191. 10.8
-100. 10.02 -194. 9.25 -178. 8.31
-100. 7.12 -165. 6.36 -150. 5.72
-100. 5.21 -146. 4.75 -142. 4.32
-100. 3.96 -135. 3.68 -124. 3.42
-100. 3.16 -112. 2.81 -76. 2.02
-100. 1.62 -0. 1.17 35. .994
60. .870 100. .726 140. .551
260. .396

END

6 FRIENE-21 THERMAL CONDUCTIVITY

-400. 0.14 0.0 0.975 250. 0.935
ADAT array for convection conductors; described on page 49 and the last argument to subroutine CONV1 on page 121
SAMPLE PROBLEM FOR SINDA VERSION 9

ADAT array (cont'd)

ADAT1 array described on page 53 and the third argument to FLOCONS on page 121

FLOW CONDUCTOR DATA
ADAT1 array (cont'd)

AFLOW array described on page 71 and the first argument in the call to PFCS on page 121 (also first argument to CONV1)

AW array described on page 72 and referenced in array 20 above

APN array described on page 72

AGF array described on page 72
SAMPLE PROBLEM FOR SINDA VERSION 9

END
24 6 VALVE POSITIONS
0.999, 0.99

END
25 6 IMPOSED FLOW RATES
200., SPACE, 23

END
26 6 FLUID TYPE DATA--WP, CSA, FLL, MFF, NHL, PPC
REM WP
CSA
FLL
MFF
NHL
PPC
.1125 .001000 12. 0. 0. 1. 4 TYPE 1
.3600 .001000 3.25 0.117, 1. 4 TYPE 2
.1125 .001000 5. 0. 0. 1. 3 TYPE 3
.0320 .003E-4 0.25 0.249, 1. 5 TYPE 4
.1125 .001000 20. 0. 0. 1. 5 TYPE 5
.1125 .001000 2.5 0. 0. 1. 6 TYPE 6
.1125 .001000 50. 0. 0. 1. 6 TYPE 7
.1125 .001000 7. 0. 0. 1. 6 TYPE 8
.1125 .001000 2. 0. 0. 1. 6 TYPE 9

END
27 6 ADDRESSED RESISTANCES
SPACE, 39

END
28 6 PRESSURE DROPS
SPACE, 39

END
29 6 SYSTEM ARRAYS, ADAT
#A31 6 ID ARRAY CONTAINING SYSTEM PROPERTY ID
#A32 6 ID ARRAY CONTAINING SOLUTION PARAMETERS
#A33 6 ID ARRAY CONTAINING MAIN NETWORK
#A36 6 ID ARRAY CONTAINING ID OF VALVE DATA
#A35 6 ID ARRAY CONTAINING PUMP DATA
0 6 CHECKOUT PRINT CODE

END
31
#A1 6 EP ARRAY
#A2 6 R3 ARRAY
#A5 6 FV ARRAY
#A6 6 BT ARRAY
#A24312000. 6 GC

END
32 6 ASOL SOLUTION PARAMETERS
0.01, 100, 0.0, 0.7 6 TOL, MPS, EPS, FADF

END
33
#A1 6 NAME
#A34 6 ARRAY ID PRESS NODES WP SPECIFIED
#A35 6 ARRAY ID ARRAYS CONTAINING VALVE DATA
REM TUBE FROM, TO, SS, FLAT 6
REM .AP-ND-P-NO. K3, LLTP 6C
1, 1, 2, 1, #A37 6 MAIN NETWORK
2, 2, 3, 1, #A38
3, 3, 4, 1, #A39
30, 4, 17, 1, #A40 6 SUBNETWORK 1
26, 17, 16, 1, #A41
11, 3, 9, 1, #A42
19, 9, 22, 1, #A43 6 SUBNETWORK 2
20, 27, 18, 1, #A44
35, 38, 23, 1, #A45

117
SAMPLE PROBLEM FOR SINDA VERSION 9

36. 2, 23, 1.4A6
37. 23, 24, 1.4A7

END

32, 29, END 6 NODES WNSPECIFIED PRESSURES

35, 1, 24, 1A13. END 6 ARRAYS CONTAINING VALVE DATA

36. *A90, *A99, END 6 ARRAYS CONTAINING VALVE DATA

ANPNS array described on p. 75 & identified in array 33

AP, array containing pump data described on page 77 and identified

AVLS and AVL array in array 30 described on page 76 and referenced in array 30(AVLS) and
array 33(AVL)

ADi arrays for the main network; described on page 75 and referenced in array 33

AVLV arrays containing valve data described on page 76 and referenced in the AVL array 36

ANET array for subnetwork 1; described on page 74 and identified in ANET for the main network, array 33

ADi arrays for subnetwork No. 1 described on page 75 and referenced in array 40
ADi arrays for subnetwork No. 1 (Cont'd)

ANET array for subnetwork 2; described on page and identified in ANET for the main network, array 33
SAMPLE PROBLEM FOR SINDA VERSION 9

AD1 array for subnetwork No. 2 described on page 75 and referenced in array 43.
SAMPLE PROBLEM FOR SIMCA VERSION 9

PR38LER FOR SIMCA VERSION 9

OATE 250273 PAGE 26

140, 0113, 8, 0, T113, END 0 TUBE 33

BCD 9SHUTTLE ECS RADIATOR FLOW SYSTEM
END

BCD 9FLOW RATES (LB/HR)
END

BCD 9PRESSURES (LB/FT**2)
END

BCD 9VALVE POSITIONS
END

BCD 9PRESSURE DROPS (LB/FT**2)
END

END

ARRAY ANALYSIS... NUMBER OF ARRAYS = 75 TOTAL LENGTH = 2511

BCD EXECUTION

DIMENSION T(2000)
NOIM = 2003

PFCS(A20,A30,A100+1)
CNBACK

END

BCD VARIABLES 1

BEGD(TIPEH, A11, T198)
HSEFF(0.9,500., A21, 37, 1.0, A1, T198, T117, T199, T200)
FLOCK(A21, A1, A17)
CONV(A20, A31, A16)

END

BCD VARIABLES 2

PFCS(A20,A30,A100+1)
TIMCHK(K1,0)
NSTRY(A22,A24,A21,DTIMEU)

END

BCD OUTPUT CALLS

TPRINT

FLPRINT(A21,A11+1)
FLPRINT(A20,A11+1)
FLPRINT(A22,A11+1)
FLPRINT(A24,A112+1)
TIMCHK(K1,1)

END
**Sample Problem for Sinda Version 9**

*Main Program*

**Storage Used:**
- CODE(1) 000014
- DATA(0) 0000001
- BLANK COMPO(0) 000000

**Common Blocks:**

<table>
<thead>
<tr>
<th>COMMON BLOCKS:</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>TITLE</td>
<td>000024</td>
</tr>
<tr>
<td>TEFP</td>
<td>000036</td>
</tr>
<tr>
<td>CAP</td>
<td>000032</td>
</tr>
<tr>
<td>SOURCE</td>
<td>000032</td>
</tr>
<tr>
<td>COND</td>
<td>000040</td>
</tr>
<tr>
<td>RC1</td>
<td>000055</td>
</tr>
<tr>
<td>RC2</td>
<td>000052</td>
</tr>
<tr>
<td>RC3</td>
<td>000203</td>
</tr>
<tr>
<td>ARRAY</td>
<td></td>
</tr>
<tr>
<td>TITLE</td>
<td>000024</td>
</tr>
<tr>
<td>TERP</td>
<td>000356</td>
</tr>
<tr>
<td>CAP</td>
<td>000352</td>
</tr>
<tr>
<td>SOURCE</td>
<td>000352</td>
</tr>
<tr>
<td>COND</td>
<td>000470</td>
</tr>
<tr>
<td>RC1</td>
<td>000655</td>
</tr>
<tr>
<td>RC2</td>
<td>000452</td>
</tr>
<tr>
<td>PC</td>
<td></td>
</tr>
<tr>
<td>COMMON BLOCKS:</td>
<td></td>
</tr>
</tbody>
</table>

**External References (Block, Name):**

- INPUT
- EXECIN
- WSTOPC

**Storage Assignment (Block, Type, Relative Location, Name):**

<table>
<thead>
<tr>
<th>STORAGE ASSIGNMENT (BLOCK, TYPE, RELATIVE LOCATION, NAME)</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>G0005 IL</td>
<td>0013 000000 A 0014 000022 ARLICA 0014 000035 ARLIC3 0014 000012 ARPPFA</td>
</tr>
<tr>
<td>G0017 ATRPPFA</td>
<td>0014 000013 BACKUP 0014 000040 RALENA 0014 000035 C 0014 000035 C3PFA</td>
</tr>
<tr>
<td>G0026 CSPAR</td>
<td>0014 000020 CSPAR 0014 000227 CSPARAL 0014 000200 CSPARCL 0014 000012 C3PFA</td>
</tr>
<tr>
<td>G0041 DAPPD</td>
<td>0014 000031 D3LICA 0014 000032 D3LYCC 0014 000017 D3PPC 0014 000017 D3PPC</td>
</tr>
<tr>
<td>G0042 D3PEI</td>
<td>0014 000020 D3PEI 0014 000016 D3PPC 0014 000032 D3PPC</td>
</tr>
<tr>
<td>G0046 G</td>
<td>0014 000005 M 0014 000096 L3EST 0014 000047 L3EST 0014 000000 K</td>
</tr>
<tr>
<td>G0050 KTEST</td>
<td>0014 000050 K3EST 0014 000050 L3EST 0014 000050 L3EST 0014 000050 L3EST 0014 000050 L3EST</td>
</tr>
<tr>
<td>G0091 LENA</td>
<td>0014 000035 L3EST 0014 000000 L3EST 0014 000000 L3EST 0014 000000 L3EST</td>
</tr>
<tr>
<td>G0096 L321</td>
<td>0014 000000 L3EST 0014 000000 L3EST 0014 000000 L3EST 0014 000000 L3EST</td>
</tr>
<tr>
<td>G0097 L351</td>
<td>0014 000000 L3EST 0014 000000 L3EST 0014 000000 L3EST 0014 000000 L3EST</td>
</tr>
<tr>
<td>G0098 L361</td>
<td>0014 000000 L3EST 0014 000000 L3EST 0014 000000 L3EST 0014 000000 L3EST</td>
</tr>
<tr>
<td>G0099 L371</td>
<td>0014 000000 L3EST 0014 000000 L3EST 0014 000000 L3EST 0014 000000 L3EST</td>
</tr>
</tbody>
</table>

**Other Codes:**

<table>
<thead>
<tr>
<th>OTHER CODES:</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>CODE1</td>
<td>1= COMMON SEFAILS</td>
</tr>
<tr>
<td>CODE2</td>
<td>2= COMMON SEPPPA</td>
</tr>
<tr>
<td>CODE3</td>
<td>3= COMMON SEPPPA</td>
</tr>
<tr>
<td>CODE4</td>
<td>4= COMMON SEPPPA</td>
</tr>
<tr>
<td>CODE5</td>
<td>5= COMMON SEPPPA</td>
</tr>
<tr>
<td>CODE6</td>
<td>6= COMMON SEPPPA</td>
</tr>
</tbody>
</table>

123
SAMPLE PROBLEM FOR SINDA VERSION 9

00110 7*  COMMON /PC2/ SEQ2
00111 8*  COMMON /CONST/ K
00112 9*  COMMON /ARRAY/ A
00113 10* COMMON /FIXCON/ TIMEN, DTIMEU, TIPEND, CSGFAC,
00113 11* IRLCP, DMPCPA, OPEFTR, OPEIM, Dphia ,
00113 12* IDMPPO, DMPPC, BACKUP, TIPED, TIPER,
00113 13* IDTPECC, ATPECC, ATPEP, OUTPUT, RALSCA,
00113 14* ILOCPY, DTIMEL, DTIMEI, CSGRAY, CSGRL,
00113 15* ICSSRL, GALYCA, GALYCC, LINCT, PASECT,
00113 16* IARLCC, LSPCC, ENGBAL, BALENG, NOCOPY,
00113 17* INCSSM, MOTPFC, RARLCC, NARLPC, NTEST,
00113 18* IJTEST, RTTEST, LTEST, MTEST, RTEST,
00113 19* ISETEST, IETEST, UTEST, VTEST, LAIFAC,
00113 20* Iancellor,
00114 21* COMMON /DIMENS/ NID, NNA, NNT, NCT, NAT, LSG1, LSO2, LENA
00115 22* DIMENSION T(238), C(234), O(239), G(312), K(131), A(25111,
00115 23* ISEQI(429), I5E02(298)
00116 24* LLOAD = .TRUE.
00116 25* LCOND = .TRUE.
00116 26* LCONST = .TRUE.
00116 27* LARRAY = .TRUE.
00116 28* I CALL INPUT
00116 29* I CALL EXECUT
00116 30* GO TO 1
00119 32* 
00120 33* END

END OF COMPILATION: NO DIAGNOSTICS.
SUBROUTINE EXECN  ENTRY POINT 000021

STORAGE USED: CODE(13) 000023; DATA(9) 000005; BLANK COMMON(2) 000000

COMMON BLOCKS:

<table>
<thead>
<tr>
<th>COMMON BLOCK</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>0001 TITLE</td>
<td>000024</td>
</tr>
<tr>
<td>0002 TEMP</td>
<td>000001</td>
</tr>
<tr>
<td>0003 CAP</td>
<td>000001</td>
</tr>
<tr>
<td>0004 SOURCE</td>
<td>000001</td>
</tr>
<tr>
<td>0005 COND</td>
<td>000001</td>
</tr>
<tr>
<td>0006 PCI</td>
<td>000001</td>
</tr>
<tr>
<td>0007 PC2</td>
<td>000001</td>
</tr>
<tr>
<td>0008 KONST</td>
<td>000001</td>
</tr>
<tr>
<td>0009 ARRAY</td>
<td>000001</td>
</tr>
<tr>
<td>0010 FIXCON</td>
<td>000042</td>
</tr>
<tr>
<td>0011 DTIPS</td>
<td>000011</td>
</tr>
<tr>
<td>0012 ESPACE</td>
<td>003722</td>
</tr>
</tbody>
</table>

EXTERNAL REFERENCES (BLOCK, NAME)

<table>
<thead>
<tr>
<th>BLOCK</th>
<th>NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>0017</td>
<td>PFFS</td>
</tr>
<tr>
<td>0020</td>
<td>CNBACK</td>
</tr>
<tr>
<td>0021</td>
<td>KERR35</td>
</tr>
</tbody>
</table>

STORAGE ASSIGNMENT (BLOCK, TYPE, RELATIVE LOCATION, NAME)

<table>
<thead>
<tr>
<th>BLOCK</th>
<th>TYPE</th>
<th>LOCATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>0013</td>
<td>A</td>
<td>000000</td>
</tr>
<tr>
<td>0014</td>
<td>000013</td>
<td>BACKUP</td>
</tr>
<tr>
<td>0015</td>
<td>000020</td>
<td>CSGNIN</td>
</tr>
<tr>
<td>0016</td>
<td>000031</td>
<td>ORLCA</td>
</tr>
<tr>
<td>0017</td>
<td>000051</td>
<td>OTIPEU</td>
</tr>
<tr>
<td>0018</td>
<td>000050</td>
<td>M</td>
</tr>
<tr>
<td>0019</td>
<td>000050</td>
<td>KTEST</td>
</tr>
<tr>
<td>0020</td>
<td>000036</td>
<td>LSPES</td>
</tr>
<tr>
<td>0021</td>
<td>000044</td>
<td>NARLCE</td>
</tr>
<tr>
<td>0022</td>
<td>000050</td>
<td>NOLM</td>
</tr>
<tr>
<td>0023</td>
<td>000050</td>
<td>KOD</td>
</tr>
<tr>
<td>0024</td>
<td>000050</td>
<td>OJEETR</td>
</tr>
<tr>
<td>0025</td>
<td>000050</td>
<td>OSEQI</td>
</tr>
<tr>
<td>0026</td>
<td>000050</td>
<td>OTIPM</td>
</tr>
<tr>
<td>0027</td>
<td>000056</td>
<td>OTIEST</td>
</tr>
</tbody>
</table>

0016 1  SUBROUTINE EXECN
0017 2  COMMON /TITLE/ H
0018 3  COMMON /STEP/ T
0019 4  COMMON /CAP/ C

DATE 250273 PAGE 30  25 FEB 73
SAMPLE PROBLEM FOR SINDA VERSION 9

DATE 250273 PAGE 31

00104 5* COMMON /SOURCE/ G
00107 6* COMMON /SCON/ G
00110 7* COMMON /PC1/ SEQ1
00111 8* COMMON /PC2/ SEQ2
00112 9* COMMON /KRONST/ K
00113 10* COMMON /ARRAY/ A
00114 11* COMMON /FIXCONS/ TIMEN, DTIMEU, DTIMEF, CSFGAC,
00115 12* COMMON /DIMENS/ I1L00P, DIPPER, IPEIPR, DIMEIP, CAMP,
00116 13* COMMON /DIMENS/ I0AMP, ATPER, BACKUP, IPEIPR, DIEM,
00117 14* COMMON /DIMENS/ I0MPCC, ATPERC, CSGMIN, OUTPUT, ARJICA,
00118 15* COMMON /DIMENS/ IPDPC, DTMEL, DIMEI, CSGRAY, CSGRAL,
00119 16* COMMON /DIMENS/ ICSEGAL, DALICA, DAULIC, LINECT, PAGECT,
00120 17* COMMON /DIMENS/ IDAR, LCSPS, LUUS, BALEW, NOTCOP,
00121 18* COMMON /DIMENS/ IJISEM, NDTAC, NARLIC, NATMPC, TTEST,
00122 19* COMMON /DIMENS/ IJTEST, KTTEST, LIEST, MTEST, RTEST,
00123 20* COMMON /DIMENS/ ISTEST, TTEST, UTEST, VTEST, LAFAC,
00124 21* COMMON /DIMENS/ ISL00P, NAL01, NAL02, L501, L502, LEN
00125 22* COMMON /DIMENS/ INM, NAL01, NAL02, L501, L502, LEN
00126 23* DIMENSION H(20)
00127 24* DIMENSION XSPACE/ H(20)
00128 25* DIMENSION H(20)
00129 26* DIMENSION XSPACE/ H(20)
00130 27* EQUIVALENCE (K(1), K(1), K(1), K(1), K(1), K(1))
00131 28* DIMENSION X(200)
00132 29* NOTM = 2000
00133 30* NOTM = 0
00134 31* CALL FCFS (A(1057), A(1051), A(2463))
00135 32* CALL CHECK
00136 33* RETURN
00137 34* END

END OF COMPILATION: NO DIAGNOSTICS.
SAMPLE PROBLEM FOR SINCA VERSION 1

FOR A VARBL1
UNIVAC 1108 FORTRAN V EXEC III LEVEL 25A - (EXEC III LEVEL E1201010A)
THIS COMPILATION WAS DONE ON 25 FEB 73 AT 14:16:09

SUBROUTINE VARBL1 ENTRY POINT 000091

STORAGE USED: CODE(1) 000093; DATA(0) 000010; BLANK COMMON(2) 000000

COMMON BLOCKS:
0003 TITLE 000024
0004 TEMP 000001
0005 CAP 000001
0006 SOURCE 000001
0007 CONT 000001
0010 FL 000001
0011 FC2 000001
0012 KONST 000001
0013 ARAY 000001
0014 FLDCOM 000002
0015 DIFENG 000011
0016 ISPACE 000003

EXTERNAL REFERENCES (BLOCK, NAME)
0017 BIDEC1
0020 WIFE
0021 MSMP1
0022 CVN1
0023 RERR3

STORAGE ASSIGNMENT (BLOCK, TYPE, RELATIVE LOCATION, NAME)
0003 M 000000 A 0010 000022 ARLXCA 0014 000055 ARLXCC 0014 000017 ATRFCC
0014 000013 BACKUP 0014 000040 BALENG 0005 000000 C 0014 000012 CTRAPC 0014 000024 CTRLX
0014 000050 CSMIN 0014 000027 CSSICAL 0014 000020 CSSICAL 0014 000012 CTRAPC 0014 000024 CTRLX
0014 000050 DRLXCA 0014 000032 DRUXCC 0014 000007 CTRLX 0014 000012 CTRAPC 0014 000024 CTRLX
0014 000050 DTIFED 0014 000055 DTPICA 0014 000016 DTPICA 0014 000012 CTRAPC 0014 000024 CTRLX
0014 000050 KTEST 0014 000060 KATFAC 0015 000010 LENA 0014 000012 CTRAPC 0014 000024 CTRLX
0014 000050 LSPCS 0015 000056 LS01 0015 000007 LS02 0014 000012 CTRAPC 0014 000024 CTRLX
0014 000050 VARLFC 0015 000056 NAT 0014 000045 NAPPPC 0014 000012 CTRAPC 0014 000024 CTRLX
0014 000050 ADIM 0014 000040 ADFPPC 0015 000000 ADEPC 0014 000012 CTRAPC 0014 000024 CTRLX
0014 000050 AARE 0015 000052 ANT 0014 000041 NANTY 0014 000012 CTRAPC 0014 000024 CTRLX
0014 000050 UPEITH 0014 000050 UPLSPC 0014 000024 CTRAPC 0014 000012 CTRAPC 0014 000024 CTRLX
0010 000050 SEQ1 0011 000050 SEQ2 0014 000014 GSTEST 0014 000026 GSTEST 0014 000034 GTEST
0014 000015 TIFEN 0014 R 000000 TIFEN 0014 000000 TIFEN 0014 000000 TIFEN 0014 000000 TIFEN
0014 000050 UTEST 0014 000055 UTEST 0016 000002 X 0012 000000 W.

00031 1* SUBROUTINE VARBL1
00032 2* COMMON / TITLE/ W
SAMPLE PROBLEM FOR SINDA VERSION 9

DATE 250273 PAGE 33

COMMON /STEPP/ T
COMMON /CAPC/ G
COMMON /SOURCE/ G
COMMON /COND/ C
COMMON /PRC/ SEQ1
COMMON /PRC/ SEQ2
COMMON /KINST/ K
COMMON /ARRAY/ A
COMMON /TIME/, DTIMEU, TIMEND, ESSPAC,
COMMON /
1

COMMON /STIMEP/, STIMEE, STIMEC, STIMEB, SAMP.

COMMON /DEPP/ ATPPC, ATPPC, ATPPC, ATPPC, ATPPC,

COMMON /DEPP/ ATPPC, ATPPC, ATPPC, ATPPC, ATPPC,

COMMON /DEPP/ ATPPC, ATPPC, ATPPC, ATPPC, ATPPC,

COMMON /DEPP/ ATPPC, ATPPC, ATPPC, ATPPC, ATPPC,

COMMON /DEPP/ ATPPC, ATPPC, ATPPC, ATPPC, ATPPC,

COMMON /DEPP/ ATPPC, ATPPC, ATPPC, ATPPC, ATPPC,

COMMON /DEPP/ ATPPC, ATPPC, ATPPC, ATPPC, ATPPC,

COMMON /DEPP/ ATPPC, ATPPC, ATPPC, ATPPC, ATPPC,

COMMON /DEPP/ ATPPC, ATPPC, ATPPC, ATPPC, ATPPC,

COMMON /DEPP/ ATPPC, ATPPC, ATPPC, ATPPC, ATPPC,

COMMON /DEPP/ ATPPC, ATPPC, ATPPC, ATPPC, ATPPC,

COMMON /DEPP/ ATPPC, ATPPC, ATPPC, ATPPC, ATPPC,

COMMON /DEPP/ ATPPC, ATPPC, ATPPC, ATPPC, ATPPC,

COMMON /DEPP/ ATPPC, ATPPC, ATPPC, ATPPC, ATPPC,

COMMON /DEPP/ ATPPC, ATPPC, ATPPC, ATPPC, ATPPC,

COMMON /DEPP/ ATPPC, ATPPC, ATPPC, ATPPC, ATPPC,

COMMON /DEPP/ ATPPC, ATPPC, ATPPC, ATPPC, ATPPC,

COMMON /DEPP/ ATPPC, ATPPC, ATPPC, ATPPC, ATPPC,

COMMON /DEPP/ ATPPC, ATPPC, ATPPC, ATPPC, ATPPC,

COMMON /DEPP/ ATPPC, ATPPC, ATPPC, ATPPC, ATPPC,

COMMON /DEPP/ ATPPC, ATPPC, ATPPC, ATPPC, ATPPC,

COMMON /DEPP/ ATPPC, ATPPC, ATPPC, ATPPC, ATPPC,

COMMON /DEPP/ ATPPC, ATPPC, ATPPC, ATPPC, ATPPC,

COMMON /DEPP/ ATPPC, ATPPC, ATPPC, ATPPC, ATPPC,

COMMON /DEPP/ ATPPC, ATPPC, ATPPC, ATPPC, ATPPC,

COMMON /DEPP/ ATPPC, ATPPC, ATPPC, ATPPC, ATPPC,

COMMON /DEPP/ ATPPC, ATPPC, ATPPC, ATPPC, ATPPC,

COMMON /DEPP/ ATPPC, ATPPC, ATPPC, ATPPC, ATPPC,

COMMON /DEPP/ ATPPC, ATPPC, ATPPC, ATPPC, ATPPC,

COMMON /DEPP/ ATPPC, ATPPC, ATPPC, ATPPC, ATPPC,

COMMON /DEPP/ ATPPC, ATPPC, ATPPC, ATPPC, ATPPC,

COMMON /DEPP/ ATPPC, ATPPC, ATPPC, ATPPC, ATPPC,

COMMON /DEPP/ ATPPC, ATPPC, ATPPC, ATPPC, ATPPC,

COMMON /DEPP/ ATPPC, ATPPC, ATPPC, ATPPC, ATPPC,

COMMON /DEPP/ ATPPC, ATPPC, ATPPC, ATPPC, ATPPC,

COMMON /DEPP/ ATPPC, ATPPC, ATPPC, ATPPC, ATPPC,

COMMON /DEPP/ ATPPC, ATPPC, ATPPC, ATPPC, ATPPC,

COMMON /DEPP/ ATPPC, ATPPC, ATPPC, ATPPC, ATPPC,

COMMON /DEPP/ ATPPC, ATPPC, ATPPC, ATPPC, ATPPC,

COMMON /DEPP/ ATPPC, ATPPC, ATPPC, ATPPC, ATPPC,

COMMON /DEPP/ ATPPC, ATPPC, ATPPC, ATPPC, ATPPC,

COMMON /DEPP/ ATPPC, ATPPC, ATPPC, ATPPC, ATPPC,

COMMON /DEPP/ ATPPC, ATPPC, ATPPC, ATPPC, ATPPC,

COMMON /DEPP/ ATPPC, ATPPC, ATPPC, ATPPC, ATPPC,

COMMON /DEPP/ ATPPC, ATPPC, ATPPC, ATPPC, ATPPC,

COMMON /DEPP/ ATPPC, ATPPC, ATPPC, ATPPC, ATPPC,

COMMON /DEPP/ ATPPC, ATPPC, ATPPC, ATPPC, ATPPC,

COMMON /DEPP/ ATPPC, ATPPC, ATPPC, ATPPC, ATPPC,

COMMON /DEPP/ ATPPC, ATPPC, ATPPC, ATPPC, ATPPC,

COMMON /DEPP/ ATPPC, ATPPC, ATPPC, ATPPC, ATPPC,

COMMON /DEPP/ ATPPC, ATPPC, ATPPC, ATPPC, ATPPC,

COMMON /DEPP/ ATPPC, ATPPC, ATPPC, ATPPC, ATPPC,

COMMON /DEPP/ ATPPC, ATPPC, ATPPC, ATPPC, ATPPC,

COMMON /DEPP/ ATPPC, ATPPC, ATPPC, ATPPC, ATPPC,

COMMON /DEPP/ ATPPC, ATPPC, ATPPC, ATPPC, ATPPC,

COMMON /DEPP/ ATPPC, ATPPC, ATPPC, ATPPC, ATPPC,

COMMON /DEPP/ ATPPC, ATPPC, ATPPC, ATPPC, ATPPC,

COMMON /DEPP/ ATPPC, ATPPC, ATPPC, ATPPC, ATPPC,

COMMON /DEPP/ ATPPC, ATPPC, ATPPC, ATPPC, ATPPC,

COMMON /DEPP/ ATPPC, ATPPC, ATPPC, ATPPC, ATPPC,

COMMON /DEPP/ ATPPC, ATPPC, ATPPC, ATPPC, ATPPC,

COMMON /DEPP/ ATPPC, ATPPC, ATPPC, ATPPC, ATPPC,

COMMON /DEPP/ ATPPC, ATPPC, ATPPC, ATPPC, ATPPC,

COMMON /DEPP/ ATPPC, ATPPC, ATPPC, ATPPC, ATPPC,
SAMPLE PROBLEM FOR SINDA VERSION 9

00105  4* COMMON /CAPI/ C
00106  5* COMMON /SOURCE/ Q
00107  6* COMMON /COND/ G
00108  7* COMMON /PC1/ SED1
00109  8* COMMON /PC2/ SED2
00110  9* COMMON /CONST/ K
00111 10* COMMON /ARRAY/ A
00112 11* COMMON /FIXCOM/ TIMEN, DTIMEU, TIPEND, CSGFAC,
00113 12* INLTOP, DTPMCC, QYEITR, TIMEN, QYMPA,
00114 13* DMPFD, ATMPCC, BACKUP, TIPED, TIPER,
00115 14* DTPMCC, ATMPCC, CSGMIN, OUTPUT, ARLICR,
00116 15* ICOPCF, OTIPCL, OTIPER, CSGMAX, CSGRAL,
00117 16* ICGRAC, ORLNEC, ORLCC, LINFCT, PACECT,
00118 17* IARLCC, LSPCS, ENGBAL, BALENG, NOCOPY,
00119 18* INCGNP, NOTPCC, NARLIC, NATHPC, ITEST,
00120 19* IJTEST, KTEST, LTEST, MTEST, NTEST,
00121 20* IJTEST, PTEST, QTEST, RTEST,
00122 21* LTEST, RTEST, TTEST, UTEST, VTEST, LARFA,
00123 22* LEXTRAL
00124 23* COMMON /DIMENS/: ANO, ANA, MAM, NGT, NCT, NAT, LS01, LS02, LENA
00125 24* DIMENSION K(20)
00126 25* COMMON /XSPACE/: ROI1, MTH, X
00127 26* DIMENSION T(I), U(I), O(I), G(I), K(I), A(I)
00128 27* EQUIVALENCE (K, X(I), X(I))
00129 28* CALL PFCS (A(1537), A(181'), A(263))
00130 29* CALL TIMCHK(K(I), O)
00131 30* CALL HSTRY (A(1586), A(1651), A(1546), DTMEU)
00132 31* RETURN
00133 32* END

END OF COMPILATION:  NO DIAGNOSTICS.
SAMPLE PROBLEM FOR SINDA VERSION 9

FOR X OUTCAL

UNIVAC 1108 FORTRAN V EXEC II LEVEL 25A - LEVELO LEVEL E12010000

THIS COMPILATION WAS DONE ON 25 FEB 73 AT 14:16:32

SUBROUTINE OUTCAL ENTRY POINT 000035

STORAGE USED: CODE(11) 000037; DATA(0) 00006; BLANK COMMON(23) 000000

COMMON BLOCKS:
0003 TITLE 000024
0004 TEMP 000001
0005 CAP 000001
0006 SOURCE 000001
0007 COND 000001
0010 PCI 000001
0011 PC2 000001
0012 ROYST 000001
0013 ARRAY 000001
0014 FIXCON 000002
0015 DIPENS 000001
0016 SPACE 000003

EXTERNAL REFERENCES (BLOCK, NAME):
0110 TPRINT
0120 FPRINT
0121 TIMCHK
0122 NERR3

STORAGE ASSIGNMENT (BLOCK, TYPE, RELATIVE LOCATION, NAME):
0113 R 000000 H 0014 000022 ARLICA 0014 000035 ARLICC 0014 000012 ATPPCA 0014 000017 ATPPCE
0114 A 000013 BACKUP 0014 000040 DATFLNG 0005 000000 C 0014 000003 CGPCS 0014 000026 CGPAC
0114 000030 CSPCM 0014 000027 CGPRAL 0014 000037 CGPRCL 0014 000010 CPRPA 0014 000031 CPRPD
0114 000001 DIPEM 0014 000005 DIPEMA 0014 000016 DIPMCA 0014 000010 DIPPC 0014 000026 DIPFC
0114 000051 DIPMUC 0014 000052 DIPMC 0014 000036 DIPNC 0014 000014 DIPPC 0014 000014 DIPPC
0114 000000 DIPMUC 0014 000000 DIPPC 0014 000000 DIPPC 0014 000000 DIPPC 0014 000000 DIPPC
0114 000000 DIPMUC 0014 000000 DIPPC 0014 000000 DIPPC 0014 000000 DIPPC 0014 000000 DIPPC
0114 000000 DIPMUC 0014 000000 DIPPC 0014 000000 DIPPC 0014 000000 DIPPC 0014 000000 DIPPC
0114 000000 DIPMUC 0014 000000 DIPPC 0014 000000 DIPPC 0014 000000 DIPPC 0014 000000 DIPPC
0114 000000 DIPMUC 0014 000000 DIPPC 0014 000000 DIPPC 0014 000000 DIPPC 0014 000000 DIPPC
0114 000000 DIPMUC 0014 000000 DIPPC 0014 000000 DIPPC 0014 000000 DIPPC 0014 000000 DIPPC
0114 000000 DIPMUC 0014 000000 DIPPC 0014 000000 DIPPC 0014 000000 DIPPC 0014 000000 DIPPC
0114 000000 DIPMUC 0014 000000 DIPPC 0014 000000 DIPPC 0014 000000 DIPPC 0014 000000 DIPPC
0114 000000 DIPMUC 0014 000000 DIPPC 0014 000000 DIPPC 0014 000000 DIPPC 0014 000000 DIPPC
0114 000000 DIPMUC 0014 000000 DIPPC 0014 000000 DIPPC 0014 000000 DIPPC 0014 000000 DIPPC
0114 000000 DIPMUC 0014 000000 DIPPC 0014 000000 DIPPC 0014 000000 DIPPC 0014 000000 DIPPC
0114 000000 DIPMUC 0014 000000 DIPPC 0014 000000 DIPPC 0014 000000 DIPPC 0014 000000 DIPPC
0114 000000 DIPMUC 0014 000000 DIPPC 0014 000000 DIPPC 0014 000000 DIPPC 0014 000000 DIPPC
0114 000000 DIPMUC 0014 000000 DIPPC 0014 000000 DIPPC 0014 000000 DIPPC 0014 000000 DIPPC

0101 1= SUBROUTINE OUTCAL
0103 2= COMMON TITLE
0104 3= COMMON STEPPS
COMMON / VARP / C
COMMON / SOURCE / G
COMMON / CONDO / G
COMMON / PCI / SEQ1
COMMON /PC2/ SEQ2
COMMON / KONST/ K
COMMON / JARARY / A
COMMON /FELCON/ TIMEN, OTIMEU, TIMEND, CSGFA,
COMMON /FELCON/ TIMEN, OTIMEU, TIMEND, DAMPA,
COMMON / FELCON/ TIMEN, OTIMEU, TIMEND, DAMPA,
COMMON / FELCON/ TIMEN, OTIMEU, TIMEND, DAMPA,
COMMON / FELCON/ TIMEN, OTIMEU, TIMEND, DAMPA,
COMMON / FELCON/ TIMEN, OTIMEU, TIMEND, DAMPA,
COMMON / FELCON/ TIMEN, OTIMEU, TIMEND, DAMPA,
COMMON / FELCON/ TIMEN, OTIMEU, TIMEND, DAMPA,
COMMON / FELCON/ TIMEN, OTIMEU, TIMEND, DAMPA,
COMMON / FELCON/ TIMEN, OTIMEU, TIMEND, DAMPA,
COMMON / FELCON/ TIMEN, OTIMEU, TIMEND, DAMPA,
COMMON / FELCON/ TIMEN, OTIMEU, TIMEND, DAMPA,
COMMON / FELCON/ TIMEN, OTIMEU, TIMEND, DAMPA,
COMMON / FELCON/ TIMEN, OTIMEU, TIMEND, DAMPA,
COMMON / FELCON/ TIMEN, OTIMEU, TIMEND, DAMPA,
COMMON / FELCON/ TIMEN, OTIMEU, TIMEND, DAMPA,
COMMON / FELCON/ TIMEN, OTIMEU, TIMEND, DAMPA,
COMMON / FELCON/ TIMEN, OTIMEU, TIMEND, DAMPA,
COMMON / FELCON/ TIMEN, OTIMEU, TIMEND, DAMPA,
COMMON / FELCON/ TIMEN, OTIMEU, TIMEND, DAMPA,
COMMON / FELCON/ TIMEN, OTIMEU, TIMEND, DAMPA,
COMMON / FELCON/ TIMEN, OTIMEU, TIMEND, DAMPA,
COMMON / FELCON/ TIMEN, OTIMEU, TIMEND, DAMPA,
COMMON / FELCON/ TIMEN, OTIMEU, TIMEND, DAMPA,
COMMON / FELCON/ TIMEN, OTIMEU, TIMEND, DAMPA,
COMMON / FELCON/ TIMEN, OTIMEU, TIMEND, DAMPA,
COMMON / FELCON/ TIMEN, OTIMEU, TIMEND, DAMPA,
COMMON / FELCON/ TIMEN, OTIMEU, TIMEND, DAMPA,
COMMON / FELCON/ TIMEN, OTIMEU, TIMEND, DAMPA,
COMMON / FELCON/ TIMEN, OTIMEU, TIMEND, DAMPA,
COMMON / FELCON/ TIMEN, OTIMEU, TIMEND, DAMPA,
COMMON / FELCON/ TIMEN, OTIMEU, TIMEND, DAMPA,
COMMON / FELCON/ TIMEN, OTIMEU, TIMEND, DAMPA,
COMMON / FELCON/ TIMEN, OTIMEU, TIMEND, DAMPA,
COMMON / FELCON/ TIMEN, OTIMEU, TIMEND, DAMPA,
COMMON / FELCON/ TIMEN, OTIMEU, TIMEND, DAMPA,
COMMON / FELCON/ TIMEN, OTIMEU, TIMEND, DAMPA,
COMMON / FELCON/ TIMEN, OTIMEU, TIMEND, DAMPA,
COMMON / FELCON/ TIMEN, OTIMEU, TIMEND, DAMPA,
COMMON / FELCON/ TIMEN, OTIMEU, TIMEND, DAMPA,
COMMON / FELCON/ TIMEN, OTIMEU, TIMEND, DAMPA,
COMMON / FELCON/ TIMEN, OTIMEU, TIMEND, DAMPA,
COMMON / FELCON/ TIMEN, OTIMEU, TIMEND, DAMPA,
COMMON / FELCON/ TIMEN, OTIMEU, TIMEND, DAMPA,
COMMON / FELCON/ TIMEN, OTIMEU, TIMEND, DAMPA,
COMMON / FELCON/ TIMEN, OTIMEU, TIMEND, DAMPA,
COMMON / FELCON/ TIMEN, OTIMEU, TIMEND, DAMPA,
COMMON / FELCON/ TIMEN, OTIMEU, TIMEND, DAMPA,
COMMON / FELCON/ TIMEN, OTIMEU, TIMEND, DAMPA,
COMMON / FELCON/ TIMEN, OTIMEU, TIMEND, DAMPA,
COMMON / FELCON/ TIMEN, OTIMEU, TIMEND, DAMPA,
COMMON / FELCON/ TIMEN, OTIMEU, TIMEND, DAMPA,
COMMON / FELCON/ TIMEN, OTIMEU, TIMEND, DAMPA,
COMMON / FELCON/ TIMEN, OTIMEU, TIMEND, DAMPA,
COMMON / FELCON/ TIMEN, OTIMEU, TIMEND, DAMPA,
COMMON / FELCON/ TIMEN, OTIMEU, TIMEND, DAMPA,
COMMON / FELCON/ TIMEN, OTIMEU, TIMEND, DAMPA,
COMMON / FELCON/ TIMEN, OTIMEU, TIMEND, DAMPA,
COMMON / FELCON/ TIMEN, OTIMEU, TIMEND, DAMPA,
COMMON / FELCON/ TIMEN, OTIMEU, TIMEND, DAMPA,
COMMON / FELCON/ TIMEN, OTIMEU, TIMEND, DAMPA,
COMMON / FELCON/ TIMEN, OTIMEU, TIMEND, DAMPA,
COMMON / FELCON/ TIMEN, OTIMEU, TIMEND, DAMPA,
COMMON / FELCON/ TIMEN, OTIMEU, TIMEND, DAMPA,
COMMON / FELCON/ TIMEN, OTIMEU, TIMEND, DAMPA,
COMMON / FELCON/ TIMEN, OTIMEU, TIMEND, DAMPA,
COMMON / FELCON/ TIMEN, OTIMEU, TIMEND, DAMPA,
COMMON / FELCON/ TIMEN, OTIMEU, TIMEND, DAMPA,
COMMON / FELCON/ TIMEN, OTIMEU, TIMEND, DAMPA,
COMMON / FELCON/ TIMEN, OTIMEU, TIMEND, DAMPA,
COMMON / FELCON/ TIMEN, OTIMEU, TIMEND, DAMPA,
COMMON / FELCON/ TIMEN, OTIMEU, TIMEND, DAMPA,
COMMON / FELCON/ TIMEN, OTIMEU, TIMEND, DAMPA,
COMMON / FELCON/ TIMEN, OTIMEU, TIMEND, DAMPA,
COMMON / FELCON/ TIMEN, OTIMEU, TIMEND, DAMPA,
COMMON / FELCON/ TIMEN, OTIMEU, TIMEND, DAMPA,
COMMON / FELCON/ TIMEN, OTIMEU, TIMEND, DAMPA,
COMMON / FELCON/ TIMEN, OTIMEU, TIMEND, DAMPA,
COMMON / FELCON/ TIMEN, OTIMEU, TIMEND, DAMPA,
COMMON / FELCON/ TIMEN, OTIMEU, TIMEND, DAMPA,
COMMON / FELCON/ TIMEN, OTIMEU, TIMEND, DAMPA,
COMMON / FELCON/ TIMEN, OTIMEU, TIMEND, DAMPA,
COMMON / FELCON/ TIMEN, OTIMEU, TIMEND, DAMPA,
COMMON / FELCON/ TIMEN, OTIMEU, TIMEND, DAMPA,
COMMON / FELCON/ TIMEN, OTIMEU, TIMEND, DAMPA,
COMMON / FELCON/ TIMEN, OTIMEU, TIMEND, DAMPA,
COMMON / FELCON/ TIMEN, OTIMEU, TIMEND, DAMPA,
COMMON / FELCON/ TIMEN, OTIMEU, TIMEND, DAMPA,
COMMON / FELCON/ TIMEN, OTIMEU, TIMEND, DAMPA,
COMMON / FELCON/ TIMEN, OTIMEU, TIMEND, DAMPA,
COMMON / FELCON/ TIMEN, OTIMEU, TIMEND, DAMPA,
COMMON / FELCON/ TIMEN, OTIMEU, TIMEND, DAMPA,
COMMON / FELCON/ TIMEN, OTIMEU, TIMEND, DAMPA,
COMMON / FELCON/ TIMEN, OTIMEU, TIMEND, DAMPA,
COMMON / FELCON/ TIMEN, OTIMEU, TIMEND, DAMPA,
COMMON / FELCON/ TIMEN, OTIMEU, TIMEND, DAMPA,
COMMON / FELCON/ TIMEN, OTIMEU, TIMEND, DAMPA,
COMMON / FELCON/ TIMEN, OTIMEU, TIMEND, DAMPA,
COMMON / FELCON/ TIMEN, OTIMEU, TIMEND, DAMPA,
COMMON / FELCON/ TIMEN, OTIMEU, TIMEND, DAMPA,
COMMON / FELCON/ TIMEN, OTIMEU, TIMEND, DAMPA,
COMMON / FELCON/ TIMEN, OTIMEU, TIMEND, DAMPA,
COMMON / FELCON/ TIMEN, OTIMEU, TIMEND, DAMPA,
COMMON / FELCON/ TIMEN, OTIMEU, TIMEND, DAMPA,
COMMON / FELCON/ TIMEN, OTIMEU, TIMEND, DAMPA,
COMMON / FELCON/ TIMEN, OTIMEU, TIMEND, DAMPA,
COMMON / FELCON/ TIMEN, OTIMEU, TIMEND, DAMPA,
COMMON / FELCON/ TIMEN, OTIMEU, TIMEND, DAMPA,
COMMON / FELCON/ TIMEN, OTIMEU, TIMEND, DAMPA,
STARTING ADDRESS 015000
CORE LIMITS 015000 044426 100000 123491 163772 163777

SINDA /Code
0 100000-100000
1 014000-014513

NSTOP/RECS
1 014014-014625

NTRC/RECS
G 100001-100001
H 014026-014333
2 100002-100074

AFATS/RECS
1 015334-015271
2 100577-105113

OPTC/RECS
1 015272-015334

RENST/RECS
1 015315-015315
2 100114-100202

NIHRA/RECS
1 016652-016411
2 100203-100246

RSPACE/RECS
1 016412-016265

DEPT/******
G 100247-100268

STEPS/RECS
G 100255-100444
1 016256-016720

MIZED/RECS
1 016721-016772
2 165445-165765

EXEC/RECS
G 155476-165772
1 016773-017011
| DATA /****** |
| 0 106490-106667 |
| INPUT/CODE |
| 0 106700-106731 |
| 1 093509-094926 |
| SPACET/****** |
| 0 106732-112653 |
| LOGIC /****** |
| 0 112459-112457 |
| DIMENS/****** |
| 0 112460-112670 |
| FINCON/****** |
| 0 112471-112752 |
| ARRAY /****** |
| 0 112753-117671 |
| KONST /****** |
| 0 117672-120579 |
| PC2 /****** |
| 0 120575-120546 |
| PCI /****** |
| 0 120547-121423 |
| COND /****** |
| 0 121424-122113 |
| SOURCE/****** |
| 0 122114-122465 |
| CAP /****** |
| 0 122466-123037 |
| TEPP /****** |
| 0 123040-123915 |
| TITLE /****** |
| 0 123916-123941 |

END OF ALLOCATION 1103 0339A 09099
### Sample Problem for SINDA Version 9

<table>
<thead>
<tr>
<th>TIME</th>
<th>0.000000</th>
<th>DTIME</th>
<th>0.000000</th>
<th>EESMINI</th>
<th>01 = 0.000000</th>
<th>TEMPCELL</th>
<th>01 = 0.000000</th>
<th>RELCELL</th>
<th>01 = 0.000000</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>1 = 70.000</td>
<td>T</td>
<td>2 = 70.000</td>
<td>T</td>
<td>3 = 70.000</td>
<td>T</td>
<td>4 = 70.000</td>
<td>T</td>
<td>5 = 70.000</td>
</tr>
<tr>
<td></td>
<td>1 = 70.000</td>
<td>T</td>
<td>2 = 70.000</td>
<td>T</td>
<td>3 = 70.000</td>
<td>T</td>
<td>4 = 70.000</td>
<td>T</td>
<td>5 = 70.000</td>
</tr>
<tr>
<td></td>
<td>1 = 70.000</td>
<td>T</td>
<td>2 = 70.000</td>
<td>T</td>
<td>3 = 70.000</td>
<td>T</td>
<td>4 = 70.000</td>
<td>T</td>
<td>5 = 70.000</td>
</tr>
<tr>
<td></td>
<td>1 = 70.000</td>
<td>T</td>
<td>2 = 70.000</td>
<td>T</td>
<td>3 = 70.000</td>
<td>T</td>
<td>4 = 70.000</td>
<td>T</td>
<td>5 = 70.000</td>
</tr>
<tr>
<td></td>
<td>1 = 70.000</td>
<td>T</td>
<td>2 = 70.000</td>
<td>T</td>
<td>3 = 70.000</td>
<td>T</td>
<td>4 = 70.000</td>
<td>T</td>
<td>5 = 70.000</td>
</tr>
<tr>
<td></td>
<td>1 = 70.000</td>
<td>T</td>
<td>2 = 70.000</td>
<td>T</td>
<td>3 = 70.000</td>
<td>T</td>
<td>4 = 70.000</td>
<td>T</td>
<td>5 = 70.000</td>
</tr>
</tbody>
</table>

**Flow Rates (L/min)**

- 248.0, 1
- 248.5
- 248.6
- 248.7
- 248.8
- 248.9
- 249.0
- 249.1
- 249.2
- 249.3
- 249.4
- 249.5
- 249.6
- 249.7
- 249.8
- 249.9
- 250.0

**Pressure Drops (KPa)**

- 507.0
- 507.1
- 507.2
- 507.3
- 507.4
- 507.5
- 507.6
- 507.7
- 507.8
- 507.9
- 508.0
- 508.1
- 508.2
- 508.3
- 508.4
- 508.5
- 508.6
- 508.7
- 508.8
- 508.9
- 509.0
### Table: Pressures and Time Values

<table>
<thead>
<tr>
<th>Positions</th>
<th>Time (min)</th>
<th>Pressure (Lb/ft²)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.00000</td>
<td>69.777</td>
<td>2226.5</td>
</tr>
<tr>
<td>2.00000</td>
<td>62.994</td>
<td>2226.5</td>
</tr>
<tr>
<td>3.00000</td>
<td>69.107</td>
<td>2226.5</td>
</tr>
<tr>
<td>4.00000</td>
<td>69.107</td>
<td>2226.5</td>
</tr>
<tr>
<td>5.00000</td>
<td>69.107</td>
<td>2226.5</td>
</tr>
<tr>
<td>6.00000</td>
<td>69.107</td>
<td>2226.5</td>
</tr>
<tr>
<td>7.00000</td>
<td>69.107</td>
<td>2226.5</td>
</tr>
<tr>
<td>8.00000</td>
<td>69.107</td>
<td>2226.5</td>
</tr>
<tr>
<td>9.00000</td>
<td>69.107</td>
<td>2226.5</td>
</tr>
<tr>
<td>10.00000</td>
<td>69.107</td>
<td>2226.5</td>
</tr>
</tbody>
</table>

### DIVIDE Check
- DIVIDE CHECK AT 021506
- DIVIDE CHECK AT 021506
- DIVIDE CHECK AT 021506
- DIVIDE CHECK AT 021506
- DIVIDE CHECK AT 021506
- DIVIDE CHECK AT 021506
- DIVIDE CHECK AT 021506
### PRESSURE CROPS

<table>
<thead>
<tr>
<th>PRESSURE</th>
<th>VALVE POSITIONS</th>
<th>TIME:</th>
<th>COMPUTER TIME = 1.952 MINUTES</th>
</tr>
</thead>
<tbody>
<tr>
<td>219 = 69.618</td>
<td>T 292 = 64.129</td>
<td>T 293 = 65.785</td>
<td>T 294 = 65.497</td>
</tr>
<tr>
<td>297 = 71.029</td>
<td>T 298 = 71.029</td>
<td>T 299 = 71.031</td>
<td>T 300 = 71.033</td>
</tr>
<tr>
<td>309 = 69.618</td>
<td>T 310 = 69.618</td>
<td>T 311 = 69.618</td>
<td>T 312 = 69.619</td>
</tr>
<tr>
<td>315 = 40.059</td>
<td>T 316 = 71.032</td>
<td>T 317 = 69.618</td>
<td>T 318 = 40.108</td>
</tr>
</tbody>
</table>

**FLOW RATES (LB/HR)**

<table>
<thead>
<tr>
<th>TIME [min]</th>
<th>COMPUTER TIME = 1.952 MINUTES</th>
</tr>
</thead>
<tbody>
<tr>
<td>219.757</td>
<td>2462.2</td>
</tr>
<tr>
<td>1587.3</td>
<td>793.81</td>
</tr>
<tr>
<td>219.949</td>
<td>217.30</td>
</tr>
<tr>
<td>397.56</td>
<td>396.25</td>
</tr>
</tbody>
</table>

**PRESSURE DROPS (LB/Ft**x2**)**

<table>
<thead>
<tr>
<th>PRESSURE</th>
<th>COMPUTER TIME = 1.952 MINUTES</th>
</tr>
</thead>
<tbody>
<tr>
<td>T 204.28</td>
<td>61946.</td>
</tr>
<tr>
<td>50619.</td>
<td>39.349</td>
</tr>
<tr>
<td>370.63</td>
<td>370.63</td>
</tr>
<tr>
<td>3622.0</td>
<td>3622.0</td>
</tr>
</tbody>
</table>

**PRESSURES (LB/FT**x2**)**

<table>
<thead>
<tr>
<th>PRESSURE</th>
<th>COMPUTER TIME = 1.952 MINUTES</th>
</tr>
</thead>
<tbody>
<tr>
<td>T 315</td>
<td>316</td>
</tr>
</tbody>
</table>

**VALVE POSITIONS**

<table>
<thead>
<tr>
<th>COMPUTER TIME = 1.952 MINUTES</th>
</tr>
</thead>
<tbody>
<tr>
<td>249990</td>
</tr>
</tbody>
</table>
### Flow Rates (lb/hr)

<table>
<thead>
<tr>
<th>Sample</th>
<th>Flow Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>205.7</td>
<td>2452.2</td>
</tr>
<tr>
<td>207.5</td>
<td>2452.2</td>
</tr>
<tr>
<td>209.0</td>
<td>2452.2</td>
</tr>
<tr>
<td>210.0</td>
<td>2452.2</td>
</tr>
</tbody>
</table>

### Pressure Drops (lb/ft**2**)

<table>
<thead>
<tr>
<th>Sample</th>
<th>Pressure Drop</th>
</tr>
</thead>
<tbody>
<tr>
<td>209.2</td>
<td>6584.2</td>
</tr>
<tr>
<td>210.1</td>
<td>3720.0</td>
</tr>
</tbody>
</table>

### Valve Positions

<table>
<thead>
<tr>
<th>Valve Position</th>
</tr>
</thead>
<tbody>
<tr>
<td>.9900</td>
</tr>
<tr>
<td>.9922</td>
</tr>
</tbody>
</table>

### Computer Time = 2.922 MINUTES

### Time (min): 3.240555 5.054983 CSFFP14

### EXIT:

<table>
<thead>
<tr>
<th>Time</th>
<th>T2</th>
<th>T3</th>
<th>T4</th>
<th>T5</th>
<th>T6</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>70.978</td>
<td>25.973</td>
<td>59.979</td>
<td>21.97</td>
<td>30.976</td>
</tr>
<tr>
<td>2</td>
<td>70.978</td>
<td>25.973</td>
<td>59.979</td>
<td>21.97</td>
<td>30.976</td>
</tr>
<tr>
<td>3</td>
<td>25.973</td>
<td>59.979</td>
<td>21.97</td>
<td>30.976</td>
<td>21.97</td>
</tr>
<tr>
<td>4</td>
<td>25.973</td>
<td>59.979</td>
<td>21.97</td>
<td>30.976</td>
<td>21.97</td>
</tr>
<tr>
<td>5</td>
<td>25.973</td>
<td>59.979</td>
<td>21.97</td>
<td>30.976</td>
<td>21.97</td>
</tr>
<tr>
<td>6</td>
<td>25.973</td>
<td>59.979</td>
<td>21.97</td>
<td>30.976</td>
<td>21.97</td>
</tr>
</tbody>
</table>
### SAMPLE PROBLEM FOR SINDA VERSION 9

| Q | 54.8 | 50.0 | 56.0 | 62.0 | 68.0 | 74.0 | 82.0 | 88.0 | 92.0 | 98.0 | 104.0 | 110.0 | 116.0 | 202.0 | 208.0 | 214.0 | 220.0 | 226.0 | 232.0 | 238.0 | 244.0 | 250.0 | 256.0 | 262.0 | 268.0 | 274.0 | 280.0 | 286.0 | 292.0 | 298.0 | 304.0 | 310.0 | 316.0 |

### FLOW RATES (L/EHR)

<table>
<thead>
<tr>
<th>T</th>
<th>2465.7</th>
<th>2462.2</th>
<th>874.70</th>
<th>37.40</th>
<th>217.19</th>
<th>217.52</th>
<th>217.96</th>
<th>107.60</th>
</tr>
</thead>
</table>

### PRESSURE DROPS (Lb/sq. ft x 2)

<table>
<thead>
<tr>
<th>T</th>
<th>324.28</th>
<th>65940.8</th>
<th>65468.0</th>
<th>13.867</th>
<th>379.23</th>
<th>379.23</th>
<th>379.23</th>
<th>379.23</th>
</tr>
</thead>
</table>

### PRESSURES (Lb/sq. ft x 2)

<table>
<thead>
<tr>
<th>T</th>
<th>12706.06</th>
<th>12706.06</th>
<th>62668.0</th>
<th>3250.3</th>
<th>2621.2</th>
<th>3164.7</th>
<th>2967.1</th>
<th>2965.2</th>
</tr>
</thead>
</table>

### VALVE POSITIONS

<table>
<thead>
<tr>
<th>T</th>
<th>90950</th>
<th>34220</th>
<th>34220</th>
<th>34220</th>
<th>34220</th>
<th>34220</th>
<th>34220</th>
<th>34220</th>
</tr>
</thead>
</table>
SAMPLE PROBLEM FOR SINDA VERSION 9

COMPUTER TIME = 3.886 MINUTES

END OF DATA
FIGURE 12
RADIATOR TEMPERATURES FOR SAMPLE PROBLEM

SINDA VERSION 9 SAMPLE PROBLEM
[1] RADIATOR INLET TEMPERATURE -- DEGF
[2] MAIN RADIATOR OUTLET TEMPERATURE -- DEGF
[3] PRIME TUBE OUTLET TEMPERATURE -- DEGF
[4] MIXED RADIATOR OUTLET TEMPERATURE -- DEGF

TEMPERATURE

TIME - (HOURS)
FIGURE 13
SYSTEM TEMPERATURES FOR SAMPLE PROBLEM

SINDA VERSION 9 SAMPLE PROBLEM

[1] RADIATOR CONTROLLED OUTLET, HX INLET -- DEGF
[2] HX OUTLET ON RADIATOR SIDE -- DEGF
[3] HX INLET ON WATER SIDE -- DEGF
[4] HX OUTLET ON WATER SIDE -- DEGF

TIME - (HOURS)
TEMPERATURE

80
70
60
50
40
3
2
1
0
1
2
3
4
5
6
7
8
9
10
FIGURE 14
SYSTEM PRESSURES FOR SAMPLE PROBLEM

SINDA VERSION 9 SAMPLE PROBLEM
[1] TOTAL PUMP FLOW RATE -- LB/HR
[2] TOTAL RADIATOR FLOW RATE -- LB/HR
[3] BYPASS FLOW RATE -- LB/HR

TIME - [HOURS]
FIGURE 15

RADIATOR FLOW RATES FOR SAMPLE PROBLEM

SINDA VERSION 9 SAMPLE PROBLEM

1. TOTAL RADIATOR FLOW RATE -- LB/HR
2. MAIN RADIATOR FLOW RATE -- LB/HR
3. PRIME TUBE FLOW RATE -- LB/HR

FLOW RATE

TIME - [HOURS]
FIGURE 16

PUMP PRESSURES FOR SAMPLE PROBLEM

SINDA VERSION 9 SAMPLE PROBLEM

[1] PUMP OUTLET PRESSURE -- PSF
[2] VALVE 1 INLET PRESSURE -- PSF
[3] VALVE 2 INLET PRESSURE -- PSF
[4] PUMP INLET PRESSURE -- PSF

TIME - (HOURS)
FIGURE 17
SYSTEM PRESSURES FOR SAMPLE PROBLEM

SINDA VERSION 9 SAMPLE PROBLEM

[1] MAIN RADIATOR INLET PRESSURE -- PSF
[2] PRIME TUBE INLET PRESSURE -- PSF
[3] PRESSURE AT RADIATOR OUTLET -- PSF
[4] PRESSURE AT HX INLET -- PSF

<table>
<thead>
<tr>
<th>Time (Hours)</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pressure</td>
<td>10000</td>
<td>8000</td>
<td>6000</td>
<td>4000</td>
</tr>
<tr>
<td></td>
<td>2000</td>
<td>4000</td>
<td>6000</td>
<td>8000</td>
</tr>
<tr>
<td></td>
<td>10000</td>
<td>8000</td>
<td>6000</td>
<td>4000</td>
</tr>
<tr>
<td></td>
<td>2000</td>
<td>4000</td>
<td>6000</td>
<td>8000</td>
</tr>
</tbody>
</table>

153
6.0 REFERENCES


APPENDIX A

SUBROUTINE LISTINGS

A Fortran listing is presented below for the subroutines which were modified or added to the SINDA preprocessor and user subroutine library which create SINDA/VERSION 9.
1116.
24-
2%1.
246.
295. LCOND.TREE.
24q.
293.
281.
240.
232.
230.
228.
227.
226.
222.
219. CALL SOUEEZ
218.
216.
214.
213.
212. IF
211.
210.
209. 210
208.
207.
206.
205.
204.
203.
202.
201.
200.
209.
208.
207.
206.
205.
204.
203.
202.
201.
200.
209.
208.
207.
206.
205.
204.
203.
202.
201.
200.
209.
208.
207.
206.
205.
204.
203.
202.
201.
200.
209.
208.
207.
206.
205.
204.
203.
202.
201.
200.
209.
208.
207.
206.
205.
204.
203.
202.
201.
200.
209.
208.
207.
206.
205.
204.
203.
202.
201.
200.
209.
208.
207.
206.
205.
204.
203.
202.
201.
200.
209.
208.
207.
206.
205.
204.
203.
202.
201.
200.
209.
208.
207.
206.
205.
204.
203.
202.
201.
200.
209.
208.
207.
206.
205.
204.
203.
202.
201.
200.
209.
208.
207.
206.
205.
204.
203.
202.
201.
200.
209.
208.
207.
206.
205.
204.
203.
202.
201.
200.
209.
208.
207.
206.
205.
204.
203.
202.
201.
200.
209.
208.
207.
206.
205.
204.
203.
202.
201.
200.
NINC=NEWS
LOC6=LOC5+LEN6
LEN6=0
I=250+I7,10
LOC1=LOC5+1+NINC
LEN1=0
250 CONTINUE
251 CALL DATAAR
252 JIST=LOC6
253 JJEND=LOC6+LEN6-1
254 WRITE (27) (B(JJ),JJ=JIST,JJEND)
255 READ (27) (B(JJ),JJ=JIST,JJEND)
256 WRITE (27) (B(JJ),JJ=JIST,JJEND)
257 IF (.NOT.LPRINT) GO TO 260
260 WRITE (NOUT,730) NGL,NGR,NGT
261 READ (21) (B(JJ),JJ=JJIST,JJEND)
262 WRITE (NOUT,700) (I,LOC(I),LEN(I),I=6,10)
263 M=LOC10*LEN10-1
264 WRITE (NOUT,10) (I,LOC(I),LEN(I),I=6,10)
265 WRITE (NOUT,710) (I,IBRCH3,I=1,6)
266 MRITE INOUT,730) NGL,NGR,NGT
267 MRITE INOUT,700) (I,LOC(I),LEN(I),I=6,10)
268 WRITE (NOUT,720) (I,LOC(I),LEN(I),I=6,10)
269 WRITE (NOUT,900) (I,LOC(I),LEN(I),I=6,10)
270 CONTINUE
271 CALL SREADC(I)
272 IF (I.EQ.COMMNT) GO TO 270
273 IF (I.EQ.REMARK) GO TO 270
274 IF (I.EQ.CONSTB) GO TO 270
275 IF (I.EQ.PRINT) .TRUE.
276 CONTINUE
277 CALL SREADC(I)
278 IF (I.EQ.COMMNT) GO TO 270
279 IF (I.EQ.REMARK) GO TO 270
280 IF (I.EQ.PRINT) .TRUE.
281 CALL SREADC(I)
282 IF (I.EQ.COMMNT) GO TO 270
283 IF (I.EQ.REMARK) GO TO 270
284 IF (I.EQ.PRINT) .TRUE.
285 CONTINUE
286 CALL SREADC(I)
287 IF (I.EQ.COMMNT) GO TO 270
288 IF (I.EQ.REMARK) GO TO 270
289 IF (I.EQ.PRINT) .TRUE.
290 CONTINUE
291 CALL SREADC(I)
292 IF (I.EQ.COMMNT) GO TO 270
293 IF (I.EQ.REMARK) GO TO 270
294 IF (I.EQ.PRINT) .TRUE.
295 CONTINUE
296 CALL SREADC(I)
297 IF (I.EQ.COMMNT) GO TO 270
298 IF (I.EQ.REMARK) GO TO 270
299 IF (I.EQ.PRINT) .TRUE.
300 CONTINUE
301 CALL SREADC(I)
302 IF (I.EQ.COMMNT) GO TO 270
303 IF (I.EQ.REMARK) GO TO 270
304 IF (I.EQ.PRINT) .TRUE.
305 CONTINUE
306 CALL SREADC(I)
307 IF (I.EQ.COMMNT) GO TO 270
308 IF (I.EQ.REMARK) GO TO 270
309 IF (I.EQ.PRINT) .TRUE.
310 CONTINUE
311 CALL SREADC(I)
2
RITE
(NOUT,670)
BLANK,COL2,.ALPHCOL ir
I
COR
2971 ,
COR
2972  CODE
2973  ,
COR
2974  IF (COLi.EO.COMMNT).G
2975  TO 290
2976  C D
2977  COR 298 ,,,
COR
2978  E
COR
2979  00ERO
314.
IF (ALPH(i).EO.REMARK)
2980  63
2981  TO
2982  280
2983  C D
2984  COR 299 , CODOERD
315.
IF (ALPH(3).NL.ARRVB)
2985  GO
2986  510
2987  C D
2988  COR 300 .
COR
2989  DERD
316.
IF
2990  ALPHIN9).EO.PRINT)
2991  AYPRNTz.TRUE.
2992  C D
2993  COR 301 CODEA
311.
2994  CONTINUE
2995  C D
2996  COR 302
COR
2997  C0DERO
318.
CALL
2998  SPREAD(I)
3000
C D
2999  WRITE (NOUT,670) BLANK,COL21,ALPHCOLI
3000
COR
3001 CODER?
320.
IF
3002  (COLi.E.COe
3003  G3 TO 290
3004  C D
3005  CDR
306.
IF
3006  (ALPH(i).EO.END)
3007  GO
3008  TO 300
3009  C D
3010
3011 C O0ER0
323. 311
3012 C 318.
3013 C00ERD
328.
C 311
3014 C 312.
3015 C00ER0
331.
C 317.
3016 C00ED
338. CALL
313
3017 C00ED
339.
C 318.
3018 C00ED
339.
C 319.
3019 C00ED
339.
C 319.
3020 C00ED
339.
C 330 CONTINUE
3021 C D
331.
3022 C00ED
332.
3023 C00ED
333.
3024 C00ED
334.
3025 C00ED
335.
3026 C00ED
336.
3027 C00ED
337.
3028 C00ED
338.
3029 C00ED
339.
3030 C00ED
340.
3031 C00ED
341.
C 342.
3032 C00ED
343.
C 343.
3033 C00ED
344.
C 344.
3034 C00ED
345.
C 345.
3035 C00ED
346.
C 346.
3036 C00ED
347.
C 347.
3037 C00ED
348.
C 348.
3038 C00ED
349.
C 349.
3039 C00ED
350.
C 350.
3040 C00ED
351.
C 351.
3041 C00ED
352.
C 352.
3042 C00ED
353.
C 353.
3043 C00ED
354.
C 354.
3044 C00ED
355.
C 355.
3045 C00ED
356.
C 356.
3046 C00ED
357.
C 357.
3047 C00ED
358.
C 358.
3048 C00ED

350  IF (M.EQ.9) IANUM=IANUM+1
360  FLD(I,1,IBADDR) = IADDR
365  LITK:FLOD,IBADDR = IADDR
370  IF (I.EQ.0) GO TO 370
375  CALL RELACT (2,KNUM,J,M)
380  GO TO 390
390  IADDR=O
395  FLD(Z3,3,IADDR) = XNUM
400  KNUM=KNUM+NCC
405  IF (LITK.EQ.0) GO TO 405
410  K=LOC(2)
415  IF (LITK.EQ.0) GO TO 415
420  IF (ITYPE.EQ.O) OR (ITYPE.EQ.3) GO TO 420
425  FLG(5,1,IBADDR) = IADDR
430  FLG(21,1,IBADDR) = IADDR
435  IF (IBADDR.EQ.0) GO TO 435
440  CALL RELACT (3,KNUM,J,M)
445  GO TO 450
450  CONTINUE
455  IF (M.EQ.9) IANUM=IANUM+1
460  NCC=NCC+NEC
465  K=K+l
470  M=LOC(2)
475  IF (M.EQ.9) GO TO 475
480  IF (K.LT.KEND) GO TO 480
485  IADDR=O
490  TYPE=FLOD,IBADDR
495  IF (K.NE.1) GO TO 495
500  CONTINUE
505  IF (M.EQ.9) IANUM=IANUM+1
510  FLD(I,1,IBADDR) = IADDR
515  LITK:FLOD,IBADDR = IADDR
520  IF (I.EQ.0) GO TO 520
525  CALL RELACT (2,KNUM,J,M)
530  GO TO 530
535  IANUM=IANUM+1
540  FLD(I,1,IBADDR) = IADDR
545  LITK:FLOD,IBADDR = IADDR
550  IF (I.EQ.0) GO TO 550
555  CALL RELACT (2,KNUM,J,M)
560  GO TO 560
565  IANUM=IANUM+1
570  FLD(I,1,IBADDR) = IADDR
575  LITK:FLOD,IBADDR = IADDR
580  IF (I.EQ.0) GO TO 580
585  CALL RELACT (2,KNUM,J,M)
590  CONTINUE
140. LCONST=:FALSE.
141. LARRAY=FALSE.
142. RETURN

410. C ERROR RETURN

420. 900 WRITE (NOUT,440)
421. ENDAT=2.0
422. GO TO 520

423. 910 WRITE (NOUT,280)
424. ENDAT=2.0
425. 520 CONTINUE
426. ENDUN=1.0
427. RETURN

428. C PARAMETER RUNS

429. 530 CONTINUE
430. LEND=.FALSE.
431. NOREAD=.TRUE.
432. ISPI
433. IF (GENR1) EST=3
434. GO A16 INSRA.
435. IF (LEND=.FALSE.) GO TO 380
436. 940 CALL SREADC(I)
437. WRITE (NOUT,670) BLANK,COL2,ALPH,COL1
438. IF (COL1.EQ.COMMT) GO TO 540
439. IF (ALPH1).EQ.REMARKS GO TO 540
440. IF (ALPH1),NE.ENDFAM1 GO TO 550
441. LEND=.TRUE.
442. GO TO 560
443. 950 BCHE=.ALPH(3)
444. GO 360 J=1..4
445. IF (BCH=.EQ.BLANK(I)) GO TO 570

446. 560 CONTINUE
447. GO TO 310
448. 570 CALL SREADC(I)
449. WRITE (NOUT,670) BLANK,COL2,ALPH,COL1
450. IF (COL1,EQ.COMMT) GO TO 540
451. IF (ALPH1. EQ.REMARKS ) GO TO 540
452. 580 CALL INCHE (I)
453. IF (ALPH1).EQ.COMMT) GO TO 600
454. NOREAD=.TRUE.
455. IF (BCH. Eمؤ.REMARKS(1)) GO TO 600
456. NOREAD=.FALSE.
457. BRANCH=1
458. GO 590 J=1..4
459. LLOGIC=.FALSE.
460. 590 CONTINUE
461. LLOGIC=.TRUE.
462. CALL EATARR
463. 600 CALL UNREAD (1)
464. 610 CONTINUE
465. GO TO 490

466. C

467. 620 FORMAT (I17/)
468. 620 FORMAT (14,AN,41,315,AN,42)
469. 620 FORMAT (AN,AN,41,315,AN,42)
470. 620 FORMAT (41,135,AN,42)
471. 620 FORMAT (14,AN,41,315,AN,42)
472. 620 FORMAT (14,AN,41,315,AN,42)
473. 670 FORMAT (41,AN,41,315,AN,42)
474. 670 FORMAT (41,135,AN,42)
475. 620 FORMAT (41,135,AN,42)
476. 620 FORMAT (41,135,AN,42)
I. BLOCK DESIGNATION ENCOUNTERED ...

492. E99 FORMAT (AN * = 4,990 THE PSEUDO COMPARE SEQUENCE INDICATOR MUST BE... COERD

493. 1 EITHER SPS OR LPCS, AND START IN COLUMN 21 COERD

494. F00 FORMAT (1NH ARRAYS LOC AND LEN,1*10DI) COERD

495. T10 FORMAT (12H DATA BUCKET,(130,120,220,5,95,220)) COERD

496. T20 FORMAT (AN AND,16,AN AND,16,AN AND,16,AN AND,16) COERD

497. T30 FORMAT (AN NOL,16,AN NOL,16,AN NOL,16) COERD

498. T40 FORMAT (AN NEC,16,AN NEC,16,AN NEC,16) COERD

499. T50 FORMAT (12NH FIXED CONSTANTS ARRAY,(113,120,220,5,85,323)) COERD

500. T60 FORMAT (15H LENX,16) COERD

501. END COERD
SUBROUTINE IMBED

COMMON /BUCKET/ (BU(11))
COMMON /DATA/ (BU(113), NGT, NUC, DUM(4), ENDA

DIMENSION KEYA(M), KEY(NUC+1, 100), KEY(100, 100)

LOGICAL CRDERR

LI = LOC(13)
LT = LEM(13) + LI - 1
MM = LEM(14)
M2 = LOC(15) - 1
DI 950 = LI, LT
MI = M2 + 1
MM = MM + 1

KEY = 4P
DI 400 = M1, M2
FLO(12, KEY) = FLO(12, IB(11))
DI 10 K=1, A
IF (KEY.EQ. KEYA(1)) GO TO 40

10 CONTINUE

GO TO 400

40 NUM = IB(11)
CALL CONVRT(12, 30, NUM, CRDERR)
EF (CRDERR) GO TO 380

C
C 45 GO TO (100, 200, 300, 350), K
C
C ARRAYS

100 L = 1
111 LL = LOC(19)
199 ISF = LOC(133)
IEND = ISF + LEM(133) - 1
DO 140 J = ISF, IEND - 1
IF (NUM .EQ. IB(13)) GO TO 390
140 L = L * 1000
199 LL = LL + 1

140 CONTINUE

GO TO 380

C
C CONSTANTS

200 NLOC = LOC(11)
AHEM = NUC
GO TO 360

C
C TEMPERATURES

300 NLOC = LOC(11)
AHEM = LEM(13)
GO TO 360

C
C CONDUCTORS

350 NLOC = LOC(11)
AHEM = NUC

C
360 CALL SEARCHNUM, (BINLOC), NLEN, L
365 IF (L) 390, 390, 390
366 ERDATA = L
367 NN = L - M1 + 1
370 WRITE (4, 380) ERDATA, NN, IB
380 FOR (I = 1, 14, M2) IF (I) 390, 390, 390
390 IF (I) = L THEN 400, 400
400 CONTINUE
410 C 900 CONTINUE
420 C 900 CONTINUE
430 RETURN
440 C END
<table>
<thead>
<tr>
<th>#</th>
<th>SEG</th>
<th>Function</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>MLINK</td>
<td>PREPARE-BLCKREAD-WRITABLE-STATE-FILE-NAME-SQUEEZE-STREAM</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>D</td>
<td>MLINK-(EDIT, MUR, GENLINK, A, B, E, C, F)</td>
<td>SINDA</td>
</tr>
<tr>
<td>3</td>
<td>M</td>
<td>PSEUDO-PESO</td>
<td>SINDA</td>
</tr>
<tr>
<td>4</td>
<td>B</td>
<td>CODING-DATAREAD-ERRORS-COUNT-TYPEX-ODATA</td>
<td>SINDA</td>
</tr>
<tr>
<td>5</td>
<td>B</td>
<td>RELATE-DATAWRITE-INCORE-SETFMT-GENUX-(NODES, CONDS)</td>
<td>SINDA</td>
</tr>
<tr>
<td>6</td>
<td>C</td>
<td>PRESUB-SINDA-PRIVATE-ALPENT</td>
<td>SINDA</td>
</tr>
<tr>
<td>7</td>
<td>F</td>
<td>SPLIT-SKIP</td>
<td>SINDA</td>
</tr>
<tr>
<td>8</td>
<td>BLK</td>
<td>BUFFER-MODNEW-MURCARD</td>
<td>SINDA</td>
</tr>
<tr>
<td>9</td>
<td>MUR</td>
<td>MODUR-BUFFER-MODNEW-MURCARD</td>
<td>SINDA</td>
</tr>
</tbody>
</table>
SUBROUTINE CABIN(NLOC, TC, SUMOL, SUMC)  
C    LOGICAL EXPLECT
C    COMMON /ARRAY/ DATA(I)
C    COMMON /TINC/ TINC(I)
C    COMMON /HIC/ HIC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
C    COMMON /HINC/ HINC(I)
AS = NDATA(NSTAT+1)
AHI = FABS(NDATA(NSTAT)) - 1
IF (NDATA(NSTAT).LT. 0) GO TO 2
IF(SN.W.E. AHI) CALL ERR(3HIST),
NDATA(NSTAT) = NDATA(NSTAT)
2 IF (NDATA(NCRV).LT. 0) GO TO 4
IF(NDATA(NCRV).NE. 0) CALL ERR(3HIST)
NDATA(NCRV) = NDATA(NCRV)
4 IF(NDATA(NCON).NE. 1) CALL ERR(3HIST)
NC = NDATA(NCON+1)
LAPA = NDATA(NCON+2)
PA = DATA(NCON+3)
PC = DATA(NCON+4)
NC = DATA(NCON+5)
MW = DATA(NCON+6)
PSICAB = DATA(NCON+7)
PO = DATA(NCON+8)
TC = DATA(NCON+9)
CV = DATA(NCON+10)
TE = DATA(NCON+11)
FLOM = 0.0
PSIN = 0.0
TIM = 0.0
FLOCP = 0.0
DO 9 I=1,NC
9 LOC = NSY + I + 1
LOC2 = LOC + 2
FLO = DATA(LOC)
IFABS(INDATA(LOC3).LE. 9999 .AND. IABS(NDATA(LOC2)).GT. 0)
FLO = POL(NDATA(LOC2),TIME)
PSI = DATA(LOC)
IFABS(INDATA(LOC3)).LE. 9999 .AND. IABS(NDATA(LOC2)).GT. 0)
PSI = POL(NDATA(LOC2),TIME)
TEMP = DATA(LOC2)
IFABS(INDATA(LOC2)).LE. 9999 .AND. IABS(NDATA(LOC3)).GT. 0)
TEMP = FLO + POL(NDATA(LOC3),TIME)
FLOM = FLOM + FLO
PSIN = PSIN + PSI
CPIN = POL(NCPA,TEMP)/PSI*POL(NCPV,TEMP) + (1.5 + PSI)
TIM = TIM + FLO + FLOM + TEMP
CONTINUE
FLOM = FLOM + FLO
PSIN = PSIN + PSI
CPIN = POL(NCPA,TEMP)/PSI*POL(NCPV,TEMP) + (1.5 + PSI)
TIM = TIM + FLO + FLOM + TEMP
CONTINUE
PSIN = PSIN + FLOW
FLOM = FLOM + FLO
FLOCP = FLOCP + FLO + CPIN
CONTINUE
5 CONTINUE
PSIN = PSIN + FLOW
TIM = TIM + FLOM
FLOM = POL(NDATA(NCON+1),TIME)
FLOM = FLOM + TIM + PSI*IN(1.0+PSIN)
RI = RA*1.0+PSI*IN(1.0+PSIN)
RHIN = POL(RH.IN,TIME)
FLOAT = FLOM + RHI + PSI*IN(1.0+PSIN)
WOUT = FCTRL + PSI*IN(1.0+PSIN)
WC = NW + WAVIN - WOUT - SOUT
DATA(NCON+63) = WC
NW = NW + WAVIN - WOUT - SOUT
DATA(NCON+61) = NW
PW = NW + WAVIN + TE-TZ)/VC
PR = PC + PW
WA = VC*PA/PA(TC-TZ)
#SICAB = NW/VC
DATA(NCON+61) = PW
DATA(NCON+61) = WC
UV = POL(I ODATA(NCVX=1),TC)  
CPA = POL(NCPA,TC)  
CPV = POL(NCV,TC)  
DC = POL(I ODATA(NCRX=1),TC)  
CN = POL(I ODATA(NCRX=1),TC)  
CC = POL(I ODATA(NCVX=1),TC)  
UC = CPA*PSICAB+CPV*VCX*PSICAB  
CPC = CPA*PSICAB+CPV*(VCX+PSICAB)  
CC = CPA*PSICAB+CPV*VCX*PSICAB  
RC = CPA*PSICAB+CPV*(VCX+PSICAB)  
TC = TC + (I ODATA(NCVX=1) - SUBJ(CPC31*WV*ICV))  
PRC = CCP*RC + CPC  
SIMU = 0.0  
SUMUL = 0.0  
LL = LAR  
1.  
IF(LHFB .eq. 0) GO TO 35  
PRC31 = PRC**.31  
IF(NOATA(LHFB,4) .ne. 0) CALL ERR(3NHTM)  
G3 20 I=1,MLT,4  
LOC = LHTB + I  
J = NOATA(LI)  
DI = DATA(LC+13)  
HI = DATA(LC+2)  
VHO = DATA(LC+8)  
VE = VHO+FLUC  
ME = VE*DI*VHO/UC  
IF(LHFB .le. 12000) SUBJ(RE**10) = 0.0  
WNU = 0.03 + 0.53*SORT(RE)  
GO TO 15  
IF(JU .eq. 0) SUBJ = 0.01  
IF(JU .ne. 0) SUBJ = 0.0065*RE**0.005+PRC31  
IF(JU .le. 15) SUBJ = 0.01  
MA = RIC+SHU+DI  
CALL OSUM  
IF(JU .eq. 0) CONTINUE  
20 CONTINUE  
25 IF(LHFP .eq. 0) GO TO 35  
PRC33 = ERTA(PRC)  
IF(NOATA(LHFP,5) .ne. 0) CALL ERR(3NHTM)  
G3 30 I=1,MLT,5  
LOC = LHFP + I  
J = NOATA(LI)  
DI = DATA(LC+13)  
HI = DATA(LC+2)  
VHO = DATA(LC+8)  
VE = VHO+FLUC  
VRU = VE*DI*VHO/UC  
SUBJ = 0.066+PRC33+SORT(VRU)/(XX+II+10) - SORT(VRU)/XX  
MA = RIC+SHU+RE  
CALL OSUM  
30 CONTINUE  
35 IF(LHC .eq. 91) GO TO 45  
IF(NOATA(LHC,29) .ne. 0) CALL ERR(3HSTH)  
G3 40 I=1,MLT,2  
LOC = LHC + I  
J = NOATA(LI)  
MA = DATA(LC+11)  
CALL OSUM  
40 CONTINUE  
45 WVPAPE = WY + WVIN - WVOU - SUMUL  
CABIN
PVPAM = WPVPM4RY(TC-TZ)/VC
SUMA = 0.
RING = RESTC(AR)/2 + AR
LJ = LJ + 1
IF(LJ.GT. 0) CALL CONCK(NN,NUM,IND)
IF(LHJ.GT. 0) CALL CONCK(NH,LMH)
IF(LHH.GT. 0) CALL CONCK(NH,LMHH)
RETURN

SUBROUTINE CONCK(NN,NUM,IND)
DO 60 I=1,NUM,IND
LL = LL + 1
PVI = EV(LI)
IF(PVPRPE-PVI)IPVPM .LT. 0.0) DWE = DWE+PVPW/(PV-PVPM)
IF(EV(LI))EVL .LT. 0.0) DWE = -EV(LI)
SUM = SUM + DWE
QL = QL+POL(LAMDA,T(J))TINC
O(J) = O(J) + QL
60 CONTINUE
RETURN

SUBROUTINE ERR(NUMB)
CALL TOPLIN
WRITE(6,100)
100 FORMAT(131(IH,1X,A3,*))

SUBROUTINE OSM(N)
IF(EXPL) DTAU(J) = DTAU(J) + HA
O(J) = O(J) + HA*(TC-T(J))
QL = QL+H0*(TC-T(J))*TINC
SUM = SUM + O(J)
SUMP = SUM + QL
EVTLL = PVI
O(J) = O(J) + QL
SUM = SUM + DWE
RETURN
END
SUBROUTINE EMPASS 1PR, A, B, L28, L1

THIS SUBROUTINE REDUCES THE COEFFICIENT MATRIX FOR PFCS

DIMENSION A(L1), B(L1), L28(L1)

LOCATE RELATIVE PRESSURE NODE NUMBER (1PR) OF ACTUAL PRESSURE

NODE NUMBER (1PR) WHICH HAS A SPECIFIED PRESSURE (SPR)

MPR1 = L28(I)

DO 5 NPR = 1, A

IF(IPR.EQ. L28(NPR+1)) GO TO 10

CONTINUE

RETURN &

CALCULATE THE STARTING LOCATION OF COLUMN NPR

LOC = LOC + I

RETURN

MODIFY THE RIGHT HAND SIDE

DO 20 J=I, PFR

IF(J .EQ. L28(NPR+1)) GO TO 25

B(J) = B(J) - SPR

CONTINUE

CALCULATE THE STARTING LOCATION OF ROW NPR

LOC = LOC + I

RETURN

DELETE COLUMN NPR

DELETE ROW NPR

IF(IPR .EQ. L28(NPR+1)) GO TO 25

LOC = LOC + I

RETURN

DELETE ACTUAL PRESSURE NODE 1PR FROM THE LIST OF ACTUAL PRESSURE NODES

END
SUBROUTINE CNFAST

C AN EXPLICIT EXECUTION SUBROUTINE FOR SIMA FORTRAN V
C THE SHORT PSEUDO COMPUTE SEQUENCE IS REQUIRED
C NODES WITH E5G BELOW DICMRI RECEIVE STEADY STATE SOLUTION
C NO BACKING UP IS DONE OR ALLOWED
C INCLUDE CONP.List
C INCLUDE DEF,LIST
C IF(ON(51).L.E.0) CON(1) = 1
C IF(ON(32).L.E.0.5) CON(2) = 1.E-8
C IF(ON(33).L.E.0.3) CON(3) = 1.0
C IF(ON(181).L.E.0.3) GO TO 999
C IF(ON(199).L.E.0.7) CON(191) = 1.E-8
C IF(ON(211).L.E.0.3) GO TO 999
C IF(ON(311).L.E.0.1) GO TO 995
C PASS = -1.0
C IF(NINC = NMX+MND) IE = NTH
C NLR = NDLN
C NTH = NTH+MND
C NDIM = NDIM-MND
C IF(ENDINT.LT.0) GO TO 997
C TSTEP = CON(11)
C TSTEP = CON(11) TSTEP = 0.84(ON(18) TSTEP)
C D0 30 I = NND
C LE = IE+1
C IF(INC(I).L.E.0.0) GO TO 40
C 02 30 I = NND
C 03 0(I) = 0.0
C CONTINUE
C KON(12) = 0
C CALL VARS
C IF(INC(12).NE.0) GO TO 10
C IF(INC(11).NE.0.1) GO TO 49
C PASS = -1.0
C IF(CON(1).L.E.0.) CON(1) = TPRINT
C IF(CON(2).L.E.0.) CON(2) = TSTEP
C CALL ITCAL
C CON(1) = TPRINT+TSTEP
C CON(2) = TSTEP
C 07 J1 = 0
C 08 J2 = 1
C 09 DI 85 I = 1,4N
C LE = IE+1
C INCLUDE VARS, LIST
C INCLUDE VARS, LIST
50 JJ = JJ+1
51 LE = FLOC(4,14,NSQ(I))
52 IF (E.Q.0.0) GO TO 95
53 TTA = FLC(23,14,NSQ(I))
54 INCLUDE YANG,LIST
55 FFLC(EQ.,1,NSQ(I,J)),EQ.0) GO TO 55
56 T1 = T1+450.0
57 T2 = TELTA*450.0
58 GV = (ELEM(T1,T2,T3)+T5)*T1
59 G2 TO 60
60 95 GV = (ELEM)
61 GO TO 95
62 60 QOST = GV*(TELTA-TC(T))
63 QST = GV*QOST
64 XLE = XLE+GV
65 EFL(EQ.,1,14,NSQ(I,J)),EQ.11) GO TO 45
66 ELE = ELEM(T1)
67 XLE4 = XLE4+GV
68 QLTA = (QLTA) - QOST
69 65 IF(WS(OJ,J1,GT.0) GO TO 56
70 65 CONTINUE
71 00 CM = 1.0
72 TCGM = 0.0
73 DJ 105 I = 1,NN
74 LE = 1E-1
75 TL = CE(I)/XLE
76 EFL(EQ.,CM) GO TO 90
77 CM = 11
78 90 CONTINUE
79 90 IF(TSTEP.GT.T1) GO TO 95
80 T1 = T1 + ISTEP*QST(T1)
81 GO TO 100
82 95 T1 = T1 + ISTEP*QST(T1)
83 100 T2 = ABS(T1-T11)
84 111 IF(T.T.EQ.1) GO TO 600
85 CONTINUE
86 01 CON(13) = TCGM
87 CON(14) = CM
88 IF(NAH.EQ.0.1) GO TO 96
89 TENC, LE, 0 GO TO 160
90 LAX = KON(5)
91 DAMP = CON(9)
92 DAMP1 = 1.0-DAMP
93 DJ 150 I = 1,LAY
94 KON201 = 1
95 REL = 0.0
96 100 JJ = JJ + 1
97 JJ = JJ + 2
98 105 DO0 11,141
99 106 KON202 = 1
100 REL = 0.0
101 NQ = NQ
102 IF(EQ.,1) GO TO 6000
103 INCLUDE VARZ,LIST
104 INCLUDE VARZ,LIST
105 JJ3 = JJ1+1
106 LG = FLC(5,16,NSQ(I,J3))
107 LTA = FLC(22,14,NSQ(I,J3))
108 EFL(EQ.,1) GO TO 4000
109 INCLUDE VARZ,LIST
110 CHECK FOR RADIATION CONDUCTIVITY
111 IF(EQ.,1,1,NSQ(I,J11)) GO TO 115
112 T1 = TELTA*450.0
113 115
T2 = TEL(1)+60.0

T2 = TEL(1)+60.0

GO TO 120

GO TO 120

IF (C2*3.I.E.3) GO TO 110

IF (C2*3.I.E.3) GO TO 110

IF (C2*3.I.E.3) GO TO 110

CALL VAR!CAL

CALL VAR!CAL

WRITE (10,1000)

WRITE (10,1000)

WRITE (10,1000)

WRITE (10,1000)

END
SUBROUTINE CORBIN(NTAPE, IT, INC, IUNIT)

DIMENSION NBUFR(21), ALPHA(NE), XSTART(I), ISTOPI = 11, NEM

DATA ALPHA /1, ICH, IHB, tHC, tHD, tHS, tEH, IIF, IHS, HH, INH/11, NEM

FORMAT(6I5.3)

IUNIT = IUNIT - 11

L = IUNIT

M = 0

I = L * IUNIT

READ(I) TIME,(ATA(K), K = 1, NTOTAL)

IF (TIME - ISTART(L)) 21, 30

WRITE(MT) TIME, (DATA(K), K = 1, NTOTAL)

IF (TIME - START(L)) 21

NTOTAL = NTOTAL * NBUFR(J)

READ(I) TIME, (DATA(K), K = 1, NTOTAL)

IF (TIME - ISTART(L)) 21

M = M - 1

MrINC

XTIME = TIME

WRITE(6, 31) TIME, L

X = 31 FORMAT(F10.5, PH HAS BEEN LOADED FROM TAPE 11)

READ(I) TIME, (DATA(K), K = 1, NTOTAL)

IF (TIME - ISTART(L)) 21

WRITE(MT) TIME, (DATA(K), K = 1, NTOTAL)

IF (NEW) ON NEW TAPE I)

REWIND I

REWIND I

WRITE(, 36) L, ITIME

FORMAT(X) X = 36 FOR RATING

ENDING 10H

E-NEU
39 CONTINUE
SPE: LE 15 WRITE (4, 92) INTAPE, ALPHACET
42 FORMAT (I10)
43 RETURN
44 END
1. SUBROUTINE CONDUCTION(NLOC)
2. C DIMENSION NLOC(1)
3. COMMON /ARRAY/ NDATA(1)
4. C COMMON TSTEP / T (1)
5. COMMON TCOND / G (1)
6. COMMON FTSTEP/냅, TMAX, VT, NLT
7. C IFMOD11(1,11),EO. 03 GO TO 20
8. CALL TOPLIN
9. WRITE(*,100) NLOC(1)
10. IF FORMAT(F90.4) * * INCORRECT NUMBER OF ELEMENTS INPUT TO CONDUCTION FOR
11. K CONDUCTION DATA, ICE = IS, TH = ** **
12. CALL WRFILE
13. C CALL EXIT
14. C
15. 20 IC = NLOC(1)
16. GO TO 110
17. IC = NLOC(+1)
18. IF (1) GO TO 19
19. WRITE(*,110) IC, NLT, NLT, NLT, NLT, NLT, NLT, NLT
20. CALL TOPLIN
21. WRITE(*,120) IC, NLT, NLT, NLT, NLT
22. CALL EXIT
23. C
24. 10 FORMAT(F90.4) * * ERROR IN CONDUCTION DATA INPUT TO CONDUCTION FOR
25. K // SHOWN = IC, 6H NLT = IS, 6H ATIF = IS, 6H ATMP = 16
26. CALL WRFILE
27. C CALL EXIT
28. C
29. 110 CONTINUE
30. C RETURN
31. END
SUBROUTINE CONDT2

DIMENSION MLOC(13)

COMMON /COND / 5 (C13)

COMMON /DIMENS/ N0C, N4A, NNT, NET

C EQUIVALENCE (INTERP,FTEMP)

IF(PL3C(NLOC(I)),3)

CALL TOPLIN

WRITE(6,10) NLOC(I)

10 FORMAT(51HERROR IN CONDUCTION DATA IC = 15, TH = +1)

CALL WERBIC

CALL EXIT

IC = NLOC(I)

DO 10 I=1,IC,3

NG = NLOC(I+2)

N4ITME = NLOC(I+3)

FTEMP = NLOC(I+4)

IF(NG .LE. NGT) GO TO 80

CALL TOPLIN

WRITE(6,60) (NLOC(I*J),J=1,3)

60 FORMAT(55HO0 * * ERROR IN CONDUCTION DATA INPUT TO CONDT2 * * *

SNG) = FTIME*FTEMP

CONTINUE

RETURN

END
SUBROUTINE CONV,LLOC,MLOC,NLOC

LOGICAL LCP, LMU, LTC

DIMENSION (LLOC(1), MLOC(1), NLOC(1))

COMMON /ARRAY/ NOATA(I)
COMMON ITEM, T(I)
COMMON /COND/ 6(I)
COMMON /D10tENS/ NND, NNA, NNT, NGT

EQUIVALENCE (ROATA,NDATA)
EQUIVALENCE (NAHT,AHT), (NX,I), (INF, Fi), (NFZ,F2)

CATA MAll 165000/

LLOC(I).EO. N0. GO TO 20
CALL TOPLIN
WRITE(6,T0)

IF (LLOC(I).LT. I) GO TO 20
CALL TOPLIN
WRITE(6,50)

IF (NOATA(L).EQ. 0) GO TO 30
CALL TOPLIN
WRITE(6,70)

IF (MLOC(I).LT. I) GO TO 80
CALL TOPLIN
WRITE(6,90)

IF (LCP) LCP = .TRUE.

NO = FLOC(2)
IF (NO .LT. I) OR (NO .GT. MAX) GO TO 82
LCP = .TRUE.
I10 IF(INFL .LT. 69.0) THEN
120 RCATA(LTYPE*I)
130 RDATA(LTYPE)
140 CALL DIOEG1(T(NFL),NDATA(NTC),CP)
150 CALL EXIT
160 IF(MMO(NLOC(I),) .LE. 86 LMU : .FALSE.
170 IF(NI .LT. 61.02 LCP = 0)
180 IF(MMU .EQ. 0) THEN
190 IF(MMM .GT. 61.0)
200 WRITE(*,*) "INCORRECT NUMBER OF ELEMENTS INPUT TO CONVI FOR CONVECTION DATA, NC = I5, NS = 00"
210 CALL MARCH
220 CALL EXIT
230 IF(MMM .EQ. 0) THEN
240 WRITE(*,*) "INCORRECT NUMBER OF ELEMENTS INPUT TO CONVI FOR CONVECTION DATA, NC = I5, NS = 00"
250 CALL MARCH
260 CALL EXIT
180 IF(LT.6.00.0.0.00.0 GO TO 190
190 M = 0.0114+T/C+4**-(1-0.0114+T/C)
200 GO TO 200
210 CRETURN
220 CEND
SUBROUTINE CONV2(LLOC,MLOC,NLOC)

C LOGICAL LCP, LRU, LTC
C
C DIMENSION LLOC(1), MLOC(11), NLOC(1)
C DIMENSION NOATA(11)
C
C COMMON /ARRAY/ NDATA(11)
C COMMON /ITEMP/ ITEMP(I)
C COMMON /ICONO/ G(I)
C
C COMMON IOIENSI NND, NNA, NWT, NGT
C
C EQUIVALENCE (ROATA,NDATA)
C EQUIVALENCE (NAHT,ANTE), (INS,1)
C
C DATA PSAXI = 165000/
C
C IF(MLOC(I).GT.60 TO 20
C CALL EXIT
C
C L2 = LLOC(2)
C LI = LLOC(7)
C IF(MLOC(I).GT.60 TO 40
C CALL EXIT
C
C IF(MLOC(I).GT.60 TO 96
C CALL EXIT
C
C 75 IF(MLOC(I).GT.60 TO 80
C CALL EXIT
C
C 80 IF(MLOC(I).GT.60 TO 82
C LCP = .TRUE.
C
GO TO 84
82 IF(ORGM .GT. 0) GO TO 86
83 CP = X
84 MK = NLOC
85 IF(ORGM .LT. 1 .OR. MK .GT. MAXI) GO TO 86
86 LPU = .FALSE.
87 RPU = MK
88 GO TO 84
89 B6 LPU = .FALSE.
90 RPU = MK
91 IF(ORGM .LT. 1 .OR. MK .GT. MAXI) GO TO 92
92 LPU = .FALSE.
93 T(L) = X
94 C 94 IF(ORGM .LT. 1) GO TO 100
95 CALL TOPLIN
96 WRITE(6,1120) NLOC(1)
97 CALL CONV2
98 CALL LTYPE
99 CALL EXIT
100 IC = NLOC(1)
101 GO TO 110
102 IF(ORGM .LT. 1) GO TO 110
103 IF(ORGM = 1) GO TO 110
104 IF (LTYPE .LT. 1 .AND. NLOCAL) GO TO 110
105 CALL CONV2
106 CALL EXIT
107 C 107 IF(ORGM .LT. 1) GO TO 110
108 IF(ORGM = 1) GO TO 110
109 IF(ORGM .LT. 1 .AND. NLOCAL) GO TO 110
110 CALL CONV2
111 WRITE(6,1120) NLOC(1)
112 CALL CONV2
113 CALL LTYPE
114 CALL EXIT
115 C 115 IF(ORGM = 1) GO TO 110
116 IF(ORGM .LT. 1 .AND. NLOCAL) GO TO 110
117 IF(ORGM .LT. 1 .AND. NLOCAL) GO TO 110
118 IF(ORGM .LT. 1 .AND. NLOCAL) GO TO 110
119 IF(ORGM .LT. 1 .AND. NLOCAL) GO TO 110
120 CALL CONV2
121 C 121 IF(ORGM = 1) GO TO 110
122 IF(ORGM .LT. 1 .AND. NLOCAL) GO TO 110
123 IF(ORGM .LT. 1 .AND. NLOCAL) GO TO 110
124 IF(ORGM .LT. 1 .AND. NLOCAL) GO TO 110
125 IF(ORGM .LT. 1 .AND. NLOCAL) GO TO 110
126 CALL CONV2
127 C 127 IF(ORGM .LT. 1) GO TO 110
128 IF(ORGM .LT. 1 .AND. NLOCAL) GO TO 110
129 IF(ORGM .LT. 1 .AND. NLOCAL) GO TO 110
130 IF(ORGM .LT. 1 .AND. NLOCAL) GO TO 110
131 IF(ORGM .LT. 1 .AND. NLOCAL) GO TO 110
132 CALL CONV2
133 C 133 IF(ORGM = 1) GO TO 110
134 IF(ORGM .LT. 1 .AND. NLOCAL) GO TO 110
135 IF(ORGM .LT. 1 .AND. NLOCAL) GO TO 110
136 IF(ORGM .LT. 1 .AND. NLOCAL) GO TO 110
137 IF(ORGM .LT. 1 .AND. NLOCAL) GO TO 110
138 CALL CONV2
139 C 139 IF(ORGM = 1) GO TO 110
140 IF(ORGM .LT. 1 .AND. NLOCAL) GO TO 110
141 IF(ORGM .LT. 1 .AND. NLOCAL) GO TO 110
142 IF(ORGM .LT. 1 .AND. NLOCAL) GO TO 110
143 IF(ORGM .LT. 1 .AND. NLOCAL) GO TO 110
144 CALL CONV2
SUBROUTINE CONV3(LOC, MLOC)

DIMENSION LOC(L1), MLOC(L1)

COMMON [ARRAY MLOC(L1)]
COMMON [TEMP / T ]
COMMON [C] / G /
COMMON [DIMENS AND, NA, ANT, GST]

EQUIVALENCE (RDATA, MLOC)
EQUIVALENCE (NANT, ANT)

DATA PART / 650000/

IF (MOD(NLOC(I), 4)) .GT. NST) GO TO 110

CALL TOPLIN

FORMAT (72H40 ARITE(6, 30)

CALL WLREC

CALL EXIT

20 L2 = MLOC(I)

L7 = MLOC(I)

IFPDATA(L1, GT, 0) GO TO 60

CALL TOPLIN

WRITE(10, 10) MLOC(L1)

30 FORMAT(200, * * INCORRECT NUMBER OF ELEMENTS INPUT TO CONV3 FOR

IFLOW DATA, IC = 15, TH = ** **

CALL WLREC

CALL EXIT

C

90 IFM(DL, LOC1, 4), EQ, 0) GO TO 100

CALL TOPLIN

WRITE(6, 90) MLOC(L1)

90 FORMAT(200, * * INCORRECT NUMBER OF ELEMENTS INPUT TO CONV3 FOR

ICONNECTION DATA, IC = 15, TH = ** **

CALL WLREC

CALL EXIT

C

100 IC = MLOC(I)

DO 220 I=2, IC

AG = MLOC(I+1)

NANT = MLOC(I+2)

TUBE = MLOC(I+3)

NWU = MLOC(I+4)

IF (AG .GT. AGT) GO TO 110

IF (TUBE .LE. MDATA(L1)) GO TO 170

110 CALL TOPLIN

WRITE(6, 120) MLOC(L1+1, J1, J2)

120 FORMAT(* * ERR32 IN CONNECTION DATA INPUT TO CONV3 * *)

5

10 INPUT = AG, TH = 123.8, THM = 14, 6H ANM = 15)
CALL WY!F
CALL EXIT
C
170 CALL OIDE(1,1)END
C
RETURN
END
123. IF (.NOT. COP) GO TO 630
124. CALL LINECK(2, )
125. CALL GENOUT(1, 0, 'MATERIAL BEFORE REDUCTION')
126. CALL LINECK(2,)
127. CALL GENOUT (L26=1, L, NEXT(L26), 'PRESSURE NODES')
128. CALL LINECK(2,)
129. CALL GENOUT (L26=3, L, NEXT(L26)-2, 'COEFFICIENT MATRIX')
130. CALL LINECK(2,)
131. CALL GENOUT (L26=1, L, NEXT(L26), 'RIGHT HAND SIDE')
132. C
133. C IMPOSED FLOW RATES INTO RHS
134. C
135. 630 IF (.NOT. LIFR) GO TO 690
136. DO 680 J = 1, NPR-1 + 1
137. NPR = NEXT (L25 * J + 3)
138. ET(L27*J) = ROTAT(L6 + NPR)
139. 680 CONTINUE
140. C
141. C INLET FLOW RATE INTO RHS
142. C
143. 690 N = NPR + 3
144. IF (NPI .LT. 1) GO TO 170
145. NIFNR = NPI
146. CALL PRN (NEXT (L25), N, NIFNR)
147. IF (N .LE. NPRN) GO TO 160
148. NIFN = NPI
149. 720 CALL TOPLIN
150. WRITE (16, 70) NIFN, NATA(L14 + 1)
151. 740 FORMAT (69HO)*. ERROR IN LOCATING PRESSURE NODES
152. IF (L22 .LT. 1) GO TO 840
153. IF (NDATA(L22) .LT. 1) Go TO 40
154. L60 = OATA(L22)
155. O0 = 820 J = 1, O0
156. NSPRN = NDATA(L22 * J)
157. CALL CMPRSSROATA(L3*NSPRN), NSPRN, ET(L26*3), ET(L27*3),
158. NEXT(L26), 860)
159. 820 CONTINUE
160. APR = .FALSE.
161. IF (NDP3) 920, 920, 925
162. APR = 0.
163. 920 CALL TOPLIN
164. WRITE (E, 880) NSPRN, NATA(L14 + 1)
165. 880 FOR PAREOHO.*. ERROR IN LOCATING PRESSURE NODES WITH IMPOSED FLO
166. IF (NPI .LT. 1) GO TO 710
167. IF (L22 .LT. 1) GO TO 840
168. IF (NDATA(L22) .LT. 1) Go TO 40
CALL MKBCK
CALL EXIT

C OUTFLET PRESSURE INTO COEFFICIENT MATRIX AND RHS

C SOLVE FOR PRESSURES

920 PFRN = NEXT(L28)
   NEXT(L26) = PFRN + PPRTM + 1/2 * 2
   NEXT(L26+1) = PFRN
   NEXT(L26+2) = PFRN
   IF.(MST. < CP) GO TO 930

CALL LINECK(2)

CALL GENOUT(I,I,0,MAITRII AFTER REDUCTION)

CALL LINECK(2)

CALL GENOUT(EXT(L28*I),1,NCATL28, "CPRESSURE TRIES")

CALL LINECK(2)

CALL GENOUT(EXT(L28*I),1,NCATL28, "CPRESSURE TRIES")

CALL LINECK(2)

930 CALL SVPSOL(EXT(L28*I),PPRN,EXT(L28*I),S1020)

IF(CP) CALL GENOUT(I,EXT(L28*I),NCATL28, "CPRESSURES")

DO 1060 **-1

1060 CALL TOPLIN

WRITE(10,100) DATA(L1+I)

1000 FORMAT(5H0*, ERROR IN SOLVING PRESSURE/FLOW EQUATIONS FOR NET)

IF(MST. > * **+1) CALL MKBCK

NTH = L25 - I

CALL JUTCAL

CALL EXIT

C UPDATE PRESSURES

DO 1120 J=1,FPRN
   NPR = NEXT(L28*J)
   RDATA(L3*NPR) = EXT(L28*J)

CONTINUE

C CALCULATE NEW FLOW RATES

DO 1120 J=9,L20,5
   NTJ = L14*J
   NTM = DATA(K)*NL
   NTM = NEXT(L25*NTM)
   TEMP = RDATA(L25+NTM) - RDATA(L25+NTJ)
   TEMP = NTM + PRM(TEMP-UNTB)
   RDATA(L25+NTM) = TEMP
   Q1 = RDATA(L120/TEMP)*Q1

C IF.(MST. < CP) GO TO 1120

CALL LINECK(3)

PNRM = RDATA(L3-NPM) + RFR

PNRJ = RDATA(L3-NTRJ) + RFR

WRITE(1100) NPR,NTJ,DATA(L14+NTJ),PRM*,PNRJ,UNTB,TEMP

CALL ELCT

CALL EXIT

END
1100 FORMAT/ 7X THMTB = 110, 8X THMTB = 110.  P4=2, GLOBAL
240.  1 0X THMTB = 110, 0X THMTB = 013.8, 320 THMTB = 013.8, NEW = 013.8, **-1 GLOBAL
250.  2 55 THMTB = 013.8, 55 THMTB = 013.8, 55 THMTB = 013.8, **-1 GLOBAL
251.  1120 CONTINUE
252.  IF (NOT. LPR) EQ TO 1160
253.  CONTINUE
254.  DO 1140 J3=1, XPAR
255.  RPR = NEXT(L3+23)
256.  RPR = ROR(L3+11)+RPR
257.  1140 CONTINUE
258.  RPR = ROR(L3+11)+RPR
259.  1160 RTH = L25 - 1
260.  RETURN
261.  END

**-1 GLOBAL
**-1 GLOBAL
**-1 GLOBAL
**-1 GLOBAL
SUBROUTINE FLOCNI(LLOC, MLOC, NLOC)
LOGICAL LCP
DIMENSION LLOC(1), MLOC(1), NLOC(1)
COMMON STEM / T /, COMON /END /, COMON /STEP/, NADC, NADI, NANT, NGT
COMMON /TEMPS/ NIND, NNA, TNT, NNT

I. EQUIVALENCE (NND, NNA, NNT, NGT)
II. IF(NTLOC(I).LT. 0) GO TO 15
III. CALL TOPLIN
IV. WRITE(6,IO) LLOC(I)
V. 10 FORMAT(75H0 INCORRECT NUMBER OF ELEMENTS INPUT TO FLOCNI FOR FLOCNI
VI. I FLOW RATES, IC = 15, TN = **) FLOCNI
VII. CALL WLKBCK
VIII. RETURN
IX. IF(NW = LLOC(I))
X. IF(NW .LT. I OR NW .GT. 65000) GO TO 13
XI. LCP = .TRUE.
XII. CP = NW
XIII. GO TO 25
XIV. LCP = .FALSE.
XV. REP = NW
XVI. CALL TOPLIN
XVII. WRITE(6,30) NLOC(I)
XVIII. 30 FORMAT(83H0 INCORRECT NUMBER OF ELEMENTS INPUT TO FLOCNI FOR FLOCNI
XIX. I FLOW CONDUCTION DATA, IC = 15, TN = **) FLOCNI
XX. CALL WLKBCK
XXI. CALL EXIT
XXII. IF(NW = LLOC(I))
XXIII. IF(NW .LT. I OR NW .GT. 65000) GO TO 13
XXIV. LCP = .TRUE.
XXV. CP = NW
XXVI. GO TO 25
XXVII. LCP = .FALSE.
XXVIII. REP = NW
XXIX. CALL TOPLIN
XXX. WRITE(6,80) (NLOC(I+J), J=I,3)
XXXI. 80 FORMAT(60H0ERROR IN FLOW CONDUCTION DATA INPUT TO FLOCNI *)
XXXII. CALL WLKBCK
XXXIII. CALL EXIT
XXXIV. NW = LLOC(I)
XXXV. IC = NLOC(I)
XXXVI. DO 20 I=1,IC,3
XXXVII. NG = NLOC(I+1)
XXXVIII. NDL = NLOC(I+2)
XXXIX. ITUBE = NLOC(I+3)
XC. IFNG .GE. .GT. NDL) GO TO 60
XL. IFITUBE .LT. NDL) GO TO 60
XLI. IFITUBE .LT. NANT) GO TO 100
XLII. DO CALL TOPLIN
XLIII. WRITE(6,90) (NLOC(I+J), J=1,3)
XLIV. 90 FORMAT(60H0 ERROR IN FLOW CONDUCTION DATA INPUT TO FLOCNI *)
XLV. CALL WLKBCK
XLVI. CALL EXIT
XLVII. NW = LLOC(I)
XLVIII. IC = NLOC(I)
XLIX. DO 20 I=1,IC,3
CXX. NG = NLOC(I+1)
CXXI. NDL = NLOC(I+2)
CXXII. ITUBE = NLOC(I+3)
CXXIII. IFNG .GE. .GT. NDL) GO TO 60
CXXIV. IFITUBE .LT. NDL) GO TO 60
CXXV. IFITUBE .LT. NANT) GO TO 100
CXXVI. DO CALL TOPLIN
CXXVII. WRITE(6,90) (NLOC(I+J), J=1,3)
CXXVIII. 90 FORMAT(60H0 ERROR IN FLOW CONDUCTION DATA INPUT TO FLOCNI *)
CXXIX. CALL WLKBCK
CXXX. CALL EXIT
CXXXI. NW = LLOC(I)
CXXXII. IC = NLOC(I)
CXXXIII. DO 20 I=1,IC,3
CXXXIV. NG = NLOC(I+1)
CXXXV. NDL = NLOC(I+2)
CXXXVI. ITUBE = NLOC(I+3)
CXXXVII. IFNG .GE. .GT. NDL) GO TO 60
CXXXVIII. IFITUBE .LT. NDL) GO TO 60
CXXXIX. IFITUBE .LT. NANT) GO TO 100
CXL. DO CALL TOPLIN
CXLI. WRITE(6,90) (NLOC(I+J), J=1,3)
CXII. 90 FORMAT(60H0 ERROR IN FLOW CONDUCTION DATA INPUT TO FLOCNI *)
CXIII. CALL WLKBCK
CXIV. CALL EXIT
CXV. NW = LLOC(I)
CXVI. IC = NLOC(I)
CXVII. DO 20 I=1,IC,3
CXVIII. NG = NLOC(I+1)
CXIX. NDL = NLOC(I+2)
CXX. ITUBE = NLOC(I+3)
CXXI. IFNG .GE. .GT. NDL) GO TO 60
CXXII. IFITUBE .LT. NDL) GO TO 60
CXXIII. IFITUBE .LT. NANT) GO TO 100
CXXIV. DO CALL TOPLIN
CXXV. WRITE(6,90) (NLOC(I+J), J=1,3)
CXXVI. 90 FORMAT(60H0 ERROR IN FLOW CONDUCTION DATA INPUT TO FLOCNI *)
CXXVII. CALL WLKBCK
CXXVIII. CALL EXIT
CXIX. NW = LLOC(I)
CXX. IC = NLOC(I)
CXXI. DO 20 I=1,IC,3
CXXII. NG = NLOC(I+1)
CXXIII. NDL = NLOC(I+2)
CXXIV. ITUBE = NLOC(I+3)
CXXV. IFNG .GE. .GT. NDL) GO TO 60
CXXVI. IFITUBE .LT. NDL) GO TO 60
CXXVII. IFITUBE .LT. NANT) GO TO 100
CXXVIII. DO CALL TOPLIN
CXXIX. WRITE(6,90) (NLOC(I+J), J=1,3)
CXXX. 90 FORMAT(60H0 ERROR IN FLOW CONDUCTION DATA INPUT TO FLOCNI *)
CXXXI. CALL WLKBCK
CXXXII. CALL EXIT
CXXXIII. NW = LLOC(I)
CXXXIV. IC = NLOC(I)
CXXXV. DO 20 I=1,IC,3
CXXXVI. NG = NLOC(I+1)
CXXXVII. NDL = NLOC(I+2)
CXXXVIII. ITUBE = NLOC(I+3)
CXXXIX. IFNG .GE. .GT. NDL) GO TO 60
CXXXX. IFITUBE .LT. NDL) GO TO 60
CXXXXI. IFITUBE .LT. NANT) GO TO 100
CXXXXII. DO CALL TOPLIN
CXXXXIII. WRITE(6,90) (NLOC(I+J), J=1,3)
CXXXXIV. 90 FORMAT(60H0 ERROR IN FLOW CONDUCTION DATA INPUT TO FLOCNI *)
CXXXXV. CALL WLKBCK
CXXXXVI. CALL EXIT
SUBROUTINE FLOCNZ(ILOC, NLOC)

DIMENSION MLOC(I), KLOC(I)  

EQUIVALENCE (NW, W)

IF (ILOC(I).GT. 0) GO TO 20  

CALL TOPLIN  
WRITE(6,10) MLOC(I)

10 FORMAT(6H0  

INCORRECT NUMBER OF ELEMENTS INPUT TO FLOCNZ FOR  
FLOW RATES, IC = 15, TH = *)  
CALL WRECK  
CALL EXIT

IF (ILOC(I).LE. 0) GO TO 40  

CALL TOPLIN
WRITE(6,80) (NLOC(I*J), J=1,2), CP  
FORMAT(80H0  

ERROR IN FLOW CONDUCTION DATA, IC = 15, TH = *)  
CALL WRECK  
CALL EXIT

NTUBE = NLOC(I+2)  

IF (NTUBE.GT. NGT) GO TO 60  

IF (ITUBE.LE. NTB) GO TO 100

CALL TOPLIN
WRITE(6,603) (MLOC(I+J), J=1,2), CP  
FORMAT(603H0  

ERROR IN FLOW CONDUCTION DATA INPUT TO FLOCNZ  
1 = // SHOAL = 14, NM ITUBE = 14, TH CP = 0.131)

CALL WRECK  
CALL EXIT

NW = MLOC(1)  
NG = MLOC(1+1)  
ITUBE = MLOC(1+2)

IF (IC.GT. 0) GO TO 60  

IF (IC.LE. 100) GO TO 100

CALL TOPLIN
WRITE(6,60) (MLOC(I+J), J=1,2), CP  
FORMAT(60H0  

ERROR IN FLOW CONDUCTION DATA INPUT TO FLOCNZ  
1 = // SHOAL = 14, NM ITUBE = 14, TH CP = 0.131)

CALL WRECK  
CALL EXIT

CONTINUE

RETURN

END
SUBROUTINE FLPRINT

4

C

DIMENSION DATA(13), HEAD(4)

C

COMMON /FICOM/ L(1)

C

EQUIVALENCE (ANT,DI)

C

1. G = DATA(1)

11. IF(L(29).LE.0 .OR. L(28).GE. 60) CALL TOPLIN

12. WRITE(6,101) HEAD

13. L(28) = L(28) + 1

14. NW = I

15. RF = 10

16. IF(NF,GE.,WNT) GO TO 20

17. WRITE(6,100) (DATA(I+1),I=NS,NF), NF

18. L(28) = L(28) + 1

19. IF(L(28).GE.60) CALL TOPLIN

20. IF(NF,GE.,WNT) RETURN

21. NW = RF + 1

22. GO TO 10

23. WRITE(6,100) (DATA(I+1),I=NS,NF)

24. IF(L(28).GE.60) CALL TOPLIN

25. RETURN

26. IF(L(28).GE.60) CALL TOPLIN

27. IF(L(28).GE.60) CALL TOPLIN

28. IF(L(28).GE.60) CALL TOPLIN

30. IF(L(28).GE.60) CALL TOPLIN

31. IF(L(28).GE.60) CALL TOPLIN

32. IF(L(28).GE.60) CALL TOPLIN

33. IF(L(28).GE.60) CALL TOPLIN

34. IF(L(28).GE.60) CALL TOPLIN

35. IF(L(28).GE.60) CALL TOPLIN

36. WRITE(6,100) (DATA(I+1),I=NS,NF)

37. IF(L(28).GE.60) CALL TOPLIN

38. IF(L(28).GE.60) CALL TOPLIN

39. IF(L(28).GE.60) CALL TOPLIN

40. IF(L(28).GE.60) CALL TOPLIN

41. IF(L(28).GE.60) CALL TOPLIN

42. IF(L(28).GE.60) CALL TOPLIN

43. IF(L(28).GE.60) CALL TOPLIN

44. IF(L(28).GE.60) CALL TOPLIN

45. IF(L(28).GE.60) CALL TOPLIN

46. IF(L(28).GE.60) CALL TOPLIN

47. IF(L(28).GE.60) CALL TOPLIN

48. IF(L(28).GE.60) CALL TOPLIN

49. IF(L(28).GE.60) CALL TOPLIN

50. IF(L(28).GE.60) CALL TOPLIN

51. IF(L(28).GE.60) CALL TOPLIN

52. IF(L(28).GE.60) CALL TOPLIN

53. IF(L(28).GE.60) CALL TOPLIN

54. IF(L(28).GE.60) CALL TOPLIN

55. IF(L(28).GE.60) CALL TOPLIN

56. IF(L(28).GE.60) CALL TOPLIN

57. IF(L(28).GE.60) CALL TOPLIN

58. IF(L(28).GE.60) CALL TOPLIN

59. IF(L(28).GE.60) CALL TOPLIN

60. IF(L(28).GE.60) CALL TOPLIN

61. IF(L(28).GE.60) CALL TOPLIN

62. IF(L(28).GE.60) CALL TOPLIN

63. IF(L(28).GE.60) CALL TOPLIN

64. IF(L(28).GE.60) CALL TOPLIN

65. IF(L(28).GE.60) CALL TOPLIN

66. IF(L(28).GE.60) CALL TOPLIN

67. IF(L(28).GE.60) CALL TOPLIN

68. IF(L(28).GE.60) CALL TOPLIN

69. IF(L(28).GE.60) CALL TOPLIN

70. IF(L(28).GE.60) CALL TOPLIN

71. IF(L(28).GE.60) CALL TOPLIN

72. IF(L(28).GE.60) CALL TOPLIN

73. IF(L(28).GE.60) CALL TOPLIN

74. IF(L(28).GE.60) CALL TOPLIN

75. IF(L(28).GE.60) CALL TOPLIN

76. IF(L(28).GE.60) CALL TOPLIN

77. IF(L(28).GE.60) CALL TOPLIN

78. IF(L(28).GE.60) CALL TOPLIN

79. IF(L(28).GE.60) CALL TOPLIN

80. IF(L(28).GE.60) CALL TOPLIN

81. IF(L(28).GE.60) CALL TOPLIN

82. IF(L(28).GE.60) CALL TOPLIN

83. IF(L(28).GE.60) CALL TOPLIN

84. IF(L(28).GE.60) CALL TOPLIN

85. IF(L(28).GE.60) CALL TOPLIN

86. IF(L(28).GE.60) CALL TOPLIN

87. IF(L(28).GE.60) CALL TOPLIN

88. IF(L(28).GE.60) CALL TOPLIN

89. IF(L(28).GE.60) CALL TOPLIN

90. IF(L(28).GE.60) CALL TOPLIN

91. IF(L(28).GE.60) CALL TOPLIN

92. IF(L(28).GE.60) CALL TOPLIN

93. IF(L(28).GE.60) CALL TOPLIN

94. IF(L(28).GE.60) CALL TOPLIN

95. IF(L(28).GE.60) CALL TOPLIN

96. IF(L(28).GE.60) CALL TOPLIN

97. IF(L(28).GE.60) CALL TOPLIN

98. IF(L(28).GE.60) CALL TOPLIN

99. IF(L(28).GE.60) CALL TOPLIN

100. IF(L(28).GE.60) CALL TOPLIN

END
SUBROUTINE FLRES(L30,MTB)

LOGICAL L92, LNU, COP

DIMENSION NDATA(L)

COMMON/ARRAY/NDATA(L)

COMMON/POINT/NDATA(L)

COMMON/NSPACE/NORM, NTH, NFR

COMMON/POINT/LNODE

1. EQUIVALENCE (RDATA,NDATA), (HL,NHL)

21. WNTB = ABS(RDATA(L2+MTB))

22. WM = 4,0*WNTB

23. RSUM = 0,0

24. IC = NDATA(L30)


26. DO 200 I=1,IC,3

27. = L30 * I

28. NFL = NDATA(K)

29. ITYPE = NDATA(K+1)

30. NTL = NDATA(K+2)

31. LTYPE = LT + ITYPE*6 - 6

32. WP = RDATA(LTYPE*1)

33. CSA = RDATA(LTYPE*2)

34. FLL = RDATA(LTYPE*3)

35. MFF = NDATA(LTYPE*9)

36. NHL = NDATA(LTYPE*5)

37. FFC = RDATA(LTYPE*6)

38. IF (L92) CALL DIOEGL(TINFL),NRO, R)

39. IF (LNU) CALL DIOEGL(T(NFL),NDATA(NMU),U)

40. IF (NHL .GT. 0 .AND. NHL .LT. MAXII) CALL DIOEGL();E,NDATA(NHL),L)

41. IF (RE .GT. 2000.0) GO TO 100

42. IF (RE .LT. 4000.0) GO TO 140

43. FF = 0.316/SQRT(SQRF())

44. GO TO 160

45. FF = 0.2086092052 - 0.655956/E-3 I

46. GO TO 160

47. RSUM = RSUM + R

48. GO TO 160

49. RSum = RSum + R

50. GO TO 160
C IF(NLT. COP) GO TO 300
62. CALL LINECK(5)
63. WRITE(6,220) NODE,NEXT,NEXTNL,TYPE,NEXTNL,NEXTNL,NEXTNL
64. IF(NLT. TYPE=10) THEN NXTL,NEXT=FLRES
65. 100 FORMAT(T7I(NFL) = T10, 6X THF, FLRES)
66. 1 X TYPE = 10, 6X THF = T10, 6X THF = T10, 6X THF = T10
67. 2 X THF = T10, 6X THF = T10, 6X THF = T10, 6X THF = T10
68. 3 X THF = T10, 6X THF = T10, 6X THF = T10, 6X THF = T10
69. 4 X THF = T10, 6X THF = T10, 6X THF = T10, 6X THF = T10
70. 5 X THF = T10, 6X THF = T10, 6X THF = T10, 6X THF = T10
71. 200 CONTINUE
72. C RDATA(N+NTB) = 1.0/RSOM
73. C IF(NLT. COP) GO TO 300
74. CALL LINECK(5)
75. WRITE(6,220) RDATA(N+NTB)
76. 220 FORMAT(T7I(NFL) = T10)
77. 300 RETURN
78. C 300 RETURN
79. C C END
SUBROUTINE FLUX(NFLITP, DATA, NCRV, DTIME, DTIME)
  
  DIMENSION DATA(*)
  
  COMMON /TIME/ TIPER
  
  EQUIVALENCE (0,N)
  
  IF(DTIME .GE. TIPER) RETURN

  IF(NFLITP .GT. 0) READ(NFLITP) FLITIM
  
  NFLITP = ABS(NFLITP - 1)

  IF(NFLITP .GT. 0) READ(NFLITP) FLITIM
  
  FLITIM = FLITIM + DTIME
  
  IF(DTIME .LE. TIPER) GO TO 10

  WRITE(6,20) DTIME
  
  LOC = I

  IFLOC = DATA(I)

  DO 30 J=1,IC,2
  
  DATA(I+J) = DATA(I+J) + DTIME
  
  LOC = LOC + IC + 1

  30 CONTINUE
  
  LOC = LOC + IC + 1

  NFLITP = -NFLITP

  RETURN

  END
SUBROUTINE GENOUT(DATA, ISTR, ISTP, NAME)

DIMENSION FM(12), NAME(22)
DIMENSION NAM(11)
DATA I, M / 33217720/
DATA FM(1), FM(2) / 6H1ISP, 6H1IO/
LOGICAL CKO, ONE

BASE = 6HE12.4,
ASSIGN 32 TO MM
ONE = .FALSE.

C = C

DIMENSION NOATA(I)

DATA MAll, I, 13, 21, 128, 1
DATA FMT(I), FMT(IZl / 6HIlIIP, I 0)

LOGICAL ONE, CKO

BASE T 6HE12.q,
ASSIGN 32 TO mM
ONE = .FALSE.

ENTRY GENI(NDATA, ISTR, ISTP, NAME)

BASE z 6HI9, 31
ASSIGN 95 TO M
CID = .TRUE.
WRITE(6, 10) NAME
FORMAT(ZZA6)
ONE = .FALSE.

IF(ISTRT .E. I AND. ISTP .EO. 1) ONE = .TRUE.
ENTRY GENR(NOATA, ISTRISTP, NAME)

CASE = 6HE12.4,
ASSIGN 32 TO MM
ONE = .FALSE.

C = C

CALL LINECK(I)

CONTINUE

IF((CKO .AND. ONE.) .OR. (CKO .AND. ONE.)

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

CONTINUE

IFEM = 0)
M = 10
J = J + 1
K = K - 1
L = L - 1
IFEM = EM)
M = 10
J = J + 1
K = K - 1
L = L - 1
IFEM = EM)
M = 10
J = J + 1
K = K - 1
L = L - 1
IFEM = EM)
M = 10
J = J + 1
K = K - 1
L = L - 1
IFEM = EM)
M = 10
J = J + 1
K = K - 1
L = L - 1
IFEM = EM)
M = 10
J = J + 1
K = K - 1
L = L - 1
IFEM = EM)
M = 10
J = J + 1
K = K - 1
L = L - 1
IFEM = EM)
M = 10
J = J + 1
K = K - 1
L = L - 1
IFEM = EM)
M = 10
J = J + 1
K = K - 1
L = L - 1
IFEM = EM)
M = 10
J = J + 1
K = K - 1
L = L - 1
IFEM = EM)
M = 10
J = J + 1
K = K - 1
L = L - 1
IFEM = EM)
M = 10
J = J + 1
K = K - 1
L = L - 1
IFEM = EM)
M = 10
J = J + 1
K = K - 1
L = L - 1
IFEM = EM)
M = 10
J = J + 1
K = K - 1
L = L - 1
IFEM = EM)
M = 10
J = J + 1
K = K - 1
L = L - 1
IFEM = EM)
M = 10
J = J + 1
K = K - 1
L = L - 1
IFEM = EM)
M = 10
J = J + 1
K = K - 1
L = L - 1
IFEM = EM)
M = 10
100 READ (30) TITLEA, TA, TPE, TPG
101 WRITE (6, MF) TITLEA, TA, TPE, TPG
102 IF (TPG .LT. 1.0) GO TO 103
103 WRITE (6, MF) TITLEA, TA, TPE, TPG
104 IF (TPG .LT. 100) GO TO 105
105 WRITE (6, MF) TITLEA, TA, TPE, TPG
106 WRITE (6, MF) TITLEA, TA, TPE, TPG
107 WRITE (6, MF) TITLEA, TA, TPE, TPG
108 WRITE (6, MF) TITLEA, TA, TPE, TPG
109 WRITE (6, MF) TITLEA, TA, TPE, TPG
110 WRITE (6, MF) TITLEA, TA, TPE, TPG
111 WRITE (6, MF) TITLEA, TA, TPE, TPG
112 WRITE (6, MF) TITLEA, TA, TPE, TPG
113 WRITE (6, MF) TITLEA, TA, TPE, TPG
114 WRITE (6, MF) TITLEA, TA, TPE, TPG
115 WRITE (6, MF) TITLEA, TA, TPE, TPG
116 WRITE (6, MF) TITLEA, TA, TPE, TPG
117 WRITE (6, MF) TITLEA, TA, TPE, TPG
118 WRITE (6, MF) TITLEA, TA, TPE, TPG
119 WRITE (6, MF) TITLEA, TA, TPE, TPG
120 WRITE (6, MF) TITLEA, TA, TPE, TPG
121 WRITE (6, MF) TITLEA, TA, TPE, TPG
122 WRITE (6, MF) TITLEA, TA, TPE, TPG
250, CONTINUE
124. C INDEX AND COUNT THE ITEMS IN THE HISTORY TAPE
125. KEVIM=1
126. GO TO 500,127
128. IF (ITEM(E).NE.0) GO TO 300
129. WRITE(6,390) NWR, CLOC(I)=I+1, KEY(1:16, KEYA(1:16), I)
130. 280 FORMAT(25X,25X,THE HISTORY TAPE LABEL.
131. 1X, I(16,24) /* MX 5 (16,23) */
132. 1X COUNTS ARE - A(16,23) / MX 5 (16,23)
133. 1X
134. 1X
135. 1X IFINIS = 0
136. JFINIS = 0
137. C READ THE ITEMS TO BE PLOTTED
138. ITEMS = 16
139. J = I
140. NOAVG = 0
141. KSW = 0
142. 320 READS,280) ITEMS,EITEM(E),EAVG,EKVE,
143. = (ITEMS(0,1:16),, LOC(E),, UNE(1)
144. 310 FORMAT(I6,280) ITEM(),ITYPE(I),IREL,KAVG,
145. *NEW
146. 300 FORMAT(I6,280) ITEM(),ITYPE(I),IREL,KAVG,
147. *NEW
148. C TEST FOR END OF JOB - BLANK CARD
149. IF (ITEM(E).LT.0) GO TO 20
150. IF (ITEM(E).LT.0) GO TO 340
151. KAVG = 0
152. NOAVG = 0
153. 340 CONTINUE
154. GO TO 340
155. 320 FORMAT(I6,280) ITEM(),ITYPE(I),IREL,KAVG,
156. *NEW
157. 310 FORMAT(I6,280) ITEM(),ITYPE(I),IREL,KAVG,
158. *NEW
159. C CHECK FOR NEW GRID SET SPECIFIED BY USER
160. IF (ITEM(E).LT.0) J = 0
161. IF (ITEM(E).LT.0) KSW = 0
162. J = J+1
163. ITYPLS(1:1) = BLANK
164. FLG(30,6,ITYPLS) = J + KSW + 80
165. IF(KAVG .EQ. 0 .OR. NOAVG .GE. 100) GO TO 345
166. NOAVG = NOAVG + 1
167. ITEM(NOAVG) = I
168. IF(KAVG .LT. 100) GO TO 345
169. ITEM(NOAVG) = -I
170. FLO(30,6,ITYPLS) = 6M YES
171. KSW = KSW + 1
172. C BUMP ITEM COUNTER AND CHECK FOR MAXIMUM NUMBER OF ITEMS
173. I(1:6,1).I. ITEMS) GO TO 320
174. 345 I = I + 1
175. IF(I .LT. I. ITEMS) GO TO 320
176. 346 ITEMS = I
177. 347 LOC(I) = J+1, NOAVG
178. NOAVG = 0
179. 370 CONTINUE
180. C SET FIRST ITEM FOR NEW GRID SET
181. ITEM(E) = -ITEM(E)
182. 330 LOC(I) = 1
183. I = I
184. 430 J = I
185. C
420 IF (TYPE(I),EQ., KEY(I))) GO TO 460
187. J = J + 1
188. IF (J .GT. 60) GO TO 420
189. C
190. LOC(I) = -1
191. IACT = I (ITEM)
192. IF (ITEM(I),NE., KEY(I))) GO TO 490
193. NN = IABS(I (ITEM)
194. IACT = NSIZE(ITEM(I)) ITEM(I))
195. 490 WRITE(6,490) IACT
196. C WRITE(I,80) ITEM(I),ITEM(I),8,2,TOT TYPE CODE IN EACH)
197. GO TO 540
198. C
199. 540 IF ((KEY(I) .EQ. -KEY(I))) = IABS(ITEM(I))) J = 0,520,520
200. C INCORRECT ITEM
201. 520 LOC(I) = -2
202. C WRITE(I,80) ITEM(I)
203. IACT = I (ITEM)
204. IF (ITEM(I),NE., KEY(I))) GO TO 490
205. NN = IABS(I (ITEM)
206. IACT = NSIZE(ITEM(I)) ITEM(I))
207. 490 WRITE(6,490) IACT, ITEM(I)
208. 500 FORMAT (490,WRITE(I,80) ITEM,ITEM(I),8,3,TOT IS OUT OF RANGE)
209. C
210. 590 LOC(I) = -3
211. C SAVE FUNCTION TYPE INCORRECT
212. 530 ISIZE(I) = 3
213. C
214. J = I + 1
215. IF (J .LT. ITEMS) GO TO 460
216. C START LOADING THE DATA FROM THE HISTORY TAPE FROM
217. 620 WRITE(6,520)
218. 520 WRITE(I,80)
219. C COMPUTE THE MAXIMUM NUMBER OF RECORDS
220. 650 NTYPE = NSIZE/ITEMS
221. C WRITE(I,80)
222. 500 FORMAT(490,WRITE(I,80)
223. C PRINT THE HISTORY TAPE FROM
224. NTPE = 0
225. J = 1
226. IF (J .LT. ITEMS) GO TO 460
227. C READ (HISTORY) BUFFER(I),L=1,WARDS)
228. C
229. IF (BUFFER(I),LT.0) GO TO 730
230. C
231. IF (BUFFER(I),GT.0) GO TO 620
232. GO TO 660
233. 660 IF (BUFFER(I),EQ.1) WRITE(I,80) BUFFER(I, TA
234. GO TO 660
235. C WRITE(I,80)
236. C WRITE(I,80)
237. C
238. C CHECK FOR REQUESTED FINAL TIME
239. 730 IF (BUFFER(I),.GT. TE) IFINS = 1
240. C PICK OF THE ITEM TYPE ARRAY QUANTITY OF
241. GO TO 420
242. M = LOC(I)
243. C
244. IF (M .LT. 0) GO TO 740
245. 740 J = J + 1
246. IF (BUFFER(I),LT.0 .GT. 752)
247. C BUMP THE RW ARRAY SUBSCRIPT
248. X = X + 1
249. GO TO 540
250. C
NTPTS = NTPTS + 1
STOP = BUF(13)
GO TO 49
MM = IEQ50(1AVG(50))
M = LCC40+13
ITEM (1, 0) GO TO 49
M = PM + J - ITEMS
AVGCL) = AVGCL) + MPM
IF NTPTS = 11) GO TO 43
IF IEQ40(1) = 63 TO 743
ISTART = 1 - 1
START = BUF(13)
AVGLOC (1) = BUF (13)
AVGLOC (1 + 507 = BUF (13)
AVG ((1 + 507 = RPM)
AVG ((1 + 1003 = RPM)
GO TO 749
743 IF IEQ(M) LE AVGCL (50)) GO TO 746
AVGLOC (1) = BUF (13)
AVG ((1 + 507 = RPM)
GO TO 749
746 IF IEQ(M) LE AVGCL (1003) GO TO 749
AVGLOC (1 + 507 = BUF (13)
AVG ((1 + 1003 = RPM)
GO TO 749
749 CONTINUE
752 LI = 3 - ITEMS + 1
LJ = J - 1
WRITE (6,760) (XYIL1,LYLJ,LJ)
FORMAT (10F1.3)
IF (EFINIS .EQ. 1) GO TO 800
I = I + 1
C CHECK FOR MAXIMUM NUMBER OF POINTS
780 NTPTS = I + 1
GO TO 820
800 NTPTS = I
C COMPUTE THE NUMBER OF WORDS USED IN THE BF ARRAY
820 NDTOL = J - 1
REWIND INSTAP
WRITE (6,840) ITEMS, 1, NDTOL
FORMAT (10E2.1,15H29 DATA VALUES HAVE BEEN STORED FOR EACH OF,16,
I 13H TIME POINTS/(T2,0,0 DATA VALUES HAVE BEEN STORED)
840 FORMAT (10F1.3)
C IF IEQ(M) GO TO 652
860 WRITE (6,840) MON, NTPTS, ASRT, MSTOP, START, STOP
840 FORMAT (10F1.3)
C SOME NUMERICAL AVERAGES FOR THE FOLLOWING ITEMS WERE REQUESTED
850 IF X FOR THE ITEM, TIME POINTS/* BEGINNING WITH FF3.3
860 IF X IS AND ENDING WITH FF3.3 ACTUAL TIMES = FF3.3
870 IF X IS AND ITEM TYPE = DESCRIB
880 X AN I AVERAGE TIME MAX VALUE/5X TIME MIN VALUE/5X TIME/5
890 NTIPS = NTIPS
900 GO TO 120
910 WRITE (6,840) X, NTIPS
920 FORMAT (15H29 VALUE (X, MIN VALUE (X), TIME (X))
930 C ITERATION (X)
940 IF IEQ(M) IF IEQ(M) IF IEQ(M)
950 IF IEQ(M) IF IEQ(M) IF IEQ(M)
960 IF IEQ(M)
<table>
<thead>
<tr>
<th>Line</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>501</td>
<td>1620 CONTINUE</td>
</tr>
<tr>
<td>502</td>
<td>100 = 0</td>
</tr>
<tr>
<td>503</td>
<td>END CALL ELNAME(D)</td>
</tr>
<tr>
<td>504</td>
<td>NCASE = NCASE + 1</td>
</tr>
<tr>
<td>505</td>
<td>60 TO 240</td>
</tr>
<tr>
<td>506</td>
<td>END</td>
</tr>
</tbody>
</table>
SUBROUTINE G0PLOT(NSYM)

DIMENSION ASYM(50),OSYM(50),BUF1(1000),ABS',13,ORD(11)

COMMON NPTS,NPP,IFG

EQUIVALENCE (BUF11,ABS111,BUF1(2001),ORD111)

INTEGER ISYM(2)

DATA ISYMAM13M.7M1/

FNP = NSYM + 2.4
DT = ABS(NPTS) - ABS13
NP = FNP + DT/TPG + 1.5

MPP = NP - 1
K = NPTS / MPP

ASYM(I) = ABS(I)

OSYM(I) = ORD(I)

ASYMAP = ABS(NPTS)

(OSYMNP = ORD(NPTS)

KK = 0
DO 16 I=2,MPP
KK = KK + K
16 CONTINUE

IX = NLYABS(KK)
IV = NLYORD(KK)
CALL REVEW(61.1Y,1023,90,1,1,SYM,SYM,

10 CONTINUE

IXA = NLYABS(111)
IXB = NLYORD(111)
DO 19 I=2,NPTS
IXB = NLYABS(I11)
IXB = NLYORD(I11)
CALL LINREV(1A,1Y,1Y,1Y)

19 CONTINUE

IX = IXB

15 CONTINUE

RETURN

END
SUBROUTINE HEATER(0,DTM,KODE,TSEN,TOL,TOFF)

1. C

2. IF(TSEN .LT. TDM) GOTO 200

3. IF(TSEN .GT. TOFF) GOTO 100

4. 0 = 0 + KODE

5. RETURN

6. 100 KODE = 0

7. RETURN

8. 200 KODE = 1

9. 0 = 0 + DMT

10. RETURN

11. END
SUBROUTINE NSTAV(PR,VP,U,TINC)

DIMENSION PR(I), VP(I), W(I)

COMMON /SPACE/ N2, N2, N2, N2
COMMON /FECN/ CON(I)
COMMON /TITLE/ HEADEN(I)
COMMON /PERS/ N, N, N

EQUIVALENCE (NPR,PRN), (NVP,VNP), (NW,W)

DATA I10/10/
DATA I11/01/
DATA LT2/13/

COMMON IDIMENSIONA, NPO, NSL

COVALENCE (NPR,PRN), (NVP, VNP), (T, W), (W, W)

If IF(I).GT.0) GO TO 70

IF(NPR.LT.1) NPR=1
If NVP.LT.1) NVP=1
IF(W.LT.11.NM.NM) NM=1

IF(I11.EQ.0) CALL NNRHAI(I)
WRITE(LT) HEADER,(LL,I:1,6),NPR,NVP,LL,LL,LL,NW,LL,LL,NSL.

TIME2=0.0
TIME1=CON(I)*CON(1)

CALL HSTIIPTIME1
KK=1
GO TO 50

TIME2=TIME2+DT
IF(CON11.LT.1.000001) GO TO 12
GO TO 19

IF(I11.LT.TINC) GO TO 50
IF(CON11.LT.TIME1) GO TO 50

TIME1=CON(I11)
TIME2=0.0
CALL HSTIP11(I11)
IF(CON11.LT.1.000001) GO TO 50
CALL HSTTP(-CON(I11))
KK=0
END FILE LT
CONTINUE
RETURN

SUBROUTINE HSTIIPTIME)

WRITE(LT) ITIME,

CALL HSTIIPTIME(I)
RETURN

END
SUBROUTINE HICNT (E1, E2, E3, E4, E5, E6, E7, E8, E9)

C ANALYSIS OF COUNTER FLOW HEAT EXCHANGERS

DIMENSION CP(2), FR(2), NCP(2), TIN(2), TOUT(2), MCP(2)

C EQUIVALENCE (NUA,UA), (NCP,CP)

1. SUBR UTINE IlCT eII,l?,:I,I5,S,I6,x?,x8,x9) u

2. CANAYSIS OF COUNTER FLOW NEAT EICHANGERS

3. 0IeFENSION CP(2), FR(2), NCP(2), TINIZ), TOUT(2), MCP2(2)

4. EOUVALENCE (kUA,UA), (kCP,CP)

5. S

6. UA

7. FR(1) = E1

8. MCP(2) = Fr(2) = E2

9. CF(2) = E3

10. CP(2) = E4

11. TIN(1) = E5

12. TIN(2) = E6

13. TOUT(1) = E7

14. TOUT(2) = E8

15. E2 10 = E9,1

16. IF(FR(1).LT.0.01) G0 TO 10

17. CONTINUE

18. IF(NCP(1).LT.1. OR. NCP(1) .ST. 65000) GO TO 3

19. TAVG = 0.5*(TIN(1)+TOUT(1))

20. CALL OLIDEI(TAVG,CP(1))

21. IF(ICP(2).LT.1. OR. NCP(2) .GT. 65000) GO TO 6

22. TAVG = 0.5*(TIN(2)+TOUT(2))

23. CALL OLIDEI(TAVG,CP(2))

24. CONTINUE

25. MCP(1) = FR(1)*CP(1)

26. MCP(2) = FR(2)*CP(2)

27. IF(MCP(1).LE.0.999 .AND. TIN(1) .GT. 0)

28. CALL OLIDEI(FR(1),CP(1),TIN(1),UA)

29. IS = 1

30. IL = 2

31. IF(ICPRAT .LT. S) GO TO 20

32. IS = 2

33. IL = 1

34. WCP(1) = MCP(1)/WCP(1)

35. IF(WCP(1).GT.0.01) GO TO 30

36. EFF = 1.0

37. GO TO 40

38. IF(WCP(1).LT.0.999 .OR. MCP(1) .GT. 1.001) GO TO 40

39. EFF = UA/MCP(1)+UA)

40. GO TO 40

41. IF(E = EXPR-[UA/MCP(1)+UA/MCP(1)])

42. EFF = (1.0*WCP(1))

43. TOUT(1) = TIN(1) + EFF*(TIN(1)-TOUT(1))

44. TOUT(2) = TIN(2) + EFF*(TIN(2)-TOUT(2))

45. X = TOUT(1)

46. X = TOUT(2)

47. RETURN

48. 100 WRITE(6,101) FR(1)

49. IF(WRITING TO FILE THE NEGATIVE FLOW RATE ES.8, IS NOT ALLOWED)

50. READ EXECUTION TERMINATED IN SUBROUTINE HICNT///IT (31411011)

51. CALL WRack

52. CALL EXIT

53. END
SUBROUTINE MICROS

C ANALYSIS OF CASES FROM HEAT EXCHNGTS.

C EQUIVALENCE (HCR), (CP,CP).

C

C CALL TAYG, CIDEGI (TAVG, I4, CP(I))

C FR(I)*CP(I)

C .LT.

C F(I)*CP(I)

C .OR.

C IABSINUA .GT. 0)

C THE NEGATIVE CAM F(I). I, 0.

C END
SUBROUTINE MIEFF (X, Y, Z, A, B, C)

C ANALYSIS OF HEAT EXCHANGERS WITH EFFECTIVENESS GIVEN

DIMENSION (X, Y, Z, A, B, C, T1, T2, T1C, T2C)

EQUIVALENCE (NIEFF, EFF), (NCP, CP)

EFF = 21
FR(1) = 22
FR(2) = 23
CP(1) = 24
CP(2) = 25
T1(1) = 26
T1(2) = 27
T2(1) = 28
T2(2) = 29
DO 10 I = 1, 2
10 IF(FR(I) .LT. 0.0) GO TO 100
CONTINUE

IF(NCP(1) .LT. 1.0 OR NCP(1) .GT. 25000) GO TO 3
CALL DOGCE(TAVG, 15, CP(1))
IF(NCP(2) .LT. 1.0 OR NCP(2) .GT. 25000) GO TO 6
CALL DOGCE(TAVG, 15, CP(2))

6 CONTINUE

IF(XABS(NEFF) .LE. 99999.0 AND XABS(NEFF) .GT. 0.0)

CALL O20E2(FR(1), FR(2), XI, EFF)

IS = 1
IL = 2
IF(NCP(1) .LE. XCP(1)) GO TO 20
WCP(1) = FR(1) + CP(1)
NEW
WCP(2) = FR(2) + CP(2)
NEW
IF(XABS(NEFF) .LE. 99999.0 AND XABS(NEFF) .GT. 0.0)

NEW

IS = 1
IL = 2
IF(NCP(1) .LE. XCP(1)) GO TO 20
WCP(1) = FR(1) + CP(1)
NEW
WCP(2) = FR(2) + CP(2)
NEW

NEW

TOUT(IS) = T1(IS) + EFF*(T1(IS)-T1(IS))
TOUT(IS) = T1(IS) + WCP(IS)*NCP(IS)*(T1(IS)-T2(IS))
TOUT(IS) = T1(IS) + WCP(IS)*NCP(IS)*(T2(IS)-T1(IS))
TOUT(IS) = TOUT(IS) + WCP(IS)*NCP(IS)*(T2(IS)-T1(IS))

100 RETURN

101 FORMAT (13X, 1H4, 10D12, 1H, ' THE NEGATIVE FLOW RATE', 1S, 0.0, ' IS NOT ALLOWED', 1H13, 1H, ' EXECUTION TERMINATED IN SUBROUTINE MIEFF /// IN LINES 131 (1H, 1S))

CALL WRECK
CALL EXIT
END
SUBROUTINE HIPAR (T1, T2, TIN(1), TOUT(1), TOUT(2), WCP(1), WCP(2))
C ANALYSIS OF PARALLEL FLOW HEAT EXCHANGERS
DIMENSION CP(2), FA(21), WCP(21), TIN(2), TOUT(2), WCP(2)

EQUIVALENCE (MUA, UA), (MCP, CP)

C

UA = 11
FA(1) = 82
FA(2) = 83
CP(1) = 84
CP(2) = 85
TIN(1) = 16
TIN(2) = 17
TOUT(1) = 80
TOUT(2) = 90
DO 10 I = 1, 2
IF(FRI(I) .LT. 0.0) GO TO 100
10 CONTINUE
IF(NCP(1) .LT. 1 .OR. NCP(1) .GT. 65000) GO TO 3
TAVG = 0.5*(TIN(1)+TOUT(1))
CALL DIOEG(TAVG, x4, CP(1))
6 CONTINUE
IF(NCP(2) .LT. 1 .OR. NCP(2) .GT. 65000) GO TO 6
TAVG = 0.5*(TIN(2)+TOUT(2))
CALL DIOEG(TAVG, x5, CP(2))

CALL DIOEG(TAVG, x6, CP(2))

20 WCPRAT = WCP(IS)/WCP(IL)
30 EFF = (1.0 - EXP(-UA/WCP(1)))/UA/WCP(1)
50 TOUT(IS) = TIN(IS) - EFF*(TIN(IS) - TOUT(IS))
39 RETURN

100 WRITE(6,101) FA(I)
101 FORMAT(131(I1,14))
THE NEGATIVE FLOW RATES IS NOT ALLOW
RED. EXECUTION TERMINATED IN SUBROUTINE HIPAR
131(I1,14)
SUBROUTINE LINECK (I)

COMMON /FICOM/ N(1)

I = I + 1

IF (N(2)) = 1 .GT. 60 .OR. N(2) .EQ. 0) CALL JPLIN

N(2) = N(2) + 1

RETURN

END
BEGIN PRINTING LABELS
CONVERT TO BCD

PRINT LEADING NUMERALS
13. PRINT DECIMAL POINT
14. CALL RITE2(V,IV,3000,90,1,1,1,1,MLAST)
15. IF (X=1) EXIT
16. NXC=1
17. IF (X .LE. 0) GO TO 1048
18. NXC = NXC + 1
19. GO TO 1049
20. 1048 IF (X .GE. 0) GO TO 1049
21. MT = 485/485
22. GO TO 1047 (NO1.NT)
23. C CALL RITE2(VI,IV,2000,90,1,1,1,1,MLAST)
24. WRITE ZEROS IN FRACTION
25. IF (X=1) EXIT
26. 1047 CONTINUE
27. 1049 CONTINUE
28. C PRINT TRAILING NUMERALS
29. CALL RITE2(V,IV,3000,90,1,1,1,1,MLAST)
30. IF (Y.GT.0) GO TO 1068
31. C PRINT MINUS SIGN
32. CALL RITE2(VMIN,Y,3000,90,1,1,1,1,MLAST)
33. IF (Y.GT.0) GO TO 1068
34. CONTINUE
35. GO TO 1068
36. CONTINUE
37. CONTINUE
38. CONTINUE
39. CONTINUE
40. CONTINUE
41. CONTINUE
42. CONTINUE
43. CONTINUE
44. CONTINUE
45. CONTINUE
46. CONTINUE
47. CONTINUE
48. CONTINUE
49. CONTINUE
50. CONTINUE
51. CALL LWIDTH (IVAY)
52. END
53. RETURN
1. DIMENSION NBURF(27), DATA(3000), ALPHA(15)
2. DIMENSION ISTART(10),STOP(10)
3. DIMENSION ADD(10)
4. DATA ISTART, STOP, ADD(10)
5. DATA ALPHA, INH, INH, INC, INH, INH, INH, INH, INH, INH, INH,
6. INH
7. WRITE(4,1)
8. 1 FORMAT(110X) "OUTPUT FROM COMB Routines"
9. READ(5,120) NTAPE, IUNIT, K, NODE, INC
10. 120 FORMAT(5)
11. IF(NTAPE .EQ. 0) NTAPE = 10000
12. IF (K .EQ. 0) K = 1
13. IF (IUNIT .EQ. 99) K = 1
14. REWIND K
15. NODE = 0
16. IF (NTAPE .LE. 0) NTAPE = 10000
17. NTAPE = IABS(NTAPE)
18. IF(NODE .NE. 0) WRITE(4,1) NODE
19. IF(NODE .NE. 0) WRITE(4,1) NODE
20. 20 CONTINUE
21. 21 FORMAT(110X) "COMB Routines"
22. DO 10 L = 1, NTAPE
23. M = 0
24. 10 I = L + IUNIT - 1
25. REWIND I
26. READ(1) NBURF
27. IF (L .EQ. 13) GO TO 20
28. WRITE(I) NBURF
29. NTOTAL = 0
30. DO 30 J = 1, NTAPE
31. NTOTAL = NTOTAL + NBURF(J)
32. 30 CONTINUE
33. READ(1) TIME, DATA(K), K = 1, NTOTAL
34. TIME = TIME + ADD(L)
35. IF (TIME .LT. 0.03) GO TO 30
36. IF (TIME .LT. 0.03) GO TO 10
37. IFTIME = KSTART(L)
38. IFSTOP(L) = 99
39. IFTIME = KSTOP(L)
40. TIME = TIME
41. IF(NTAPE .LE. 0) NTAPE = 10000
42. IFM = 1
43. READ(1) TIME, DATA(L), L = 1, NTOTAL
44. M = INC
45. WRITE(I) TIME
46. 10 WRITE(I) TIME, DATA(L), L = 1, NTOTAL
47. IF TIME .LE. 0.03 GO TO 10
48. 0 READ(1) TIME, DATA(L), L = 1, NTOTAL
49. TIME = TIME + ADD(L)
50. IFTIME = KSTART(L)
51. IFSTOP(L) = 99
52. IFTIME = KSTOP(L)
53. WRITE(6,24)
54. 24 FORMAT(110X) "TIME, DATA(L), L = 1, NTOTAL"
55. IF STOP(L) .GT. 9 GO TO 10
56. CALL EXIT
57. END FILE K
58. REWIND K
59. 15 REWIND 1
60. TIME = TIME
61. WRITE(6,20) L, TIME
62. WRITE(6,20) L, TIME
II.20 Format

13 12 QTAPE

13 HIGH ENDING

F10.5.

Z91: HAS BEEN LOADED

61.

11 NEW TAPE.

62 |8 CONT|

63. |FRE|

61.

64. |FRE|

63. |FRE|

61.

62 |8 CONT|

63. |FRE|

61.

62 |8 CONT|

63. |FRE|
3. SUBROUTINE MFSO(A, M, N)
4. DOUBLE PRECISION DMPV, OSM!
5. C INITIALIZE DIAGONAL-LOOP
6. RP = 0
7. DO 11 I=1,N
8. RP = RP + I
9. END = RP - 1
10. C START FACTORIZATION-LOOP OVER K-TH ROW
11. DO 11 K=1,N
12. OSM = 0.00
13. IF(LEND) 2,4,2
14. C START INNER LOOP
15. 2 DO 3 L=1,LEND
16. LANF = RP - L
17. IEND = IND - L
18. OSM = OSM + A(LANF)*A(LIND)
19. C TRANSFORM ELEMENT AMIND)
20. OSM = AMIND - OSM
21. IF(=K) 10, 5, 10
22. C TEST FOR NEGATIVE PIVOT ELEMENT AND FOR LOSS OF SIGNIFICANCE
23. IF(DSM) 12, 12, 9
24. C COMPUTE PIVOT ELEMENT
25. RP = OSM/(OSM)
26. AKP = PK/SM
27. RP = 1.0/SM
28. GO TO 11
29. C CALCULATE TERMS IN ROW
30. 10 AMIND = OSM*P
31. 11 IND = IND + 1
32. RETURN
33. 12 RETURN 3
34. END
1. **FUNCTION NBLANK (WORD, N)**
2. **INTEGER WORD, M, NBLANK**
3. **DATA NBLANK = 6**
4. **M = N + 1**
5. **D O 20 M = 1, N**
6. **I = M - N**
7. **IF (WORD(I) - NBLANK) = 0, 20, 40**
8. **20 CONTINUE**
9. **40 NBLANK = 6 + I**
10. **RETURN**
11. **END**
SUBROUTINE NEWTAP(PR, VP, W, TMPTIM

INTEGER HEADER(12), PR(I), VP(I), M(I)

COMMON /FILE/ CON(I)

COMMON /ITEM/ IT(I)

COMMON /DIMENSION/ NDI, NNA, NTL

DATA IUT, NPE, NWTMP

READ(IUT) HEADER, (NP, I=1,6), NPR, NVP, NP, NP, NP, NW, NP, NNL

IF(HEADER .NE. NPR) GO TO 10

IF(HEADER .NE. NVP) GO TO 10

IF(HEADER .NE. NW) GO TO 10

IF(HEADER .NE. NNL) GO TO 20

CALL TOPLIN

MWRITE(6,15,AEADER, PR(I), M(I), NP, NVP, NP, NP, NP, NP, NW, NP, NNL)

15 FORMAT(2X,12A6)

10 CALL TOPLIN

WRITE(IUT, HEADER, (PR(I), I=1,NPR), (VP(I), I=1,NVP),
(M(I), I=1,NM)

IF(ITIME .LT. 0.0) GO TO 30

IF(ITIME .LT. TMPTIM) GO TO 20

60 GO TO 50

30 ITIME = -ITIME

WRITE(IUT,40)

90 FORMAT(62HINITIAL TEMPERATURES AND VALVE POSITIONS INPUT FROM U-T TAPE AT 012.5 )

RETURN

END
SUBROUTINE NONLIN

INCLUDE COMM,LISI

I. NNC = NNA+NND

3. DO 50 KEE = 1,NMC

50 RKEE = 0.0

6. KEEP1 = NOIM

7. KEEP2 = NTH

8. NOIM = NOIM - NMC

9. NTH = NTH + NMC

10. CALL VARP1

11. NOIM = KEEP1

12. NTH = KEEP2

13. JJ = 0

14. J2 = 1

15. IF(NND.EQ.0) GO TO 86

16. DO 50 J = 1,NND

17. INCLUDE VARP,LIST

18. INCLUDE VARQ,LIST

19. TO JJ = JJ+1

20. LG = FLO(5,16,NDDO(JJ))

21. IF(LG.EQ.0) GO TO 85

22. LTA = FLO(22,14,NDDO(JJ))

23. INCLUDE VARQ,LIST

24. C CHECK FOR LAST CONDUCTOR

25. IF(NDDO(JJ).GT.0) GO TO 10

26. CONTINUE

27. 85 CONTINUE

28. 86 CONTINUE

29. IF(NNA.EQ.0) GO TO 166

30. JJ1 = JJ+1

31. JJ2 = JJ

32. DO 165 L = 1,NNC

33. INCLUDE VAQ2,LIST

34. 135 JJ1 = JJ+1

35. LG = FLO(5,16,NDO(JJ1))

36. LTA = FLO(22,14,NDO(JJ1))

37. INCLUDE VAQ2,LIST

38. C CHECK FOR LAST CONDUCTOR

39. IF(NDO(JJ1).GT.0) GO TO 135

40. 165 CONTINUE

41. 166 CONTINUE

42. RETURN

43. END
SUBROUTINE INTWRK()
IF (MVT.COD) GO TO 515
CALL LNECK(1) *NEW NTWRK
WRITE(6,506) NPASS,NDATA(L1+1) *NEW NTWRK
506 FORMAT(23H ** CONTINUING PASS 15, 13H FOR NETWORK A6, *NEW NTWRK
1 TH ** ) *NEW NTWRK
GO TO 515
510 CALL FRLS(L3,NMT) *NEW NTWRK
C APPLY USER ADDED RESISTANCE TO FLOW CONDUCTOR
C
517 IF(L3>3) RDATA(L4+NMT) = 1.0/RDATA(L4+NMT) *NEW NTWRK
515 IF(MVT. FIRST) GO TO 520
513 CALL FRLS(L19,NMT,NDATA(L1+1)) *NEW NTWRK
511 CALL FRLS(L22,NMT,NDATA(L2+2)) *NEW NTWRK
520 CONTINUE
C CALL FLOAD(NMT,L1,L4,0,0,EFROF,DFMAX) *NEW NTWRK
C IF(DM.T.TOL) 60 TO 530
525 CONTINUE
RETURN
530 FIRST = .FALSE.*NEW NTWRK
EFROF = FRDF *NEW NTWRK
900 CONTINUE
CALL TOPLIN
WRITE(6,560) RDATA(L4+NMT), NPASS, DFMAX, TOL
560 FORMAT(BSHO5 ** SUBROUTINE NTURK FAILED TO CONVERGE TO A SOLU** NTWRK
10 FOR PRESSURES FOR NETWORK A6, TH ** ) *NEW NTWRK
101 2X I0NMAXIMUM PASSES - 110 / *NEW NTWRK
102 2X I0NMAXIMUM CHANGE - G13.8 / *NEW NTWRK
103 4X I0NMAXIMUM ALLOWABLE - G13.8 ) *NEW NTWRK
104 C CALL EXIT *NEW NTWRK
105 C CALL WWK *NEW NTWRK
106 CALL JUTIC *NEW NTWRK
107 C CALL FLP(RNDATA(L4),ISFLOW CONDUCTORS) NTWRK
108 C CALL EXIT *NEW NTWRK
109 C END
SUBROUTINE NKURU(14,WIN,MP,ND)
LOGICAL LNP, LIFR, LAR, LOP, COP, FIRST
DIMENSION RDATA(1)
COMMON ARRAY /NDATA(1)/
COMMON /DATA/L2, L3, L4, L5, L6, L7, L8, L9
COMMON /DATA/LLP, LIFR, LAR, LOP
COMMON /DATA/COP, H3, H4, H5, H6, H7, H8, H9
COMMON /DATA/TOL, MPASS, EPS, FRDF
COMMON FFTSPACE/ND1M, NTH, NEXT(3)
EQUIVALENCE (RDATA,NDATA)

1. \text{DO } 940 MPASS, MPASS
2. DAMI = 0.0
3. \text{DO } 970 1=1,120,5
4. \text{IF (NOT. COP) GO TO 470}
5. \text{IF (NOT. FIRST) CALL TOPLIN}
6. CALL LINECK(4)
7. WRITE(6,440) MPASS, NDATA(L14+1)
8. \text{FORMAT(12H * * PASS IS, 13H FOR NETWORK A0, PM * * *)}
9. \text{PASS LOOP}
10. \text{WRITE(16,1480)}
11. \text{RSO FORATA}
12. \text{IF (FIRST) GO TO 475}
13. \text{NEXT(L254,NFRM)
14. CALL LINECK(3)
15. WRITE(6,440) NFRM, NDAT(A,L254+1)
16. \text{FORMAT(12H * * PASS IS, 13H FOR NETWORK A0, PM * * *)}
17. \text{CALL TOPLIN(4)}
18. \text{WRITE(6,440) MPASS, NDATA(L14+1)
19. \text{PASS LOOP}
20. \text{DO } 940 MPASS, MPASS
21. DAMI = 0.0
22. \text{DO } 970 1=1,120,5
23. \text{IF (NOT. COP) GO TO 470}
24. \text{IF (NOT. FIRST) CALL TOPLIN}
25. CALL LINECK(4)
26. WRITE(6,440) MPASS, NDATA(L14+1)
27. \text{FORMAT(12H * * PASS IS, 13H FOR NETWORK A0, PM * * *)}
28. \text{PASS LOOP}
29. \text{WRITE(16,1480)}
30. \text{RSO FORATA}
31. \text{IF (FIRST) GO TO 475}
32. \text{NEXT(L254,NFRM)
33. CALL LINECK(3)
34. WRITE(6,440) NFRM, NDAT(A,L254+1)
35. \text{FORMAT(12H * * PASS IS, 13H FOR NETWORK A0, PM * * *)}
36. \text{CALL TOPLIN(4)}
37. \text{WRITE(6,440) MPASS, NDATA(L14+1)
38. \text{FORMAT(12H * * PASS IS, 13H FOR NETWORK A0, PM * * *)}
39. \text{PASS LOOP}
40. \text{DO } 940 MPASS, MPASS
41. DAMI = 0.0
42. \text{DO } 970 1=1,120,5
43. \text{IF (NOT. COP) GO TO 470}
44. \text{IF (NOT. FIRST) CALL TOPLIN}
45. CALL LINECK(4)
46. WRITE(6,440) MPASS, NDATA(L14+1)
47. \text{FORMAT(12H * * PASS IS, 13H FOR NETWORK A0, PM * * *)}
48. \text{CALL TOPLIN(4)}
49. \text{WRITE(6,440) MPASS, NDATA(L14+1)
50. \text{FORMAT(12H * * PASS IS, 13H FOR NETWORK A0, PM * * *)}
51. \text{CALL TOPLIN(4)}
52. \text{WRITE(6,440) MPASS, NDATA(L14+1)
53. \text{FORMAT(12H * * PASS IS, 13H FOR NETWORK A0, PM * * *)}
54. \text{CALL TOPLIN(4)}
55. \text{WRITE(6,440) MPASS, NDATA(L14+1)
56. \text{FORMAT(12H * * PASS IS, 13H FOR NETWORK A0, PM * * *)}
57. \text{CALL TOPLIN(4)}
58. \text{WRITE(6,440) MPASS, NDATA(L14+1)
59. \text{FORMAT(12H * * PASS IS, 13H FOR NETWORK A0, PM * * *)}
60. \text{CALL TOPLIN(4)}}
IF L.NEQ. COP) GO TO 515
CALL (NEQER3)
WRITE(*,550) NPPASS, NDATA(L+1)
506 FORMAT(3E12.0) CONTINUING PASS 15, 13H FOR NETWORK A6,
1 TH + ++ 1
GO TO 515
510 CALL FILESV(3D,MTB)
C APPLY USER ADDED RESISTANCE TO FLOW CONDUCTOR
515 IF (.LT. FIRST) GO TO 516
CALL FILENXT(L3S,APN,NDATA=11)
517 CALL FILENXT(L2S,APN,NDATA=21)
520 CONTINUE
C CALL FROB(PLN1,NP2,MP3,NPD, EPROF,OMX)
525 CONTINUE
530 FIRST = .FALSE.
535 EPROF = EPROF
CALL WOBK
CALL OUTCAL
CALL ElIT
CALL TOPLINE
WRITE(*,560) NDATA(L1N+13, NPPASS, DUNK, TOL
560 FORMAT(5SPF + ++ SUBROUTINE NEWRHI FAILED TO CONVERGE TO A
5204 1 UN FOR PRESSURES FOR NETWORK A6, TH ++ ++ //
5205 2 OE 13HMAXIMUM PASSES = 110 /
5206 3 OE 13HMAXIMUM CHANGE = 013.0 /
5207 4 OE 13HMAXIMUM ALLOWABLE = 013.0 )
5210 CALL WELBER
CALL OUTCAL
CALL FLPRTX(RDATA(L4),15HFLOW CONDUCTORS)
CALL EXIT
END
SUBROUTINE WTKN(LIN,WIN,NPI,NPD)
LOGICAL LVP, LIFR, LAI, LDP, COP, FIRST
DIMENSION RDATA(1)
COMMON ARRAYS(NDATA(1))
COMMON RDATA / L2, L3, L4, L5, L6, L7, L8, L9
COMMON /DATA / LVP, LIFR, LAI, LDP
COMMON /DATA / COP, END, NAD, R0, LPU, NPU, NCP, BCP
COMMON /DATA / TOL, NEPSS, EPS, FROF
COMMON /SPACE NDMP, NTH, NEST(1)
COMMON //NDPI, NTH, FIRST(1)
COMMON //NDPI, NTH, FIRST
EQUIVALENCE (RDATA),RDATA
L20:NDATA(14)+1
L25 = NTH + 1
NEST(L25) = NDIM
NFRM = 0
FIRST = .TRUE.
EPSF = 1.0
NEW
PASS LOOP
D2 = RASS +1,MREPASS
DANT = 0.0
EFF = .NOT. COP; GO TO 470
EFF = .NOT. FIRST) CALL TOPLIN
CALL LINECK(4)
WRITE(6,MOD) RDISS, RDATA(1+1)
440 FORMAT(///IF17 = ** PASS IS, 1SH FOR NETWORK Ai, TH * *=)
TUBE LOOP
D2 = 920 J=1,120,S
K = 114 + J
NTB = NDATA(K)
NFRM = NDATA(NTB+1)
!K = NDATA(K+2)
KOAT = NDATA(K+3)
L30 = NDATA(K+4)
IFK = .TRUE. GO TO 470
IFK = NESt(L25+NFRM)
NF0 = NESt(L25+NF0)
NESt = NESt(L2S+NTO)
NESt = RDATA(NTO)
490 IF(FIRST) GO TO 475
490 IF(FIRST) GO TO 476
490 IF(FIRST) GO TO 476
500 CALL TOPLIN
WRITE(6,MOD) NTB, NFRM, NTB, KOAT, RDATA(1+NTB)
480 FORMAT(///, IF7 = THAI = [10], IF7 = THAI = [10], IF7 = THAI = [10], IF7 = THAI = [10], IF7 = THAI = [10]
500 IF(NODATI = 505,517,510
500 CALL TOPLIN
WRITE(6,MOD) RDATA(NODATI+1)
506 FORMAT(///, NETWORK Ai, SiM MUST NOT CONTAiN A SUBNETWORK =

10. CALL WORK
20. CALL E1/F
30. C
40. GO TO 910
50. CALL FLRES(30,MTB)
60. C APPLY USER ADDED RESISTANCE TO FLOW CONDUCTORS
70. C
80. IF(FLK=3) NDATA(L+NTB) = 1.5*(L1/NDATA(L+NTB)+2*DATA(L+NTB))
90. IF(FLK=1) GO TO 920
100. CALL PRAKNET(S5,MTB,NTA+1)
110. CALL PRAKNET(S5,MTB,NTA+1)
120. 620 CONTINUE
130. C
140. CALL FLRES(L30,T9B,NTB)
150. CALL FLRES(L30,T9B,NTB)
160. 630 CONTINUE
170. C
180. CALL PRAKNET(S5,MTB,NTA+1)
190. CALL PRAKNET(S5,MTB,NTA+1)
200. 640 CONTINUE
210. C
220. CALL GLOBAL(NCM,LK,MTB,MTA,EFROF,DUM1)
230. IF(LK .GE. TOL) GO TO 530
240. DO 525 J=1,20,5
250. K = L3 + J
260. ND = NDATA(K+1)
270. MT = NDATA(K+2)
280. NDATA(K+1) = NDATA(K+1) + NDATA(K+2) - NDATA(K+3)
290. 525 CONTINUE
300. RETURN
310. 530 FIRST = .FALSE.
320. EFROF = PROF
330. 540 CONTINUE
340. C
350. CALL TOPLINE
360. WRITE(6,650) NDATA(L+1), MTB, DUM1, TOL
370. 660 FORMAT(5HMAXIMUM PRESSURES FOR NETWORK A6, JH + + //
380. 33X 19HMAXIMUM PASSES - 110 /
390. 33X 19HMAXIMUM CHANGE - G13.8 /
400. 33X 19HMAXIMUM ALLOWABLE - G13.8 )
410. C
420. CALL WORK
430. CALL DUTCAL
440. CALL FFNT(ADATA(L+1),65FLOW CONDUCTORS)
450. CALL EXIT
460. C
470. END
SUBROUTINE PFCS(MLOC,NAME)

LOGICAL LVF, LiFR, LAR, LDP, LOP, LRQ, LPW, LPASS, LPWP

DIMENSION MLOC(1), MLOC(1), NAME(*)

DIMENSION MDATA(11), E(11)

COMMON/ARRAY/MDATA(1)

COMMON/STEP/F(1)

COMMON/SPACE/RDIM, RTH, NEST(1)

COMMON/CON1/CON(1)

COMMON/IFDATA/L2, L3, L4, L5, L6, L7, L8, L9

COMMON/MDATA/LVP, LIFR, LAR, LDP

COMMON/MDATA/COP, LRQ, MRQ, RO, LPW, RNP, ELM, GCS

COMMON/MDATA/TOL, MPASS, EPS, MPF

COMM/N/PMASTER, INODE

EQUIVALENCE (MDATA, MDATA), (EXT, NEST)

EQUIVALENCE (MCON1, TIPMEN), (MCON2), DTIMEU, (CMCON2, LCON)

EQUIVALENCE (TSET, NSET)

DATA MAXI /85000/

LVR = .FALSE.

LiFR = .FALSE.

LAR = .FALSE.

LDP = .FALSE.

IF(MLOC(1) .EQ. 83) GO TO 20

CALL TOPLIN

WRITE(*,103) MLOC(1), NAME

10 FORMAT(*+** INCORRECT NUMBER OF ELEMENTS INPUT TO PFCS FOR

IFLOW DATA, IC = 15, TH = ** // 80 MPF 9463

GO TO 140

20 L2 = MLOC(2)

IF(MDATA(2) .GT. 0) GO TO 40

CALL TOPLIN

WRITE(*,103) MDATA(2), NAME

30 FORMAT(*+** INCORRECT NUMBER OF ELEMENTS INPUT TO PFCS FOR

IFLOW RATES, IC = 15, TH = ** // 80 MPF 9463

GO TO 140

140 L3 = MLOC(3)

IF(MDATA(3) .GT. 0) GO TO 60

CALL TOPLIN

WRITE(*,50) MDATA(3), NAME

50 FORMAT(*+** INCORRECT NUMBER OF ELEMENTS INPUT TO PFCS FOR

IFPRESSURES, IC = 15, TH = ** // 80 MPF 9463

GO TO 140

140 L4 = MLOC(4)

IF(MDATA(4) .EQ. MDATA(2)) GO TO 80

CALL TOPLIN

WRITE(*,103) MDATA(4), NAME

70 FORMAT(*+** INCORRECT NUMBER OF ELEMENTS INPUT TO PFCS FOR

IFLOW CONDUCTORS, IC = 15, TH = ** // 80 MPF 9463

GO TO 140

140
00 15 = MLC(5)
  IF(IFLT(13, 1, 1) GO TO 100
  IFENDATA(5) .GT. 03, IFV = .TRUE.
  100 15 = MLC(6)
  IF(IFLT(13, 1, 1) GO TO 120
  IFENDATA(6) .EQ. MDATA(31) MFLF = .TRUE.
  120 17 = MLC(7)
  IFM2 > MDATA(2), 6) .EQ. 01 GO TO 160
  CALL TOPLIN
  WRITE(6, 130) MDATA(2), NAME
  IF(NR3 .EQ. 1) GO TO 166
  IF(NLOC() .EQ. 16, 6H ASP = 16, 6H AVP = 16,
  IF(NLOC(2) .GT. 15, 6H ASP = 16, 6H AVP = 16)
  CALL WRKCF
  CALL EXIT
  3
  160 19 = MLC(8)
  IF(IFLT(13, 11) GO TO 162
  IFENDATA(8) .EQ. MDATA(2) LAR = .TRUE.
  162 19 = MLC(9)
  IF(IFLT(13, 11) GO TO 166
  IFENDATA(9) .EQ. MDATA(2) LDP = .TRUE.
  166 IFNLOC(1) .EQ. 0, 16, 0 IFNLOC(1) GO TO 100
  CALL TOPLIN
  WRITE(6, 130) MLOC(11), NAME
  170 IFNLOC(11) = 16, 6H ASP = 16, 6H AVP = 16,
  IF(NLOC(11) .GT. 15, 6H ASP = 16, 6H AVP = 16)
  CALL WRKCF
  CALL EXIT
  91
  100 112 = MLOC(1)
  IFNLOC(1) .EQ. 5, 16, 9 IFNLOC(1) GO TO 200
  CALL TOPLIN
  200 113 = MLOC(3)
  IFNLOC(3) .EQ. 9, 16, 9 IFNLOC(1) GO TO 240
  CALL TOPLIN
  WRITE(6, 210) MDATA(11), NAME
  190 IFNLOC(11) = 16, 6H ASP = 16, 6H AVP = 16,
  IF(NLOC(11) .GT. 15, 6H ASP = 16, 6H AVP = 16)
  GO TO 240
  240 114 = MLOC(1)
  IFNLOC(1) .EQ. 9, 16, 9 IFNLOC(1) GO TO 280
  CALL TOPLIN
  WRITE(6, 210) MDATA(11), NAME
  210 IFNLOC(11) = 16, 6H ASP = 16, 6H AVP = 16,
  IF(NLOC(11) .GT. 15, 6H ASP = 16, 6H AVP = 16)
  GO TO 280
  280 115 = MLOC(1)
  IFNLOC(1) .EQ. 9, 16, 9 IFNLOC(1) GO TO 320
  CALL TOPLIN
  WRITE(6, 210) MDATA(11), NAME
  310 IFNLOC(11) = 16, 6H ASP = 16, 6H AVP = 16,
  IF(NLOC(11) .GT. 15, 6H ASP = 16, 6H AVP = 16)
  GO TO 320
  CALL WRKCF
  CALL EXIT
  110
  240 119 = MLOC(1)
  IFNLOC(1) .EQ. 9, 16, 9 IFNLOC(1) GO TO 280
  CALL TOPLIN
  WRITE(6, 210) MDATA(11), NAME
  210 IFNLOC(11) = 16, 6H ASP = 16, 6H AVP = 16,
  IF(NLOC(11) .GT. 15, 6H ASP = 16, 6H AVP = 16)
  GO TO 280
  280 115 = MLOC(1)
  IFNLOC(1) .EQ. 9, 16, 9 IFNLOC(1) GO TO 320
  CALL TOPLIN
  WRITE(6, 210) MDATA(11), NAME
  310 IFNLOC(11) = 16, 6H ASP = 16, 6H AVP = 16,
  IF(NLOC(11) .GT. 15, 6H ASP = 16, 6H AVP = 16)
  GO TO 320
  CALL WRKCF
  CALL EXIT
  110
  CALL WRKCF
  CALL EXIT
  110
  CALL WRKCF
  CALL EXIT
  110
GO TO 310

310 LRD = .FALSE.
RJ = RDATA(L12+2)
RMD = RDATA(L12+3)
IF(RMU .LT. 1 .OR. RMD .GT. MAX1) GO TO 330

181

330 ERL = .FALSE.
TPU = RDATA(L12+3)
GO TO 340

340 BZ = RDATA(L)+5*2.0

C SOLUTION PARAMETERS

TOL = RDATA(L)+1
MNSA = RDATA(L)+2
EPS = RDATA(L)+3
PROM = RDATA(L)+4

C VALVES

IF(IS .LT. 1) GO TO 510
IF(NDATA(LIS) .LT. 1) GO TO 510
IF(LMP1) GO TO 340

343 WRITE(3,344) NDATA(LS), NAME

344 FORMAT(1X, * INCORRECT NUMBER OF ELEMENTS INPUT TO PFCS FOR

IVALE POSITIONS, IC = 15, TH = *, */ 8x 4MP3R 9469

CALL MRACK
CALL EXIT

C

340 L=0 = RDATA(L15)
G3 350 J=1,440
M = RDATA(L)+5
NUK = RDATA(N1)+1
HDE = RDATA(N1)+9
IF(MOD .EQ. 0) GO TO 250
IF(M = .GT. NDATA(L13)) GO TO 343

150

IF(M = .LT. NDATA(L13)) GO TO 380

154

155

156

157

158

159

160

161

162

163

164

165

166

167

168

169

170

171

172

173

174

175

176

177

178

179

180

181

182

183

184

185

C

G3 370 T=J,100
G3 380 T=J,100
TSEN = RDATA(T12+3) TSET = RDATA(T12+2)

J=1,0 TO 500
J=1,0 TO 500

IF(ABS(TSEN-TSET) = TOB) GO0

C

IC = NDATA(L13)
IF(NDATA(LIS) = .LT. 1) GO0

IC = NDATA(L13)
IF(TH .LT. 10000) TSEN = TSEN
IF(TH .LT. 10000) TSET = TSET

C

IF(TH .LT. 10000) TSEN = TSEN
IF(TH .LT. 10000) TSET = TSET

C

C

IC = NDATA(L13)
IF(TH .LT. 10000) TSEN = TSEN
IF(TH .LT. 10000) TSET = TSET

C

C

C

CALL TOPLIN

WRITE(3,350) IC, NAME

350 FORMAT(1X, * INCORRECT NUMBER OF ELEMENTS INPUT TO PFCS FOR

VALVE DATA, IC = 15, TH = *, */ 8x 4MP3R 9461

C

C

C

C

C

C

360 T = RDATA(91+1)

361 IF(IS .LT. -TSET) - TDATA(1) RDATA(1) + 300

362 IF(IS .LT. TDATA(1) + J) GO TO 400

363 IF(TDATA(1) + J) GO TO 400

364 IF(TDATA(1) + J) GO TO 400

365 GO TO 400

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C
450 301 = AMG15(RDATA(L1+4)+1) + TSET + 200, RDATA(L1+12)
107. RDATA(L8+NY) = AMG15(RDATA(L8+NY) + 20) + 916, RDATA(L8+NY)
188. GO TO 400
189. C POLYNOMIAL
190. C
191. 420 Qt = TSET - TSET
192. 95S = RDATA(L1+10) + QT(RDATA(L1+11) + D1(RDATA(L1+12) +
193. 1 + QT + RDATA(L1+13) + D1(RDATA(L1+14) +
194. 2 + QT + RDATA(L1+15) + D1(RDATA(L1+16) +
195. IF(95S < TMIN) 420, 440, 430
196. 925 95S = RDATA(L1+5)
197. GO TO 440
198. 430 95S = TMIN + 440, 440, 435
199. 60 RDATA(L9+NT) = TMIN
200. GO TO 450
201. 450 RDATA(L8+NY) = 95S + (RDATA(L8+NY)+95S*EXP-DT+RDATA(L1+16))
202. GO TO 450
203. C
204. C SWITCHING
205. C
206. 450 IF((2.0*TSET - RDATA(L9+NT)) > 0) GO TO 500
207. 460 RDATA(L8+NY) = TMIN
208. GO TO 450
209. 480 RDATA(L8+NY) = TMAX
210. C
211. 500 CONTINUE
212. C
213. C CHECK PUMP OPTION
214. C
215. 510 IF(L16 < .LT. 8) GO TO 500
216. IF(RDATA(L16) - 21 540, 520, 560
217. 420 NPE = RDATA(L16+1)
218. NPUMP = RDATA(L16+2)
219. CALL BIDGE(TPM1, RDATA(NPUMP), RDATA(L16+NP))
220. 540 LPUMP = .FALSE.
221. GO TO 600
222. 560 LPUMP = .TRUE.
223. NPE = RDATA(L16+1)
224. NPUMP = RDATA(L16+2)
225. IF(RDATA(L16) > 3) GO TO 500 +12
226. NPE = RDATA(L16+1)
227. LPUMP = .FALSE.
228. GO TO 600
229. IF(RDATA(L16) > 3) GO TO 500
230. NPE = RDATA(L16+1)
231. DPE = RDATA(NPUMP+2)
232. GO TO 590
233. 980 KPUMP = 2
234. AD = RDATA(L16+3)
235. A1 = RDATA(L16+4)
236. A2 = RDATA(L16+5)
237. A3 = RDATA(L16+6)
238. A4 = RDATA(L16+7)
239. C
240. C SYSTEM SOLUTION
241. C
242. 590 NPASS = .FALSE.
243. 600 GO 965 KPASS = 1, 20
244. C
245. IF(NPASS = .NOT. 0) GO TO 960
246. CALL EOLHP
247. WRITE(*,620) NPASS, NPE
248. 620 FORMAT(2010.0 = CHECKOUT PRINT FOR PRESSURE.FLOW COMPUTATION SIM
249. TRIBUTE = ** * // NPASS = 13, 98 FOR 960)
Il9.
PLC
LC *
"PCS
250.
"IPLASS) GO TO 490
WRITE6,490 TOL, &PASS, EPS, PROF
430 FORMATTED IOR E SINOL = 610.5, 7M &PASS = 15, 6H EPS = 610.5,
431 & TH PROF = 610.5
434. LC = LC + 2
435. C
436. C640 CALL &NLRI81(1)
437. C
438. C &NLRT, &LPUMP) GO TO 1900
439. C
440. C PUMP
441. C
442. WS = &OKTALLX-NS1)
443. TEST = 0.002+WS
444. EPS = &OKTAllX-NS1 - &OKTAlLX-NS3)
445. WE = WS
446. EPS = EPS
447. G3 TO 600,8001, &PUMP
448. C
449. C TABULATED PUMP CURVE
450. C
451. 660 IF(LPASS) GO TO 665
452. 660 E = EPS/WE
453. 660 = 0.0
454. 660 TO 670
455. 665 E = EPS-DPL1(MS-WS)
456. 665 = DPL - WS+C
457. 670 WP = &OKTALLX-PUMP)
458. 670 IF. &NLRT, &CUP ) GO TO 690
459. CALL LINECK(2)
460. WRITE6,680)
461. 680 FORMAT(IHD
7 39HCHECKOUT PRINT FOR TABULATED PUMP CURVE 1
462. 690 GO TO 741,100
463. 741 E = AMN(MS,WE)
464. 741 EPS = AMN(EPS,OPP)
465. CALL DIEMAIN(WA, &OKTALLX-PUMP),OPP)
466. CALL REVPOL(EPS, &OKTALLX-PUMP),OPP)
467. CALL &NLRT. CUP ) GO TO 710
468. CALL LINECK(3)
469. WRITE6,690)
470. 690 IF(LPASS) GO TO 695
471. 695 705 = EPS/WE
472. 705 WS = EPS/WS
473. 705 WP = EPS/WS
474. 710 EPS = EPS-DEP1(MS-WS)
475. EPS = EPS
476. EPS = EPS
477. EPS = EPS
478. EPS = EPS
479. EPS = EPS
480. EPS = EPS
481. EPS = EPS
482. EPS = EPS
483. EPS = EPS
484. EPS = EPS
485. EPS = EPS
486. EPS = EPS
487. EPS = EPS
488. EPS = EPS
489. EPS = EPS
490. EPS = EPS
491. EPS = EPS
492. EPS = EPS
493. EPS = EPS
494. EPS = EPS
495. EPS = EPS
496. EPS = EPS
497. EPS = EPS
498. EPS = EPS
499. EPS = EPS
500. EPS = EPS
501. EPS = EPS
502. EPS = EPS
503. EPS = EPS
504. EPS = EPS
505. EPS = EPS
506. EPS = EPS
507. EPS = EPS
508. EPS = EPS
509. EPS = EPS
510. EPS = EPS
511. EPS = EPS
ADD = AD - CPS/W

ALL = A1 - CPS/W

WIN = WS

IF(1.3T, 1PASS) GO TO 820

TEMP = (CPS/DPL/WS+WL)

A0 = A0 - DPL + TEMP+4

A1 = A1 - TEMP

820 03 003 =1,100

FNEW = ADD + UNEW(A2 - UNEW(A3 + UNEW(A3))

IF(ABS(FNEW) = CHECK) 140,140,040

840 FP = ALL + UNEW(2.0*A2 + UNEW(3.0*A3 + UNEW(4.0*A4))

NEW = WHEN - FNEW/FPP

860 CONTINUE

C

CALL TOPLIN

WRITE(6,160)

WRITE(6,880)

880 FORMAT(I/88HSYSTEM TOTAL PRESSURE DROP

IS SUPPLIED

BY A FOURTH

ORDER POLYNOMIAL FUNCTION OF FLOW RATE)

900 CALL WBACK

CALL EXIT

C

CALL TOPLIN

WRITE(6,980)

980 FORMAT(I/88HSUBROUTINE PFCS FAILED TO CONVERGE TO A SOLUTION

IN TO TRUE SYSTEM CHARACTERISTICS AND TRUE PUMP CURVE

IN TRUE SYSTEM CHARACTERISTICS AND TRUE PUMP CURVE)

1000 RETURN

END
FUNCTION POL(LOC, X)
C COMMON /ARRAY/ MORTAI3
C CALL DIDEIG(X,MORTAI3, V)
POL = V
RETURN
END
1. SUBROUTINE PAN(LOC,N,MODE)
2. C
3. DIMENSION LOC(13)
4. C
5. IF(M .LT. 1) GO TO 20
6. DO 10 J=1,M
7. IF(LOC(J+13 .EQ. MODE) GO TO 30
8. 10 CONTINUE
9. C
10. DO N = N + 1
11. IF(M .GT. LOC(1)) GO TO 40
12. LOC(N+13) = MODE
13. C
14. RETURN
15. C
16. RETURN
17. C
18. 40 NEED = N - LOC(1)
19. CALL TOPLIN
20. WRITE(*,120) NEED
21. 120 FORMAT(3X) NEED = * * INSUFFICIENT DYNAMIC STORAGE AVAILABLE FOR FLOW
22. 130 FORMAT(3X) IMBALANCED SUBROUTINE = * * # AX SHORT IS, 1HM LOCATIONS
23. CALL MERCA
24. CALL EXIT
25. C
26. END
SUBROUTINE PSOR(A,M,A,PMR)

C SUBROUTINE PSOR SOLVES A SYSTEM OF SIMULTANEOUS EQUATIONS USING A
C STATIONARY POINT ITERATIVE SUCCESSIVE OVERRELAXATION METHOD.

AM: DIMENSION A(M,M), AM, XM

MA = M - 1.0

C DO 10 K = 1, MA

DO 10 J = 1, M

CALL GENR(A(I,J),I,M,'COEFFICIENTS OF P(J)')

10 CONTINUE

RETURN

END
SUBROUTINE RADIR(LOC,SIGMA,TIER)

CALCULATION FOR CROSS RADIATION

LOGICAL EXP
COMMON SOURCE, DATA
COMMON SPICE, NOIP, NTH, EST(1)
COMMON ARRAY, CURV0)
COMMON DIMNSR, RNY, MAN
COMMON TEMP, T1(3)
COMMON RFOC, COM(1)

DIMENSION NCURV0

EQUIVALENCE (CON(I), ICON(I)), TPE1)
EQUIVALENCE (CURV0, NCURV0)

DEFINE DTAU(I) = NON, NADIR
ANC = NTH - NADIR

IF (TIME .GT. 0.0) EXPLCT = .FALSE.

IF (RLOC(I) .EQ. 6) GO TO 2

CALL TOPLIN
WRITE(6,1) NLOC(I)
CALL LKBCK
CALL EXIT

2 ISNA = LOC0(2)
ISNE = LOC0(3)
ISEN = LOC0(4)
ISEN = LOC0(4)
NNA = LOC0(6)
ISEC = LOC0(7)
NS = NCURV0(ISNA)
NC = NCURV0(ISNE)
IFNCURV0(ISNE) = 500,
IFNCURV0(ISNE) = 500,
IFNCURV0(ISNE) = 500,
IFNCURV0(ISNE) = 500,

IFNCURV0(ISNE) = 500,
IFNCURV0(ISNE) = 500,
IFNCURV0(ISNE) = 500,
IFNCURV0(ISNE) = 500,

IFNCURV0(ISNE) = 500,
IFNCURV0(ISNE) = 500,
IFNCURV0(ISNE) = 500,
IFNCURV0(ISNE) = 500,

IFNCURV0(ISNE) = 500,
IFNCURV0(ISNE) = 500,
IFNCURV0(ISNE) = 500,
IFNCURV0(ISNE) = 500,

IFNCURV0(ISNE) = 500,
IFNCURV0(ISNE) = 500,
IFNCURV0(ISNE) = 500,
IFNCURV0(ISNE) = 500,

IFNCURV0(ISNE) = 500,
IFNCURV0(ISNE) = 500,
IFNCURV0(ISNE) = 500,
IFNCURV0(ISNE) = 500,

IFNCURV0(ISNE) = 500,
IFNCURV0(ISNE) = 500,
IFNCURV0(ISNE) = 500,
IFNCURV0(ISNE) = 500,

IFNCURV0(ISNE) = 500,
IFNCURV0(ISNE) = 500,
IFNCURV0(ISNE) = 500,
IFNCURV0(ISNE) = 500,

IFNCURV0(ISNE) = 500,
IFNCURV0(ISNE) = 500,
IFNCURV0(ISNE) = 500,
IFNCURV0(ISNE) = 500,

IFNCURV0(ISNE) = 500,
IFNCURV0(ISNE) = 500,
IFNCURV0(ISNE) = 500,
IFNCURV0(ISNE) = 500,

IFNCURV0(ISNE) = 500,
IFNCURV0(ISNE) = 500,
IFNCURV0(ISNE) = 500,
IFNCURV0(ISNE) = 500,

IFNCURV0(ISNE) = 500,
IFNCURV0(ISNE) = 500,
IFNCURV0(ISNE) = 500,
IFNCURV0(ISNE) = 500,
```
CURVOLOCX = .9999

20 CONTINUE
21 WRITE(*,MS1) (CURVOLOCX(LREF+1,KK),KK=1,NS)
453 FORMAT(*SURFACE REFLECTIVITY DATA/(1000000.6))
454 PRINT *,NCURVO(LREF+1),G0 TO 52
455 DO 45 I=1,NS
456 LOC = LREF + I
457 IF (CURVOLOCX(LOC ) GT. 0.016D0 TO 3D
458 CURVOLOCX(LOC ) = .9999
459 GO TO 40
460 IF (CURVOLOCX(LOC ) .LT. 1.0D0 GO TO 40
461 CURVOLOCX(LOC ) = .99999
462 CONTINUE
463 IF (CURVOLOCX(LOC ) .NE. .99999D0 TO 3D
464 CURVOLOCX(LOC ) = .99999
465 CONTINUE
466 IBEG = ISON + 1
467 IEND = ISON + NCURVO(ISCON)
468 WRITE(*,MS2) (NCURVO(KK),KK=1,NCON)
469 IF (NCURVO(ISCON) Ne. 502 GO TO 59
470 LOC = NNA
471 WRITE(*,MS0)
472 DO 10 I=1,NS
473 D0 60 I=1,NS
474 LL = ISNA + 3*I
475 IBEG = LOC + I
476 IEND = LOC + NCURVO(LOC)
477 WRITE(*,MS15) NCURVO(ISCON),NCURVO(ISCON),LOC
478 IG = NCURVO(ISCON)/2I GO TO 59
479 IST = LOC + 2
480 LOC = LOC + NCURVO(LOC)
481 AGS = 0.
482 LOC = LOC + NCURVO(LOC)
483 AGS = AGS + CURVOLOCX
484 CONTINUE
485 IF (ABS(AGS-CURVOLOCX))/CURVOLOCX(LL) GT. .01 GO TO 55
486 LOC = LOC + 1
487 DO 60 I=1,NS
488 LL = ISNA + 3*I
489 LOC = LOC + I
490 NCURVO(ISCON) GT. .91 GO TO 56
491 LOC = LOC + I
492 CONTINUE
493 IF (NCURVO(ISCON) Ne. 502 GO TO 59
494 LOC = LOC + 1
495 DO 60 I=1,NS
496 LL = ISNA + 3*I
497 IF (NCURVO(ISCON) GT. .91 GO TO 56
498 LOC = LOC + I
499 CONTINUE
500 IF (NCURVO(ISCON) Ne. 502 GO TO 59
501 LOC = LOC + 1
502 CONTINUE
503 IF (NCURVO(ISCON) GT. .91 GO TO 56
504 LOC = LOC + I
505 CONTINUE
506 LOC = LOC + I
507 DO 60 I=1,NS
508 LL = ISNA + 3*I
509 LOC = LOC + I
510 CONTINUE
511 IF (NCURVO(ISCON) GT. .91 GO TO 56
512 LOC = LOC + I
513 CONTINUE
514 IF (NCURVO(ISCON) GT. .91 GO TO 56
515 LOC = LOC + I
516 CONTINUE
517 IF (NCURVO(ISCON) GT. .91 GO TO 56
518 LOC = LOC + I
519 CONTINUE
520 IF (NCURVO(ISCON) GT. .91 GO TO 56
521 LOC = LOC + I
522 IF (NCURVO(ISCON) GT. .91 GO TO 56
523 LOC = LOC + I
524 IF (NCURVO(ISCON) GT. .91 GO TO 56
525 LOC = LOC + I
526 IF (NCURVO(ISCON) GT. .91 GO TO 56
527 LOC = LOC + I
528 IF (NCURVO(ISCON) GT. .91 GO TO 56
529 LOC = LOC + I
530 IF (NCURVO(ISCON) GT. .91 GO TO 56
531 LOC = LOC + I
532 IF (NCURVO(ISCON) GT. .91 GO TO 56
533 LOC = LOC + I
534 IF (NCURVO(ISCON) GT. .91 GO TO 56
535 LOC = LOC + I
536 IF (NCURVO(ISCON) GT. .91 GO TO 56
537 LOC = LOC + I
538 IF (NCURVO(ISCON) GT. .91 GO TO 56
539 LOC = LOC + I
540 IF (NCURVO(ISCON) GT. .91 GO TO 56
541 LOC = LOC + I
542 IF (NCURVO(ISCON) GT. .91 GO TO 56
543 LOC = LOC + I
544 IF (NCURVO(ISCON) GT. .91 GO TO 56
545 LOC = LOC + I
546 IF (NCURVO(ISCON) GT. .91 GO TO 56
547 LOC = LOC + I
548 IF (NCURVO(ISCON) GT. .91 GO TO 56
549 LOC = LOC + I
550 IF (NCURVO(ISCON) GT. .91 GO TO 56
551 LOC = LOC + I
552 IF (NCURVO(ISCON) GT. .91 GO TO 56
553 LOC = LOC + I
554 IF (NCURVO(ISCON) GT. .91 GO TO 56
555 LOC = LOC + I
556 IF (NCURVO(ISCON) GT. .91 GO TO 56
557 LOC = LOC + I
558 IF (NCURVO(ISCON) GT. .91 GO TO 56
559 LOC = LOC + I
560 IF (NCURVO(ISCON) GT. .91 GO TO 56
561 LOC = LOC + I
562 IF (NCURVO(ISCON) GT. .91 GO TO 56
563 LOC = LOC + I
564 IF (NCURVO(ISCON) GT. .91 GO TO 56
565 LOC = LOC + I
566 IF (NCURVO(ISCON) GT. .91 GO TO 56
567 LOC = LOC + I
568 IF (NCURVO(ISCON) GT. .91 GO TO 56
569 LOC = LOC + I
570 IF (NCURVO(ISCON) GT. .91 GO TO 56
571 LOC = LOC + I
572 IF (NCURVO(ISCON) GT. .91 GO TO 56
573 LOC = LOC + I
574 IF (NCURVO(ISCON) GT. .91 GO TO 56
575 LOC = LOC + I
576 IF (NCURVO(ISCON) GT. .91 GO TO 56
577 LOC = LOC + I
578 IF (NCURVO(ISCON) GT. .91 GO TO 56
579 LOC = LOC + I
580 IF (NCURVO(ISCON) GT. .91 GO TO 56
581 LOC = LOC + I
582 IF (NCURVO(ISCON) GT. .91 GO TO 56
583 LOC = LOC + I
584 IF (NCURVO(ISCON) GT. .91 GO TO 56
585 LOC = LOC + I
586 IF (NCURVO(ISCON) GT. .91 GO TO 56
587 LOC = LOC + I
588 IF (NCURVO(ISCON) GT. .91 GO TO 56
589 LOC = LOC + I
590 IF (NCURVO(ISCON) GT. .91 GO TO 56
591 LOC = LOC + I
592 IF (NCURVO(ISCON) GT. .91 GO TO 56
593 LOC = LOC + I
594 IF (NCURVO(ISCON) GT. .91 GO TO 56
595 LOC = LOC + I
596 IF (NCURVO(ISCON) GT. .91 GO TO 56
597 LOC = LOC + I
598 IF (NCURVO(ISCON) GT. .91 GO TO 56
599 LOC = LOC + I
600 IF (NCURVO(ISCON) GT. .91 GO TO 56
601 LOC = LOC + I
602 IF (NCURVO(ISCON) GT. .91 GO TO 56
603 LOC = LOC + I
604 IF (NCURVO(ISCON) GT. .91 GO TO 56
605 LOC = LOC + I
606 IF (NCURVO(ISCON) GT. .91 GO TO 56
607 LOC = LOC + I
608 IF (NCURVO(ISCON) GT. .91 GO TO 56
609 LOC = LOC + I
610 IF (NCURVO(ISCON) GT. .91 GO TO 56
611 LOC = LOC + I
612 IF (NCURVO(ISCON) GT. .91 GO TO 56
613 LOC = LOC + I
614 IF (NCURVO(ISCON) GT. .91 GO TO 56
615 LOC = LOC + I
616 IF (NCURVO(ISCON) GT. .91 GO TO 56
617 IF (NFR-1*NS = (NFR-NFR-NFR)/2 + NFR
618 NWLO = NFR
619 NFR = NFR
620 NWLO = NWLO
621 LOC2 = (NFR-1+1*NS = (NFR-NFR-NFR)/2 + NFR
622 CURVOLOCX(LOC2) = EA
```
190 CONTINUE
191 LOC = NMA
192 D3 210 I=1,NS
193 LOC = ISMA + I-1
194 AREAS = CURVOLOCI
195 WN = NCURVOLOCI
196 LOC = ISMA + I-1
197 FA = CURVOLOCIAUX
198 FAT = CURVOLOC2
199 D3 359 J=1,NN
200 LOC = LOC + 1
201 NDE = NCURVOLOC
202 LOC = LOC + 1
203 ARAT = CURVOLOCIAREA
204 TP = T(NODE) - TIERD
205 SIGMAT = SIGMA TaTP*TP*TP
206 SIGMAT = SIGMAT + 9.0*SIGMAT*ARAT
207 Q(NODE) = Q(NODE) + ARAT*(FAT*TP*SIGMAT)
208 CONTINUE
209 LOC = LOC + 1
210 CONTINUE
211 RETURN
212 END

206 FORMAT(1316+9=I-3)
207 T(ISN) = (T(ISN)**.25 + TIERD
208 300 WRITE(6,600)
209 400 FORMAT(1316+9=I-3/THE SURFACE EMISSIVITY DATA DOES NOT HAVE
210 THE CORRECT NUMBER OF VALUES//IX 1316=I)
211 GO TO 900
212 501 WRITE(6,601)
213 601 FORMAT(1316+9=I-3/THE SURFACE DATA DOES NOT HAVE THE CORRECT-
214 2 NUMBER OF VALUES//IX 1316
215 GO TO 900
216 RETURN
217 END

218 502 WRITE(6,602)
219 602 FORMAT(1316+9=I-3/THE SURFACE REFLECTIVITY DATA DOES NOT HAVE
220 THE CORRECT NUMBER OF VALUES//IX 1316=I)
221 GO TO 900
222 503 WRITE(6,603)
223 603 FORMAT(1316+9=I-3/THE SUM OF THE AREAS FOR THE NODES 3M SURF
224 FACE 'IS,' IS NOT EQUAL TO THE AREA OF THAT SURFACE//IX 1316=I)
225 GO TO 900
226 504 WRITE(6,604)
227 604 FORMAT(1316+9=I-3/THE EXTRAS SPACE ARAT DOES NOT HAVE THE COR-
228 TRECT NUMBER OF SPACES//IX 1316=I)
229 GO TO 900
230 505 WRITE(6,605)
231 605 FORMAT(1316+9=I-3/THE SURFACE NUMBER '15,' IS NOT SUPPLIED IN THE SURFACE DATA//IX 1316=I)
232 GO TO 900
233 506 WRITE(6,606)
234 606 FORMAT(1316+9=I-3/THE SPACE ARAT WAS NOT SUPPLIED IN THE SURFACE DATA//IX 1316=I)
235 GO TO 900
236 507 WRITE(6,607)
237 607 FORMAT(1316+9=I-3/SURFACE NUMBER '15,' IN THE SURFACE CONNECT
238 NODE DATA WAS NOT SUPPLIED IN THE SURFACE DATA//IX 1316=I)
239 GO TO 900
240 508 WRITE(6,608)
241 608 FORMAT(1316+9=I-3/THE EXECUTION TERMINATED BY A PEARED MALT.)
242 CALL WWRK
243 CALL EXIT
244 END
1. SUMRUTIEE RADSOLOC

2. CALCULATION FOR SOLAR CROSS RADIATION

3. DIMENSION NLOC(I)

4. COMMON /CURVOL/ TIME

5. COMMON /CURVOL/ CURVOL

6. COMMON /CURVOL/ CURVOL

7. DIMENSION NCUVOL1, FAM(12)

8. EQUIVALENCE (CURVOL, NCUVOL)

9. IF(NLOC(I)) .EQ. 71 GO TO 2

10. CALL TOPLIN

11. WRITE(6,1152)

12. IF(A10) PRINT * 1 INCORRECT NUMBER OF ELEMENTS INPUT TO RADSOL

13. E = 1.0, TH = 93

14. CALL MEND

15. CALL EXIT

16. IF(ISL) PRINT * 3 "DIMENSION NLOC(I)

17. COMMON /CURVOL/ CURVOL

18. COMMON /CURVOL/ CURVOL

19. IF(ISL) PRINT * 5 "DIMENSION NCUVOL1, FAM(12)

20. EQUIVALENCE (CURVOL, NCUVOL)

21. IF(NLOC(I)) .EQ. 71 GO TO 2

22. CALL TOPLIN

23. WRITE(6,1152)

24. IF(A10) PRINT * 1 INCORRECT NUMBER OF ELEMENTS INPUT TO RADSOL

25. E = 1.0, TH = 93

26. CALL MEND

27. CALL EXIT

28. IF(ISL) PRINT * 3 "DIMENSION NLOC(I)

29. COMMON /CURVOL/ CURVOL

30. COMMON /CURVOL/ CURVOL

31. IF(ISL) PRINT * 5 "DIMENSION NCUVOL1, FAM(12)

32. EQUIVALENCE (CURVOL, NCUVOL)

33. IF(NLOC(I)) .EQ. 71 GO TO 2

34. CALL TOPLIN

35. WRITE(6,1152)

36. IF(A10) PRINT * 1 INCORRECT NUMBER OF ELEMENTS INPUT TO RADSOL

37. E = 1.0, TH = 93

38. CALL MEND

39. CALL EXIT

40. IF(ISL) PRINT * 3 "DIMENSION NLOC(I)

41. COMMON /CURVOL/ CURVOL

42. COMMON /CURVOL/ CURVOL

43. IF(ISL) PRINT * 5 "DIMENSION NCUVOL1, FAM(12)

44. EQUIVALENCE (CURVOL, NCUVOL)

45. IF(NLOC(I)) .EQ. 71 GO TO 2

46. CALL TOPLIN

47. WRITE(6,1152)

48. IF(A10) PRINT * 1 INCORRECT NUMBER OF ELEMENTS INPUT TO RADSOL

49. E = 1.0, TH = 93

50. CALL MEND

51. CALL EXIT

52. IF(ISL) PRINT * 3 "DIMENSION NLOC(I)

53. COMMON /CURVOL/ CURVOL

54. COMMON /CURVOL/ CURVOL

55. IF(ISL) PRINT * 5 "DIMENSION NCUVOL1, FAM(12)

56. EQUIVALENCE (CURVOL, NCUVOL)

57. IF(NLOC(I)) .EQ. 71 GO TO 2

58. CALL TOPLIN

59. WRITE(6,1152)

60. IF(A10) PRINT * 1 INCORRECT NUMBER OF ELEMENTS INPUT TO RADSOL

61. E = 1.0, TH = 93

62. CALL MEND

63. CALL EXIT

64. IF(ISL) PRINT * 3 "DIMENSION NLOC(I)

65. COMMON /CURVOL/ CURVOL

66. COMMON /CURVOL/ CURVOL

67. IF(ISL) PRINT * 5 "DIMENSION NCUVOL1, FAM(12)

68. EQUIVALENCE (CURVOL, NCUVOL)

69. IF(NLOC(I)) .EQ. 71 GO TO 2

70. CALL TOPLIN

71. WRITE(6,1152)

72. IF(A10) PRINT * 1 INCORRECT NUMBER OF ELEMENTS INPUT TO RADSOL

73. E = 1.0, TH = 93

74. CALL MEND

75. CALL EXIT

76. IF(ISL) PRINT * 3 "DIMENSION NLOC(I)

77. COMMON /CURVOL/ CURVOL

78. COMMON /CURVOL/ CURVOL

79. IF(ISL) PRINT * 5 "DIMENSION NCUVOL1, FAM(12)

80. EQUIVALENCE (CURVOL, NCUVOL)

81. IF(NLOC(I)) .EQ. 71 GO TO 2

82. CALL TOPLIN

83. WRITE(6,1152)

84. IF(A10) PRINT * 1 INCORRECT NUMBER OF ELEMENTS INPUT TO RADSOL

85. E = 1.0, TH = 93

86. CALL MEND

87. CALL EXIT

88. IF(ISL) PRINT * 3 "DIMENSION NLOC(I)

89. COMMON /CURVOL/ CURVOL

90. COMMON /CURVOL/ CURVOL

91. IF(ISL) PRINT * 5 "DIMENSION NCUVOL1, FAM(12)

92. EQUIVALENCE (CURVOL, NCUVOL)

93. IF(NLOC(I)) .EQ. 71 GO TO 2

94. CALL TOPLIN

95. WRITE(6,1152)

96. IF(A10) PRINT * 1 INCORRECT NUMBER OF ELEMENTS INPUT TO RADSOL

97. E = 1.0, TH = 93

98. CALL MEND

99. CALL EXIT

100. IF(ISL) PRINT * 3 "DIMENSION NLOC(I)

101. COMMON /CURVOL/ CURVOL

102. COMMON /CURVOL/ CURVOL

103. IF(ISL) PRINT * 5 "DIMENSION NCUVOL1, FAM(12)

104. EQUIVALENCE (CURVOL, NCUVOL)
CURREMLOC) = CURREMLOC) + FAX(CURREMLOC) + NER

180 CONTINUE
LOC = NNR
DC 223 1.4, 05
LOC = ISNA = 01
ARIES = CURREMLOC
NN = NCRVOLLOC 11
GO = CURREMLOC 1, 09
DC 210 3, 1.M.
LOC = LOC + 1
NODE = NCRVOLLOC
LOC = LOC + 1
QID = QID(NODE) = CURREMLOC 1, NER
190 CONTINUE
LOC = LOC + 1
200 CONTINUE
RETURN
200 WRITE(6,400)
600 FORMATTING 131111H+/IX* THE SURFACE ABSORPTIVITY DATA DOES NOT HAVE
XE THE CORRECT NUMBER OF VALUES */IX 13411H+11
GO TO 900
200 WRITE(6,400)
600 FORMATTING 131111H+/IX* THE SURFACE DATA DOES NOT HAVE THE CORRECT
X NUMBER OF VALUES */IX 13411H+11
GO TO 900
200 WRITE(6,400)
600 FORMATTING 131111H+/IX* THE SURFACE REFLECTIVITY DATA DOES NOT HAVE
XE THE CORRECT NUMBER OF VALUES */IX 13411H+11
GO TO 900
200 WRITE(6,400)
600 FORMATTING 131111H+/IX* THE SURFACE INCIDENT WAVE DATA DOES NOT HAVE
XE THE CORRECT NUMBER OF VALUES */IX 13411H+11
GO TO 900
200 WRITE(6,400)
600 FORMATTING 131111H+/IX* THE NOISE DATA FOR SURFACE I5, I Does NOT HAVE
X THE CORRECT NUMBER OF VALUES */IX 13411H+11
GO TO 900
200 WRITE(6,400)
600 FORMATTING 131111H+/IX* THE SUM OF THE AREAS FOR THE NODES ON SURF
FACE I5, I IS NOT EQUAL TO THE AREA OF THAT SURFACE */IX 13411H+11
GO TO 900
200 WRITE(6,400)
600 FORMATTING 131111H+/IX* THE EXTRA SPACE DATA DOES NOT HAVE THE COR
RECT NUMBER OF SPACES */IX 13411H+11
GO TO 900
200 WRITE(6,400)
600 FORMATTING 131111H+/IX* THE SURFACE NUMBER I5, I IN THE SURFACE CONNECT
ED DATA WAS NOT SUPPLIED IN THE SURFACE DATA */IX 13411H+11
GO TO 900
200 WRITE(6,400)
600 FORMATTING 131111H+/IX* ERROR OCCURRED IN SUBROUTINE RADSOL /X
200 WRITE(6,400)
600 FORMATTING 131111H+/IX* EXECUTION TERMINATED BY A PROGRAMMED HALT. /X
CALL ML581
200 WRITE(6,400)
600 FORMATTING 131111H+/IX* EXECUTION TERMINATED BY A PROGRAMMED HALT. /X
CALL ML581
200 WRITE(6,400)
600 FORMATTING 131111H+/IX* EXECUTION TERMINATED BY A PROGRAMMED HALT. /X
CALL ML581
200 WRITE(6,400)
600 FORMATTING 131111H+/IX* EXECUTION TERMINATED BY A PROGRAMMED HALT. /X
CALL ML581
200 WRITE(6,400)
600 FORMATTING 131111H+/IX* EXECUTION TERMINATED BY A PROGRAMMED HALT. /X
CALL ML581
1. SUBROUTINE REVPOLY(A,X)
2. C
3. DIMENSION A(I)
4. C
5. EQUIVALENCE (D,N)
6. C
7. D = A(1)
8. N = N
9. IF(MOD(N,2) .GT. 0) GO TO 20
10. IF(NAT+1) = 3) GO TO 10
11. X = A(2)
12. IF(Y .GE. A(3)) RETURN
13. X = A(I)
14. IF(Y .LE. A(N+1)) RETURN
15. DO 13, N = 2, 3
16. IF(Y = A(I+1)) RETURN
17. 13 = N+1
18. I = A(N-2) * (Y-A(I-1))/(A(I) - A(I-2))
19. RETURN
20. X = A(I)
21. RETURN
22. CONTINUE
23. GO TO 20
24. X = A(I)
25. IF(Y .LE. A(I)) RETURN
26. X = A(I)
27. IF(Y .GE. A(N)) RETURN
28. DO 27, N = 2, N
29. IF(Y = A(I+1)) RETURN
30. X = A(N-2) * (Y-A(I-1))/(A(I) - A(I-2))
31. RETURN
32. X = A(I)
33. RETURN
34. CONTINUE
35. WRITE (6,25) A(I)
36. 25 FORMAT ('WRONG ARRAY LENGTH FOR REVPOL, IC = ') CALL M60
37. CALL EXIT
38. END
SUBROUTINE SVASOLIA(U,$) 
DIMENSION A(N), B(N)
DOUBLE PRECISION OIN, IaOR
FACTORIZE GIVEN MATRIX BY MEANS OF SUBROUTINE WNSOL
A: TRANSPOSE(T)vT
CALL WNSOL(A,N)
FACTORIZE UPPER TRIANGLE MATRIX TO PREPARE INVERSION LOOP
IPIV = N*(N-1)/2
IND = IPIV
C INITIALIZE INVERSION-LOOP
DO I = 1,N
MIN = I
KEND = I - 1
LENF = N - KEND
IF(KEND) 9,9,2
J = IND
C INITIALIZE ROW-LOOP
DO K = IND,N
MORK = 0.00
RII = IN - I
LMOR = IPIV
LVER = J
START INNER LOOP
DO L = LVERS,MIN
LHOR LR *HO L
WORK = WORK * A(LVER)*A(LHOA)
AJ = WORK/A(LI)
IND = IPIV - MIN
LD : N*(N*I)/2
00 12 I=N,I,-I
ST: LD : LD * I
K = LO
SUM = 0.0
DO J = 1,N
SUM = SUM + A(K)*B(J)
K = K + J
CONTINUE
= K
CONTINUE
S2. (11) = SUM
12 CONTINUE
LD = 0
DO 9 I=1, N
LD = LD + 1
= LD
SUM = 0.0
DO 9 J=1,M
SUM = SUM + A(K)*B(J)
= K + J
CONTINUE
BE1 = SUM
12 CONTINUE
LD = 0
DO 9 I=1, N
LD = LD + 1
= LD
SUM = 0.0
DO 9 J=1,M
SUM = SUM + A(K)*B(J)
= K + J
CONTINUE
BE1 = SUM
7 CONTINUE
BE1 = SUM
8 CONTINUE
RETURN
15 RETURN
END
SUBROUTINE TIMECHK(TIME, RTIME)
  COMMON / FCON / CON1
  EQUIVALENCE (CON13, TIME1, (CON3, TIMEND), (CON10, OUTPUT)
  DATA CTIME1 / 0.0 /
  IF(CTIME1 .GT. 0.0) GO TO 100
  CALL CLOCK(TIME)
  ETIME = 0.0
  GO TO 200
  100 CALL CLOCK(TIME)
  ETIME = CTIME - CTIME1
  200 IF(KODE .EQ. 0) GO TO 350
  CALL LINECK(3)
  WRITE(6,300) ETIME
  300 FORMAT(16HCOMPUTER TIME = F9.3, 8H MINUTES)
  350 IF(TIME .GT. ETIME) RETURN
  CALL LINECK(2)
  WRITE(6,400) RTIME
  400 FORMAT(67HEXECUTION TERMINATED BECAUSE COMPUTER TIME EXCEEDS TIME REQUESTED, F9.3, 8H MINUTES)
  450 WRITE(6,475) RTIME
  475 FORMAT(38HCOMPUTER TIME EXCEEDS TIME REQUESTED, F9.3, 8H MINUTES)
  OUTPUT = 0.0
  RETURN
END
SUBROUTINE TOPLIN

COMMON /TITLE / N(20)

COMMON /FISCOV/ N(11)

C IF(N(20).EQ. 11) RETURN

N(20) = 11

R(N(20)) = N(N(20)) - 1

WRITE(6,100) N(20), N

10 FORMAT('IMPROVED NUMERICAL DIFFERENCING ANALYZER',/ 'SIENCA-1108 FORTRAN-V VERSION',/ 'RETURN')

11. RETURN

12. +GE .35 // 52 2041

13. RETURN

14. END
SUBROUTINE TPANT
LOGICAL LSRT, CHK
DIMENSION EXT(I)
COMMON /TEMP/ (I, J)
COMMON /SPACE/ NOIM, MTH, NEXT(I)
COMMON /TICOH/ KON(I)
COMMON /DOPI/ NAM, NAM, NTH
COMMON /IPINTW/ LNODE
EQUivalence (NEXT,EXT)
DATA LSRT / .FALSE. /
DATA NT / INT/
DATA NAC / INT /
IF(LN3DE .EQ. 0) CALL NNREAD()
CALL STNORD
IF(LSRT) GO TO 50
LSAT = .TRUE.
N0OM = NDIM * NNT
IF(NODE .LT. 0) GO TO 100
NNODE = NOIM *
CONTINUE
DO 30 J=2,NNT
K = NNT - J + 1
CHK = .TRUE.
00 20 N=I,K
NNI : NEXT(NNODE*N)
NEXT(KNODE#N)
NEXT(NODE#N)
CONTINUE
IF(CHK) GO TO 50
90 30 CONTINUE
IF(ND1 .LT. 12) GO TO 100
J = 6
L = 6
M = NTH + 1
IF( EXP .LT. M) L = M
00 50 J=1,L
M = NEXT(NNODE+N)
NEXT(MODE+N+1)
CHK = .FALSE.
NEXT(MODE+N) = NN1
NEXT(MODE+N+1) = NN
CONTINUE
IF(ECH) GO TO 50
90 50 CONTINUE
50 IF(NODM .LT. 12) GO TO 100
J = 1
L = 6
M = NTH + 1
50 IF( EXP .LT. M) L = M
00 50 J=1,L
M = NEXT(NNODE+N)
NEXT(MODE+N+1)
NEXT(MODE+N) = NN1
NEXT(MODE+N+1) = NN
CONTINUE
ECH = 50
50 IF(K3N(28) .LT. 60) GO TO 80
CALL TOPLIN
WRITE(6,75)
FORAT(IH)
ON(Z 2) = I
WRITE(6,90) (HT, NEXTI), EFFIT(I),IzI1,K,2)
00 FORMAT(1X, A1, 16, 1WH, 612.5, 151)
31.  034 = 034(2B) + 1
32.  IVL, ER, WR1 RETURN
33.  J = J + 1
34.  L = L + 6
35.  GO TO 40
36.  100 WRITE(6, 110) RDIM
37.  110 FORMAT(1SH1) * = INSUFFICIENT DYNAMIC STORAGE AVAILABLE FOR SUBR
38.  120 WRITE(6, 115) RH = RH
39.  STOP
40.  END.
APPENDIX B

USERS DESCRIPTION FOR PLOTA

This appendix presents user descriptions for a SINDA plotting routine, PLOTA, and a tape combining routine, MCOMB. Both routines are on the second file of the SINDA/Version 9 program tape but are main routines rather than user subroutines. A brief description of the routines and the user input description is given below.

PLOTA DESCRIPTION

The plot routine which is available on the SINDA program tape can be used with a history tape from a previous SINDA run to generate microfilm output. The items available for plotting are (1) pressures for each pressure node or pressure drop values for each tube, (2) valve positions for each valve, (3) flowrates for each tube and (4) temperatures for each temperature lump. Each of these items may be plotted as a function of mission time. The user specifies the grid time range to be plotted, a time label, and the items to be plotted. A number of history tapes may be combined prior to plotting the results. The user has the option of averaging any portion of the plotted curve and of specifying the range of the ordinate axis.

The system control cards and the data input cards for PLOTA are described below:

SYSTEM CONTROL CARDS FOR PLOTA

```
7 8 RUN
7 8 MSG
7 8 PLT
7 8 ASG A=XXX (SINDA PROGRAM TAPE)
7 8 ASG E=XXX (FIRST TAPE TO BE COMBINED)
7 8 ASG F=XXX (SECOND TAPE TO BE COMBINED)
```

Add additional ASG cards as required for tapes to be combined.

B-1
ASG T=XXX (COMBINED TAPE)
XQT CUR
TRW A
PEF A
IN A
TRI A
XQT PLOTA
DATA CARDS
EØF

PLOTA DATA CARDS

<table>
<thead>
<tr>
<th>Columns</th>
<th>Format</th>
<th>Title</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Card 1 (Title Card)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1-72</td>
<td>12A6</td>
<td>TITLEA</td>
<td>Any 72 alphanumeric characters to be used as heading for each frame of plots</td>
</tr>
<tr>
<td>Card 2 (Parameter Card)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1-10</td>
<td>F10.0</td>
<td>TA</td>
<td>First value of time to be plotted (hours).</td>
</tr>
<tr>
<td>11-20</td>
<td>F10.0</td>
<td>TZ</td>
<td>Last value of time to be plotted (hours).</td>
</tr>
<tr>
<td>21-30</td>
<td>F10.0</td>
<td>TPG</td>
<td>Time range for each grid. Number of grids drawn will be (TZ-TA)/TPG. (If TPG is left blank, the job will terminate.)</td>
</tr>
<tr>
<td>31-35</td>
<td>15</td>
<td>ITMX</td>
<td>Time scale label:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>= 1, &quot;SECONDS&quot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>= 2, &quot;MINUTES&quot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>= 3, &quot;HOURS&quot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Any other value, &quot;****&quot;</td>
</tr>
<tr>
<td>36-40</td>
<td>15</td>
<td>MPNT</td>
<td>Print control code</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>= 1, prints information to be plotted while loading the plot tape</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>≠ 1, will not print information to be plotted</td>
</tr>
<tr>
<td>Columns</td>
<td>Format</td>
<td>Title</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>--------</td>
<td>-------</td>
<td>-------------</td>
</tr>
<tr>
<td>41-45</td>
<td>I5</td>
<td>NTP</td>
<td>Number of tapes to be combined. Use a negative number if start and/or stop times are specified on Card 3 for any tape to be combined.</td>
</tr>
<tr>
<td>46-50</td>
<td>I5</td>
<td>KT</td>
<td>Logical unit number to which tape to be plotted is assigned. If left blank, unit 23 is assumed. (See Table B-I) The combined tape is assigned to this unit.</td>
</tr>
<tr>
<td>51-55</td>
<td>I5</td>
<td>INC</td>
<td>( \text{INC} = 1, ) every time point and associated data value from the tapes to be combined will be transferred to the combined tape. ( \text{INC} = 2, ) every second time point and associated data values will be transferred to the combined tape. etc.</td>
</tr>
<tr>
<td>56-60</td>
<td>I5</td>
<td>IUNIT</td>
<td>Logical unit number to which first tape to be combined is assigned. If left blank, unit 7 is assumed.</td>
</tr>
<tr>
<td>61-70</td>
<td>F10.0</td>
<td>ASTRT</td>
<td>Beginning time for averages (hours).</td>
</tr>
<tr>
<td>71-80</td>
<td>F10.0</td>
<td>ASTOP</td>
<td>Ending time for averages (hours).</td>
</tr>
</tbody>
</table>

Card 3 (Required only if NTP < 0. See Card 2 columns 41-45)

| 1-5     | F5.3   | XSTART | First time point from first tape to be combined which will be transferred to the combined tape. |
| 6-10    | F5.3   | XSTOP  | Last time point from first tape to be combined which will be transferred to the combined tape. |

Repeat XSTART and XSTOP in five column fields for each tape to be combined.

Card 4 (Item Card)

| 1-5     | I5     | ITEM   | The item number to be plotted. Use a negative value if this item is to start a |

B-3
<table>
<thead>
<tr>
<th>Columns</th>
<th>Format</th>
<th>Title</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>6-7</td>
<td>A2</td>
<td>ITYPE</td>
<td>A two character item type code which determines the type of item to be plotted.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>= XX, pressures or pressure drop values</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>= VP, valve positions</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>= FR, flow rates</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>= ST, node-temperatures</td>
</tr>
<tr>
<td>8</td>
<td>I1</td>
<td>IREL</td>
<td>= 0, node numbers are actual numbers</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>= 1, node numbers are relative numbers</td>
</tr>
<tr>
<td>9-10</td>
<td>I2</td>
<td>KAVG</td>
<td>&gt; 0, calculate the numerical average of this item over the interval specified by ASTRT and ASTØP in columns 61-70 and 71-80 of Card 2.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>&gt; 9, plot the average on the frame with this item.</td>
</tr>
<tr>
<td>11-58</td>
<td>8A6</td>
<td>TITLES</td>
<td>Item description to be printed at the top of each grid, along with the plotting symbol which is generated and used by the program.</td>
</tr>
<tr>
<td>59-60</td>
<td></td>
<td>Blank</td>
<td>The next two values are optional (may be left blank) on the cards whose item numbers are negative and are ignored on all other item cards.</td>
</tr>
<tr>
<td>61-70</td>
<td>F10.0</td>
<td>YLØ</td>
<td>The minimum (reference) value on the Y-axis</td>
</tr>
<tr>
<td>71-80</td>
<td>F10.0</td>
<td>YHI</td>
<td>The maximum value on the Y-axis</td>
</tr>
</tbody>
</table>

The above limits will be replaced by the program if numbers outside this range are found in the histories of the items to be plotted on this grid.
Columns | Format | Title | Description
--- | --- | --- | ---
Repeat Card 4 for each item to be plotted.

Card 5
1-80 | Blank

Card 6
1-80 | Blank

If additional history tapes are to be plotted, repeat Card 1 and subsequent cards for each additional history tape.

**COMBINE ROUTINE DESCRIPTION**

The combine routine, MCOMB, can be used to combine as many as six history tapes into one history tape prior to its being plotted or being compared to another tape. The combined tape which is generated can be saved for future use if required. The user selects the frequency with which the time points and associated data values on the original tapes are added to the new tape. That is, every time point on the original tape can be added to the new tape or every second, third, etc., point can be added depending on the requirements for the combined tape.

The combine routine is a very useful feature if several history tapes are generated on a long mission run. By combining these tapes before plotting, a continuous plot of the mission can be obtained. The convenience of the combine routine can also be observed when mission runs made with different time increments are compared. Obviously, the run made with the smaller time increment will take more computer time than the run made with the larger time increment, and will probably require at least one "restart". In such a situation, there would be two history tapes with the smaller time increment to compare to one with the larger time increment. Normally, this would take two separate runs. However, with the new arrangement, the two tapes with the smaller time increment can be combined and then compared to the tape with the larger time increment on the same run.

The system control cards and the data input cards for MCOMB are described below.
USER'S MANUAL FOR MCOMB ROUTINE

CONTROL AND PROGRAM CARDS

RUN

MSG

ASG A=XXX (SINDA PROGRAM TAPE)

ASG E=XXX (First Tape to be combined)

ASG F=XXX (Second tape to be combined)

Add additional ASG cards as required for tapes to be combined.

ASG K=XXX (Combined Tape)

XQT CUR

TRW A

PEF A

IN A

TRI A

XQT MCOMB

DATA CARDS

E0F
### DATA CARDS FOR MCOMB ROUTINE

<table>
<thead>
<tr>
<th>Columns</th>
<th>Format</th>
<th>Title</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Card 1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1-5</td>
<td>I5</td>
<td>NTAPE</td>
<td>Number of tapes to be combined. Use a negative value if start and/or stop times are specified on Card 2.</td>
</tr>
<tr>
<td>6-10</td>
<td>I5</td>
<td>IUNIT</td>
<td>Logical unit number to which first tape to be combined is assigned. If left blank, unit 7 is assumed.</td>
</tr>
<tr>
<td>11-15</td>
<td>I5</td>
<td>KT</td>
<td>Logical unit number to which combined tape is assigned. If left blank, unit 13 is assumed.</td>
</tr>
</tbody>
</table>
| 16-20   | I5     | KODE2   | = 1, time to be added to the times read from tapes to be combined will be supplied on Card 3.  
|         |        |         | = 0, transfer times as read from tapes to be combined. |
| 21-25   | I5     | INC     | = 1; every time point and associated data values read from the tapes to be combined will be transferred to the combined tape.  
|         |        |         | = 2, every second time point and associated data values will be transferred to the combined tape.  
|         |        |         | etc. |

Card 2 (Required only if NTAPE < 0. See Card 1 columns 1-5)

<table>
<thead>
<tr>
<th>Columns</th>
<th>Format</th>
<th>Title</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-5</td>
<td>F5.3</td>
<td>XSTART</td>
<td>First time point from first tape to be combined which will be transferred to the combined tape.</td>
</tr>
<tr>
<td>6-10</td>
<td>F5.3</td>
<td>XSTOP</td>
<td>Last time point from first tape which will be transferred to the combined tape.</td>
</tr>
<tr>
<td>Columns</td>
<td>Format</td>
<td>Title</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>--------</td>
<td>-------</td>
<td>-------------</td>
</tr>
<tr>
<td>1-10</td>
<td>F10.0</td>
<td>ADD</td>
<td>Time to be added to each time read from first tape to be combined.</td>
</tr>
</tbody>
</table>

Repeat ADD in 10 column fields for each tape to be combined.
<table>
<thead>
<tr>
<th>FORTRAN UNIT NO.</th>
<th>I/O DEVICE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A</td>
</tr>
<tr>
<td>2</td>
<td>B</td>
</tr>
<tr>
<td>3</td>
<td>C</td>
</tr>
<tr>
<td>4</td>
<td>D</td>
</tr>
<tr>
<td>7</td>
<td>E</td>
</tr>
<tr>
<td>8</td>
<td>F</td>
</tr>
<tr>
<td>9</td>
<td>G</td>
</tr>
<tr>
<td>10</td>
<td>H</td>
</tr>
<tr>
<td>11</td>
<td>I</td>
</tr>
<tr>
<td>12</td>
<td>J</td>
</tr>
<tr>
<td>13</td>
<td>K</td>
</tr>
<tr>
<td>14</td>
<td>L</td>
</tr>
<tr>
<td>15</td>
<td>M</td>
</tr>
<tr>
<td>16</td>
<td>N</td>
</tr>
<tr>
<td>18</td>
<td>O</td>
</tr>
<tr>
<td>19</td>
<td>P</td>
</tr>
<tr>
<td>20</td>
<td>Q</td>
</tr>
<tr>
<td>21</td>
<td>R</td>
</tr>
<tr>
<td>22</td>
<td>S</td>
</tr>
<tr>
<td>23</td>
<td>T</td>
</tr>
<tr>
<td>24</td>
<td>U</td>
</tr>
<tr>
<td>25</td>
<td>V</td>
</tr>
<tr>
<td>26</td>
<td>W</td>
</tr>
<tr>
<td>27</td>
<td>X</td>
</tr>
<tr>
<td>28</td>
<td>Y</td>
</tr>
<tr>
<td>29</td>
<td>Z</td>
</tr>
</tbody>
</table>