Natural Circulation in the Blanket Heat Removal System During a Loss-of-Pumping Accident (LOFA) Based on Initial Conceptual Design

by
L. L. Hamm
Westinghouse Savannah River Company
Savannah River Site
Aiken, South Carolina 29808

S. Y. Lee

M. A. Shadday

F. G. Smith III

DOE Contract No. DE-AC09-96SR18500

This paper was prepared in connection with work done under the above contract number with the U. S. Department of Energy. By acceptance of this paper, the publisher and/or recipient acknowledges the U. S. Government's right to retain a nonexclusive, royalty-free license in and to any copyright covering this paper, along with the right to reproduce and to authorize others to reproduce all or part of the copyrighted paper.
NATURAL CIRCULATION IN THE BLANKET HEAT REMOVAL SYSTEM DURING A LOSS-OF-PUMPING ACCIDENT (LOFA) BASED ON INITIAL CONCEPTUAL DESIGN

L. Larry Hamm
Si Young Lee
M. Andy Shadday
Frank G. Smith, III

Westinghouse Savannah River Company
Savannah River Site
Aiken, SC 29808
DISCLAIMER

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

This report has been reproduced directly from the best available copy.

Available to DOE and DOE contractors from the Office of Scientific and Technical Information, P.O. Box 62, Oak Ridge, TN 37831; prices available from (615) 576-8401.

Available to the public from the National Technical Information Service, U.S. Department of Commerce, 5285 Port Royal Road, Springfield, VA 22161.
DISCLAIMER

Portions of this document may be illegible in electronic image products. Images are produced from the best available original document.
NATURAL CIRCULATION IN THE BLANKET HEAT REMOVAL SYSTEM DURING A LOSS-OF-PUMPING ACCIDENT (LOFA) BASED ON INITIAL CONCEPTUAL DESIGN

SAVANNAH RIVER TECHNOLOGY CENTER

L. Larry Hamm
Si Young Lee
M. Andy Shadday
Frank G. Smith, Ill

Publication Date: July, 1998

Westinghouse Savannah River Company
Savannah River Site
Aiken, SC 29808

Prepared for the U.S. Department of Energy under Contract No. DE-AC09-96SR18500
The internal technical review function is being performed at the APT project level and is coordinated through LANL.
Table of Contents

1 Summary .................................................................................................................. 1
2 Background .............................................................................................................. 1
3 Description of the Model ........................................................................................ 2
4 Vertical Mixed Convection Friction Factors .......................................................... 4
5 Discussion and Results ............................................................................................ 4
6 Conclusions ............................................................................................................. 12
7 References .................................................................................................................. 12

Appendix A: Derivation of the Governing Equations
Appendix B: Buoyancy-Assisted and Buoyancy-Opposed Friction Factor Multipliers
Appendix C: Sample Input Deck
Appendix D: Code Listing
List of Figures

Figure 1  Schematic of the blanket primary HR system with three lumped modules and one lumped heat exchanger loop ................................................................. 3
Figure 2  Natural circulation flowrates during a T/B LOFA ........................................... 6
Figure 3  Expanded view of the flowrates during the flow reversal process .......................... 6
Figure 4  Transient fluid temperatures of the two headers and the shell side of the heat exchanger during a T/B LOFA ................................................................. 7
Figure 5  Transient temperatures in the headers and Leg 1 during a T/B LOFA ...................... 8
Figure 6  Transient temperatures in the headers and Leg 2 during a T/B LOFA ...................... 8
Figure 7  Transient temperatures in the headers and Leg 3 during a T/B LOFA ...................... 8
Figure 8  Transient temperatures in the headers and the tube side of the heat exchanger during a T/B LOFA ................................................................. 9
Figure 9  Outlet header pressure during a T/B LOFA ..................................................... 10
Figure 10  Natural circulation flowrates during a blanket only LOFA ................................. 11
Figure 11  Transient fluid temperatures in the two headers and the shell side of the heat exchanger during a blanket only LOFA ...................................................... 11
Figure A-1  Schematic of the blanket primary HR system with three lumped modules and one lumped heat exchanger loop ................................................................. A1
Figure A-2  Schematic of one of the module legs of the natural convection network showing the control volumes used to derive the momentum equation ................................................. A2
Figure A-3  Schematic of the heat exchanger leg of the natural convection network .................. A3
Figure A-4  Schematic of the outlet header showing the mass flowrates ............................... A4
Figure A-5  Schematic of the heat exchanger showing the heat transfer to the secondary side ................................................................. A6
Figure B-1  Friction factor multipliers for buoyancy-assisted and buoyancy-opposed mixed convection in vertical channels ................................................................. B2

List of Tables

Table 1  Cruciform-type module descriptions showing NC model relationship ................................................................. 2
Table 2  Normal operation deposited power levels ................................................................. 5
1 Summary

A transient natural convection model of the APT blanket primary heat removal (HR) system was developed to demonstrate that the blanket could be cooled for a sufficient period of time for long term cooling to be established following a loss-of-flow accident (LOFA). The particular case of interest in this report is a complete loss-of-pumping accident. For the accident scenario in which pumps are lost in both the target and blanket HR systems, natural convection provides effective cooling of the blanket for approximately 68 hours, and, if only the blanket HR systems are involved, natural convection is effective for approximately 210 hours. The heat sink for both of these accident scenarios is the assumed stagnant fluid and metal on the secondary sides of the heat exchangers.

2 Background

Loss of power could leave the APT Target/Blanket (T/B) Facility without running pumps resulting in no forced circulation in the primary and secondary heat removal systems for an extended period of time. Natural circulation (NC) would occur in the primary HR system; however, without circulation through the secondary sides of the heat exchangers there would not be an ultimate heat sink for the decay power in the blanket modules. To prevent some or all of the blanket modules from eventually overheating and losing structural integrity, either power would have to be restored or the cavity flood system would have to be activated. There would be a period of time following the loss of forced convection in the HR systems when the modules would be cooled by natural convection. The stagnant water in the secondary sides of the heat exchangers could serve as a temporary heat sink, diminishing in effectiveness as it heats up. A transient single-phase natural convection model of the primary HR system was developed to determine the duration of time of effective passive cooling of the modules. A special purpose fortran program was written, called NCLOFA, and a listing of the code is provide in Appendix D.

The accident under consideration is a facility loss-of-power that results in the simultaneous loss of both the T/B primary and secondary coolant pumps. This accident is unmitigated except for a beam shutdown. The primary HR systems would ultimately dry out by boiling off the coolant inventories, if corrective action is not taken. This would occur much sooner in the target system than the blanket because of the differences in both metal and coolant masses. The target can survive high temperatures where thermal radiation to the blanket is an acceptable heat removal mechanism; whereas, without cavity flooding, the blanket cannot maintain its structural integrity very long after dryout. Based on results presented in Ref. [1], the target is assumed to dry out twenty-four hours after the onset of the accident and the target decay power is then deposited in the Module 1 decouplers. The model is run until the HR system coolant reaches its saturation temperature. Beyond this point two-phase flow is established; however, the NC model chosen for this set of analyses is limited to single-phase conditions. Even though it is anticipated that significant cooling capability further exists once boiling begins. No formal credit for boiling capability is taken at this time, since a significant increase in modeling complexity would only extend the existing acceptable period of time available prior to taking corrective action. The consequences of this accident
scenario envelop those of the scenario where only the blanket primary and secondary coolant pumps are lost.

3 Description of the Model

The transient single-phase NC model of the primary HR system is based on a flow network with three lumped modules and one lumped heat exchanger, as shown in Fig. 1. The three lumped modules constitute three parallel legs between the fixed inlet and outlet headers, and the heat exchanger is in a fourth leg that forms the network into a closed loop. The various legs of the flow network that make up the NC model represent lumped flow paths. Table 1 provides a brief description of the 16 module cruciform-type units in the current blanket design (see Ref. [2] for further details). These modules are combined into lumped flow paths (referred to as legs) as listed below:

- "Leg 1" – the lateral Row-1/decoupler modules listed as Modules 1 and 4 in Table 1;
- "Leg 2" – the remaining vertical modules listed as Modules 2, 3, 5, 6, 7, 8, and 9 in Table 1;
- "Leg 3" – all horizontal modules listed as Modules 10, 11, 12, 13, 14, 15, and 16 in Table 1;
- "HX leg" – blanket primary HR loop containing the two heat exchangers, two pumps, and piping from the outlet header back around to the inlet header closing the flow network.

Table 1 Cruciform-type module descriptions showing NC model relationship.

<table>
<thead>
<tr>
<th>Module Number (Location)</th>
<th>Module Types (downflow / upflow)</th>
<th>NC Model ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>Module 1: (Front 1&lt;sup&gt;st&lt;/sup&gt; Lateral)</td>
<td>Decoupler / Row 1</td>
<td>Leg 1</td>
</tr>
<tr>
<td>Module 4: (Back 1&lt;sup&gt;st&lt;/sup&gt; Lateral)</td>
<td>Decoupler / Row 1</td>
<td></td>
</tr>
<tr>
<td>Module 2: (Front 2&lt;sup&gt;nd&lt;/sup&gt; Lateral)</td>
<td>Row 3 / Row 2</td>
<td></td>
</tr>
<tr>
<td>Module 3: (Front 2&lt;sup&gt;nd&lt;/sup&gt; Lateral)</td>
<td>Row 3 / Row 2</td>
<td></td>
</tr>
<tr>
<td>Module 5: (Back 2&lt;sup&gt;nd&lt;/sup&gt; Lateral)</td>
<td>Row 3 / Row 2</td>
<td></td>
</tr>
<tr>
<td>Module 6: (Back 2&lt;sup&gt;nd&lt;/sup&gt; Lateral)</td>
<td>Row 3 / Row 2</td>
<td></td>
</tr>
<tr>
<td>Module 7: (1&lt;sup&gt;st&lt;/sup&gt; Downstream)</td>
<td>Decoupler / Row 1</td>
<td></td>
</tr>
<tr>
<td>Module 8: (2&lt;sup&gt;nd&lt;/sup&gt; Downstream)</td>
<td>Row 2 / Row 2</td>
<td></td>
</tr>
<tr>
<td>Module 9: (3&lt;sup&gt;rd&lt;/sup&gt; Downstream)</td>
<td>Row 2 / Row 2</td>
<td></td>
</tr>
<tr>
<td>Module 10: (Blanket Upstream)</td>
<td>Decoupler / Row 2</td>
<td></td>
</tr>
<tr>
<td>Module 11: (Lower Front Upper)</td>
<td>Decoupler / Row 2</td>
<td></td>
</tr>
<tr>
<td>Module 12: (Lower Front Lower)</td>
<td>Row 2 / Row 2</td>
<td></td>
</tr>
<tr>
<td>Module 13: (Lower Back Upper)</td>
<td>Decoupler / Row 2</td>
<td></td>
</tr>
<tr>
<td>Module 14: (Lower Back Lower)</td>
<td>Row 2 / Row 2</td>
<td></td>
</tr>
<tr>
<td>Module 15: (Upper Front)</td>
<td>Row 2 / Row 2</td>
<td></td>
</tr>
<tr>
<td>Module 16: (Upper Lower)</td>
<td>Row 2 / Row 2</td>
<td></td>
</tr>
<tr>
<td>Primary HR Loop</td>
<td>Piping, Heat Exchangers, Pumps</td>
<td>HX Leg</td>
</tr>
</tbody>
</table>
The labels above are also provided in Fig. 1. The pressurizer is not modeled explicitly but does establish the network reference pressure at the inlet header. The heated upflow and downflow sections of a module are treated separately in the momentum equations, and the impact of buoyancy-assisted/opposed mixed convection flow is accounted for through a Rayleigh number dependent modification of the Darcy friction factor. The heated solid sections of the modules are lumped together with the coolant flowing through them in the thermal energy equations. The aluminum clad lead plates and the fluid in the adjacent channels are assumed to be isothermal. This is reasonable at the low metal power densities encountered under post beam shutdown conditions and with the high heated surface area to fluid volume ratios for the flow channels. This simplification avoids the necessity of solving conjugate heat transfer problems within the modules. There are four momentum equations, one for each leg in the loop and a continuity equation for the outlet header. These five equations are solved simultaneously for the four leg mass flowrates and the outlet header pressure each time step using a standard Newton-Raphson method. The inlet header pressure is a boundary condition that specifies the system reference pressure.

![Figure 1 Schematic of the blanket primary HR system with three lumped modules and one lumped heat exchanger loop.](image)

For determining the transient temperature distribution within the network, each of the module legs of the network is divided into five cells, while the heat exchanger leg is divided into three. Under low flow conditions for which this model is intended, this spatial resolution is adequate. A thermal energy equation is derived for each leg cell, each of the headers, and for the shell (or secondary) side of the heat exchanger. The energy equations are differenced in time allowing a varying degree of implicitness.
specified by the user. The energy equations are solved iteratively each time step in a two-step process. The two header temperatures and the temperature on the secondary side of the heat exchanger are updated. With these updated values, the network leg temperatures are updated by marching in the flow direction from the upstream to the downstream header for each leg. This process is repeated until convergence occurs. The momentum and thermal energy equations are derived in Appendix A.

The natural circulation is driven by the time dependent decay power in the heated sections of the module legs. After 24 hours have elapsed, the total target decay power is deposited in the decoupler section of Leg 1. This is the heated section in Leg 1 closest to the inlet header. This additional power is ramped in over a period of one hour. The NC simulation commences one hour after the onset of the accident and continues until the primary coolant reaches its saturation temperature. The model predicts the transient mass flowrates in the network legs, the outlet header pressure, the transient temperature distribution in the primary coolant, and the shell side heat exchanger temperature. Appendix C contains a sample-input file and Appendix D contains a code listing of the NCLOFA program.

4 Vertical Mixed Convection Friction Factors

The Darcy friction factors for mixed forced/free laminar convection through vertical passages can differ significantly from the forced convection values defined as sixty-four over the Reynolds number. In upflow through a heated vertical channel, buoyancy flattens the parabolic radial velocity profile and increases the wall velocity gradient, thereby, increasing the wall shear. In downflow, buoyancy decreases the radial velocity gradient at the wall and therefore the wall shear. The friction factor for heated upflow is increased, and for downflow it is decreased. No effect exists for horizontal flow paths.

Based on a plot of Fanning friction factors for buoyancy assisted and retarded flows between vertical flat plates in Ref. [3], correlations were developed for friction factor multipliers as functions of a modified Rayleigh number. These correlations are developed in Appendix B. They are used in the NC model to modify the friction factors in the heated sections of the two network vertical module legs.

5 Discussion and Results

Two facility loss-of-power accident scenarios were simulated with the NC model: a simultaneous loss of both the T/B primary and secondary coolant pumps, and a loss of only the blanket primary and secondary coolant pumps. The first scenario is the more interesting of the two because flow reversals occur. It is also the more limiting case because, with the addition of the target decay power, more power (approximately double) is transferred to the coolant. The results of this accident simulation are described first and in detail. The results for the latter scenario are described primarily by comparison and contrast with the results of the former.

Figure 2 shows, for the first accident scenario, the network mass flowrates for the three modules and the heat exchanger leg as functions of elapsed time. By definition, positive flow through the three module legs travels from the inlet header to the outlet header and return flow occurs through the heat exchanger leg. During normal operation (NO) and prior to the depositing of the target power in the decoupler, the flow is positive throughout the network. The target is assumed to dry out within twenty-four hours of the
onset of the accident, and thereafter, the target decay power is dissipated by thermal radiation to the cooler adjacent blanket system decouplers. The blanket decouplers virtually surround the ladder assemblies, neutronically decoupling the target from the lead blanket. It is assumed that all thermal radiation emanating from the target ladders is deposited into the decouplers. This power is assumed to be ramped in over a period of one hour. With the addition of the target power, significantly more power is dissipated in the decoupler than in the Row-1 section of Leg 1, and buoyancy causes a flow reversal to occur. Deposited power levels under NO conditions are given in Table 2. Figure 3 shows an expanded view of the network mass flowrates during the period in which the flow reversals occur. Shortly after the flow reversal occurs in Leg 1 of the network, the flows reverse in Leg 3 and the Hx leg. The flow does not reverse in Leg 2. There are no preferred flow directions, due to buoyancy, in Leg 3 and the Hx leg of the network, so the flows in these legs readily reverse as consequences of the flow reversal in Leg 1. Buoyancy dictates the flow direction in Leg 2, and since the relative power distribution in this module doesn’t change, neither does its flow direction.

Table 2 Normal operation deposited power levels.

<table>
<thead>
<tr>
<th>Model ID</th>
<th>Deposited Power (MW)</th>
<th>Deposited Power (MW)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Leg 1</td>
<td>8.222</td>
<td>15.768</td>
</tr>
<tr>
<td>Leg 2</td>
<td>9.083</td>
<td>17.695</td>
</tr>
<tr>
<td>Leg 3</td>
<td>0.0</td>
<td>5.712</td>
</tr>
<tr>
<td>HX Leg</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>Totals</td>
<td>56.48 (blanket)</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>68.01 (targets)</td>
<td>-</td>
</tr>
</tbody>
</table>

Figure 4 shows the temperatures of the two headers and the shell side of the heat exchanger. Prior to the flow reversals, the inlet header is colder than the outlet header, which is to be expected since the flows in all three modules are positive. After the flow reversals, the outlet header is colder than the inlet header. The shell side of the heat exchanger has the lowest of the three temperatures. This is the system heat sink. The increased heating rate of the fluid after the target power is deposited in the decoupler is evident from the slopes of the transient temperature plots.
Figure 2 Natural circulation flowrates during a T/B LOFA.

Figure 3 Expanded view of the flowrates during the flow reversal process.
Figures 5 through 7 show the two header temperatures and the temperatures in the two heated sections of each of the three legs, respectively. Figure 8 shows the two header temperatures and the heat exchanger temperature. The effects of the flow reversals on the relative magnitudes of the temperatures in the heated sections of legs one and three are evident. The flow does not reverse in Leg 2, and therefore, the downstream heated section remains hotter than the upstream heated section throughout the accident scenario. These two temperatures exceed the inlet header temperature after the flow reversals because the flow in this leg occurs from the inlet header to the outlet header. The highest system temperature occurs in this module. The heat exchanger primary (or tube side) temperature is equal to the inlet header temperature prior to the flow reversals, and it is slightly lower than the outlet header temperature after the flow reversals. After the flow reversals, the outlet header temperature is slightly greater than the heat exchanger temperature because Leg 2 continues to discharge into the outlet header.

![Graph showing fluid temperatures of headers and heat exchanger shell](image.png)

**Figure 4** Transient fluid temperatures of the two headers and the shell side of the heat exchanger during a T/B LOFA.
Figure 5  Transient temperatures in the headers and Leg 1 during a T/B LOFA.

Figure 6  Transient temperatures in the headers and Leg 2 during a T/B LOFA.
Figure 7 Transient temperatures in the headers and Leg 3 during a T/B LOFA.

Figure 8 Transient temperatures in the headers and the tube side of the heat exchanger during a T/B LOFA.
Figure 9 shows the transient outlet header pressure. It is constant at 107.3 psia, given a specified inlet header pressure of 106.24 psia. The difference between the two pressures is hydrostatic. In each leg of the network, the net buoyant force balances the dynamic pressure drop force. At the inlet header pressure, the saturation temperature is 166.8°C and the inlet header reaches this temperature 67.9 hours after the onset of the accident. Once the mean fluid temperature reaches saturation, bulk boiling will commence and the system will start to pressurize. The primary and secondary coolants will continue to heat up as the saturation temperatures increase due to system pressurization. Since pressurizing the primary blanket HR system would threaten its integrity, bulk boiling should be avoided. The point at which the primary coolant temperature reaches saturation is defined as the limit of effective natural convection cooling by the blanket HR system.

If the LOFA is confined only to the blanket HR system, the period of adequate natural convection cooling under single-phase flow is significantly longer than for the scenario just considered, in which both the target and blanket systems are involved. As Table 2 shows, the blanket only constitutes ~45% of the total deposited power being generated in the T/B facility. Figure 10 shows the mass flowrates in the three legs and the Hx leg of the NC network. The flowrates decrease monotonically due to the decay of the internal heat generation in the blanket modules. There is no flow reversal because the relative power distribution within the blanket modules remains constant. Figure 11 shows the transient fluid temperatures in the inlet header, outlet header, and on the shell side of the heat exchanger. Approximately 210 hours are required for the fluid in the outlet header to reach its saturation temperature of 167.2°C. This is considerably longer than 68 hours of effective natural convection cooling in the accident scenario that involves the target.
Figure 10  Natural circulation flowrates during a blanket only LOFA.

Figure 11  Transient fluid temperatures in the two headers and the shell side of the heat exchanger during a blanket only LOFA.
6 Conclusions

The period of effective single-phase natural convection cooling of the APT blanket, following a facility loss-of-power accident that results in the simultaneous loss of both the T/B primary and secondary pumps, is approximately 68 hours. This is the time required for the coolant to reach its saturation temperature. The transient heat sink for this accident scenario is the stagnant fluid in the shell side of the heat exchanger. If the target HR system is not involved in this accident scenario, the period of effective natural convection cooling is extended to approximately 210 hours. These are the allowable time periods available for restoring electrical power to the pumps or establishing an alternate means for cooling the blanket assemblies.

The volume of assumed stagnant water on the shell sides of the heat exchangers and the masses of metal in the heat exchangers are the heat sinks for the NC model. Other than sensible heating, no other phenomena with respect to the shell side of the heat exchanger were considered in this accident analysis. If the pressure in the blanket secondary HR system is less than that in the primary system, the secondary side fluid would boil before the primary fluid reached saturation. This probably will not be the case since maintaining the secondary side heat exchanger pressure at a higher value than the primary side is desirable for radionuclide containment. This aspect of the accident analysis does warrant further consideration. Boiling on the secondary side could provide effective cooling for a while, but the heat sink would completely lose its effectiveness after dryout. The fluid in the secondary HR system could also circulate by natural convection, thereby, extending the period of time that the system is an effective heat sink for the primary HR system.

7 References


Appendix A: Derivation of the Governing Equations

The primary HR system flow network consisting of three lumped legs, as shown in Fig. A-1, is divided up into several interconnected discrete control volumes. To derive the governing equations for transient single-phase natural convection for this simplified model, the incompressible continuity, momentum, and thermal energy equations were applied to the control volumes. A Boussinesq fluid was assumed to handle the buoyancy terms in the momentum equations. Four momentum equations, one for each leg, and the continuity equation for the outlet header are solved simultaneously for the mass flowrates in each leg and the outlet header pressure. The thermal energy equations for the header temperatures, the heat exchanger shell side temperature, and the temperature distributions within each network leg are solved separately from the momentum equations.

Figure A-1 Schematic of the blanket primary HR system with three lumped modules and one lumped heat exchanger loop.

Figure A-2 shows one of the lumped module legs of the network and a close up of a representative control volume of one of the sections of the leg. The external forces acting on the fluid in the control volume are shown. They are the normal facing pressure forces, the body force, and the wall shear (or frictional force). The linear momentum equation (Eq. a1) is applied to each of the five control volumes within a leg. Equation a2 is the body force term and Eq. a3 is the frictional force term. The "n" in the subscript refers to the leg number, and the second number refers to the cell number, one through five. Equation a4 is the momentum equation for the first cell of the leg.
There are four additional equations, one for each of the other cells. If the area changes between cells are neglected, the five momentum equations can be added together to yield a single leg momentum equation (Eq. a5) with the header pressures and the leg mass flowrate as unknowns. At the low flowrates expected under natural convection conditions, the pressure distributions in the network legs will be close to hydrostatic. The flow area expansions/contractions occur in pairs with the same elevation, points 2 and 5 and points 3 and 4 in Fig. A-2. For these reasons the area changes can be neglected. The alternative would be to calculate intermediate pressures at the cell faces, and this would significantly increase the number of simultaneous non-linear equations to be solved and from an accuracy perspective is not warranted.

Figure A-2  Schematic of one of the module legs of the natural convection network showing the control volumes used to derive the momentum equation.

\[
\frac{d}{dt} (m\bar{v}) + \dot{m}(\bar{v}_{out} - \bar{v}_{in}) = \sum \bar{F} \quad (a1)
\]

\[
F_B = \rho_o [1 - \beta(T_{in} - T_o)]A_{nl}g\Delta z_{nl} \quad (a2)
\]

\[
F_f = \frac{f_{nl}L_{nl}\dot{m}_a^2}{2\rho_o D_{nl}A_{nl}} \quad (a3)
\]

\[
L_{nl} \frac{d\dot{m}_a}{dt} = F_B + F_f + P_1A_{nl} - P_2A_{nl} \quad (a4)
\]
\[
\sum_{i} L_{ai} \frac{d\dot{m}_{ai}}{dt} = \sum_{i} (F_{fai} + F_{fa}) + P_{in} A_{ai} - P_{out} A_{n5}
\]  

(a5)

where

- \( F \) - force
- \( \rho \) - fluid density
- \( f \) - Darcy friction factor
- \( \dot{m} \) - mass flowrate
- \( \beta \) - coefficient of thermal expansion
- \( P \) - pressure
- \( A \) - flow area
- \( T \) - temperature
- \( L \) - cell length
- \( g \) - gravitational constant
- \( D \) - hydraulic diameter
- \( \Delta z \) - elevation difference

Figure A-3 shows a schematic of the network leg that contains the heat exchanger. This leg is divided into three sections. The overall momentum equation for the mass flowrate in this leg is derived in the same manner as the equations for the module legs.

Figure A-3 Schematic of the heat exchanger leg of the natural convection network.

Figure A-4 is a schematic of the outlet header that shows the mass flowrates entering and exiting the header. Positive flowrates are shown. The continuity equation is applied to the control volume to yield Eq. a6. This equation and the four network leg momentum equations are solved simultaneously each time step for the leg mass flowrates and the outlet header pressure. The inlet header pressure is a boundary condition.

A mixed implicit/explicit scheme is used to derive the thermal energy equations. A separate equation is derived for each cell of the network legs and for each header. There is also an energy equation for the secondary side of the heat exchanger. Equation a7 is the thermal energy equation for the second cell of the module leg shown.
in Fig. A-2. This cell is the downflow section of the heated module. Upwind differencing is used, and Eq. a7 applies to the case with positive flow. With negative flow the convective terms would be appropriately modified. The value of the coefficient “α” determines the degree of implicitness of the equation. The equation is fully implicit with “α” set equal to one and fully explicit with “α” set equal to zero. A value of one-half results in the Crank-Nicholson scheme. The superscript “n” denotes the new time level. Equation a8 is the expression for the updated nodal temperature. Note that only upstream temperatures at the new time level appear on the right-hand-side of the equation. Therefore, the temperatures in a specified leg of the network can be updated in a marching scheme starting at its upstream header.

\[
\dot{m}_1 + \dot{m}_2 + \dot{m}_3 - \dot{m}_{nx} = 0 \quad (a6)
\]

\[
(n_s + m)\overline{C_p}\frac{T_{n2}^n - T_{e2}^n}{\Delta t} = \dot{m}_s C_p [\alpha T_{n1}^n + (1 - \alpha) T_{e1}^n] - \dot{m}_s C_p [\alpha T_{n2}^n + (1 - \alpha) T_{e2}^n] + \dot{Q}_{n2}(t) \quad (a7)
\]

\[
T_{n2}^n = \frac{(n_s + m)\overline{C_p}}{\Delta t} T_{n2}^n + \dot{m}_s C_p [\alpha T_{n1}^n + (1 - \alpha) T_{e1}^n] - \dot{m}_s C_p (1 - \alpha) T_{e2}^n + \dot{Q}_{n2}(t) \quad (a8)
\]

where

- \( m \) - fluid mass
- \( m_s \) - solid (metal) mass
- \( C_p \) - fluid specific heat
\[ \overline{C}_p \] - mass weighted liquid/solid composite specific heat
\[ \dot{Q}_{a2}(t) \] - deposited decay power for cell n2
\[ \alpha \] - implicitness parameter
\[ \Delta t \] - time step size

Similar equations can be derived for the remaining cells of the module leg shown in Fig. A-2. The storage terms for the two heated cells represent both the masses of the lead/aluminum metal and the fluid in the channels. Their specific heats are composite values. There is also a time dependent internal heat generation source term in each of the two equations representing heated sections.

There are also thermal energy equations for the two headers and the secondary side of the heat exchanger. Equation a9 is the energy equation for the outlet header. The flows are assumed to be positive, as shown in Fig. A-4. If any of the leg flows are negative, the equation would have to be appropriately modified to use the upstream value of temperature. Equation a10 is the expression for the updated outlet header temperature. The inlet header thermal energy equation is derived in a similar fashion.

\[
\frac{mC_p}{\Delta t} \left( T_{out}^n - T_{out} \right) = \sum_{i=1,3} \dot{m}_i C_p \left[ \alpha T_{i,5}^n + (1 - \alpha) T_{i,5} \right] - \dot{m}_{Hx} C_p \left[ \alpha T_{out}^n + (1 - \alpha) T_{out} \right]
\] (a9)

\[
T_{out}^n = \frac{\left[ mC_p - \dot{m}_{Hx} C_p (1 - \alpha) \right] T_{out} + \sum_{i=1,3} \dot{m}_i C_p \left[ \alpha T_{i,5}^n + (1 - \alpha) T_{i,5} \right]}{\frac{mC_p}{\Delta t} + \dot{m}_{Hx} C_p \alpha}
\] (a10)

Figure A-5 is a schematic of the heat transfer between the primary and secondary sides of the heat exchanger. Equation a11 is the thermal energy equation for the tube side of the heat exchanger, and Eq. a12 is the energy equation for the shell side. The shell side of the heat exchanger is assumed to exchange energy only with the tube side. The outer surface of the shell is assumed to be an adiabatic boundary. Equation a13 is the expression for the heat transfer rate between the two sides of the heat exchanger. The heat transfer coefficient accounts for single-phase convective heat transfer on both sides of the heat exchanger and conduction through the tube walls. The mass and specific heat for the secondary side of the heat exchanger are composites of both the metal and the liquid.

\[
\frac{mC_p}{\Delta t} \left( T_{Hx,2}^n - T_{Hx,2} \right) = \dot{m}_{Hx} C_p \left[ \alpha (T_{Hx,1}^n - T_{Hx,2}^n) + (1 - \alpha) (T_{Hx,1} - T_{Hx,2}) \right] - Q_{conv}
\] (a11)

\[
\left( mC_p \right)_{sh} \frac{T_{sh}^n - T_{sh}}{\Delta t} = \dot{Q}_{conv}
\] (a12)

\[
\dot{Q}_{conv} = h_{conv} A_{Hx} \left[ \alpha T_{Hx,2}^n + (1 - \alpha) T_{Hx,2} - \alpha T_{sh}^n - (1 - \alpha) T_{sh} \right]
\] (a13)
Figure A-5  Schematic of the heat exchanger showing the heat transfer to the secondary side.
Appendix B: Buoyancy-Assisted and Buoyancy-Opposed Friction Factor Multipliers

To account for the influence of wall heating on the friction factors for flow through the vertical modules, results of an analytical study of fully developed laminar flow between vertical parallel plates were used (see Ref. [3]). A plot of the product of the friction factor and the Reynolds number versus the one-fourth power of a modified Rayleigh number was used to develop functional relations for mixed convection multipliers to friction factors. The modified Rayleigh number used is defined as

\[
Ra = \frac{\beta g \left( \frac{dT_w}{dx} \right) D^4_h \rho^2 Pr}{\mu^2}
\]

For fully developed flow with a constant wall heat flux, the wall and fluid axial temperature gradients are equal. The wall axial temperature gradient can, therefore, be expressed as a function of the wall power and the fluid mass flowrate (Eq. b2).

\[
\frac{\Delta T}{L} = \frac{Q_w}{L \ln C_p}
\]

Substituting Eq. b2 and the definition of the Prandtl number into the expression for the modified Rayleigh number results in Eq. b3, the form of the modified Rayleigh number that is used in the NC model of the primary HR system. The Rayleigh number is directly proportional to the deposited power in the fluid and inversely proportional to the mass flowrate. It expresses the relative importance of buoyancy in vertical mixed convection.

\[
Ra = \frac{g \beta Q_w D^4_h \rho^2}{m \mu L k}
\]

Reference 3 presents results for the range of Rayleigh numbers to the one-fourth power from zero to twelve. Results are presented for both buoyancy-assisted and buoyancy-opposed flows. For buoyancy-opposed flow, results are presented for the Rayleigh number to the one-fourth-power range up to four. Beyond this point the buoyancy-opposed mixed convection friction behavior is not defined. From the results in Ref. [3], Darcy friction factor multiplier relations were developed. These relations are Eqs. b-4 through b-8, and they are shown graphically by the solid lines in Fig. B-1. The curves are first-order continuous over the entire Rayleigh number to the one-fourth-power range shown. The curve for buoyancy-opposed flow at Rayleigh numbers to the one-fourth power greater than four is simply conjecture, and it is included in the model to insure that a reasonable value of the friction factor multiplier is returned in all cases. Having first-order continuous constitutive relations in the model is desirable from a numerical perspective.

\[
\phi = \frac{f_{\text{Mixed}}}{f_{L_\text{aminar}}} = 1.0 \quad \text{for} \quad Ra^{1/4} \leq 1.5
\]

Buoyancy-Assisted Flow:
Buoyancy-Opposed Flow:

\[
\phi = \begin{cases} 
\exp \left[ \left( C_3 \left( \frac{\text{Ra}}{1.5} \right)^{1.5} + C_4 \left( \frac{\text{Ra}}{1.5} \right)^{1.5} \right) \right] & \text{for } 1.5 < \frac{\text{Ra}}{1.5} \leq 8.0 \\
0.0416 & \text{for } \frac{\text{Ra}}{1.5} > 8.0 
\end{cases}
\]  

(b7 & b8)

where:

\[
\begin{align*}
C_1 &= 0.11815501 \\
C_2 &= -2.9378955 \times 10^{-2} \\
C_3 &= -3.3784101 \times 10^1 \\
C_4 &= 0.11636369 \times 10^1 \\
C_5 &= -3.8259936 \\
C_6 &= 0.71433520 \times 10^{-1} \\
C_7 &= -3.7145261 \times 10^{-2}
\end{align*}
\]
The Rayleigh number is inversely proportional to the mass flowrate. If the Rayleigh number to the one-fourth power were not limited to a maximum value of twelve in the NC model, it could grow without bound during flow reversals. Even with this restriction on the value of the parameter, there will be a large discontinuity in the values of the friction factor multipliers as the flow direction reverses in a heated vertical passage. To prevent this discontinuity, Eq. b-9 modifies the friction factor multiplier when the Reynolds number is less than twenty-five. This equation interpolates non-linearly between the buoyancy-assisted and buoyancy-opposed values of the friction factor multiplier at a Reynolds number of twenty-five and a value of unity when the Reynolds number is zero. It can easily be seen that, with Eq. b9, at a Reynolds number of twenty-five the friction factor multiplier is unaltered, and at zero the friction factor multiplier is unity. This equation insures that the friction factor and its first derivative are continuous as the flow reverses. The dotted lines in Fig. B-1 show the values of the friction factor multiplier at a Reynolds number of one.

\[
\phi_{(Re=25)} = 1 - \left[ 3 - 2 \left( \frac{Re}{25} \right) \left( \frac{Re}{25} \right)^2 \right] \phi + \left[ 3 - 2 \left( \frac{Re}{25} \right) \left( \frac{Re}{25} \right)^2 \right] \phi
\]  

(b9)
Appendix C: Sample Input Deck

A sample-input file for the NCLOFA code (see Appendix D) is provided below. The input file applies to the LOFA in which both the target and blanket HR system pumps are lost and the target power is deposited into the Module 1 Decoupler 24 hours after beam shutdown. The input is structured such that data unique to a specific leg is grouped together and comment lines are provided describing the input variables.

Input file data.in

NCLOFA input file:

This is the input deck for a natural convection model of the APT target/blanket HR cooling system. Three module loops are modeled and the outer loop with a heat exchanger. The secondary side of the heat exchanger is a heat sink. Units are SI.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tstart</td>
<td>start time in hours</td>
</tr>
<tr>
<td>tend</td>
<td>end time in hours</td>
</tr>
<tr>
<td>dt</td>
<td>time step in seconds</td>
</tr>
<tr>
<td>nplot</td>
<td>frequency for plotting</td>
</tr>
<tr>
<td>nsolve</td>
<td>frequency for updating mass/momentum calculations</td>
</tr>
<tr>
<td>itrans</td>
<td>transient mass/momentum eqns (1-yes; 0-no)</td>
</tr>
<tr>
<td>tgstrt</td>
<td>start time for target power to blanket in hours</td>
</tr>
<tr>
<td>tgramp</td>
<td>ramp time to full target power deposition in hours</td>
</tr>
<tr>
<td>target</td>
<td>initial deposited power for entire target ladders</td>
</tr>
<tr>
<td>alpha</td>
<td>implicitness parameter for time differencing</td>
</tr>
<tr>
<td>dfdtmx</td>
<td>multiplier to reduce the size of the timestep during the flow reversal: ( \Delta t(t) = \text{dfdtmx} \cdot \Delta t )</td>
</tr>
<tr>
<td>eps_t</td>
<td>convergence tolerance for energy eqn solver</td>
</tr>
<tr>
<td>maxit_t</td>
<td>maximum number of iterations for energy eqn solver</td>
</tr>
<tr>
<td>eps_m</td>
<td>convergence tolerance for mass/momentum eqn solver</td>
</tr>
<tr>
<td>maxit_m</td>
<td>maximum # of iterations for mass/momentum eqn solver</td>
</tr>
</tbody>
</table>

```
tstart  tend  dt
1.0     70.0  10.0
nplot  nsolve  itrans
100     1     1
tgstrt  tgramp  target 68.01e6
24.0    1.0    68.01e6
alpha  dfdtmx
.5      0.005
eps_t  maxit_t  eps_m  maxit_m
0.001   100    0.001   500
```

module #1 loop from the inlet header to the outlet header (5 sections):

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aml(1-5)</td>
<td>the 5 flow areas</td>
</tr>
<tr>
<td>dzl(1-5)</td>
<td>the 5 delta elevations</td>
</tr>
<tr>
<td>zll(1-5)</td>
<td>the 5 lengths</td>
</tr>
<tr>
<td>dhl(1-5)</td>
<td>the 5 hydraulic diameters</td>
</tr>
<tr>
<td>frmkal</td>
<td>form loss term (K/area)</td>
</tr>
<tr>
<td>sml2,sm14</td>
<td>metal masses in heat structures</td>
</tr>
<tr>
<td>cpl2,cpl4</td>
<td>specific heats of heat structures</td>
</tr>
<tr>
<td>rough</td>
<td>pipe roughness</td>
</tr>
<tr>
<td>am11</td>
<td>am12</td>
</tr>
<tr>
<td>------</td>
<td>------</td>
</tr>
<tr>
<td>0.0609</td>
<td>0.199</td>
</tr>
<tr>
<td>dz11</td>
<td>dz12</td>
</tr>
<tr>
<td>6.1506</td>
<td>2.7582</td>
</tr>
<tr>
<td>z111</td>
<td>z112</td>
</tr>
<tr>
<td>10.77</td>
<td>2.7582</td>
</tr>
<tr>
<td>dh11</td>
<td>dh12</td>
</tr>
<tr>
<td>0.1905</td>
<td>0.00861</td>
</tr>
</tbody>
</table>

Page 2 of 4

module #2 loop from the inlet header to the outlet header (5 sections):

<table>
<thead>
<tr>
<th>am21</th>
<th>am22</th>
<th>am23</th>
<th>am24</th>
<th>am25</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.08574</td>
<td>0.34647</td>
<td>6.5617</td>
<td>0.34647</td>
<td>0.08574</td>
</tr>
<tr>
<td>dz21</td>
<td>dz22</td>
<td>dz23</td>
<td>dz24</td>
<td>dz25</td>
</tr>
<tr>
<td>6.1506</td>
<td>3.5</td>
<td>0.0</td>
<td>-3.5</td>
<td>-5.3866</td>
</tr>
<tr>
<td>z121</td>
<td>z122</td>
<td>z123</td>
<td>z124</td>
<td>z125</td>
</tr>
<tr>
<td>10.77</td>
<td>3.5</td>
<td>0.3048</td>
<td>3.5</td>
<td>10.0152</td>
</tr>
<tr>
<td>dh21</td>
<td>dh22</td>
<td>dh23</td>
<td>dh24</td>
<td>dh25</td>
</tr>
<tr>
<td>0.1524</td>
<td>0.00311</td>
<td>0.289</td>
<td>0.00311</td>
<td>0.1524</td>
</tr>
</tbody>
</table>

heat exchanger loop (inlet to outlet headers 3 sections):

<table>
<thead>
<tr>
<th>ahx1</th>
<th>ahx2</th>
<th>ahx3</th>
<th>ahx4</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.09843</td>
<td>0.09526</td>
<td>0.289</td>
<td>0.09526</td>
</tr>
</tbody>
</table>

Page 2 of 4
**WE~TINGHOUSE SAVANNAH RIVER COMPANY**

**BLANKET SAFETY ANALYSIS FOR LOFA**

*APPENDIX C*

**08/06/98**

---

<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>form loss term (K/area)</td>
<td>$f_{rmkax}$</td>
</tr>
<tr>
<td>mass of HX secondary shell side</td>
<td>$w_{mssh}$</td>
</tr>
<tr>
<td>heat transfer area of HX secondary shell side</td>
<td>$a_{rsh}$</td>
</tr>
<tr>
<td>convection $h_T$ coef. of HX secondary shell side</td>
<td>$c_{convh}$</td>
</tr>
<tr>
<td>specific heat of HX secondary shell side</td>
<td>$c_{psh}$</td>
</tr>
</tbody>
</table>

---

<table>
<thead>
<tr>
<th>ahx1</th>
<th>ahx2</th>
<th>ahx3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.17936</td>
<td>1.13116</td>
<td>0.17936</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>dzhx1</th>
<th>dzhx2</th>
<th>dzhx3</th>
</tr>
</thead>
<tbody>
<tr>
<td>-3.7859</td>
<td>0.0</td>
<td>3.0239</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>zlhx1</th>
<th>zlhx2</th>
<th>zlhx3</th>
</tr>
</thead>
<tbody>
<tr>
<td>52.885</td>
<td>7.92</td>
<td>36.122</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>dhhx1</th>
<th>dhhx2</th>
<th>dhhx3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.47788</td>
<td>0.01021</td>
<td>0.47788</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$w_{mssh}$</th>
<th>$a_{rsh}$</th>
<th>$c_{convh}$</th>
<th>$c_{psh}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>50852.0</td>
<td>3510.0</td>
<td>5.0</td>
<td>4009.0</td>
</tr>
</tbody>
</table>

---

**inlet and outlet headers:**

<table>
<thead>
<tr>
<th>volin</th>
<th>volume of inlet header</th>
</tr>
</thead>
<tbody>
<tr>
<td>Volout</td>
<td>volume of outlet header</td>
</tr>
<tr>
<td>pin</td>
<td>inlet header pressure</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>volin</th>
<th>Volout</th>
<th>pin</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.31727</td>
<td>0.31727</td>
<td>7.325e5</td>
</tr>
</tbody>
</table>

---

**fluid properties:**

<table>
<thead>
<tr>
<th>rho</th>
<th>density</th>
</tr>
</thead>
<tbody>
<tr>
<td>cp</td>
<td>specific heat</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>rho</th>
<th>cp</th>
</tr>
</thead>
<tbody>
<tr>
<td>987.0</td>
<td>4184.3</td>
</tr>
</tbody>
</table>

---

**initial guess at temperatures and flowrates:**

<table>
<thead>
<tr>
<th>tin, tout</th>
<th>inlet and outlet header temperatures</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_{rm1}$, $t_{rm2}$, $t_{rm3}$, $t_{rm4}$, $t_{rm5}$</td>
<td>temperatures in module #1</td>
</tr>
<tr>
<td>$t_{rm2}$, $t_{rm3}$</td>
<td>temperatures in module #2</td>
</tr>
<tr>
<td>$t_{rm3}$, $t_{rm4}$</td>
<td>temperatures in module #3</td>
</tr>
<tr>
<td>$t_{rmh}$, $t_{rmx}$</td>
<td>temperatures in the HX loop</td>
</tr>
<tr>
<td>$t_{thx}$</td>
<td>HX secondary shell side temperature</td>
</tr>
<tr>
<td>$f_{flw1}$-$f_{flw3}$</td>
<td>mass flowrates through the three modules</td>
</tr>
<tr>
<td>$f_{flwhx}$</td>
<td>mass flowrate through the HX loop</td>
</tr>
<tr>
<td>$q_{22q}$, $q_{24q}$</td>
<td>initial deposited powers in module #2</td>
</tr>
<tr>
<td>$q_{32q}$, $q_{34q}$</td>
<td>initial deposited powers in module #3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>tin</th>
<th>tout</th>
</tr>
</thead>
<tbody>
<tr>
<td>322.0</td>
<td>325.5</td>
</tr>
<tr>
<td>tm1</td>
<td>tm2</td>
</tr>
<tr>
<td>322.0</td>
<td>322.5</td>
</tr>
<tr>
<td>tm21</td>
<td>tm22</td>
</tr>
<tr>
<td>322.0</td>
<td>323.5</td>
</tr>
<tr>
<td>tm3</td>
<td>tm34</td>
</tr>
<tr>
<td>322.0</td>
<td>323.5</td>
</tr>
<tr>
<td>thx1</td>
<td>thx2</td>
</tr>
<tr>
<td>325.5</td>
<td>324.0</td>
</tr>
<tr>
<td>flw1</td>
<td>flw2</td>
</tr>
<tr>
<td>5.440</td>
<td>4.926</td>
</tr>
</tbody>
</table>
### Calibration Data

<table>
<thead>
<tr>
<th>icalib</th>
<th>pincal</th>
<th>poutcal</th>
<th>pmpcal</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>7.325e5</td>
<td>4.5671e5</td>
<td>7.6048e5</td>
</tr>
</tbody>
</table>

*Initial guess (kg/s):*  
- FLW1: 544.0  
- FLW2: 615.0  
- FLW3: 410.0  
- FLW4: 1569.0

<table>
<thead>
<tr>
<th>q12</th>
<th>q14</th>
<th>q22</th>
<th>q24</th>
<th>q32</th>
<th>q34</th>
</tr>
</thead>
<tbody>
<tr>
<td>8.222e6</td>
<td>15.768e6</td>
<td>9.083e6</td>
<td>17.695e6</td>
<td>0.0e6</td>
<td>5.712e6</td>
</tr>
</tbody>
</table>

---

### FLW Calibration Data

<table>
<thead>
<tr>
<th>icalib</th>
<th>pincal</th>
<th>poutcal</th>
<th>pmpcal</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>7.325e5</td>
<td>4.5671e5</td>
<td>7.6048e5</td>
</tr>
</tbody>
</table>

*Initial guess (kg/s):*  
- FLW1: 5.440  
- FLW2: 4.926  
- FLW3: 1.965  
- FLW4: 12.331

<table>
<thead>
<tr>
<th>q12</th>
<th>q14</th>
<th>q22</th>
<th>q24</th>
<th>q32</th>
<th>q34</th>
</tr>
</thead>
<tbody>
<tr>
<td>8.222e6</td>
<td>15.768e6</td>
<td>9.083e6</td>
<td>17.695e6</td>
<td>0.0e6</td>
<td>5.712e6</td>
</tr>
</tbody>
</table>
Appendix D: Code Listing

A code listing of the fortran program NCLOFA and its associated fortran subroutines is presented below. A algorithm description is provided at the front of each routine. The following is a listing of the routines:

- nclofa.f — main program that reads in the input and writes out the output.
- momtm.f — performs Newton iterations for momentum equations.
- momloop.f — generates each leg's momentum equation and Jacobian.
- rayleigh.f — computes Rayleigh number and heating orientation.
- frict.f — computes multiplier and friction factor.
- outhdr.f — generates header Jacobian terms.
- tmean.f — computes mean fluid temperatures.
- cond1q.f — computes fluid thermal conductivity.
- dpower.f — computes deposited decay power levels.
- ludcmp.f — generates LUD decomposition of linearized equation set.
- lubsks.f — performs forward/backward passes of LUD process.

Main program nclofa.f

program nclofa

*************** *************** **************** **************** *****
* *
* PROGRAM nclofa *
* *
* THIS PROGRAM COMPUTES NATURAL CIRCULATION DURING A LOFA WHERE *
* 3 COMPOSITE MODULES ARE EMPLOYED ALONG WITH A HEAT EXCHANGER *
* LOOP. *
* *
*************** *************** **************** **************** *****

implicit real*8 (a-h,o-z), integer (i-n)
include 'param.inc'
parameter (g=9.8066d0)

c
character*80 line1
c
character esc
c
dimension flw(mxloop), flwcal(mxnode), flwpred(mxnode)
dimension flwn(mxloop)
dimension ncell(mxloop), ntype(mxloop)
c
dimension zlt(mxloop)
dimension tml(mxnode), am1(mxnode), dh1(mxnode), dz1(mxnode),
& z11(mxnode), tm1n(mxnode), wms1(mxnode), q1(mxnode),
& sm1(mxnode), cp1(mxnode)
dimension tm2(mxnode), am2(mxnode), dh2(mxnode), dz2(mxnode),
& zl2(mxnode), tm2n(mxnode), wms2(mxnode), q2(mxnode),
& sm2(mxnode), cp2(mxnode)
dimension tm3(mxnode), am3(mxnode), dh3(mxnode), dz3(mxnode),
& zl3(mxnode), tm3n(mxnode), wms3(mxnode), q3(mxnode),
& sm3(mxnode), cp3(mxnode)
dimension thx(mxnode), ahx(mxnode), dhhx(mxnode), dzhx(mxnode),
& zlxh(mxnode), thxn(mxnode), wmmxh(mxnode)
dimension qd1(mxnode), qd2(mxnode), qd3(mxnode), qdhx(mxnode),
& fold1(mxnode),fold2(mxnode), fold3(mxnode),foldhx(mxnode),
& fnew1(mxnode),fnew2(mxnode), fnew3(mxnode),fnewhx(mxnode)
c
data ncell/5,5,5,3/
data ntype/1,1,1,2/
c c === SET I/O UNITS AND OPEN FILES ===============================
c
c
iin1 = 5
iout1 = 7
iout2 = 8
iout3 = 17
iout4 = 19
iout5 = 22
iout6 = 23
iout7 = 26
itec = 12
itec1 = 13
itec2 = 14
itec3 = 15
itec4 = 16
conv = 273.15d0
convp = 14.696d0/101325.0d0
conv_tim = 3600.0d0
c
INITIALIZE TIME COUNTERS
c
esc = char(27)
write (*,1O) esc
c open files:
c
open (unit=iin1 , file='data.in', status='old')
open (unit=iout1, file='results.out', status='unknown')
open (unit=iout2, file='cal.out', status='unknown')
open (unit=iout3, file='rayl.out', status='unknown')
open (unit=iout4, file='rayfrct.out', status='unknown')
open (unit=iout5, file='jacob.out', status='unknown')
open (unit=iout6, file='convg.out', status='unknown')
open (unit=iout7, file='restrt.out', status='unknown')
open (unit=itec, file='tec.dat', status='unknown')
open (unit=itec1, file='tec1.dat', status='unknown')
open (unit=itec2, file='tec2.dat', status='unknown')
open (unit=itec3, file='tec3.dat', status='unknown')
open (unit=itec4, file='tec4.dat', status='unknown')
c
START OF INPUT PROCESSING ===============================
c
Read in the input:
c
Print input data for 1-D 3-module model.
write (iout1,1000)
write (iout1,901)
write (iout1,1000)
c --- CONTROL PARAMETER INPUT: ------------------------------------------
c
do i=1,26
   read (iin1,900) linel
   write (iout1,900) linel
endo

c TIME STEP AND RUN TIME:
   read (iin1,900) linel
   write (iout1,900) linel
   read (iin1,*) tstart, tend, dt
   write (iout1,*) tstart, tend, dt

c FREQUENCIES FOR PLOTTING AND MASS/MOM SOLVING:
   read (iin1,900) linel
   write (iout1,900) linel
   read (iin1,*) nplot, nsolve, itrans
   write (iout1,*) nplot, nsolve, itrans

c TARGET POWER TIMING:
   read (iin1,900) linel
   write (iout1,900) linel
   read (iin1,*) tgstrt, tgramp, target
   write (iout1,*) tgstrt, tgramp, target

c IMPLICITNESS PARAMETER FOR TIME DIFFERENCING & FF RATE CHANGE:
   read (iin1,900) linel
   write (iout1,900) linel
   read (iin1,*) alpha, dfdtmx
   write (iout1,*) alpha, dfdtmx

c CHECK (AND LIMIT) RANGE OF ALPHA
   alpha = dmax1(alpha, zero)
   alpha = dmin1(alpha, one)

c ENERGY AND MASS/MOMENTUM SOLVERS CONVERGENCE TOLERANCES AND
MAXIMUM NUMBER OF ITERATIONS ALLOWED:
   read (iin1,900) linel
   write (iout1,900) linel
   read (iin1,*) eps_t, maxit_t, eps_m, maxit_m
   write (iout1,*) eps_t, maxit_t, eps_m, maxit_m

c --- MODULE #1 INPUT (VERTICAL LATERAL MODULES): ----------------------
c
do i=1,ll
   read (iin1,900) linel
   write (iout1,900) linel
endo

c read (iin1,900) linel
write (iout1,900) linel
read (iin1,*) (aml(i), i=1,ncell(l))
write (iout1,*) (aml(i), i=1,ncell(l))
c read (iin1,900) linel
write (iout1,900) linel
read (iin1,*) (dzl(i), i=1,ncell(l))
write (iout1,*) (dz1(i), i=1,ncell(1))
read (iiin1,900) line1
write (iout1,900) line1
read (iiin1,*) (zl1(i), i=1,ncell(1))
write (iout1,*) (zl1(i), i=1,ncell(1))
read (iiin1,900) line1
write (iout1,900) line1
read (iiin1,*) (dh1(i), i=1,ncell(1)), frmkal
write (iout1,*) (dh1(i), i=1,ncell(1)), frmkal
read (iiin1,900) line1
write (iout1,900) line1
read (iiin1,*) sm12,cp12, sm14,cp14,rough
write (iout1,*) sm12,cp12, sm14,cp14,rough

MODULE #2 INPUT (ALL REMAINING VERTICAL MODULES):

do i=1,10
   read (iiin1,900) line1
   write (iout1,900) line1
enddo

read (iiin1,900) line1
write (iout1,900) line1
read (iiin1,*) (am2(i), i=1,ncell(2))
write (iout1,*) (am2(i), i=1,ncell(2))
read (iiin1,900) line1
write (iout1,900) line1
read (iiin1,*) (dz2(i), i=1,ncell(2))
write (iout1,*) (dz2(i), i=1,ncell(2))
read (iiin1,900) line1
write (iout1,900) line1
read (iiin1,*) (z12(i), i=1,ncell(2))
write (iout1,*) (z12(i), i=1,ncell(2))
read (iiin1,900) line1
write (iout1,900) line1
read (iiin1,*) (dh2(i), i=1,ncell(2)), frmka2
write (iout1,*) (dh2(i), i=1,ncell(2)), frmka2
read (iiin1,900) line1
write (iout1,900) line1
read (iiin1,*) sm22,cp22, sm24,cp24
write (iout1,*) sm22,cp22, sm24,cp24

MODULE #3 INPUT (ALL HORIZONTAL MODULES):

do i=1,10
   read (iiin1,900) line1
   write (iout1,900) line1
enddo

read (iiin1,900) line1
write (iout1,900) line1
read (iiin1,*) (am3(i), i=1,ncell(3))
write (iout1,*) (am3(i), i=1,ncell(3))
read (iiin1,900) line1
write (iout1,900) line1
read (iiin1,*) (dz3(i), i=1,ncell(3))
write (iout1,*) (dz3(i), i=1,ncell(3))
read (iinl,900) linel
write (iout1,900) linel
read (iinl,*), (z13(i), i=1,ncell(3))
write (iout1,*), (z13(i), i=1,ncell(3))
read (iinl,900) linel
write (iout1,900) linel
read (iinl,*), (dh3(i), i=1,ncell(3)), frmka3
write (iout1,*), (dh3(i), i=1,ncell(3)), frmka3
read (iinl,900) linel
write (iout1,900) linel
read (iinl,*), sm32,cp32,sm34,cp34
write (iout1,*), sm32,cp32,sm34,cp34

--- HEAT EXCHANGER LOOP INPUT #4: -------------------------------

do i=1,12
read (iin1,900) linel
write (iout1,900) linel
enddo

read (iinl,900) linel
write (iout1,900) linel
read (iinl,*), (ahx(i), i=1,ncell(4))
write (iout1,*), (ahx(i), i=1,ncell(4))
read (iinl,900) linel
write (iout1,900) linel
read (iinl,*), (dzhx(i), i=1,ncell(4))
write (iout1,*), (dzhx(i), i=1,ncell(4))
read (iinl,900) linel
write (iout1,900) linel
read (iinl,*), (zlhx(i), i=1,ncell(4))
write (iout1,*), (zlhx(i), i=1,ncell(4))
read (iinl,900) linel
write (iout1,900) linel
read (iinl,*), (dhhx(i), i=1,ncell(4)), frmkax
write (iout1,*), (dhhx(i), i=1,ncell(4)), frmkax
read (iinl,900) linel
write (iout1,900) linel
read (iinl,*), wmasshl,arshl,convh,cpshl
write (iout1,*), wmasshl,arshl,convh,cpshl

--- FIXED HEADER INPUT: ----------------------------------------

do i=1,6
read (iin1,900) linel
write (iout1,900) linel
enddo

read (iin1,900) linel
write (iout1,900) linel
read (iinl,*), volin,volout,pin
write (iout1,*), volin,volout,pin

--- FLUID PROPERTIES INPUT: -------------------------------------

do i=1,5
read (iin1,900) linel

write (iout1,900) linel  
enddo  
c  
read (iinl,900) linel  
write (iout1,900) linel  
read (iinl,*) rho,cp  
write (iout1,*) rho,cp  

c --- INITIAL CONDITIONS (TEMPERATURES AND FLOWRATES) INPUT:  

do i=1,14  
read (iinl,900) linel  
write (iout1,900) linel  
enddo  
c  
FLUID TEMPERATURES:  
read (iinl,900) linel  
write (iout1,900) linel  
read (iinl,*) tin,tout  
write (iout1,*) tin,tout  
read (iinl,900) linel  
write (iout1,900) linel  
read (iinl,*) (tmi(i), i=1,ncell(1))  
write (iout1,*) (tmi(i), i=1,ncell(1))  
read (iinl,900) linel  
write (iout1,900) linel  
read (iinl,*) (tm2(i), i=1,ncell(2))  
write (iout1,*) (tm2(i), i=1,ncell(2))  
read (iinl,900) linel  
write (iout1,900) linel  
read (iinl,*) (tm3(i), i=1,ncell(3))  
write (iout1,*) (tm3(i), i=1,ncell(3))  
read (iinl,900) linel  
write (iout1,900) linel  
read (iinl,*) (thx(i), i=1,ncell(4)),tshl  
write (iout1,*) (thx(i), i=1,ncell(4)),tshl  

Mass Flow Trans:  
read (iinl,900) linel  
write (iout1,900) linel  
read (iinl,*) (flw(i), i=1,mxloop)  
write (iout1,*) (flw(i), i=1,mxloop)  

DEPOSITED POWERS:  
read (iinl,900) linel  
write (iout1,900) linel  
read (iinl,*) q12,q14,q22,q24,q32,q34  
write (iout1,*) q12,q14,q22,q24,q32,q34  

--- CALIBRATION DATA INPUT:  

do i=1,7  
read (iinl,900) linel  
write (iout1,900) linel  
enddo  

HEADER PRESSURES:
read (iin1,900) linel
write (iout1,900) linel
read (iin1,*), icalib,pincal,poutcal,pmpcal
write (iout1,*), icalib,pincal,poutcal,pmpcal

**c**
**c** INITIAL GUESS FOR MASS FLOWRATES:
read (iin1,900) linel
write (iout1,900) linel
read (iin1,*), (flwcal(i), i=1,mxloop)
write (iout1,*), (flwcal(i), i=1,mxloop)

**c**
**c** === END OF INPUT PROCESSING

==
**c**
**c** === PERFORM INITIAL CALCULATIONS

==
**c**
print the header for output file.
write (iout1,1000)
write (iout1,902)
write (iout1,1000)

**c**
**c** COMPUTE FLUID CELL MASSES FOR EACH LOOP, HEADERS, AND SHELL.
**c** ALSO COMPUTE TOTAL LENGTH OF EACH LOOP.

**c**
**c** do j=1,mxloop
zlt(j) = zero
enddo

**c**
**c** do i=1,ncell(1)
 wms1(i) = rho*aml(i)*zll(i)
 zlt(1) = zlt(1) + zll(i)
 enddo

**c**
**c** do i=1,ncell(2)
 wms2(i) = rho*am2(i)*z12(i)
 zlt(2) = zlt(2) + z12(i)
 enddo

**c**
**c** do i=1,ncell(3)
 wms3(i) = rho*am3(i)*z13(i)
 zlt(3) = zlt(3) + z13(i)
 enddo

**c**
**c** do i=1,ncell(4)
 wmsihx(i) = rho*ahx(i)*zlhx(i)
 zlt(4) = zlt(4) + zlhx(i)
 enddo

**c**
**c** wmsin = rho*volin
wmsout = rho*volout
pout = pin + 7360.5d0

**c**
**c** SET SOLID CELL MASSES AND SPECIFIC HEATS FOR EACH MODULE LOOP:
**c**
**c** do i=1,ncell(1)
 sml(i) = zero
cpi(i) = zero
enddo
sml(2) = sml2
sml(4) = sml4
cpl(2) = cpl2
cpl(4) = cpl4
do i=1,ncell(2)
  sm2(i) = zero
cp2(i) = zero
endo
sm2(2) = sm22
sm2(4) = sm24
cp2(2) = cp22
cp2(4) = cp24
do i=1,ncell(3)
  sm3(i) = zero
cp3(i) = zero
endo
sm3(2) = sm32
sm3(4) = sm34
cp3(2) = cp32
cp3(4) = cp34

COMPUTE INITIAL AVERAGE FLUID TEMPERATURE, BETA, AND VISCOITY:
iskip = 0
call tmean (wmsin,wmsout,wmsl,wnw2 ,wms3,wmshx,
  & tin,tout,tml,tm2,tm3,thx,tnot,beta,vis,
  & iskip,ncell)

COMPUTE FLUID THERMAL CONDUCTIVITY:
ifld = 1
call condlq (tnot,ifld,condl,dummy)

INITIAL TARGET POWER PARAMETER SETTINGS AND TIMING POINTS:
tstart = conv_tim*tstart
tendhr = tend
tend = conv_tim*tend
ccc nstep = int((tend-tstart)/dt) + 1
tgstrt = conv_tim*tgstrt
tgtramp = conv_tim*tgramp
tgdry = tgstrt + tgramp
nstep = int(((tend-tstart-2.5*tgramp)/dt)+
  & (2.5*tgramp)/(dfdtmx*dt))+1
eps_p = 1.0d-3
do i=1,ncell(1)
  ql(i) = zero
endo
ql(2) = q12
ql(4) = q14
do i=1,ncell(2)
q2(i) = zero
enddo
q2(2) = q22
q2(4) = q24
c
do i=1,ncell(3)
q3(i) = zero
enddo
q3(2) = q32
q3(4) = q34
c
INITIAL DECAY POWER LEVELS:
c
eltim_p = tstart + eps_p
call dpower (eltim_p,tgstrt,tgdry,
& target,q1,q2,q3,qd1,qd2,qd3,qdhx,
& ncell)
c
Stuff the old friction factor arrays for the loops:
c
do i=1,ncell(1)
foldl(i) = 64.0/100.0
enddo
do i=1,ncell(2)
fold2(i) = 64.0/100.0
enddo
do i=1,ncell(3)
fold3(i) = 64.0/100.0
enddo
do i=1,ncell(4)
foldhx(i) = 64.0/100.0
enddo
c
COMPUTE INITIAL FLOW STATE (IF TRANSIENT TERMS NOT USED):
c
tmhr = tstart/conv_tim
cwrite (*,1015) tmhr
cwrite (*,20) esc, tmhr, tendhr
cc if (itrans.eq.0) then
c
eltim = 3600.0
call momtm (rho,vis,g,beta,rough,zlt,dt,condl,eltim,
& qd1,qd2,qd3,qdhx,
& foldl,fold2,fold3,foldhx,dfdtmx,
& fnew1,fnew2,fnew3,fnewhx,
& aml,dz1 ,z1l ,dh1 ,fri,ka1,tml,
& am2,dz2 ,z12 ,dh2 ,fri,ka2,tm2,
& am3,dz3 ,z13 ,dh3 ,frik,a3,tm3,
& ahx,dxhx,zlhx,dhx,frmkax,thx,
& pin,pout,flw,flwn,tnot,ncell,ntype,
& eps_m,maxit_m,itrans,icnvg)
c
SET OLD TIME MASS FLOWRATES TO NEW TIME VALUES
do j=1,mxloop
flw(j) = flwn(j)
enddo
cc
cc

== WRITE INITIAL CONDITIONS TO TECPLT FILES ==

**COMPOSITE FILE:**
```
write (itec,210)
  write (itec,'(17d12.4)') tstart,conv_tim,flw(1),flw(2),flw(3),
  & flw(4), tin-convt, tout-convt, thx(2)-convt,
  & tshl-convt, tml(2)-convt, tml(4)-convt, tm2(2)-convt,
  & tm2(4)-convt, tm3(2)-convt, tm3(4)-convt, tnot-convt,
  & pout*convp
```

**MODULE #1 FILE:**
```
write (itecl,211)
  write (itecl,'(8d12.4)')  tstart,conv_tim,tin-convt,
  & tml(1)-convt, tml(2)-convt, tml(3)-convt,
  & tml(4)-convt, tml(5)-convt, tout-convt
```

**MODULE #2 FILE:**
```
write (itec2,212)
  write (itec2,'(8d12.4)') tstart,conv_tim,tin-convt,
  & tm2(1)-convt, tm2(2)-convt, tm2(3)-convt,
  & tm2(4)-convt, tm2(5)-convt, tout-convt
```

**MODULE #3 FILE:**
```
write (itec3,213)
  write (itec3,'(8d12.4)') tstart,conv_tim,tin-convt,
  & tm3(1)-convt, tm3(2)-convt, tm3(3)-convt,
  & tm3(4)-convt, tm3(5)-convt, tout-convt
```

**HEAT EXCHANGER #4 FILE:**
```
write (itec4,214)
  write (itec4,'(7d12.4)') tstart, conv_tim, tout-convt,
  & thx(1)-convt, thx(2)-convt, thx(3)-convt,
  & tin-convt, tshl-convt
```

== PERFORM CALIBRATION CHECKING ==

**COMPUTE LOOP FLOWRATES BASED ON INPUTTED HEADER PRESSURE DROP**
**AND WRITE RESULTS TO OUTPUT FILE.**
```
if (icalib.eq.1) then
  do i=1,mxloop
    flwpred(i) = flwcal(i)
  enddo
 CALL flwloop (rho,vis,g,beta,rough,zlt,dt,
   & am1,dz1 ,z11 ,dh1 ,frmk1,
   & am2,dz2 ,z12 ,dh2 ,frmk2,
   & am3,dz3 ,z13 ,dh3 ,frmk3,
   & ahx,dzdx,zlhx,dhdx,frmkax,
   & pincal,poutcal,pmpcal,flwpred,tnot,
   & ncell,ntype,eps_m,maxit_m)
  dp = pincal - poutcal
  dp = dp*convp
  dpmp = pmpcal*convp
  write (iout2,2000) dp, dpmp
  do i=1,mxloop
    write (iout2,'(i5,10x,2f14.5)') i,flwcal(i),flwpred(i)
```

PERFORM TRANSIENT CALCULATIONS

dtshort = dfdtmx*dt
dtreg = dt
tshst = tgstrt+.75*tgramp
tshstp = tshst+2.5*tgramp
eltim = tstart
do it=1,nstep
  if ((eltim .ge. tshst) .and. (eltim .lt. tshstp)) then
    dt = dtshort
  else
    dt = dtreg
  end if
  eltim = eltim + half*dt
  eltim = eltim
enddo

COMPUTE BLANKET/TARGET POWER DEPOSITED TO FLUID CELLS

call dpower (eltim_p, tgstrt, tgdry, 
  & target, q1, q2, q3, qd1, qd2, qd3, qdhx, 
  & ncell)

COMPUTE NEW TIME (n) FLUID TEMPERATURES:

call energy (flw, tin, tout, tml, tm2, tm3, thx, tshl, dt, 
  & wms1, wms2, wms3, wmshx, wmsshl, wmsin, wmsout, 
  & sm1, sm2, sm3, cp, cp1, cp2, cp3, cpshl, 
  & qd1, qd2, qd3, qdhx, convh, arshl, 
  & tinn, toutn, tmln, tm2n, tm3n, thxn, tshln, 
  & ncell, alpha, eps_t, maxit_t)

UPDATE TEMPERATURE ARRAYS:

do i=1,5
  tml(i) = tmln(i)
  tm2(i) = tm2n(i)
  tm3(i) = tm3n(i)
endo

do i=1,3
  thx(i) = thxn(i)
endo

tshl = tshln
tin = tinn
tout = toutn

UPDATE MEAN TEMPERATURE (LOOP AVERAGE) AND BETA:

iskip = 1
  call tmean (wmsin, wmsout, wms1, wms2, wms3, wmshx, 
    & tin, tout, tml, tm2, tm3, thx, tnot, beta, vis, 
    & iskip, ncell)
c COMPUTE FLUID THERMAL CONDUCTIVITY:
c
ifld = 1
  call condlq (tnot, ifld, condl, dummy)
c
COMPUTE NEW TIME \( n \) MASS FLOWRATES WITHIN LOOPS:
c
(only update flowrates/pressure every nsolve time steps.)
c
if (mod(it, nplot).eq.0) then
  tmnr = eltim/conv_tim
  write (iout5,1001)
  write (iout5,1015) tmnr
  write (*,20) esc, tmnr, tendhr
endif
if (mod(it, nsolve).eq.0) then
  call momtm (rho, vis, g, beta, rough, zlt, dt, condl, eltim,
&  qdl, qd2, qd3, qdhh,
&  foldl, fold2, fold3, foldhh, dfdtmx,
&  am1, d11, d1, dh1, frmka1, tml,
&  am2, d12, d2, dh2, frmka2, tm2,
&  am3, d13, d3, dh3, frmka3, tm3,
&  ahx, dzhh, z1h, dhx, frmkax, thx,
&  pin, pout, flw, flwn, tnot, ncell, ntype,
&  eps_m, maxit_m, itrans, icnv)

  "icnv = 1" is a flag that indicates that the momentum equation
c has not converged. A restart file is printed out and this is used in
c a test program to see what is needed to get convergence.
c
if (icnv .eq. 1)then
  write (iout7, 3000) eltim, tin, tout, tshl
  stop
end if
3000 format (lx,4g16.8)
c
SET OLD TIME MASS FLOWRATES TO NEW TIME VALUES
c
do j=1, mxloop
  flw(j) = flwn(j)
endo
c
Set the old values of the friction factor to the new values:
c
do i=1, ncell(1)
  foldl(i) = fnewl(i)
endo
do i=1, ncell(2)
  fold2(i) = fnew2(i)
endo
do i=1, ncell(3)
  fold3(i) = fnew3(i)
endo
do i=1, ncell(4)
  foldhh(i) = fnewhh(i)
endo
WRITE RESULTS OUT TO TECPLT AND OUTPUT FILES

if (mod(it,nplot).eq.0) then

write results to graphics tecplot file:

write(itec, '(17d12.4)') eltim/conv_tim, flw(1), flw(2), flw(3),
& flw(4), tin-convt, tout-convt, thx(2)-convt,
& tshl-convt, tm1(2)-convt, tm1(4)-convt, tm2(2)-convt,
& tm1(4)-convt, tm3(2)-convt, tm3(4)-convt, tnot-convt,
& pout*convp

MODULE #1 FILE:
write (itec1, ' (8d12.4) ')

MODULE #2 FILE:
write (itec2, ' (8d12.4) ')

MODULE #3 FILE:
write (itec3, ' (8d12.4) ')

write (itec4, ' (7d12.4) ') eltim/conv_tim, tout-convt,
& thx(1)-convt, thx(2)-convt, thx(3)-convt,
& tshl-convt

write time results to output file:

write (ioutl,1000)
eltim/conv_tim, tin-convt,
tml(l)-convt, tml(2)-convt, tml(3)-convt,
tml(4)-convt, tml(5)-convt, tout-convt,

write (ioutl,1015) tmhr
write (ioutl,1020) tml(1), tml(2), tml(3), tml(4), tml(5)
write (ioutl,1020) tm2(1), tm2(2), tm2(3), tm2(4), tm2(5)
write (ioutl,1020) tm3(1), tm3(2), tm3(3), tm3(4), tm3(5)
write (ioutl,1020) thx(1), thx(2), thx(3), tshl
write (ioutl,1030) tin, tout, tnot
write (ioutl,1025) flw(1), flw(2), flw(3), flw(4), pout

FORMAT STATEMENTS

210 format(' VARIABLES = t, Mod-1, Mod-2, Mod-3, Hx, tin, tout,',
& ' thx, tshl, tlidwn, tlup, t2dwn, t2up, t3in, t3out,',
& ' tavg, pout')
Subroutine energy.f

subroutine energy (flw, tin, tout, tml, tm2, tm3, thx, thshl, dt,
     wmsl, wms2, wms3, wmsx, wmsshl, wmsin, wmsout,
     sml, sm2, sm3, cp, cp1, cp2, cp3, cpshl,
     qdl, qd2, qd3, qdshx, convh, arshl,
     tinn, toutn, tmln, tm2n, tm3n, thxn, thshln,
     ncell, alpha, eps_t, maxit_t)

* ROUTINE energy
* THIS ROUTINE COMPUTES FLUID TEMPERATURES WITHIN EACH LOOP AND
* IN THE SHELL BASED ON THEIR THERMAL ENERGY BALANCES. THE SET
* OF EQNS ARE SOLVED BY PICARD ITERATION OVER THE HEADERS AND
* SHELL WITH MARCHING THRU EACH LOOP IN THEIR FLOW DIRECTION.
* A VARIABLE IMPLICITNESS (ALPHA) IS USED IN THE FINITE
* DIFFERENCE REPRESENTATION OF THE EQNS.

implied real*8 (a-h,o-z), integer (i-n)
include 'param.inc'

c argument arrays
     dimension wmsl(mxnode), wms2(mxnode), wms3(mxnode), wmsx(mxnode)
     dimension tmln(mxnode), tm1(mxnode), qdl(mxnode),
     sm1(mxnode), cp1(mxnode)
     dimension tm2n(mxnode), tm2(mxnode), qd2(mxnode),
     sm2(mxnode), cp2(mxnode)
     dimension tm3n(mxnode), tm3(mxnode), qd3(mxnode),
&    sm3(mxnode), cp3(mxnode)
dimension thxn(mxnode), thx(mxnode), qdhx(mxnode)
dimension flw(mxloop)
dimension ncell(mxloop)

c local arrays
dimension gama(mxnode), gamal(mxnode)
dimension tmli(mxnode), tm2i(mxnode), tm3i(mxnode), thxi(mxnode)
dimension err(mxloop)

*== COMPUTE EACH LOOP DONOR VARIABLE (GAMA) ==*

do i=1,mxloop
   gama(i) = one
   if (flw(i)<.lt.zero) then
      gama(i) = zero
   endif
enddo

do i=1,mxloop
   gamal(i) = one - gama(i)
enddo

*== SOLVE LINEAR SET OF ENERGY EQNS ==*

* INITIALIZE NEW/ITERATE (n/i) TIME FLUID TEMPERATURES TO OLD VALUES*

   m1 = ncell(1)
   m2 = ncell(2)
   m3 = ncell(3)
   m4 = ncell(4)
do i=1,m1
   tmli(i) = tml(i)
   tmln(i) = tml(i)
enddo

do i=1,m2
   tm2i(i) = tm2(i)
   tm2n(i) = tm2(i)
enddo

do i=1,m3
   tm3i(i) = tm3(i)
   tm3n(i) = tm3(i)
enddo

do i=1,m4
   thxi(i) = thx(i)
   thxn(i) = thx(i)
enddo

tini = tin
touti = tout
tshli = tshl
tinn = tin
toutn = tout
tshln = tshl

*== PERFORM PICARD ITERATIONS ==*

write (22,'()')
c write (23, '(')
c write (24, '()')
htc = convh*arshl
alpha1 = one - alpha
do iter=1,maxit_t

INLET HEADER:

term0 = gama(1)*flw(1) + gama(2)*flw(2) + gama(3)*flw(3) 
& -gama(4)*flw(4)
term1 = gama(1)*cp*flw(1)*(alpha*tmli(1) + alphal*tm1(1))
term2 = gama(2)*cp*flw(2)*(alpha*tm2i(1) + alphal*tm2(1))
term3 = gama(3)*cp*flw(3)*(alpha*tm3i(1) + alphal*tm3(1))
term4 = gama(4)*cp*flw(4)*(alpha*thxi(m4) + alphal*thx(m4))
heap = cp*wmssin/dt
coef = hcap - alphal*cp*term0
coefn = hcap + alpha*cp*term0
tini = (coef*tin - term1 - term2 - term3 + term4)/coefn
sum = flw(4) - (flw(1) + flw(2) + flw(3))
write (23,322) iter,sum,tmli(1),tm2i(1),tm3i(1),thxi(m4),
& (gama(k),k=1,4)
c322 format(' iter,sum,tmli(1),tm2i(1),tm3i(1),thxi(m4),gama ',
& i4,9f10.3)

OUTLET HEADER:

term0 = gama(1)*flw(1) + gama(2)*flw(2) + gama(3)*flw(3) 
& - gama(4)*flw(4)
term1 = gama(1)*cp*flw(1)*(alpha*tmli(ml) + alphal*tm1(ml))
term2 = gama(2)*cp*flw(2)*(alpha*tm2i(m2) + alphal*tm2(m2))
term3 = gama(3)*cp*flw(3)*(alpha*tm3i(m3) + alphal*tm3(m3))
term4 = gama(4)*cp*flw(4)*(alpha*thxi(1) + alphal*thx(1))
heap = cp*wmssout/dt
coef = hcap + alphal*cp*term0
coefn = hcap - alpha*cp*term0	
touti = (coef*tout + term1 + term2 + term3 - term4)/coefn
SHELL OF HEAT EXCHANGER:

term4 = htc*(alpha*thxi(2) + alphal*thx(2))
hcap = cpshl*wmsshl/dt
coef = hcap - alphal*htc
coefn = hcap + alpha*htc
tshli = (coef*tshl + term4)/coefn
write (24,422) iter,thxi(2),tshl,tshli
c422 format(' iter,thxi(2),tshl,tshli ',i4,3f10.5)
MODULE #1 (VERTICAL LATERAL MODULES):

if (flw(1).ge.zero) then
  do i=1,ml
    if (i.eq.1) then
tinlet = alpha*tini + alphal*tin
else
    tinlet = alpha*tmli(i-1) + alphal*tml(i-1)
endif

termi = cp*flw(1)*tinlet
hcapi = (sml(i)*cp1(i) + wmsl(i)*cp)/dt
coefi = hcapi - alphal*cp*flw(1)
coefni = hcapi + alpha*cp*flw(1)
tmli(i) = (coefi*tml(i) + termi + qd1(i))/coefni
enddo
else
    do i=1,m1
        if (i.eq.ml) then
            tinlet = alpha*touti + alphal*tout
        else
            tinlet = alpha*tmli(i+1) + alphal*tml(i+1)
        endif
        termi = cp*flw(1)*tinlet
        hcapi = (sml(i)*cp1(i) + wmsl(i)*cp)/dt
        coefi = hcapi - alphal*cp*flw(1)
        coefni = hcapi + alpha*cp*flw(1)
        tmli(i) = (coefi*tmli(i) - termi + qd1(i))/coefni
    enddo
endif

endc
endc

MODULE #2 (ALL REMAINING VERTICAL MODULES):
endc
endc

if (flw(2).ge.zero) then
    do i=1,m2
        if (i.eq.1) then
            tinlet = alpha*tini + alphal*tin
        else
            tinlet = alpha*tm2i(i-1) + alphal*tm2(i-1)
        endif
        termi = cp*flw(2)*tinlet
        hcapi = (sm2(i)*cp2(i) + wms2(i)*cp)/dt
        coefi = hcapi - alphal*cp*flw(2)
        coefni = hcapi + alpha*cp*flw(2)
        tm2i(i) = (coefi*tm2(i) + termi + qd2(i))/coefni
    enddo
else
    do i=1,m2
        if (i.eq.m2) then
            tinlet = alpha*touti + alphal*tout
        else
            tinlet = alpha*tm2i(i+1) + alphal*tm2(i+1)
        endif
        termi = cp*flw(2)*tinlet
        hcapi = (sm2(i)*cp2(i) + wms2(i)*cp)/dt
        coefi = hcapi - alphal*cp*flw(2)
        coefni = hcapi + alpha*cp*flw(2)
        tm2i(i) = (coefi*tm2(i) - termi + qd2(i))/coefni
    enddo
endif
c MODULE #3 (ALL HORIZONTAL MODULES):
c
if (flw(3) .ge. zero) then
  do i=1,m3
    if (i.eq.l) then
      tinlet = alpha*tini + alphal*tin
    else
      tinlet = alpha*tm3i(i-1) + alphal*tm3(i-1)
    endif
    term1 = cp*flw(3)*tinlet
    hcap = (sm3(i)*cp3(i) + wms3(i)*cp)/dt
    coef = hcap + alphal*cp*flw(3)
    coefn = hcap - alphal*cp*flw(3)
    tm3i(i) = (coef*tm3(i) + term1 + qd3(i))/coefn
    write (23,322) iter, i,tini, tinlet,tm3i(i),tm3(i)
  c322 format('+ iter, i,tini, tinlet, tm3i(i),tm3(i) ',2i4,4f10.5)
  enddo
else
  do i=1,m3
    if (i.eq.m3) then
      tinlet = alpha*touti + alphal*tout
    else
      tinlet = alpha*tm3i(i+1) + alphal*tm3(i+1)
    endif
    term1 = cp*flw(3)*tinlet
    hcap = (sm3(i)*cp3(i) + wms3(i)*cp)/dt
    coef = hcap + alphal*cp*flw(3)
    coefn = hcap - alphal*cp*flw(3)
    tm3i(i) = (coef*tm3(i) - term1 + qd3(i))/coefn
    write (23,323) iter, i,touti, tinlet,tm3i(i),tm3(i)
  c323 format('- iter, i,touti, tinlet, tm3i(i),tm3(i) ',2i4,4f10.5)
  enddo
endif
c HEAT EXCHANGER LOOP #4:
c
if (flw(4).ge.zero) then
  do i=1,m4
    if (i.eq.1) then
      tinlet = alpha*thxi(i-1) + alphal*thx(i-1)
    else
      tinlet = alpha*tm3i(i-1) + alphal*tm3(i-1)
    endif
    term0 = cp*flw(4) + htc
    term1 = cp*flw(4)*tinlet
    term2 = htc*(alpha*tshli + alphal*tshl)
    hcap = wmshx(i)*cp/dt
    coef = hcap - alphal*term0
    coefn = hcap + alpha*term0
    thxi(i) = (coef*thx(i) + term1 + term2 + qdhx(i))/coefn
  else
    if (i.eq.1) then
      tinlet = alpha*touti + alphal*tout
    else
      tinlet = alpha*thxi(i-1) + alphal*thx(i-1)
    endif
  endif
c
endif
term1 = cp*flw(4)*tinlet
hcap = wmshx(i)*cp/dt
coef = hcap - alphal*cp*flw(4)
coeff = hcap + alpha*cp*flw(4)

thxi(i) = (coefficient*thx(i) + term1 + qdhx(i))/coeffn
endif
dendo
else

do i=1,m
if (i.eq.2) then
tinlet= alpha*thxi(i+1) + alphal*thx(i+1)
term0 = cp*flw(4) - htc
term1 = cp*flw(4)*tinlet
term2 = htc*(alpha*tshli + alphal*tsh1)
hcap = wmshx(i)*cp/dt
coeff = hcap + alphal*term0
coeffn = hcap - alphal*term0

thxi(i) = (coefficient*thx(i) - term1 - term2 + qdhx(i))/coeffn
else
if (i.eq.m) then
tinlet = alpha*tini + alphal*tin
else
tinlet = alpha*thxi(i+1) + alphal*thx(i+1)
endif
term1 = cp*flw(4)*tinlet
hcap = wmshx(i)*cp/dt
coeff = hcap + alphal*cp*flw(4)
coeffn = hcap - alphal*cp*flw(4)

thxi(i) = (coefficient*thx(i) - term1 + term2 + qdhx(i))/coeffn
endif
dendo
endif

c TEST FOR CONVERGENCE AND UPDATE NEW (n) TIME FLUID TEMPERATURES:
c
er(1) = zero
do i=1,m
err(1) = err(1) + abs(tmli(i) - tmln(i))
tmln(i) = tmli(i)
enddo

er(2) = zero
do i=1,m
err(2) = err(2) + abs(tm2i(i) - tm2n(i))
tm2n(i) = tm2i(i)
enddo

er(3) = zero
do i=1,m
err(3) = err(3) + abs(tm3i(i) - tm3n(i))
tm3n(i) = tm3i(i)
enddo
err(4) = zero

do i=1,m4
    err(4) = err(4) + abs(thxi(i) - thxn(i))
    thxn(i) = thxi(i)
enddo

c
errin = abs(tini - tinn)
errout = abs(touti - toutn)
errshl = abs(tshli - tshln)
tinn = tini
toutn = touti
tshln = tshli

c
write (22,222) iter,tinn,toutn,tshl,tmln(4),tm3n(3)
c222 format(' iter,tinn,toutn,tshl,tml4,tm33 ',i4,5f10.3)
c
write (22,223) (err(k),k=1,4),errin,errout,errshl
c223 format(' err ',4x,7e12.5)
c
if ((err(1) .lt.eps_t) .and. (err(2) .lt.eps_t) .and.
    & (err(3) .lt.eps_t) .and. (err(4) .lt.eps_t) .and.
    & (errin .lt.eps_t) .and. (errout.lt.eps_t) .and.
    & (errshl.lt.eps_t)) then
    goto 1000
endif

c
endif

c
CONVERGENCE FAILURE:
c
write (*,200)
stop
c

CHECK INLET AND OUTLET MASS BALANCES VERSUS TIME STEP SIZE:
c
dt_max = 1.0d10
c
dflow = flw(4) - (flw(1) + flw(2) + flw(3))
c
if (dflow.lt.zero) then
    INLET HEADER:
    dt_max = -wmsin/dflow
c
elseif (dflow.gt.zero) then
    INLET HEADER:
    dt_max = wmsout/dflow
c
endif
c
if (dt.gt.dt_max) then
    write (*,100) eltim,dt,dt_max
    stop
c
endif

c === FORMAT STATEMENTS ===========================================
c
c100 format (///,' Program Terminated Due to Time Step Size Error',
& ///,' Mass balance error at time = ',f10.4,' sec',
& ///,' Current time step size = ',f10.4,' sec',

Subroutine momtm.f

subroutine momtm(rho,vis,g,beta,rough,zlt,dt,condl,eltim,
& qd1,qd2,qd3,qdhx,
& fold1,fold2,fold3,foldhx,dfdtmx,
& fnew1,fnew2,fnew3,fnewhx,
& am1,dz1 ,z11 ,dh1 ,frmka1,tml1,
& am2,dz2 ,z12 ,dh2 ,frmka2,tm2,
& am3,dz3 ,z13 ,dh3 ,frmka3,tm3,
& ahx,dzhx,zlhx,dhhx,frmkax,thx,
& pin,pout,flw,flwn,tnot,ncell,ntype,
& eps,m,maxit_m,itrans,icnvg)

******************************************************************************
This subroutine solves the four momentum equations (two vertical
module momentums, one horizontal module momentum, one heat exchanger
momentum) and the continuity
equation for the outlet header simultaneously, using Newton's
method. The values returned are the updated mass flowrates
in the four loops and the outlet header pressure.
******************************************************************************

implicit real*8 (a-h.o-z), integer (i-n)
include 'param.inc'

dimension a(5,5),b(5),indx(5)

dimension ncell(mxloop), ntype(mxloop)
dimension flw(mxloop), flwn(mxloop), zlt(mxloop)

dimension tm1(mxnode), am1(mxnode), dh1(mxnode), dz1(mxnode),
& z11(mxnode)
dimension tm2(mxnode), am2(mxnode), dh2(mxnode), dz2(mxnode),
& z12(mxnode)
dimension tm3(mxnode), am3(mxnode), dh3(mxnode), dz3(mxnode),
& z13(mxnode)
dimension thx(mxnode), ahx(mxnode), dhhx(mxnode), dzhx(mxnode),
& zlhx(mxnode)
dimension ral(mxnode),ra2(mxnode),ra3(mxnode),rahx(mxnode),
& iral(mxnode),ira2(mxnode),ira3(mxnode),irahx(mxnode),
& qd1(mxnode),qd2(mxnode),qd3(mxnode),qdhx(mxnode),
& fold1(mxnode),fold2(mxnode),fold3(mxnode),foldhx(mxnode),
& fnew1(mxnode),fnew2(mxnode),fnew3(mxnode),fnewhx(mxnode)

c iout5=22
iout6 = 23

c INITIALIZE NEW TIME MASS FLOWRATES TO PREVIOUS TIME VALUES
c do j=1,mxloop
  flwn(j) = flw(j)
endo

c == SOLVE NON-LINEAR SET OF MASS/MOMENTUN EQNS ==

do iter=1,maxit_m
  call rayleigh(flwn,g,beta,condl,rho,vis,qd1,qd2,qd3,qdhx,
  & dh1,dh2,dh3,dhhx,z11,z12,z13,z1hx,
  & dz1,dz2,dz3,dzhx,eltim,
  & ncell,ral,ra2,ra3,rahx,ira1,ira2,ira3,irahx)

c MODULE #1 (VERTICAL LATERAL MODULES):

c call momloop (rho,vis,g,beta,aml,dzl,zl1,dhl,rough,frmka1,
  & pin,pout,tml,tnot,flw(1),flwn(1),zlt(1),dt,
  & ral,ira1,fold1,
  & flm,df1dm1,df1dpo,ncell(1),ntype(1),itrans,
  & fnew1,eltim,iter)

c a(1,1) = df1dm1
a(1,2) = zero
a(1,3) = zero
a(1,4) = zero
a(1,5) = df1dpo
b(1) = -fml
elthrs = eltim/3600.0d0
write (iout5,500)
write (iout5,502)iter,elthrs
write (iout5,505)flm,df1dm1,df1dpo

c MODULE #2 (ALL REMAINING VERTICAL MODULES):

c call momloop (rho,vis,g,beta,am2,dz2,zl2,dh2,rough,frmka2,
  & pin,pout,tml,tnot,flw(2),flwn(2),zlt(2),dt,
  & ra2,ira2,fold2,
  & flm,df2dm2,df2dpo,ncell(2),ntype(2),itrans,
  & fnew2,eltim,iter)

c a(2,1) = zero
a(2,2) = df2dm2
a(2,3) = zero
a(2,4) = zero
a(2,5) = df2dpo
b(2) = -fml
write (iout5,505)flm,df2dm2,df2dpo

c MODULE #3 (ALL HORIZONTAL MODULES):

c call momloop (rho,vis,g,beta,am3,dz3,zl3,dh3,rough,frmka3,
  & pin,pout,tml,tnot,flw(3),flwn(3),zlt(3),dt,
  & ra3,ira3,fold3,
  & flm,df3dm3,df3dpo,ncell(3),ntype(3),itrans,
  & fnew3,eltim,iter)

c a(3,1) = zero
a(3,2) = zero
a(3,3) = df3dm3
a(3,4) = zero
a(3,5) = df3dpo
b(3) = -fm3
write (iout5, 505) fm3, df3dm3, df3dpo

c
HEAT EXCHANGER LOOP #4:

c
call momloop (rho, vis, g, beta, ahx, dzhx, zlhx, dhhx, rough, frmkax, pin, pout, thx, tnot, flw(4), flwn(4), zlt(4), dt, rahx, irahx, foldhx, & fmhx, dfxdmx, dfxdpo, ncell(4), ntype(4), itrans, & fnewhx, eltim, iter)
c
a(4,1) = zero
a(4,2) = zero
a(4,3) = zero
a(4,4) = dfxdmx
a(4,5) = dfxdpo
b(4) = -fmhx
write (iout5, 505) fmhx, dfxdmx, dfxdpo

c
OUTLET HEADER CONTINUITY BALANCE:
c
c
c
SOLVE FOR NEWTON ITERATE BY LU DECOMPOSITION:
c
n = 5
np = 5
call ludcmp (a,n,np,indx,d)
call lubksb (a,n,np,indx,b)
c
UPDATE NEW TIME MASS FLOWRATE ARRAY AND OUTLET HEADER PRESSURE:
c
write (iout6, 510) iter, flwn(l), flwn(2), flwn(3), flwn(4), pout
write (iout6, 510) iter, (b(i), i=1, 5)
do i=1, mxloop
cc if (j .lt. 20) then
cc flwn(i) = flwn(i) + b(i)
cc else
cc flwn(i) = flwn(i) + 0.5*b(i)
cc endif
endo
cc if (j .lt. 20) then
cc pout = pout + b(5)
cc else
cc
cc  pout = pout + 0.5*b(5)
cc  endif

c  TEST FOR CONVERGENCE:

c  err1 = abs(b(1))/(abs(flwn(1))+eps_m)
err2 = abs(b(2))/(abs(flwn(2))+eps_m)
err3 = abs(b(3))/(abs(flwn(3))+eps_m)
err4 = abs(b(4))/(abs(flwn(4))+eps_m)
err5 = abs(b(5))/pout

c  if ((err1.lt.1.) and. (err2.lt.1.) and. (err3.lt.1.) and. (err4.lt.1.) and. (err5.lt.1.) then
    icnvg = 0
  goto 1000
endif

enddo

write (iout6,501)
write (iout6,50())
write (iout6,501)

cc  CONVERGENCE FAILURE:

c  icnvg = 1
write (*,530)
go to 1000

cc  flwn(4) = flwn(1)+flwn(2)+flwn(3)
cc  stop

cc === FORMAT STATEMENTS  ================================

cc  20 write (*,500)
cc  write (iout6,520)ii
cc  write (iout6,510)flwn(1),flwn(2),flwn(3),flwn(4),pout
c  write (iout6,510)err1, err2, err3, err4, err5
cc  100 continue

cc  500 format (lx,'************************************')
501 format (lx,')'
502 format (lx,i6,' time = ',g16.8)
505 format (lx,3g16.8)
510 format (i5,5g16.8)
511 format (lx,5g16.8)
520 format (lx,' ii = ',i6)
530 format (lx,' Program Terminated Due To:','/
  & ' MASS/MOMENTUM SOLUTION CONVERGENCE FAILURE')

cc  1000 continue
return
end

Subroutine momloop.f

subroutine momloop (rho,vis,g,beta,am,dz,zl,dh,rough,frmka,
  & pin,pout,tm,tnot,flow,flown,zlt,dt,ra,ira,
  & fold,fm,dfdm,dfdpo,nell,itype,itrans,
& fnew, eltim, iter)

**********************************************************************************************

* ROUTINE momloop
* THIS SUBROUTINE CALCULATES THE VALUE OF THE "F" EQUATION AND
* THE PARTIAL DERIVATIVES WITH RESPECT TO THE "DELTA" VARIABLES
* FOR THE MOMENTUM EQUATION FOR EACH SPECIFIED FLOW LOOP. THESE
* VALUES ARE USED IN THE SOLUTION OF THE NON-LINEAR MOMENTUM
* EQUATIONS BY NEWTON'S METHOD WHEN THE VARIOUS FLOW LOOPS ARE
* CONNECTED.

* ITYPE = 1 FOR MODULES
* ITYPE = 2 FOR HEAT EXCHANGER

**********************************************************************************************

implicit real*8 (a–h,o-z), integer (i-n)
include 'param.inc'
dimension tm(mxnode), am(mxnode), dh(mxnode), dz(mxnode), zl(mxnode)
dimension fric(mxnode), fricp(mxnode), ra(mxnode), ira(mxnode),
& fold(mxnode), fnew(mxnode)

c CALCULATE VALUES FOR THE FRICTION FACTORS:
c CALCULATE PERTURBED FLOWS AND FRICTION FACTORS:

eps = 1.0d-4
if (flown .ge. 0.0)then
  flownp = flown + eps
else
  flownp = flown - eps
end if

c write (19, ('elhim, iter: ',f10.6,i5)) eltim/3600.0d0, iter
do i=1,ncell
  ed = rough/dh(i)
  re = (abs(flown)*dh(i))/(vis*am(i))
  rep = (abs(flownp)*dh(i))/(vis*am(i))
  rap = ra(i)*((flown/flownp)**.25d0)
  rap = ra(i)

  call frict(re, ed, ra(i), ira(i), fric(i))
  call frict(rep, ed, rap, ira(i), fricp(i))
  dfrctp = fricp(i)-fric(i)
  dffdt = (fric(i)-fold(i))/dt
dffabs = abs(dffdt)
if (dffabs .gt. dfdtmx)then
  dffdt = sign(dffdt,dfdtmx)
endif
fric(i) = fold(i) + dffdt*dt
dfrctp
fnew(i) = fric(i)
endo
CALCULATE THE VALUE OF THE FUNCTION AND THE JACOBEAN ENTRIES:

\[ \text{sum} = \text{zero} \]
\[ \text{do } i = 1, \text{ncell} \]
\[ \text{sum} = \text{sum} + (\text{one} - \beta \ast (\text{tm}(i) - \text{tnot})) \ast \text{am}(i) \ast \text{dz}(i) \]
\[ \text{enddo} \]
\[ \text{bterm} = \rho \ast g \ast \text{sum} \]

COMPUTE FTERM AT CURRENT ITERATE AND PERTURBED POINTS:

\[ \text{sum} = \text{zero} \]
\[ \text{sump} = \text{zero} \]
\[ \text{do } i = 1, \text{ncell} \]
\[ \text{sum} = \text{sum} + \text{fric}(i) \ast \text{z}(i) / \left( \text{dh}(i) \ast \text{am}(i) \right) \]
\[ \text{sump} = \text{sum} + \text{fricp}(i) \ast \text{z}(i) / \left( \text{dh}(i) \ast \text{am}(i) \right) \]
\[ \text{enddo} \]
\[ \text{fterm} = (\text{sum} + \text{frmka}) / (2 \ast \rho) \]
\[ \text{ftermp} = (\text{sum} + \text{frmka}) / (2 \ast \rho) \]

COMPUTE ATERM AT CURRENT ITERATE:

\[ \text{if } (\text{itrans} \ast \text{eq.} 0) \text{ then} \]
\[ \text{aterm} = \text{zero} \]
\[ \text{else} \]
\[ \text{aterm} = \text{zlt} / \text{dt} \]
\[ \text{endif} \]

EFFICIENT COMPUTE FUNCTION AND DERIVATIVES BASED ON LOOP TYPE ==————

\[ \text{if } (\text{itype} \ast \text{eq.} 1) \text{ then} \]

CALCULATE FUNCTION AT CURRENT ITERATE AND PERTURBED POINTS:

\[ \text{fm} = \text{pin} \ast \text{am}(1) - \text{pout} \ast \text{am(ncell)} + \text{bterm} \]
\[ + \text{aterm} \ast (\text{flow} - \text{flown}) - \text{flown} \ast \text{abs}(\text{flown}) \ast \text{fterm} \]
\[ \text{fmp} = \text{pin} \ast \text{am}(1) - \text{pout} \ast \text{am(ncell)} + \text{bterm} \]
\[ + \text{aterm} \ast (\text{flow} - \text{flownp}) - \text{flownp} \ast \text{abs}(\text{flownp}) \ast \text{ftermp} \]

ESTIMATE FUNCTION DERIVATIVE NUMERICALLY BY SIMPLE FORWARD DIFFERENCING FOR FLOW TERM AND ANALYTICALLY FOR PRESSURE TERM:

\[ \text{if } (\text{flown} \geq 0.0) \text{ then} \]
\[ \text{dfdm} = (\text{fmp} - \text{fm}) / \epsilon \]
\[ \text{else} \]
\[ \text{dfdm} = (\text{fm} - \text{fmp}) / \epsilon \]
\[ \text{endif} \]
\[ \text{dfdpo} = -\text{am(ncell)} \]
\[ \text{else} \]

CALCULATE FUNCTION AT CURRENT ITERATE AND PERTURBED POINTS:

\[ \text{fm} = \text{pout} \ast \text{am}(1) - \text{pin} \ast \text{am(ncell)} + \text{bterm} \]
\[ + \text{aterm} \ast (\text{flow} - \text{flown}) - \text{flown} \ast \text{abs}(\text{flown}) \ast \text{fterm} \]
\[ \text{fmp} = \text{pout} \ast \text{am}(1) - \text{pin} \ast \text{am(ncell)} + \text{bterm} \]
\[ + \text{aterm} \ast (\text{flow} - \text{flownp}) - \text{flownp} \ast \text{abs}(\text{flownp}) \ast \text{ftermp} \]
c ESTIMATE FUNCTION DERIVATIVE NUMERICALLY BY SIMPLE FORWARD
C DIFFERENCING FOR FLOW TERM AND ANALYTICALLY FOR PRESSURE TERM:
c
   if (flown .ge. 0.0) then
      dfdm = (fmp - fm)/eps
   else
      dfdm = (fm - fmp)/eps
   end if
   dfdpo = am(1)
endif

c return
end

Subroutine rayleigh.f

subroutine rayleigh (flwn,g,beta,condl,rho,vis,qd1,qd2,qd3,qdhx,
& dh1,dh2,dh3,dhhx,z11,z12,z13,zlhx,
& dz1,dz2,dz3,dzhx,eltim,
& ncell,ral,ra2,ra3,rahx,iral,ira2,ira3,irahx)

*************** **************** *************** **************** *****
******** ROUTINE rayleigh ********
********** THIS ROUTINE COMPUTES EACH FLUID CELL RAYLEIGH NUMBER. ********
******** A RAYLEIGH NUMBER DEFINED FOR MIXED CONVECTION FLOW IS ********
******** COMPUTED FOR EACH FLUID CELL BASED ON HEATING/COOLING AND ********
******** CURRENT FLOW DIRECTION. ********
******** A FLAG (ira) IS SET INDICATING IF CELL IS IN: ********
******** ira = 1 FOR HEATED UPFLOW (OR COOLED DOWNFLOW) ********
******** ira = 0 FOR HORIZONTAL HEATED, COOLED, OR ADIABATIC FLOW ********
******** ira =-1 FOR HEATED DOWNFLOW (OR COOLED UPLFLOW) ********
********

implicit real*8 (a-h,o-z), integer (i-n)
include 'param.inc'
c argument arrays
dimension ral(mxnode), ra2(mxnode) , ra3(mxnode), rahx(mxnode)
dimension iral(mxnode) ,ira2(mxnode) ,ira3(mxnode) ,irahx(mxnode)
dimension qdl(mxnode), qd2(mxnode), qd3(mxnode), qdhx(mxnode)
dimension dhl(mxnode), dh2(mxnode), dh3(mxnode), dhhx(mxnode)
dimension zll(mxnode) , z12(mxnode), z13(mxnode), zlhx(mxnode)
dimension dzl(mxnode), dz2(mxnode), dz3(mxnode), dzhx(mxnode)
dimension flwn(mxloop)
dimension ncell(mxloop)
c
*** CALCULATE EACH FLUID CELL RAYLEIGH NUMBER ==============
c
iout3 = 17
eps = 1.0d-2
prop = g*beta*(rho**2)/(vis*condl)
ramax = 12.0d0
cc    prop = 0.0
cc
cccc    write (iout3,990)prop
990   format (1x,'PROP = ',g16.8)
cc
ehrs = eltim/3600.0
cc
m1 = ncell(1)
m2 = ncell(2)
m3 = ncell(3)
m4 = ncell(4)
cc
cc MODULE #1 (VERTICAL LATERAL MODULES):
c
    flown = abs(flwn(1))
do i=1,m1
       ral(i) = (prop*abs(qd1(i))*(dh1(i)**4)/
                      & (z11(i)*flown))**.25d0
       if (ral(i) .gt. ramax)ral(i)=ramax
cccc    if ((i .eq. 2) .or. (i .eq. 4))
cccc    &    write (iout3,1000)i,dh1(i),z11(i),flown,qd1(i),ral(i)
cccc 1000 format (1x,i6,5g14.7)
enddo
cc
cc MODULE #2 (ALL REMAINING VERTICAL MODULES):
c
    flown = abs(flwn(2))
do i=1,m2
       ra2(i) = (prop*abs(qd2(i))*(dh2(i)**4)/
                      & (z12(i)*flown))**.25d0
       if (ra2(i) .gt. ramax)ra2(i)=ramax
cccc    if ((i .eq. 2) .or. (i .eq. 4))
cccc    &    write (iout3,1000)i,dh2(i),z12(i),flown,qd2(i),ra2(i)
enddo
write (iout3,1000)elhrs,ral(2),ral(4),ra2(2),ra2(4)
1000 format (5g14.7)
cc
cc MODULE #3 (ALL HORIZONTAL MODULES):
c
    flown = abs(flwn(3))
do i=1,m3
       ra3(i) = (prop*abs(qd3(i))*(dh3(i)**4)/
                      & (z13(i)*flown))**.25d0
       if (ra3(i) .gt. ramax)ra3(i)=ramax
enddo
c
cc HEAT EXCHANGER LOOP #4:
c
    flown = abs(flwn(4))
do i=1,m4
       rahx(i) = (prop*abs(qdhx(i))*(dhhx(i)**4)/
                      & (zlhx(i)*flown))**.25d0
       if (rahx(i) .gt. ramax)rahx(i)=ramax
enddo
cc
=== SET HEATING/COOLING UP/DOWN FLOW STATE ===
MODULE #1 (VERTICAL LATERAL MODULES):

\[ \text{flown} = \text{flwn}(i) \]
\[ \text{do } i = 1, m_l \]
\[ \text{point} = d z l(i) \]

--- HEATED -----------------------------

\[ \text{if } (q_d l(i) \cdot \text{gt. zero) then} \]
\[ \text{if } (\text{point} \cdot \text{lt. zero}) \text{ then} \]
\[ \text{if } (\text{flown} \cdot \text{ge. zero}) \text{ then} \]

--- heated upflow
\[ \text{iral}(i) = 1 \]

else

--- heated downflow
\[ \text{iral}(i) = -1 \]

endif

elseif (point.gt.zero) then

--- heated downflow
\[ \text{iral}(i) = -1 \]

else

--- heated upflow
\[ \text{iral}(i) = 1 \]

endif

else

--- horizontal
\[ \text{iral}(i) = 0 \]

endif

--- COOLED -----------------------------

elseif (q_d l(i) \cdot \text{lt. zero) then} \]
\[ \text{if } (\text{point} \cdot \text{lt. zero}) \text{ then} \]
\[ \text{if } (\text{flown} \cdot \text{ge. zero}) \text{ then} \]

--- cooled upflow
\[ \text{iral}(i) = -1 \]

else

--- cooled downflow
\[ \text{iral}(i) = 1 \]

endif

elseif (point.gt.zero) then

--- cooled downflow
\[ \text{iral}(i) = 1 \]

else

--- cooled upflow
\[ \text{iral}(i) = -1 \]

endif

else

--- horizontal
\[ \text{iral}(i) = 0 \]

endif

--- ADIABATIC -----------------------------
\[ \text{iral}(i) = 0 \]

endif

endo

MODULE #2 (VERTICAL LATERAL MODULES):
C
flown = flwn(2)
do i=1,m2
  point = d2z(i)
C-HEATED-----------------------------
  if (qd2(i).gt.zero) then
    if (point.lt.zero) then
      if (flown.ge.zero) then
        heated upflow
        ira2(i) = 1
      else
        heated downflow
        ira2(i) = -1
      endif
    elseif (point.gt.zero) then
      if (flown.ge.zero) then
        heated downflow
        ira2(i) = -1
      else
        heated upflow
        ira2(i) = 1
      endif
    else
      horizontal
      ira2(i) = 0
    endif
  elseif (qd2(i).lt.zero) then
    if (point.lt.zero) then
      if (flown.ge.zero) then
        cooled upflow
        ira2(i) = -1
      else
        cooled downflow
        ira2(i) = 1
      endif
    elseif (point.gt.zero) then
      if (flown.ge.zero) then
        cooled downflow
        ira2(i) = 1
      else
        cooled upflow
        ira2(i) = -1
      endif
    else
      horizontal
      ira2(i) = 0
    endif
  else
    adiabatic
    ira2(i) = 0
  endif
enddo
C
C
C
C
C MODULE #3 (HORIZONTAL LATERAL MODULES):
C
flown = flwn(3)
do i=1,m3
   point = dz3(i)
c --- HEATED ----------------------------------------
   if (qd3(i).gt.zero) then
      if (point.lt.zero) then
         if (flown.ge.zero) then
            heated upflow
            ira3(i) = 1
         else
            heated downflow
            ira3(i) = -1
         endif
      elseif (point.gt.zero) then
         if (flown.ge.zero) then
            heated downflow
            ira3(i) = -1
         else
            heated upflow
            ira3(i) = 1
         endif
      else
         horizontal
         ira3(i) = 0
      endif
c --- COOLED -------------------------------------
   elseif (qd3(i).lt.zero) then
      if (point.lt.zero) then
         if (flown.ge.zero) then
            cooled upflow
            ira3(i) = -1
         else
            cooled downflow
            ira3(i) = 1
         endif
      elseif (point.gt.zero) then
         if (flown.ge.zero) then
            cooled downflow
            ira3(i) = 1
         else
            cooled upflow
            ira3(i) = -1
         endif
      else
         horizontal
         ira3(i) = 0
      endif
c --- ADIABATIC ----------------------------------
   else
      adiabatic
      ira3(i) = 0
   endif
endo

MODULE #4 (=T EXCHANGER LOOP):

flown = flwn(4)
do i=1,m4
  point = dzhx(i)
  c --- HEATED ------------------------------
    if (qdhx(i).gt.zero) then
      if (point.lt.zero) then
        if (flown.ge.zero) then
          c --- heated upflow
          irahx(i) = 1
        else
          c --- heated downflow
          irahx(i) = -1
        endif
      elseif (point.gt.zero) then
        if (flown.ge.zero) then
          c -– heated downflow
          irahx(i) = -1
        else
          c -– heated upflow
          irahx(i) = 1
        endif
      else
        c --- horizontal
        irahx(i) = 0
      endif
    endif
  c --- COOLED ------------------------------
  elseif (qdhx(i).lt.zero) then
    if (point.lt.zero) then
      if (flown.ge.zero) then
        c --– cooled upflow
        irahx(i) = -1
      else
        c –– cooled downflow
        irahx(i) = 1
      endif
    elseif (point.gt.zero) then
      if (flown.ge.zero) then
        c -– cooled downflow
        irahx(i) = 1
      else
        c -– cooled upflow
        irahx(i) = -1
      endif
    else
      c --- horizontal
      irahx(i) = 0
    endif
  else
    c --- ADIABATIC -------------------------------
    irahx(i) = 0
  endif
endo
c
return
end
Subroutine frict.f

subroutine frict(re,ed,ran,iran,zf)

This subroutine calculates the Moody friction factor for a tube. Inputs are the Reynolds #, the hydraulic diameter, and the relative roughness. The Darcy friction factor is returned.

implicit real*8 (a-h,o-z), integer (i-n)

eps = 1.0d-6
remst = 25.0d0
rem = dmax1(re,eps)
cfclu = 0.11815501d0
cfd2u = -0.29378955d-2
cfc3u = -0.33784101d1
cfd4u = 0.11636369d1
cfcld = -0.38259936d0
cfd2d = 0.71433520d1
cfc3d = -0.37145261d-2

This is the mixed convection Rayleigh # multiplier:

if (iran .le. 1.5d0) then
  bm = 1.0d0
else
  if (iran .gt. 0) then
    if (ran .le. 8.0d0) then
      expn = cfclu*((ran-1.5d0)**2)+cfd2u*((ran-1.5d0)**3)
      bm = dexp(expn)
    else
      expn = cfc3u+cfd4u*(ran-1.5d0)
      bm = dexp(expn)
    endif
  else if (iran .lt. 0) then
    if (ran .le. 8.0d0) then
      expn = cfcld*( (ran-1.5d0)**2) +cfd2d* (ran-1.5d0)**3)
      & +cfc3d*( (ran-1.5d0)**4)
      bm = dexp(expn)
    else
      bm = 1.0d0/24.0d0
    endif
  else
    bm = 1.0d0
  endif
endif

if (rem .le. remst) then
  xx = rem/remst
  w2 = (xx**2)*(3.0d0-2.0d0*xx)
  w1 = 1.0d0-w2
  bm = w1+w2*bm
endif

end if
Calc. standard friction factor

if (rem.le.2300.0d0) then
  zf = 64.0d0/rem
elseif (rem.ge.4000.0d0) then
  zf = 1.0d0/((1.14d0 & - 2.0d0*log10(ed + 21.25d0/(rem**0.9d0)))**2)
else
  zf = 0.23158d0 - 2.0549d-4*rem & + 6.6351d-8*(rem**2) - 6.7336d-12*(rem**3)
endif
zf = bm*zf

Subroutine outhdr.f

subroutine outhdr (flwn, fmout, dfotdl, dfotd2, dfotd3, dfotdx)

This subroutine calculates the value of the \( f \) equation and the partial derivatives with respect to the \( \delta \) variables for the continuity equation for the outlet header. These values are used in the solution of the non-linear momentum equations by Newton's method.

Subroutine tmean.f

This subroutine calculates the mean (average) fluid temperature over the entire set of flow loops. The coefficient for thermal expansion and fluid viscosity are also updated based on new mean fluid temperature.
implicit real*8 (a-h,o-z), integer (i-n)
include 'param. inc'
dimension ncell(mxloop), coefb(5), coefv(8)
dimension tml(mxnode), wmsl(mxnode)
dimension tm2(mxnode), wms2(mxnode)
dimension tm3(mxnode), wms3(mxnode)
dimension thx(mxnode), wmshx(mxnode)
data coefb/ 4.9323453d-03,-6.3224634d-05, 2.8526660d-07,
& -5.4026781d-10, 3.7406938d-13/
data coefv/ 2.5217246d+00, -4.3320035d-02, 3.1823267d-04,
& -1.2943418d-06, 3.1456297d-09, -4.5657684d-12 ,
& 3.6636052d-15,-1.2534347d-18/
c CALCULATE MEAN TEMPERATURE (LOOP AVERAGE):
if (iskip.eq.0) then
  totmas = wmsin + wmsout
  do i=1,ncell(1)
    totmas = totmas + wmsl(i)
  enddo
  do i=1,ncell(2)
    totmas = totmas + wms2(i)
  enddo
  do i=1,ncell(3)
    totmas = totmas + wms3(i)
  enddo
  do i=1,ncell(4)
    totmas = totmas + wmslx(i)
  enddo
totemas = wmsin*tin + wmsout*tout
  do i=1,ncell(1)
    totemas = totemas + wmsl(i)*tml(i)
  enddo
  do i=1,ncell(2)
    totemas = totemas + wms2(i)*tm2(i)
  enddo
  do i=1,ncell(3)
    totemas = totemas + wms3(i)*tm3(i)
  enddo
  do i=1,ncell(3)
    totemas = totemas + wmslx(i)*thx(i)
  enddo
tnot = totemas/totmas
c CALCULATE NEW BETA:
beta = coefb(1) + (coefb(2) + (coefb(3) + (coefb(4)
& + coefb(5)*tnot)*tnot)*tnot)*tnot
**Subroutine condlq.f**

```fortran
subroutine condlq (t, l, condl, dkdt)

   *************** *************** *************** *************** *******
   * INPUT: *
   * t = TEMPERATURE IN K *
   * l = IDENTIFIER (1=1 FOR H2O, 1=2 FOR D2O) *
   *
   * OUTPUT: *
   * condl = THERMAL CONDUCTIVITY OF SATURATED LIQUID (W/m-K) *
   * dkdt = NORMALIZED DERIVATIVE OF THERMAL CONDUCTIVITY WITH *
   * RESPECT TO TEMPERATURE *
   *
   * RANGE: *
   * 10C < t < 300C *
   *
   ************* ********** ************ *************** ***************

   implicit real*8 (a-h,o-z), integer (i-n)
   parameter (tconv=273.15d0)
   dimension cl(2), c2(2), c3(2), c4(2), c3x2(2), c4x3(2)

   H2O PROPERTIES
   data cl(l) / 5.7032432d-1/, c2(l) /1.7996615d-3/,
   & c3(l) /-7.2881959d-6/, c4(l) /3.2412245d-9/,
   & c3x2(l) /-1.4576392d-5/, c4x3(l) /9.7236735d-9/

   D2O PROPERTIES
   data cl(2) / 5.6340135d-1/, c2(2) /1.4504443d-3/,
   & c3(2) /-7.9650470d-6/, c4(2) /7.1584948d-9/,
   & c3x2(2) /-1.5930094d-5/, c4x3(2) /2.1475484d-8/

   tinc = t - tconv
   condl = cl(1) + (c2(1) + (c3(1) + c4(1)*tinc)*tinc)*tinc
   dkdt = (c2(1) + (c3x2(1) + c4x3(1)*tinc)*tinc) / condl

return
end
```

**Subroutine dpower.f**

```fortran
subroutine dpower (eltin_p, tgstrt, tgdry,
                   & target, q1, q2, q3, qd1, qd2, qd3, qdhx,
                   ...)
```

The rest of the subroutine is cut off in the image provided.
ROUTINE dpower

**THIS ROUTINE COMPUTES EACH FLUID CELL DEPOSITED DECAY POWERS.**

- **CALCULATE EACH FLUID CELL DECAY POWER**

```plaintext
implicit real*8 (a-h,o-z), integer (i-n)
include 'param.inc'

argument arrays
dimension qdl(mxnode), qd2(mxnode), qd3(mxnode), qdhx(mxnode)
dimension ql(mxnode), q2(mxnode), q3(mxnode)
dimension ncell(mxloop)

=== CALCULATE EACH FLUID CELL DECAY POWER -------------------------------

conv_tim = 3600.0d0
ml = ncell(1)
m2 = ncell(2)
m3 = ncell(3)
m4 = ncell(4)
tau = eltim_p/conv_tim

COMPUTE DECAY FRACTION
if (tau.le.1.0d0) then
    write (*,200) tau
    stop
else
    decay = 3.849d-3*(0.96d0 - 0.363d0*log10(tau))
endif

COMPUTE TARGET POWER DEPOSITED TO DECOUPLER
if (eltim_p.gt.tgstrt) then
    qtgt = target
elseif (eltim_p.lt.tgdry) then
    qtgt = zero
else
    qtgt = target*(eltim_p-tgstrt) / (tgdry-tgstrt)
endif

COMPUTE BLANKET MODULE POWER DEPOSITED
MODULE #1
    do i=1,ml
        qdl(i) = decay*ql(i)
    enddo
    qdl(2) = qdl(2) + decay*qtgt

MODULE #2
    do i=1,m2
        qd2(i) = decay*q2(i)
```
enddo

c
Module 3

    do i=l, m3
        qd3(i) = decay*q3(i)
    enddo

c
Heat Exchanger #4

    do i=l, m4
        qdhx(i) = zero
    enddo

c
=== FORMAT STATEMENTS

200 format (' Program Terminated Due To:', /,
& ' INITIAL START TIME MUST BE GREATER THAN OR',
& ' EQUAL TO ONE HOUR', /,
& ' CURRENT ELAPSED TIME =', f10.3, ' hours')

c
    return
    end

Subroutine ludcmp.f

subroutine ludcmp(a,n,p,n,pindx,v)

***********

* ROUTINE ludcmp
* THIS SUBROUTINE PERFORMS LU DECOMPOSITION OF MATRIX A.
* PARTIAL ROW PIVOTING USED.

***********

implicit real*8 (a-h, o-z), integer (i-n)
parameter (nmax=100, tiny=1.0d-20, zero=0.0d0, one=1.0d0)

d=one

   do 12 i=l, n
       aamax=zero

   do 11 j=l, n
       if (abs(a(i,j)) .gt. aamax) aamax=abs(a(i,j))

11     continue

   if (aamax .eq. zero) pause 'singular matrix'
   vv(i)=one/aamax

12     continue

   do 19 j=l, n

19     continue

   do 14 i=l, j-1

14     continue
aamax=zero
do 16 i=j,n
   sum=a(i,j)
   do 15 k=1,j-1
      sum=sum-a(i,k)*a(k,j)
   15 continue
   a(i,j)=sum
dum=vv(i)*abs(sum)
if (dum .ge. aamax) then
   imax=i
   aamax=dum
endif
16 continue
if (j .ne. imax) then
   do 17 k=1,n
      dum=a(imax,k)
      a(imax,k)=a(j,k)
      a(j,k)=dum
   17 continue
d=-d
vv(imax)=vv(j)
endif
indx(j)=imax
if (a(j,j) .eq. zero) a(j,j)=tiny
if (j .ne. n) then
   dum=one/a(j,j)
   do 18 i=j+1,n
      a(i,j)=a(i,j)*dum
   18 continue
endif
19 continue
return
end

Subroutine lubksb.f

subroutine lubksb(a,n,np, indx,b)

*****'*********** **************** **************** **************** ***
 ROUTINE lubksb
 THIS SUBROUTINE PERFORMS FORWARD AND BACKWARD SWEEP OF LU
 DECOMPOSED MATRIX A GIVEN A RHS VECTOR B FOR SOLUTION.

implicit real*8 (a-h,o-z), integer (i–n)
parameter (zero=O.OdO)
dimension a(np,np), indx(n),b(n)
i=0
do 12 i=1,n
   ll=indx(i)
   sum=b(ll)
   b(ll)=b(i)
if (ii .ne. 0) then
    do 11 j=ii,i-1
        sum=sum-a(i,j)*b(j)
    11 continue
  else if (sum .ne. zero) then
    ii=i
  endif
  b(i)=sum
12. continue
  do 14 i=n,1,-1
    sum=b(i)
  do 13 j=i+1,n
    sum=sum-a(i,j)*b(j)
  13 continue
  b(i)=sum/a(i,i)
14 continue
return
end