APPENDIX

C

MISCELLANEOUS INFORMATION{ TC "MISCELLANEOUS INFORMATION" \l 1 }

C1 TRNSYS Manual Page{ TC "C1 TRNSYS Manual Page" \l 1 }

TYPE 60: STRATIFIED FLUID STORAGE TANK (WITH OPTIONAL INTERNAL HEATERS)

General Description

The thermal performance of a fluid-filled sensible energy storage tank, subject to thermal stratification, can be modeled by assuming that the tank consists of N (N 100) fully-mixed equal volume segments, as shown in Fig. 4.3.1.1. The degree of stratification is determined by the value of N. If N is equal to 1, the storage tank is modeled as a fully-mixed tank and no stratification effects are possible. Options of fixed or variable inlets, unequal size nodes, temperature deadband on heater thermostats, incremental loss coefficients, internal submersed heat exchangers, non-circular tanks, horizontal tanks, and losses to the flue of a gas auxiliary heater are all available.

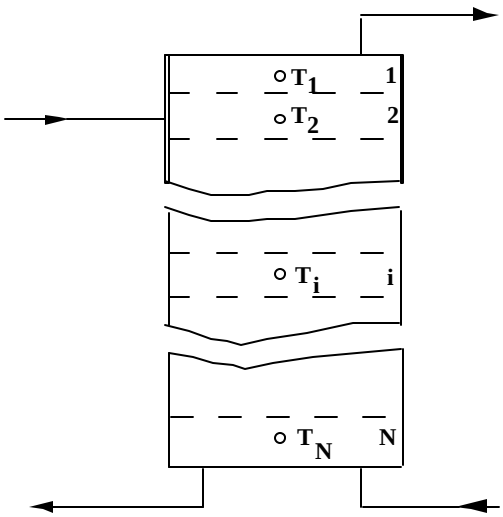


Figure 4.3.1.1. Stratified Fluid Storage Tank

To make it simple to change the number of nodes, tank component locations are entered as heights, measured from the floor up, rather than node numbers. These components include: inlet and outlet flows, auxiliary heaters, thermostats, and heat exchangers.

There are two inlet modes available. In mode 1, the flow stream enters the node that is closest to it in temperature. With sufficient nodes, this permits a maximum degree of stratification. In mode 2, flow streams enter the tank at fixed positions, as specified by the user. At the end of each time interval, any temperature inversions that exist are eliminated by mixing of the appropriate adjacent nodes.

The user may specify the height of each node using the last set parameters. Optionally, equal size nodes may be specified quite simply by setting parameter 31 to zero. The model will automatically divide the tank into equal segments. In this case, no additional node size specifications are required.

The volume of the tank used by the model is assumed to be the actual volume of the tank. A Sears 80 gallon hot water heater does not necessarily hold 80 gallons of water. It is recommended to the user that a tape measure and calculator be used to determine the tank volume, not the sticker on the outside.

The model optionally includes two electric resistance heating elements, subject to temperature and/or time control. The control option allows the addition of electrical energy to the tank during selected periods of each day (e.g., off-peak hours). The electric resistance heaters may operate in one of two modes. Mode 1 is a master/slave relationship. It allows the bottom heating element to be

enabled only when the top element is satisfied. In this control mode, it is impossible for both electric heaters to be on simultaneously. However, it is possible for both heaters to be on during the same TRNSYS time step (upper heater may be on during first half of the time step and lower heater may be on during the second half of the time step). Mode 1 is common to most domestic hot water applications.

In mode 2, both heaters may be on simultaneously. This allows for quicker heating of the storage tank, but at a significantly higher electric demand. If no electric heating elements are present in the tank to be modeled, the user may set the control switch (inputs 8 and 9) to have a constant value of zero, or specify the heater power as zero.

The auxiliary heaters employ a temperature deadband. The heater is enabled if the temperature of the node containing the thermostat is less than (T_{set} - ΔT_{db}) or if it was on for the previous time step and the thermostat temperature is less than T_{set}. If the lower heater meets these criteria and the master/slave relationship is employed, a check will be made to see if the upper electric heater is off before enabling the second heating element.

In many circumstances, the tank may not be uniformly insulated or users may wish to account for pipe entrances on the storage tank. With version 14, it is possible for users to incrementally insulate certain nodes of stratified storage tanks by the specification of additional parameters. To utilize the incremental loss coefficients, the user must set the parameter 32 to 1 and specify the incremental loss coefficients. (This must be done for every node in the tank, even if some nodes have an additional loss coefficient of zero). The loss coefficient for the ith node is then:

$$U_i = U_{tank} + \Delta U_i$$

ΔU_i can be positive or negative, just as long as $U_i \geq 0$.

A pressure relief valve has been added to the storage tank to account for boiling effects. The user must specify the boiling temperature of the fluid; venting will release sufficient energy to keep the tank at the boiling temperature. The loss of mass due to venting is neglected. Energy lost to boiling will be added to the energy lost to the environment. (Output 7)

The model allows for tanks with a gas auxiliary heater. The model treats gas auxiliary energy the same as electric energy, so the heat rate when the burner is firing is the same as if it was electric power. When the burner is on, no energy is lost to the flue. When the burner is off, energy is lost to the flue at a temperature T_{flue}, which is entered as a parameter. The user specifies the overall conductance for heat loss to the flue and the flue temperature.

Because a tank could have a gas flue but be heated strictly with electric, the user must differentiate between electric and gas auxiliary heat. If gas heat is to be used, (only one heater may be gas) it must be auxiliary heater #2, and parameter 29 (GasAux Mode) must be set to 1. If both elements are electric, paramter 29 is set to 0.

The model is capable of modeling tanks of any shape as long as the cross-sectional area is uniform. The user must enter the tank's height, volume and perimeter. If the tank is a cylinder, the user may enter a perimeter of -1 and the model will compute the perimeter automatically.

The Type 60 tank can also model horizontal cylindrical tanks. The user must enter a perimeter of -2. The model automatically divides the tank into N equal *volume* segments. If the user desires nodes of equal height, they must be entered manually (Hmode=1). Keep in mind that with this option, the tank's height will also be its diameter. All heights are still measured from the floor up.

The model includes the option for up to 3 internal heat exchangers. The user must specify the dimensions of the heat exchangers, as well as the temperature and flow rate of the inside fluid. The inside fluid can be water, water/propylene glycol mixture or water/ethylene glycol mixture. Two wall conductivity parameters are used: one for the conductivity of the heat exchanger material itself, and one for the conductivity of the heat exchanger wall, which allows for contact resistance. Contact resistance would be present with a double wall heat exchanger where the heat exchanger is a tube-within-a-tube design. Refer to the Mathematical Description section for details of the equations used.

To model de-stratification due to mixing at node interfaces and conduction along the tank wall, the user may enter an additional conductivity parameter Δk. The additional conductivity term is added to the conductivity of the tank fluid and is applied to all nodes. For rough calculations, one may calculate Δk as:

$$\Delta k = k_{tankwall} \frac{A_{c,tankwall}}{A_{c,water}} \qquad (4.3.1.1)$$

Where $k_{tank\ wall}$ is the thermal conductivity of the tank wall and A_c is the cross-sectional area.

To minimize errors, the Type60 tank model uses its own internal time steps. This has the advantage of results being unaffected by the size of the TRNSYS time step. The model internally computes the critical Euler time step. The user choose the size of the internal time step by entering the fraction of the critical Euler time step. This has the effect of increasing either the model's speed or accuracy. The table below shows the relative effect of cutting the critical time step.

Table 4.3.1.1 Model time step size's effect on accuracy

Fraction of critical	Iterations / time step	Total iterations required	%error
1/1*10^7	1.0	10,000,000	0.000
1/100	3.0	300	0.002
1/50	3.0	150	0.007
1/20	4.0	80	0.048
1/10	4.8	48	0.192
1/6	5.5	33	0.536
1/4	6.5	26	1.221
1/3	7.7	23	2.199
1/2	10.0	20	5.154
1	21.0	21	26.622

The user specifies parameter 28 (Crit Fraction) as the denominator in the first column of the table shown above. For example to use time steps that are 1/6 the critical Euler time step size, enter parameter 28 as 6.

Nomenclature (for variables not defined in Component Configuration section)

$A_{c,i}$	cross-sectional area of a node
$A_{s,i}$	surface area of a node
h_o	outside convection coefficient for internal heat exchanger
i	node index. i=1 is the top node, i=N is the bottom node
$\dot{m}_{down}$	bulk fluid flow rate down the tank
M_i	mass of node i
$\dot{m}_{up}$	bulk fluid flow rate up the tank
N	number of nodes
Nu_D	Nusselt number for external flow around a tube
T_{statN}	Temperature of the node containing the thermostat
t	time
$\Delta x_{i+1\rightarrow i}$	center-to-center distance between node i and the node below it
$\Delta x_{i-1\rightarrow i}$	center-to-center distance between node i and the node above it

Mathematical Description

An energy balance written about the ith tank segment is expressed as:

$$(M_i C_p)\frac{dT_i}{dt} = \frac{(k+\Delta k)A_{c,i}}{\Delta x_{i+1\to i}}(T_{i+1}-T_i)$$

$$+\frac{(k+\Delta k)A_{c,i}}{\Delta x_{i-1\to i}}(T_{i-1}-T_i)+(U_{tank}+\Delta U_i)A_{s,i}(T_{env}-T_i)$$

$$+UA_{flue,i}(T_{flue}-T_i)+\dot{m}_{down}C_p(T_{i-1})-\dot{m}_{up}C_p(T_i)$$

$$-\dot{m}_{down}C_p(T_i)-\dot{m}_{up}C_p(T_{i+1})+\gamma_{htr1}\dot{Q}_{aux1}+\gamma_{htr2}\dot{Q}_{aux2}$$

$$+UA_{hx1}(lmtd_1)+UA_{hx2}(lmtd_2)+UA_{hx3}(lmtd_3)$$

$$+\dot{m}_{1in}C_p T_{1in}-\dot{m}_{1out}C_p T_i+\dot{m}_{2in}T_{2in}-\dot{m}_{2out}C_p T_i \qquad (4.3.1.2)$$

Not all terms in (4.3.1.2) will be non-zero. Only two of the $\dot{m}_{up} \setminus \dot{m}_{down}$ terms can be non-zero at once. Other terms, such as flows in/out and auxiliary heat may occur at different nodes, hence they would be zero for node i also.

The auxiliary heater is off if: $\gamma_{htr} = 0$, the heater was previously off and $T_{statN} \quad (T_{set} - \Delta T_{db})$ or if $T_{statN} \quad T_{set}$. If a master/slave relationship is specified between the two auxiliary heaters, the lower auxiliary heater is also off if the first auxiliary heater is on. The model assumes that energy supplied to the tank from the heater is placed in the tank segment containing the heater, and then temperature inversions are eliminated.

The temperatures of each of the N tank segments are determined by the integration of their time derivatives expressed in the above equation. At the end of each time step, temperature inversions are eliminated by mixing appropriate adjacent nodes.

The UA_{hx} and lmtd of each heat exchanger is determined iteratively. The outside natural convection coefficient h_o is determined from:

$$h_o = \frac{Nu_D(k)}{d_o} \qquad (4.3.1.3)$$

where

$$Nu_D = C\ Ra^n$$

A typical value for C is about 0.5 and n is usually 0.25. These values are generally constant, but can vary slightly for different log-mean temperature differences. Therefore, these values are entered as

INPUTS. If the user has curve-fit information for C and n, they can be incorporated into the model using an equation. (UA_{hx}, lmtd, and T_{out} are OUTPUTS of the model and can be used as part of a curve fitting equation)

When modeling a tank with a heat exchanger coil, the program iterates from the node containing the inlet of the main coil to the node containing the outlet of the main coil. Figure 4.3.1.2 shows an example.

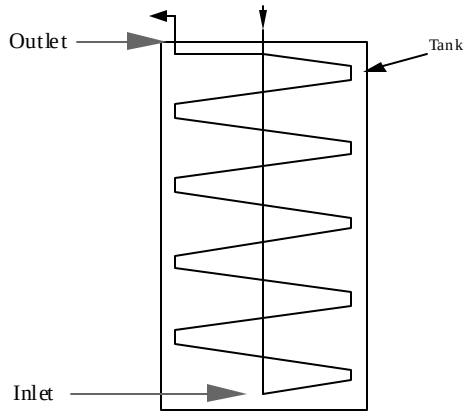
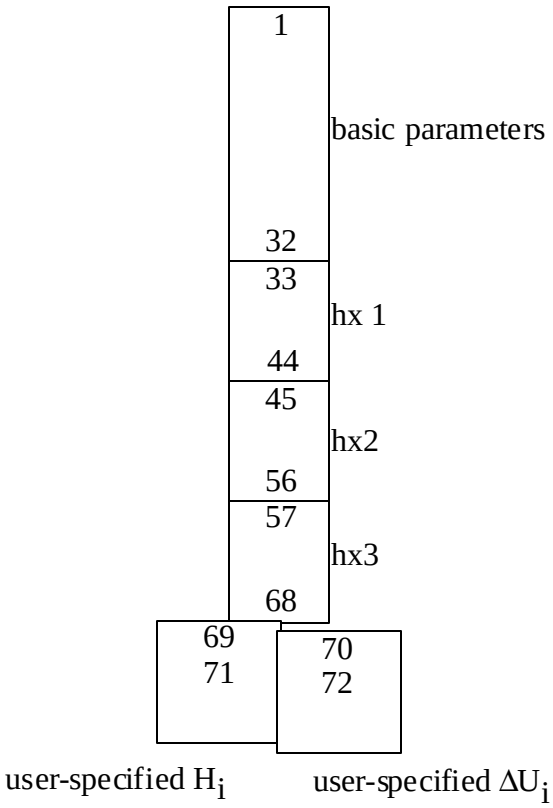


Figure 4.3.1.2 Proper heat exchanger inlet

Although the actual inlet to the heat exchanger is through the top of the tank (shown by the small black arrow), the user must designate the inlet height as shown by the large gray arrow. If the inlet and outlet were both specified as the top of the tank, the model would assume the entire heat exchanger was in the top node.

TRNSYS Component Configuration

Figure 4.3.1.2 graphically depicts the order in which parameters are entered.



Components shown in the diagram that do not exist would drop out and the bottom blocks would move up. For example, if the tank only had 2 heat exchangers,user-specified node heights, and no additional losses, H_1 would be parameter 57, ΔU_1 would be parameter 58 (but would be 0), H_2 would be parameter 59, etc.

Figure 4.3.1.2 Parameter Setup

PARAMETER NO.		DESCRIPTION
1	Mode:	inlet position
		1 = automatic temperature seeking
		2 = fixed at location specified
2	V_{tank} -	tank volume (m^3)
3	H_{tank} -	tank height (m^3)
4	per -	tank perimeter (m)
		<value> = uniform cross-section tank
		-1 = automatic cylindrical vertical tank
		-2 = automatic cylindrical horizontal tank

Note: specifying a perimeter of -2 automatically divides the tank into N equal volume segments. Any user-specified node heights are ignored.

5	H_{1in} -	Height of flow inlet 1 (m)
		<value> = normal
		-1 = non-existent

6	H_{1out}	-	Height of flow outlet 1 (m) <value> = normal -1 = non-existent
7	H_{2in}	-	Height of flow inlet 2 (m) <value> = normal -1 = non-existent
8	H_{2out}	-	Height of flow outlet 2 (m) <value> = normal -1 = non-existent
9	C_p	-	fluid specific heat (kJ/kg-° C)
10	ρ	-	fluid density (kg/m^3)
11	U_{tank}	-	overall tank loss coefficient per unit area ($kJ/hr\text{-}m^2\text{-}°\ C$)
12	k	-	tank fluid thermal conductivity (kJ/hr-m-° C)
13	Δk		de-stratification conductivity (kJ/hr-m-° C)
14	T_{boil}	-	tank boiling temperature (° C)
15	AuxMode		auxiliary heater mode 1 = master / slave relationship 2 = both heaters may be on simultaneously
16	H_{aux1}	-	height of auxiliary heater 1 in tank (m)
17	H_{stat1}	-	height of thermostat for auxiliary heater 1 (m)
18	T_{set1}	-	set point temperature for first auxiliary heater (° C)
19	ΔT_{db1}	-	first auxiliary heater temperature deadband (° C)
20	Q_{aux1}	-	maximum heating rate of first auxiliary heater (kJ/hr)
21	H_{aux2}	-	height of auxiliary heater 2 in tank (m)
22	H_{stat2}	-	height of thermostat for auxiliary heater 2 (m)
23	T_{set2}	-	set point temperature for second auxiliary heater (° C)
24	ΔT_{db2}	-	second auxiliary heater temperature deadband (° C)
25	Q_{aux2}	-	maximum heating rate of second auxiliary heater (kJ/hr)
26	UA_{flue}	-	overall conductance loss coefficient of flue (kJ/hr-° C)
27	T_{flue}	-	flue temperature (° C)
28	Crit Fraction		# of divisions of time step ($1 \leq$ Crit Fraction ≤ 1000)
29	GasAuxMode		gas auxiliary heat mode 0 = both heaters are electric 1 = heater #2 is gas

30	hxMode	-	number of internal heat exchangers ($0 \leq$ hxMode ≤ 3)
31	HMode	-	user-specified node height mode 0 = tank is divided into N equal segments 1 = user must specify each node's height
32	UMode	-	Additional U_{loss} for a particular node 0 = U_{node} is the same as U_{tank} 1 = user must specify additional loss coefficient ΔU for each node

The parameters listed below may or may not exist, depending on the values of parameters 30, 31, and 32. Refer to diagram after this section for a description of how parameters are entered.

The following 12 parameters are repeated for each heat exchanger

33	HX fluid		fluid used inside heat exchanger 1 = water 2 = propylene glycol / water mixture 3 = ethylene glycol / water mixture
34	gly	-	fraction of glycol in heat exchanger fluid 0 < gly < 1 propylene glycol mixtures .55 < gly < .85 ethylene glycol mixtures (ignored if HX fluid = 1)
35	d_i	-	heat exchanger tube inside diameter (m)
36	d_o	-	heat exchanger tube outside diameter (m)
37	d_f	-	heat exchanger fin diameter (d_f=d_o for a smooth tube heat exchanger) (m)
38	A_o	-	total outside surface area of heat exchanger (m^2)
39	fpm	-	fins per meter (ignored for smooth tubes) (fins/m)
40	L	-	heat exchanger length (m)
41	k_{wall}	-	heat exchanger wall conductivity (includes contact resistance) (kJ/hr-m-° C)
42	k_{matl}	-	heat exchanger material conductivity (kJ/hr-m-° C)
43	$H_{hx, in}$		height of heat exchanger inlet (m)
44	$H_{hx,out}$		height of heat exchanger outlet (m)

if HMode=1 or UMode=1

(31+12*hxMode+2*i)	H_i	-	height of node i

(32+12*hxMode+2*i) ΔU_i - additional loss for node i

Note: specifying HMode or UMode as 1 requires that <u>both</u> be entered.

# of parameters = 32+2*N*max(HMode, UMode)+12*hxMode

INPUT NUMBER		DESCRIPTION
1	$\dot{m}_{1in}$	-mass flow rate of entering fluid 1 (kg/hr) -1 = non-existent (entered as a constant) -2 = unknown (entered as a constant)
2	$\dot{m}_{1out}$	-mass flow rate of exiting fluid 1 (kg/hr) -1 = non-existent (entered as a constant) -2 = unknown (entered as a constant)
3	$\dot{m}_{2in}$	-mass flow rate of entering fluid 2 (kg/hr) -1 = non-existent (entered as a constant) -2 = unknown (entered as a constant)
4	$\dot{m}_{2out}$	-mass flow rate of exiting fluid 2 (kg/hr) -1 = non-existent (entered as a constant) -2 = unknown (entered as a constant)

Note: the user must specify j-1 flow rates for j flows. One (and only one) of parameters 1-4 above must be -2.

5	T_{1in}	temperature of entering fluid 1 (° C)
6	T_{2in}	temperature of entering fluid 2 (° C)
7	T_{env}	temperature of environment (° C)
8	$\gamma_{htr,1}$	enable signal for first heating element
9	$\gamma_{htr,2}$	enable signal for second heating element

The following 4 inputs are repeated for each heat exchanger (if hxMode>0)

10	$\dot{m}_{hx\,in}$	mass flow rate of fluid entering heat exchanger (kg/hr)
11	$T_{hx\,in}$	temperature of fluid entering heat exchanger (° C)
12	C	Nusselt number constant
13	n	Nusselt number exponent

#inputs = 9+4*hxMode

OUTPUT NUMBER			DESCRIPTION
1	$\dot{m}_{1in}$	-	inlet 1 flow rate (kg/hr)
2	$\dot{m}_{1out}$	-	outlet 1 flow rate (kg/hr)
3	$\dot{m}_{2in}$	-	inlet 2 flow rate (kg/hr)
4	$\dot{m}_{2out}$	-	outlet 2 flow rate (kg/hr)
5	T_{1out}	-	temperature of outlet flow 1 (° C)
6	T_{2out}	-	temperature of outlet flow 2 (° C)
7	$\dot{Q}_{env}$	-	rate of energy loss to the environment (kJ/hr)
8	$\dot{Q}_{1in}$	-	rate of energy supplied through flow 1 (kJ/hr)
9	$\dot{Q}_{1out}$	-	rate of energy leaving through flow 1 (kJ/hr)
10	$\dot{Q}_{2in}$	-	rate of energy supplied through flow 2 (kJ/hr)
11	$\dot{Q}_{2out}$	-	rate of energy leaving through flow 2 (kJ/hr)
12	Q_{aux}	-	rate at which auxiliary energy is added by both auxiliary heaters (kJ/hr)
13	$Q_{aux,1}$	-	rate at which auxiliary energy is added by the first auxiliary heater (kJ/hr)
14	$Q_{aux,2}$	-	rate at which auxiliary energy is added by the second auxiliary heater (kJ/hr)
15	$\dot{Q}_{flue}$	-	rate of energy loss through the flue (kJ/hr)
16	ΔE	-	internal energy change of the tank from the beginning of the simulation (kJ)
17	$\overline{T}$	-	average storage temperature (° C)
18	ΔP_{1in}	-	static pressure difference between top of tank and inlet flow 1 (kPa)
19	ΔP_{1out}		static pressure difference between top of tank and outlet flow 1 (kPa)
20	ΔP_{2in}	-	static pressure difference between top of tank and inlet flow 2 (kPa)
21	ΔP_{2out}		static pressure difference between top of tank and outlet flow 2 (kPa)

The following 5 outputs are repeated for each heat exchanger (if hxMode>0)

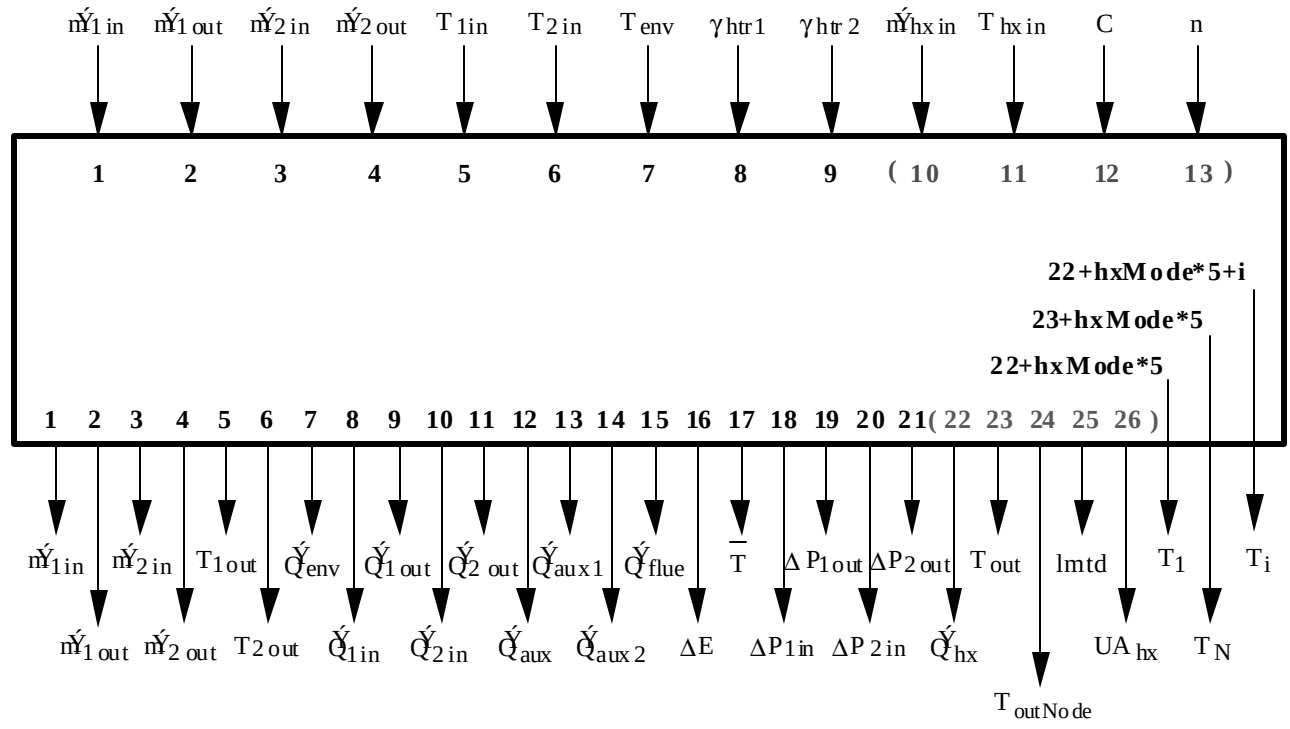
22	Q_{hx}	-	rate of energy input from heat exchanger (kJ/hr)
23	T_{out}	-	temperature of fluid exiting heat exchanger (° C)
23	$T_{out\ Node}$		temperature of node containing heat exchanger outlet (C)
25	lmtd	-	average heat exchanger log-mean temperature difference (° C)
26	UA_{hx}	-	average heat exchanger UA value (kJ/hr-° C)

22+hxMode*5	T_1		-time-averaged temperature of the top node (° C)
23+hxMode*5	T_N		-time-averaged temperature of the bottom node (° C)
22+hxMode*5+i	T_i	-	time-averaged temperature of the ith node (° C)

DERIVATIVE NO.	DESCRIPTION		
1	T_1	-	initial temperature of the fluid in the top tank segment (° C)
.	.		
.	.		
i	T_i	-	initial temperature of the fluid in the ith tank segment (° C)
.	.		
.	.		
N	T_N	-	initial temperature of the fluid in the Nth tank segment (° C)

Note: the model obtains the number of nodes from the number of derivatives

Information Flow Diagram



Reference

l. Klein, S.A., "A Design Procedure for Solar Heating Systems", Ph.D. Thesis, Department of Chemical Engineering, University of Wisconsin-Madison, (1976)

C2 Eigenvalue / Eigenvector Subroutines{ TC "C2 Eigenvalue / Eigenvector Subroutines" \l 1 }

```
c     ************************************************************************
      subroutine eigen(a,n,eigval,eigvec,z)
c
c     THIS SUBROUTINE COMPUTES THE EIGENVALUES AND EIGENVECTORS
c     OF THE MATRIX a
c
      integer n,nm,matz,ierr
```

```
      double precision a(15,15),wr(15),wi(15),z(15,15),fv1(15)
      double precision eigval(15),eigvec(15,15),atemp(15,15)
      integer iv1(15)
      nm=n
      matz=1
      do 13 i=1,n
         do 12 j=1,n
            atemp(i,j)=a(i,j)
 12      continue
 13   continue

      call rg(nm,n,a,wr,wi,matz,z,iv1,fv1,ierr)

      do 15 i=1,n
         do 14 j=1,n
            a(i,j)=atemp(i,j)
            eigvec(i,j)=z(i,j)
 14      continue
         eigval(i)=wr(i)
 15   continue

      return
      end

c     ************************************************************************

      subroutine inv(a,ainv,n)
c
c     THIS SUBROUTINE COMPUTES THE INVERSE OF MATRIX a
c
      double precision a(15,15),ainv(15,15),indx(15),atemp(15,15)
      np=n
      do 10 i=1,n
         do 9 j=1,n
            atemp(i,j)=a(i,j)
 9       continue
 10   continue
      do 12 i=1,n
         do 11 j=1,n
            ainv(i,j)=0.0
 11      continue
         ainv(i,i)=1.
 12   continue

      call ludcmp(a,n,np,indx,d)

      do 13 j=1,n
         call lubksb(a,n,np,indx,ainv(1,j))
 13   continue
      do 15 i=1,n
         do 14 j=1,n
            a(i,j)=atemp(i,j)
 14      continue
 15   continue
```

```
      return
      end

c     ************************************************************************

      subroutine mult(rowsA,colsA,colsB,a,b,c)
c
c     THIS SUBROUTINE MULTIPLIES MATRIX a BY MATRIX b AND RETURNS MATRIX c
c
      integer rowsA,colsA,colsB,i,k,s
      double precision sum,a(15,15),b(15,15),c(15,15)

      do 100 i=1,rowsA
         do 200 k=1,colsB
            sum=0.
            do 300 s=1,colsA
               sum=sum+a(i,s)*b(s,k)
 300        continue
            c(i,k)=sum
 200     continue
 100  continue

      return
      end

c     ************************************************************************
c     ************************************************************************
c     The subroutines below are "support" for the three subroutines above,
c     eigen, inv, and mult.

c     Subroutines for computing the inverse of a matrix taken from "Numerical
c     Recipes, The Art of Scientific Computing," W.H. Press, B.P. Flannery,
c     S.A. Teukolshy, and W.T. Vetterling, Cambridge University Press, 1989,
c     New York, pp 35-38.
c
c     Subroutines for computing the eigenvectors and eigenvalues of a matrix
c     obtained from netlib@ornl.gov  To save space, introductory comments
c     from the netlib subroutines have been erased.
c     ************************************************************************
c     ************************************************************************

      subroutine rg(nm,n,a,wr,wi,matz,z,iv1,fv1,ierr)
c
      integer n,nm,is1,is2,ierr,matz
      double precision a(15,15),wr(15),wi(15),z(15,15),fv1(15)
      integer iv1(15)
      if (n .le. nm) go to 10
      ierr = 10 * n
      go to 50
c
   10 call  balanc(nm,n,a,is1,is2,fv1)
      call  elmhes(nm,n,is1,is2,a,iv1)
c     .......... find both eigenvalues and eigenvectors ..........
```

```
   20 call  eltran(nm,n,is1,is2,a,iv1,z)
      call  hqr2(nm,n,is1,is2,a,wr,wi,z,ierr)
      if (ierr .ne. 0) go to 50
      call  balbak(nm,n,is1,is2,fv1,n,z)
   50 return
      end

c     ************************************************************************

      subroutine balanc(nm,n,a,low,igh,scale)
c
      integer i,j,k,l,m,n,jj,nm,igh,low,iexc
      double precision a(15,15),scale(15)
      double precision c,f,g,r,s,b2,radix
      logical noconv

      radix = 16.0d0
c
      b2 = radix * radix
      k = 1
      l = n
      go to 100
c     ..... in-line procedure for row and
c                column exchange .....
   20 scale(m) = j
      if (j .eq. m) go to 50
c
      do 30 i = 1, l
         f = a(i,j)
         a(i,j) = a(i,m)
         a(i,m) = f
   30 continue
c
      do 40 i = k, n
         f = a(j,i)
         a(j,i) = a(m,i)
         a(m,i) = f
   40 continue
c
   50 go to (80,130), iexc
c     ..... search for rows isolating an eigenvalue
c                and push them down .....
   80 if (l .eq. 1) go to 280
      l = l - 1
c     ..... for j=l step -1 until 1 do -- .....
  100 do 120 jj = 1, l
         j = l + 1 - jj
c
         do 110 i = 1, l
            if (i .eq. j) go to 110
            if (a(j,i) .ne. 0.0d0) go to 120
  110    continue
c
         m = l
```

```
         iexc = 1
         go to 20
  120 continue
c
      go to 140
c     .......... search for columns isolating an eigenvalue
c                and push them left ..........
  130 k = k + 1
c
  140 do 170 j = k, l
c
         do 150 i = k, l
            if (i .eq. j) go to 150
            if (a(i,j) .ne. 0.0d0) go to 170
  150    continue
c
         m = k
         iexc = 2
         go to 20
  170 continue
c     .......... now balance the submatrix in rows k to l ..........
      do 180 i = k, l
  180 scale(i) = 1.0d0
c     .......... iterative loop for norm reduction ..........
  190 noconv = .false.
c
      do 270 i = k, l
         c = 0.0d0
         r = 0.0d0
c
         do 200 j = k, l
            if (j .eq. i) go to 200
            c = c + dabs(a(j,i))
            r = r + dabs(a(i,j))
  200    continue
c     .......... guard against zero c or r due to underflow ..........
         if (c .eq. 0.0d0 .or. r .eq. 0.0d0) go to 270
         g = r / radix
         f = 1.0d0
         s = c + r
  210    if (c .ge. g) go to 220
         f = f * radix
         c = c * b2
         go to 210
  220    g = r * radix
  230    if (c .lt. g) go to 240
         f = f / radix
         c = c / b2
         go to 230
c     .......... now balance ..........
  240    if ((c + r) / f .ge. 0.95d0 * s) go to 270
         g = 1.0d0 / f
         scale(i) = scale(i) * f
         noconv = .true.
```

```
c
         do 250 j = k, n
  250    a(i,j) = a(i,j) * g
c
         do 260 j = 1, l
  260    a(j,i) = a(j,i) * f
c
  270 continue
c
      if (noconv) go to 190
c
  280 low = k
      igh = l
      return
      end

c     ************************************************************************

      subroutine balbak(nm,n,low,igh,scale,m,z)
c
      integer i,j,k,m,n,ii,nm,igh,low
      double precision scale(15),z(15,15)
      double precision s

      if (m .eq. 0) go to 200
      if (igh .eq. low) go to 120
c
      do 110 i = low, igh
         s = scale(i)
c     ..........  left hand eigenvectors are back transformed
c                 if the foregoing statement is replaced by
c                 s=1.0d0/scale(i). ..........
         do 100 j = 1, m
  100    z(i,j) = z(i,j) * s
c
  110 continue
c     .........  for i=low-1 step -1 until 1,
c               igh+1 step 1 until n do -- ..........
  120 do 140 ii = 1, n
         i = ii
         if (i .ge. low .and. i .le. igh) go to 140
         if (i .lt. low) i = low - ii
         k = scale(i)
         if (k .eq. i) go to 140
c
         do 130 j = 1, m
            s = z(i,j)
            z(i,j) = z(k,j)
            z(k,j) = s
  130    continue
c
  140 continue
c
  200 return
```

```
      end

c     ************************************************************************

      subroutine cdiv(ar,ai,br,bi,cr,ci)
      double precision ar,ai,br,bi,cr,ci
c
c     complex division, (cr,ci) = (ar,ai)/(br,bi)
c
      double precision s,ars,ais,brs,bis
      s = dabs(br) + dabs(bi)
      ars = ar/s
      ais = ai/s
      brs = br/s
      bis = bi/s
      s = brs**2 + bis**2
      cr = (ars*brs + ais*bis)/s
      ci = (ais*brs - ars*bis)/s
      return
      end

c     ************************************************************************

      subroutine elmhes(nm,n,low,igh,a,int)
c
      integer i,j,m,n,la,nm,igh,kp1,low,mm1,mp1
      double precision a(15,15)
      double precision x,y
      integer int(igh)

      la = igh - 1
      kp1 = low + 1
      if (la .lt. kp1) go to 200
c
      do 180 m = kp1, la
         mm1 = m - 1
         x = 0.0d0
         i = m
c
         do 100 j = m, igh
            if (dabs(a(j,mm1)) .le. dabs(x)) go to 100
            x = a(j,mm1)
            i = j
  100    continue
c
         int(m) = i
         if (i .eq. m) go to 130
c     ..........  interchange rows and columns of a ..........
         do 110 j = mm1, n
            y = a(i,j)
            a(i,j) = a(m,j)
            a(m,j) = y
  110    continue
c
```

```
         do 120 j = 1, igh
            y = a(j,i)
            a(j,i) = a(j,m)
            a(j,m) = y
  120    continue
c     .......... end interchange ..........
  130    if (x .eq. 0.0d0) go to 180
         mp1 = m + 1
c
         do 160 i = mp1, igh
            y = a(i,mm1)
            if (y .eq. 0.0d0) go to 160
            y = y / x
            a(i,mm1) = y
c
            do 140 j = m, n
  140       a(i,j) = a(i,j) - y * a(m,j)
c
            do 150 j = 1, igh
  150       a(j,m) = a(j,m) + y * a(j,i)
c
  160    continue
c
  180 continue
c
  200 return
      end

c     ***********************************************************************

      subroutine eltran(nm,n,low,igh,a,int,z)
c
      integer i,j,n,kl,mm,mp,nm,igh,low,mp1
      double precision a(15,15),z(15,15)
      integer int(15)

      do 80 j = 1, n
c
         do 60 i = 1, n
   60    z(i,j) = 0.0d0
c
         z(j,j) = 1.0d0
   80 continue
c
      kl = igh - low - 1
      if (kl .lt. 1) go to 200
c     .......... for mp=igh-1 step -1 until low+1 do -- ..........
      do 140 mm = 1, kl
         mp = igh - mm
         mp1 = mp + 1
c
         do 100 i = mp1, igh
  100    z(i,mp) = a(i,mp-1)
c
```

```
         i = int(mp)
         if (i .eq. mp) go to 140
c
         do 130 j = mp, igh
            z(mp,j) = z(i,j)
            z(i,j) = 0.0d0
  130    continue
c
         z(i,mp) = 1.0d0
  140 continue
c
  200 return
      end

c     ************************************************************************

      subroutine hqr2(nm,n,low,igh,h,wr,wi,z,ierr)
c
      integer i,j,k,l,m,n,en,ii,jj,ll,mm,na,nm,nn,
     x        igh,itn,its,low,mp2,enm2,ierr
      double precision h(15,15),wr(15),wi(15),z(15,15)
      double precision p,q,r,s,t,w,x,y,ra,sa,vi,vr,zz,norm,tst1,tst2
      logical notlas

      ierr = 0
      norm = 0.0d0
      k = 1
c     ......... store roots isolated by balanc
c                and compute matrix norm .........
      do 50 i = 1, n
c
         do 40 j = k, n
   40    norm = norm + dabs(h(i,j))
c
         k = i
         if (i .ge. low .and. i .le. igh) go to 50
         wr(i) = h(i,i)
         wi(i) = 0.0d0
   50 continue
c
      en = igh
      t = 0.0d0
      itn = 30*n
c     ......... search for next eigenvalues .........
   60 if (en .lt. low) go to 340
      its = 0
      na = en - 1
      enm2 = na - 1
c     ......... look for single small sub-diagonal element
c                for l=en step -1 until low do -- .........
   70 do 80 ll = low, en
         l = en + low - ll
         if (l .eq. low) go to 100
         s = dabs(h(l-1,l-1)) + dabs(h(l,l))
```

```
         if (s .eq. 0.0d0) s = norm
         tst1 = s
         tst2 = tst1 + dabs(h(l,l-1))
         if (tst2 .eq. tst1) go to 100
   80 continue
c     .......... form shift ..........
  100 x = h(en,en)
      if (l .eq. en) go to 270
      y = h(na,na)
      w = h(en,na) * h(na,en)
      if (l .eq. na) go to 280
      if (itn .eq. 0) go to 1000
      if (its .ne. 10 .and. its .ne. 20) go to 130
c     .......... form exceptional shift ..........
      t = t + x
c
      do 120 i = low, en
  120 h(i,i) = h(i,i) - x
c
      s = dabs(h(en,na)) + dabs(h(na,enm2))
      x = 0.75d0 * s
      y = x
      w = -0.4375d0 * s * s
  130 its = its + 1
      itn = itn - 1
c     .......... look for two consecutive small
c                sub-diagonal elements.
c                for m=en-2 step -1 until l do -- ..........
      do 140 mm = l, enm2
         m = enm2 + l - mm
         zz = h(m,m)
         r = x - zz
         s = y - zz
         p = (r * s - w) / h(m+1,m) + h(m,m+1)
         q = h(m+1,m+1) - zz - r - s
         r = h(m+2,m+1)
         s = dabs(p) + dabs(q) + dabs(r)
         p = p / s
         q = q / s
         r = r / s
         if (m .eq. l) go to 150
         tst1 = dabs(p)*(dabs(h(m-1,m-1)) + dabs(zz) + dabs(h(m+1,m+1)))
         tst2 = tst1 + dabs(h(m,m-1))*(dabs(q) + dabs(r))
         if (tst2 .eq. tst1) go to 150
  140 continue
c
  150 mp2 = m + 2
c
      do 160 i = mp2, en
         h(i,i-2) = 0.0d0
         if (i .eq. mp2) go to 160
         h(i,i-3) = 0.0d0
  160 continue
c     .......... double qr step involving rows l to en and
```

```
c               columns m to en ..........
      do 260 k = m, na
         notlas = k .ne. na
         if (k .eq. m) go to 170
         p = h(k,k-1)
         q = h(k+1,k-1)
         r = 0.0d0
         if (notlas) r = h(k+2,k-1)
         x = dabs(p) + dabs(q) + dabs(r)
         if (x .eq. 0.0d0) go to 260
         p = p / x
         q = q / x
         r = r / x
  170    s = dsign(dsqrt(p*p+q*q+r*r),p)
         if (k .eq. m) go to 180
         h(k,k-1) = -s * x
         go to 190
  180    if (l .ne. m) h(k,k-1) = -h(k,k-1)
  190    p = p + s
         x = p / s
         y = q / s
         zz = r / s
         q = q / p
         r = r / p
         if (notlas) go to 225
c     .......... row modification ..........
         do 200 j = k, n
            p = h(k,j) + q * h(k+1,j)
            h(k,j) = h(k,j) - p * x
            h(k+1,j) = h(k+1,j) - p * y
  200    continue
c
         j = min0(en,k+3)
c     .......... column modification ..........
         do 210 i = 1, j
            p = x * h(i,k) + y * h(i,k+1)
            h(i,k) = h(i,k) - p
            h(i,k+1) = h(i,k+1) - p * q
  210    continue
c     .......... accumulate transformations ..........
         do 220 i = low, igh
            p = x * z(i,k) + y * z(i,k+1)
            z(i,k) = z(i,k) - p
            z(i,k+1) = z(i,k+1) - p * q
  220    continue
         go to 255
  225    continue
c     .......... row modification ..........
         do 230 j = k, n
            p = h(k,j) + q * h(k+1,j) + r * h(k+2,j)
            h(k,j) = h(k,j) - p * x
            h(k+1,j) = h(k+1,j) - p * y
            h(k+2,j) = h(k+2,j) - p * zz
  230    continue
```

```
c
         j = min0(en,k+3)
c     .......... column modification ..........
         do 240 i = 1, j
            p = x * h(i,k) + y * h(i,k+1) + zz * h(i,k+2)
            h(i,k) = h(i,k) - p
            h(i,k+1) = h(i,k+1) - p * q
            h(i,k+2) = h(i,k+2) - p * r
  240    continue
c     .......... accumulate transformations ..........
         do 250 i = low, igh
            p = x * z(i,k) + y * z(i,k+1) + zz * z(i,k+2)
            z(i,k) = z(i,k) - p
            z(i,k+1) = z(i,k+1) - p * q
            z(i,k+2) = z(i,k+2) - p * r
  250    continue
  255    continue
c
  260 continue
c
       go to 70
c     .......... one root found ..........
  270 h(en,en) = x + t
       wr(en) = h(en,en)
       wi(en) = 0.0d0
       en = na
       go to 60
c     .......... two roots found ..........
  280 p = (y - x) / 2.0d0
       q = p * p + w
       zz = dsqrt(dabs(q))
       h(en,en) = x + t
       x = h(en,en)
       h(na,na) = y + t
       if (q .lt. 0.0d0) go to 320
c     .......... real pair ..........
       zz = p + dsign(zz,p)
       wr(na) = x + zz
       wr(en) = wr(na)
       if (zz .ne. 0.0d0) wr(en) = x - w / zz
       wi(na) = 0.0d0
       wi(en) = 0.0d0
       x = h(en,na)
       s = dabs(x) + dabs(zz)
       p = x / s
       q = zz / s
       r = dsqrt(p*p+q*q)
       p = p / r
       q = q / r
c     .......... row modification ..........
       do 290 j = na, n
          zz = h(na,j)
          h(na,j) = q * zz + p * h(en,j)
          h(en,j) = q * h(en,j) - p * zz
```

```
  290 continue
c     .......... column modification ..........
      do 300 i = 1, en
         zz = h(i,na)
         h(i,na) = q * zz + p * h(i,en)
         h(i,en) = q * h(i,en) - p * zz
  300 continue
c     .......... accumulate transformations ..........
      do 310 i = low, igh
         zz = z(i,na)
         z(i,na) = q * zz + p * z(i,en)
         z(i,en) = q * z(i,en) - p * zz
  310 continue
c
      go to 330
c     .......... complex pair ..........
  320 wr(na) = x + p
      wr(en) = x + p
      wi(na) = zz
      wi(en) = -zz
  330 en = enm2
      go to 60
c     .......... all roots found.  backsubstitute to find
c                vectors of upper triangular form ..........
  340 if (norm .eq. 0.0d0) go to 1001
c     .......... for en=n step -1 until 1 do -- ..........
      do 800 nn = 1, n
         en = n + 1 - nn
         p = wr(en)
         q = wi(en)
         na = en - 1
         if (q) 710, 600, 800
c     .......... real vector ..........
  600    m = en
         h(en,en) = 1.0d0
         if (na .eq. 0) go to 800
c     .......... for i=en-1 step -1 until 1 do -- ..........
         do 700 ii = 1, na
            i = en - ii
            w = h(i,i) - p
            r = 0.0d0
c
            do 610 j = m, en
  610       r = r + h(i,j) * h(j,en)
c
            if (wi(i) .ge. 0.0d0) go to 630
            zz = w
            s = r
            go to 700
  630       m = i
            if (wi(i) .ne. 0.0d0) go to 640
            t = w
            if (t .ne. 0.0d0) go to 635
               tst1 = norm
```

```
               t = tst1
  632          t = 0.01d0 * t
               tst2 = norm + t
               if (tst2 .gt. tst1) go to 632
  635       h(i,en) = -r / t
            go to 680
c     ..... solve real equations .....
  640       x = h(i,i+1)
            y = h(i+1,i)
            q = (wr(i) - p) * (wr(i) - p) + wi(i) * wi(i)
            t = (x * s - zz * r) / q
            h(i,en) = t
            if (dabs(x) .le. dabs(zz)) go to 650
            h(i+1,en) = (-r - w * t) / x
            go to 680
  650       h(i+1,en) = (-s - y * t) / zz
c
c     ..... overflow control .....
  680       t = dabs(h(i,en))
            if (t .eq. 0.0d0) go to 700
            tst1 = t
            tst2 = tst1 + 1.0d0/tst1
            if (tst2 .gt. tst1) go to 700
            do 690 j = i, en
               h(j,en) = h(j,en)/t
  690       continue
c
  700    continue
c     ..... end real vector .....
         go to 800
c     ..... complex vector .....
  710    m = na
c     ..... last vector component chosen imaginary so that
c                eigenvector matrix is triangular .....
         if (dabs(h(en,na)) .le. dabs(h(na,en))) go to 720
         h(na,na) = q / h(en,na)
         h(na,en) = -(h(en,en) - p) / h(en,na)
         go to 730
  720    call cdiv(0.0d0,-h(na,en),h(na,na)-p,q,h(na,na),h(na,en))
  730    h(en,na) = 0.0d0
         h(en,en) = 1.0d0
         enm2 = na - 1
         if (enm2 .eq. 0) go to 800
c     ..... for i=en-2 step -1 until 1 do -- .....
         do 795 ii = 1, enm2
            i = na - ii
            w = h(i,i) - p
            ra = 0.0d0
            sa = 0.0d0
c
            do 760 j = m, en
               ra = ra + h(i,j) * h(j,na)
               sa = sa + h(i,j) * h(j,en)
  760       continue
```

```
c
            if (wi(i) .ge. 0.0d0) go to 770
            zz = w
            r = ra
            s = sa
            go to 795
  770       m = i
            if (wi(i) .ne. 0.0d0) go to 780
            call cdiv(-ra,-sa,w,q,h(i,na),h(i,en))
            go to 790
c     ..... solve complex equations .....
  780       x = h(i,i+1)
            y = h(i+1,i)
            vr = (wr(i) - p) * (wr(i) - p) + wi(i) * wi(i) - q * q
            vi = (wr(i) - p) * 2.0d0 * q
            if (vr .ne. 0.0d0 .or. vi .ne. 0.0d0) go to 784
               tst1 = norm * (dabs(w) + dabs(q) + dabs(x)
     x                      + dabs(y) + dabs(zz))
               vr = tst1
  783          vr = 0.01d0 * vr
               tst2 = tst1 + vr
               if (tst2 .gt. tst1) go to 783
  784       call cdiv(x*r-zz*ra+q*sa,x*s-zz*sa-q*ra,vr,vi,
     x                h(i,na),h(i,en))
            if (dabs(x) .le. dabs(zz) + dabs(q)) go to 785
            h(i+1,na) = (-ra - w * h(i,na) + q * h(i,en)) / x
            h(i+1,en) = (-sa - w * h(i,en) - q * h(i,na)) / x
            go to 790
  785       call cdiv(-r-y*h(i,na),-s-y*h(i,en),zz,q,
     x                h(i+1,na),h(i+1,en))
c
c     ..... overflow control .....
  790       t = dmax1(dabs(h(i,na)), dabs(h(i,en)))
            if (t .eq. 0.0d0) go to 795
            tst1 = t
            tst2 = tst1 + 1.0d0/tst1
            if (tst2 .gt. tst1) go to 795
            do 792 j = i, en
               h(j,na) = h(j,na)/t
               h(j,en) = h(j,en)/t
  792       continue
c
  795    continue
c     ..... end complex vector .....
  800 continue
c     ..... end back substitution.
c                vectors of isolated roots .....
      do 840 i = 1, n
         if (i .ge. low .and. i .le. igh) go to 840
c
         do 820 j = i, n
  820    z(i,j) = h(i,j)
c
  840 continue
```

```
c     ......... multiply by transformation matrix to give
c                vectors of original full matrix.
c                for j=n step -1 until low do -- .........
      do 880 jj = low, n
         j = n + low - jj
         m = min0(j,igh)
c
         do 880 i = low, igh
            zz = 0.0d0
c
            do 860 k = low, m
  860       zz = zz + z(i,k) * h(k,j)
c
            z(i,j) = zz
  880 continue
c
      go to 1001
c     ......... set error -- all eigenvalues have not
c                converged after 30*n iterations .........
 1000 ierr = en
 1001 return
      end

c     ***********************************************************************

      subroutine ludcmp(a,n,np,indx,d)

      parameter (nmax=100, tiny=1.0e-20)
      double precision a(15,15),indx(15),vv(15)

      d=1.
      do 12 i=1,n
         aamax=0.
         do 11 j=1,n
            if(abs(a(i,j)).gt.aamax) aamax=abs(a(i,j))
 11      continue
         if (aamax.eq.0.) pause 'singular matrix'
         vv(i)=1./aamax
 12   continue
      do 19 j=1,n
         do 14 i=1,j-1
            sum=a(i,j)
            do 13 k=1,i-1
               sum=sum-a(i,k)*a(k,j)
 13         continue
            a(i,j)=sum
 14      continue
         aamax=0.
         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.0.) a(j,j)=tiny
          if(j.ne.n) then
             dum=1./a(j,j)
             do 18 i=j+1,n
                a(i,j)=a(i,j)*dum
 18          continue
          endif
 19    continue
       return
       end

c      *********************************************************************

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

       double precision a(15,15),indx(15),bb(15)

       ii=0
       do 12 i=1,n
          ll=indx(i)
          sum=bb(ll)
          bb(ll)=bb(i)
          if (ii.ne.0) then
             do 11 j=ii,i-1
                sum=sum-a(i,j)*bb(j)
 11          continue
          elseif(sum.ne.0) then
             ii=i
          endif
          bb(i)=sum
 12    continue
       do 14 i=n,1,-1
          sum=bb(i)
          do 13 j=i+1,n
             sum=sum-a(i,j)*bb(j)
 13       continue
          bb(i)=sum/a(i,i)
 14    continue
```

```
      return
      end
```

C3 Solution Method Comparison Program{ TC "C3 Solution Method Comparison Program" \l 1 }

```
c
c     THIS PROGRAM NUMERICALLY COMPUTES THE SOLUTION TO A FIRST-ORDER,
c     CONSTANT-COEFFICIENT, NON-HOMOGENEOUS DIFFERENTIAL EQUATION.
c     THE PURPOSE OF THE PROGRAM IS TO COMPARE THE ACCURACY AND
c     EFFICIENCY OF VARIOUS SOLUTION TECHNIQUES.
c
      double precision a(15,15),wr(15),wi(15),z(15,15),fv1(15)
      double precision eigval(15),eigvec(15,15),atemp(15,15)
      double precision ainv(15,15)
      integer n,nm,matz,ierr
      double precision Tnew(15),T(15),a2(15),aa(15),bb(15)
      double precision TnewX(15),M(15)

      open(unit=1,file='results.out',status='old')
c
c     SET MATRIX COEFFICIENT
c
      a(1,1)=-1.
      a(2,1)=2.
      a(3,1)=0.
      a(1,2)=2.
      a(2,2)=-1.
      a(3,2)=1.
      a(1,3)=0.
      a(2,3)=1.
      a(3,3)=-.5

      a2(1)=1.
      a2(2)=2.
      a2(3)=10.
      n=3

      err=2.

      print*,'enter time step:'
      read(*,*)dt
c
c     CRANK-NICOLSON SOLUTION
c
      do 100 i=1,int(1./dt)
         do 150 while(abs(err).gt..0001)

            Told1=Tnew(1)
```

```
            Told2=Tnew(2)
            Told3=Tnew(3)

            TnewX(1)=dt*(a(1,1)*(T(1)+Tnew(1))/2.+a(1,2)*(T(2)+
     &           Tnew(2))/2.+a2(1))+T(1)
            TnewX(2)=dt*(a(2,1)*(T(1)+Tnew(1))/2.+a(2,2)*(T(2)+
     &           Tnew(2))/2.+a(2,3)*(T(3)+Tnew(3))/2.+a2(2))+T(2)
            TnewX(3)=dt*(a(3,2)*(T(2)+Tnew(2))/2.+a(3,3)*(T(3)+
     &           Tnew(3))/2.+a2(3))+T(3)

            Tnew(1)=TnewX(1)
            Tnew(2)=TnewX(2)
            Tnew(3)=TnewX(3)

            call mix(Tnew)

            ie=ie+1

            err=dmax1(abs(Tnew(1)-Told1),abs(Tnew(2)-Told2),
     &           abs(Tnew(3)-Told3))

 150     continue

         T(1)=Tnew(1)
         T(2)=Tnew(2)
         T(3)=Tnew(3)

         err=2.

 100  continue

      write(1,*)'time step: ',dt
      write(1,*)'solution after time=1'
      write(1,*)

      write(1,*)'Crank-Nicolson'
      write(1,*)T(1)
      write(1,*)T(2)
      write(1,*)T(3)
      write(1,*)'iterations: ',ie
      write(1,*)' '

c
c     DIFFEQ SOLUTION
c
      T(1)=0.
      T(2)=0.
      T(3)=0.
      Tnew(1)=0.
      Tnew(2)=0.
      Tnew(3)=0.
      err=2.

      do 200 i=1,int(1./dt)
```

```
          do 250 while (abs(err).gt..0001)

             Told1=Tnew(1)
             Told2=Tnew(2)
             Told3=Tnew(3)

             aa(1)=a(1,1)
             bb(1)=a(1,2)*(T(2)+Tnew(2))/2.+a2(1)
             TnewX(1)=(T(1)+bb(1)/aa(1))*exp(aa(1)*dt)-bb(1)/aa(1)

             aa(2)=a(2,2)
             bb(2)=a(2,1)*(T(1)+Tnew(1))/2.+a(2,3)*(T(3)+Tnew(3))/2.
     &            +a2(2)
             TnewX(2)=(T(2)+bb(2)/aa(2))*exp(aa(2)*dt)-bb(2)/aa(2)

             aa(3)=a(3,3)
             bb(3)=a(3,2)*(T(2)+Tnew(2))/2.+a2(3)
             TnewX(3)=(T(3)+bb(3)/aa(3))*exp(aa(3)*dt)
     &            -bb(3)/aa(3)

             id=id+1

             Tnew(1)=TnewX(1)
             Tnew(2)=TnewX(2)
             Tnew(3)=TnewX(3)

             call mix(Tnew)

             err=dmax1(abs(Tnew(1)-Told1),abs(Tnew(2)-Told2),
     &            abs(Tnew(3)-Told3))

  250     continue

          T(1)=Tnew(1)
          T(2)=Tnew(2)
          T(3)=Tnew(3)
          err=2.

  200   continue

        write(1,*)'DIFFEQ'
        write(1,*)T(1)
        write(1,*)T(2)
        write(1,*)T(3)
        write(1,*)'iterations: ',id
        write(1,*)' '
c
c       DIFFEQ 2 SOLUTION
c
        T(1)=0.
        T(2)=0.
        T(3)=0.
        Tnew(1)=0.
```

```
      Tnew(2)=0.
      Tnew(3)=0.
      err=2.

      do 275 i=1,int(1./dt)

         aa(1)=a(1,1)
         bb(1)=a(1,2)*T(2)+a2(1)
         Tnew(1)=(T(1)+bb(1)/aa(1))*exp(aa(1)*dt)-bb(1)/aa(1)

         aa(2)=a(2,2)
         bb(2)=a(2,1)*T(1)+a(2,3)*T(3)+a2(2)
         Tnew(2)=(T(2)+bb(2)/aa(2))*exp(aa(2)*dt)-bb(2)/aa(2)

         aa(3)=a(3,3)
         bb(3)=a(3,2)*T(2)+a2(3)
         Tnew(3)=(T(3)+bb(3)/aa(3))*exp(aa(3)*dt)
     &        -bb(3)/aa(3)

         id2=id2+1

         call mix(Tnew)

         T(1)=Tnew(1)
         T(2)=Tnew(2)
         T(3)=Tnew(3)

 275  continue

      write(1,*)'DIFFEQ 2'
      write(1,*)T(1)
      write(1,*)T(2)
      write(1,*)T(3)
      write(1,*)'iterations: ',id2
      write(1,*)' '
c
c     STRAIGHT EULER SOLUTION
c
      T(1)=0.
      T(2)=0.
      T(3)=0.
      Tnew(1)=0.
      Tnew(2)=0.
      Tnew(3)=0.
      err=2.

      do 400 i=1,int(1./dt)

         Tnew(1)=dt*(a(1,1)*T(1)+a(1,2)*T(2)+a2(1))+T(1)
         Tnew(2)=dt*(a(2,1)*T(1)+a(2,2)*T(2)+a(2,3)*T(3)
     &        +a2(2))+T(2)
         Tnew(3)=dt*(a(3,2)*T(2)+a(3,3)*T(3)+a2(3))+T(3)

         call mix(Tnew)
```

```
         T(1)=Tnew(1)
         T(2)=Tnew(2)
         T(3)=Tnew(3)

         ic=ic+1

 400  continue

      write(1,*)'Straight Euler:'
      write(1,*)T(1)
      write(1,*)T(2)
      write(1,*)T(3)
      write(1,*)'iterations: ',ic
      write(1,*)' '

      end


c     ***********************************************************************
      subroutine mix(Tnew)
      double precision Tnew(15),M(15),dum1,dum2,dum3
      integer trip1

      M(1)=1.
      M(2)=1.
      M(3)=1.
      n=3
c
c     IF A TEMPERATURE INVERSION (GREATER THAN .00001 deg C)
c     OCCURS, MIX.  TO ACCOUNT FOR EITHER A COLD NODE SINKING TO
c     TO THE BOTTOM OR A HOT NODE RISING TO THE TOP, THE LOOP
c     "SWEEPS" FIRST FROM THE BOTTOM UP, AND THEN FROM THE TOP
c     DOWN.
c
 30      trip1=0

 do 60 i=n-1,1,-1

    if((Tnew(i)+.00001).ge.Tnew(i+1).and.trip1.ne.0) trip1=0

    if((Tnew(i)+.00001).lt.Tnew(i+1)) then ! MIX

       if(trip1.eq.0) trip1=i+1

       dum2=0.

       dum3=0.

       do 40 i2=trip1,i,-1


  dum2=dum2+M(i2)*Tnew(i2)
```

```
 dum3=dum3+M(i2)
40            continue

      do 50 i2=trip1,i,-1


 Tnew(i2)=dum2/dum3
50            continue

   endif
60      continue
c
c     TOP --> DOWN
c

trip1=0

do 90 i=2,n

   if(Tnew(i).le.(Tnew(i-1)+.00001).and.trip1.ne.0) trip1=0

   if(Tnew(i).gt.(Tnew(i-1)+.00001)) then ! MIX

      if(trip1.eq.0) trip1=i-1

      dum2=0.

      dum3=0.

      do 70 i2=trip1,i


 dum2=dum2+M(i2)*Tnew(i2)


 dum3=dum3+M(i2)
70            continue

      do 80 i2=trip1,i


 Tnew(i2)=dum2/dum3
80            continue
           endif
90      continue


do 100 i=2,n   ! CHECK TO MAKE SURE EVERYTHING GOT MIXED

   if((Tnew(i-1)+.000015).lt.Tnew(i)) goto 30
100     continue

        return
```

```
          end
```

C4 FLUENT Input Deck{ TC "C4 FLUENT Input Deck" \l 1 }

```
 *-------------------------------------------------------------------------------------------------
--------------------------*
 I  LICENSED BY AND THE PROPERTY OF FLUENT INC., CENTERRA RESOURCE PARK, 10 CAVENDISH CT.,
LEBANON, NH 03766.  603-643-2600  I
 I-------------------------------------------------------------------------------------------------
--------------------------I
 I
I
 I  FFFFF  L      U   U  EEEEE  N   N  TTTTT
I
 I  F      L      U   U  E      NN  N    T
I
 I  FFFF   L      U   U  EEEE   N N N    T
I
 I  F      L      U   U  E      N  NN    T
I
 I  F      LLLLL   UUU   EEEEE  N   N    T          F L U I D    F L O W    M O D E L L I N G
I
 I
I
 I-------------------------------------------------------------------------------------------------
--------------------------I
 I  OUTPUT PRODUCED BY VERSION  4.25
I
 *-------------------------------------------------------------------------------------------------
--------------------------*
```

```
    ************************************************************
  *                                                            *
  *            FLUENT (V4.25) Fluid Flow Modeling              *
  *                                                            *
  *     Copyright (C) 1984, 1989, 1991, 1994 by Fluent Inc.    *
```

 - UNITS SYSTEM -

INDEX	PROPERTY	UNITS	S.I. CONVERSION FACTOR
1	DIMENSIONLESS	DIMENSIONLESS	1.000E+00
2	MASS	KILOGRAMS	1.000E+00
3	LENGTH	METERS	1.000E+00
4	TIME	SECONDS	1.000E+00
5	VELOCITY	METERS/SEC	1.000E+00
6	FORCE	NEWTONS	1.000E+00
7	ACCELERATION	METERS/SEC/SEC	1.000E+00
8	ENERGY	JOULES	1.000E+00
9	POWER	WATTS	1.000E+00
10	MASS FLOW RATE	KILOGRAMS/SEC	1.000E+00
11	TEMPERATURE	KELVIN	1.000E+00
12	ENTHALPY	JOULES/KILOGRAM	1.000E+00
13	PRESSURE	PASCALS	1.000E+00
14	DENSITY	KILOGRAMS/CU.M	1.000E+00
15	VISCOSITY	KG/M-SEC.	1.000E+00
16	K.E. OF TURBLNCE	M.SQ/SEC/SEC	1.000E+00
17	K.E. DISS. RATE	M.SQ/SEC/SEC/SEC	1.000E+00
18	SPEC. HEAT CAP.	JOULES/KG-K	1.000E+00
19	THERMAL CONDUCT.	WATTS/M-K	1.000E+00
20	DIFFUSIVITY	M.SQ/SEC.	1.000E+00
21	ACTIVATION ENRGY	JOULES/KGMOL	1.000E+00
22	ANGLE	RADIANS	1.000E+00
23	HEAT FLUX	WATTS/M.SQ.	1.000E+00
24	PARTICLE DIAM.	METERS	1.000E+00
25	MOMENTUM TR RATE	KG.M/SEC/SEC	1.000E+00
26	HEAT TRANSF COEF	WATTS/M.SQ-K	1.000E+00
27	PERMEABILITY	M.SQ.	1.000E+00
28	(INTERNAL MISC.)	UNDEFINED	1.000E+00
29	VOLUME. FLOWRATE	CU.M/SEC.	1.000E+00
30	AREA	M.SQ.	1.000E+00
31	ARRHENIUS FACTOR	CONSISTENT UNITS	1.000E+00

32	INERTIAL FACTOR	PER METER	1.000E+00
33	VOL. HEAT RATE	WATTS/CU.M.	1.000E+00
34	ABSORB./SCATTER.	PER METER	1.000E+00
35	ANGULAR VELOCITY	RADIANS/SECOND	1.000E+00
36	MOL. SIZE PARM.	ANGSTROMS	1.000E+00
37	PRESSURE GRAD.	PASCALS/METER	1.000E+00
38	MUSHY ZONE CON.	KG/CU.M.-SEC	1.000E+00
39	SURFACE TENSION	NEWTONS/METER	1.000E+00
40	SURF. TEN. GRAD.	NEWTONS/M-K	1.000E+00
41	CONTACT RESIST.	M.SQ.-K/WATT	1.000E+00

- GEOMETRY -

RECTANGULAR CARTESIAN COORDINATES
AXISYMETRIC FLOW

NI = 40 NJ = 20 NK = 1

--- GRID GENERATION INPUTS ---

X-GRID SEGMENT INFORMATION

SEGMENT	START-POINT	LENGTH	# CELLS	WEIGHTING-FACTORS START-POINT	WEIGHTING-FACTORS END-POINT
1	.0000E+00	5.0000E-01	38	.0000	.0000

RADIAL-GRID SEGMENT INFORMATION

SEGMENT	START-POINT	LENGTH	# CELLS	WEIGHTING-FACTORS START-POINT	WEIGHTING-FACTORS END-POINT
1	.0000E+00	1.2500E-01	18	.0000	.0000

CELL TYPES:

```
  J I=  2   4   6   8  10  12  14  16  18  20  22  24  26  28  30  32  34  36  38  40
 20  WAWAWAWAWAWAWAWAWAWAWAWAWAWAWAWAWAWAWAWAWAWAWAWAWAWAWAWAWAWAWAWAWAWAWAWAWAWA  20
 19  WA . . . . . .WA  19
 18  WA . . . . . .WA  18
 17  WA . . . . . .WA  17
 16  WA . . . . . .WA  16
 15  WA . . . . . .WA  15
 14  WA . . . . . .WA  14
 13  WA . . . . . .WA  13
 12  WA . . . . . .WA  12
 11  WA . . . . . .WA  11
 10  WA . . . . . .WA  10
  9  WA . . . . . .WA   9
  8  WA . . . . . .WA   8
  7  WA . . . . . .WB . . . . .WA   7
```

```
6  WA . . . . . .WA    6
5  WA . . . . . .WA    5
4  WA . . . . . .WA    4
3  WA . . . . . .WA    3
2  WA . . . . . .WA    2
1  WA S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S SWA    1
J I=  2   4   6   8  10  12  14  16  18  20  22  24  26  28  30  32  34  36  38  40
```

- MULTI-GRID PARAMETERS -

U-VELOCITY IS SOLVED BY MULTI-GRID METHOD.
TERMINATION CRITERION: .1
RESIDUAL REDUCTION RATE: .8999999
MAX. LEVEL OF BLOCK CORRECTIONS IN I-DIR.: 3
MAX. LEVEL OF BLOCK CORRECTIONS IN J-DIR.: 3
MAX. LEVEL OF BLOCK CORRECTIONS IN K-DIR.: 0

V-VELOCITY IS SOLVED BY MULTI-GRID METHOD.
TERMINATION CRITERION: .1
RESIDUAL REDUCTION RATE: .8999999
MAX. LEVEL OF BLOCK CORRECTIONS IN I-DIR.: 3
MAX. LEVEL OF BLOCK CORRECTIONS IN J-DIR.: 3
MAX. LEVEL OF BLOCK CORRECTIONS IN K-DIR.: 0

ENTHALPY IS SOLVED BY MULTI-GRID METHOD.
TERMINATION CRITERION: .1
RESIDUAL REDUCTION RATE: .8999999
MAX. LEVEL OF BLOCK CORRECTIONS IN I-DIR.: 0
MAX. LEVEL OF BLOCK CORRECTIONS IN J-DIR.: 0
MAX. LEVEL OF BLOCK CORRECTIONS IN K-DIR.: 0

MAXIMUM NO. OF FINE GRID ITERATIONS: 30
MAXIMUM NO. OF ITERATIONS PER LEVEL: 500
COARSE GRID SPACING IN I-DIRECTION: 2
COARSE GRID SPACING IN J-DIRECTION: 2
COARSE GRID SPACING IN K-DIRECTION: 2
MONITOR MG SOLVER: NO
MAX.-MG-LEVEL: 4

- VELOCITY BOUNDARY CONDITIONS -

ZONE	U-VEL.	V-VEL.
WA	.00E+00	.00E+00
WB	.00E+00	.00E+00

- TEMPERATURE BOUNDARY CONDITIONS -

ZONE	TEMPERATURE
WA	HEAT FLUX
WB	3.3000E+02

- SPECIAL TEMPERATURE BOUNDARIES -

ZONE	HEAT FLUX BOUNDARY	HEAT FLUX VALUE	EXT. H-T BOUNDARY	EXTERNAL HEAT TRANSFER COEFF.	EXT. TEMP.
WA	Y	.0000E+00	N	N/A	N/A

ZONE	EXT. RAD BOUNDARY	T-INFINITY	EXT. EMISS.
WA	N	N/A	N/A

- GRAVITATIONAL ACCELERATIONS -

X = 9.810E+00
Y = .000E+00
Z = .000E+00

- USER DEFINED PHYSICAL MODELS -

NON-NEWTONIAN FLUID - NO

- USER DEFINED PROPERTIES -

FLUID VISCOSITY - NO
FLUID DENSITY - NO
FLUID SPECIFIC HEAT - NO
FLUID THERMAL CONDUCTIVITY - NO

- USER DEFINED SOURCE TERMS -

X-MOMENTUM EQUATION - NO
Y-MOMENTUM EQUATION - NO
PRESSURE CORRECTION EQUATION - NO
ENTHALPY EQUATION - NO

- USER STARTUP SUBROUTINE IS NOT ACTIVE -

```
  - USER DEFINED ADJUSTMENTS -


   X-MOMENTUM EQUATION  - NO
   X-MOMENTUM EQUATION  - NO
   X-MOMENTUM EQUATION  - NO


  - USER DEFINED REAL VARIABLES -


   USER DEFINED REAL VARIABLE, USPAR1 -   .00000E+00
   USER DEFINED REAL VARIABLE, USPAR2 -   .00000E+00
   USER DEFINED REAL VARIABLE, USPAR3 -   .00000E+00
   USER DEFINED REAL VARIABLE, USPAR4 -   .00000E+00
   USER DEFINED REAL VARIABLE, USPAR5 -   .00000E+00
   USER DEFINED REAL VARIABLE, USPAR6 -   .00000E+00
   USER DEFINED REAL VARIABLE, USPAR7 -   .00000E+00
   USER DEFINED REAL VARIABLE, USPAR8 -   .00000E+00
   USER DEFINED REAL VARIABLE, USPAR9 -   .00000E+00


  - USER DEFINED INTEGER VARIABLES -


   USER DEFINED INTEGER VARIABLE, IUFLG1 -     0
   USER DEFINED INTEGER VARIABLE, IUFLG2 -     0
   USER DEFINED INTEGER VARIABLE, IUFLG3 -     0
   USER DEFINED INTEGER VARIABLE, IUFLG4 -     0
   USER DEFINED INTEGER VARIABLE, IUFLG5 -     0
   USER DEFINED INTEGER VARIABLE, IUFLG6 -     0
   USER DEFINED INTEGER VARIABLE, IUFLG7 -     0
   USER DEFINED INTEGER VARIABLE, IUFLG8 -     0
   USER DEFINED INTEGER VARIABLE, IUFLG9 -     0

- DENSITY DEFINITION  -

 DENSITY    =   7.218E+02 + 2.110E+00*T**1 - 3.980E-03*T**2

- VISCOSITY DEFINITION -
 VISCOSITY =   3.658E-02 - 2.174E-04*T**1 + 3.276E-07*T**2

- SPECIFIC HEAT DEFINITION -
 CP        =   4.179E+03

ENTHALPY REFERENCE TEMPERATURE =   3.0000E+02

- THERMAL CONDUCTIVITY DEFINITION -
 K         =   6.130E-01


 - SOLUTION CONTROL PARAMETERS -

   SOLVER SWEEP DIRECTION                - J-DIRECTION
```

```
       ALTERNATE SWEEP DIRECTION              - NO
       SOLUTION METHOD                        - SIMPLE
       ALLOW PATCHING OF BOUNDARY VALUES      - NO
       CONVERGENCE/DIVERGENCE CHECK ON        - NO
       MINIMUM RESIDUAL SUM                   - 5.000E-02
       MINIMUM ENTHALPY RESIDUAL              - 1.000E-03
       NORMALIZE RESIDUALS                    - YES
       CONTINUITY CHECK                       - NO
       TEMPERATURE CHANGE LIMITER             - 1.000E+00
       MONITOR SOLVER                         - NO
       FIX VARIABLE OPTION ENABLED            - NO
       SET PRESSURE REFERENCE LOCATION        - NO
       VISCOUS DISSIPATION                    - NO

       DIFFERENCING SCHEME  - POWER LAW

       REFERENCE PRESSURE LOCATION :
         I =    2
         J =    2
```

VARIABLE	SOLVED	BLOCK CORRECT	NO. SWEEPS	UNDERRELAX 1	UNDERRELAX 2	RESIDUAL AT 6639 ITERATIONS
PRESSURE	YES	NO	5	5.0000E-01	5.0000E-01	1.5548E-05
U-VELOCITY	YES	NO	3	3.0000E-01	2.0000E-01	2.9561E-03
V-VELOCITY	YES	NO	3	3.0000E-01	2.0000E-01	2.2914E-03
ENTHALPY	YES	NO	5	1.0000E+00	2.0000E-01	1.7811E-05
PROPERTIES	YES	N/A	N/A	N/A	N/A	N/A
VISCOSITY	N/A	N/A	N/A	5.0000E-02	2.0000E-01	N/A
TEMPERATURE	N/A	N/A	N/A	1.0000E+00	3.0000E-01	N/A

```
 - TIME DEPENDENCE -

 TIME STEP LENGTH (SECONDS)                 = 2.5000E-01
 MAXIMUM NUMBER OF ITERATIONS PER STEP      = 200
 MINIMUM RESIDUAL SUM (TO TRIP STEP)        = 1.0000E-02
 MINIMUM ENTHLAPY RESIDUAL (TO TRIP STEP)   = 1.0000E-03
 AUTO SAVE                                  = YES
 NUMBER OF TIME STEPS BETWEEN SAVES         = 240


 FLOW FIELD AFTER  2400  TIME STEPS     T =   6.000E+02 (SECONDS)     TOTAL NO.
ITERATIONS =  6639
 ------------------------------------------------------------------------------------------
---
```