# Teaching Thermal Hydraulics & Numerical Methods
# An Introductory Control Volume Primer
# By
# D. Scott Lucas, Ph.D.
# INEEL- Advanced Test Reactor (ATR)

P.O. Box 1625
Idaho Falls, Idaho 83415-7136
email: **lucads@inel.gov**
Ph: 208 526 2366

**Abstract**

A graduate level course for Thermal Hydraulics (T/H) was taught through Idaho State University in the spring of 2004. A numerical approach was taken for the content of this course since the students were from the Idaho National Laboratory and most had been users of T/H codes. The majority of the students had expressed an interest in learning about the Courant Limit, mass error, semi-implicit and implicit numerical integration schemes in the context of a computer code. Since no introductory text was found the author developed notes taught from his own research and courses taught for Westinghouse on the subject. The course started with a primer on control volume methods and the construction of a Homogeneous Equilibrium Model (HEM) (T/H) code. The primer was valuable for giving the students the basics behind such codes and their evolution to more complex codes for Thermal Hydraulics and Computational Fluid Dynamics (CFD). The course covered additional material including the Finite Element Method and non-equilibrium (T/H). The control volume primer and the construction of a three-equation (mass, momentum and energy) HEM code are the subject of this paper. The Fortran version of the code covered in this paper is elementary compared to its descendants. The steam tables used are less accurate than the available commercial version written in C coupled to a Graphical User Interface (GUI). The Fortran version and input files can be downloaded at www.microfusionlab.com.

## 1.0 Introduction

One of the main impediments for learning how to use production type codes such as RELAP5-3D© [1] is the lack of a foundation in basic numerical methods and how they are implemented in (T/H) computer codes. Although code manuals provide much of the knowledge base for the program the user may not know the basics of a how a code is put together. Learning how such codes are formulated and their evolution gives the user knowledge that can be used to accurately understand the intricacies of how advanced codes work and the limitations of numerical methods. A good knowledge of numerical methods can also help the user understand the reasons for complex code behavior and result in improved dialogue between the code user and developer. This paper will cover the basics of the implementation of the control volume method in the context of a Homogeneous Equilibrium Model (HEM) (T/H) code using the conservation equations of mass, momentum and energy. This primer uses the advection equation as a template. The discussion will cover the basic equations of the control volume portion of the course in the primer, which includes the advection equation, numerical methods, along with the implementation of the various equations via FORTRAN into computer programs and the final result for

a three equation HEM code and its validation.

## 2.0 Equations

There are many texts and papers relating to the derivation and development of the equations for mass, momentum and energy. Some of these are by Smith & Dixon[2] and Lahey & Moody[3]. The conservation equations for mass, momentum and energy are:

$$A\frac{\partial \rho}{\partial t} + \vec{\nabla} \cdot (\rho \vec{V} A) = 0 \tag{1}$$

$$\frac{\partial \rho VA}{\partial t} + \frac{\partial \rho VAV}{\partial x} = -g_c CA\frac{\partial P}{\partial x} - \rho gA\sin\theta - \frac{K(\rho VA)^2}{2\rho A}\delta(x) \tag{2}$$

$$\frac{\partial(\rho h)}{\partial t} + \vec{\nabla} \cdot (\rho h \vec{V}) = \frac{C}{J}\frac{\partial P}{\partial t} + Q \tag{3}$$

with the equation of state as:

$$\rho = \rho(p,h) \tag{4}$$

with

$\rho$ = density, Lbm/ft$^3$
$P$ = pressure, Lbf/ft$^2$
$h$ = enthalpy, Btu/Lbm
$V$ = scalar velocity, ft/s
$\vec{V}$ = vector velocity, ft/s
$t$ = time, sec
$Q$ = heat source, Btu/(ft$^3$sec)
$\vec{\nabla} \circ \vec{V} = \frac{\partial V}{\partial x}$, in x for one dimension
$g$ = gravitational constant, 32.2 ft/s$^2$
$\theta$ = angle with horizontal
$C$ = 144 in$^2$/ft$^2$
$J$ = mechanical equivalent of heat 778.16 ft Lbf/Btu
$K = fL/D$ = dimensionless form loss
$f$ = friction (dimensionless)
$L, D$ = length (ft), diameter (ft)
$g_c = \frac{Lbm}{Lbf}32.2\frac{ft}{s^2}$

Note that there are four unknowns in the three equations of (1), (2) and (3), the density $\rho$, the velocity $V$, the pressure $P$ and the enthalpy $h$. The equation of state is the fourth equation, which allows the solution of the four equations in the four unknowns.

These conservation equations are developed from the Reynolds Transport Theorem and the Liebnitz Rule as shown in Reference 2.0. The relations above are partial differential equations since they are functions of more than one variable, such as the state mixture density in the state equation of $\rho = \rho(P, h)$.

Relations (1) through (4) can be represented by the general advection equation

$$\frac{\partial f(x,t)}{\partial t} + \frac{\partial f(x,t)V}{\partial x} = S(x,t) \tag{5}$$

where $f(x,t)$ is a continuous function with $x$ as the spatial variable in one dimension and t as the time. $V$ is an advection velocity and $S(x,t)$ is a source term for $f$.

## 3.0 Advection Equation

For a control volume the material derivative is given by:

$$\frac{df(x,t)}{dt} = \frac{\partial f}{\partial x}\frac{\partial x}{\partial t} + \frac{\partial f}{\partial t} = S(x,t) \tag{6}$$

or the rate of change of $f(x,t)$ is equal to the source of $f(x,t)$, which is $S(x,t)$. This relation can be written as

$$\frac{\partial f}{\partial t} + \frac{\partial f V}{\partial x} = S(x,t). \tag{7}$$

The "trick" to understanding the formulation of (T/H) codes is in using Equation (7) as a "template" for the development of each of the conservation equations relating to mass, momentum and energy. The various methods used in code development such as Finite Differences, Finite Volumes and the Finite Element Method (FEM) were developed in the course via the Advection Equation and used with the HEM formulation. Time and space do not permit the discussion of the other methods used in the course. The discussion will be constrained to the primer on control volume methods.

The advection equation is a hyperbolic partial differential equation $(pde)$. Writing the total differential as

$$\frac{df(x(t),t)}{dt} = \frac{\partial f(x(t),t)}{\partial x}\left(\frac{dx}{dt}\right) + \frac{\partial f(x,t)}{\partial t} \tag{8}$$

and comparing this to a version of (7), wherein $V$ is constant, the relation

$$\frac{\partial f(x,t)}{\partial t} + V\frac{\partial f(x,t)}{\partial x} = S(x,t) \tag{9}$$

is obtained. We can identify $dx/dt = V$ and $df(x(t),t)/dt = S(x,t)$. These two relations can be integrated to get

$$x = Vt + C$$
$$f = St + A$$

and
(10a,b)

$$x = x_0 + Vt$$
$$f = f_0 + St$$

after applying the initial conditions $x(0) = x_0$ and $f(x,0) = f_0$. Figure 1.0 illustrates that these relations are "characteristics" and they are lines (equation of a straight line $y = mx+b$) with constant slopes in the x and t planes.
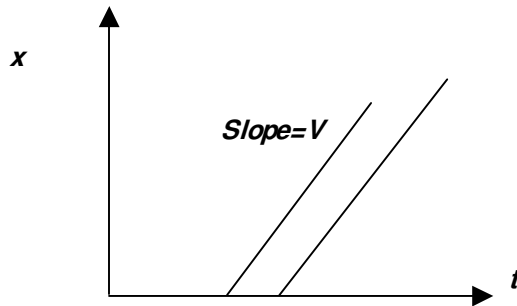


*Figure 1.0*

If there is no source term, $S(x,t) = 0$ in (9), the second equation of (10) becomes

$$f = f_0$$
(11)

wherein **f** is a constant. If we put some **f(x,t)** in at one end of a pipe, where the fluid is flowing with velocity **V**, we expect to get the same amount out at the other end of the pipe for the constant velocity case. As simple as this appears, propagating material exactly may well be one of the most "difficult" problems in computational physics.

There are three types of boundary conditions for the function **f(x,t)**. They are:

1. Dirichlet conditions with $f(x,t) = f_0$ on the boundary, the boundary denoted as $\partial R$
2. Neumann conditions with $\partial f / \partial x = g_0$ on $\partial R$
3. Mixed (Robin) conditions such as $\partial f / \partial x + kf = g_0$ on $\partial R$

Using (5) as a template we can program features of all the codes and their equation sets by using the advection equation, differenced as

$$\frac{\partial f(x,t)}{\partial t} \approx \frac{f^{n+1}(x_i) - f^n(x_i)}{\Delta t} = \frac{f_i^{n+1} - f_i^n}{\Delta t}$$

<div style="text-align: right">(12)</div>

$$\frac{\partial f(x,t)V}{\partial x} \approx \frac{(fV)_i^{n+1} - (fV)_{i-1}^n}{\Delta x}$$

on the grid shown in Figure 2.0. This type of differencing is referred to as "backwards" differencing. The **n or n+1** superscript for the temporal derivative refers to the intervals the time domain is broken up into, more simply called the "time step."

The spatial interval is known as a cell, a volume or a node. Since **f(x, t)** is a scalar function, not a vector, we can see a general pattern developing. In the example in Figure 2.0 the spatial domain is broken into three nodes, **i- 1**, **i** and **i+1**. We take the length of the boxes surrounding the interior nodes as $\Delta x$, a constant spatial interval. Node **i- 1** does not need a cell (length) since it is a boundary node with constant properties. The time domain is also broken into constant intervals (or steps), given as $\Delta t = t^{n+1} - t^n$. For the three cells in Figure 2.0 we write two equations for the interior nodes as
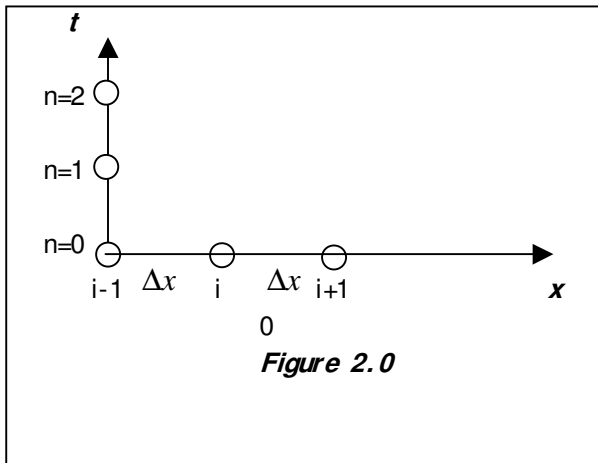
$$\frac{\partial f(x_i,t)}{\partial t} + V\frac{f(x_i,t^{n+1}) - f(x_{i-1},t^n)}{\Delta x} = \frac{f_i^{n+1} - f_i^n}{\Delta t} + V\frac{f_i^{n+1} - f_{i-1}^n}{\Delta x} = 0$$

and

$$\frac{f_{i+1}^{n+1} - f_{i+1}^n}{\Delta t} + V\frac{f_{i+1}^{n+1} - f_i^n}{\Delta x} = 0$$

<div style="text-align: right">(13a,b)</div>

or

$$\frac{f_{i+1}^{n+1} - f_{i+1}^n}{\Delta t} + \frac{(Vf)_{i+1}^{n+1} - (Vf)_i^n}{\Delta x} = 0$$



**Figure 2.0**

since **V** is a constant for the present.

When we gather terms we get the two equations

<div style="text-align: right">5</div>

$$f_i^{n+1}\left(1+V\frac{\Delta t}{\Delta x}\right)-V\frac{\Delta t}{\Delta x}f_{i-1}^n=f_i^n$$

$$f_{i+1}^{n+1}\left(1+V\frac{\Delta t}{\Delta x}\right)-V\frac{\Delta t}{\Delta x}f_i^n=f_{i+1}^n$$

(14a,b)

The reason we write just two equations for the three nodes or cells is that we need a boundary value at cell *i- 1*, i.e., we need to have something flowing in (actually we don't but propagating zero is not very interesting).

Using the nth time level gets us away from having to do a simultaneous matrix solution. However, this method can be non-conserving. Its use should be limited to emulation and not applied in safety analysis. If we write Equation (14a,b) at new time levels for the spatial derivative terms as,

$$f_i^{n+1}\left(1+V\frac{\Delta t}{\Delta x}\right)-V\frac{\Delta t}{\Delta x}f_{i-1}^{n+1}=f_i^n$$

(15a,b)

$$f_{i+1}^{n+1}\left(1+V\frac{\Delta t}{\Delta x}\right)-V\frac{\Delta t}{\Delta x}f_i^{n+1}=f_{i+1}^n$$

we have to solve the system of equations as a matrix, and expend more effort, since we have to solve a matrix form of the equations as

$$\begin{bmatrix}\left(1+V\dfrac{\Delta t}{\Delta x}\right) & 0 \\ -V\dfrac{\Delta t}{\Delta x} & \left(1+V\dfrac{\Delta t}{\Delta x}\right)\end{bmatrix}\begin{bmatrix}f_i^{n+1} \\ f_{i+1}^{n+1}\end{bmatrix}=\begin{bmatrix}f_i^n+V\dfrac{\Delta t}{\Delta x}f_{i-1}^{n+1} \\ f_{i+1}^n\end{bmatrix}$$

(16)

Equation (16) is referred to as an "implicit" scheme since the spatial derivative and time derivative terms are connected at the same temporal level. Another interesting time level approximation (discretization) is to use the derivative terms at the past time level as:

$$\frac{f_i^{n+1}-f_i^n}{\Delta t}+V\frac{f_i^n-f_{i-1}^n}{\Delta x}=0$$

(17)

$$\frac{f_{i+1}^{n+1}-f_{i+1}^n}{\Delta t}+V\frac{f_{i+1}^n-f_i^n}{\Delta x}=0$$

or

$$f_i^{n+1} = f_i^n \left( 1 - \frac{\Delta t}{\Delta x} V \right) + \frac{\Delta t}{\Delta x} V f_{i-1}^n$$

$$(18)$$

$$f_{i+1}^{n+1} = f_{i+1}^n \left( 1 - \frac{\Delta t}{\Delta x} V \right) + \frac{\Delta t}{\Delta x} V f_i^n$$

This is referred to as an "explicit" scheme since all the quantities of interest are explicitly known from past time or the initial values. Let's look at the stability of this set of equations, that is, do they ever give us a (**NaN**), defined as "not a number" on a computer, not something you want to see in your output. For the first time step in (18) we can write

$$n = 0$$

$$f_i^1 = f_i^0 \left( 1 - \frac{\Delta t}{\Delta x} V \right) + \frac{\Delta t}{\Delta x} V f_{i-1}^n$$

$$(19)$$

$$n = 1$$

$$f_i^2 = f_i^1 \left( 1 - \frac{\Delta t}{\Delta x} V \right) + \frac{\Delta t}{\Delta x} V f_{i-1}^n$$

where $f_{i-1}^n$ is a constant boundary value and write it in the following equations as $f_{i-1}$. We substitute the value of $f_i^1$ into $f_i^2$ in (19) to obtain

$$f_i^2 = f_i^0 \left( 1 - \frac{\Delta t}{\Delta x} V \right)^2 + \left( 1 - \frac{\Delta t}{\Delta x} V \right) \frac{\Delta t}{\Delta x} V f_{i-1} + \frac{\Delta t}{\Delta x} V f_{i-1}$$

$$f_i^3 = f_i^0 \left( 1 - \frac{\Delta t}{\Delta x} V \right)^3 + \left( 1 - \frac{\Delta t}{\Delta x} V \right)^2 \frac{\Delta t}{\Delta x} V f_{i-1} + \left( 1 - \frac{\Delta t}{\Delta x} V \right) \frac{\Delta t}{\Delta x} V f_{i-1} + \frac{\Delta t}{\Delta x} V f_{i-1} \qquad (20)$$

$$f_i^N = f_i^0 \left( 1 - \frac{\Delta t}{\Delta x} V \right)^N + \left( 1 - \frac{\Delta t}{\Delta x} V \right)^{N-1} \frac{\Delta t}{\Delta x} V f_{i-1} + \left( 1 - \frac{\Delta t}{\Delta x} V \right)^{N-2} \frac{\Delta t}{\Delta x} V f_{i-1} + \circ \circ \circ + \frac{\Delta t}{\Delta x} V f_{i-1}$$

by induction. We use **N** as the superscript denoting the number of time steps and it approaches $\infty$ (infinity) for a large number of time steps.

As long as $\Delta t V / \Delta x < 1.0$ in $\left( 1 - \Delta t V / \Delta x \right)^N$, this term will approach zero and we are left with

$$f_i^{Nth} = \frac{\Delta t}{\Delta x} V f_{i-1}$$

$$(21)$$

which is the correct answer for propagating $f_{i-1}$ from left to right exactly. Conditions such as this are referred to as "stability criteria". If $\Delta t V / \Delta x > 1.0$ in the term $\left( 1 - \Delta t V / \Delta x \right)^N$ the value will be either "large positive" or "large negative" as $N \rightarrow \infty$ and the output will look like that of Figure 3.0 below for separate time steps.
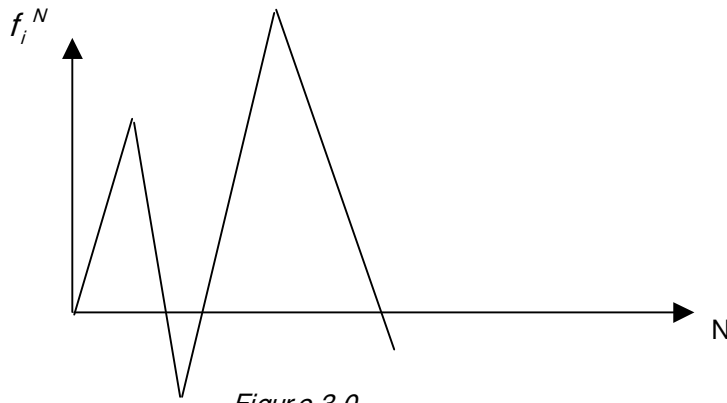
$f_i^N$

N

*Figure 3.0*

We write (15a) as

$$f_i^{n+1}\left(1+V\frac{\Delta t}{\Delta x}\right)=f_i^{n}+V\frac{\Delta t}{\Delta x}f_{i-1}^{n+1} \tag{22}$$

If we were to set the datum of (22), the term $f_{i-1}^{n+1}$, to zero, the equation would become

$$f_i^{n+1}\left(1+V\frac{\Delta t}{\Delta x}\right)=f_i^{n} \tag{23}$$

with $V$ as a constant in the limit. For each time level we can write out the expression as

$$f_i^{n+1}\left(1+V\frac{\Delta t}{\Delta x}\right)=f_i^{n}$$
$$f_i^{1}=f_i^{0}/\left(1+V\frac{\Delta t}{\Delta x}\right)$$
$$f_i^{2}=f_i^{1}/\left(1+V\frac{\Delta t}{\Delta x}\right)=f_i^{0}/\left(1+V\frac{\Delta t}{\Delta x}\right)^{2}$$
$$f_i^{N}=f_i^{0}/\left(1+V\frac{\Delta t}{\Delta x}\right)^{N}$$

(24)

This form will be unconditionally stable for positive $V$, since as **N** becomes large, $1+V\Delta t/\Delta x$ is always greater than zero. We do propagate $f_{i-1}$ but at earlier times we get some lesser value, so instead of making mountains out of molehills, we make molehills out of mountains, a phenomena called "damping" or diffusion.

The term $C=V\Delta t/\Delta x$ is known as the Courant Number. As discussed, we need $C\leq 1.0$ for stability in the explicit scheme. Setting $C=1.0$ yields "exact" propagation from (19) and (20).

To illustrate these ideas, we construct a Fortran program for the relations we have discussed. You can simply copy and paste the listing in an appropriate Fortran compiler. This text is in Comic Sans so you might want to change the font if necessary.

Here is the Fortran listing:

```
    program hyperback
c
c
c
c   This is a simple program to read 2 numbers and print the sum
c
    implicit none
    real dt,dx,fimin1,fi,fiplus1,C,V,time
    integer ndt,i
c
c   dt = time step
c   dx = cell width
c   C = courant number
c   fimin1 = f at left boundary
c   fi  = f at internal node
c   fiplus1 = f at right node
c   V = advection velocity
c   ndt = number of time steps
c   time = running account of time
c
c   open some files &
c   ask for the number of time steps
c
    open(6,file='out.o',status='old')
c
    print *,'Input the number of time steps'
        read *, ndt
c
c  Get initial values and boundary value
c
    print *,' This program computes the propagation of f'
    print *,' Input dt,dx,fimin1,fi,fiplus1,V'
        read *, dt,dx,fimin1,fi,fiplus1,V
c
c   Compute the Courant number
c
    if(dx.le.0.0) then
        print *,'cannot compute courant number, dx is <= 0.0'
        goto 300
        endif
c
c
    C=V*(dt/dx)
c
c    initialize time to zero
c
    time = 0.0
c
```

```
c     do loop for calculation
c
c     header for output
c
      print *,' time              C              fimin1  fi              fiplus1'
      print *,' '
c
         write(6,*) time,fimin1,fi,fiplus1
         do i =1,ndt
         time = time + dt
         fi = (fi + C*fimin1)/(1+C)
         fiplus1 = (fiplus1 + C*fi)/(1+C)
         print *, time,C,fimin1,fi,fiplus1
         write(6,*) time,fimin1,fi,fiplus1
         end do
c
300   continue
      stop
      end
```

Here is what the output file on unit 6 looks like

| | | | |
|---|---|---|---|
| 0.00E+00 | 1.000000 | 0.000E+00 | 0.000E+00 |
| 1.000000 | 1.000000 | 0.5000000 | 0.2500000 |
| 2.000000 | 1.000000 | 0.7500000 | 0.5000000 |
| 3.000000 | 1.000000 | 0.8750000 | 0.6875000 |
| 4.000000 | 1.000000 | 0.9375000 | 0.8125000 |
| 5.000000 | 1.000000 | 0.9687500 | 0.8906250 |
| 6.000000 | 1.000000 | 0.9843750 | 0.9375000 |
| 7.000000 | 1.000000 | 0.9921875 | 0.9648438 |
| 8.000000 | 1.000000 | 0.9960938 | 0.9804688 |
| 9.000000 | 1.000000 | 0.9980469 | 0.9892578 |
| 10.00000 | 1.000000 | 0.9990234 | 0.9941406 |

for the input typed in as the following when you run the program:

*Input the number of time steps*
*10*
*This program computes the propagation of f*
*Input dt,dx,fimin1,fi,fiplus1,V*
*1,1,1,0,0,1*

Note that the output file, is registered as unit 6 and it's status is designated as "old". It must be there or you will get an error. To plot the values of each of the variables we will use an excel spreadsheet. The steps are:

- Go to the excel program, double click and open excel.
- Go to the file header and click on open file.
- Go to the directory where the ouput file is, click on it and open it.
- Make sure you select "Fixed Width"
- Hit finish and the values are lined up.
- Highlight all the values by using your left mouse button, drag across and down

- Go to the plot icon labeled "Chart Wizard" and select scatter, use the last scatter option
- Select "Finish" to look at the plot

If we return to our set of equations for three nodes, we use them in the form of

$$\frac{\partial f}{\partial t} + \frac{\partial (fV)}{\partial x} = 0$$

$$\frac{f_i^{n+1} - f_i^n}{\Delta t} + \frac{(fV)_i - (fV)_{i-1}}{\Delta x} = 0$$

(25)

We have finite differenced the spatial terms as a product with the velocity. This form becomes more useful for staggered grid formulations. Staggered grids are based on control volumes where the energy and pressure are found within the control volume and the flows or velocities are found on the faces. For the three nodes of Figure 2.0, we can write the equations as

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & (1+C_i) & 0 \\ 0 & 0 & (1+C_{i+1}) \end{bmatrix} \begin{bmatrix} f_{i-1} \\ f_i \\ f_{i+1} \end{bmatrix} = \begin{bmatrix} f_{i-1} \\ f_i^n + C_{i-1}f_{i-1}^n \\ f_{i+1}^n + C_i f_i^n \end{bmatrix}$$

(26)

with $C = V\dfrac{\Delta t}{\Delta x}$ for the **Courant Number**. For **N** nodes we have,

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & (1+C_i) & 0 & 0 & 0 \\ 0 & 0 & (1+C_{i+1}) & 0 & 0 \\ 0 & 0 & 0 & (1+C_m) & 0 \\ 0 & 0 & 0 & 0 & (1+C_N) \end{bmatrix} \begin{bmatrix} f_{i-1} \\ f_i \\ f_{i+1} \\ f_m \\ f_N \end{bmatrix}^{n+1} = \begin{bmatrix} f_{i-1}^* \\ f_i^n + C_{i-1}f_{i-1}^* \\ f_{i+1}^n + C_i f_i^n \\ f_m^n + C_{m-1}f_{m-1}^n \\ f_N^n + C_{N-1}f_{N-1}^n \end{bmatrix}$$

(27)

for the finite difference matrix with *m* as an intermediate value between *i+1* and **N** and $f_{i-1}^*$ is a boundary value.

The above equation can also be written as

$$\underline{\underline{D}}f^{n+1} = E$$

(28)

in symbolic matrix notation. Note that **E** is a *(5x1)* column vector and $\underline{\underline{D}}$ is a *(5x5)* matrix in our notation. The double underline denotes a matrix and boldface denotes a vector. The types of operations that we perform in this course will always be with square matrices, otherwise the rules will not be applicable.

The inverse of $\underline{\underline{D}}$ in (28) is

$$\underline{\underline{\mathbf{D}}}^{-1} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1/(1+C_i) & 0 & 0 & 0 \\ 0 & 0 & 1/(1+C_{i+1}) & 0 & 0 \\ 0 & 0 & 0 & 1/(1+C_m) & 0 \\ 0 & 0 & 0 & 0 & 1/(1+C_N) \end{bmatrix} \qquad (29)$$

Symbollically, the solution of (28) can be written as

$$\mathbf{f}^{n+1} = \underline{\underline{\mathbf{D}}}^{-1}\mathbf{E} \qquad (30)$$

We will implement (27) in a computer program called hypersysD, the **D** denoting a diagonal matrix form.

The following is a listing of this program.

```
      program hypersysD
c
c     program to solve a diagonal set of matrix equations
c
c
c
      implicit none
      real dt,dx,C(100),time
      real f(100),V(100),tiny,D(100),E(100)
      integer ndt,i,BN(100),N,nout,j
c
c  i is gthe time index
c  dt = time step
c  dx = cell width
c  C = courant number
c  f = convected quantity
c  V = advection velocity
c  ndt = number of time steps
c  time = running account of time
c  BN(node) = 0, internal node
c  BN(node) = 1, boundary node
c  N = total number of nodes
c  tiny = a tiny number
c  nout = node number for output
c  D = the lhs diagonal matrix
c  j is an extra integer index
c
c  define tiny
c
      tiny = 1.0e-6
c
c
c  open some files
c
```

```fortran
      open(5,file='inp.i',status='old')
      open(6,file='out.o',status='old')
c
c   read the number of time steps
c
      read(5,*) ndt
c
c   read number of nodes & output node number
c
      read(5,*) N,nout
c
c   read dt,dx
c
      read(5,*) dt,dx
c
c   read in the node number, BN flag and initial value of f & V
c
      do i = 1,N
          read(5,*) j,BN(j),f(j),V(j)
          C(j)=V(j)*(dt/(dx+tiny))
          enddo
c
c   initialize time to zero
c
      time = 0.0
c
c   do loop for calculation
c
c   header for output
c
      print *,' time      =',time
          print *,' '
          print *,' node#          C              f'
      print *,' '
c
c       write(6,*) time,f(nout)
          do i =1,ndt
          time = time + dt
c
c   construct the D & E matrix
c   do loop to build matrix
c
      do j =1,N
c
          if (BN(j).eq.0) then
          D(j)=(1.0+C(j))
          E(j)=f(j)+C(j-1)*f(j-1)
          else
          D(j) = 1.0
          E(j) = f(j)
```

```fortran
        Endif
c
c  solve for f so that we update it to new time
c
      f(j)=E(j)/D(j)
      enddo
c
c     print out the results
c
      print *, 'time=',time
         print *,' '
         do j=1,N
         print *, j,f(j),C(j)
         enddo
c
         write(6,11) time,f(1),f(2),f(3),f(4),f(5),f(6),f(7),f(8),
     &f(9),f(10)
11   format(1x,f6.2,10(1x,f6.2))
         end do
c
300   continue
      stop
      end
```

The input file is also shown below. The problem is for *10* nodes with node one as the boundary node, using a value of *f=10.0*.

```
1000
10,10
1,1
1,1,10,1
2,0,0,1
3,0,0,1
4,0,0,1
5,0,0,1
6,0,0,1
7,0,0,1
8,0,0,1
9,0,0,1
10,0,0,1


ndt
Nodes,node# for printout
dt,dx
j,BN,f,V
```

It's always a good idea to put the input order in the input file as a reminder. Part of the output listing

14

is also shown. The first entry is time and the next 10 entries are the values of *f (j)* as *j* goes from 1 to 10.

```
 1.00  10.00   5.00   2.50   1.25   0.62   0.31   0.16   0.08   0.04   0.02
 2.00  10.00   7.50   5.00   3.12   1.87   1.09   0.62   0.35   0.20   0.11
 3.00  10.00   8.75   6.87   5.00   3.44   2.27   1.45   0.90   0.55   0.33
 4.00  10.00   9.37   8.12   6.56   5.00   3.63   2.54   1.72   1.13   0.73
 5.00  10.00   9.69   8.91   7.73   6.37   5.00   3.77   2.74   1.94   1.33
 6.00  10.00   9.84   9.37   8.55   7.46   6.23   5.00   3.87   2.91   2.12
 7.00  10.00   9.92   9.65   9.10   8.28   7.26   6.13   5.00   3.95   3.04
 8.00  10.00   9.96   9.80   9.45   8.87   8.06   7.09   6.05   5.00   4.02
 9.00  10.00   9.98   9.89   9.67   9.27   8.67   7.88   6.96   5.98   5.00
10.00  10.00   9.99   9.94   9.81   9.54   9.10   8.49   7.73   6.85   5.93
11.00  10.00  10.00   9.97   9.89   9.71   9.41   8.95   8.34   7.60   6.76
12.00  10.00  10.00   9.98   9.94   9.82   9.62   9.28   8.81   8.20   7.48
13.00  10.00  10.00   9.99   9.96   9.89   9.75   9.52   9.16   8.68   8.08
14.00  10.00  10.00  10.00   9.98   9.94   9.85   9.68   9.42   9.05   8.57
15.00  10.00  10.00  10.00   9.99   9.96   9.90   9.79   9.61   9.33   8.95
16.00  10.00  10.00  10.00   9.99   9.98   9.94   9.87   9.74   9.53   9.24
17.00  10.00  10.00  10.00  10.00   9.99   9.96   9.92   9.83   9.68   9.46
18.00  10.00  10.00  10.00  10.00   9.99   9.98   9.95   9.89   9.78   9.62
19.00  10.00  10.00  10.00  10.00  10.00   9.99   9.97   9.93   9.86   9.74
20.00  10.00  10.00  10.00  10.00  10.00   9.99   9.98   9.95   9.90   9.82
21.00  10.00  10.00  10.00  10.00  10.00  10.00   9.99   9.97   9.94   9.88
22.00  10.00  10.00  10.00  10.00  10.00  10.00   9.99   9.98   9.96   9.92
23.00  10.00  10.00  10.00  10.00  10.00  10.00  10.00   9.99   9.97   9.95
24.00  10.00  10.00  10.00  10.00  10.00  10.00  10.00   9.99   9.98   9.96
25.00  10.00  10.00  10.00  10.00  10.00  10.00  10.00  10.00   9.99   9.98
26.00  10.00  10.00  10.00  10.00  10.00  10.00  10.00  10.00   9.99   9.99
27.00  10.00  10.00  10.00  10.00  10.00  10.00  10.00  10.00  10.00   9.99
28.00  10.00  10.00  10.00  10.00  10.00  10.00  10.00  10.00  10.00   9.99
```

## 2.0    Control Volume Method

For the control volume method the basic idea is to use spatial regions or control volumes for the scalar quantities such as pressure, density, energy, temperature, quality or void fraction, and links or junctions on the faces of the control volumes for vector quantities such as velocity or mass flow. This is accomplished by the use of a staggered grid" as shown in Figure 4.0. A section of pipe has flows going in and out. These flows are represented by links (or junctions) *j-1, j, j+1*. The pipe of length *L* is broken into two volumes *i-1 and i,* connected by a link *j*. The pipe segment could have easily been broken into more volumes and links. Volumes *K* and *L* are "boundary' volumes. These are needed for the convection of flow into or out of the system, which has density associated with it. They can also be thought of as reservoirs whose conditions can or cannot change with time.
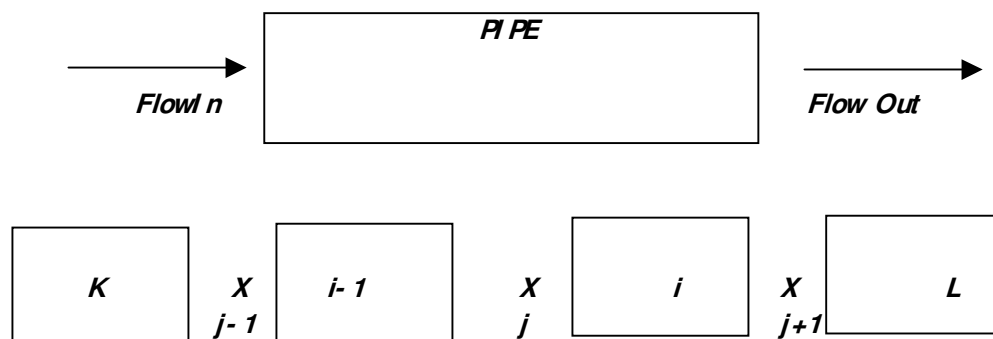


*Figure 4.0*

In order to accomplish the objective of discretization, the partial differential equations in space must be broken down into differential equations in time. This is accomplished by the use of the Gauss Divergence Theorem, given as

$$\int_{V} div\vec{f}\; dV = \int_{A} \vec{n} \circ \vec{f}\; dA \tag{31}$$

for some variable **f** which relates a volume integral over the volume $V$ to a surface integral and a differential area **dA**. For the vector field $\vec{f} = \vec{f}(x)$ Equation (3-1) is given with $\vec{n}$ as the outward unit normal to the bounding surface **A** of the volume $V$ in which the vector field is defined as shown in Figure 5.0.
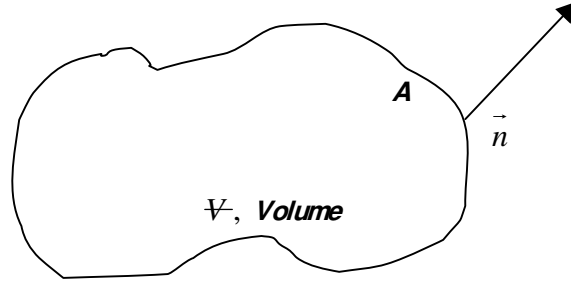


*Figure 5.0*

The theorem can also be written as:

$$\int_{V} \vec{\nabla} \circ \vec{f}\; dV = \int_{A} \vec{n} \circ \vec{f}\; dA \tag{32}$$

where

$$\vec{\nabla} = \frac{\partial}{\partial x}\vec{i} + \frac{\partial}{\partial y}\vec{j} + \frac{\partial}{\partial z}\vec{k}$$

$$and \tag{33}$$

$$\vec{f} = f_x\vec{i} + f_y\vec{j} + f_z\vec{k}$$

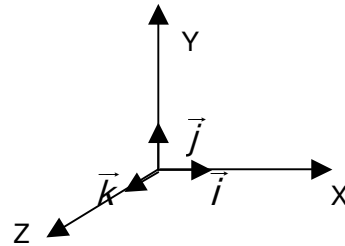with $\vec{i}, \vec{j}, \vec{k}$ as unit vectors along the x,y and z axis.

Using (33) the advection relation of (7)

$$\frac{\partial f}{\partial t} + \frac{\partial f V}{\partial x} = S(x,t)$$

can be written as

$$\frac{\partial f}{\partial t} + \vec{\nabla} \circ f\vec{V} = S(x,t). \tag{34}$$

using the velocity as a vector. For a control volume **N,** as shown in Figure 6.0, surrounded by $k$ links (junctions), we integrate Equation (34) over $dV$ (or $V$ the volume) to obtain



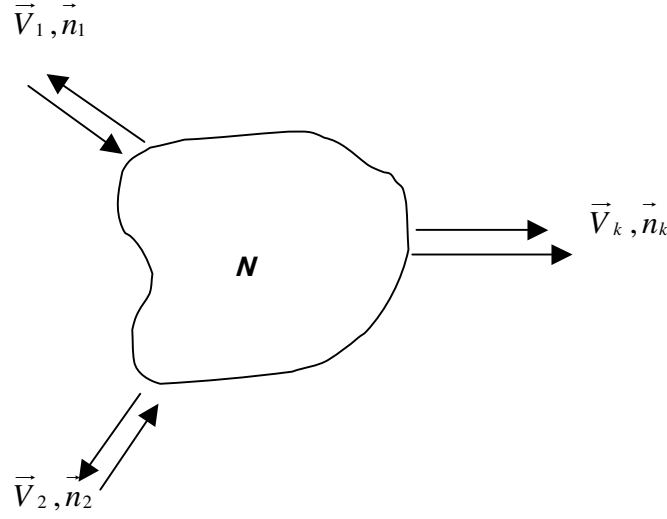$$\vec{V}_1, \vec{n}_1$$

$$\vec{V}_k, \vec{n}_k$$

$$N$$

$$\vec{V}_2, n_2$$

**Figure 6.0**

$$\int_V \left( \frac{\partial f}{\partial t} + \vec{\nabla} \circ f\,\vec{V} \right) dV = \int_V S(x,t)\,dV \tag{35}$$

For the moment, the assumption is made that the scalar quantities in the control volume are well mixed at some average value. Using the Gauss Divergence Theorem and the assumption of perfect mixing, the following relation is obtained:

$$\frac{d\,(f\ V)_N}{dt} + \int_A \left( \vec{n} \circ f\,\vec{V} \right) dA = S_N V_N \tag{36}$$

for volume **N**.

Recall that the $\vec{n}$ surface vectors are outward unit normal vectors. For **k** links connecting to a volume, the **k** links connecting to the surface **A** of the volume, there will be $\hat{n}_k$ associated unit normal vectors, with the incoming links denoted as **in** or terminal links, **t,** (going into node **N**) or **out** incident links, **i,** (leaving node **N**) as outgoing links.

The Riemann surface integral in Equations (36) can be represented as a sum of the link surface areas as:

$$\int_S \left( \vec{n} \circ f\,\vec{V} \right) dS \equiv \sum_k \left( f\,\vec{V} \right)_k \circ \vec{n}_k A_k \tag{37}$$

using **A_k** as the link surface area.

Note that for the terminal links, the unit vector associated with the incoming velocity is opposite to the outward normal of the control volume **N** surface, while for the incident or outgoing flows, the unit vector associated with the velocity is in the same direction as the outward normal of the surface.
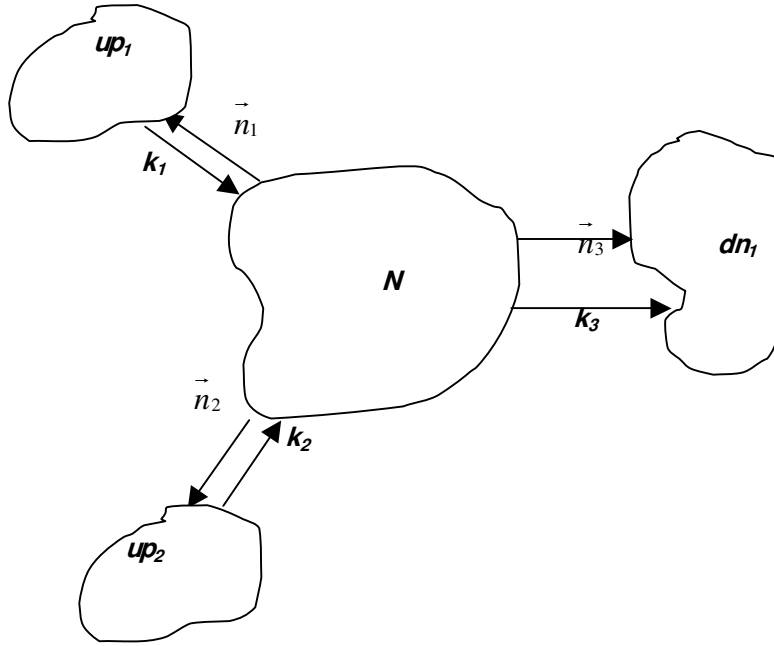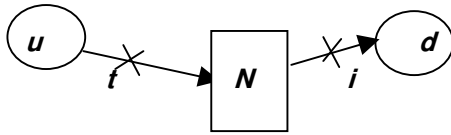
17

*Figure 7.0*

Thus, there will be two sums for the surface integral representation, one sum over the terminal links and one sum for the incident links. The terminal links will have a negative sign due to the dot product of the outward unit normal and the unit normal of the incoming velocity being negative. Relation (37) becomes

$$V_N \frac{df_N}{dt} - \sum_t a_{N,t} (f\,VA)_t + \sum_i a_{N,i} (f\,VA)_i = S_N V_N \tag{38}$$



for the graphical depiction above. A more compact notation can be used but can result in more confusion. $a_{N,t}$ is an indexing symbol for the inlet links *(t)* of the assumed flow direction starting at the *(from)* volume, the upstream volume, to the terminal volume **N** ; the $a_{N,i}$ represents the outlet link's from volume **N** to its inlet *(to)* volume.

In general,

$$k = \{k : t, i \, \varepsilon \, of \, k\} \tag{39}$$

The total number of links is the set that has a link as outlet and inlet to different nodes. As noted earlier the $\rho V A$ term is usually written as **w**, and denoted as the mass flow

$$w_k = (\rho V A)_k \tag{40}$$

for a link **k**.

An additional discussion on the nature of the $a_{N,k}$ is given in Appendix I.

For the mass and energy equations (1) and (3), comparison to the template equation of (35) gives

$$\frac{dM_N}{dt} = \sum_t a_{N,t} w_t - \sum_i a_{N,i} w_i \tag{41}$$

and

$$\frac{d(M_N h_N)}{dt} = \sum_t a_{N,t} (wh)_t - \sum_i a_{N,i} (wh)_i + \frac{C}{J} V_N \dot{P}_N + V_N Q_N \tag{42}$$

which clearly shows the sum of the inlet flows is positive and the outlet flows are negative.

The basic idea behind numerical modeling for Thermal Hydraulics **(T/H)** codes is the use of a staggered grid, where the scalar quantities, such as pressure and energy, are solved for in control volumes and the vector quantities which have direction, mass flows or velocities, are obtained at the links. The mass flows or velocities are staggered across the control volumes.

In order to do this, some assumptions and prescriptions have to be made and followed throughout the development for the momentum equation. As shown in Figure 8.0, a link or junction **j** spans one-half of each adjoining control volume i-1 and i.
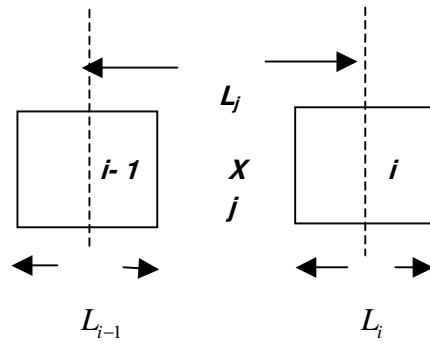


**Figure 8.0**

Thus, the local length that each link spans is

$$L_j = \frac{1}{2} L_{j-1} + \frac{1}{2} L_j \tag{43}$$

or one-half of the upstream control volume and one-half of the downstream control volume.

To discretize the momentum equation, integrate it from **0** to $L_j$ for a link, recalling that

$w_j = (\rho V A)_j$, or

$$\frac{1}{A}\int_0^{L_j}\frac{\partial w}{\partial t}dx + \frac{1}{A}\int_0^{L_j}\frac{\partial wV}{\partial x}dx =$$

$$-g_c C\int_0^{L_j}\frac{\partial P}{\partial x}dx - g\int_0^{L_j}\rho\,dz - \int_0^{L_j}\frac{Kw\,|w|}{2\rho A^2}\delta(x)dx$$

with

$$g\int_0^{L_j}\rho\,Sin\theta dx = g\int_0^{L_j}\rho\,dz = g\left[\int_{Z_{i-1}}^{Z_j}\rho_{i-1}dz + \int_{Z_j}^{Z_i}\rho_i\,dz\right]$$ (44)

$$g\left[\int_{Z_{i-1}}^{Z_j}\rho_{i-1}dz + \int_{Z_j}^{Z_i}\rho_i\,dz\right] = g\left[\rho_i\left(Z_i - Z_j\right) - \rho_{i-1}\left(Z_{i-1} - Z_j\right)\right]$$

as in Figure 9.0.

The assumptions are as follows:

- **w**, the mass flow, is uniform over the link length.
- The pressures at **0, $L_j$** are the upstream and downstream pressures for the link (junction).
- The computed pressure is defined at the center of the volume.
- Density is uniform at the link (junction).
- The head term is integrated from the center of node **i- 1** to the link elevation and from the link elevation to the center of node **i.**

Also, note that an absolute value sign has been placed on one of the mass flow terms in the frictional pressure drop portion of the momentum equation. This is to properly account for flow reversals. The sifting property of the delta distribution, of

$$\int_0^\infty f(x)\delta(x)dx = f(0)$$ (45)

is used in the integration.

Using the assumptions above, Equation (3-14) becomes:

$$\frac{L_j}{A_j}\frac{dw_j}{dt} + \frac{1}{A_j}(wV)\big|_0^L = -g_c C\left[P(L_j) - P(0)\right]$$

$$-g\left[\rho_i\left(Z_i - Z_j\right) - \rho_{i-1}\left(Z_{i-1} - Z_j\right)\right] - \left(\frac{Kw\,|w|}{2\rho A^2}\right)_j$$ (46)

The terms are:

$$\frac{L_j}{A_j} = I_j = \text{Link or junction Inertia, } 1/ft \tag{47}$$

$$P(L_j) = P_i = P_{Downstream} = P_j^D \tag{48}$$

$$P(0) = P_{i-1} = P_{Upstream} = P_j^U \tag{49}$$

for link $j$. Equation (46) becomes, upon neglecting the momentum flux terms,

$$I_j \frac{dw_j}{dt} = g_c C\left[P_j^U - P_j^D\right] - g\Delta Z_j - \left(\frac{Kw|w|}{2\rho A^2}\right)_j \tag{50}$$
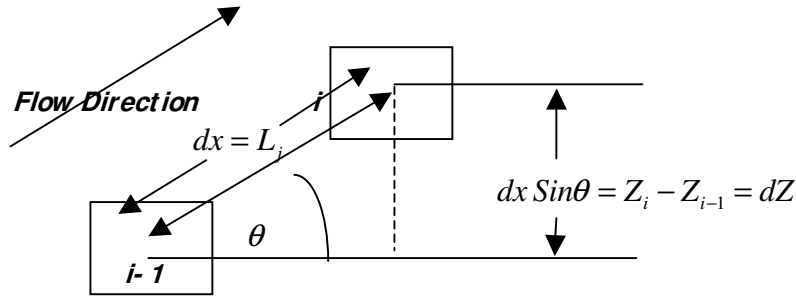


**Figure 9.0**

The various elevations are $Z_{i-1}$ for the centerline elevation of node $i$-$1$, $Z_i$ for the centerline elevation of node $i$, and $Z_j$ for the elevation of link $j$. The second term on the right hand side of (50) is abbreviated notation for the head difference terms of equation (46).

By observing Figure 9, the elevation term in the integration was used as:

$$\Delta Z_j = \left[\rho_i\left(Z_i - Z_j\right) - \rho_{i-1}\left(Z_{i-1} - Z_j\right)\right] \tag{51}$$

where $Z_{i-1}$ and $Z_i$ are the centerline elevations of the upstream and downstream volumes of link $j$. The elevation term in (41) represents the nodal elevation and density difference terms of (46). Using the nodal densities is necessary in order to prevent work being done with flow reversals for the head terms. The integration of the head term is done from the elevation of the centerline of node $i$-$1$ to the link elevation, and from the link elevation to the centerline of node $i$.

If we let $dw/dt \sim \left(w^{n+1} - w^n\right)/\Delta t$ in (50) and gather terms we can write

$$w_j^{n+1} = \frac{\Delta t \; g_c C\left[P_j^U - P_j^D\right] + I_j w_j^n - g\Delta t\, \Delta Z_j}{\left(I_j + \Delta t\, \dfrac{Kw_j^{n+1}\left|w_j^n\right|}{2\rho_j^n R_j^2 A_j^2}\right)} \tag{50a}$$

which can be shortened symbolically to

$$w_j^{n+1} = Y_j^n\left[P_j^U - P_j^D\right] + D_j \tag{50b}$$

with obvious definitions for $Y_j^n$, the past time "admittance" and $D_j$, the explicit flow and head term. In order to model valve aperture positions, $\boldsymbol{A^2}$ in the form loss term was written as

$$A_j^2 = A_j^2 R_j^2 \tag{52}$$

where

$$\varepsilon \le R_j \le 1 \tag{53}$$

with $\varepsilon$ a small number, $\boldsymbol{1.0\ x10^{-6}}$, in order to avoid division by zero.

Also, note that since the pressure is usually expressed in $\boldsymbol{lbf/in^2}$, a constant is necessary to convert to $\boldsymbol{lbf/ft^2}$. This constant is

$$C = 144 \; in^2 / ft^2 \tag{54}$$

Using the above, the momentum equation becomes:

$$I_j \frac{dw_j}{dt} = g_c\left[P_j^U - P_j^D\right] - g\Delta Z_j - \left(\frac{Kw|w|}{2\rho A^2 R^2}\right)_j \tag{55}$$

The discrete control volume/link conservation and closure relations can now be written as the following:

**Mass**

$$V_N \frac{d\rho_N}{dt} - \sum_t a_t^{fm,to} w_t + \sum_i a_i^{fm,to} w_i = 0$$

or

$$V_N \frac{d\rho_N}{dt} - \sum_t a_{N,t} w_t + \sum_i a_{N,i} w_i = 0 \tag{56a,b}$$

**Momentum (Transient)**

$$I_j \frac{dw_j}{dt} = -g_c C\left[P_j^U - P_j^D\right] - g\Delta Z_j - \left(\frac{Kw|w|}{2\rho A^2 R^2}\right)_j \tag{57}$$

**Energy**

$$\frac{d(Mh)_N}{dt} - \sum_t a_{N,t}(wh)_t + \sum_i a_{N,i}(wh)_i = \frac{C}{J}V_N\frac{dP_N}{dt} + V_N Q_N$$

or

$$V_N\frac{d(\rho h)_N}{dt} - \sum_t a_{N,t}(wh)_t + \sum_i a_{N,i}(wh)_i = \frac{C}{J}V_N\frac{dP_N}{dt} + V_N Q_N$$

(58a,b)

The control volume $V_N$ is fixed in space and the mass is the product of density and volume.

### *Equations of State*

The most popular is

$$\rho_s = \rho_s(P,h)$$

(59)

Others include

$$P_s = P_s(\rho,h), P_s = P_s(\rho,T), \rho_S = \rho_S(P,T)$$

(60)

with the subscript *s* denoting a state function.

These are generally for the two-phase flow homogeneous case. Various subsets can be used for two-phase and single-phase conditions. For single and two-phase conditions the state equation $P_s = P_s(\rho,h)$ or $P_s = P_s(\rho,T)$ is very sensitive to changes in density while (59) is not very sensitive to changes in pressure. This is why Equation (59) is the preferred state equation for use in (T/H) codes.

To place the energy equation in its pre-programming form, we need to take care of the terminal and incident enthalpy link sums in (58b) as:

$$(wh)_t = \frac{w_t}{2}\left[(1+\beta_t)\quad(1-\beta_t)\right]\begin{bmatrix} h_t^u \\ h_t^N \end{bmatrix} = \vec{\beta}^T \vec{h}_t$$

$$(wh)_i = \frac{w_i}{2}\left[(1+\beta_i)\quad(1-\beta_i)\right]\begin{bmatrix} h_i^N \\ h_i^d \end{bmatrix} = \vec{\beta}^T \vec{h}_i$$

(61a,b)

with $\beta_k = |w_k|/(w+\varepsilon)$. $\varepsilon$ is a small number to prevent a divide by zero when $w_k = 0$. This prescription "donors" the upstream/downstream enthalpy to link k, a terminal or incident link. More on this subject is discussed in Appendix I.

For the terminal links to node *N* the upstream nodes are represented by *u* and for the incident links

from node **N** the downstream nodes are represented as **d.** Using (61a,b) in (58) the time differenced relation is

$$\left(\rho V\right)_N \frac{dh_N}{dt} + \left(hV\right)_N \frac{d\rho_N}{dt} - \sum_t a_{N,t} \circ \left(\vec{\beta}^T \vec{h}\right)_t + \sum_i a_{N,i} \circ \left(\vec{\beta}^T \vec{h}\right)_i$$
$$= \left(VQ + V\frac{C}{J}\frac{dP}{dt}\right)_N \tag{62}$$

where we have taken the derivative of the first term in (58). Recall that

$$V_N \frac{d\rho_N}{dt} = \sum_t a_{N,t} \circ w_t - \sum_i a_{N,i} \circ w_i \tag{56}$$

and upon using this in (62) we get

$$h_N^{n+1}\left\{ \left(\rho V\right)_N + \Delta t\left[\sum_t a_{N,t} \circ w_t - \sum_i a_{N,i} \circ w_i\right] - \Delta t \sum_t \frac{a_{N,t} w_t}{2}\left(1 - \beta_t\right)\right.$$
$$\left. + \Delta t \sum_i \frac{a_{N,i} w_i}{2}\left(1 + \beta_i\right)\right\} = \Delta t\, V_N\, Q_N + V_N \frac{C}{J}\left(P^{n+1} - P^n\right)_N$$
$$+ \left(\rho V h^n\right)_N + \Delta t \sum_t \frac{a_{N,t} w_t}{2}\left(1 - \beta_t\right)h_t^u \tag{63}$$
$$- \Delta t \sum_i \frac{a_{N,i} w_i}{2}\left(1 + \beta_i\right)h_i^d$$

Equation (63) is compactly written as

$$\sigma_N h_N^{n+1} = \tau_N + V_N \frac{C}{J}\left(P^{n+1} - P^n\right)_N \tag{64}$$

with obvious definitions for the terms.

**Mass Error**

For this discussion we will drop the volume **N** subscript notation.

The definition of mass error is the difference between the mixture and state density multiplied by the volume or

$$\varepsilon = V\left(\rho_M - \rho_s\left(P,h\right)\right) \tag{65}$$

with the mixture density defined Equation (56) and the state density by (59). Equation (65) can be expanded in a Taylor's series as

$$\varepsilon^{n+1} = \varepsilon^n + \frac{\partial \varepsilon}{\partial \rho_M}\Delta\rho_M + \frac{\partial \varepsilon}{\partial \rho_s}\Delta\rho_s \tag{66}$$

Taking derivatives in (66) by using (65) we get

$$\varepsilon^{n+1} = \varepsilon^n + V\!\!\!\!\!/\,\Delta\rho_M - V\!\!\!\!\!/\,\Delta\rho_s$$

$$\varepsilon^{n+1} = \varepsilon^n + V\!\!\!\!\!/\,\Delta\rho_M - V\!\!\!\!\!/\,\frac{\partial\rho_s}{\partial P}\Delta P - V\!\!\!\!\!/\,\frac{\partial\rho_s}{\partial h}\Delta h$$

(67a,b)

The mixture and state densities will never be exactly equal due to round-off, precision and the accuracy of the property fits or tables for the state density as a function of pressure and enthalpy.

From Equation (56) the mixture density is determined as

$$V\!\!\!\!\!/\,\Delta\rho_{M} = \Delta t\left(\sum_t a_{N,t} w_t - \sum_i a_{N,i} w_i\right)$$

(68)

and using (68) in (67)

$$\varepsilon^{n+1} = \varepsilon^n + \Delta t\left(\sum_t a_{N,t} w_t - \sum_i a_{N,i} w_i\right) - V\!\!\!\!\!/\,\frac{\partial\rho_s}{\partial p}\Delta p - V\!\!\!\!\!/\,\frac{\partial\rho_s}{\partial h}\Delta h$$

(69)

is obtained.

The term $\varepsilon^{n+1}$ is a residual at the new time step. In order to overcome mass error we force this residual to zero locally by setting $\varepsilon^{n+1}$ to zero to get

$$V\!\!\!\!\!/\,\frac{\partial\rho_s}{\partial p}\Delta p + V\!\!\!\!\!/\,\frac{\partial\rho_s}{\partial h}\Delta h = \Delta t\left(\sum_t a_{N,t} w_t - \sum_i a_{N,i} w_i\right) + \varepsilon^n$$

(70)

If we recall the definition of mass error as $\varepsilon = V\!\!\!\!\!/\,(\rho_M - \rho_s)$ we get

$$V\!\!\!\!\!/\,\frac{\partial\rho_s}{\partial p}\Delta p + V\!\!\!\!\!/\,\frac{\partial\rho_s}{\partial h}\Delta h = \Delta t\left(\sum_t a_{N,t} w_t - \sum_i a_{N,i} w_i\right) + V\!\!\!\!\!/\,(\rho_M - \rho_s)^n$$

(71)

with

$$\Delta P = P^{n+1} - P^n$$

$$\Delta h = h^{n+1} - h^n$$

(72)

We restore the **N** subscript notation, to obtain

$$\left(V\!\!\!\!\!/\,\frac{\partial\rho_s}{\partial p}\right)_N P_N^{n+1} + \left(V\!\!\!\!\!/\,\frac{\partial\rho_s}{\partial h}\right)_N h_N^{n+1} - \Delta t\left(\sum_t a_{N,t} w_t - \sum_i a_{N,i} w_i\right) =$$

$$V\!\!\!\!\!/_N\,(\rho_M - \rho_s)_N^n + \left(V\!\!\!\!\!/\,\frac{\partial\rho_s}{\partial p}\right)_N P_N^n + \left(V\!\!\!\!\!/\,\frac{\partial\rho_s}{\partial h}\right)_N h_N^n$$

(73)

and using the relations

$$w_t^{n+1} = Y_t^n \left[ P_t^U - P_t^D \right] + D_t$$
$$w_i^{n+1} = Y_i^n \left[ P_i^U - P_i^D \right] + D_i$$

(74)

$$\left( \Psi \frac{\partial \rho_s}{\partial p} \right)_N P_N^{n+1} + \left( \Psi \frac{\partial \rho_s}{\partial h} \right)_N h_N^{n+1} -$$

$$\Delta t \left( \sum_t a_{N,t} \left( Y_t^n \left[ P_t^U - P_N^N \right] + D_t \right) - \sum_i a_{N,i} \left( Y_i^n \left[ P_i^N - P_i^D \right] + D_i \right) \right) =$$

(75)

$$\Psi_N \left( \rho_M - \rho_s \right)_N^n + \left( \Psi \frac{\partial \rho_s}{\partial p} \right)_N P_N^n + \left( \Psi \frac{\partial \rho_s}{\partial h} \right)_N h_N^n$$

Finally, using

$$h_N^{n+1} = \frac{\tau_N + \Psi_N \dfrac{C}{J} \left( P^{n+1} - P^n \right)_N}{\sigma_N}$$

(76)

in (75) the relation

$$\left( \Psi \frac{\partial \rho_s}{\partial p} \right)_N P_N^{n+1} + \left( \Psi \frac{\partial \rho_s}{\partial h} \right)_N \left( \frac{\Psi_N \dfrac{C}{J} P_N^{n+1}}{\sigma_N} \right) -$$

$$\Delta t \left( \sum_t a_{N,t} \left( Y_t^n \left[ P_t^U - P_N^N \right] \right) - \sum_i a_{N,i} \left( Y_i^n \left[ P_i^N - P_i^D \right] \right) \right) =$$

(77)

$$\Psi_N \left( \rho_M - \rho_s \right)_N + \left( \Psi \frac{\partial \rho_s}{\partial p} \right)_N P_N^n + \left( \Psi \frac{\partial \rho_s}{\partial h} \right)_N \left( h_N^n - \frac{\tau_N}{\sigma_N} + \frac{\Psi_N \dfrac{C}{J} P_N^n}{\sigma_N} \right)$$

$$+ \Delta t \sum_t a_{N,t} D_t - \Delta t \sum_i a_{N,i} D_i$$

is obtained.

Once the pressure equation is solved we can get the flows from

$$w_j^{n+1} = Y_j^n \left[ P_j^U - P_j^D \right] + D_j$$

(78)

and using the updated flows we can solve for the nodal enthalpies from (62) and (76). Finally, the mixture density can be solved from (68).

The equations and state relation used for the three-equation model are:

*Pressure*

$$\left(V\frac{\partial\rho_s}{\partial p}\right)_N P_N^{n+1} + \left(V\frac{\partial\rho_s}{\partial h}\right)_N\left(\frac{V_N\frac{C}{J}P_N^{n+1}}{\sigma_N}\right) -$$

$$\Delta t\left(\sum_t a_{N,t}\left(Y_t^n\left[P_t^U - P_t^N\right]\right)\sum_i a_{N,i}\left(Y_i^n\left[P_i^N - P_i^D\right]\right)\right) =$$

$$V_N\left(\rho_M - \rho_s\right)_N + \left(V\frac{\partial\rho_s}{\partial p}\right)_N P_N^n + \left(V\frac{\partial\rho_s}{\partial h}\right)_N\left(h_N^n - \frac{\tau_N}{\sigma_N} + \frac{V_N\frac{C}{J}P_N^n}{\sigma_N}\right)$$

$$+\Delta t\sum_t a_{N,t}D_t - \Delta t\sum_i a_{N,i}D_i$$

(79)

*Enthalpy*

$$\sigma_N h_N^{n+1} = \tau_N + V_N\frac{C}{J}\left(P^{n+1} - P^n\right)_N$$

(80)

*Momentum*

$$w_j^{n+1} = Y_j^n\left[P_j^U - P_j^D\right] + D_j$$

(81)

*Mixture Density*

$$V_N\frac{d\rho_N}{dt} = \sum_t a_{N,t}\circ w_t - \sum_i a_{N,i}\circ w_i$$

(82)

*State Equation*

$$\rho_S = \rho_S\left(P,h\right)$$

(83)

## 5.0 Compressible Two-Phase Flow

The final mechanics of a two-phase flow program are the constitutive relations and derivatives. To present these relations we need some definitions first.

The static or mass quality is

$$X = \frac{M_g}{M} \tag{84}$$

where $M_g$ is the gas mass and $M$ is the total mass of the system,

$$M = M_g + M_l \tag{85}$$

with $M_l$ as the liquid mass. Since $M_l = M - M_g$ we can divide this by $M$ to get the relation

$$1 - X = \frac{M_l}{M} \tag{86}$$

Equation (85) can also be written as

$$\rho V = \rho_g V_g + \rho_l V_l \tag{87}$$

since mass is the product of density and volume. Dividing through by the volume gives

$$\rho = \alpha \rho_g + (1 - \alpha) \rho_l \tag{88}$$

since $V = V_g + V_l$. The definition for the gas void fraction is

$$\alpha = \frac{V_g}{V} \tag{89}$$

and

$$\alpha_l = (1 - \alpha) = \frac{V_l}{V} \tag{90}$$

for the liquid void fraction.

From the definition of static quality we can write

$$X = \frac{M_g}{M} = \frac{\rho_g V_g}{\rho V} = \frac{\alpha \rho_g}{\rho} \tag{91}$$

which gives us a relationship between quality and void fraction. From the relation

$$V = V_g + V_l \tag{92}$$

we get

$$V = v M = v_g(P)M_g + v_l(P)M_l$$
or
$$\tag{93}$$
$$v = v_g(P)\frac{M_g}{M} + v_l(P)\frac{M_l}{M} = Xv_g(P) + (1-X)v_l(P)$$

for water and steam at two-phase saturation conditions. $v$ is the specific volume, the inverse of the density. The $v_g$, $v_l$, $\rho_g$ and $\rho_l$ are functions of pressure at saturation. From (84) the maximum amount of gas is when $0 \leq M_g \leq M$ so that the quality ranges from

$$0 \leq X \leq 1 \tag{94}$$

and the void fraction range is

$$0 \leq \alpha \leq 1 \tag{95}$$

For the extensive enthalpy (units of Btu) we can write

$$H = H_g + H_l$$
$$Mh = M_g h_g + M_l h_l$$
$$Mh = M_g h_g + (M - M_g)h_l$$
$$h = Xh_g(p) + (1-X)h_l(p)$$
$$\tag{96a,b,c,d}$$

from the definition of the static quality. $h$ is the mixture enthalpy, $h_g(p)$ is the saturation gas enthalpy and $h_l(p)$ is the saturation liquid enthalpy, all intensive quantities. From (96)

$$X_e = \frac{h - h_l(p)}{h_g(p) - h_l(p)} \tag{97}$$

which is referred to as the "equilibrium" quality.

**State Derivatives**

The two derivatives used in *(79)* are:

$$\frac{\partial \rho_s(P,h)}{\partial P} \quad and \quad \frac{\partial \rho_s(P,h)}{\partial h} \tag{98}$$

where it is inferred that taking the derivative with respect to (wrt) *P* that the enthalpy *h*, is held constant and vice versa.

It is a common practice to evaluate these derivatives in terms of specific volumes, since most of the fits and table data are in this form.

$$\frac{\partial \rho_s (P,h)}{\partial P} = \frac{\partial \frac{1}{v_s}}{\partial P} = -\frac{1}{v_s^2}\frac{\partial v_s}{\partial P}$$

$$\frac{\partial \rho_s (P,h)}{\partial h} = \frac{\partial \frac{1}{v_s}}{\partial h} = -\frac{1}{v_s^2}\frac{\partial v_s}{\partial h}$$

(99)

The derivatives and densities have to be defined for three regions:

$$X_e \le 0.0 \qquad \text{\textit{for subcooled liquid}}$$
$$0.0 < X_e < 1.0 \qquad \text{\textit{for stauration mixture}}$$
$$X_e \ge 1.0 \qquad \text{\textit{for superheated steam}}$$

(100)

For the first and third regions the derivatives of (99) are applicable. Polynomial curve fits and tables can be used for the properties. For the two-phase region the constitutive properties and derivatives are more complicated.

**Two-Phase Derivatives**

For the two-phase region:

$$v = X_e v_g (P) + (1 - X_e)v_l (P)$$

(101)

Recall that the two independent variables are **h** and **P**. The derivative of specific volume (wrt) h is given as:

$$\frac{\partial v}{\partial h} = \frac{\partial}{\partial h}\Big[X_e v_g (P) + (1 - X_e)v_l (P)\Big]$$

*but*

$$X_e = \frac{h - h_l (p)}{h_g (p) - h_l (p)}$$

(102)

*so that*

$$\frac{\partial v}{\partial h} = \frac{\partial}{\partial h}\left[\left(\frac{h - h_l (p)}{h_g (p) - h_l (p)}\right)v_g (P) + \left(1 - \left(\frac{h - h_l (p)}{h_g (p) - h_l (p)}\right)\right)v_l (P)\right]$$

$$\frac{\partial v}{\partial h} = \frac{v_g (P) - v_l (P)}{h_g (p) - h_l (p)} = \frac{v_{lg}}{h_{lg}}$$

For the phasic pressure derivative

$$\frac{dv(P,h(P))}{dP} = \frac{\partial v}{\partial P} + \frac{\partial v}{\partial h}\frac{\partial h}{\partial P}$$

or

$$\frac{\partial v}{\partial P} = \frac{dv(P,h)}{dP} - \frac{\partial v}{\partial h}\frac{\partial h}{\partial P}$$

(103)

We know that

$$\frac{dv}{dP} = \frac{d}{dP}\Big[ X_e v_g(P) + (1-X_e)v_l(P) \Big]$$

(104)

for the "explicit" function of pressure or

$$\frac{dv}{dP} = X_e \frac{dv_g(P)}{dP} + (1-X_e)\frac{dv_l(P)}{dP}$$

(105)

Recall that $\partial v/\partial h$ is already known from (102) and we need to compute $dh/dP$. Thus,

$$\frac{dh}{dP} = \frac{d}{dP}\Big[ X_e h_g(p) + (1-X_e)h_l(p) \Big]$$

or

$$\frac{dh}{dP} = X_e \frac{dh_g(p)}{dP} + (1-X_e)\frac{dh_l(p)}{dP}$$

(106)

Recall that the pressure derivative is

$$\frac{\partial v}{\partial P} = \frac{dv(P,h)}{dP} - \frac{\partial v}{\partial h}\frac{\partial h}{\partial P}$$

(107)

or

$$\frac{\partial v}{\partial P} = X_e \frac{dv_g(P)}{dP} + (1-X_e)\frac{dv_l(P)}{dP} - \frac{v_{lg}}{h_{lg}}\left( X_e \frac{dh_g(p)}{dP} + (1-X_e)\frac{dh_l(p)}{dP} \right)$$
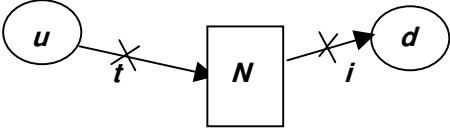
(108)

The final form of this relation is

$$\frac{\partial v}{\partial P} = X_e\left[ \frac{dv_g(P)}{dP} - \frac{v_{lg}}{h_{lg}}\frac{dh_g(p)}{dP} \right] + (1-X_e)\left[ \frac{dv_l(P)}{dP} - \frac{v_{lg}}{h_{lg}}\frac{dh_l(p)}{dP} \right]$$

(109)

When computing these properties the past time values of dependent and independent variables are used unless an iterative procedure is applied.

## 6.0 Pressure Matrix & Boundary Conditions

Equation (79) showed how we eliminated the energy and momentum equations in favor of the pressure. This section shows how explicit flow inflow or outflow can be implemented. We will gather all the terms in the pressure equation of (79) and form a pressure matrix for the solution. After the pressures are obtained we will use them to get h flows and then compute the energy propagation.

We need to see how all the terms will be programmed in the pressure matrix solution of $\underline{\underline{a}}\,\vec{p}=\vec{b}$ and take the approach of examining the indices of the links and volumes in Equation (79). The row index is **N** in the matrix for the **Nth** pressure node. The **t** and **i** indices denote the link numbers or types, the **t** link is the link terminal to node **N** from node **u** and **i** is the link leaving node **N** to node **d.** This situation is depicted in the figure below. For the situation shown of one terminal link and one incident link, we write (79) as



$$\left[ \mathbb{V}\frac{\partial \rho_s}{\partial p} + \mathbb{V}\frac{\partial \rho_s}{\partial h}\left( \frac{\mathbb{V}\frac{C}{J}}{\sigma} \right) \right]_N P_N^{n+1} -$$

$$\Delta t \left( a_{N,t}\left( Y_t^n\left[ P_t^U - P_t^N \right] \right) - a_{N,i}\left( Y_i^n\left[ P_i^N - P_i^D \right] \right) \right) = \tag{79}$$

$$\mathbb{V}_N\left( \rho_M - \rho_s \right)_N + \left( \mathbb{V}\frac{\partial \rho_s}{\partial p} \right)_N P_N^n + \left( \mathbb{V}\frac{\partial \rho_s}{\partial h} \right)_N \left( h_N^n - \frac{\tau_N}{\sigma_N} + \frac{\mathbb{V}_N\frac{C}{J}P_N^n}{\sigma_N} \right)$$

$$+\Delta t\, a_{N,t}D_t - \Delta t\, a_{N,i}D_i$$

At this point it might be wise to prepare the computer code for using flow boundary conditions. We do this by examining the pressure solution without the energy terms (for convienence) as:

$$\left( \mathbb{V}\frac{\partial \rho_s}{\partial P} \right)_N \left( P_N^{n+1} - P_N^n \right) - \Delta t\left( 1 - H(t) \right)a_{N,t}w_t + \Delta t\left( 1 - H(i) \right)a_{N,i}w_i =$$

$$\mathbb{V}_N\left( \rho_M - \rho_s \right)_N^n + \Delta t H(t)a_{Nt}w_t - \Delta t H(i)a_{Ni}w_i \tag{110}$$

with $H(t)$ and $H(i)$ as unit functions associated with the two types of links, terminal and incident.

The $H(t)$ and $H(i)$ functions are given by the general relation

$$H(k) = 1,\ iff\ \ link\ k\ is\ a\ boundary\ link,\ i.e.,\ a\ flow\ boundary\ condition$$

$$\tag{111}$$

$H(k) = 0$, *iff link k is not a boundary link, i.e., a regular pressure difference link*

If the links are regular links (no flow boundary condition) connecting the two volumes, the $H(k)$ are zero, and we get (79). If link $t$, the terminal link is a flow boundary condition then (110) becomes

$$\left( V \frac{\partial \rho_s}{\partial p} \right)_N (P_N^{n+1} - P_N^n) + \Delta t\, a_{N,i} w_i = V_N \left( \rho_M - \rho_s \right)_N^n + \Delta t\, a_{N,t} w_t \tag{112}$$

This is an inflow boundary condition from volume $u$ to $N$ at the terminal link.

The mass flows at links $t$ and $i$ can be written, without the head terms, as

$$w_t = Y_t \left[ P_t^u \quad -P_t^N \right] = Y_t \left[ 1 \quad -1 \right] \begin{bmatrix} P_t^u \\ P_t^N \end{bmatrix}$$

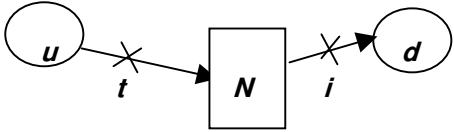$$w_i = Y_i \left[ P_i^N \quad -P_i^d \right] = Y_i \left[ 1 \quad -1 \right] \begin{bmatrix} P_i^N \\ P_i^d \end{bmatrix} \tag{113}$$

and upon using these relations in (112) we get

$$\left( V \frac{\partial \rho_s}{\partial p} \right)_N (P_N^{n+1} - P_N^n) -$$
$$\Delta t\, Y_t (1 - H(t)) a_{N,t} \left[ P_t^u \quad -P_t^N \right] +$$
$$\Delta t\, Y_i (1 - H(i)) a_{N,i} \left[ P_i^N \quad -P_i^d \right] =$$
$$V_N \left( \rho_M - \rho_s \right)_N^n + \Delta t\, H(t) a_{N,t} w_t - \Delta t\, H(i) a_{N,i} w_i \tag{114}$$

which can be rewritten as

$$\left( V \frac{\partial \rho_s}{\partial p} \right)_N (P_N^{n+1} - P_N^n) -$$
$$\Delta t\, Y_t (1 - H(t)) a_{N,t} \left[ 1 \quad -1 \quad 0 \right] \begin{bmatrix} P^{t,u} \\ P^{t,N} \\ 0 \end{bmatrix} +$$
$$\Delta t\, Y_i (1 - H(i)) a_{N,i} \left[ 0 \quad 1 \quad -1 \right] \begin{bmatrix} 0 \\ P^{i,N} \\ P^{i,d} \end{bmatrix} = \tag{115}$$
$$V_N \left( \rho_M - \rho_s \right)_N^n + \Delta t\, H(t) a_{N,t} w_t - \Delta t\, H(i) a_{N,i} w_i$$

In equation (115) the **t** and **i** indices in the second and third terms on the left are dummy indices that can be contracted, not summed. For the **Nth** row of the grid show below the **a** matrix coefficients in $\underline{\underline{a}}\,\vec{p}=\vec{b}$ are indexed as



$$\left(\nu\frac{\partial\rho_s}{\partial p}\right)_N\left(P_N^{n+1}-P_N^n\right)-$$

$$\Delta t\,Y_t\left(1-H(t)\right)\begin{bmatrix}a_{N,u} & -a_{N,N} & 0\end{bmatrix}\begin{bmatrix}P_u\\P_N\\0\end{bmatrix}+$$

$$\Delta t\,Y_i\left(1-H(i)\right)\begin{bmatrix}0 & a_{N,N} & -a_{N,d}\end{bmatrix}\begin{bmatrix}0\\P_N\\P_d\end{bmatrix}=$$

$$\nu_N\left(\rho_M-\rho_s\right)_N^n+\Delta t\,H(t)a_{N,u}w_u-\Delta t\,H(i)a_{N,d}w_d$$

(116)

The $a_{N,t}$ and $a_{N,i}$ have been changed to more correctly indicate the positions in the pressure matrix. Also, the **t** and **i** subscripts correspond to **u** (upstream) and **d** (downstream) links or junctions.

We can see that for two links connecting to node **N** we get two diagonal contributions to $a_{NN}$ in the pressure matrix of 1.0, one for $a_{N,t\,(to\ node)}$ of 1.0 from the terminal link, one for $a_{N,i\,(from\ node)}$ of 1.0 from the incident link, one off-diagonal contribution from the terminal link at $a_{N\,u}$ and $a_{N\,d}$ from the incident link of 1.0. If we connected another terminal or incident link to node **N** we would see the same pattern of $\sum_j Y_j$ to the diagonal members of the matrix and $-Y_t\ and -Y_i$ to the off-diagonal members. This method is known as assembly by nodes or rows. It is called that because we look at a single node, not a link, and assemble the adjoining link nodal contributions to the matrix. We also have the topology to add in a boundary flow link. Of course, the number of links connecting to a node would be read in as Fortran input, as is shown in Appendix II.

The following Fortran code shows how the assembly processes would occur for input processing and in the pressure matrix assembly.

```
c -- input processing
c
c -- get the number of links per node and their values
```

```fortran
c
c -- first initialize arrays
c
      do i =1,nvols
c   itl is the # of terminal links to node i
      itl(i) = 0
c   iil is the # of incident links from node i
         iil(i) = 0
c    maxnjnds is the max # of links surrounding a node, set at 6
         do j = 1,maxnjnds
c   jt, array of node i, for 6 terminal links to node i
c   ji, array of node i, for 6 incident links from node i
         jt(i,j)=0
         ji(i,j)=0
         enddo
         enddo
c
         do i = 1, nvols
         do j = 1, links
c
c
c    tooo node = N, then j=t, a terminal link
c
c
         if (tooo(j).eq.i) then
         itl(i)=itl(i)+1
         jt(i,itl(i))=j
         endif
c
c
c    from node = N, then j=i, an incident link
c
c
         if (from(j).eq.i) then
         iil(i)=iil(i)+1
         ji(i,iil(i))=j
         endif
         enddo
         enddo
```

The matrix assembly process following (116) would take place in two steps:

1. The coefficients of the new time pressure would be assembled into the matrix pressure solution, $\underline{\underline{a}}\ \vec{p} = \vec{b}$ on the left hand side diagonal terms and the coefficients of the past time pressure term would be placed on the right hand side, using the following fortran.

```fortran
c
c – diagonal & rhs assembly
c
```

35

```
      do i = 1,nvols
          if (ipf(i).eq.0) then
      amat(i,i) = vol(i)*drodp(i)
      brhs(i) = vol(i)*drodp(i)*pvol(i)
          else
c
          amat(i,i) =1.0
          brhs(i) = pvol(i)
          endif
          enddo
c
```

The **if** statement sets the diagonal term to 1.0 and sets the right hand side term to the pressure if node **i** is a boundary node.

2.  The row by row assembly process following the mnemonic of equation (116) is given by the following fortran.

```
c
      do i = 1,nvols
          if (ipf(i).eq.1) goto 13
c    sum over terminal links to node i
      do k=1,itl(i)
c    gives the link #
          j = jt(i,itl(i))
c    chk to see if node # is i
      nt = tooo(j)
          nu = from(j)
c    diagonal & off-diagonal term
          if (nt.eq.i) then
          amat(i,i) = amat(i,i)  +dt*c144*ylink(j)*(1.0-ibw(j))
          amat(i,nu)= amat(i,nu) -dt*c144*ylink(j)*(1.0-ibw(j))
          brhs(i)=brhs(i)+dt*ibw(j)*wlink(j)
          endif
          enddo
c    sum over the incident links to node i
      do k=1,iil(i)
c    gives the link #
          j = ji(i,iil(i))
c    chk to see if node # is i
      ni = from(j)
          nd = tooo(j)
c    diagonal & off-diagonal term
          if (ni.eq.i) then
          amat(i,i) = amat(i,i)  +dt*c144*ylink(j)*(1.0-ibw(j))
          amat(i,nd)= amat(i,nd) -dt*c144*ylink(j)*(1.0-ibw(j))
          brhs(i)=brhs(i)-dt*ibw(j)*wlink(j)
          endif
          enddo
13    continue
```

enddo

Prior to performing the link assembly on a row by row basis we perform a number of operations for the link quantities. The fortran for the entire code is shown in Appendix II.

As shown previously, there are four contributions from two links that connect to a node. These are the two diagonal contributions to $a_{NN}$, one for $a_{N,t(to\ node)}$ from the terminal link, one for $a_{N,i(from\ node)}$ from the incident link, one off-diagonal contribution from the terminal link at $a_{Nu}$ and $a_{Nd}$ from the incident link. An easier method is just to sum over all the links and add their contributions to the matrix rows and columns. This form of matrix assembly is known as assembly by links. The following Fortran shows how this can be accomplished.

```
c
c --  add in link contributions
c
    do 20 i = 1,links
c
c    nd = terminal link node
c    nu = incident link node
c
    nd = tooo(i)
    nu = from(i)
c
c --  add link terms to pressure equation
c
    amat(nd,nu) = amat(nd,nu) - dt*(1-ibw(i))
   &*c144*ylink(i)
    amat(nu,nd) = amat(nu,nd) - dt*(1-ibw(i))
   &*c144*ylink(i)
    amat(nd,nd) = amat(nd,nd) + dt*(1-ibw(i))
   &*c144*ylink(i)
    amat(nu,nu) = amat(nu,nu) + dt*(1-ibw(i))
   &*c144*ylink(i)
c
c --  add in link rhs flow b.c. terms
c
    brhs(nd) = brhs(nd) + ibw(i)*dt*wlink(i)
    brhs(nu) = brhs(nu) - ibw(i)*dt*wlink(i)
```

This code is listed in Appendix II.

In order to keep the solution of the pressure matrix elementary a square matrix solver will be used initially. A number of different methods will be explored for instituting boundary conditions during the course. A boundary node can be initialized in the program by the use of an array, such as

*IPF(I) =1*, if node *I* is a boundary node

*IPF(I) =0*, if not

The simplest way to approach this is the method of Payne & Irons by writing the matrix relation as

$$
\begin{pmatrix}
a_{11} & \cdots & a_{1,NV} \\
\vdots & \ddots & \vdots \\
a_{NV,1} & \cdots & a_{NV,NV}
\end{pmatrix}
\begin{bmatrix}
P_1 \\
P_2 \\
P_{NV}
\end{bmatrix}
=
\begin{bmatrix}
B_1 \\
B_2 \\
B_{NV}
\end{bmatrix}
\tag{117}
$$

and to replace the diagonal terms corresponding to a boundary condition by

*B(I)=BIG x P(I) x A(I,I)* (118)

*A(I,I) = BIG x A(I,I),IPF(I))* (119)

where *BIG* is a large number, *BIG=1.e+20* and *IPF(I)* corresponds to a boundary node number. This can be programmed as

```
c
      do i=1,NV
      if (IPF(I).eq.1) then
      b(i)=BIG*p(i)*a(i,i)
      a(i,i)=BIG*a(i,i)
      else
      continue
      endif
      enddo
```

It is important to note that we do *b(i)=BIG*p(i)*a(i,i)* before *a(i,i)=BIG*a(i,i)*, otherwise we will get a code error.

A square Gauss Elimination routine can be used from Brebbia[4]. As we perform the Gauss Elimination the effect of the boundary condition implementation is to divide by a very large number to set intervening terms to zero except for the boundary row. Other methods can be used such as *A(I,I)=1.0* for boundary row *I* and *A(I,J)=A(J,I)=0* with *B(I)=P(I).* Other forms of matrix storage can be used such as bandwidth storage.

# 7.0 Comparison to Experiment

In order to verify the code two independent assessment problems were performed. The first is Edwards Blowdown and the second is the General Electric Level Swell experiment.

**Edwards Blowdown (Description & Figure from the TRAC V&V Manual)**

Edwards horizontal-pipe blowdown experiment studied depressurization phenomena of initially non-flowing subcooled water. The experimental apparatus consisted of a 4.096-m-long straight steel pipe with a 0.073-m i.d. The apparatus was designed for a maximum 17.24-MPa (2000 psia) pressure at temperatures to 616.5 K (460 F). The discharge end of the horizontal pipe was sealed with a 0.0127-m-thick glass disk.

The pipe was filled with demineralized water. A hydraulic pump and a control valve regulated the system pressure. The pipe was evacuated by a vacuum pump before it was filled with water. Before the glass disk was ruptured, the pipe was isolated from the supply tank to prevent the discharge of cold water into the pipe during blowdown.
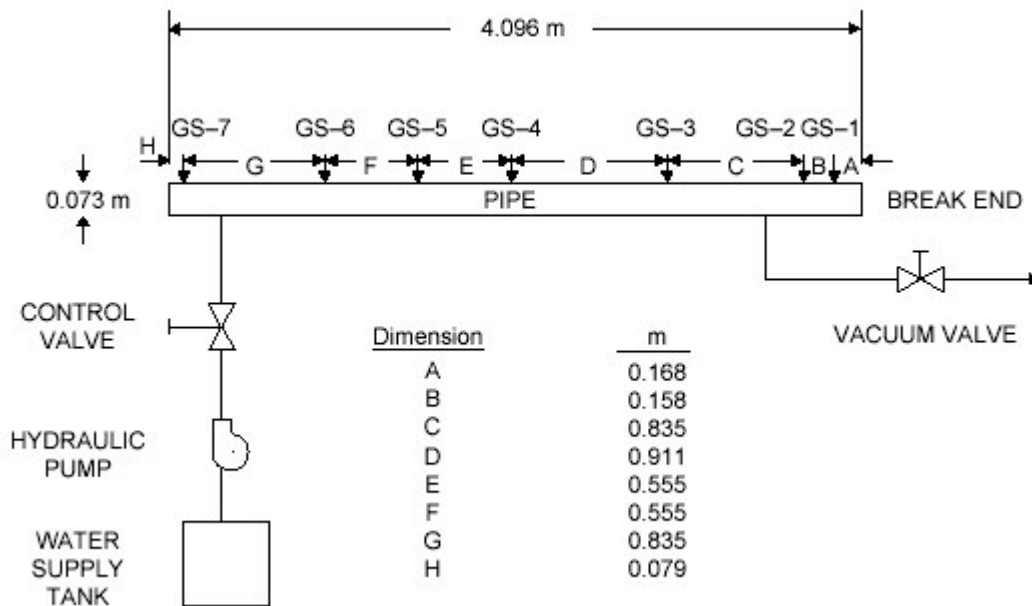


*Figure 10.0 Edwards Blowdown Apparatus*

Seven volumes and six junctions (links) are used for the model. Figures 11 and 12 show the pressure and mass flow at the break as a function of time.

## Edwards Blowdown Break Pressure



**Figure 11.0**
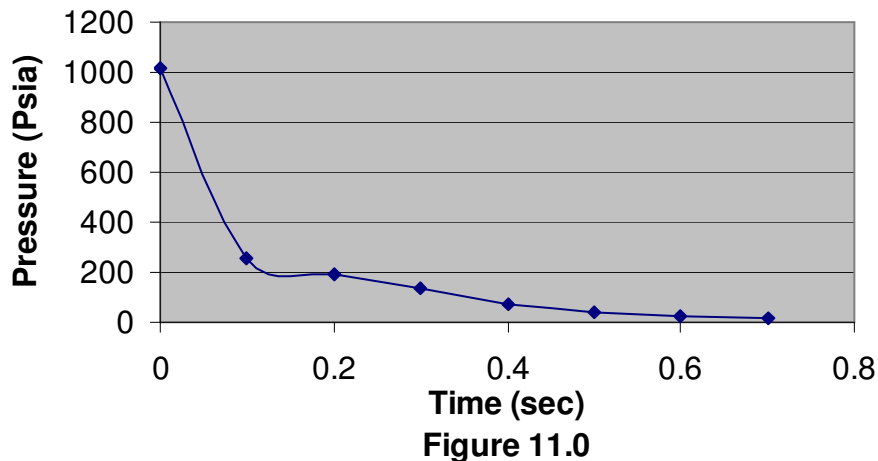
## Edwards Blowdown Break Flow



**Figure 12.0**

**GE Level Swell Experiment (Aumiller Reference 6)**

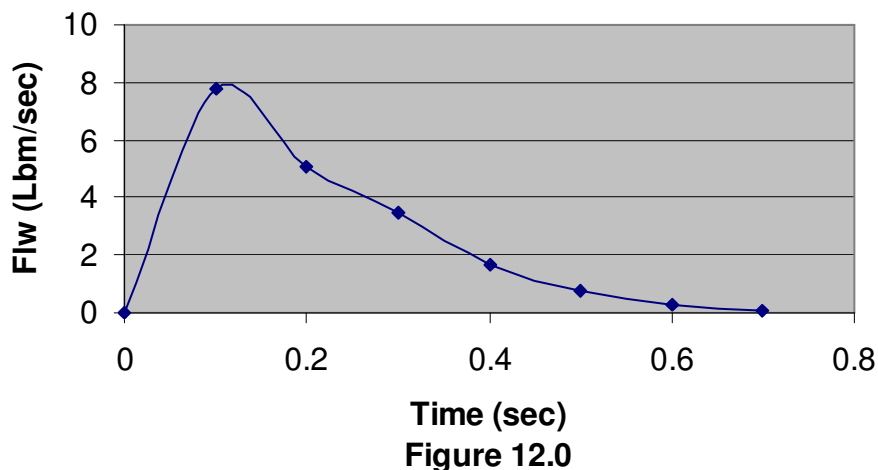The GE Level Swell experiments were designed to measure transient void fraction profiles in a large tank, depressurized via a blowdown line and orifice. Two different vessels sizes (1 and 4 ft) nominal diameter) were used in the experimental program. This paper will focus on test number 1004-3 performed with the smaller of the two vessels, the one-foot diameter vessel.

A schematic of the experimental facility for the small vessel blowdown tests is shown in Figure 13.0. The experimental vessel was constructed from a length of 12 inch, schedule 80 pipe. The volume of the vessel is 0.28 m3 (10.0 ft3). In an attempt to prevent liquid from being entrained out of the test, the blowdown pipe was connected near the top of the vessel. The depressurization rate was controlled via an orifice in the blowdown line. For the test being considered, the diameter of the blowdown orifice was 0.00952 m (0.375 in). A perforated plate could be inserted in the vessel to examine the effect of

a hydraulic resistance on the experiments; however, this plate was not installed for test number 1004-3. The instrumentation of the test included one absolute and six differential pressure gauges and several temperatures detectors. As shown in Figure 13.0, the regions between adjacent pressure taps are referred to as Levels (or segments) and are numbered sequentially starting at the bottom. The differential pressure measurements were used to infer the void fraction in each segment by assuming that hydrostatic head was the only component contributing to the pressure difference.

The height of the two-phase level was determined using a two-step process. First, the segment containing the two-phase level was heuristically determined using the axial void profile in the vessel. Next the position of the two-phase level in that segment was calculated assuming the void fraction below the two-phase level was equal to the void fraction in the segment directly beneath it.

The initial conditions for test number 1004-3 were a system pressure of 6.92 MPa (1011 psia) and a water level of 3.167 m (10.4 ft). Since the experimental fluid temperatures were not included in the test report, the initial liquid temperature was assumed to correspond to the saturation temperature, 559 K (546 °F).
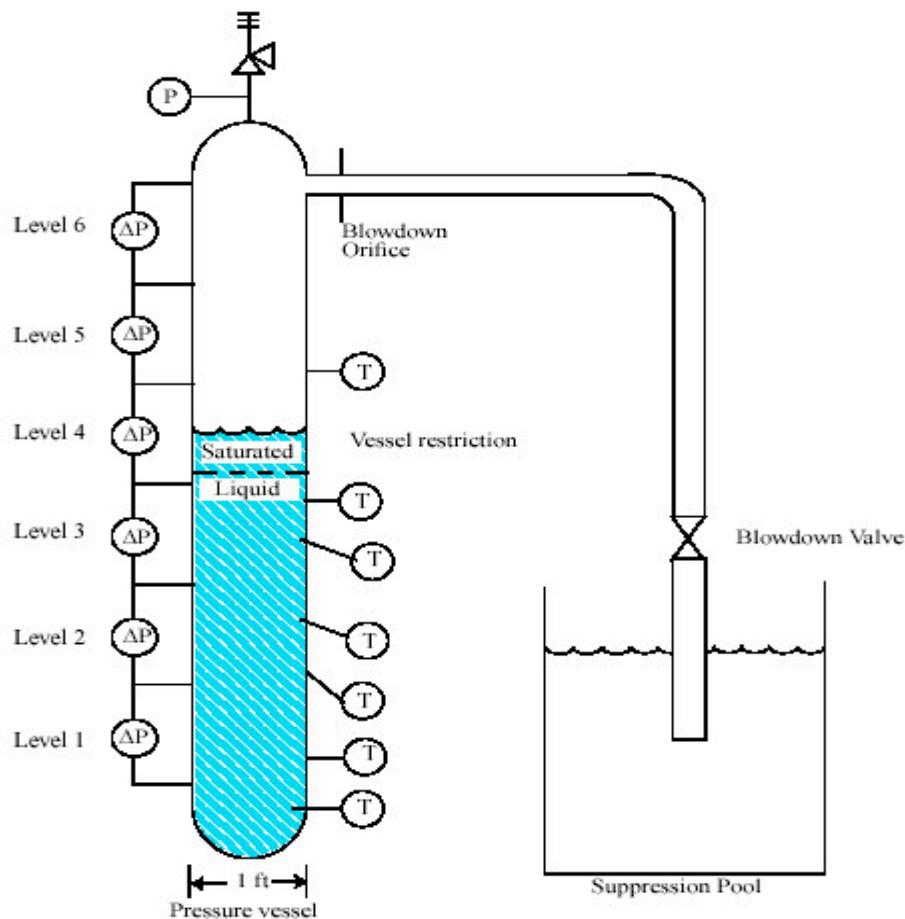


*Figure 13.0 GE Level Swell*

As in the previous simulation seven nodes and six junctions (links) were used for the model. Figure 14.0 shows the pressure at the top of the vessel for two form losses at the break of 1.0 and 10.0 respectively. Obviously, the larger form loss results in a slower depressurization.

**GE Level Swell**

Pressure (Psia) vs Time (sec)

K=1.0
K=10.0

**Figure 14.0**
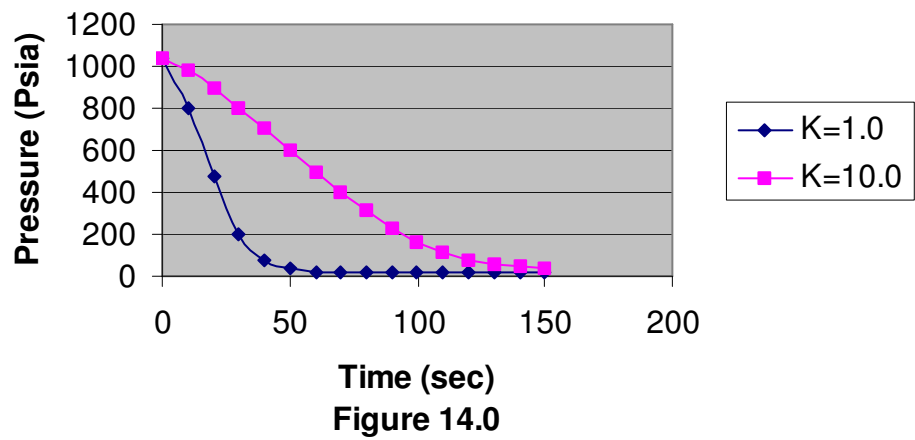
## 8.0 Summary

A Thermal Hydraulics (T/H) Primer has been written for the control volume method. The Primer illustrates the code through theory and programming. The Fortran program and input files can be downloaded from the web at:

http://www.microfusionlab.com

The author and Microfusionlab take no responsibility for the use of the code or any mistakes in this manual or the code. It is meant as an educational exercise not a safety analysis. The Fortran version uses a less accurate version of the steam tables. The Fortran code version is not meant for commercial or safety analysis applications. Microfusionlab maintains a copyright on the use of this code except for educational purposes.

A graphics version of the code written in C is available for simulation. Contact Microfusionlab at the above web site for a demo of the commercial version.

# References

1.0 "RELAP5-3D© Code Manual Volume I: Code Structure, System Models and Solution Methods," Revision 2.2 October 2003, *The RELAP5-3D© Code Development Team,* INEEL-EXT-98-00834. http://www.inel.gov/relap5/r5manuals/ver_2.2/vol1.pdf

2.0 L. Smith and R. Dixon, "Modular Modeling System (MMS): A Code for the Dynamic Simulation of Fossil and Nuclear Power Plants," EPRI CS/NP-2989, March 1983.

3.0 Richard T. Lahey and Frederick J. Moody, *The Thermal-Hydraulics of Boiling Water Reactors,* Second Edition, American Nuclear Society, 1993.

4.0 R. A. Adey and C. A. Brebbia, "Basic Computational Techniques for Engineers," John Wiley and Sons, 1983.

5.0 D. Liles, et. al, "TRAC-Pf1/MOD1 Correlations and Models," NUREG/CR-5059, LA-11208-MS, December 1988.

6.0 D. L. Aumiller, et. al, "A New Assessment of Relap5-3D using a General Electric Level Swell Problem," USDOE Contract No. DE-AC11-98PN38206, BAPL.

# Appendix I

Equation (38), the discrete version of the advection equation, is used for illustrating the matrix assembly and the donoring.

$$\mathcal{V}_N \frac{df_N}{dt} = \sum_t a_{N,t} \left(f\, VA\right)_t - \sum_i a_{N,i} \left(f\, VA\right)_i + S_N \mathcal{V}_N \qquad (38)$$

This equation can be compacted into the relation

$$\mathcal{V}_N \frac{df_N}{dt} = -\sum_k a_{N,k} \left(f\, VA\right)_k + S_N \mathcal{V}_N \qquad (AI\text{-}1)$$

where k denotes an inlet or outlet link and:

$$a_{N,k} = -1, iff\ k = t$$
$$a_{N,k} = 1, iff\ k = i \qquad (AI\text{-}2)$$
$$a_{N,k} = 0, iff\ k\ not\ linked\ to\ node\ N$$

The $a_{N,k}$ is -1.0 for flow into volume **N**, link **k** is a terminal link **t** and $a_{N,k}$ is 1.0 for flow out of **N**, link **k** is an incident link **i**. The $a_{N,k}$ as defined in AI-1 and AI-2 are the dot product (cosine) of the flow or velocity vector with the outward unit normal vector.
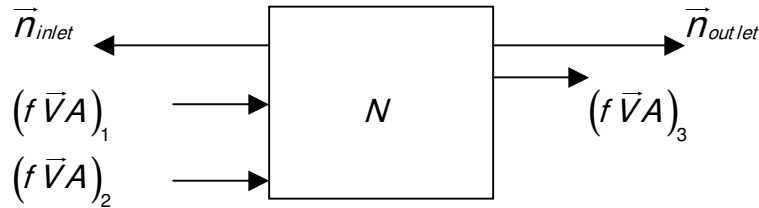


For the figure above we can see that the cosine of the angle between the flux vectors of $\left(f\,\vec{V}A\right)_1$ and $\left(f\,\vec{V}A\right)_2$ and the area unit normal $\vec{n}_{inlet}$ on the inlet face is –1.0 or $a_{N,1} = a_{N,2} = -1.0$. Conversely, the cosine of the angle between $\left(f\,\vec{V}A\right)_3$ and $\vec{n}_{outlet}$ is 1.0 or $a_{N,3} = 1.0$. Using this information in AI-1 we get

$$\mathcal{V}_N \frac{df_N}{dt} = -a_{N,1} \left(f\, VA\right)_1 - a_{N,2} \left(f\, VA\right)_2 - a_{N,3} \left(f\, VA\right)_3 + S_N \mathcal{V}_N \qquad (AI\text{-}2a)$$

and upon using the values for the $a_{N,k}$ we get

$$V_N \frac{df_N}{dt} = \left(f\,VA\right)_1 + \left(f\,VA\right)_2 - \left(f\,VA\right)_3 + S_N\,V_N \tag{AI-2c}$$

Since we already know that there are only two types of links, terminal links to node $N$ for which we can use $a_{N,t}$ with a poisitve sign in (38) and incident links with $a_{N,i}$ using a negative sign in (38), the programming is easier and more lucid. If we were doing more complex geometries then AI-1 would be more general. Thus, in our notation the $a_{N,k}$ are used simply as a matrix location device.

For the flow directions fixed as in Figure AI.1, the Equation (38) becomes

$$V_1 \frac{df_1}{dt} = a_{1,3}\left(f\,VA\right)_3 - a_{1,1}\left(f\,VA\right)_1 + V_1 S_1$$

$$V_2 \frac{df_2}{dt} = a_{2,1}\left(f\,VA\right)_1 - a_{2,2}\left(f\,VA\right)_2 + V_2 S_2 \tag{AI-3}$$

$$V_3 \frac{df_3}{dt} = a_{3,2}\left(f\,VA\right)_2 - a_{3,3}\left(f\,VA\right)_3 + V_3 S_3$$

or rearranged as:

$$V_1 \frac{df_1}{dt} + a_{1,1}\left(f\,VA\right)_1 + a_{1,2}\left(0\right) - a_{1,3}\left(f\,VA\right)_3 = V_1 S_1$$

$$V_2 \frac{df_2}{dt} - a_{2,1}\left(f\,VA\right)_1 + a_{2,2}\left(f\,VA\right)_2 + a_{1,3}\left(0\right) = V_2 S_2 \tag{AI-4}$$

$$V_3 \frac{df_3}{dt} + a_{3,1}\left(0\right) - a_{3,2}\left(f\,VA\right)_2 + a_{3,3}\left(f\,VA\right)_3 = V_3 S_3$$

The trems associated with $a_{N,k}$ are link (junction) terms, not to be confused with the volume 1, 2 and 3 indices. The $a_{N,k}$ indicate the position in the matrix of the flow terms. The matrix representation can be written as:

$$\begin{bmatrix} V_1 & 0 & 0 \\ 0 & V_2 & 0 \\ 0 & 0 & V_3 \end{bmatrix} \begin{bmatrix} df_1/dt \\ df_2/dt \\ df_3/dt \end{bmatrix} + \begin{bmatrix} a_{11} & 0 & -a_{13} \\ -a_{21} & a_{22} & 0 \\ 0 & -a_{32} & a_{33} \end{bmatrix} \begin{bmatrix} \left(f\,VA\right)_1 \\ \left(f\,VA\right)_2 \\ \left(f\,VA\right)_3 \end{bmatrix} = \begin{bmatrix} \left(VS\right)_1 \\ \left(VS\right)_2 \\ \left(VS\right)_3 \end{bmatrix} \tag{AI-5}$$
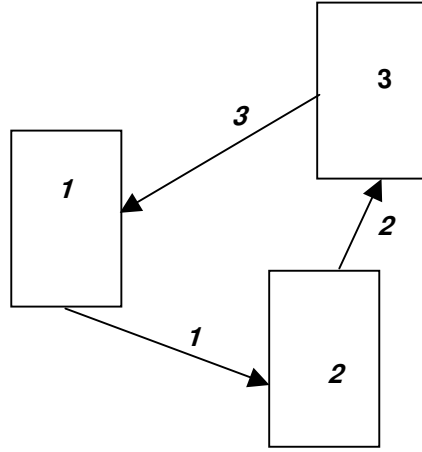
*Figure AI.1*

As stated previously the $(fVA)_k$ terms are link (junction) quantities. Recall that $f$ in $(fVA)_k$ is not known at the junctions, it is a volume, cell or nodal quantity. Thus, we must "donor" $f$ from the volumes to the links (junctions).

This can be done using the relation

$$
(fVA)_t = A_t \frac{V_t}{2}\left[(1+\beta_t) \quad (1-\beta_t)\right]\begin{bmatrix} f_t^{\,u} \\ f_t^{\,N} \end{bmatrix} = \vec{\beta}^T \vec{f}_t
$$

$$
(fVA)_i = A_i \frac{V_i}{2}\left[(1+\beta_i) \quad (1-\beta_i)\right]\begin{bmatrix} f_i^{\,N} \\ f_i^{\,d} \end{bmatrix} = \vec{\beta}^T \vec{f}_i
$$

(61a,b)

for the terminal and incident links. The $\beta_k$ are defined as $\beta_k = V_k / (|V_k| + \varepsilon)$, which takes into account the flow direction. For the link term $(fVA)_1$ at link one in Figure AI.1 which is terminal to node 2, (61a) becomes

$$
(fVA)_1 = A_1 \frac{V_1}{2}\left[(1+\beta_1) \quad (1-\beta_1)\right]\begin{bmatrix} f_1 \\ f_2 \end{bmatrix}
$$

(AI-6)

For positive flow from node 1 to node 2 in Figure AI.1,

$$
\beta_1 = V_1 / |V_1| = 1
$$

(AI-7)

and AI-6 becomes

$$\left(f\,VA\right)_1^* = A_1^* \frac{V_1^*}{2}\begin{bmatrix}(2) & 0\end{bmatrix}\begin{bmatrix}f_1 \\ f_2\end{bmatrix} = A_1^* V_1^* f_1 \qquad \text{(AI-8)}$$

The asterisk is used to denote a link (junction) quantity. If the flow direction reverses and goes from node 2 to node 1 then $V_1$ is negative and

$$\beta_1 = -V_1 / |V_1| = -1 \qquad \text{(AI-9)}$$

and AI-6 becomes

$$\left(f\,VA\right)_1^* = A_1^* \frac{V_1^*}{2}\begin{bmatrix}(0) & (2)\end{bmatrix}\begin{bmatrix}f_1 \\ f_2\end{bmatrix} = A_1^* V_1^* f_2 \qquad \text{(AI-10)}$$

with $f$ donored from node two.

We can write

$$\begin{bmatrix}\left(f\,VA\right)_1 \\ \left(f\,VA\right)_2 \\ \left(f\,VA\right)_3\end{bmatrix} \qquad \text{(AI-11)}$$

as

$$\begin{bmatrix}\left(f\,VA\right)_1 \\ \left(f\,VA\right)_2 \\ \left(f\,VA\right)_3\end{bmatrix} = \begin{bmatrix} A_1 \dfrac{V_1}{2}\begin{bmatrix}(1+\beta_1) & (1-\beta_1)\end{bmatrix}\begin{bmatrix}f_1 \\ f_2\end{bmatrix} \\ A_2 \dfrac{V_2}{2}\begin{bmatrix}(1+\beta_2) & (1-\beta_2)\end{bmatrix}\begin{bmatrix}f_2 \\ f_3\end{bmatrix} \\ A_3 \dfrac{V_3}{2}\begin{bmatrix}(1+\beta_3) & (1-\beta_3)\end{bmatrix}\begin{bmatrix}f_3 \\ f_1\end{bmatrix} \end{bmatrix}$$

$$= \begin{bmatrix} A_1 \dfrac{V_1}{2}(1+\beta_1) & A_1 \dfrac{V_1}{2}(1-\beta_1) & 0 \\ 0 & A_2 \dfrac{V_2}{2}(1+\beta_2) & A_2 \dfrac{V_2}{2}(1-\beta_2) \\ A_3 \dfrac{V_3}{2}(1-\beta_3) & 0 & A_3 \dfrac{V_3}{2}(1+\beta_3) \end{bmatrix}\begin{bmatrix}f_1 \\ f_2 \\ f_3\end{bmatrix} \qquad \text{(AI-12)}$$

This can be used with AI-5 as

$$\begin{bmatrix} V_1 & 0 & 0 \\ 0 & V_2 & 0 \\ 0 & 0 & V_3 \end{bmatrix} \begin{bmatrix} df_1/dt \\ df_2/dt \\ df_3/dt \end{bmatrix} +$$

$$\begin{bmatrix} 1 & 0 & -1 \\ -1 & 1 & 0 \\ 0 & -1 & 1 \end{bmatrix} \begin{bmatrix} A_1\dfrac{V_1}{2}(1+\beta_1) & A_1\dfrac{V_1}{2}(1-\beta_1) & 0 \\ 0 & A_2\dfrac{V_2}{2}(1+\beta_2) & A_2\dfrac{V_2}{2}(1-\beta_2) \\ A_3\dfrac{V_3}{2}(1-\beta_3) & 0 & A_3\dfrac{V_3}{2}(1+\beta_3) \end{bmatrix} \begin{bmatrix} f_1 \\ f_2 \\ f_3 \end{bmatrix} = \begin{bmatrix} (VS)_1 \\ (VS)_2 \\ (VS)_3 \end{bmatrix} \qquad \text{(AI-13)}$$

where we have replaced the $a_{N,k}$ with their values of 1.0. Performing the matrix multiplication on the second term gives

$$\begin{bmatrix} V_1 & 0 & 0 \\ 0 & V_2 & 0 \\ 0 & 0 & V_3 \end{bmatrix} \begin{bmatrix} df_1/dt \\ df_2/dt \\ df_3/dt \end{bmatrix} +$$

$$\text{(AI-14)}$$

$$\begin{bmatrix} A_1\dfrac{V_1}{2}(1+\beta_1)-A_3\dfrac{V_3}{2}(1-\beta_3) & A_1\dfrac{V_1}{2}(1-\beta_1) & -A_3\dfrac{V_3}{2}(1+\beta_3) \\ -A_1\dfrac{V_1}{2}(1+\beta_1) & -A_1\dfrac{V_1}{2}(1-\beta_1)+A_2\dfrac{V_2}{2}(1+\beta_2) & A_2\dfrac{V_2}{2}(1-\beta_2) \\ A_3\dfrac{V_3}{2}(1-\beta_3) & -A_2\dfrac{V_2}{2}(1+\beta_2) & -A_2\dfrac{V_2}{2}(1-\beta_2)+A_3\dfrac{V_3}{2}(1+\beta_3) \end{bmatrix} \begin{bmatrix} f_1 \\ f_2 \\ f_3 \end{bmatrix} = \begin{bmatrix} (VS)_1 \\ (VS)_2 \\ (VS)_3 \end{bmatrix}$$

If all the flows are positive as shown in Figure AI.1, then the various $\beta_k$ are 1.0. This results in AI-14 becoming

$$\begin{bmatrix} V_1 & 0 & 0 \\ 0 & V_2 & 0 \\ 0 & 0 & V_3 \end{bmatrix} \begin{bmatrix} df_1/dt \\ df_2/dt \\ df_3/dt \end{bmatrix} = \begin{bmatrix} -A_1V_1 & 0 & A_3V_3 \\ A_1V_1 & -A_2V_2 & 0 \\ 0 & A_2V_2 & -A_3V_3 \end{bmatrix} \begin{bmatrix} f_1 \\ f_2 \\ f_3 \end{bmatrix} + \begin{bmatrix} (VS)_1 \\ (VS)_2 \\ (VS)_3 \end{bmatrix} \qquad \text{(AI-15)}$$

which the student should be able to verify from examination of Figure AI.1.

# Appendix II  Elementary Examples and Code Programming

This section illustrates test cases for elementary verification and validation and goes through the code programming. The student who understands the theory and the programming that goes into the code will have a knowledge base that can be extended to more complex codes and code development. Equation numbers used in this Appendix reference equation numbers in the main body of text.
The time dependent momentum equation is

$$I_j \frac{dw_j}{dt} = g_c C\left[P_j^u \quad -P_j^d\right] - \rho g \Delta Z_j - \left(\frac{Kw^2}{2\rho A^2}\right) \tag{50}$$

and at steady state

$$\frac{dw_j}{dt} = 0$$

which yields

$$w^2 = \frac{2\rho A^2}{K}\left\{g_c C\left[P_j^u \quad -P_j^d\right] - \rho g \Delta Z_j\right\}.$$

So, if we use a two-node model as shown below and fix the pressures and enthalpies as boundary nodes, with the pressures at the same value, the steady state mass flow equation becomes
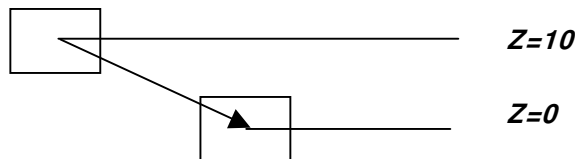


**Figure AII.1 Head Term Test Case**

$$w^2 = -\frac{2\rho A^2}{K}\rho g \Delta Z_j$$

Recall the definition of $\Delta Z_j$ as $\Delta Z_j = Z_j^d - Z_j^u$, so for our problem shown above, $\Delta Z_j$ will be negative. The input deck for this problem is show below

```
0
elevation test case
1 0
2 1
.25
2
1 1
2 1
1 1 2 1. 1. 1 0. 64
1 1 1 15 100   10 0
2 1 1 15 100   0 0
0
0
```

50

in file 2node.txt. Use the set values of pressure and enthalpy and show that the "analytical" and code solution are the same. Look in the code, identify the input and repeat the calculation for a height difference of 20.0 feet. Use the code called full-matrx-square.f.

The following 3 node example is a test case to walk through the 3-equation code and test it for a three- node model as shown below.



*Figure AII.2 Walk-through Test Case*

Three input decks have been prepared, they are:

1.  Sub-Cooled Liquid Test Case     (testsub.dat)
2.  Saturation Test Case (testsat.dat)
3.  Super-heated steam Case (testsup.dat)

**Input File Listings**

The following three decks are the input listings for the walk-through.

testsub.txt

```
0
3 node test case
1 0
3 2
.25
2
1 1
3 1
1 1 2 1. .1  1. 0. 64
2 2 3 1. .1  1. 0. 64
1 1. 1. 100  100    0 0
2 1. 1. 100  100    0 0
3 1. 1. 99  100    0 0
0
0
```

51

## // input explanation

iset, = 1, open an extra output file, = 0, don't open
title card
mprt,mprtflg, mprt=# of time steps per printout,mprtflg, chk for print
nvols,links nvols= # of volumes in problem, links= # of links
bn = number of boundary nodes
num,ipf(num) num= bn # , ipf(num)= 1, is a BN, ipf(num)=0, is not BN

Link Input Info

k,from(k),tooo(k),rlink(k),xk(k),xa(k),wlink(k),xi(k)
k=link number
from(k) = from or upstream node of link
tooo(k) = to or downstream node of link
rlink(k) = 1, fully open link, = 0, closed link
xk(k) = link form loss
wlink(k) = link mass flow rate
xi(k) = link inertia

Volume Input Info

n,avol(n),vol(n),pvol(n),hvol(n),zvol(n),qvol(n)
n=node number
avol(n)= area of volume
vol(n) = volume of volume
pvol(n) = pressure of volume
hvol(n) = enthalpy of volume
zvol(n) = height of volume center
qvol(n) = heat source in volume

The explanation of the input is listed at the bottom of the input deck. The next two input decks are
for testsat.dat and testsup.dat.

Testsat.dat

0
3 node test case
1 0
3 2
.25
2
1 1
3 1
1 1 2 1. .1  1. 0. 64
2 2 3 1. .1  1. 0. 64
1 1. 1. 100  460    0 0
2 1. 1. 100  460    0 0
3 1. 1. 99  460      0 0

Test sup.dat

```
0
3 node test case
1 0
3 2
.25
2
1 1
3 1
1 1 2 1. .1  1. 0. 64
2 2 3 1. .1  1. 0. 64
1 1. 1. 100  1300    0 0
2 1. 1. 100  1300    0 0
3 1. 1. 99  1300     0 0
```

As an example we will go through the sub-cooled liquid case first. The student should have access to a debugger utility so as to be able to step through the code.

Upon executing the code, the first group of Fortran statements we see is:

```
      open(unit=10,file='cv.out',status='old')
c
      jdone=.false.
c
      call cvi
      time = 0.
      do while(.not.jdone)
      write(*,*) ' '
      write(*,*) ' input # time steps , dt , mprt '
      read(*,*) nts,dt,mprt
      if (nts.gt.0) then
      do i=1,nts
      time = time + dt
      call cv
      end do
c
      else
        jdone=.true.
      endif
c
      end do
      stop
      end
```

The open statement opens the output file called cv.out. Notice this file is designated as "old" so it must be in the directory or an error will result.

Next, a logical variable jdone is set to false. This is done so that we can perform a ***do while loop*** as shown in the coding. As long as nts, the number of time steps is positive the code will keep going in the do while loop. If nts is set to zero or negative the code will quit executing.

The first subroutine called is the initialization routine, used to read in the input data.

The next group of Fortran statements that are encountered gets the name of the input file.

```
        character fname*16, answer*8
        logical fexist
c
c
   5  write(*,1000)
 1000 format ('Provide name of the input file:')
      read *, fname
      write(*,*)
      inquire (file=fname,exist=fexist)
      if (.not.fexist) then
        write(*,*) fname,' does not exist'
        write(*,*) 'Do you want to quit (yes or no)?'
        read (*,1001) answer
        if (answer(1:1).eq.'y'.or.answer(1:1).eq.'Y') stop
        go to 5
      endif
      open (9,file=fname,status='old')
 1001 format (a)
```

and should be self-explanatory.

The next parameter read is iset which controls whether a plot file is written out. Next we have some of the constants in the problem set up, which are:

c144 = 144 in$^2$/ft$^2$
cgc  = 32.2 ft/s s

which are used in the momentum equation

$$I_j \frac{dw_j}{dt} = g_C C\left[ P_j^u \quad -P_j^d \right] - \rho g \Delta Z_j - \left( \frac{Kw^2}{2\rho A^2} \right) \tag{50}$$

as $g_C, C$ and $g$.

We also set the maximum number of nodes and links as:

```
c
      maxnodes = 25
      maxlinks = 60
```

The next group of statements are used for reading some of the code boundary and initial conditions as:

54

```
c
      read(9,100) title
c
c     read(9,*) iset
c
c --  set the print flag & initial value for printing
c
      read(9,*) mprt,mprtflg
c
c
c --- nvols is the number of nodes in the problem
c --- links is the number of links (junctions, elements) in the problem
c
      read(9,*) nvols,links
c
      if(nvols.gt.maxnodes) write(*,*) 'exceeded maxnodes'
      if(links.gt.maxlinks) write(*,*) 'exceeded maxlinks'
c
c    dt is the time step
      read(9,*) dt
c
c  ipf(node) = 1 ,the node is a boundary node
c  ipf(node) = 0 ,the node is not a boundary node
c
c --  initialize all nodal arrays used to zero
c
      do 20 i = 1,maxnodes
      ipf(i) = 0
      pvol(i) = 0
      hvol(i) = 0
      alpvol(i) = 0
      tvol(i) = 0
      dvol(i) = 0
      dvolm(i) = 0
      xvol(i) = 0
      qvol(i) = 0
      avol(i) = 0
      zvol(i) = 0
      vol(i) = 0
      zdrodh(i) = 0
      zdrodp(i) = 0
      hvolo(i) = 0
      pvolo(i) = 0
      diag(i) = 0
      gcvsi(i) = 0
      glink(i) = 0
20    continue
c
c --  read # of boundary nodes, bn
```

```
c --  and read their values
c --  set the boundary nvols
c --   set to 1 for regular node b.c., for both pressure & enthalpy
c
      read(9,*) bn
      do 21 k = 1,bn
      read(9,*) num,ipf(num)
 21   continue
c
```

The explanation is included in the coding above.

The next set of executable statements counts the number of boundary links and nodes, useful when we go to a symmetric matrix solver.

```
c
c --- count the number of boundary nodes with the ipf flag
c
      nbndry = 0
c
      do i = 1,nvols
      if (ipf(i).eq.1) nbndry = nbndry + 1
      end do
c
c --- # of internal nodes , intn = nvols - nbndry
c --- # of external nodes , nsgsextn = nbndry
c
      intn = nvols - nbndry
      nsgsextn = nbndry
c
c
c
c
c --- the topology is next, which is the element connection information
c --- along with the loss coefficent xk, valve position rlink=rk,
c --- and the element (junction) initial flow wlink = w
c --- write an output heading
c
c
c
c --- initialize the half-bandwidth hbw, which will be calculated
c --- from the element topology
c
      hbw = 0
c
c --- element (link) connections are placed in one dimensional array from(i)
c --- tooo(i) where i is the element number , iu denotes
c --- the from
c --- node of the link and id denotes the to node of the link
c --- each link has a from and a to node as shown below
```

```
c
c    1         2
c    (2)---x---(3)
c         2
c
c --- for link 2 whose "from node" is 2 and whose "to node" is 3
c
c
```

Next we zero out or initialize all the link quantities and read in the link data.

```
c
c*********************************************************************
c******************** element topology ***************************
c************** vlv positions,losses,link areas,flows *****************
c*********************************************************************
c -- read in base admittance and multipliers
c -- rlink = multipliers or valve positions
c -- note : these are link quantites
c -- xa = junction area
c -- xk = junction losses
c -- xi = junction inertia
c
c -- zero out the link quantities
c
     do 22 i = 1,maxlinks
     from(i) = 0
     tooo(i) = 0
     rlink(i) = 0
     xk(i) = 0
     xa(i) = 0
     wlink(i) = 0
     xi(i) = 0
     ylink(i) = 0
     zdh(i) = 0
     rhoj(i) = 0
22   continue
c
     do 24 i = 1,links
     read(9,*) k,from(k),tooo(k),rlink(k),xk(k)
    &,xa(k),wlink(k),xi(k)
24   continue
c
```

which is described in the fortran. For our problem there are 2 links, 1 and 2. Link 1 is from node 1 to node 2 and link 2 is from node 2 to 3.

The next set of Fortran is included but only used when we need a symmetric pressure matrix solver. After this, we read the nodal or volume data.

```
c
```

```
c -- read in nodal volumes note that 1st letter v denotes volume
c -- last letter denotes volume
c -- read in the pressure & enthalpy
c
c -- read the vol area,volume,pressure,enthalpy,
c -- node center elevations, and heat source q in
c
     do 25 i = 1,nvols
     read(9,*) k,avol(k),vol(k),pvol(k),hvol(k)
    &,zvol(k),qvol(k)
25   continue
c
c -- calculate elev changes
c
     do 605 i = 1,links
     nu = from(i)
     nd = tooo(i)
     zdh(i) = zvol(nd) - zvol(nu)
605  continue
```

We next perform a check to make sure we do not have a zero volume case. Remember from our single phase 2-equation code we do not want a zero on the diagonal of our pressure matrix.

```
c
c -- check for zero volume
c
     do 606 i = 1,nvols
     if (vol(i).gt.0.) go to 606
     write(*,*) ' zero volume for node = ',i
     write(10,*) ' zero volume for node = ',i
 606 continue
c
c
```

We next perform the calculations for our initial thermodynamic properties by looping over all the volumes.

```
c
c -- calculate the quality from enthalpy,pressure
c
     do i = 1,nvols
     zcvhfg = hsvsp(pvol(i)) - hwvsp(pvol(i))
     xvol(i) = (hvol(i) - hwvsp(pvol(i)))/zcvhfg
     if (xvol(i).lt.0.) xvol(i) = 0.
     if (xvol(i).gt.1.) xvol(i) = 1.
     end do
```

The equation relevant to the above code is

$$X_e = \frac{h - h_l(p)}{h_g(p) - h_l(p)} \hspace{4cm} (97)$$

with hsvsp(pvol(i)) - hwvsp(pvol(i)) as the sat enthalpies for steam and liquid. Notice that we set limits on the equilibrium quality xvol. We next calculate the specific volumes of the mixture with three checks, one for sub-cooled, one for sat and one for superheated steam for the three thermodynamic regions.

```
c
c
c
c  -- specific volume of mixture calc
c
      do 562 i = 1,nvols
c
c --  do superheated steam first
c
      if (xvol(i).ge.1.0) go to 559
c
      if (xvol(i).gt.0.0) then
      znumix = xvol(i)*svsvsp(pvol(i)) + (1.-xvol(i))*
     &svwvsp(pvol(i))
      dvol(i) = 1./znumix
      dvolm(i) = dvol(i)
      tvol(i) = tvsp(pvol(i))
      else
```

**The above is for sat. Note that tvsp is the sat temperature as a function pressure.**

```
c
c --  sp. volume , invert
c
```

**This set is for sub-cooled properties since the test is for X .ge. 0.0. extvvwhp(hvol(i),pvol(i)) is the sub-cooled property fortran function. Note the temperature is calculated also. The same applies for the superheated steam.**

```
      dvol(i) = extvvwhp(hvol(i),pvol(i))
      dvol(i) = 1./dvol(i)
      dvolm(i) = dvol(i)
      tvol(i) = exttttwhp(pvol(i),hvol(i))
      endif
      go to 562
559   dvol(i) = extvvvhp(hvol(i),pvol(i))
      dvol(i) = 1./dvol(i)
      dvolm(i) = dvol(i)
      tvol(i) = exttttvhp(pvol(i),hvol(i))
562   continue
```
The next group of info is for output and is self-explanatory.

We then return to the main program and input the number of time steps, the time step and however many time steps that we want a printout with.

At last we call routine cv, the main routine for executing the code.

We then see a group of data statements in F77 and initialization of the old time calculated variables to the input variables. Also, the initial mixture density is set to the state density $\rho_s = \rho_s(P,h)$. The only way of knowing the mixture density is to set it to state at the beginning of the program. We then call the state routine to get the derivatives for the sub-cooled liquid state with our previous relations of

$$\frac{\partial \rho_s(P,h)}{\partial P} = \frac{\partial \frac{1}{v_s}}{\partial P} = -\frac{1}{v_s^2}\frac{\partial v_s}{\partial P}$$

(99)

$$\frac{\partial \rho_s(P,h)}{\partial h} = \frac{\partial \frac{1}{v_s}}{\partial h} = -\frac{1}{v_s^2}\frac{\partial v_s}{\partial h}$$

and the pressure (continuity) equation

$$\left(\mathcal{V}\frac{\partial \rho_s}{\partial p}\right)_N P_N^{n+1} + \left(\mathcal{V}\frac{\partial \rho_s}{\partial h}\right)_N \left(\frac{\mathcal{V}_N \frac{C}{J}P_N^{n+1}}{\sigma_N}\right)_i$$

$$\Delta t\left(\sum_t\left(a_{N,t}Y_t^n\left[P_t^U - P_t^N\right]\right) - \sum_i a_{N,i}Y_i^n\left[P_i^N - P_i^D\right]\right) =$$

(79)

$$\mathcal{V}_N(\rho_M - \rho_s)_N + \left(\mathcal{V}\frac{\partial \rho_s}{\partial p}\right)_N P_N^n + \left(\mathcal{V}\frac{\partial \rho_s}{\partial h}\right)_N\left(h_N^n - \frac{\tau_N}{\sigma_N} + \frac{\mathcal{V}_N \frac{C}{J}P_N^n}{\sigma_N}\right)$$

$$+\Delta t\sum_t a_{N,t}D_t - \Delta t\sum_i a_{N,i}D_i$$

which uses the calculated state derivatives in these two equations in subroutine **dprop**. The **d** stands for "derivative".

```
c
c -- initialize arrays to zero
c
      do 3 i = 1,nvols
      dvolo(i) = dvol(i)
      dvol(i) = dvolm(i)
      pvolo(i) = pvol(i)
      hvolo(i) = hvol(i)
```

```
3    continue
c
c
c -- get the derivatives and constituitive values at all nodes
c -- the fluid temperatures for heat transfer
c
    call dprops
c
c
```

The coding for dprop is listed below.

```
c
c -- get the derivatives at all nodes
c
```

*We first loop over all the volumes, nvols.*

```
    do 555 m=1,nvols
c
c -- get the density derivative per equation (82)
c
c -- zdrgdp   = -(1/nu)*(1/nu)*(dnu/dp)
c -- zdrgdp1 = derivative for node n1
c -- zdrgdp2 = derivative for node n2
c -- znug1   = specific volume for node n1
c -- znug2   = specific volume for node n2
c
```

*The above are the definitions*

```
c
c -- the derivative of specific volume is done numerically
c
c -- the following logic avoids the derivative property calls for boundary nodes
c -- since they are the off diagonal terms
c
c
c -- calculate the mixture specific volume, set pressures, quality
c -- enthalpy derivative first (zdrgdh1,2) , for n1,n2
c -- pressure second (zdrgdp1,2) , for n1,n2
c
    if (xvol(m).ge.1.0) go to 551
c
c   sat
c
    if (xvol(m).gt.0.0) then
c
c
c -- call the derivative function for two phase for drho/dp, drho/dh, rho is density
c
c -- the dv/dp uses the dv/dh derivative, which is why the
```

```
c -- two are done together
c
c -- dv/dh , derivative of specifc volume wrt enthalpy
c
      drhomdh = (svsvsp(pvol(m)) - svwvsp(pvol(m)))
      drhomdhh = drhomdh/(hsvsp(pvol(m)) - hwvsp(pvol(m)))
      drhomdh = -dvol(m)*dvol(m)*drhomdhh
c
c -- do the pressure derivative
c
c//////////////////////////////////////////////////////////////
c////////////////// dv/dp function ////////////////////////////
c//////////////////////////////////////////////////////////////
c
c
c -- dv/dp , derivative of mixture specific volume wrt pressure
c
c -- calculate dvgdp
c
      dcvdpd  = 1.
      zcvp1 = pvol(m) + dcvdpd
      dcvdp = xvol(m)*(svsvsp(zcvp1) - svsvsp(pvol(m)))/dcvdpd
      dcvdp = dcvdp + (1.-xvol(m))*
     &(svwvsp(zcvp1) - svwvsp(pvol(m)))/dcvdpd
c
c -- get dh/dp , derivative of mixture enthalpy wrt pressure
c
      dcvhdp = xvol(m)*(hsvsp(zcvp1) - hsvsp(pvol(m)))/dcvdpd
      dcvhdp = dcvhdp + (1. - xvol(m))*
     &(hwvsp(zcvp1) - hwvsp(pvol(m)))/dcvdpd
c
c -- get dvm/dp , partial derivative of spec volume wrt pressure
c -- remember that dv/dh is used here
c
      drhomdp = dcvdp - drhomdhh*dcvhdp
      drhomdp = -dvol(m)*dvol(m)*drhomdp
c
c
c
      zdrodh(m) = drhomdh
      zdrodp(m) = drhomdp
c
      else
c
c
c --  deriv of liq density wrt enthalpy
c
      hvalr = hvol(m) + 1.
      rholr = extvvwhp(hvalr,pvol(m))
      rholr = 1./rholr
```

```
      rholl = extvvwhp(hvol(m),pvol(m))
      rholl = 1./rholl
      drholdh = rholr - rholl
c
c
c --  deriv of liq density wrt pressure
c
      pvalr = pvol(m) + 1.
      rholr1 = extvvwhp(hvol(m),pvalr)
      rholr1 = 1. / rholr1
c     rholl = extvvwhp(h,p)
c     rholl = 1./rholl
      drholdp = rholr1 - rholl
c
      zdrodh(m) = drholdh
      zdrodp(m) = drholdp
      endif
c
c
      go to 555
c
 551  continue
c
c//////////////////////////////////////////////////////////////c
c///////////// superheated enthalpy derivative drhogdh /////////////////c
c//////////////////////////////////////////////////////////////c
c
c -- deriv of gas density wrt enthalpy
c
      hvalr = hvol(m) + 1.
      rhogr = extvvvhp(hvalr,pvol(m))
      rhogr = 1./rhogr
      rhogl = extvvvhp(hvol(m),pvol(m))
      rhogl = 1./rhogl
      drhogdh = rhogr - rhogl
c
c
c//////////////////////////////////////////////////////////////c
c///////////// superheated pressure derivative drhogdp /////////////////c
c//////////////////////////////////////////////////////////////c
c
c -- deriv of gas density wrt pressure
c
      pvalr = pvol(m) + 1.
      rhogr = extvvvhp(hvol(m),pvalr)
      rhogr = 1./rhogr
c     rhogl = extvvvhp(h,p)
c     rhogl = 1./rhogl
      drhogdp = rhogr - rhogl
c
```

```
      zdrodh(m) = drhogdh
      zdrodp(m) = drhogdp
c
c
c
 555  continue
c
      return
      end
c
```

The highlighted entries above are for the sub-cooled liquid state. Note that we are performing numerical derivatives from the state functions. Actually, we do not need the derivatives for a boundary volume. As we exit subroutine **dprop** we see the coding

```
c
c -- init arrays to zero
c
      do 14 i = 1,nvols
      sigma(i) = 0.
      brhs(i) = 0.
      taurhs(i) = 0.
      do 14 j = 1,nvols
      amat(i,j) = 0.
  14      continue
```

which initializes the arrays in in the energy equation and the matrix pressure equation.

$$\sigma_N h_N^{n+1} = \tau_N + V_N \frac{C}{J} \left( P^{n+1} - P^n \right)_N \tag{64}$$

and

$$\underline{\underline{a}} \, \vec{p} = \vec{b} \tag{I-1}$$

for the pressure matrix solution.

We next get the volume and link contributions to the energy equation from the relations:

$$h_N^{n+1}\left\{\left(\rho \not{V}\right)_N + \Delta t\left[\sum_t a_{N,t} \circ w_t - \sum_i a_{N,i} \circ w_i\right] - \Delta t \sum_t \frac{a_{N,t} w_t}{2}\left(1-\beta_t\right)\right.$$

$$\left. + \Delta t \sum_i \frac{a_{N,i} w_i}{2}\left(1+\beta_i\right)\right\} = \mathbf{\Delta t\, \not{V} q''}\frac{\mathbf{P_H}}{\mathbf{A}}$$

$$+ \mathbf{\Delta t \left(\not{V}q'''\right)_N} + \not{V}_N \frac{C}{J}\left(P^{n+1}-P^n\right)_N \qquad (63)$$

$$+ \left(\mathbf{\rho \not{V} h^n}\right)_N + \Delta t \sum_t \frac{a_{N,t} w_t}{2}\left(1-\beta_t\right)h_t^u$$

$$- \Delta t \sum_i \frac{a_{N,i} w_i}{2}\left(1+\beta_i\right)h_i^d$$

with the highlighted terms the ones in the coding. One of the terms has been left out of the coding, you will place back in there. There are also some additional link terms that contribute to the diagonal. What are they? Are they important for steady state? Why or why not?

```
c
c -- get the volume contribution energy elimination terms
c
      do 16 i = 1,nvols
      sigma(i) = vol(i)*dvol(i)
c
      taurhs(i) = dt*qvol(i)
    & + vol(i)*dvol(i)*hvol(i)
c
16    continue
c
      do 17 i = 1,links
c
      nd = tooo(i)
      nu = from(i)
c
c -- get beta
c
      if (wlink(i).eq.0.) then
      betaj(i) = 0.
      else
      betaj(i) = wlink(i)/(abs(wlink(i)))
      endif
c
      beta = betaj(i)
c
      sigma(nd) = sigma(nd) + dt*
    & (wlink(i) - wlink(i)*.5*(1-beta))
      sigma(nu) = sigma(nu) - dt*
    & (wlink(i) - wlink(i)*.5*(1+beta))
      taurhs(nu) = taurhs(nu) - dt*.5*wlink(i)*
```

```
   &(1-beta)*hvol(nd)
    taurhs(nd) = taurhs(nd) + dt*.5*wlink(i)*
   &(1+beta)*hvol(nu)
17    continue
```

The above coding has built up the energy equation into the form of

$$\sigma_N h_N^{n+1} = \tau_N + V_N \frac{C}{J}\left(P^{n+1} - P^n\right)_N \tag{64}$$

The term

$$V_N \frac{C}{J}\left(P^{n+1} - P^n\right)_N$$

will be added in the pressure equation.

The next piece of coding adds in components from the pressure equation, first the diagonal matrix contributions are performed.

```
c
c -- form the global matrix entries
c -- volume contributions and then links
c -- diagonal contributions
c
    do 19 i = 1,nvols
    amat(i,i) = vol(i)*zdrodp(i) + c144*vol(i)
   &*vol(i)*zdrodh(i)/(zcvj*sigma(i))
    brhs(i) = vol(i)*zdrodp(i)*pvol(i) + vol(i)
   &*zdrodh(i)*hvol(i) - vol(i)*zdrodh(i)*taurhs(i)/sigma(i)
   & + eps*vol(i)*(dvolm(i) - dvolo(i))
   & + c144*vol(i)*vol(i)*zdrodh(i)*pvol(i)/
   & (zcvj*sigma(i))
19    continue
c
```

performed according to to the highlighted terms below. Eps is usually set to 1.0 for the mass error. If the code gets larger pressures eps should be scaled back to 0.1.

$$\left(\mathbf{V}\frac{\partial \rho_s}{\partial P}\right)_N P_N^{n+1} + \left(\mathbf{V}\frac{\partial \rho_s}{\partial h}\right)_N \left(\frac{\mathbf{V_N}\dfrac{\mathbf{C}}{\mathbf{J}}P_N^{n+1}}{\sigma_N}\right) -$$

$$\Delta t\left(\sum_t \left(a_{N,t}Y_t^n\left[P_t^U - P_t^N\right]\right) - \sum_i a_{N,i}Y_i^n\left[P_i^N - P_i^D\right]\right) = \qquad (79)$$

$$\mathbf{V_N}\left(\rho_M - \rho_s\right)_N + \left(\mathbf{V}\frac{\partial \rho_s}{\partial p}\right)_N P_N^n + \left(\mathbf{V}\frac{\partial \rho_s}{\partial h}\right)_N \left(h_N^n - \frac{\tau_N}{\sigma_N} + \frac{\mathbf{V_N}\dfrac{\mathbf{C}}{\mathbf{J}}\mathbf{P_N^n}}{\sigma_N}\right)$$

$$+ \Delta t\sum_t a_{N,t}D_t - \Delta t\sum_i a_{N,i}D_i$$

The next piece of coding performs the link sums.

```
c
c --  add in link contributions
c
     do 20 i = 1,links
c
c --  link density
c
c
     nd = tooo(i)
     nu = from(i)
c
     rhoj(i) = .5*(dvol(nu) + dvol(nd))
    & + .5*betaj(i)*(dvol(nu) - dvol(nd))
c
c --  get the y factor from the momentum equation
c
     wabs = abs(wlink(i))
c
c --  limit r from going to zero
c
     if (rlink(i).le.tiny) then
     ylink(i) = 0.
     else
     z = (xi(i) + .5*dt*xk(i)*wabs/
    &(rhoj(i)*rlink(i)*rlink(i)*xa(i)*xa(i)))
         z = 1.0/z
     ylink(i) = z*dt*cgc*c144

c
     endif
c
     d(i) = z*xi(i)*wlink(i)
    & - dt*z*rhoj(i)*cgc*zdh(i)
c
```

c
c
```
      amat (nd,nu) = amat (nd,nu) - dt*ylink(i)
      amat (nu,nd) = amat (nu,nd) - dt*ylink(i)
      amat (nd,nd) = amat (nd,nd) + dt*ylink(i)
      amat (nu,nu) = amat (nu,nu) + dt*ylink(i)
c
      brhs(nd) = brhs(nd) + dt*d(i)
      brhs(nu) = brhs(nu) - dt*d(i)
c
20    continue
c
30    continue
```

We sum over the links, get the upstream (from) and downstream (to) node numbers connected by the link. We then perform the donoring for the link density and calculate the valve (link) aperture position. Next we calculate the terms in

$$w_j^{n+1} = Y_j^n \left[ P_j^U - P_j^D \right] + D_j \tag{50b}$$

compared to the full momentum equation of

$$w_j^{n+1} = \frac{\Delta t \, g_c C \left[ P_j^U - P_j^D \right] + I_j w_j^n - g \Delta t \Delta Z_j}{\left( I_j + \Delta t \dfrac{K w_j^{n+1} \left| w_j^n \right|}{2 \rho_j^n R_j^2 A_j^2} \right)} \tag{50b}$$

or

$$I_j \frac{dw_j}{dt} = g_c C \left[ P_j^U - P_j^D \right] - g \Delta Z_j - \left( \frac{K w |w|}{2 \rho A^2} \right)_j \tag{50}$$

We can see from the above equations what $Y_j^n \, and \, D_j$ are and we can see from

$$\left( \mathbf{V} \frac{\partial \rho_s}{\partial P} \right)_{\mathbf{N}} P_N^{n+1} + \left( \mathbf{V} \frac{\partial \rho_s}{\partial h} \right)_N \left( \frac{\mathbf{V_N} \dfrac{\mathbf{C}}{\mathbf{J}} P_N^{n+1}}{\mathbf{\sigma_N}} \right) -$$

$$\Delta t \left( \sum_t \left( Y_t^n \left[ P_t^U - P_t^N \right] \right) - \sum_i Y_i^n \left[ P_i^N - P_i^D \right] \right) =$$

$$\mathbf{V_N} \left( \mathbf{\rho_M} - \mathbf{\rho_s} \right)_{\mathbf{N}} + \left( \mathbf{V} \frac{\partial \rho_s}{\partial p} \right)_N P_N^n + \left( \mathbf{V} \frac{\partial \rho_s}{\partial h} \right)_N \left( \mathbf{h_N^n} - \frac{\mathbf{\tau_N}}{\mathbf{\sigma_N}} + \frac{\mathbf{V_N} \dfrac{\mathbf{C}}{\mathbf{J}} \mathbf{P_N^n}}{\mathbf{\sigma_N}} \right)$$

$$+ \Delta t \sum_t D_t - \Delta t \sum_i D_i$$

what the incident and terminal link sums are in the coding.

The next piece of coding sets the boundary conditions:

```
c
c -- set boundary conditions
c
    do 34 i = 1,nvols
    if (ipf(i).eq.1) then
    brhs(i) = amat(i,i)*big*pvol(i)
    amat(i,i) = amat(i,i)*big
    else
    continue
    endif
34    continue
```

as discussed previously in section 3.5 as:

A boundary node can be initialized in the program by the use of an array, such as

**IPF(I) =1**, if node **I** is a boundary node

**IPF(I) =0**, if not

The simplest way to approach this is the method of Payne & Irons by writing the matrix relation as

$$
\begin{pmatrix} a_{11} & \cdots & a_{1,NV} \\ \vdots & \ddots & \vdots \\ a_{NV,1} & \cdots & a_{NV,NV} \end{pmatrix} \begin{bmatrix} P_1 \\ P_2 \\ P_{NV} \end{bmatrix} = \begin{bmatrix} B_1 \\ B_2 \\ B_{NV} \end{bmatrix}
\tag{117}
$$

and to replace the diagonal terms corresponding to a boundary condition by

**B(I)=BIG x P(I) x A(I,I)** (118)

**A(I,I) = BIG x A(I,I)** (119)

where **BIG** is a large number, **BIG=1.e+20** and **IPF(I)** corresponds to a boundary node number. This can be programmed as

```
c
    do i=1,NV
    if (IPF(I).eq.1) then
    b(i)=BIG*p(i)*a(i,i)
    a(i,i)=BIG*a(i,i)
    else
    continue
    endif
    enddo
```

It is important to note that we do **b(i)=BIG\*p(i)\*a(i,i)** before **a(i,i)=BIG\*a(i,i)**, otherwise we will get a code error.

A square Gauss Elimination routine is be used from Brebbia. As we perform the Gauss Elimination the effect of the BC is to divide by a very large number to set intervening terms to zero except for the boundary row. Other methods can be used such as **A(I,I)=1.0** for boundary row **I** and **A(I,J)=A(J,I)=0** with **B(I)=P(I).** Other forms of matrix storage can be used such as bandwidth storage.

We then call the gaussian elimination routine and recover the solution:

```
c
c --- call gaussian elimination routine
c
      call gauss
c
c -- recover the solution from the rhs of the solver
c -- sum over all internal nodes
c -- if the node is a boundary node , the value will not change
c -- if not, use the solver (solution) value
c
      do 35 i = 1,nvols
      pvol(i) = brhs(i)
c
c -- limit the pressure to avoid steam table problems
c
      if (pvol(i).lt.zcvcp1) pvol(i) = zcvcp1
35    continue
c
```

The gaussian routine places the solution in brhs vector. We also set the pressure above a lower limit. Why?

After we obtain the pressure solution we can find the mass flows from the relation

$$w_j^{n+1} = Y_j^n \left[ P_j^U - P_j^D \right] + D_j \qquad\qquad (50b)$$

```
c
c///////////////////////////////////////////////////////////c
c////////////////////// section for flow solution //////////////////////c
c///////////////////////////////////////////////////////////c
c
c -- solve for the flow , eq (81)
c
      do 37 i = 1,links
c
c
      wlink(i) = ylink(i)*(pvol(from(i))-
```

```
      &pvol(t ooo(i))) + d(i)
c
c
37    continue
c
```

Since we have the new time-step pressures, the new time flows we can calculate the updated enthalpy from:

$$h_N^{n+1}\{(\boldsymbol{\rho \forall})_{\mathbf{N}} + \Delta t\left[\sum_t a_{Nt} \circ w_t - \sum_i a_{Ni} \circ w_i\right] - \Delta t \sum_t \frac{a_{Nt}w_t}{2}(1-\beta_t)$$

$$+\Delta t \sum_i \frac{a_{Ni}w_i}{2}(1+\beta_i)\} = \Delta \mathbf{t} \, \boldsymbol{\forall} \, \mathbf{q}'' \frac{\mathbf{P_H}}{\mathbf{A}}$$

$$+\Delta \mathbf{t}\,(\boldsymbol{\forall}\mathbf{q}''')_{\mathbf{N}} + \boldsymbol{\forall}_N \frac{C}{J}\left(P^{n+1}-P^n\right)_N \tag{63}$$

$$+\left(\boldsymbol{\rho \forall}\mathbf{h^n}\right)_{\mathbf{N}} + \Delta t \sum_t \frac{a_{Nt}w_t}{2}(1-\beta_t)h_t^u$$

$$-\Delta t \sum_i \frac{a_{Ni}w_i}{2}(1+\beta_i)h_i^d$$

and

$$\sigma_N h_N^{n+1} = \tau_N + \boldsymbol{\forall}_N \frac{C}{J}\left(P^{n+1}-P^n\right)_N \tag{64}$$

as

```
c
c//////////////////////////////////////////////////////////////////c
c////////////////////  enthalpy solution  /////////////////////////c
c//////////////////////////////////////////////////////////////////c
c
c
c -- solve for the enthalpy
c
c
c
c -- initialize the global matrix & rhs vectors
c -- rows ->nodes, columns ->bandwidth
c -- note that this is diagonal matrix assembly
c
c
c
c*************************************************************
      do 39 i = 1,nvols
c
c -- rhs term
```

```
c
      brhs(i) = vol(i)*dvol(i)*hvol(i)
     & + dt*qvol(i)
     & + c144*vol(i)*(pvol(i)-pvolo(i))/zcvj
c
c --  diagonal term
c
      diag(i) = vol(i)*dvol(i)
c
39    continue
c*****************************************************************
c     link (junction) enthalpies
c
      do 41 i = 1,links
c
c --  get the upstream & downstream node numbers
c
      nu = from(i)
      nd = tooo(i)
c
c --  donor junction density
c
      if (wlink(i).ne.0.) go to 43
      beta = 0.
      go to 45
43    beta = wlink(i)/(abs(wlink(i)))
45    continue
c
c --  compute the ret of the terms
c --  note: diag is a vector not a matrix
c
      diag(nd) = diag(nd) + dt*wlink(i)*(1. - .5*(
     &1.- beta))
      diag(nu) = diag(nu) - dt*wlink(i)*(1. - .5*(
     &1.+ beta))
      brhs(nd) = brhs(nd) + dt*wlink(i)*.5*hvol(nu)
     &*(1.+beta)
      brhs(nu) = brhs(nu) - dt*wlink(i)*.5*hvol(nd)
     &*(1.-beta)
41    continue
C*****************************************************************
c
c     compute qualities
c
      do 49 i = 1,nvols
          if (ipf(i).eq.0) then
      hvol(i) = brhs(i)/diag(i)
          endif
      zcvhfg  = hsvsp(pvol(i)) - hwvsp(pvol(i))
      xvol(i) = (hvol(i) - hwvsp(pvol(i)))/zcvhfg
```

```
      if (xvol(i).lt.0.) xvol(i) = 0.
      if (xvol(i).gt.1.) xvol(i) = 1.
c
49    continue
c
```

Note that we solve for the equilibrium qualities for each node as part of the solution process.

We also solve for the state densities, void fractions and temperatures from the coding

```
c
c -- get the nodal densities
c
      do 51 i = 1,nvols
c
c -- get the densities from the pressures, state equation, section 5.0
c
      if (xvol(i).ge.1.0) go to 53
c
c     sat
c
      if (xvol(i).gt.0.0) then
      vnug = svsvsp(pvol(i))
      vnuf = svwvsp(pvol(i))
      zcvnum = xvol(i)*vnug +
     & (1.-xvol(i))*vnuf
c
      dvol(i) = 1./zcvnum
      alpvol(i) = dvol(i)*xvol(i)*vnug
      tvol(i) = tvsp(pvol(i))
      else
c     subcooled
      dvol(i) = extvvwhp(hvol(i),pvol(i))
      dvol(i) = 1./dvol(i)
      alpvol(i) = 0.0
      tvol(i) = exttttwhp(pvol(i),hvol(i))
      endif
c
      go to 51
c     superheated
 53   dvol(i) = extvvvhp(hvol(i),pvol(i))
      dvol(i) = 1./dvol(i)
      alpvol(i) = 1.0
      tvol(i) = exttttvhp(pvol(i),hvol(i))
c
51    continue
c
c
```

which parallel the state relations given earlier of

$$X = \frac{M_g}{M} = \frac{\rho_g V_g}{\rho V} = \frac{\alpha \rho_g}{\rho} \qquad (91)$$

The densities are defined for three regions:

$$
\begin{aligned}
&X_e \leq 0.0 && \textit{for subcooled liquid} \\
&0.0 < X_e < 1.0 && \textit{for stauration mixture} \\
&X_e \geq 1.0 && \textit{for superheated steam}
\end{aligned}
\qquad (100)
$$

For the two-phase region:

$$v = X_e v_g (P) + (1 - X_e) v_l (P) \qquad (101)$$

Finally, the mixture density equation is solved as:

$$V_N \frac{d\rho_N}{dt} = \sum_t a_{Nt} \circ w_t - \sum_i a_{Ni} \circ w_i \qquad (56b)$$

```
c
c///////////////////////////////////////////////////////////////////////
c//////////////////// mixture mass solution ////////////////////////////
c///////////////////////////////////////////////////////////////////////
c
c
c -- get the diagonal volume contributions
c -- for the lhs/rhs of the mixture density equation
c -- note that this is diagonal matrix assembly
c
c
      do 55 i = 1,nvols
      brhs(i) = dvol(i)*vol(i)
      diag(i) = vol(i)
 55   continue
c
c
c
      do 71 i = 1,links
c
c -- get the upstream & downstream node numbers
c
      nu = from(i)
      nd = tooo(i)
c
c -- if the upstream or downstream node is a boundary node,
```

74

```
c --  don't add in any flow contribution
c --  doesn't matter since we only use internal nodes
c --  for mixture density in the next step
c
      brhs(nu) = brhs(nu) - dt*wlink(i)
      brhs(nd) = brhs(nd) + dt*wlink(i)
c
71    continue
c
c --  get the density solution, set boundary nodes to state density
c
      do 73 i = 1,nvols
      if (ipf(i).eq.0) then
      dvolm(i) = brhs(i)/diag(i)
      else
      dvolm(i) = dvol(i)
      endif
c
      if (dvolm(i).lt.0.) dvolm(i) = dvol(i)
c
73    continue
c
```

We then call the output routine and repeat the process.