Transient Response Analysis for Fault Detection and Pipeline Wall Condition Assessment in Field Water Transmission and Distribution Pipelines and Networks

by

Mark Leslie Stephens

February 2008

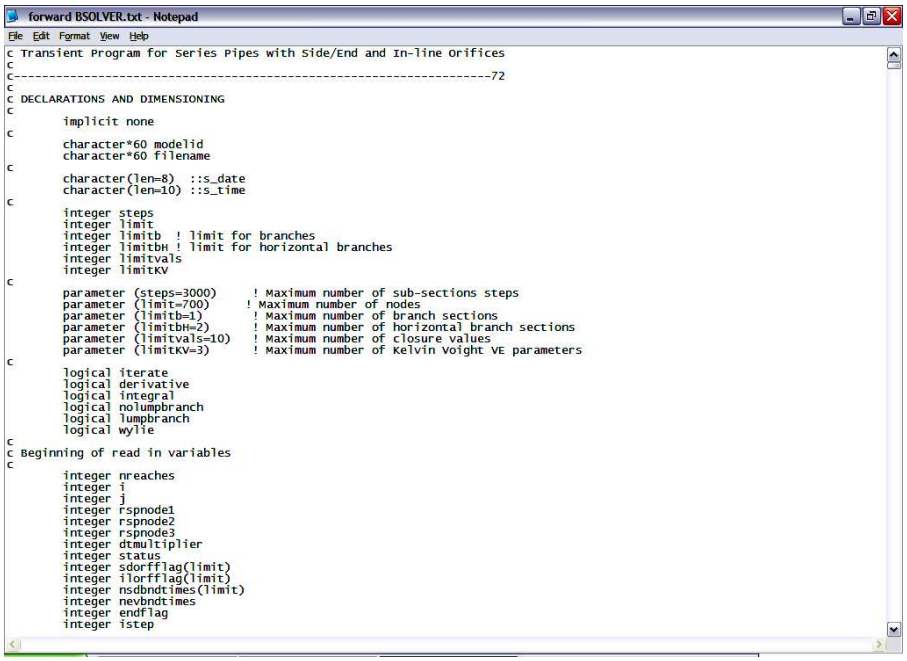
A Thesis Submitted for the Degree of Doctor of Philosophy

School of Civil and Environmental Engineering
The University of Adelaide, SA 5005
South Australia

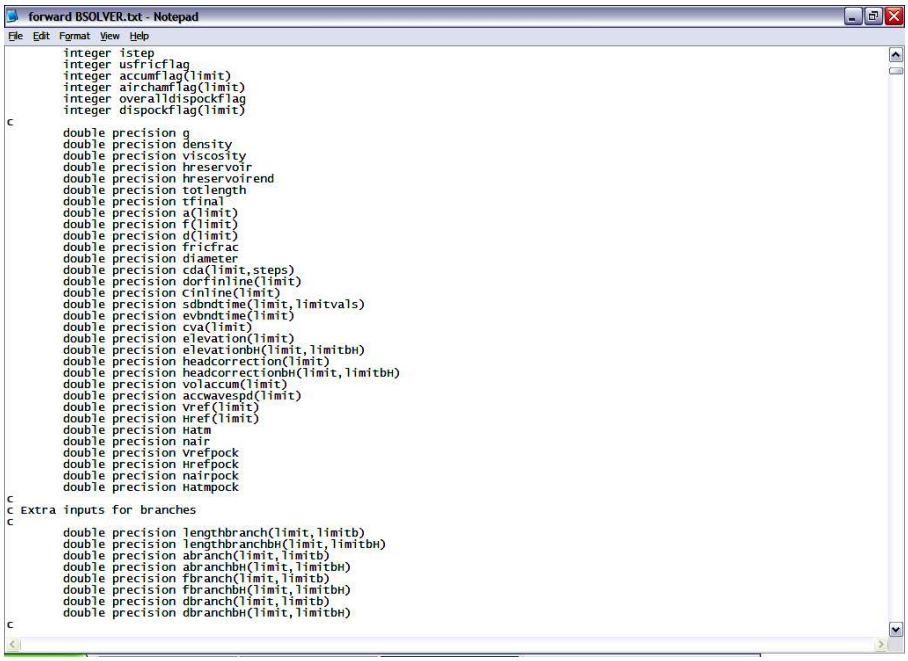
Appendix M

Fortran code for BSOLVER and NLFIT subroutines

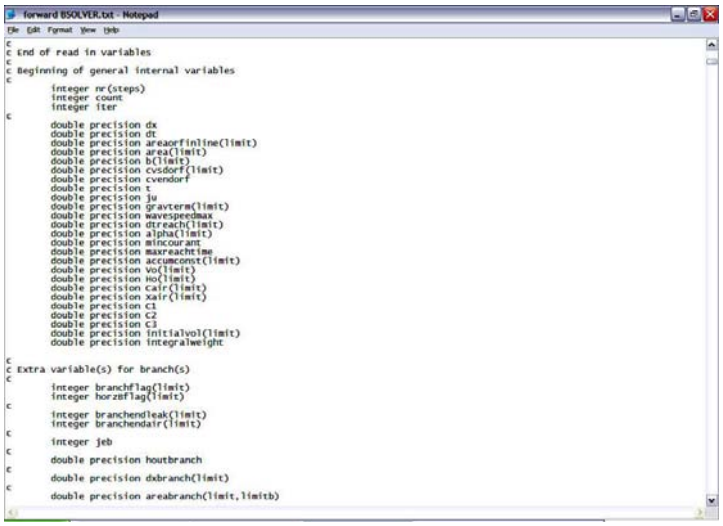
M.1 Forward transient program BSOLVER



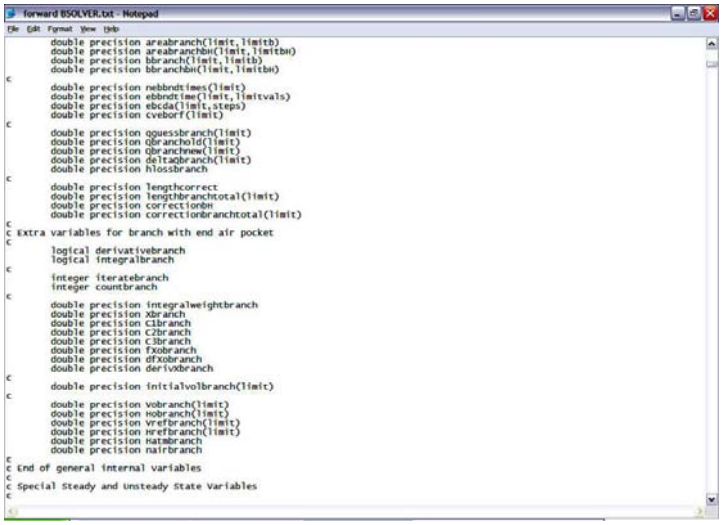

```
c Transient Program for Series Pipes with Side/End and In-line Orifices
c
c-----------------------------------------------------------------------72
c
c DECLARATIONS AND DIMENSIONING
c
      implicit none
c
      character*60 modelid
      character*60 filename
c
      character(len=8)  ::s_date
      character(len=10) ::s_time
c
      integer steps
      integer limit
      integer limitb  ! limit for branches
      integer limitbH ! limit for horizontal branches
      integer limitvals
      integer limitKV
c
      parameter (steps=3000)     ! Maximum number of sub-sections steps
      parameter (limit=700)     ! Maximum number of nodes
      parameter (limitb=1)       ! Maximum number of branch sections
      parameter (limitbH=2)      ! Maximum number of horizontal branch sections
      parameter (limitvals=10)   ! Maximum number of closure values
      parameter (limitKV=3)      ! Maximum number of Kelvin Voight VE parameters
c
      logical iterate
      logical derivative
      logical integral
      logical nolumpbranch
      logical lumpbranch
      logical wylie
c
c Beginning of read in variables
c
      integer nreaches
      integer i
      integer j
      integer rspnode1
      integer rspnode2
      integer rspnode3
      integer dtmultiplier
      integer status
      integer sdorfflag(limit)
      integer ilorfflag(limit)
      integer nsdbndtimes(limit)
      integer nevbndtimes
      integer endflag
      integer istep
```



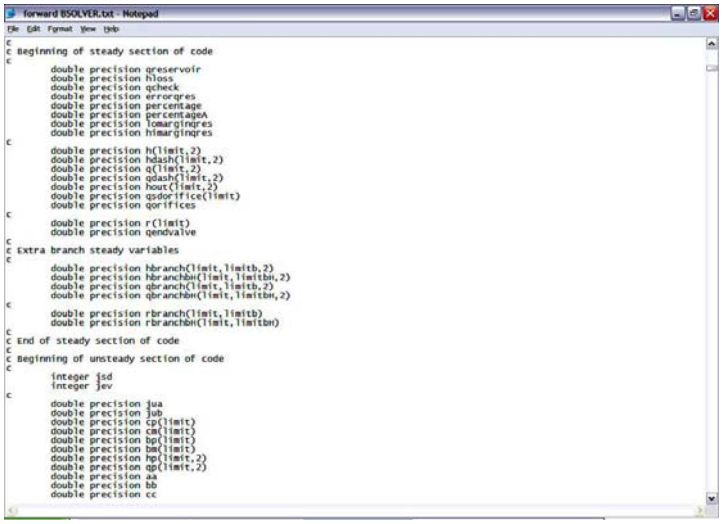

```
      integer istep
      integer usfricflag
      integer accumflag(limit)
      integer airchamflag(limit)
      integer overalldispockflag
      integer dispockflag(limit)
c
      double precision g
      double precision density
      double precision viscosity
      double precision hreservoir
      double precision hreservoirend
      double precision totlength
      double precision tfinal
      double precision a(limit)
      double precision f(limit)
      double precision d(limit)
      double precision fricfrac
      double precision diameter
      double precision cda(limit,steps)
      double precision dorfinline(limit)
      double precision Cinline(limit)
      double precision sdbndtime(limit,limitvals)
      double precision evbndtime(limit)
      double precision cva(limit)
      double precision elevation(limit)
      double precision elevationbH(limit,limitbH)
      double precision headcorrection(limit)
      double precision headcorrectionbH(limit,limitbH)
      double precision volaccum(limit)
      double precision accwavespd(limit)
      double precision Vref(limit)
      double precision Href(limit)
      double precision Hatm
      double precision nair
      double precision Vrefpock
      double precision Hrefpock
      double precision nairpock
      double precision Hatmpock
c
c Extra inputs for branches
c
      double precision lengthbranch(limit,limitb)
      double precision lengthbranchbH(limit,limitbH)
      double precision abranch(limit,limitb)
      double precision abranchbH(limit,limitbH)
      double precision fbranch(limit,limitb)
      double precision fbranchbH(limit,limitbH)
      double precision dbranch(limit,limitb)
      double precision dbranchbH(limit,limitbH)
c
```


forward BSOLVER.txt - Notepad

```
c
c End of read in variables
c
c Beginning of general internal variables
c
      integer nr(steps)
      integer count
      integer iter
c
      double precision dx
      double precision dt
      double precision areaorfinline(limit)
      double precision area(limit)
      double precision b(limit)
      double precision cvsdorf(limit)
      double precision cvendorf
      double precision t
      double precision ju
      double precision gravterm(limit)
      double precision wavespeedmax
      double precision dtreach(limit)
      double precision alpha(limit)
      double precision mincourant
      double precision maxreachtime
      double precision accumconst(limit)
      double precision vo(limit)
      double precision ho(limit)
      double precision cair(limit)
      double precision xair(limit)
      double precision c1
      double precision c2
      double precision c3
      double precision initialvol(limit)
      double precision integralweight

c
c Extra variable(s) for branch(s)
c
      integer branchflag(limit)
      integer horzbflag(limit)
c
      integer branchendleak(limit)
      integer branchendair(limit)
c
      integer jeb
c
      double precision houtbranch
c
      double precision dxbranch(limit)
c
      double precision areabranch(limit,limitb)
```

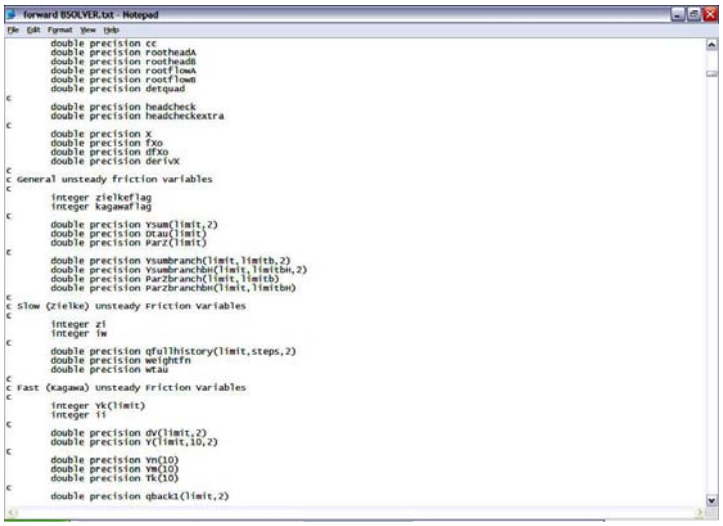

forward BSOLVER.txt - Notepad

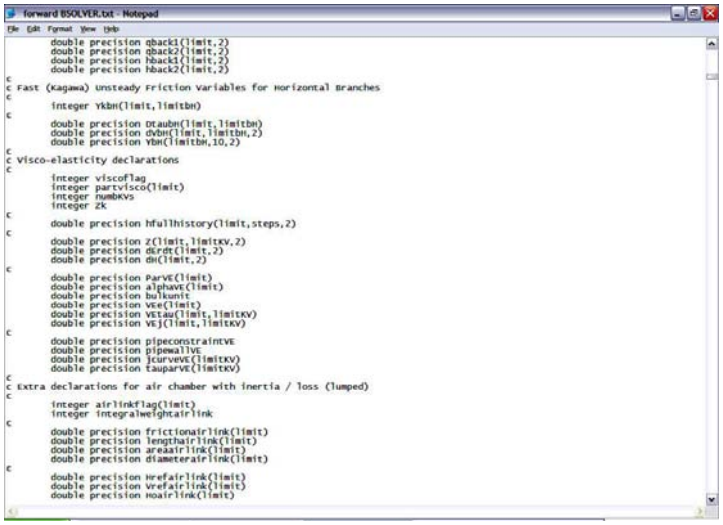
```
      double precision areabranch(limit,limitb)
      double precision areabranchbn(limit,limitbn)
      double precision bbranch(limit,limitb)
      double precision bbranchbn(limit,limitbn)
c
      double precision nebbrdtimes(limit)
      double precision ebbrdtime(limit,limitvals)
      double precision ebcda(limit,steps)
      double precision cveborf(limit)
c
      double precision qguessbranch(limit)
      double precision qbranchold(limit)
      double precision qbranchnew(limit)
      double precision deltaqbranch(limit)
      double precision hlossbranch
c
      double precision lengthcorrect
      double precision lengthbranchtotal(limit)
      double precision correctionbn
      double precision correctionbranchtotal(limit)
c
c Extra variables for branch with end air pocket
c
      logical derivativebranch
      logical integralbranch
c
      integer iteratebranch
      integer countbranch
c
      double precision integralweightbranch
      double precision xbbranch
      double precision c1branch
      double precision c2branch
      double precision c3branch
      double precision fxobranch
      double precision dfxobranch
      double precision derivxbranch
c
      double precision initialvolbranch(limit)
c
      double precision vobranch(limit)
      double precision hobranch(limit)
      double precision vrefbranch(limit)
      double precision hrefbranch(limit)
      double precision hatmbranch
      double precision nairbranch
c
c End of general internal variables
c
c Special Steady and Unsteady State Variables
c
```

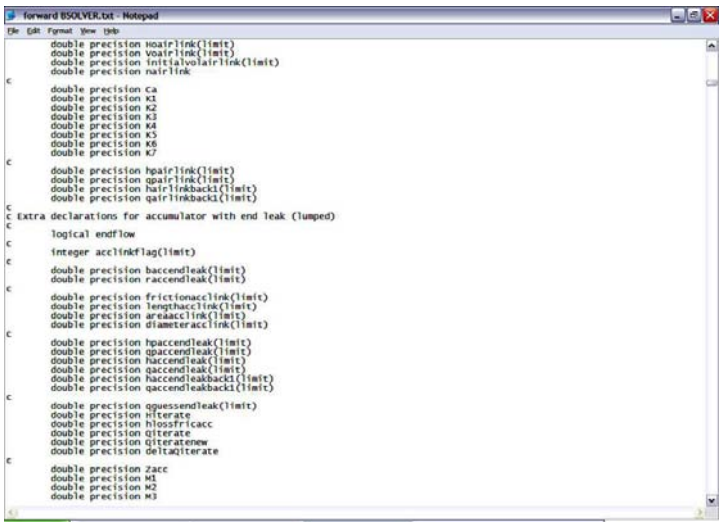

forward BSOLVER.txt - Notepad

```
c
c Beginning of steady section of code
c
      double precision qreservoir
      double precision hloss
      double precision qcheck
      double precision errorqres
      double precision percentage
      double precision percentageA
      double precision lomarginqres
      double precision himarginqres
c
      double precision h(limit,2)
      double precision hdash(limit,2)
      double precision q(limit,2)
      double precision qdash(limit,2)
      double precision hout(limit,2)
      double precision qsdorifice(limit)
      double precision qorifices
c
      double precision r(limit)
      double precision qendvalve
c
c Extra branch steady variables
c
      double precision hbranch(limit,limitb,2)
      double precision hbranchbn(limit,limitbn,2)
      double precision qbranch(limit,limitb,2)
      double precision qbranchbn(limit,limitbn,2)
c
      double precision rbranch(limit,limitb)
      double precision rbranchbn(limit,limitbn)
c
c End of steady section of code
c
c Beginning of unsteady section of code
c
      integer jsd
      integer jev
c
      double precision jua
      double precision jub
      double precision cp(limit)
      double precision cm(limit)
      double precision bp(limit)
      double precision bm(limit)
      double precision hp(limit,2)
      double precision qp(limit,2)
      double precision aa
      double precision bb
      double precision cc
```

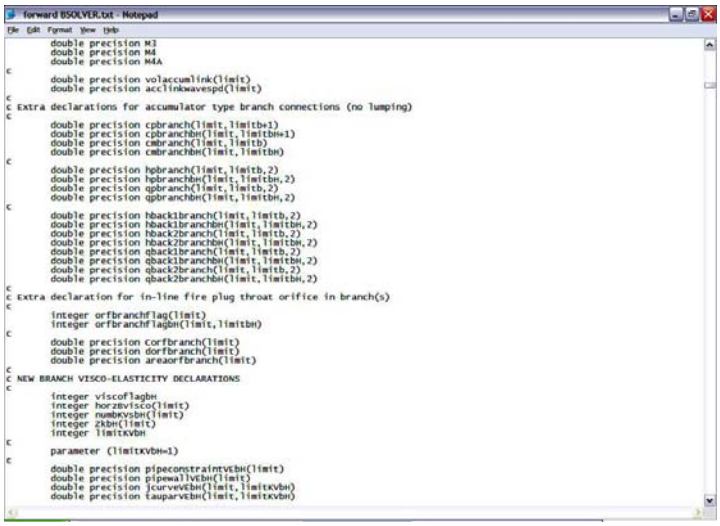
Appendix M – Fortran code for BSOLVER and NLFIT subroutines



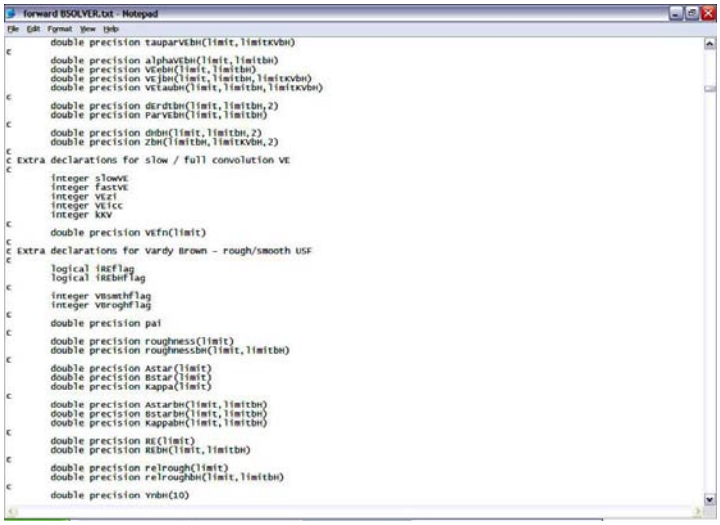




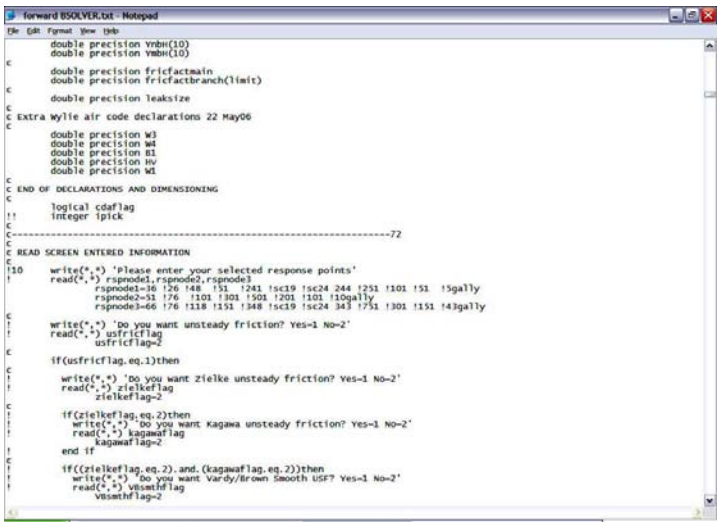
Appendix M – Fortran code for BSOLVER and NLFIT subroutines



```
forward BSOLVER.txt - Notepad
File  Edit  Format  View  Help
      double precision M3
      double precision M4
      double precision M4A
C
      double precision volaccumlink(limit)
      double precision acclinkwavespd(limit)
C
C Extra declarations for accumulator type branch connections (no lumping)
C
      double precision cpbranch(limit,limitb+1)
      double precision cpbranchbh(limit,limitbh+1)
      double precision cmbranch(limit,limitb)
      double precision cmbranchbh(limit,limitbh)
C
      double precision hpbranch(limit,limitb,2)
      double precision hpbranchbh(limit,limitbh,2)
      double precision qpbranch(limit,limitb,2)
      double precision qpbranchbh(limit,limitbh,2)
C
      double precision hback1branch(limit,limitb,2)
      double precision hback1branchbh(limit,limitbh,2)
      double precision hback2branch(limit,limitb,2)
      double precision hback2branchbh(limit,limitbh,2)
      double precision qback1branch(limit,limitb,2)
      double precision qback1branchbh(limit,limitbh,2)
      double precision qback2branch(limit,limitb,2)
      double precision qback2branchbh(limit,limitbh,2)
C
C Extra declaration for in-line fire plug throat orifice in branch(s)
C
      integer orfbranchflag(limit)
      integer orfbranchflagbh(limit,limitbh)
C
      double precision corfbranch(limit)
      double precision dorfbranch(limit)
      double precision areaorfbranch(limit)
C
C NEW BRANCH VISCO-ELASTICITY DECLARATIONS
C
      integer viscoflagbh
      integer horzbvisco(limit)
      integer numbkvsbh(limit)
      integer zkbh(limit)
      integer limitkvbh
C
      parameter (limitkvbh=1)
C
      double precision pipeconstraintvebh(limit)
      double precision pipewallvebh(limit)
      double precision jcurvevebh(limit,limitkvbh)
      double precision tauparvebh(limit,limitkvbh)
```

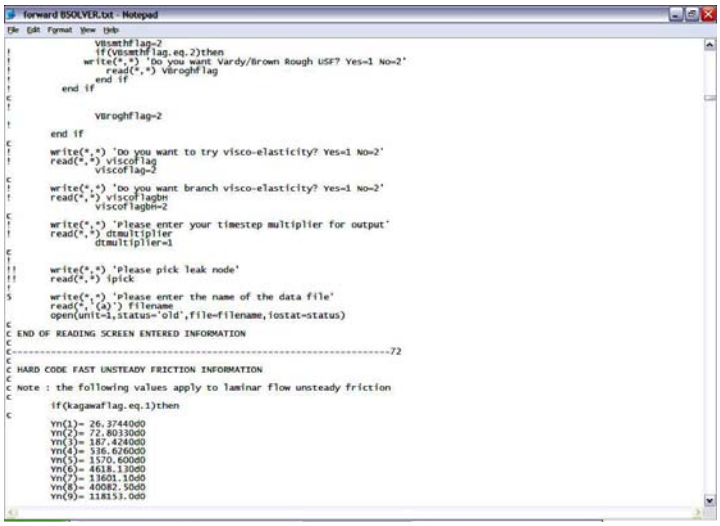


```
forward BSOLVER.txt - Notepad
File  Edit  Format  View  Help
      double precision tauparvebh(limit,limitkvbh)
C
      double precision alphavebh(limit,limitbh)
      double precision vebh(limit,limitbh)
      double precision vejbh(limit,limitbh,limitkvbh)
      double precision vetaubh(limit,limitbh,limitkvbh)
C
      double precision derdtbh(limit,limitbh,2)
      double precision parvebh(limit,limitbh)
C
      double precision debh(limit,limitbh,2)
      double precision zbh(limitbh,limitkvbh,2)
C
C Extra declarations for slow / full convolution VE
C
      integer slowve
      integer fastve
      integer vlzi
      integer velcc
      integer kkv
C
      double precision vefn(limit)
C
C Extra declarations for Vardy Brown - rough/smooth USF
C
      logical iREflag
      logical iREbhflag
C
      integer vbsmthflag
      integer vbroghflag
C
      double precision pai
C
      double precision roughness(limit)
      double precision roughnessbh(limit,limitbh)
C
      double precision Astar(limit)
      double precision Bstar(limit)
      double precision Kappa(limit)
C
      double precision Astarbh(limit,limitbh)
      double precision Bstarbh(limit,limitbh)
      double precision Kappabh(limit,limitbh)
C
      double precision RE(limit)
      double precision REbh(limit,limitbh)
C
      double precision relrough(limit)
      double precision relroughbh(limit,limitbh)
C
      double precision Ynbh(10)
```

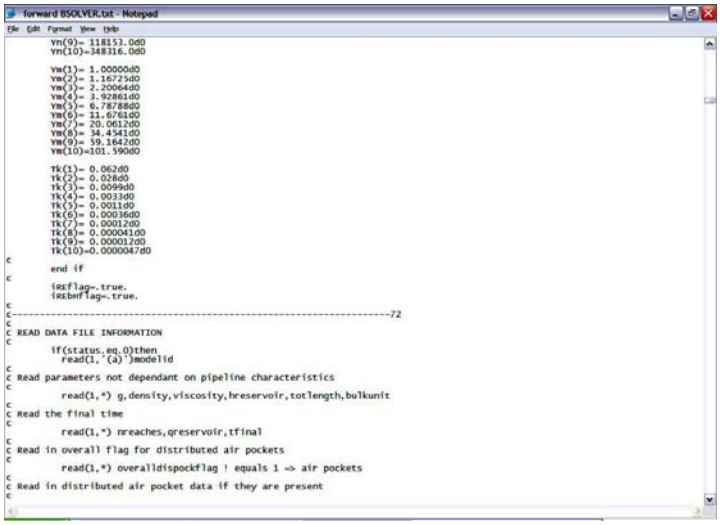


```
forward BSOLVER.txt - Notepad
File  Edit  Format  View  Help
      double precision Ynbh(10)
      double precision Ymbh(10)
C
      double precision fricfactmain
      double precision fricfactbranch(limit)
C
      double precision leaksize
C
C Extra Wylie air code declarations 22 May06
C
      double precision W3
      double precision W4
      double precision B1
      double precision Hv
      double precision W1
C
C END OF DECLARATIONS AND DIMENSIONING
C
      logical cdaflag
!!    integer ipick
C
C-----------------------------------------------------------------------72
C
C READ SCREEN ENTERED INFORMATION
C
!10   write(*,*) 'Please enter your selected response points'
!     read(*,*) rspnode1,rspnode2,rspnode3
            rspnode1=36 !26 !48  !51  !241 !sc19 !sc24 244 !251 !101 !51  !5gally
            rspnode2=51 !76  !101 !301 !501 !201 !101 !10gally
            rspnode3=66 !76 !118 !151 !348 !sc19 !sc24 343 !751 !301 !151 !43gally
C
      write(*,*) 'Do you want unsteady friction? Yes=1 No=2'
!     read(*,*) usfricflag
            usfricflag=2
C
      if(usfricflag.eq.1)then
C
!       write(*,*) 'Do you want Zielke unsteady friction? Yes=1 No=2'
!       read(*,*) zielkeflag
            zielkeflag=2
C
!       if(zielkeflag.eq.2)then
!         write(*,*)  'Do you want Kagawa unsteady friction? Yes=1 No=2'
!         read(*,*) kagawaflag
            kagawaflag=2
!       end if
C
!       if((zielkeflag.eq.2).and.(kagawaflag.eq.2))then
!         write(*,*)  'Do you want Vardy/Brown Smooth USF? Yes=1 No=2'
!         read(*,*) vbsmthflag
            vbsmthflag=2
```

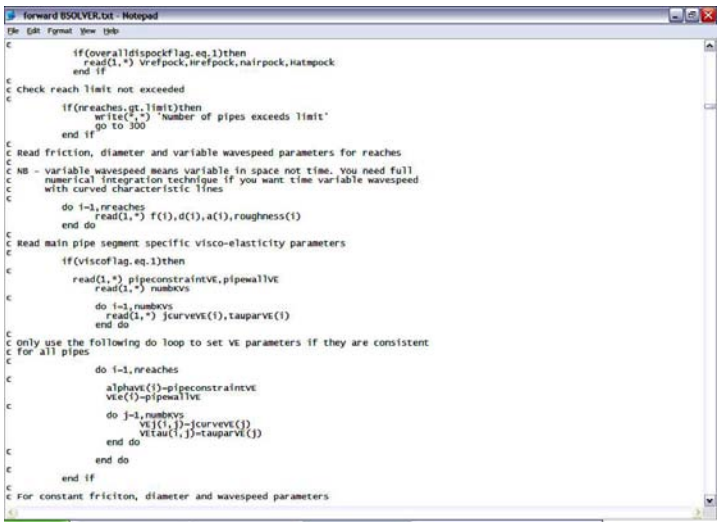
Appendix M – Fortran code for BSOLVER and NLFIT subroutines



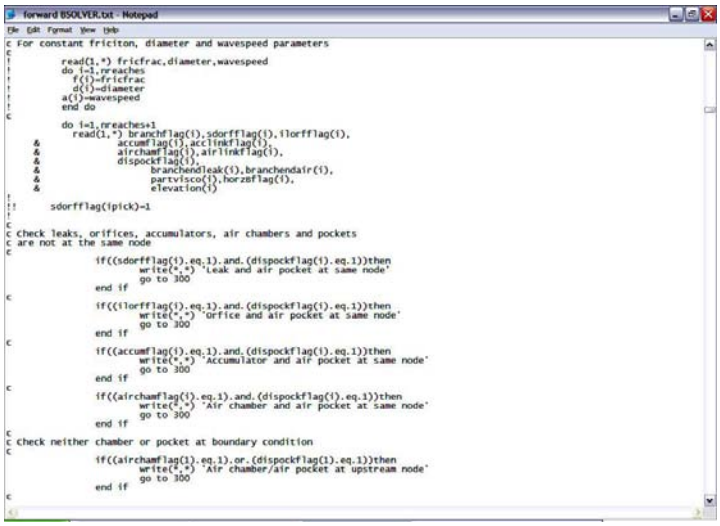
```
forward BSOLVER.txt - Notepad
File  Edit  Format  View  Help
                VBsmthflag=2
                if(VBsmthflag.eq.2)then
!             write(*,*) 'Do you want Vardy/Brown Rough USF? Yes=1 No=2'
!               read(*,*) VBroghflag
                end if
!       end if
C
!
                VBroghflag=2
!
        end if
C
!       write(*,*) 'Do you want to try visco-elasticity? Yes=1 No=2'
!       read(*,*) viscoflag
                viscoflag=2
C
!       write(*,*) 'Do you want branch visco-elasticity? Yes=1 No=2'
!       read(*,*) viscoflagbh
                viscoflagbh=2
C
!       write(*,*) 'Please enter your timestep multiplier for output'
!       read(*,*) dtmultiplier
                dtmultiplier=1
C
!
!!      write(*,*) 'Please pick leak node'
!!      read(*,*) ipick
!
5       write(*,*) 'Please enter the name of the data file'
        read(5,'(a)') filename
        open(unit=1,status='old',file=filename,iostat=status)
C
C END OF READING SCREEN ENTERED INFORMATION
C
C-----------------------------------------------------------------------72
C
C HARD CODE FAST UNSTEADY FRICTION INFORMATION
C
C Note : the following values apply to laminar flow unsteady friction
C
        if(kagawaflag.eq.1)then
C
        Yn(1)= 26.37440d0
        Yn(2)= 72.80330d0
        Yn(3)= 187.4240d0
        Yn(4)= 536.6260d0
        Yn(5)= 1570.600d0
        Yn(6)= 4618.130d0
        Yn(7)= 13601.10d0
        Yn(8)= 40082.50d0
        Yn(9)= 118153.0d0
```



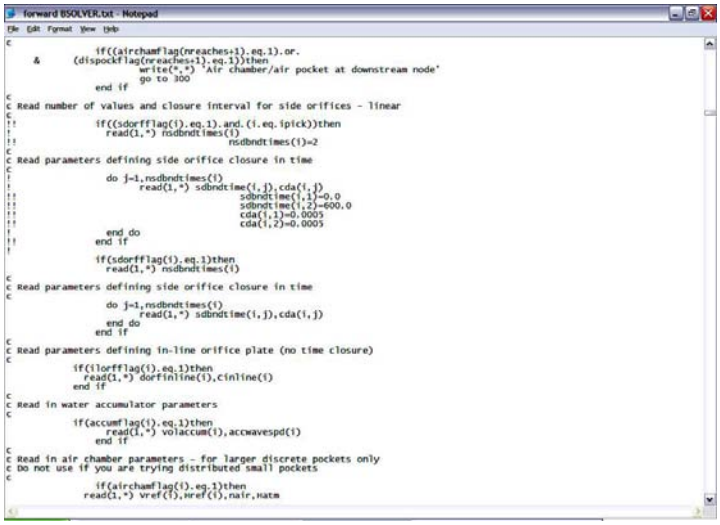
```
forward BSOLVER.txt - Notepad
File  Edit  Format  View  Help
        Yn(9)= 118153.0d0
        Yn(10)=348316.0d0

        Ym(1)= 1.00000d0
        Ym(2)= 1.16725d0
        Ym(3)= 2.20064d0
        Ym(4)= 3.92861d0
        Ym(5)= 6.78788d0
        Ym(6)= 11.6761d0
        Ym(7)= 20.0612d0
        Ym(8)= 34.4541d0
        Ym(9)= 59.1642d0
        Ym(10)=101.590d0

        Tk(1)= 0.062d0
        Tk(2)= 0.028d0
        Tk(3)= 0.0099d0
        Tk(4)= 0.0033d0
        Tk(5)= 0.0011d0
        Tk(6)= 0.00036d0
        Tk(7)= 0.00012d0
        Tk(8)= 0.000041d0
        Tk(9)= 0.000012d0
        Tk(10)=0.0000047d0
C
        end if
C
        iREflag=.true.
        iREbhflag=.true.
C
C-----------------------------------------------------------------------72
C
C READ DATA FILE INFORMATION
C
        if(status.eq.0)then
          read(1,'(a)')modelid
C
C Read parameters not dependant on pipeline characteristics
C
          read(1,*) g,density,viscosity,hreservoir,totlength,bulkunit
C
C Read the final time
C
          read(1,*) nreaches,qreservoir,tfinal
C
C Read in overall flag for distributed air pockets
C
          read(1,*) overalldispockflag ! equals 1 => air pockets
C
C Read in distributed air pocket data if they are present
C
```



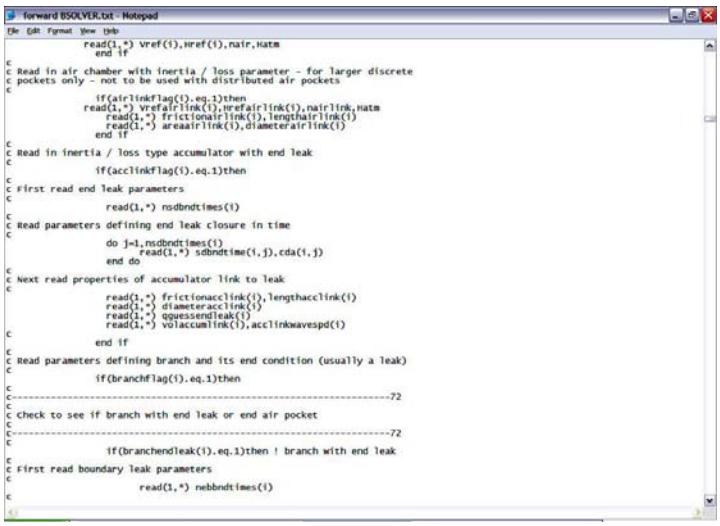
```
forward BSOLVER.txt - Notepad
File  Edit  Format  View  Help
C
            if(overalldispockflag.eq.1)then
              read(1,*) Vrefpock,Hrefpock,nairpock,Hatmpock
            end if
C
C Check reach limit not exceeded
C
          if(nreaches.gt.limit)then
              write(*,*) 'Number of pipes exceeds limit'
              go to 300
          end if
C
C Read friction, diameter and variable wavespeed parameters for reaches
C
C NB - variable wavespeed means variable in space not time. You need full
C      numerical integration technique if you want time variable wavespeed
C      with curved characteristic lines
C
          do i=1,nreaches
              read(1,*) f(i),d(i),a(i),roughness(i)
          end do
C
C Read main pipe segment specific visco-elasticity parameters
C
          if(viscoflag.eq.1)then
C
            read(1,*) pipeconstraintVE,pipewallVE
                read(1,*) numbKVs
C
                do i=1,numbKVs
                  read(1,*) jcurveVE(i),tauparVE(i)
                end do
C
C Only use the following do loop to set VE parameters if they are consistent
C for all pipes
C
                do i=1,nreaches
C
                  alphaVE(i)=pipeconstraintVE
                  wie(i)=pipewallVE
C
                  do j=1,numbKVs
                        VEj(i,j)=jcurveVE(j)
                        VEtau(i,j)=tauparVE(j)
                  end do
C
                end do
C
          end if
C
C For constant friciton, diameter and wavespeed parameters
```



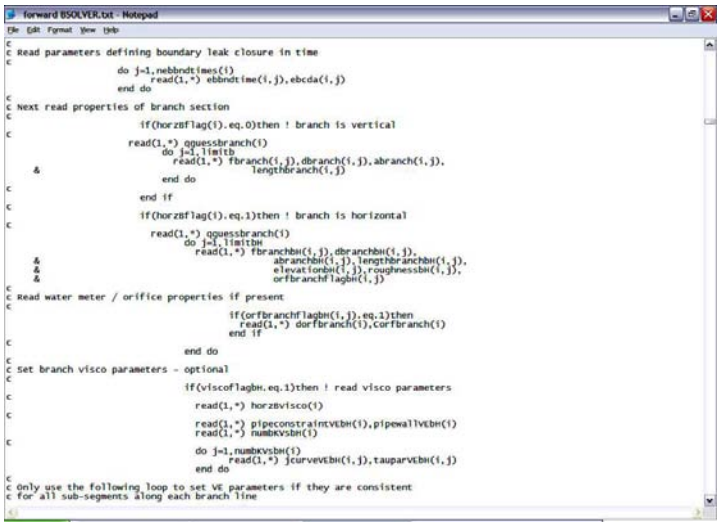
```
forward BSOLVER.txt - Notepad
File  Edit  Format  View  Help
C For constant friciton, diameter and wavespeed parameters
C
!         read(1,*) fricfrac,diameter,wavespeed
!         do i=1,nreaches
!           f(i)=fricfrac
!           d(i)=diameter
!         a(i)=wavespeed
!         end do
C
          do i=1,nreaches+1
            read(1,*) branchflag(i),sdorfflag(i),ilorfflag(i),
     &                accumflag(i),acclinkflag(i),
     &                airchamflag(i),airlinkflag(i),
     &                dispockflag(i),
     &                branchendleak(i),branchendair(i),
     &                partvisco(i),horzbflag(i),
     &                elevation(i)
!
!!      sdorfflag(ipick)=1
!
C
C Check leaks, orifices, accumulators, air chambers and pockets
C are not at the same node
C
                  if((sdorfflag(i).eq.1).and.(dispockflag(i).eq.1))then
                        write(*,*) 'Leak and air pocket at same node'
                        go to 300
                  end if
C
                  if((ilorfflag(i).eq.1).and.(dispockflag(i).eq.1))then
                        write(*,*) 'Orfice and air pocket at same node'
                        go to 300
                  end if
C
                  if((accumflag(i).eq.1).and.(dispockflag(i).eq.1))then
                        write(*,*) 'Accumulator and air pocket at same node'
                        go to 300
                  end if
C
                  if((airchamflag(i).eq.1).and.(dispockflag(i).eq.1))then
                        write(*,*) 'Air chamber and air pocket at same node'
                        go to 300
                  end if
C
C Check neither chamber or pocket at boundary condition
C
                  if((airchamflag(1).eq.1).or.(dispockflag(1).eq.1))then
                        write(*,*) 'Air chamber/air pocket at upstream node'
                        go to 300
                  end if
C
```



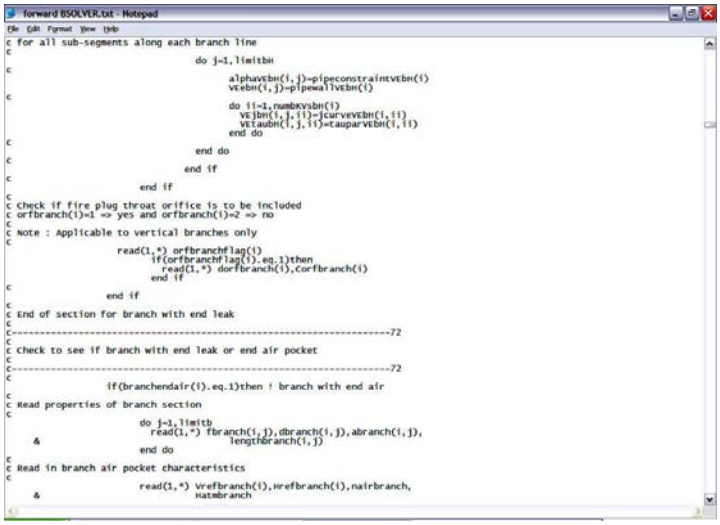
```
forward BSOLVER.txt - Notepad
File  Edit  Format  View  Help
C
                  if((airchamflag(nreaches+1).eq.1).or.
     &       (dispockflag(nreaches+1).eq.1))then
                        write(*,*) 'Air chamber/air pocket at downstream node'
                        go to 300
                  end if
C
C Read number of values and closure interval for side orifices - linear
C
!!                if((sdorfflag(i).eq.1).and.(i.eq.ipick))then
!                   read(1,*) nsdbndtimes(i)
!!                                        nsdbndtimes(i)=2
C
C Read parameters defining side orifice closure in time
C
!                   do j=1,nsdbndtimes(i)
!                         read(1,*) sdbndtime(i,j),cda(i,j)
!!                                        sdbndtime(i,1)=0.0
!!                                        sdbndtime(i,2)=600.0
!!                                        cda(i,1)=0.0005
!!                                        cda(i,2)=0.0005
!                   end do
!!                end if
!
                  if(sdorfflag(i).eq.1)then
                    read(1,*) nsdbndtimes(i)
C
C Read parameters defining side orifice closure in time
C
                    do j=1,nsdbndtimes(i)
                          read(1,*) sdbndtime(i,j),cda(i,j)
                    end do
                  end if
C
C Read parameters defining in-line orifice plate (no time closure)
C
              if(ilorfflag(i).eq.1)then
                read(1,*) dorfinline(i),cinline(i)
              end if
C
C Read in water accumulator parameters
C
              if(accumflag(i).eq.1)then
                   read(1,*) volaccum(i),accwavespd(i)
                end if
C
C Read in air chamber parameters - for larger discrete pockets only
C Do not use if you are trying distributed small pockets
C
                  if(airchamflag(i).eq.1)then
                read(1,*) vref(i),href(i),nair,natm
```



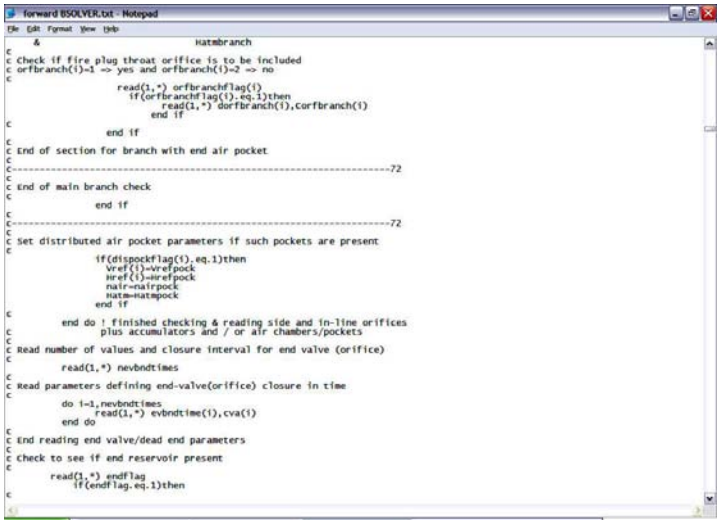
```
forward BSOLVER.txt - Notepad
File  Edit  Format  View  Help
                read(1,*) vref(i),href(i),nair,natm
                  end if
C
C Read in air chamber with inertia / loss parameter - for larger discrete
C pockets only - not to be used with distributed air pockets
C
                  if(airlinkflag(i).eq.1)then
                read(1,*) vrefairlink(i),hrefairlink(i),nairlink,natm
                    read(1,*) frictionairlink(i),lengthairlink(i)
                    read(1,*) areaairlink(i),diameterairlink(i)
                  end if
C
C Read in inertia / loss type accumulator with end leak
C
                  if(acclinkflag(i).eq.1)then
C
C First read end leak parameters
C
                    read(1,*) nsdbndtimes(i)
C
C Read parameters defining end leak closure in time
C
                    do j=1,nsdbndtimes(i)
                          read(1,*) sdbndtime(i,j),cda(i,j)
                    end do
C
C Next read properties of accumulator link to leak
C
                    read(1,*) frictionacclink(i),lengthacclink(i)
                    read(1,*) diameteracclink(i)
                    read(1,*) qguessendleak(i)
                    read(1,*) volaccumlink(i),acclinkwavespd(i)
C
                  end if
C
C Read parameters defining branch and its end condition (usually a leak)
C
                  if(branchflag(i).eq.1)then
C
C-----------------------------------------------------------------72
C
C Check to see if branch with end leak or end air pocket
C
C-----------------------------------------------------------------72
C
                    if(branchendleak(i).eq.1)then ! branch with end leak
C
C First read boundary leak parameters
C
                          read(1,*) nebbndtimes(i)
C
```


forward BSOLVER.txt - Notepad

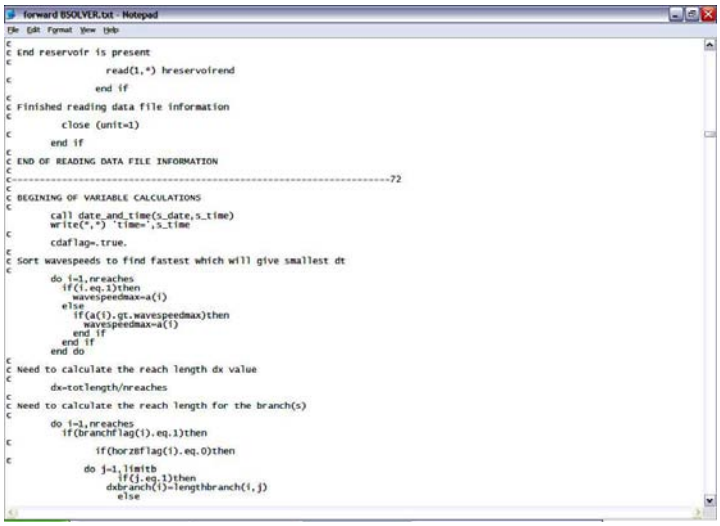
```
File  Edit  Format  View  Help
C
C Read parameters defining boundary leak closure in time
C
                    do j=1,nebbndtimes(i)
                       read(1,*) ebbndtime(i,j),ebcda(i,j)
                    end do
C
C Next read properties of branch section
C
                      if(horzbflag(i).eq.0)then ! branch is vertical
C
                     read(1,*) qguessbranch(i)
                          do j=1,limitb
                            read(1,*) fbranch(i,j),dbranch(i,j),abranch(i,j),
     &                                    lengthbranch(i,j)
                          end do
C
                      end if
C
                      if(horzbflag(i).eq.1)then ! branch is horizontal
C
                        read(1,*) qguessbranch(i)
                             do j=1,limitbH
                               read(1,*) fbranchbH(i,j),dbranchbH(i,j),
     &                                       abranchbH(i,j),lengthbranchbH(i,j),
     &                                       elevationbH(i,j),roughnessbH(i,j),
     &                                       orfbranchflagbH(i,j)
C
C Read water meter / orifice properties if present
C
                                       if(orfbranchflagbH(i,j).eq.1)then
                                        read(1,*) dorfbranch(i),corfbranch(i)
                                       end if
C
                                  end do
C
C Set branch visco parameters - optional
C
                                  if(viscoflagbH.eq.1)then ! read visco parameters
C
                                    read(1,*) horzbvisco(i)
C
                                    read(1,*) pipeconstraintvEbH(i),pipewallvEbH(i)
                                    read(1,*) numbKvsbH(i)
C
                                    do j=1,numbKvsbH(i)
                                          read(1,*) jcurveveEbH(i,j),tauparvEbH(i,j)
                                    end do
C
C Only use the following loop to set VE parameters if they are consistent
C for all sub-segments along each branch line
```


forward BSOLVER.txt - Notepad

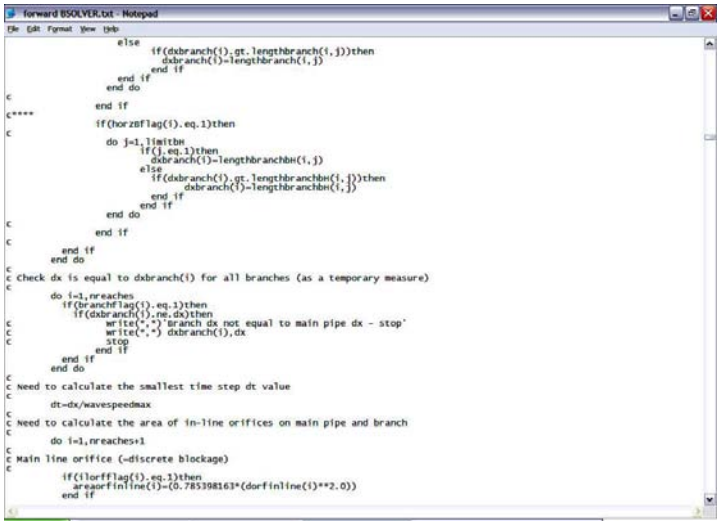
```
File  Edit  Format  View  Help
C for all sub-segments along each branch line
C
                                    do j=1,limitbH
C
                                          alphavEbH(i,j)=pipeconstraintvEbH(i)
                                          vEebH(i,j)=pipewallvEbH(i)
C
                                          do ii=1,numbKvsbH(i)
                                            vEjbH(i,j,ii)=jcurveveEbH(i,ii)
                                            vEtaubH(i,j,ii)=tauparvEbH(i,ii)
                                          end do
C
                                    end do
C
                                  end if
C
                           end if
C
C Check if fire plug throat orifice is to be included
C orfbranch(i)=1 => yes and orfbranch(i)=2 => no
C
C Note : Applicable to vertical branches only
C
                     read(1,*) orfbranchflag(i)
                          if(orfbranchflag(i).eq.1)then
                            read(1,*) dorfbranch(i),corfbranch(i)
                          end if
C
                    end if
C
C End of section for branch with end leak
C
C-----------------------------------------------------------------------72
C
C Check to see if branch with end leak or end air pocket
C
C-----------------------------------------------------------------------72
C
                   if(branchendair(i).eq.1)then ! branch with end air
C
C Read properties of branch section
C
                      do j=1,limitb
                        read(1,*) fbranch(i,j),dbranch(i,j),abranch(i,j),
     &                                lengthbranch(i,j)
                      end do
C
C Read in branch air pocket characteristics
C
                      read(1,*) vrefbranch(i),hrefbranch(i),nairbranch,
     &                          hatmbranch
```


forward BSOLVER.txt - Notepad

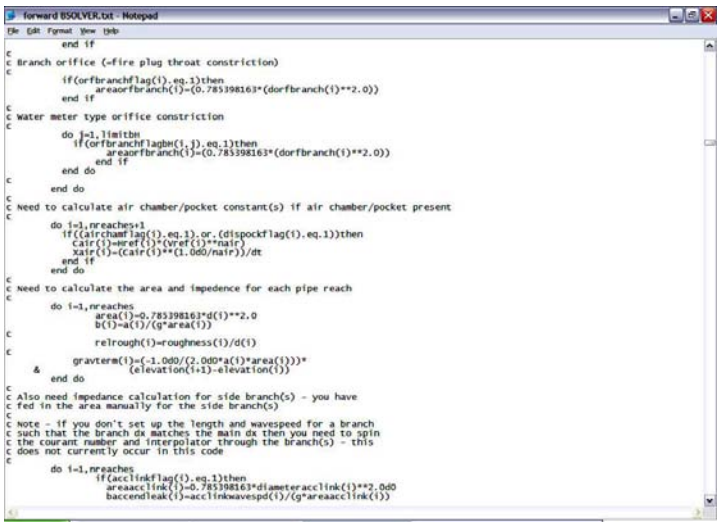
```
File  Edit  Format  View  Help
     &                          hatmbranch
C
C Check if fire plug throat orifice is to be included
C orfbranch(i)=1 => yes and orfbranch(i)=2 => no
C
                     read(1,*) orfbranchflag(i)
                          if(orfbranchflag(i).eq.1)then
                            read(1,*) dorfbranch(i),corfbranch(i)
                          end if
C
                    end if
C
C End of section for branch with end air pocket
C
C-----------------------------------------------------------------------72
C
C End of main branch check
C
                  end if
C
C-----------------------------------------------------------------------72
C
C Set distributed air pocket parameters if such pockets are present
C
                if(dispockflag(i).eq.1)then
                  vref(i)=vrefpock
                  href(i)=hrefpock
                  nair=nairpock
                  natm=natmpock
                end if
C
          end do ! finished checking & reading side and in-line orifices
C                plus accumulators and / or air chambers/pockets
C
C Read number of values and closure interval for end valve (orifice)
C
          read(1,*) nevbndtimes
C
C Read parameters defining end-valve(orifice) closure in time
C
          do i=1,nevbndtimes
             read(1,*) evbndtime(i),cva(i)
          end do
C
C End reading end valve/dead end parameters
C
C Check to see if end reservoir present
C
        read(1,*) endflag
            if(endflag.eq.1)then
C
```


forward BSOLVER.txt - Notepad

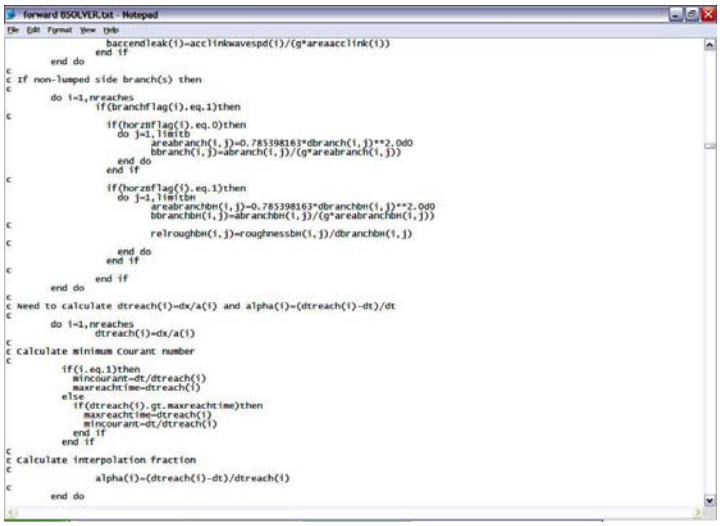
```
File  Edit  Format  View  Help
C
C End reservoir is present
C
                read(1,*) hreservoirend
C
              end if
C
C Finished reading data file information
C
         close (unit=1)
C
        end if
C
C END OF READING DATA FILE INFORMATION
C
C-------------------------------------------------------------------------72
C
C BEGINING OF VARIABLE CALCULATIONS
C
        call date_and_time(s_date,s_time)
        write(*,*) 'time=',s_time
C
        cdaflag=.true.
C
C Sort wavespeeds to find fastest which will give smallest dt
C
        do i=1,nreaches
          if(i.eq.1)then
            wavespeedmax=a(i)
          else
            if(a(i).gt.wavespeedmax)then
              wavespeedmax=a(i)
            end if
          end if
        end do
C
C Need to calculate the reach length dx value
C
        dx=totlength/nreaches
C
C Need to calculate the reach length for the branch(s)
C
        do i=1,nreaches
          if(branchflag(i).eq.1)then
C
                if(horzbflag(i).eq.0)then
C
              do j=1,limitb
                  if(j.eq.1)then
                dxbranch(i)=lengthbranch(i,j)
                  else
```


forward BSOLVER.txt - Notepad

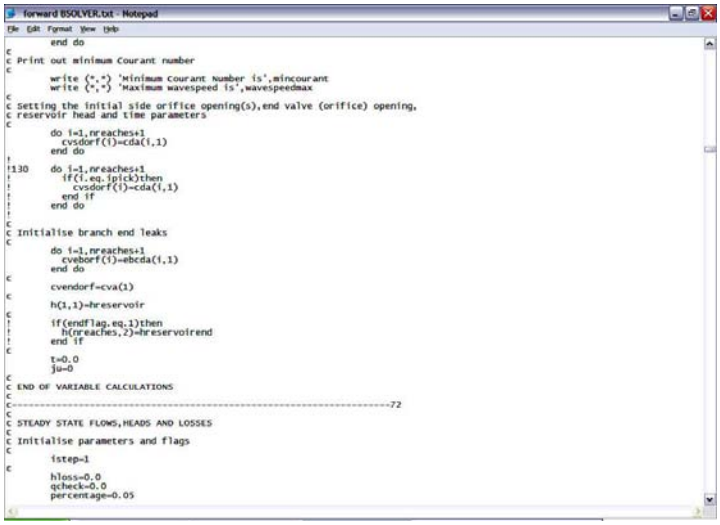
```
File  Edit  Format  View  Help
                    else
                          if(dxbranch(i).gt.lengthbranch(i,j))then
                            dxbranch(i)=lengthbranch(i,j)
                          end if
                  end if
                end do
C
              end if
C****
              if(horzbflag(i).eq.1)then
C
                do j=1,limitbh
                    if(j.eq.1)then
                      dxbranch(i)=lengthbranchbh(i,j)
                    else
                      if(dxbranch(i).gt.lengthbranchbh(i,j))then
                            dxbranch(i)=lengthbranchbh(i,j)
                      end if
                    end if
                end do
C
              end if
C
          end if
        end do
C
C Check dx is equal to dxbranch(i) for all branches (as a temporary measure)
C
        do i=1,nreaches
          if(branchflag(i).eq.1)then
            if(dxbranch(i).ne.dx)then
C             write(*,*) 'Branch dx not equal to main pipe dx - stop'
C             write(*,*) dxbranch(i),dx
C             stop
            end if
          end if
        end do
C
C Need to calculate the smallest time step dt value
C
        dt=dx/wavespeedmax
C
C Need to calculate the area of in-line orifices on main pipe and branch
C
        do i=1,nreaches+1
C
C Main line orifice (=discrete blockage)
C
          if(ilorfflag(i).eq.1)then
            areaorfinline(i)=(0.785398163*(dorfinline(i)**2.0))
          end if
```


forward BSOLVER.txt - Notepad

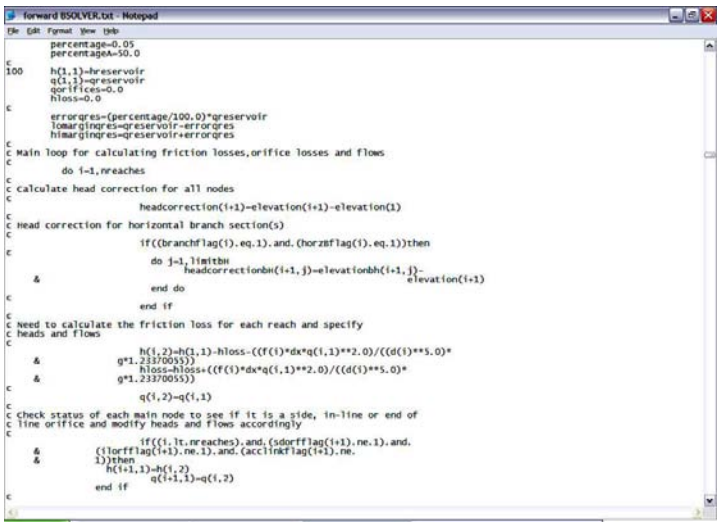
```
File  Edit  Format  View  Help
          end if
C
C Branch orifice (=fire plug throat constriction)
C
          if(orfbranchflag(i).eq.1)then
            areaorfbranch(i)=(0.785398163*(dorfbranch(i)**2.0))
          end if
C
C Water meter type orifice constriction
C
          do j=1,limitbh
            if(orfbranchflagbh(i,j).eq.1)then
              areaorfbranch(i)=(0.785398163*(dorfbranch(i)**2.0))
            end if
          end do
C
        end do
C
C Need to calculate air chamber/pocket constant(s) if air chamber/pocket present
C
        do i=1,nreaches+1
          if((airchamflag(i).eq.1).or.(dispockflag(i).eq.1))then
            cair(i)=href(i)*(vref(i)**nair)
            xair(i)=(cair(i)**(1.0d0/nair))/dt
          end if
        end do
C
C Need to calculate the area and impedence for each pipe reach
C
        do i=1,nreaches
              area(i)=0.785398163*d(i)**2.0
              b(i)=a(i)/(g*area(i))
C
              relrough(i)=roughness(i)/d(i)
C
          gravterm(i)=(-1.0d0/(2.0d0*a(i)*area(i)))*
     &              (elevation(i+1)-elevation(i))
        end do
C
C Also need impedance calculation for side branch(s) - you have
C fed in the area manually for the side branch(s)
C
C Note - if you don't set up the length and wavespeed for a branch
C such that the branch dx matches the main dx then you need to spin
C the courant number and interpolator through the branch(s) - this
C does not currently occur in this code
C
        do i=1,nreaches
              if(acclinkflag(i).eq.1)then
                areaacclink(i)=0.785398163*diameteracclink(i)**2.0d0
                baccendleak(i)=acclinkwavespd(i)/(g*areaacclink(i))
```



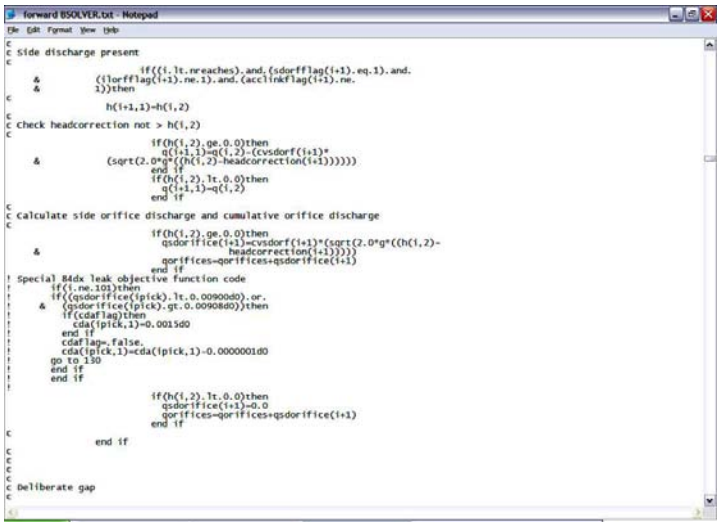
```
forward BSOLVER.txt - Notepad
File  Edit  Format  View  Help
                  baccendleak(i)=acclinkwavespd(i)/(g*areaacclink(i))
                end if
        end do
c
c If non-lumped side branch(s) then
c
        do i=1,nreaches
                if(branchflag(i).eq.1)then
c
                  if(horzbflag(i).eq.0)then
                    do j=1,limitb
                          areabranch(i,j)=0.785398163*dbranch(i,j)**2.0d0
                          bbranch(i,j)=abranch(i,j)/(g*areabranch(i,j))
                    end do
                  end if
c
                  if(horzbflag(i).eq.1)then
                    do j=1,limitbh
                          areabranchbh(i,j)=0.785398163*dbranchbh(i,j)**2.0d0
                          bbranchbh(i,j)=abranchbh(i,j)/(g*areabranchbh(i,j))
c
                          relroughbh(i,j)=roughnessbh(i,j)/dbranchbh(i,j)
c
                    end do
                  end if
c
                end if
        end do
c
c Need to calculate dtreach(i)=dx/a(i) and alpha(i)=(dtreach(i)-dt)/dt
c
        do i=1,nreaches
                dtreach(i)=dx/a(i)
c
c Calculate minimum Courant number
c
         if(i.eq.1)then
           mincourant=dt/dtreach(i)
           maxreachtime=dtreach(i)
         else
           if(dtreach(i).gt.maxreachtime)then
             maxreachtime=dtreach(i)
             mincourant=dt/dtreach(i)
           end if
         end if
c
c Calculate interpolation fraction
c
                alpha(i)=(dtreach(i)-dt)/dtreach(i)
c
        end do
```



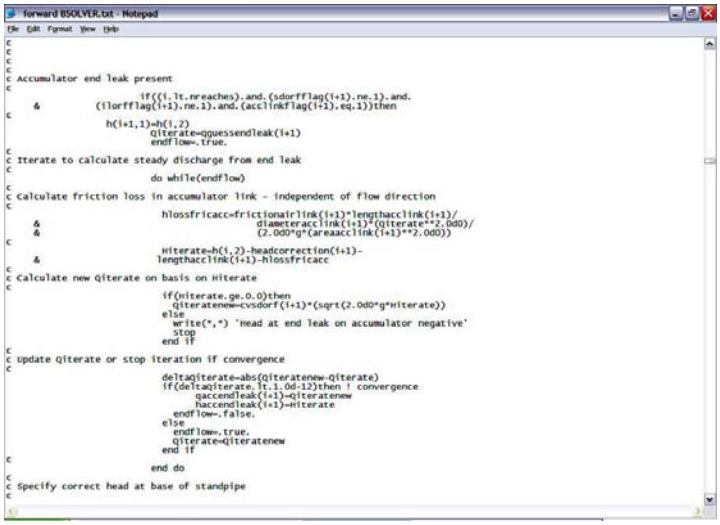
```
forward BSOLVER.txt - Notepad
File  Edit  Format  View  Help
        end do
c
c Print out minimum Courant number
c
        write (*,*) 'Minimum Courant Number is',mincourant
        write (*,*) 'Maximum wavespeed is',wavespeedmax
c
c Setting the initial side orifice opening(s),end valve (orifice) opening,
c reservoir head and time parameters
c
        do i=1,nreaches+1
          cvsdorf(i)=cda(i,1)
        end do
!
!130    do i=1,nreaches+1
!         if(i.eq.ipick)then
!           cvsdorf(i)=cda(i,1)
!         end if
!       end do
!
c
c Initialise branch end leaks
c
        do i=1,nreaches+1
          cvebdorf(i)=ebcda(i,1)
        end do
c
        cvendorf=cva(1)
c
        h(1,1)=hreservoir
c
!       if(endflag.eq.1)then
!         h(nreaches,2)=hreservoirend
!       end if
c
        t=0.0
        jw=0
c
c END OF VARIABLE CALCULATIONS
c
c-----------------------------------------------------------------------72
c
c STEADY STATE FLOWS,HEADS AND LOSSES
c
c Initialise parameters and flags
c
        istep=1
c
        hloss=0.0
        qcheck=0.0
        percentage=0.05
```



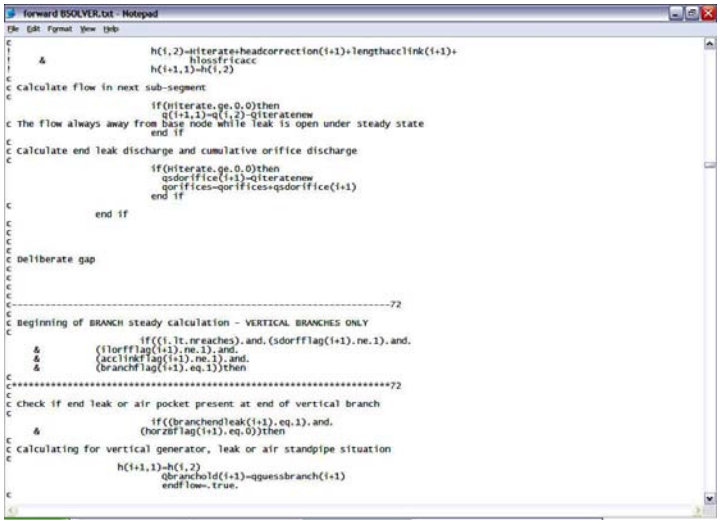
```
forward BSOLVER.txt - Notepad
File  Edit  Format  View  Help
        percentage=0.05
        percentageA=50.0
c
100     h(1,1)=hreservoir
        q(1,1)=qreservoir
        qorifices=0.0
        hloss=0.0
c
        errorqres=(percentage/100.0)*qreservoir
        lomarginqres=qreservoir-errorqres
        himarginqres=qreservoir+errorqres
c
c Main loop for calculating friction losses,orifice losses and flows
c
          do i=1,nreaches
c
c Calculate head correction for all nodes
c
                        headcorrection(i+1)=elevation(i+1)-elevation(1)
c
c Head correction for horizontal branch section(s)
c
                        if((branchflag(i).eq.1).and.(horzbflag(i).eq.1))then
c
                          do j=1,limitbh
                                headcorrectionbh(i+1,j)=elevationbh(i+1,j)-
     &                                                          elevation(i+1)
                          end do
c
                        end if
c
c Need to calculate the friction loss for each reach and specify
c heads and flows
c
                        h(i,2)=h(1,1)-hloss-((f(i)*dx*q(i,1)**2.0)/((d(i)**5.0)*
     &                g*1.2337055))
                          hloss=hloss+((f(i)*dx*q(i,1)**2.0)/((d(i)**5.0)*
     &                g*1.2337055))
c
                        q(i,2)=q(i,1)
c
c Check status of each main node to see if it is a side, in-line or end of
c line orifice and modify heads and flows accordingly
c
                        if((i.lt.nreaches).and.(sdorfflag(i+1).ne.1).and.
     &          (ilorfflag(i+1).ne.1).and.(acclinkflag(i+1).ne.
     &          1))then
                          h(i+1,1)=h(i,2)
                          q(i+1,1)=q(i,2)
                end if
c
```



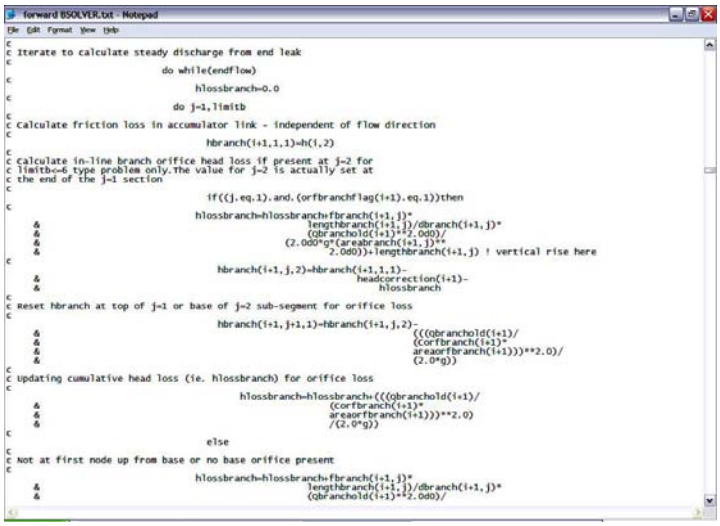

```
c
c Side discharge present
c
                        if((i.lt.nreaches).and.(sdorfflag(i+1).eq.1).and.
     &          (ilorfflag(i+1).ne.1).and.(acclinkflag(i+1).ne.
     &          1))then
c
                          h(i+1,1)=h(i,2)
c
c Check headcorrection not > h(i,2)
c
                        if(h(i,2).ge.0.0)then
                          q(i+1,1)=q(i,2)-(cvsdorf(i+1)*
     &            (sqrt(2.0*g*((h(i,2)-headcorrection(i+1))))))
                        end if
                        if(h(i,2).lt.0.0)then
                          q(i+1,1)=q(i,2)
                        end if
c
c Calculate side orifice discharge and cumulative orifice discharge
c
                        if(h(i,2).ge.0.0)then
                          qsdorifice(i+1)=cvsdorf(i+1)*(sqrt(2.0*g*((h(i,2)-
     &                                  headcorrection(i+1)))))
                          qorifices=qorifices+qsdorifice(i+1)
                        end if
! Special B4dx leak objective function code
!       if(i.ne.101)then
!       if((qsdorifice(ipick).lt.0.00900d0).or.
!     &   (qsdorifice(ipick).gt.0.00908d0))then
!         if(cdaflag)then
!           cda(ipick,1)=0.0015d0
!         end if
!         cdaflag=.false.
!         cda(ipick,1)=cda(ipick,1)-0.0000001d0
!       go to 130
!       end if
!       end if
!
                        if(h(i,2).lt.0.0)then
                          qsdorifice(i+1)=0.0
                          qorifices=qorifices+qsdorifice(i+1)
                        end if
c
                end if
c
c
c
c
c Deliberate gap
c
```




```
c
c
c
c
c Accumulator end leak present
c
                        if((i.lt.nreaches).and.(sdorfflag(i+1).ne.1).and.
     &          (ilorfflag(i+1).ne.1).and.(acclinkflag(i+1).eq.1))then
c
                  h(i+1,1)=h(i,2)
                        qiterate=qguessendleak(i+1)
                        endflow=.true.
c
c Iterate to calculate steady discharge from end leak
c
                        do while(endflow)
c
c Calculate friction loss in accumulator link - independent of flow direction
c
                          hlossfricacc=frictionairlink(i+1)*lengthacclink(i+1)/
     &                                 diameteracclink(i+1)*(qiterate**2.0d0)/
     &                                 (2.0d0*g*(areaacclink(i+1)**2.0d0))
c
                          Hiterate=h(i,2)-headcorrection(i+1)-
     &                    lengthacclink(i+1)-hlossfricacc
c
c Calculate new Qiterate on basis on Hiterate
c
                          if(Hiterate.ge.0.0)then
                            qiteratenew=cvsdorf(i+1)*(sqrt(2.0d0*g*Hiterate))
                          else
                            write(*,*) 'Head at end leak on accumulator negative'
                            stop
                          end if
c
c Update Qiterate or stop iteration if convergence
c
                          deltaQiterate=abs(qiteratenew-qiterate)
                          if(deltaQiterate.lt.1.0d-12)then ! convergence
                                qaccendleak(i+1)=qiteratenew
                                haccendleak(i+1)=Hiterate
                            endflow=.false.
                          else
                            endflow=.true.
                            qiterate=qiteratenew
                          end if
c
                        end do
c
c Specify correct head at base of standpipe
c
```



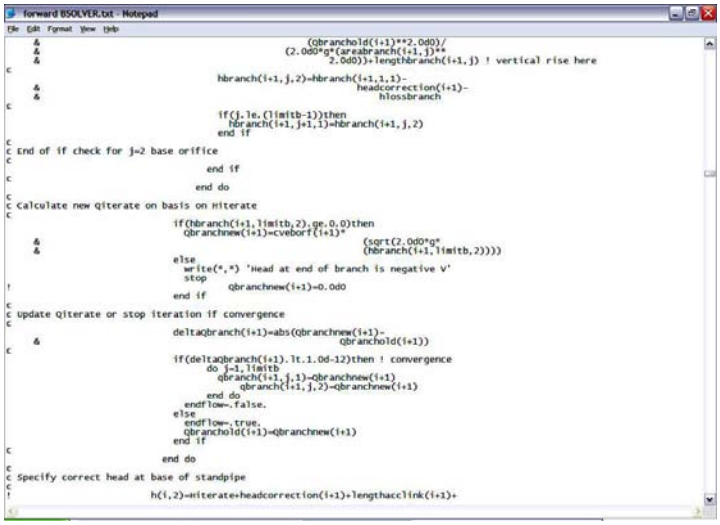

```
c
!                       h(i,2)=Hiterate+headcorrection(i+1)+lengthacclink(i+1)+
!     &                        hlossfricacc
!                       h(i+1,1)=h(i,2)
c
c Calculate flow in next sub-segment
c
                        if(Hiterate.ge.0.0)then
                          q(i+1,1)=q(i,2)-qiteratenew
c The flow always away from base node while leak is open under steady state
                        end if
c
c Calculate end leak discharge and cumulative orifice discharge
c
                        if(Hiterate.ge.0.0)then
                          qsdorifice(i+1)=qiteratenew
                          qorifices=qorifices+qsdorifice(i+1)
                        end if
c
                end if
c
c
c
c
c Deliberate gap
c
c
c
c-------------------------------------------------------------------72
c
c Beginning of BRANCH steady calculation - VERTICAL BRANCHES ONLY
c
                        if((i.lt.nreaches).and.(sdorfflag(i+1).ne.1).and.
     &          (ilorfflag(i+1).ne.1).and.
     &          (acclinkflag(i+1).ne.1).and.
     &          (branchflag(i+1).eq.1))then
c
c*****************************************************************72
c
c Check if end leak or air pocket present at end of vertical branch
c
                        if((branchendleak(i+1).eq.1).and.
     &                 (horzbflag(i+1).eq.0))then
c
c Calculating for vertical generator, leak or air standpipe situation
c
                  h(i+1,1)=h(i,2)
                          qbranchold(i+1)=qguessbranch(i+1)
                          endflow=.true.
c
```



forward BSOLVER.txt - Notepad

File Edit Format View Help

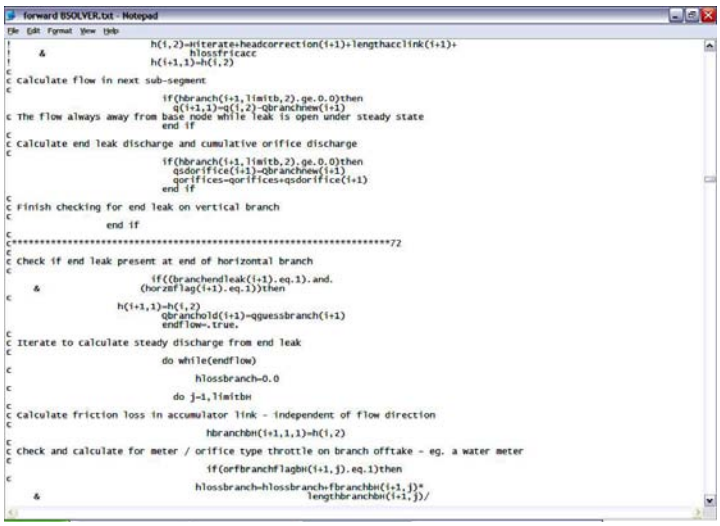
```
c
c Iterate to calculate steady discharge from end leak
c
                              do while(endflow)
c
                                   hlossbranch=0.0
c
                                 do j=1,limitb
c
c Calculate friction loss in accumulator link - independent of flow direction
c
                                   hbranch(i+1,1,1)=h(i,2)
c
c Calculate in-line branch orifice head loss if present at j=2 for
c limitb<=6 type problem only.The value for j=2 is actually set at
c the end of the j=1 section
c
                                   if((j.eq.1).and.(orfbranchflag(i+1).eq.1))then
c
                                  hlossbranch=hlossbranch+fbranch(i+1,j)*
     &                                        lengthbranch(i+1,j)/dbranch(i+1,j)*
     &                                        (qbranchold(i+1)**2.0d0)/
     &                                  (2.0d0*g*(areabranch(i+1,j)**
     &                                        2.0d0))+lengthbranch(i+1,j) ! vertical rise here
c
                                   hbranch(i+1,j,2)=hbranch(i+1,1,1)-
     &                                              headcorrection(i+1)-
     &                                                hlossbranch
c
c Reset hbranch at top of j=1 or base of j=2 sub-segment for orifice loss
c
                                   hbranch(i+1,j+1,1)=hbranch(i+1,j,2)-
     &                                                 (((qbranchold(i+1)/
     &                                                 (corfbranch(i+1)*
     &                                                 areaorfbranch(i+1)))**2.0)/
     &                                                 (2.0*g))
c
c Updating cumulative head loss (ie. hlossbranch) for orifice loss
c
                                       hlossbranch=hlossbranch+(((qbranchold(i+1)/
     &                                             (corfbranch(i+1)*
     &                                             areaorfbranch(i+1)))**2.0)
     &                                             /(2.0*g))
c
                                   else
c
c Not at first node up from base or no base orifice present
c
                                  hlossbranch=hlossbranch+fbranch(i+1,j)*
     &                                        lengthbranch(i+1,j)/dbranch(i+1,j)*
     &                                        (qbranchold(i+1)**2.0d0)/
```



forward BSOLVER.txt - Notepad

File Edit Format View Help

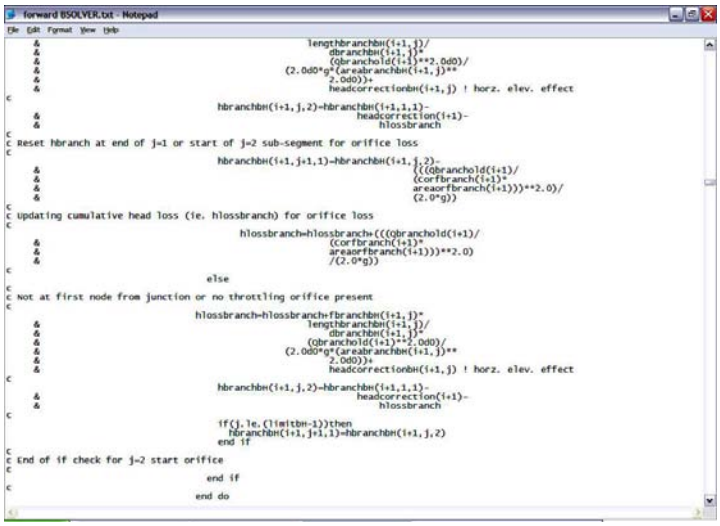
```
     &                                        (qbranchold(i+1)**2.0d0)/
     &                                  (2.0d0*g*(areabranch(i+1,j)**
     &                                        2.0d0))+lengthbranch(i+1,j) ! vertical rise here
c
                                   hbranch(i+1,j,2)=hbranch(i+1,1,1)-
     &                                              headcorrection(i+1)-
     &                                                hlossbranch
c
                                   if(j.le.(limitb-1))then
                                     hbranch(i+1,j+1,1)=hbranch(i+1,j,2)
                                   end if
c
c End of if check for j=2 base orifice
c
                                   end if
c
                                 end do
c
c Calculate new qiterate on basis on hiterate
c
                               if(hbranch(i+1,limitb,2).ge.0.0)then
                                 qbranchnew(i+1)=cveborf(i+1)*
     &                                           (sqrt(2.0d0*g*
     &                                           (hbranch(i+1,limitb,2))))
                               else
                                 write(*,*) 'Head at end of branch is negative V'
                                 stop
!                                qbranchnew(i+1)=0.0d0
                               end if
c
c Update qiterate or stop iteration if convergence
c
                               deltaqbranch(i+1)=abs(qbranchnew(i+1)-
     &                                             qbranchold(i+1))
c
                               if(deltaqbranch(i+1).lt.1.0d-12)then ! convergence
                                     do j=1,limitb
                                       qbranch(i+1,j,1)=qbranchnew(i+1)
                                       qbranch(i+1,j,2)=qbranchnew(i+1)
                                     end do
                                 endflow=.false.
                               else
                                 endflow=.true.
                                 qbranchold(i+1)=qbranchnew(i+1)
                               end if
c
                              end do
c
c Specify correct head at base of standpipe
c
!                              h(i,2)=hiterate+headcorrection(i+1)+lengthacclink(i+1)+
```



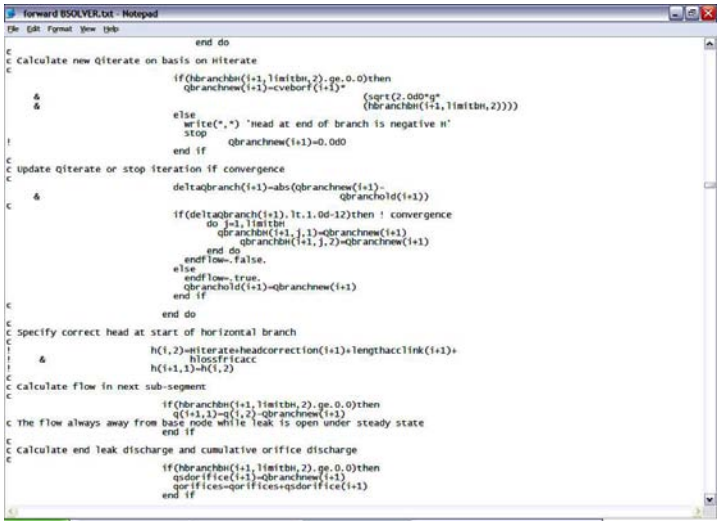
forward BSOLVER.txt - Notepad

File Edit Format View Help

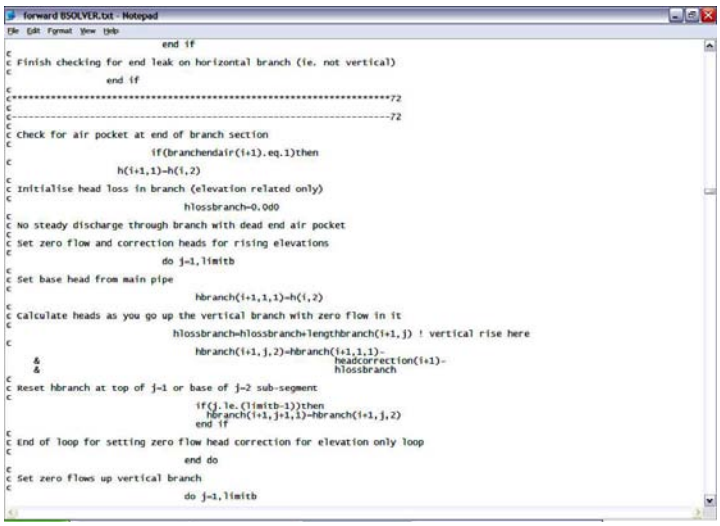
```
!                              h(i,2)=hiterate+headcorrection(i+1)+lengthacclink(i+1)+
!     &                               hlossfricacc
!                              h(i+1,1)=h(i,2)
c
c Calculate flow in next sub-segment
c
                               if(hbranch(i+1,limitb,2).ge.0.0)then
                                 q(i+1,1)=q(i,2)-qbranchnew(i+1)
c The flow always away from base node while leak is open under steady state
                               end if
c
c Calculate end leak discharge and cumulative orifice discharge
c
                               if(hbranch(i+1,limitb,2).ge.0.0)then
                                 qsdorifice(i+1)=qbranchnew(i+1)
                                 qorifices=qorifices+qsdorifice(i+1)
                               end if
c
c Finish checking for end leak on vertical branch
c
                            end if
c
c*****************************************************************72
c
c Check if end leak present at end of horizontal branch
c
                           if((branchendleak(i+1).eq.1).and.
     &                        (horzmflag(i+1).eq.1))then
c
                         h(i+1,1)=h(i,2)
                           qbranchold(i+1)=qguessbranch(i+1)
                           endflow=.true.
c
c Iterate to calculate steady discharge from end leak
c
                           do while(endflow)
c
                                hlossbranch=0.0
c
                             do j=1,limitbH
c
c Calculate friction loss in accumulator link - independent of flow direction
c
                                hbranchbH(i+1,1,1)=h(i,2)
c
c Check and calculate for meter / orifice type throttle on branch offtake - eg. a water meter
c
                                if(orfbranchflagbH(i+1,j).eq.1)then
c
                               hlossbranch=hlossbranch+fbranchbH(i+1,j)*
     &                                     lengthbranchbH(i+1,j)/
```



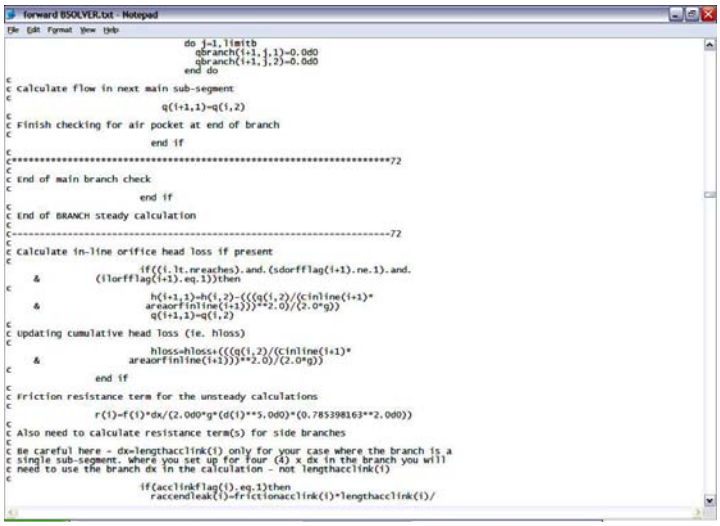
```
forward BSOLVER.txt - Notepad
File  Edit  Format  View  Help
     &                                                lengthbranchbH(i+1,j)/
     &                                                  dbranchbH(i+1,j)*
     &                                                  (qbranchold(i+1)**2.0d0)/
     &                                              (2.0d0*g*(areabranchbH(i+1,j)**
     &                                                2.0d0))+
     &                                                headcorrectionbH(i+1,j) ! horz. elev. effect
c
                                    hbranchbH(i+1,j,2)=hbranchbH(i+1,1,1)-
     &                                                  headcorrection(i+1)-
     &                                                    hlossbranch
c
c Reset hbranch at end of j=1 or start of j=2 sub-segment for orifice loss
c
                                    hbranchbH(i+1,j+1,1)=hbranchbH(i+1,j,2)-
     &                                                  (((qbranchold(i+1)/
     &                                                  (corfbranch(i+1)*
     &                                                  areaorfbranch(i+1)))**2.0)/
     &                                                  (2.0*g))
c
c Updating cumulative head loss (ie. hlossbranch) for orifice loss
c
                                        hlossbranch=hlossbranch+(((qbranchold(i+1)/
     &                                                  (corfbranch(i+1)*
     &                                                  areaorfbranch(i+1)))**2.0)
     &                                                  /(2.0*g))
c
                                    else
c
c Not at first node from junction or no throttling orifice present
c
                                    hlossbranch=hlossbranch+fbranchbH(i+1,j)*
     &                                                lengthbranchbH(i+1,j)/
     &                                                  dbranchbH(i+1,j)*
     &                                                (qbranchold(i+1)**2.0d0)/
     &                                              (2.0d0*g*(areabranchbH(i+1,j)**
     &                                                2.0d0))+
     &                                                headcorrectionbH(i+1,j) ! horz. elev. effect
c
                                    hbranchbH(i+1,j,2)=hbranchbH(i+1,1,1)-
     &                                                  headcorrection(i+1)-
     &                                                    hlossbranch
c
                                     if(j.le.(limitbH-1))then
                                       hbranchbH(i+1,j+1,1)=hbranchbH(i+1,j,2)
                                     end if
c
c End of if check for j=2 start orifice
c
                                      end if
c
                                    end do
```



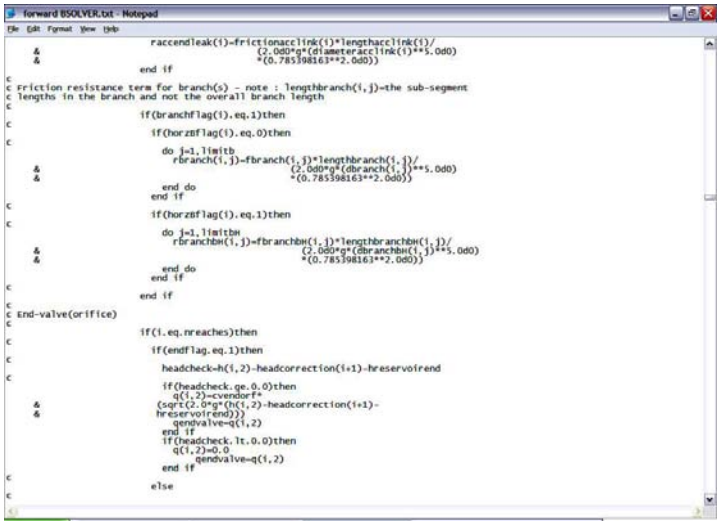
```
forward BSOLVER.txt - Notepad
File  Edit  Format  View  Help
                                    end do
c
c Calculate new Qiterate on basis on Hiterate
c
                                   if(hbranchbH(i+1,limitbH,2).ge.0.0)then
                                     qbranchnew(i+1)=cvebort(i+1)*
     &                                                  (sqrt(2.0d0*g*
     &                                                  (hbranchbH(i+1,limitbH,2))))
                                   else
                                     write(*,*) 'Head at end of branch is negative H'
                                     stop
!                                          qbranchnew(i+1)=0.0d0
                                   end if
c
c Update Qiterate or stop iteration if convergence
c
                                   deltaqbranch(i+1)=abs(qbranchnew(i+1)-
     &                                                  qbranchold(i+1))
c
                                   if(deltaqbranch(i+1).lt.1.0d-12)then ! convergence
                                       do j=1,limitbH
                                         qbranchbH(i+1,j,1)=qbranchnew(i+1)
                                         qbranchbH(i+1,j,2)=qbranchnew(i+1)
                                       end do
                                     endflow=.false.
                                   else
                                     endflow=.true.
                                     qbranchold(i+1)=qbranchnew(i+1)
                                   end if
c
                                  end do
c
c Specify correct head at start of horizontal branch
c
!                                 h(i,2)=Hiterate+headcorrection(i+1)+lengthacclink(i+1)+
!     &                             hlossfricacc
!                                 h(i+1,1)=h(i,2)
c
c Calculate flow in next sub-segment
c
                                   if(hbranchbH(i+1,limitbH,2).ge.0.0)then
                                     q(i+1,1)=q(i,2)-qbranchnew(i+1)
c The flow always away from base node while leak is open under steady state
                                   end if
c
c Calculate end leak discharge and cumulative orifice discharge
c
                                   if(hbranchbH(i+1,limitbH,2).ge.0.0)then
                                     qsdorifice(i+1)=qbranchnew(i+1)
                                     qorifices=qorifices+qsdorifice(i+1)
                                   end if
```



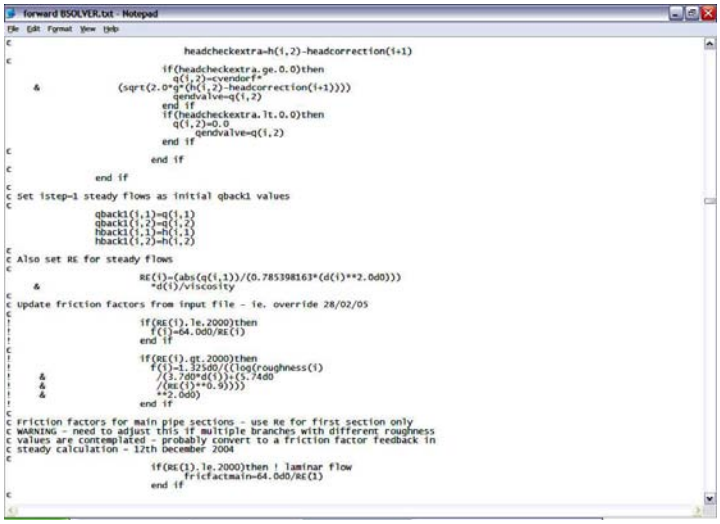
```
forward BSOLVER.txt - Notepad
File  Edit  Format  View  Help
                                 end if
c
c Finish checking for end leak on horizontal branch (ie. not vertical)
c
                               end if
c
c*********************************************************************72
c
c---------------------------------------------------------------------72
c
c Check for air pocket at end of branch section
c
                              if(branchendair(i+1).eq.1)then
c
                            h(i+1,1)=h(i,2)
c
c Initialise head loss in branch (elevation related only)
c
                                  hlossbranch=0.0d0
c
c No steady discharge through branch with dead end air pocket
c
c Set zero flow and correction heads for rising elevations
c
                                  do j=1,limitb
c
c Set base head from main pipe
c
                                    hbranch(i+1,1,1)=h(i,2)
c
c Calculate heads as you go up the vertical branch with zero flow in it
c
                                  hlossbranch=hlossbranch+lengthbranch(i+1,j) ! vertical rise here
c
                                    hbranch(i+1,j,2)=hbranch(i+1,1,1)-
     &                                                  headcorrection(i+1)-
     &                                                  hlossbranch
c
c Reset hbranch at top of j=1 or base of j=2 sub-segment
c
                                     if(j.le.(limitb-1))then
                                       hbranch(i+1,j+1,1)=hbranch(i+1,j,2)
                                     end if
c
c End of loop for setting zero flow head correction for elevation only loop
c
                                   end do
c
c Set zero flows up vertical branch
c
                                   do j=1,limitb
```



```
forward BSOLVER.txt - Notepad
File  Edit  Format  View  Help
                                do j=1,limitb
                                  qbranch(i+1,j,1)=0.0d0
                                  qbranch(i+1,j,2)=0.0d0
                                end do
c
c calculate flow in next main sub-segment
c
                                q(i+1,1)=q(i,2)
c
c Finish checking for air pocket at end of branch
c
                                end if
c
c*******************************************************************72
c
c End of main branch check
c
                              end if
c
c End of BRANCH steady calculation
c
c-----------------------------------------------------------------72
c
c Calculate in-line orifice head loss if present
c
                                if((i.lt.nreaches).and.(sdorfflag(i+1).ne.1).and.
     &          (ilorfflag(i+1).eq.1))then
c
                                  h(i+1,1)=h(i,2)-(((q(i,2)/(cinline(i+1)*
     &                            areaorfinline(i+1)))**2.0)/(2.0*g))
                                  q(i+1,1)=q(i,2)
c
c Updating cumulative head loss (ie. hloss)
c
                                  hloss=hloss+(((q(i,2)/(cinline(i+1)*
     &                          areaorfinline(i+1)))**2.0)/(2.0*g))
c
                              end if
c
c Friction resistance term for the unsteady calculations
c
                  r(i)=f(i)*dx/(2.0d0*g*(d(i)**5.0d0)*(0.785398163**2.0d0))
c
c Also need to calculate resistance term(s) for side branches
c
c Be careful here - dx=lengthacclink(i) only for your case where the branch is a
c single sub-segment. Where you set up for four (4) x dx in the branch you will
c need to use the branch dx in the calculation - not lengthacclink(i)
c
                              if(acclinkflag(i).eq.1)then
                                raccendleak(i)=frictionacclink(i)*lengthacclink(i)/
```

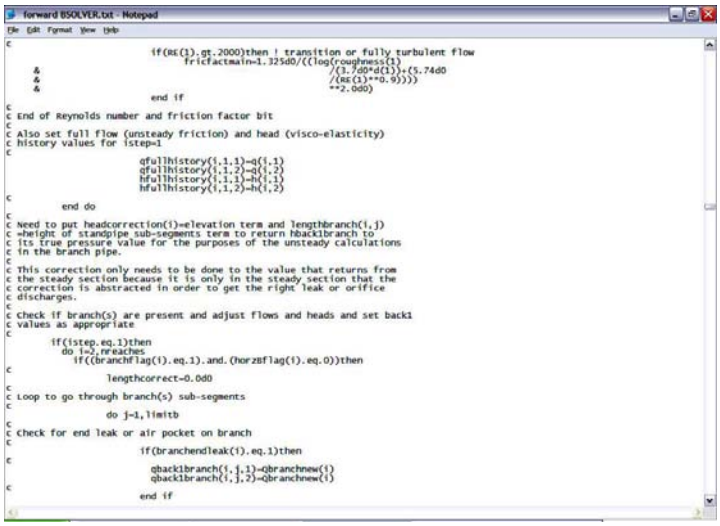


```
forward BSOLVER.txt - Notepad
File  Edit  Format  View  Help
                                raccendleak(i)=frictionacclink(i)*lengthacclink(i)/
     &                                         (2.0d0*g*(diameteracclink(i)**5.0d0)
     &                                         *(0.785398163**2.0d0))
                              end if
c
c Friction resistance term for branch(s) - note : lengthbranch(i,j)=the sub-segment
c lengths in the branch and not the overall branch length
c
                              if(branchflag(i).eq.1)then
c
                                if(horzbflag(i).eq.0)then
c
                                  do j=1,limitb
                                    rbranch(i,j)=fbranch(i,j)*lengthbranch(i,j)/
     &                                           (2.0d0*g*(dbranch(i,j)**5.0d0)
     &                                           *(0.785398163**2.0d0))
                                  end do
                                end if
c
                                if(horzbflag(i).eq.1)then
c
                                  do j=1,limitbH
                                    rbranchbH(i,j)=fbranchbH(i,j)*lengthbranchbH(i,j)/
     &                                             (2.0d0*g*(dbranchbH(i,j)**5.0d0)
     &                                             *(0.785398163**2.0d0))
                                  end do
                                end if
c
                              end if
c
c End-valve(orifice)
c
                              if(i.eq.nreaches)then
c
                                if(endflag.eq.1)then
c
                                  headcheck=h(i,2)-headcorrection(i+1)-hreservoirend
c
                                  if(headcheck.ge.0.0)then
                                    q(i,2)=cvendorf*
     &                            (sqrt(2.0*g*(h(i,2)-headcorrection(i+1)-
     &                            hreservoirend)))
                                    qendvalve=q(i,2)
                                  end if
                                  if(headcheck.lt.0.0)then
                                    q(i,2)=0.0
                                    qendvalve=q(i,2)
                                  end if
c
                                else
c
```

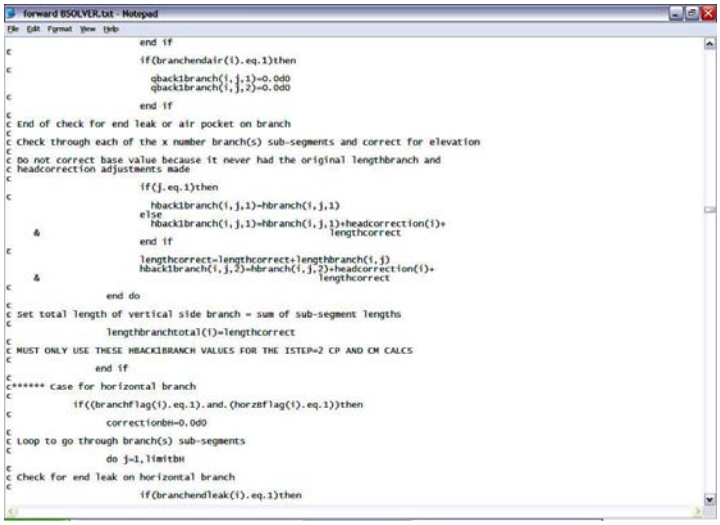


```
forward BSOLVER.txt - Notepad
File  Edit  Format  View  Help
c
                                    headcheckextra=h(i,2)-headcorrection(i+1)
c
                                  if(headcheckextra.ge.0.0)then
                                    q(i,2)=cvendorf*
     &                (sqrt(2.0*g*(h(i,2)-headcorrection(i+1))))
                                    qendvalve=q(i,2)
                                  end if
                                  if(headcheckextra.lt.0.0)then
                                    q(i,2)=0.0
                                    qendvalve=q(i,2)
                                  end if
c
                                end if
c
                              end if
c
c Set istep=1 steady flows as initial qback1 values
c
                  qback1(i,1)=q(i,1)
                  qback1(i,2)=q(i,2)
                  hback1(i,1)=h(i,1)
                  hback1(i,2)=h(i,2)
c
c Also set RE for steady flows
c
                              RE(i)=(abs(q(i,1))/(0.785398163*(d(i)**2.0d0)))
     &                        *d(i)/viscosity
c
c Update friction factors from input file - ie. override 28/02/05
c
!                             if(RE(i).le.2000)then
!                               f(i)=64.0d0/RE(i)
!                             end if
!
!                             if(RE(i).gt.2000)then
!                               f(i)=1.325d0/((log(roughness(i)
!    &                            /(3.7d0*d(i)))+(5.74d0
!    &                            /(RE(i)**0.9)))
!    &                            **2.0d0)
!                             end if
c
c Friction factors for main pipe sections - use Re for first section only
c WARNING - need to adjust this if multiple branches with different roughness
c values are contemplated - probably convert to a friction factor feedback in
c steady calculation - 12th December 2004
c
                                if(RE(1).le.2000)then ! laminar flow
                                    fricfactmain=64.0d0/RE(1)
                                end if
c
```

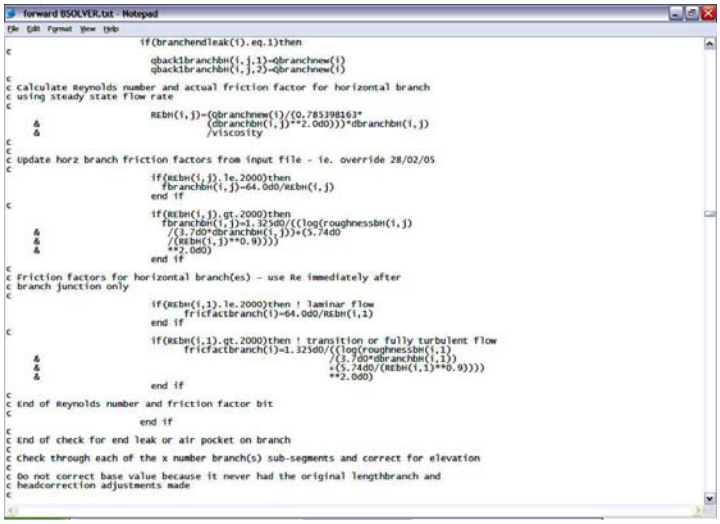
Appendix M – Fortran code for BSOLVER and NLFIT subroutines


forward BSOLVER.txt - Notepad

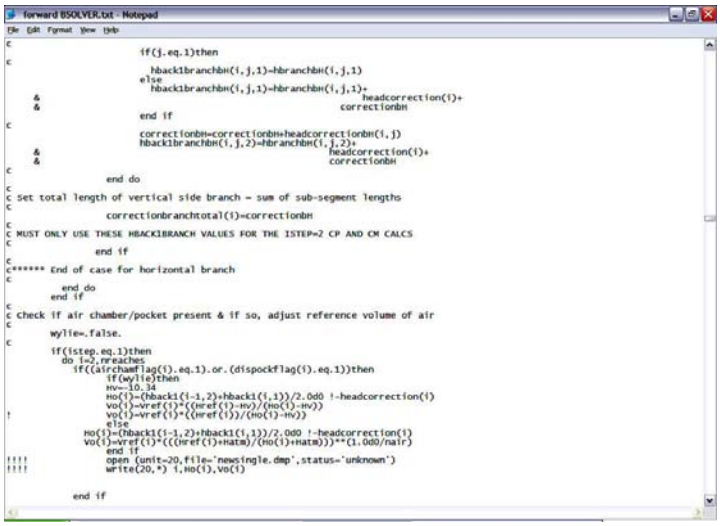
```
File Edit Format View Help
C                         if(RE(1).gt.2000)then ! transition or fully turbulent flow
                            fricfactmain=1.325d0/((log(roughness(1)
     &                                              /(3.7d0*d(1))+(5.74d0
     &                                              /(RE(1)**0.9))))
     &                                              **2.0d0)
                          end if
C
C End of Reynolds number and friction factor bit
C
C Also set full flow (unsteady friction) and head (visco-elasticity)
C history values for istep=1
C
                          qfullhistory(i,1,1)=q(i,1)
                          qfullhistory(i,1,2)=q(i,2)
                          hfullhistory(i,1,1)=h(i,1)
                          hfullhistory(i,1,2)=h(i,2)
C
        end do
C
C Need to put headcorrection(i)=elevation term and lengthbranch(i,j)
C =height of standpipe sub-segments term to return hback1branch to
C its true pressure value for the purposes of the unsteady calculations
C in the branch pipe.
C
C This correction only needs to be done to the value that returns from
C the steady section because it is only in the steady section that the
C correction is abstracted in order to get the right leak or orifice
C discharges.
C
C Check if branch(s) are present and adjust flows and heads and set back1
C values as appropriate
C
       if(istep.eq.1)then
         do i=2,nreaches
           if((branchflag(i).eq.1).and.(horzbflag(i).eq.0))then
C
                  lengthcorrect=0.0d0
C
C Loop to go through branch(s) sub-segments
C
                  do j=1,limitb
C
C Check for end leak or air pocket on branch
C
                    if(branchendleak(i).eq.1)then
C
                      qback1branch(i,j,1)=qbranchnew(i)
                      qback1branch(i,j,2)=qbranchnew(i)
C
                    end if
```


forward BSOLVER.txt - Notepad

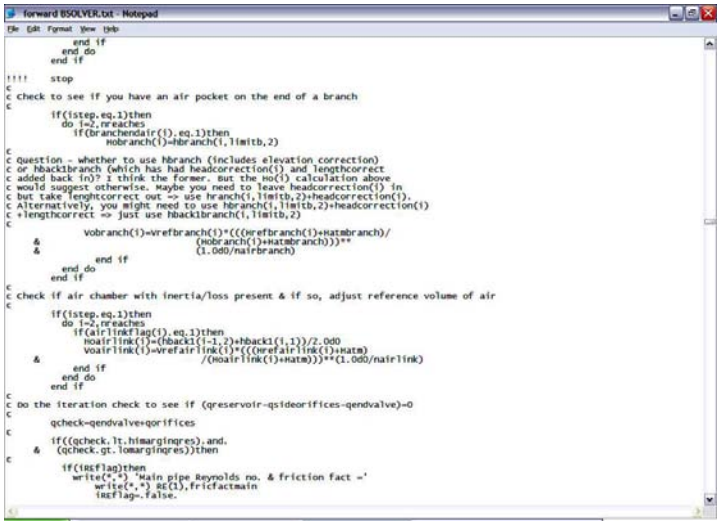
```
File Edit Format View Help
                    end if
C
                    if(branchendair(i).eq.1)then
C
                      qback1branch(i,j,1)=0.0d0
                      qback1branch(i,j,2)=0.0d0
C
                    end if
C
C End of check for end leak or air pocket on branch
C
C Check through each of the x number branch(s) sub-segments and correct for elevation
C
C Do not correct base value because it never had the original lengthbranch and
C headcorrection adjustments made
C
                    if(j.eq.1)then
C
                      hback1branch(i,j,1)=hbranch(i,j,1)
                    else
                      hback1branch(i,j,1)=hbranch(i,j,1)+headcorrection(i)+
     &                                                   lengthcorrect
                    end if
C
                    lengthcorrect=lengthcorrect+lengthbranch(i,j)
                    hback1branch(i,j,2)=hbranch(i,j,2)+headcorrection(i)+
     &                                                 lengthcorrect
C
                  end do
C
C Set total length of vertical side branch = sum of sub-segment lengths
C
                  lengthbranchtotal(i)=lengthcorrect
C
C MUST ONLY USE THESE HBACK1BRANCH VALUES FOR THE ISTEP=2 CP AND CM CALCS
C
                end if
C
C****** case for horizontal branch
C
          if((branchflag(i).eq.1).and.(horzbflag(i).eq.1))then
C
                  correctionbh=0.0d0
C
C Loop to go through branch(s) sub-segments
C
                  do j=1,limitbh
C
C Check for end leak on horizontal branch
C
                    if(branchendleak(i).eq.1)then
```


forward BSOLVER.txt - Notepad

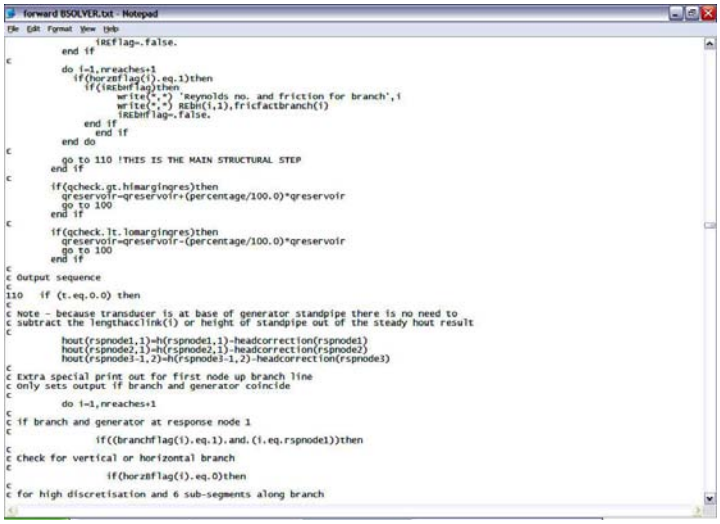
```
File Edit Format View Help
                    if(branchendleak(i).eq.1)then
C
                      qback1branchbh(i,j,1)=qbranchnew(i)
                      qback1branchbh(i,j,2)=qbranchnew(i)
C
C Calculate Reynolds number and actual friction factor for horizontal branch
C using steady state flow rate
C
                      REbh(i,j)=(qbranchnew(i)/(0.785398163*
     &                          (dbranchbh(i,j))**2.0d0)))*dbranchbh(i,j)
     &                          /viscosity
C
C
C Update horz branch friction factors from input file - ie. override 28/02/05
C
                      if(REbh(i,j).le.2000)then
                        fbranchbh(i,j)=64.0d0/REbh(i,j)
                      end if
C
                      if(REbh(i,j).gt.2000)then
                        fbranchbh(i,j)=1.325d0/((log(roughnessbh(i,j)
     &                    /(3.7d0*dbranchbh(i,j)))+(5.74d0
     &                    /(REbh(i,j)**0.9))))
     &                    **2.0d0)
                      end if
C
C Friction factors for horizontal branch(es) - use Re immediately after
C branch junction only
C
                      if(REbh(i,1).le.2000)then ! laminar flow
                        fricfactbranch(i)=64.0d0/REbh(i,1)
                      end if
C
                      if(REbh(i,1).gt.2000)then ! transition or fully turbulent flow
                        fricfactbranch(i)=1.325d0/((log(roughnessbh(i,1)
     &                                              /(3.7d0*dbranchbh(i,1))
     &                                              +(5.74d0/(REbh(i,1)**0.9))))
     &                                              **2.0d0)
                      end if
C
C End of Reynolds number and friction factor bit
C
                    end if
C
C End of check for end leak or air pocket on branch
C
C Check through each of the x number branch(s) sub-segments and correct for elevation
C
C Do not correct base value because it never had the original lengthbranch and
C headcorrection adjustments made
C
```



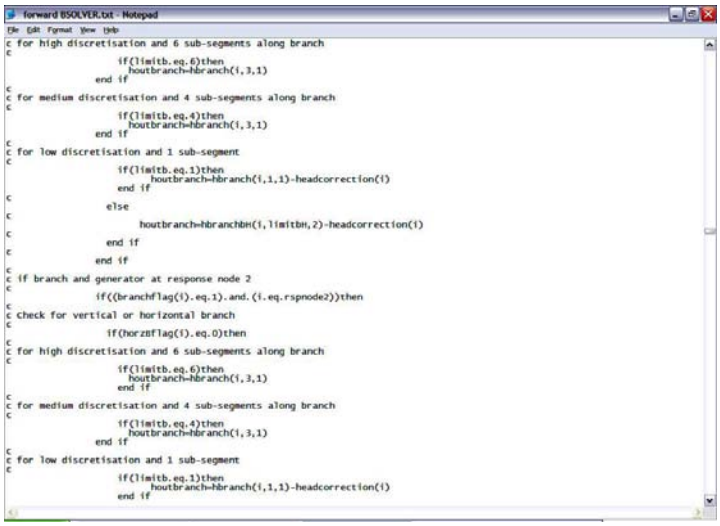

```
c                       if(j.eq.1)then
c                         hback1branchbh(i,j,1)=hbranchbh(i,j,1)
                        else
                          hback1branchbh(i,j,1)=hbranchbh(i,j,1)+
     &                                          headcorrection(i)+
     &                                     correctionbh
                        end if
c
                        correctionbh=correctionbh+headcorrectionbh(i,j)
                        hback1branchbh(i,j,2)=hbranchbh(i,j,2)+
     &                                     headcorrection(i)+
     &                                     correctionbh
c
                  end do
c
c Set total length of vertical side branch = sum of sub-segment lengths
c
                  correctionbranchtotal(i)=correctionbh
c
c MUST ONLY USE THESE HBACK1BRANCH VALUES FOR THE ISTEP=2 CP AND CM CALCS
c
                end if
c
c****** End of case for horizontal branch
c
          end do
        end if
c
c Check if air chamber/pocket present & if so, adjust reference volume of air
c
        wylie=.false.
c
        if(istep.eq.1)then
          do i=2,nreaches
            if((airchamflag(i).eq.1).or.(dispockflag(i).eq.1))then
                if(wylie)then
                mv=-10.34
                Ho(i)=(hback1(i-1,2)+hback1(i,1))/2.0d0 !-headcorrection(i)
                Vo(i)=Vref(i)*((Href(i)-Hv)/(Ho(i)-Hv))
!               Vo(i)=Vref(i)*((Href(i))/(Ho(i)-Hv))
                else
              Ho(i)=(hback1(i-1,2)+hback1(i,1))/2.0d0 !-headcorrection(i)
              Vo(i)=Vref(i)*(((Href(i)+Hatm)/(Ho(i)+Hatm)))**(1.0d0/nair)
                end if
!!!!            open (unit=20,file='newsingle.dmp',status='unknown')
!!!!            write(20,*) i,Ho(i),Vo(i)

            end if
```



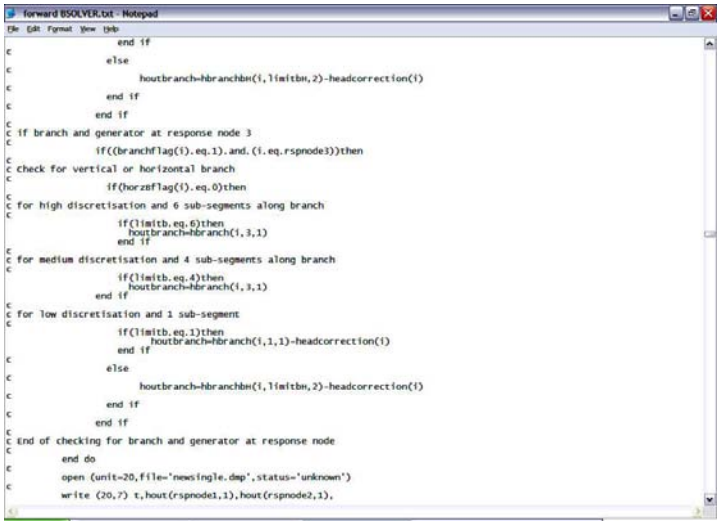

```
          end if
        end do
      end if

!!!!    stop
c
c Check to see if you have an air pocket on the end of a branch
c
        if(istep.eq.1)then
          do i=2,nreaches
            if(branchendair(i).eq.1)then
                Hobranch(i)=Hbranch(i,limitb,2)
c
c Question - whether to use Hbranch (includes elevation correction)
c or Hback1branch (which has had headcorrection(i) and lengthcorrect
c added back in)? I think the former. But the Ho(i) calculation above
c would suggest otherwise. Maybe you need to leave headcorrection(i) in
c but take lenghtcorrect out => use branch(i,limitb,2)+headcorrection(i).
c Alternatively, you might need to use Hbranch(i,limitb,2)+headcorrection(i)
c +lengthcorrect => just use hback1branch(i,limitb,2)
c
                Vobranch(i)=Vrefbranch(i)*(((Hrefbranch(i)+Hatmbranch)/
     &                          (Hobranch(i)+Hatmbranch)))**
     &                          (1.0d0/nairbranch)
              end if
          end do
        end if
c
c Check if air chamber with inertia/loss present & if so, adjust reference volume of air
c
        if(istep.eq.1)then
          do i=2,nreaches
            if(airlinkflag(i).eq.1)then
              Hoairlink(i)=(hback1(i-1,2)+hback1(i,1))/2.0d0
              Voairlink(i)=Vrefairlink(i)*(((Hrefairlink(i)+Hatm)
     &                          /(Hoairlink(i)+Hatm)))**(1.0d0/nairlink)
            end if
          end do
        end if
c
c Do the iteration check to see if (qreservoir-qsideorifices-qendvalve)=0
c
        qcheck=qendvalve+qorifices
c
        if((qcheck.lt.himargingres).and.
     &   (qcheck.gt.lomargingres))then
c
          if(iRsflag)then
            write(*,*) 'Main pipe Reynolds no. & friction fact ='
                write(*,*) RE(1),fricfactmain
                iRsflag=.false.
```



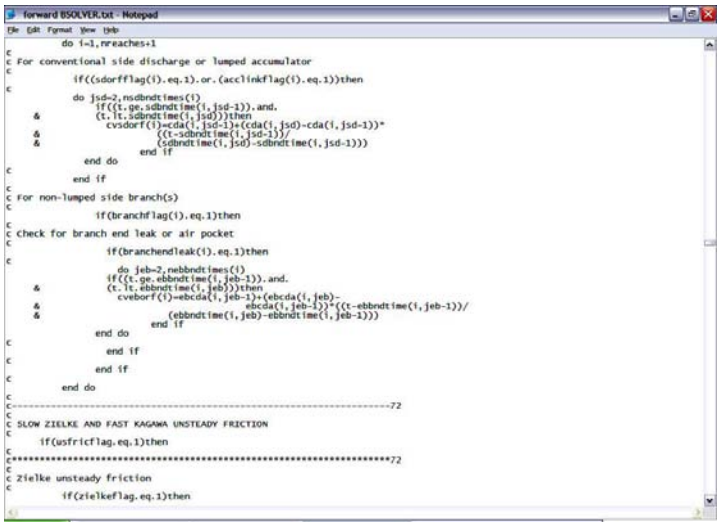

```
                iRsflag=.false.
          end if
c
          do i=1,nreaches+1
            if(horzBflag(i).eq.1)then
              if(iREBHflag)then
                    write(*,*) 'Reynolds no. and friction for branch',i
                    write(*,*) REBh(i,1),fricfactbranch(i)
                    iREBHflag=.false.
              end if
                end if
          end do
c
          go to 110 !THIS IS THE MAIN STRUCTURAL STEP
        end if
c
        if(qcheck.gt.himargingres)then
          qreservoir=qreservoir-(percentage/100.0)*qreservoir
          go to 100
        end if
c
        if(qcheck.lt.lomargingres)then
          qreservoir=qreservoir-(percentage/100.0)*qreservoir
          go to 100
        end if
c
c Output sequence
c
110   if (t.eq.0.0) then
c
c Note - because transducer is at base of generator standpipe there is no need to
c subtract the lengthacclink(i) or height of standpipe out of the steady hout result
c
          hout(rspnode1,1)=h(rspnode1,1)-headcorrection(rspnode1)
          hout(rspnode2,1)=h(rspnode2,1)-headcorrection(rspnode2)
          hout(rspnode3-1,2)=h(rspnode3-1,2)-headcorrection(rspnode3)
c
c Extra special print out for first node up branch line
c only sets output if branch and generator coincide
c
          do i=1,nreaches+1
c
c if branch and generator at response node 1
c
                if((branchflag(i).eq.1).and.(i.eq.rspnode1))then
c
c Check for vertical or horizontal branch
c
                  if(horzBflag(i).eq.0)then
c
c for high discretisation and 6 sub-segments along branch
```



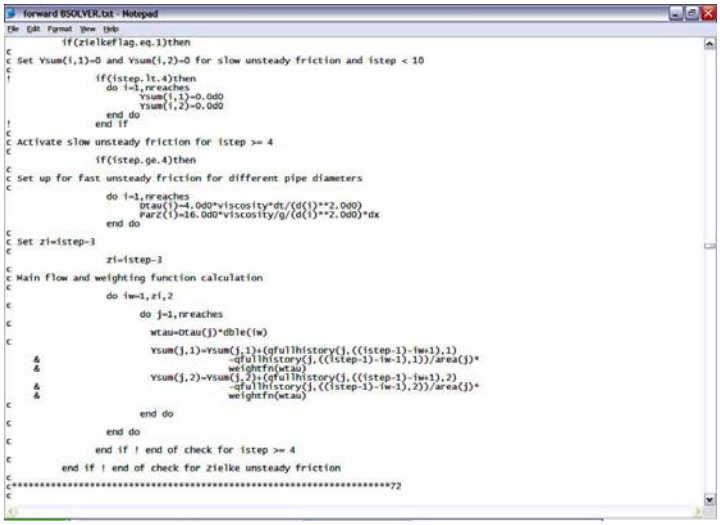
```
forward BSOLVER.txt - Notepad
File  Edit  Format  View  Help
c for high discretisation and 6 sub-segments along branch
c                     if(limitb.eq.6)then
                        houtbranch=hbranch(i,3,1)
                    end if
c
c for medium discretisation and 4 sub-segments along branch
c                     if(limitb.eq.4)then
                        houtbranch=hbranch(i,3,1)
                    end if
c
c for low discretisation and 1 sub-segment
c                     if(limitb.eq.1)then
                        houtbranch=hbranch(i,1,1)-headcorrection(i)
                      end if
c                   else
c                         houtbranch=hbranchbn(i,limitbn,2)-headcorrection(i)
c                     end if
c                   end if
c
c if branch and generator at response node 2
c                   if((branchflag(i).eq.1).and.(i.eq.rspnode2))then
c
c check for vertical or horizontal branch
c                     if(horzbflag(i).eq.0)then
c
c for high discretisation and 6 sub-segments along branch
c                     if(limitb.eq.6)then
                        houtbranch=hbranch(i,3,1)
                      end if
c
c for medium discretisation and 4 sub-segments along branch
c                     if(limitb.eq.4)then
                        houtbranch=hbranch(i,3,1)
                    end if
c
c for low discretisation and 1 sub-segment
c                     if(limitb.eq.1)then
                        houtbranch=hbranch(i,1,1)-headcorrection(i)
                      end if
c                   else
c                         houtbranch=hbranchbn(i,limitbn,2)-headcorrection(i)
c                     end if
c                   end if
c
c if branch and generator at response node 3
c                   if((branchflag(i).eq.1).and.(i.eq.rspnode3))then
c
c check for vertical or horizontal branch
c                     if(horzbflag(i).eq.0)then
c
c for high discretisation and 6 sub-segments along branch
c                     if(limitb.eq.6)then
                        houtbranch=hbranch(i,3,1)
                      end if
c
c for medium discretisation and 4 sub-segments along branch
c                     if(limitb.eq.4)then
                        houtbranch=hbranch(i,3,1)
                    end if
c
c for low discretisation and 1 sub-segment
c                     if(limitb.eq.1)then
                        houtbranch=hbranch(i,1,1)-headcorrection(i)
                      end if
c                   else
c                         houtbranch=hbranchbn(i,limitbn,2)-headcorrection(i)
c                     end if
c                   end if
c
c End of checking for branch and generator at response node
c
          end do
c
          open (unit=20,file='newsingle.dmp',status='unknown')
c
          write (20,7) t,hout(rspnode1,1),hout(rspnode2,1),
```



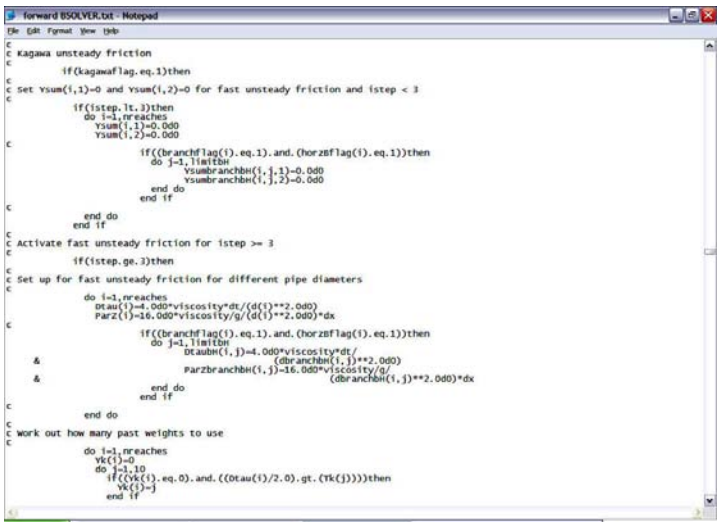
```
forward BSOLVER.txt - Notepad
File  Edit  Format  View  Help
                      end if
c                   else
c                         houtbranch=hbranchbn(i,limitbn,2)-headcorrection(i)
c                     end if
c                   end if
c
c End of checking for branch and generator at response node
c
          end do
c
          open (unit=20,file='newsingle.dmp',status='unknown')
c
          write (20,7) t,hout(rspnode1,1),hout(rspnode2,1),
     &                   hout(rspnode3-1,2),houtbranch
!!       write (20,8) qreservoir,qbranchnew(37),qsdorifice(102),
!!     &                          qsdorifice(ipick),qendvalve,cda(ipick,1),RE(1),
!!     &                                  fricfactmain
          write (*,*)  qsdorifice(659)
          write (*,*)  qreservoir-qsdorifice(659)
          write (*,*)  qorifices
          write (*,*)  qcheck-qorifices
!         write (*,*) RE(10),RE(260),RE(305)
!         write (*,*) f(10),f(260),f(305)
!         leaksize=qorifices-qsdorifice(441)
!          write (*,*) leaksize
!
!!        do i=1,nreaches
!!        write (*,*) i,f(i)
!!        end do
!
! special little harcode output for Hanson leak calcs. above - 84.4m dx only
!
7         format (f20.12,1x,f20.12,1x,f20.12,1x,f20.12,1x,f20.12,1x,f20.12)
8          format (f20.12,1x,f20.12,1x,f20.12,1x,f20.12,1x,f20.12,1x,f20.12,1x,
     &                f20.12,1x,f20.12,1x,f20.12)
           ju=ju+dtmultiplier*dt
!          do i=1,nreaches+1
!                write (20,*) i,f(i)
!          end do
          end if
c
c END OF STEADY STATE FLOWS,HEADS AND LOSSES
c
!         stop
c
c-------------------------------------------------------------------------72
c
c START THE TIME ITERATION FOR UNSTEADY CALCULATIONS
c
c Increment time and set response point variables
c
200       t=t+dt
c
          istep=istep+1
c
          jua=ju-1.0d-11*ju
          jub=ju+1.0d-11*ju
          if (i.gt.ifinal) go to 300
c
c Closure position interpolator for side orifices and accumulator
c end leaks (not at first and last nodes)
c
            do i=1,nreaches+1
```



```
forward BSOLVER.txt - Notepad
File  Edit  Format  View  Help
        do i=1,nreaches+1
C
C For conventional side discharge or lumped accumulator
C
          if((sdorfflag(i).eq.1).or.(acclinkflag(i).eq.1))then
C
          do jsd=2,nsdbndtimes(i)
            if((t.ge.sdbndtime(i,jsd-1)).and.
     &        (t.lt.sdbndtime(i,jsd)))then
              cvsdorf(i)=cda(i,jsd-1)+(cda(i,jsd)-cda(i,jsd-1))*
     &                    ((t-sdbndtime(i,jsd-1))/
     &                     (sdbndtime(i,jsd)-sdbndtime(i,jsd-1)))
                        end if
           end do
C
          end if
C
C For non-lumped side branch(s)
C
            if(branchflag(i).eq.1)then
C
C Check for branch end leak or air pocket
C
              if(branchendleak(i).eq.1)then
C
                do jeb=2,nebbndtimes(i)
                if((t.ge.ebbndtime(i,jeb-1)).and.
     &            (t.lt.ebbndtime(i,jeb)))then
                  cveborf(i)=ebcda(i,jeb-1)+(ebcda(i,jeb)-
     &                       ebcda(i,jeb-1))*((t-ebbndtime(i,jeb-1))/
     &                        (ebbndtime(i,jeb)-ebbndtime(i,jeb-1)))
                            end if
            end do
C
              end if
C
            end if
C
        end do
C
C-------------------------------------------------------------------72
C
C SLOW ZIELKE AND FAST KAGAWA UNSTEADY FRICTION
C
     if(usfricflag.eq.1)then
C
C**********************************************************************72
C
C Zielke unsteady friction
C
        if(zielkeflag.eq.1)then
```

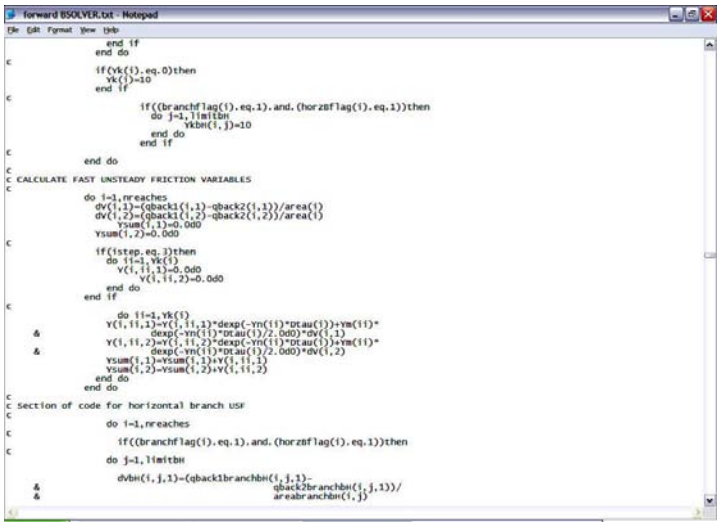


```
forward BSOLVER.txt - Notepad
File  Edit  Format  View  Help
        if(zielkeflag.eq.1)then
C
C Set Ysum(i,1)=0 and Ysum(i,2)=0 for slow unsteady friction and istep < 10
C
          if(istep.lt.4)then
            do i=1,nreaches
                Ysum(i,1)=0.0d0
                Ysum(i,2)=0.0d0
            end do
!         end if
C Activate slow unsteady friction for istep >= 4
C
          if(istep.ge.4)then
C
C Set up for fast unsteady friction for different pipe diameters
C
            do i=1,nreaches
                Dtau(i)=4.0d0*viscosity*dt/(d(i)**2.0d0)
                Parz(i)=16.0d0*viscosity/g/(d(i)**2.0d0)*dx
            end do
C
C Set zi=istep-3
C
            zi=istep-3
C
C Main flow and weighting function calculation
C
            do iw=1,zi,2
C
                do j=1,nreaches
C
                  wtau=Dtau(j)*dble(iw)
C
                  Ysum(j,1)=Ysum(j,1)+(qfullhistory(j,((istep-1)-iw+1),1)
     &                              -qfullhistory(j,((istep-1)-iw-1),1))/area(j)*
     &                              weightfn(wtau)
                  Ysum(j,2)=Ysum(j,2)+(qfullhistory(j,((istep-1)-iw+1),2)
     &                              -qfullhistory(j,((istep-1)-iw-1),2))/area(j)*
                              weightfn(wtau)
C
                end do
C
            end do
C
          end if ! end of check for istep >= 4
C
        end if ! end of check for zielke unsteady friction
C
C**********************************************************************72
C
```

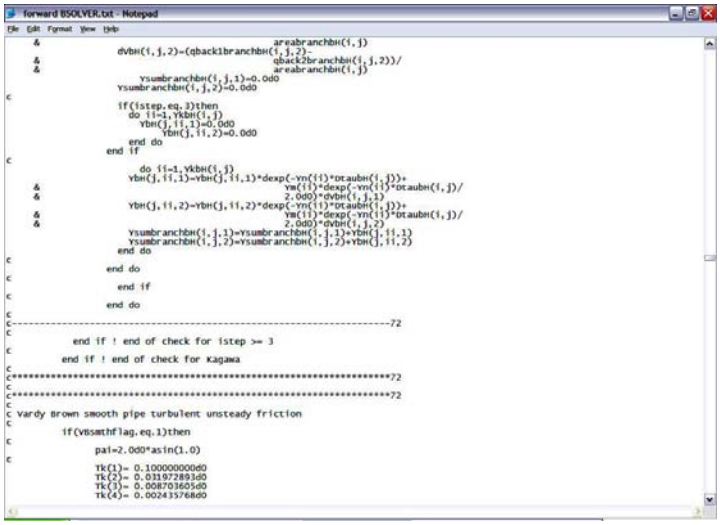


```
forward BSOLVER.txt - Notepad
File  Edit  Format  View  Help
C
C Kagawa unsteady friction
C
        if(kagawaflag.eq.1)then
C
C Set Ysum(i,1)=0 and Ysum(i,2)=0 for fast unsteady friction and istep < 3
C
          if(istep.lt.3)then
            do i=1,nreaches
              Ysum(i,1)=0.0d0
              Ysum(i,2)=0.0d0
C
                    if((branchflag(i).eq.1).and.(horzbflag(i).eq.1))then
                      do j=1,limitbh
                        Ysumbranchbh(i,j,1)=0.0d0
                        Ysumbranchbh(i,j,2)=0.0d0
                      end do
                    end if
C
            end do
          end if
C
C Activate fast unsteady friction for istep >= 3
C
          if(istep.ge.3)then
C
C Set up for fast unsteady friction for different pipe diameters
C
            do i=1,nreaches
              Dtau(i)=4.0d0*viscosity*dt/(d(i)**2.0d0)
              Parz(i)=16.0d0*viscosity/g/(d(i)**2.0d0)*dx
C
                    if((branchflag(i).eq.1).and.(horzbflag(i).eq.1))then
                      do j=1,limitbh
                        Dtaubh(i,j)=4.0d0*viscosity*dt/
     &                                  (dbranchbh(i,j)**2.0d0)
                        Parzbranchbh(i,j)=16.0d0*viscosity/g/
     &                                  (dbranchbh(i,j)**2.0d0)*dx
                      end do
                    end if
C
            end do
C
C Work out how many past weights to use
C
            do i=1,nreaches
              Yk(i)=0
              do j=1,10
                if((Yk(i).eq.0).and.((Dtau(i)/2.0).gt.(Tk(j))))then
                  Yk(i)=j
                end if
```

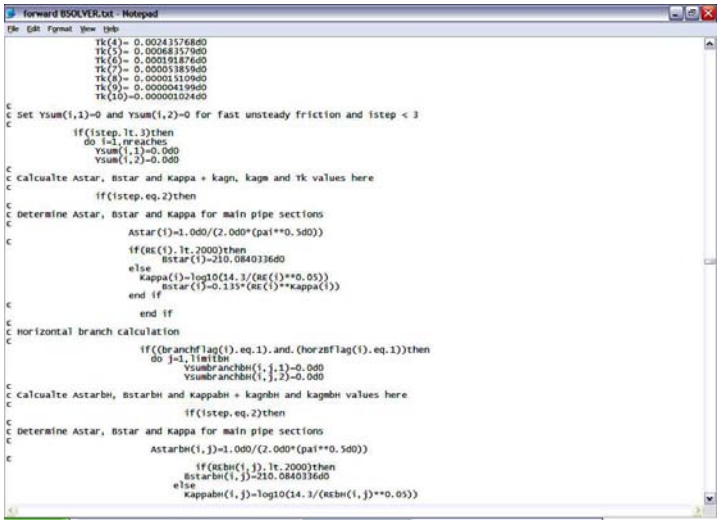
Appendix M – Fortran code for BSOLVER and NLFIT subroutines



```
forward BSOLVER.txt - Notepad
File  Edit  Format  View  Help
                end if
              end do
c
              if(Yk(i).eq.0)then
                Yk(i)=10
              end if
c
                    if((branchflag(i).eq.1).and.(horzBflag(i).eq.1))then
                      do j=1,limitbh
                        Ykbh(i,j)=10
                      end do
                    end if
c
            end do
c
c CALCULATE FAST UNSTEADY FRICTION VARIABLES
c
            do i=1,nreaches
              dV(i,1)=(qback1(i,1)-qback2(i,1))/area(i)
              dV(i,2)=(qback1(i,2)-qback2(i,2))/area(i)
                Ysum(i,1)=0.0d0
              Ysum(i,2)=0.0d0
c
              if(istep.eq.1)then
                do ii=1,Yk(i)
                  Y(i,ii,1)=0.0d0
                    Y(i,ii,2)=0.0d0
                end do
              end if
c
                  do ii=1,Yk(i)
                Y(i,ii,1)=Y(i,ii,1)*dexp(-Yn(ii)*dtau(i))+Ym(ii)*
     &              dexp(-Yn(ii)*dtau(i)/2.0d0)*dV(i,1)
                Y(i,ii,2)=Y(i,ii,2)*dexp(-Yn(ii)*dtau(i))+Ym(ii)*
     &              dexp(-Yn(ii)*dtau(i)/2.0d0)*dV(i,2)
                Ysum(i,1)=Ysum(i,1)+Y(i,ii,1)
                Ysum(i,2)=Ysum(i,2)+Y(i,ii,2)
              end do
            end do
c
c Section of code for horizontal branch USF
c
              do i=1,nreaches
c
                if((branchflag(i).eq.1).and.(horzBflag(i).eq.1))then
c
              do j=1,limitbh

                dVbh(i,j,1)=(qback1branchbh(i,j,1)-
     &                                        qback2branchbh(i,j,1))/
     &                                        areabranchbh(i,j)
```

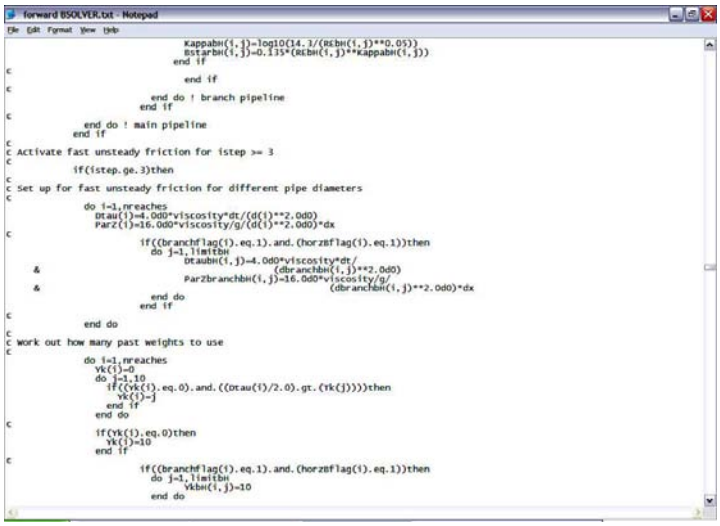


```
forward BSOLVER.txt - Notepad
File  Edit  Format  View  Help
     &                                        areabranchbh(i,j)
                dVbh(i,j,2)=(qback1branchbh(i,j,2)-
     &                                        qback2branchbh(i,j,2))/
     &                                        areabranchbh(i,j)
                    Ysumbranchbh(i,j,1)=0.0d0
                Ysumbranchbh(i,j,2)=0.0d0
c
                if(istep.eq.1)then
                  do ii=1,Ykbh(i,j)
                    Ybh(j,ii,1)=0.0d0
                        Ybh(j,ii,2)=0.0d0
                  end do
                end if
c
                    do ii=1,Ykbh(i,j)
                  Ybh(j,ii,1)=Ybh(j,ii,1)*dexp(-Yn(ii)*dtaubh(i,j))+
     &                                  Ym(ii)*dexp(-Yn(ii)*dtaubh(i,j)/
     &                                  2.0d0)*dVbh(i,j,1)
                  Ybh(j,ii,2)=Ybh(j,ii,2)*dexp(-Yn(ii)*dtaubh(i,j))+
     &                                  Ym(ii)*dexp(-Yn(ii)*dtaubh(i,j)/
     &                                  2.0d0)*dVbh(i,j,2)
                  Ysumbranchbh(i,j,1)=Ysumbranchbh(i,j,1)+Ybh(j,ii,1)
                  Ysumbranchbh(i,j,2)=Ysumbranchbh(i,j,2)+Ybh(j,ii,2)
                end do
c
              end do
c
                end if
c
              end do
c
c-----------------------------------------------------------------72
c
          end if ! end of check for istep >= 3
c
        end if ! end of check for Kagawa
c
c*******************************************************************72
c
c*******************************************************************72
c
c Vardy Brown smooth pipe turbulent unsteady friction
c
        if(VBsmthflag.eq.1)then
c
            pai=2.0d0*asin(1.0)
c
            Tk(1)= 0.100000000d0
            Tk(2)= 0.031972893d0
            Tk(3)= 0.008703605d0
            Tk(4)= 0.002435768d0
```

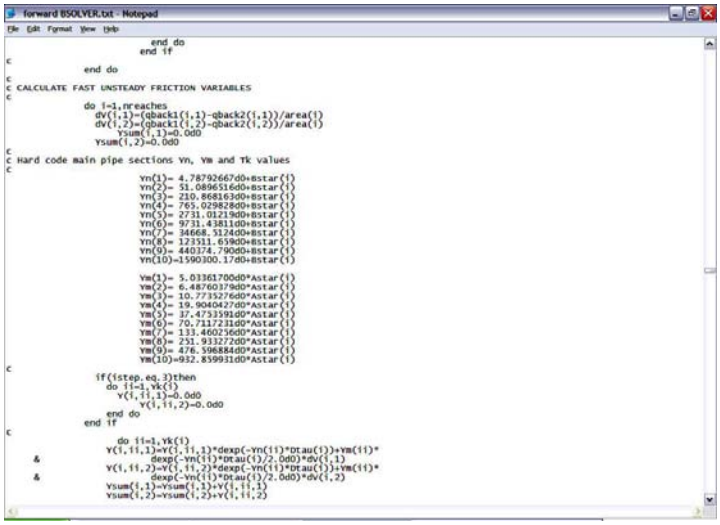


```
forward BSOLVER.txt - Notepad
File  Edit  Format  View  Help
            Tk(4)= 0.002435768d0
            Tk(5)= 0.000683579d0
            Tk(6)= 0.000191876d0
            Tk(7)= 0.000053859d0
            Tk(8)= 0.000015109d0
            Tk(9)= 0.000004199d0
            Tk(10)=0.000001024d0
c
c Set Ysum(i,1)=0 and Ysum(i,2)=0 for fast unsteady friction and istep < 3
c
          if(istep.lt.3)then
            do i=1,nreaches
              Ysum(i,1)=0.0d0
              Ysum(i,2)=0.0d0
c
c Calcualte Astar, Bstar and Kappa + kagn, kagm and Tk values here
c
              if(istep.eq.2)then
c
c Determine Astar, Bstar and Kappa for main pipe sections
c
                  Astar(i)=1.0d0/(2.0d0*(pai**0.5d0))
c
                  if(RE(i).lt.2000)then
                      Bstar(i)=210.0840336d0
                  else
                    Kappa(i)=log10(14.3/(RE(i)**0.05))
                      Bstar(i)=0.135*(RE(i)**Kappa(i))
                  end if
c
                    end if
c
c Horizontal branch calculation
c
                  if((branchflag(i).eq.1).and.(horzBflag(i).eq.1))then
                    do j=1,limitbh
                        Ysumbranchbh(i,j,1)=0.0d0
                        Ysumbranchbh(i,j,2)=0.0d0
c
c Calcualte Astarbh, Bstarbh and Kappabh + kagnbh and kagmbh values here
c
                          if(istep.eq.2)then
c
c Determine Astar, Bstar and Kappa for main pipe sections
c
                           Astarbh(i,j)=1.0d0/(2.0d0*(pai**0.5d0))
c
                              if(REbh(i,j).lt.2000)then
                            Bstarbh(i,j)=210.0840336d0
                          else
                            Kappabh(i,j)=log10(14.3/(REbh(i,j)**0.05))
```

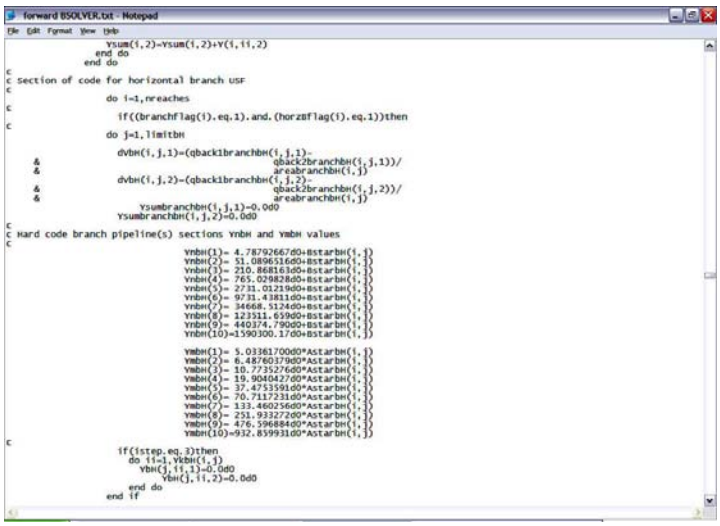
Appendix M – Fortran code for BSOLVER and NLFIT subroutines



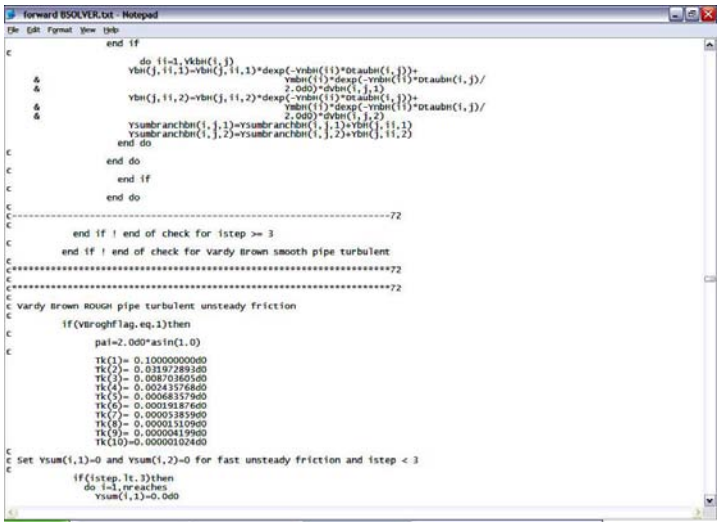
```
forward BSOLVER.txt - Notepad
File  Edit  Format  View  Help
                                Kappabh(i,j)=log10(14.7/(Rsbh(i,j)**0.05))
                                Bstarbh(i,j)=0.135*(Rsbh(i,j)**Kappabh(i,j))
                              end if
c
                                end if
c
                            end do ! branch pipeline
                          end if
c
                end do ! main pipeline
              end if
c
c Activate fast unsteady friction for istep >= 3
c
              if(istep.ge.3)then
c
c Set up for fast unsteady friction for different pipe diameters
c
                do i=1,nreaches
                  Dtau(i)=4.0d0*viscosity*dt/(d(i)**2.0d0)
                  Parz(i)=16.0d0*viscosity/g/(d(i)**2.0d0)*dx
c
                        if((branchflag(i).eq.1).and.(horzbflag(i).eq.1))then
                          do j=1,limitbh
                            Dtaubh(i,j)=4.0d0*viscosity*dt/
     &                                        (dbranchbh(i,j)**2.0d0)
                            Parzbranchbh(i,j)=16.0d0*viscosity/g/
     &                                        (dbranchbh(i,j)**2.0d0)*dx
                          end do
                        end if
c
                end do
c
c Work out how many past weights to use
c
                do i=1,nreaches
                  Yk(i)=0
                  do j=1,10
                    if((Yk(i).eq.0).and.((Dtau(i)/2.0).gt.(Tk(j))))then
                      Yk(i)=j
                    end if
                  end do
c
                  if(Yk(i).eq.0)then
                    Yk(i)=10
                  end if
c
                        if((branchflag(i).eq.1).and.(horzbflag(i).eq.1))then
                          do j=1,limitbh
                            Ykbh(i,j)=10
                          end do
```



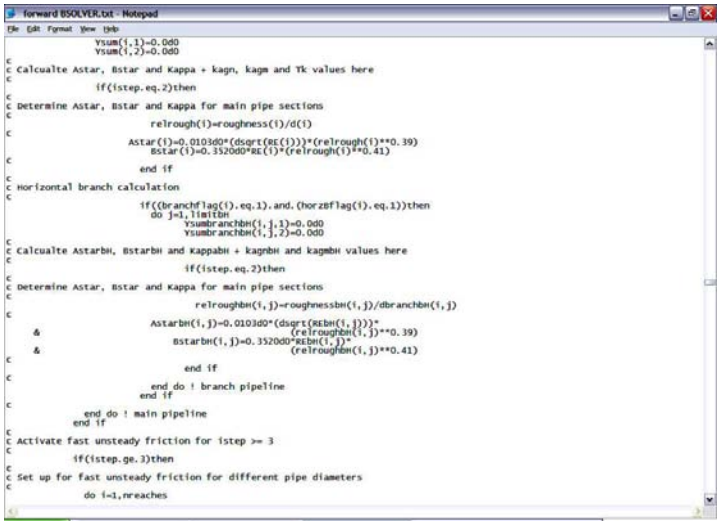
```
forward BSOLVER.txt - Notepad
File  Edit  Format  View  Help
                            end do
                          end if
c
                end do
c
c CALCULATE FAST UNSTEADY FRICTION VARIABLES
c
                do i=1,nreaches
                  dV(i,1)=(qback1(i,1)-qback2(i,1))/area(i)
                  dV(i,2)=(qback1(i,2)-qback2(i,2))/area(i)
                    Ysum(i,1)=0.0d0
                  Ysum(i,2)=0.0d0
c
c Hard code main pipe sections Yn, Ym and Tk values
c
                            Yn(1)= 4.78792667d0*Bstar(i)
                            Yn(2)= 51.0896516d0*Bstar(i)
                            Yn(3)= 210.868163d0*Bstar(i)
                            Yn(4)= 765.029828d0*Bstar(i)
                            Yn(5)= 2731.01219d0*Bstar(i)
                            Yn(6)= 9731.43811d0*Bstar(i)
                            Yn(7)= 34668.5124d0*Bstar(i)
                            Yn(8)= 123511.659d0*Bstar(i)
                            Yn(9)= 440374.790d0*Bstar(i)
                            Yn(10)=1590300.17d0*Bstar(i)

                            Ym(1)= 5.03361700d0*Astar(i)
                            Ym(2)= 6.48760379d0*Astar(i)
                            Ym(3)= 10.7735276d0*Astar(i)
                            Ym(4)= 19.9040427d0*Astar(i)
                            Ym(5)= 37.4753591d0*Astar(i)
                            Ym(6)= 70.7117231d0*Astar(i)
                            Ym(7)= 133.460256d0*Astar(i)
                            Ym(8)= 251.933272d0*Astar(i)
                            Ym(9)= 476.596884d0*Astar(i)
                            Ym(10)=932.859931d0*Astar(i)
c
                  if(istep.eq.3)then
                    do ii=1,Yk(i)
                      Y(i,ii,1)=0.0d0
                      Y(i,ii,2)=0.0d0
                    end do
                  end if
c
                    do ii=1,Yk(i)
                  Y(i,ii,1)=Y(i,ii,1)*dexp(-Yn(ii)*Dtau(i))+Ym(ii)*
     &                dexp(-Yn(ii)*Dtau(i)/2.0d0)*dV(i,1)
                  Y(i,ii,2)=Y(i,ii,2)*dexp(-Yn(ii)*Dtau(i))+Ym(ii)*
     &                dexp(-Yn(ii)*Dtau(i)/2.0d0)*dV(i,2)
                  Ysum(i,1)=Ysum(i,1)+Y(i,ii,1)
                  Ysum(i,2)=Ysum(i,2)+Y(i,ii,2)
```



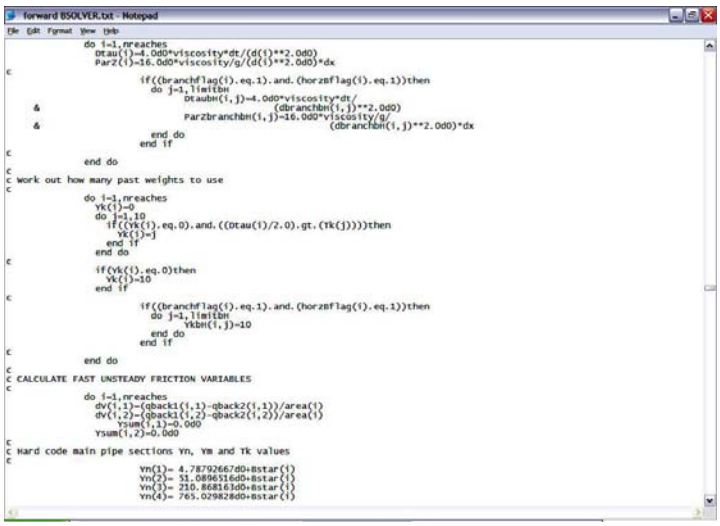
```
forward BSOLVER.txt - Notepad
File  Edit  Format  View  Help
                  Ysum(i,2)=Ysum(i,2)+Y(i,ii,2)
                end do
              end do
c
c Section of code for horizontal branch USF
c
                do i=1,nreaches
c
                  if((branchflag(i).eq.1).and.(horzbflag(i).eq.1))then
c
                do j=1,limitbh

                  dVbh(i,j,1)=(qback1branchbh(i,j,1)-
     &                                    qback2branchbh(i,j,1))/
     &                                    areabranchbh(i,j)
                  dVbh(i,j,2)=(qback1branchbh(i,j,2)-
     &                                    qback2branchbh(i,j,2))/
     &                                    areabranchbh(i,j)
                      Ysumbranchbh(i,j,1)=0.0d0
                  Ysumbranchbh(i,j,2)=0.0d0
c
c Hard code branch pipeline(s) sections Ynbh and Ymbh values
c
                                    Ynbh(1)= 4.78792667d0*Bstarbh(i,j)
                                    Ynbh(2)= 51.0896516d0*Bstarbh(i,j)
                                    Ynbh(3)= 210.868163d0*Bstarbh(i,j)
                                    Ynbh(4)= 765.029828d0*Bstarbh(i,j)
                                    Ynbh(5)= 2731.01219d0*Bstarbh(i,j)
                                    Ynbh(6)= 9731.43811d0*Bstarbh(i,j)
                                    Ynbh(7)= 34668.5124d0*Bstarbh(i,j)
                                    Ynbh(8)= 123511.659d0*Bstarbh(i,j)
                                    Ynbh(9)= 440374.790d0*Bstarbh(i,j)
                                    Ynbh(10)=1590300.17d0*Bstarbh(i,j)

                                    Ymbh(1)= 5.03361700d0*Astarbh(i,j)
                                    Ymbh(2)= 6.48760379d0*Astarbh(i,j)
                                    Ymbh(3)= 10.7735276d0*Astarbh(i,j)
                                    Ymbh(4)= 19.9040427d0*Astarbh(i,j)
                                    Ymbh(5)= 37.4753591d0*Astarbh(i,j)
                                    Ymbh(6)= 70.7117231d0*Astarbh(i,j)
                                    Ymbh(7)= 133.460256d0*Astarbh(i,j)
                                    Ymbh(8)= 251.933272d0*Astarbh(i,j)
                                    Ymbh(9)= 476.596884d0*Astarbh(i,j)
                                    Ymbh(10)=932.859931d0*Astarbh(i,j)
c
                  if(istep.eq.3)then
                    do ii=1,Ykbh(i,j)
                      Ybh(j,ii,1)=0.0d0
                          Ybh(j,ii,2)=0.0d0
                    end do
                  end if
```



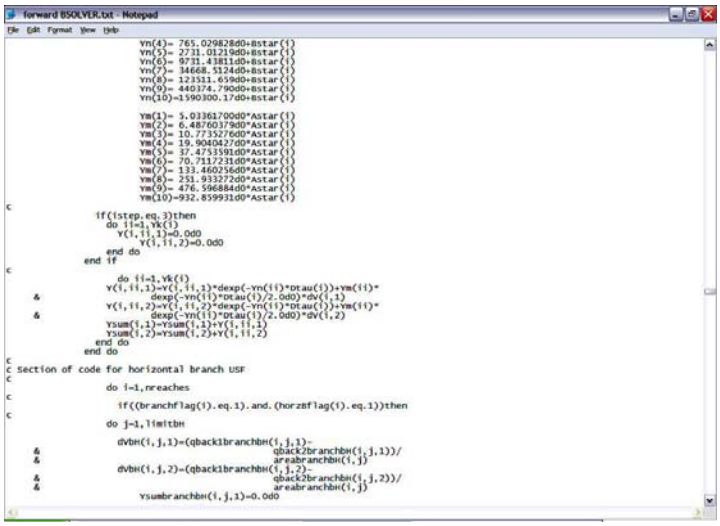
```
forward BSOLVER.txt - Notepad
File  Edit  Format  View  Help
                    end if
c
                      do ii=1,Ykbh(i,j)
                    Ybh(j,ii,1)=Ybh(j,ii,1)*dexp(-Ynbh(ii)*Dtaubh(i,j))+
     &                                      Ynbh(ii)*dexp(-Ynbh(ii)*Dtaubh(i,j)/
     &                                                    2.0d0)*dYbh(i,j,1)
                    Ybh(j,ii,2)=Ybh(j,ii,2)*dexp(-Ynbh(ii)*Dtaubh(i,j))+
     &                                      Ynbh(ii)*dexp(-Ynbh(ii)*Dtaubh(i,j)/
     &                                                    2.0d0)*dYbh(i,j,2)
                    Ysumbranchbh(i,j,1)=Ysumbranchbh(i,j,1)+Ybh(j,ii,1)
                    Ysumbranchbh(i,j,2)=Ysumbranchbh(i,j,2)+Ybh(j,ii,2)
                  end do
c
                end do
c
                  end if
c
                end do
c
c-----------------------------------------------------------------------72
c
            end if ! end of check for istep >= 3
c
          end if ! end of check for Vardy Brown smooth pipe turbulent
c
c**********************************************************************72
c
c**********************************************************************72
c
c Vardy Brown ROUGH pipe turbulent unsteady friction
c
          if(VBroghflag.eq.1)then
c
                pai=2.0d0*asin(1.0)
c
                Tk(1)= 0.100000000d0
                Tk(2)= 0.031972893d0
                Tk(3)= 0.008703605d0
                Tk(4)= 0.002435768d0
                Tk(5)= 0.000683579d0
                Tk(6)= 0.000191876d0
                Tk(7)= 0.000053859d0
                Tk(8)= 0.000015109d0
                Tk(9)= 0.000004199d0
                Tk(10)=0.000001024d0
c
c Set Ysum(i,1)=0 and Ysum(i,2)=0 for fast unsteady friction and istep < 3
c
            if(istep.lt.3)then
              do i=1,nreaches
                Ysum(i,1)=0.0d0
```



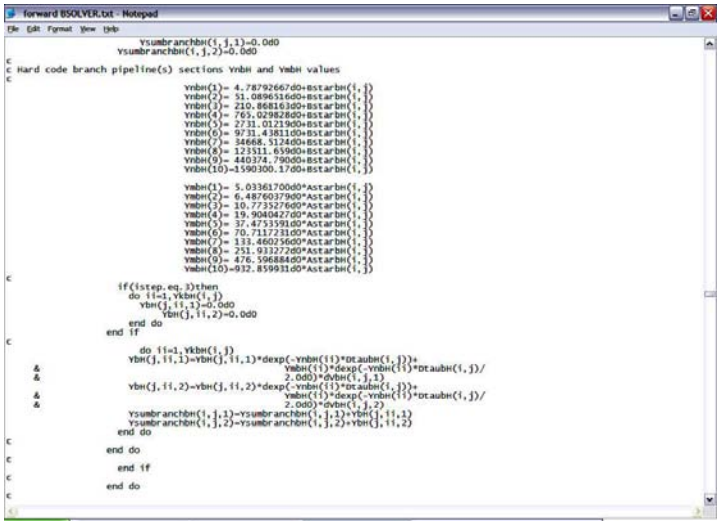
```
forward BSOLVER.txt - Notepad
File  Edit  Format  View  Help
                Ysum(i,1)=0.0d0
                Ysum(i,2)=0.0d0
c
c Calcualte Astar, Bstar and Kappa + kagn, kagm and Tk values here
c
                if(istep.eq.2)then
c
c Determine Astar, Bstar and Kappa for main pipe sections
c
                    relrough(i)=roughness(i)/d(i)
c
                  Astar(i)=0.0103d0*(dsqrt(RE(i)))*(relrough(i)**0.39)
                    Bstar(i)=0.3520d0*RE(i)*(relrough(i)**0.41)
c
                  end if
c
c Horizontal branch calculation
c
                  if((branchflag(i).eq.1).and.(horzbflag(i).eq.1))then
                    do j=1,limitbh
                      Ysumbranchbh(i,j,1)=0.0d0
                      Ysumbranchbh(i,j,2)=0.0d0
c
c Calcualte Astarbh, Bstarbh and Kappabh + kagnbh and kagmbh values here
c
                      if(istep.eq.2)then
c
c Determine Astar, Bstar and Kappa for main pipe sections
c
                        relroughbh(i,j)=roughnessbh(i,j)/dbranchbh(i,j)
c
                      Astarbh(i,j)=0.0103d0*(dsqrt(REbh(i,j)))*
     &                                  (relroughbh(i,j)**0.39)
                        Bstarbh(i,j)=0.3520d0*REbh(i,j)*
     &                                  (relroughbh(i,j)**0.41)
c
                        end if
c
                    end do ! branch pipeline
                  end if
c
              end do ! main pipeline
            end if
c
c Activate fast unsteady friction for istep >= 3
c
            if(istep.ge.3)then
c
c Set up for fast unsteady friction for different pipe diameters
c
              do i=1,nreaches
```



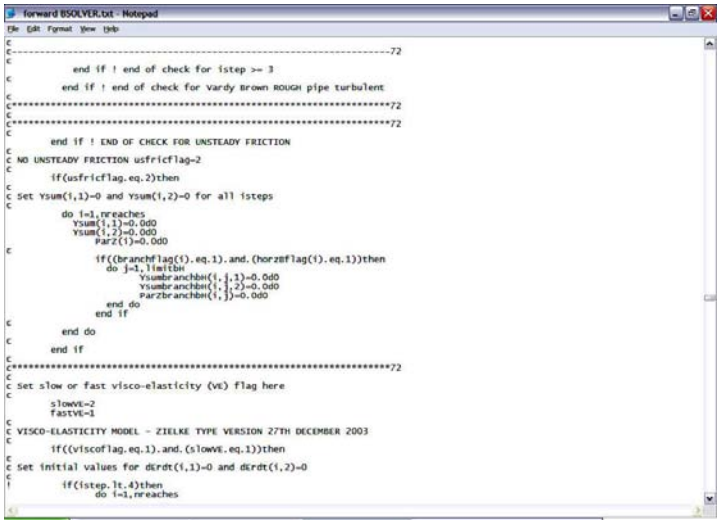
```
forward BSOLVER.txt - Notepad
File  Edit  Format  View  Help
              do i=1,nreaches
                Dtau(i)=4.0d0*viscosity*dt/(d(i)**2.0d0)
                Par2(i)=16.0d0*viscosity/g/(d(i)**2.0d0)*dx
c
                  if((branchflag(i).eq.1).and.(horzbflag(i).eq.1))then
                    do j=1,limitbh
                      Dtaubh(i,j)=4.0d0*viscosity*dt/
     &                                  (dbranchbh(i,j)**2.0d0)
                      Parzbranchbh(i,j)=16.0d0*viscosity/g/
     &                                  (dbranchbh(i,j)**2.0d0)*dx
                    end do
                  end if
c
              end do
c
c Work out how many past weights to use
c
              do i=1,nreaches
                Yk(i)=0
                do j=1,10
                  if((Yk(i).eq.0).and.((Dtau(i)/2.0).gt.(Tk(j))))then
                    Yk(i)=j
                  end if
                end do
c
                if(Yk(i).eq.0)then
                  Yk(i)=10
                end if
c
                  if((branchflag(i).eq.1).and.(horzbflag(i).eq.1))then
                    do j=1,limitbh
                      Ykbh(i,j)=10
                    end do
                  end if
c
              end do
c
c CALCULATE FAST UNSTEADY FRICTION VARIABLES
c
              do i=1,nreaches
                dV(i,1)=(qback1(i,1)-qback2(i,1))/area(i)
                dV(i,2)=(qback1(i,2)-qback2(i,2))/area(i)
                  Ysum(i,1)=0.0d0
                Ysum(i,2)=0.0d0
c
c Hard code main pipe sections Yn, Ym and Tk values
c
                  Yn(1)= 4.78792667d0*Bstar(i)
                  Yn(2)= 51.0896516d0*Bstar(i)
                  Yn(3)= 210.868163d0*Bstar(i)
                  Yn(4)= 765.029828d0*Bstar(i)
```



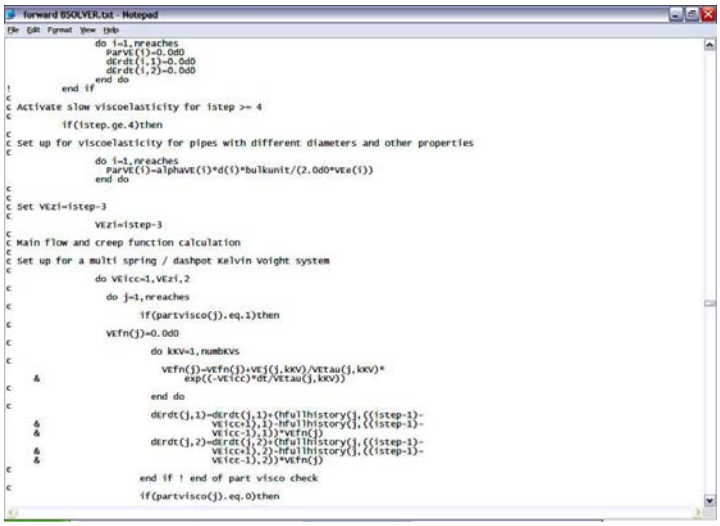

```
                        Yn(4)= 765.029828d0*Bstar(i)
                        Yn(5)= 2731.01219d0*Bstar(i)
                        Yn(6)= 9731.43811d0*Bstar(i)
                        Yn(7)= 34668.5124d0*Bstar(i)
                        Yn(8)= 123511.659d0*Bstar(i)
                        Yn(9)= 440374.790d0*Bstar(i)
                        Yn(10)=1590300.17d0*Bstar(i)

                        Ym(1)= 5.03361700d0*Astar(i)
                        Ym(2)= 6.48760379d0*Astar(i)
                        Ym(3)= 10.7735276d0*Astar(i)
                        Ym(4)= 19.9040427d0*Astar(i)
                        Ym(5)= 37.4753591d0*Astar(i)
                        Ym(6)= 70.7117231d0*Astar(i)
                        Ym(7)= 133.460256d0*Astar(i)
                        Ym(8)= 251.933272d0*Astar(i)
                        Ym(9)= 476.596884d0*Astar(i)
                        Ym(10)=932.859931d0*Astar(i)
c
                if(istep.eq.3)then
                  do ii=1,Vk(i)
                    Y(i,ii,1)=0.0d0
                    Y(i,ii,2)=0.0d0
                  end do
                end if
c
                  do ii=1,Vk(i)
                Y(i,ii,1)=Y(i,ii,1)*dexp(-Yn(ii)*Dtau(i))+Ym(ii)*
     &                dexp(-Yn(ii)*Dtau(i)/2.0d0)*dv(i,1)
                Y(i,ii,2)=Y(i,ii,2)*dexp(-Yn(ii)*Dtau(i))+Ym(ii)*
     &                dexp(-Yn(ii)*Dtau(i)/2.0d0)*dv(i,2)
                Ysum(i,1)=Ysum(i,1)+Y(i,ii,1)
                Ysum(i,2)=Ysum(i,2)+Y(i,ii,2)
                  end do
                end do
c
c Section of code for horizontal branch USF
c
                do i=1,nreaches
c
                  if((branchflag(i).eq.1).and.(horzBflag(i).eq.1))then
c
                do j=1,limitbH

                  dvbH(i,j,1)=(qback1branchbH(i,j,1)-
     &                                qback2branchbH(i,j,1))/
     &                                areabranchbH(i,j)
                  dvbH(i,j,2)=(qback1branchbH(i,j,2)-
     &                                qback2branchbH(i,j,2))/
     &                                areabranchbH(i,j)
                      Ysumbranchbh(i,j,1)=0.0d0
```



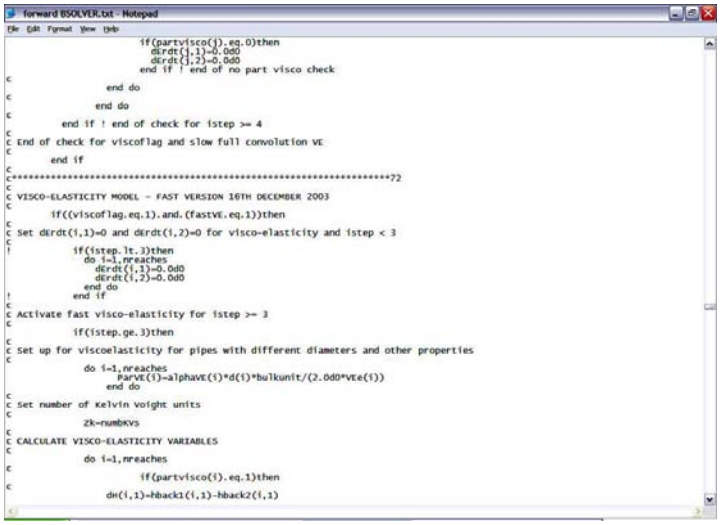

```
                      Ysumbranchbh(i,j,1)=0.0d0
                    Ysumbranchbh(i,j,2)=0.0d0
c
c Hard code branch pipeline(s) sections YnbH and YmbH values
c
                                Ynbh(1)= 4.78792667d0*Bstarbh(i,j)
                                Ynbh(2)= 51.0896516d0*Bstarbh(i,j)
                                Ynbh(3)= 210.868163d0*Bstarbh(i,j)
                                Ynbh(4)= 765.029828d0*Bstarbh(i,j)
                                Ynbh(5)= 2731.01219d0*Bstarbh(i,j)
                                Ynbh(6)= 9731.43811d0*Bstarbh(i,j)
                                Ynbh(7)= 34668.5124d0*Bstarbh(i,j)
                                Ynbh(8)= 123511.659d0*Bstarbh(i,j)
                                Ynbh(9)= 440374.790d0*Bstarbh(i,j)
                                Ynbh(10)=1590300.17d0*Bstarbh(i,j)

                                Ymbh(1)= 5.03361700d0*Astarbh(i,j)
                                Ymbh(2)= 6.48760379d0*Astarbh(i,j)
                                Ymbh(3)= 10.7735276d0*Astarbh(i,j)
                                Ymbh(4)= 19.9040427d0*Astarbh(i,j)
                                Ymbh(5)= 37.4753591d0*Astarbh(i,j)
                                Ymbh(6)= 70.7117231d0*Astarbh(i,j)
                                Ymbh(7)= 133.460256d0*Astarbh(i,j)
                                Ymbh(8)= 251.933272d0*Astarbh(i,j)
                                Ymbh(9)= 476.596884d0*Astarbh(i,j)
                                Ymbh(10)=932.859931d0*Astarbh(i,j)
c
                if(istep.eq.3)then
                  do ii=1,Vkbh(i,j)
                    Ybh(j,ii,1)=0.0d0
                        Ybh(j,ii,2)=0.0d0
                  end do
                end if
c
                    do ii=1,Vkbh(i,j)
                  Ybh(j,ii,1)=Ybh(j,ii,1)*dexp(-Ynbh(ii)*Dtaubh(i,j))+
     &                                Ymbh(ii)*dexp(-Ynbh(ii)*Dtaubh(i,j)/
     &                                2.0d0)*dvbh(i,j,1)
                  Ybh(j,ii,2)=Ybh(j,ii,2)*dexp(-Ynbh(ii)*Dtaubh(i,j))+
     &                                Ymbh(ii)*dexp(-Ynbh(ii)*Dtaubh(i,j)/
     &                                2.0d0)*dvbh(i,j,2)
                  Ysumbranchbh(i,j,1)=Ysumbranchbh(i,j,1)+Ybh(j,ii,1)
                  Ysumbranchbh(i,j,2)=Ysumbranchbh(i,j,2)+Ybh(j,ii,2)
                    end do
c
                end do
c
                  end if
c
                end do
c
```



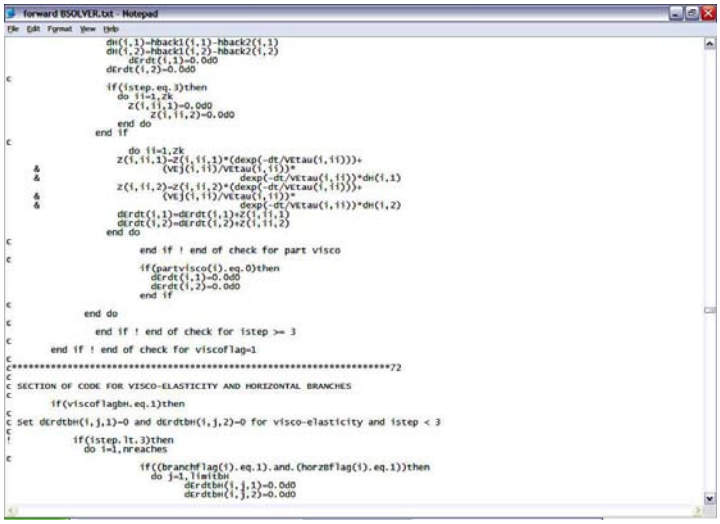

```
c
c-----------------------------------------------------------------72
c
            end if ! end of check for istep >= 3
c
          end if ! end of check for Vardy Brown ROUGH pipe turbulent
c
c**********************************************************************72
c
c**********************************************************************72
c
        end if ! END OF CHECK FOR UNSTEADY FRICTION
c
c NO UNSTEADY FRICTION usfricflag=2
c
        if(usfricflag.eq.2)then
c
c Set Ysum(i,1)=0 and Ysum(i,2)=0 for all isteps
c
          do i=1,nreaches
            Ysum(i,1)=0.0d0
            Ysum(i,2)=0.0d0
              Parz(i)=0.0d0
c
              if((branchflag(i).eq.1).and.(horzBflag(i).eq.1))then
                do j=1,limitbH
                      Ysumbranchbh(i,j,1)=0.0d0
                      Ysumbranchbh(i,j,2)=0.0d0
                      Parzbranchbh(i,j)=0.0d0
                end do
              end if
c
          end do
c
        end if
c
c**********************************************************************72
c
c Set slow or fast visco-elasticity (VE) flag here
c
        slowVE=2
        fastVE=1
c
c VISCO-ELASTICITY MODEL - ZIELKE TYPE VERSION 27TH DECEMBER 2003
c
        if((viscoflag.eq.1).and.(slowVE.eq.1))then
c
c Set initial values for dErdt(i,1)=0 and dErdt(i,2)=0
c
!         if(istep.lt.4)then
            do i=1,nreaches
```



```
forward BSOLVER.txt - Notepad
File  Edit  Format  View  Help
                do i=1,nreaches
                  ParVE(i)=0.0d0
                  dErdt(i,1)=0.0d0
                  dErdt(i,2)=0.0d0
                end do
!         end if
C
C Activate slow viscoelasticity for istep >= 4
C
          if(istep.ge.4)then
C
C Set up for viscoelasticity for pipes with different diameters and other properties
C
                do i=1,nreaches
                  ParVE(i)=alphaVE(i)*d(i)*bulkunit/(2.0d0*VEe(i))
                end do
C
C
C Set VEzi=istep-3
C
                VEzi=istep-3
C
C Main flow and creep function calculation
C
C Set up for a multi spring / dashpot Kelvin Voight system
C
                do VEicc=1,VEzi,2
C
                  do j=1,nreaches
C
                        if(partvisco(j).eq.1)then
C
                  VEfn(j)=0.0d0
C
                          do kkV=1,numbKVs
C
                            VEfn(j)=VEfn(j)+VEj(j,kkV)/VEtau(j,kkV)*
     &                           exp((-VEicc)*dt/VEtau(j,kkV))
C
                          end do
C
                          dErdt(j,1)=dErdt(j,1)+(hfullhistory(j,((istep-1)-
     &                                 VEicc+1),1)-hfullhistory(j,((istep-1)-
     &                                 VEicc-1),1))*VEfn(j)
                          dErdt(j,2)=dErdt(j,2)+(hfullhistory(j,((istep-1)-
     &                                 VEicc+1),2)-hfullhistory(j,((istep-1)-
     &                                 VEicc-1),2))*VEfn(j)
C
                        end if ! end of part visco check
C
                        if(partvisco(j).eq.0)then
```



```
forward BSOLVER.txt - Notepad
File  Edit  Format  View  Help
                        if(partvisco(j).eq.0)then
                          dErdt(j,1)=0.0d0
                          dErdt(j,2)=0.0d0
                        end if ! end of no part visco check
C
                  end do
C
                end do
C
          end if ! end of check for istep >= 4
C
C End of check for viscoflag and slow full convolution VE
C
        end if
C
C*********************************************************************72
C
C VISCO-ELASTICITY MODEL - FAST VERSION 16TH DECEMBER 2003
C
        if((viscoflag.eq.1).and.(fastVE.eq.1))then
C
C Set dErdt(i,1)=0 and dErdt(i,2)=0 for visco-elasticity and istep < 3
C
!           if(istep.lt.3)then
!             do i=1,nreaches
!               dErdt(i,1)=0.0d0
!               dErdt(i,2)=0.0d0
!             end do
!           end if
C
C Activate fast visco-elasticity for istep >= 3
C
            if(istep.ge.3)then
C
C Set up for viscoelasticity for pipes with different diameters and other properties
C
              do i=1,nreaches
                ParVE(i)=alphaVE(i)*d(i)*bulkunit/(2.0d0*VEe(i))
              end do
C
C Set number of Kelvin Voight units
C
              zk=numbKVs
C
C CALCULATE VISCO-ELASTICITY VARIABLES
C
              do i=1,nreaches
C
                      if(partvisco(i).eq.1)then
C
                dh(i,1)=hback1(i,1)-hback2(i,1)
```



```
forward BSOLVER.txt - Notepad
File  Edit  Format  View  Help
                dh(i,1)=hback1(i,1)-hback2(i,1)
                dh(i,2)=hback1(i,2)-hback2(i,2)
                  dErdt(i,1)=0.0d0
                dErdt(i,2)=0.0d0
C
                if(istep.eq.3)then
                  do ii=1,zk
                    z(i,ii,1)=0.0d0
                      z(i,ii,2)=0.0d0
                  end do
                end if
C
                    do ii=1,zk
                  z(i,ii,1)=z(i,ii,1)*(dexp(-dt/VEtau(i,ii)))+
     &                    (VE)(i,ii)/VEtau(i,ii))*
     &                                    dexp(-dt/VEtau(i,ii))*dh(i,1)
                  z(i,ii,2)=z(i,ii,2)*(dexp(-dt/VEtau(i,ii)))+
     &                    (VE)(i,ii)/VEtau(i,ii))*
     &                                    dexp(-dt/VEtau(i,ii))*dh(i,2)
                  dErdt(i,1)=dErdt(i,1)+z(i,ii,1)
                  dErdt(i,2)=dErdt(i,2)+z(i,ii,2)
                end do
C
                      end if ! end of check for part visco
C
                      if(partvisco(i).eq.0)then
                        dErdt(i,1)=0.0d0
                        dErdt(i,2)=0.0d0
                      end if
C
              end do
C
                end if ! end of check for istep >= 3
C
          end if ! end of check for viscoflag=1
C
C*********************************************************************72
C
C SECTION OF CODE FOR VISCO-ELASTICITY AND HORIZONTAL BRANCHES
C
          if(viscoflagbH.eq.1)then
C
C Set dErdtbH(i,j,1)=0 and dErdtbH(i,j,2)=0 for visco-elasticity and istep < 3
C
!           if(istep.lt.3)then
              do i=1,nreaches
C
                      if((branchflag(i).eq.1).and.(horzbflag(i).eq.1))then
                        do j=1,limitbH
                              dErdtbH(i,j,1)=0.0d0
                              dErdtbH(i,j,2)=0.0d0
```