1 C

From: Joseph Staudenmeier NRR
To: Jennifer Uhle RES
Date: 10/5/00 5:42PM
Subject: Re: air/water reaction

Jennifer,

Atached are the 4 routines I made modifications to for spent fuel pool calculations. There were no modifications to RodCrunch.f90.

Joe

>>> Jennifer Uhle 10/05 5:17 PM >>>
Joe, can you email me the RodCrunchM.f90 with the air/water reaction. A guy at SCIENTECH in Albuquerque is interested in using TRAC-M for spent fuel work similar to what you did. I thought I could give him a modified version with this reaction in it if you don't mind.

Jennifer

B/10

```
      MODULE HeatCor
!
!     BEGIN MODULE USE
      USE IntrType
!
      CONTAINS
!
      SUBROUTINE chen(alp,cl,cpl,cpv,cv,gm,gravg,hdavg,hfg,hforc,hnucb,  &
     & p,pdrat,rol,rom,rov,sig,tl,tsat,tv,tw,visl,visv,vlc,vlz,vmc,vmz,  &
     & vvc,vvz,x,ca,ridreg)
!     BEGIN MODULE USE
      USE Io
      USE Flt
      USE EosNoInline
      USE GlobalDat
      USE HTPar
!
      IMPLICIT NONE
!
!     Declaration Generated by genImpDecs.pl 5/98
      REAL(sdk) alp,ca,cl,clohd,cpl,cpv,cv,delp,deltx,f,gm,gravg,hdavg,  &
     & hfg,hforc,hnucb,hturb,p,pdrat,prl,rel,retp,retpp,ridreg,rol,rom,  &
     & rov,sig,tl,tsat,tv,tw,visl,visv,vlc,vlz,vmc,vmz,vvc,vvz,x,xxtt
!
!
      INCLUDE 'constant.h'
!
!     this subroutine uses the chen correlation to evaluate the
!     forced convection nucleate boiling heat transfer coef.
!     asme 63-ht-34
!     curve fit approximations to f and s by butterworth
!
      f=0.d0
      IF (x.GE.1.e-5_sdk) THEN
        xxtt=  min(100.0_sdk,1.0_sdk/(exp(.9_sdk*log((1.0_sdk-x)/x)     &
     &        +.1_sdk*log(visl /visv))*sqrt(rov/rol)))
        IF (xxtt.GT.0.1d0) f=.8544153_sdk+.736*log(xxtt+.213_sdk)
      ENDIF
      rel=abs(vlz)*rol*(1.0_sdk-alp)*hdavg/visl
      retp=rel*exp(1.25_sdk*f)
      prl=visl*cpl/cl
      clohd=cl/hdavg
      IF (.NOT.(rel.LE.0.0_sdk.OR.prl.LE.0.0_sdk)) THEN
        hturb=.023d0*clohd*exp(.8_sdk*log(retp)+.4_sdk*log(prl))
      ELSE
        hturb=0.0_sdk
      ENDIF
      hforc=  max(4.0_sdk*clohd,hturb)
      retpp=  min(1.e-4_sdk*retp,70._sdk)
      hnucb=0.0_sdk
      ca=0.0_sdk
!
!     ridreg = 11.0 for condensation
!
      IF (ridreg.NE.11.0_sdk) THEN
!
```

```
      supres=1.0_sdk
      IF (retpp.GT.0.0_sdk) THEN
       IF (retpp.GE.32.5_sdk) THEN
        supres=one/(one+0.42_sdk*exp(.78_sdk*log(retpp)))
       ELSE
        supres=one/(one+0.12_sdk*exp(1.14_sdk*log(retpp)))
       ENDIF
      ENDIF
!
      deltx=tw-tsat
      IF (deltx.GT.zero) THEN
       delp=satprs(tw)-p
       IF (delp.GT.zero) THEN
        IF (.NOT.(cl.LE.0.d0.OR.cpl.LE.0.d0.OR.rol.LE.0.d0.OR.visl  &
     &    .LE.0.d0.OR.sig.LE.0.d0.OR.hfg.LE.0.d0)) THEN
         ca=supres*exp(-6.708904_sdk+.79_sdk*log(cl)+.45_sdk       &
     &       *log(cpl)+.49_sdk*log(rol)-.29_sdk*log(visl)-.24_sdk   &
     &       *log(hfg*rov))/sqrt(sig)
         hnucb=ca*exp(.24d0*log(deltx)+.75d0*log(delp))
        ENDIF
       ENDIF
      ENDIF
     ENDIF
     IF (.NOT.(hnucb.GT.0.d0.OR.hforc.GT.0.d0)) THEN
      WRITE (imout,200) genTab(cco)%num,nstep
      WRITE (iout,200) genTab(cco)%num,nstep
 200  FORMAT (                                          &
     &  ' **** the chen correlation has been passed bad numbers',     &
     &  ' for component',i5, ' at time step no.',i10,/,               &
     &  ' the heat transfer coefficients have been set to zero.',/,   &
     &  ' you should assure yourself the code has recovered',         &
     &  ' properly **** ')
      WRITE (imout,250) alp,x,p,tl,rol,cl,cpl,visl,vlz
      WRITE (iout,250) alp,x,p,tl,rol,cl,cpl,visl,vlz
 250  FORMAT ( ' alp,x,p,tl,rol,cl,cpl,visl,vlz: ',3e15.6,/,33x,      &
     &  3e15.6,/,33x,3e15.6)
      nittab=-mod(iabs(nittab),2)
      npwtab=0
     ENDIF
     RETURN
     END SUBROUTINE chen

     SUBROUTINE chf(alp,ca,cl,cpl,cpv,cv,gma,grava,hdavg,hfg,hforc,    &
    & hnucb,ichf,p,pdrat,qppchf,rol,rom,rov,sig,tchf,tl,tsat,tv,tw,    &
    & visl,visv,vlc,vlz,vmc,vmz,vvc,vvz,x)
!
!    BEGIN MODULE USE
     USE Io
     USE EosNoInline
!
     IMPLICIT NONE
!
!    Declaration Generated by genImpDecs.pl 5/98
     INTEGER(sik) icbs,ichf,icount,imax,isecmx
!
!    Declaration Generated by genImpDecs.pl 5/98
```

```
      REAL(sdk) alp,ca,cl,cpl,cpv,cv,dtchf,epsol,f1,f2,fchf,fcn,gm1,gma, &
     & grava,hdavg,hfg,hforc,hnucb,htc,htctmp,p,pdrat,psat,qppchf,rol,   &
     & rom,rov,sig,t1,t2,tchf,tchfo,tl,tsat,tv,tw,visl,visv,vlc,vlz,vmc, &
     & vmz,vvc,vvz,x,xchf,y1,y2,y3,y4,y5,y6,y7,y8
!
      REAL(sdk) dtchfo
!
!     this subroutine evaluates the critical heat flux based on
!     a local conditions formulation
!     chf options
!      1 -  chf package - biasi
!
!
!
      DATA epsol/0.5d0/,imax/35/,isecmx/7/
      fchf(y1,y2,y3,y4,y5,y6,y7,y8)=y4-(y6*(y1-y2)+y5*exp(.24d0*log(y1   &
     & -y3)+.75d0*log(  max(y7-y8,1.d-5)))*(y1-y3))
!
!
!
      tchfo=tchf
      gm1=  max(200.d0,abs(gma))
      xchf=x
      CALL chf1(gm1,hdavg,p,qppchf,xchf,alp)
!
      icount=1
! ensure that there is a positive solution for tchf
      tchf=tsat+0.5d0
      psat=satprs(tchf)
      fcn=fchf(tchf,tl,tsat,qppchf,ca,hforc,psat,p)
      IF (fcn.GT.0.d0) THEN
! generate first quess at the solution for tchf
        htctmp=hforc*(tw-tl)/(tw-tsat)
        htc=htctmp+hnucb
        tchf=  min(qppchf/htc,100.0d0)+tsat
        tchf=  max(tchf,tsat+.5d0)
        psat=satprs(tchf)
        fcn=fchf(tchf,tl,tsat,qppchf,ca,hforc,psat,p)
        t1=tchf
        f1=fcn
! generate second point for secient(newton's) method
        dtchf=1.d0
        DO WHILE (.TRUE.)
          tchf=tchf+dtchf
          psat=satprs(tchf)
          fcn=fchf(tchf,tl,tsat,qppchf,ca,hforc,psat,p)
          IF (fcn.EQ.0.d0) GOTO 1000
          t2=tchf
          f2=fcn
          IF (f2.NE.f1) GOTO 90
          dtchf=dtchf*2.0d0
        ENDDO
   90   dtchf=  min(-(t2-t1)*f1/(f2-f1),50.d0)
        DO WHILE (.TRUE.)
          icount=icount+1
          tchf=tchf+dtchf
```

```
        IF (abs(dtchf).LT.epsol) GOTO 1000
        IF (icount.GT.imax) GOTO 900
        psat=satprs(tchf)
        fcn=fchf(tchf,tl,tsat,qppchf,ca,hforc,psat,p)
        IF (fcn.EQ.0.d0) GOTO 1000
! this checks to see if the solution has been bounded yet
        IF (fcn/f1.LT.0.d0) GOTO 175
        IF (fcn.LT.0.d0) THEN
! solution still to the left, store & move backward
         t2=t1
         f2=f1
         t1=tchf
         f1=fcn
         IF (f2.EQ.f1) THEN
          dtchf=-epsol-5.d0
          GOTO 1001
         ENDIF
        ELSE
! solution still to the right, store & move forward
         t1=t2
         f1=f2
         t2=tchf
         f2=fcn
         IF (f2.EQ.f1) THEN
          dtchf=epsol+5.d0
          GOTO 1001
         ENDIF
        ENDIF
! use newton's method (secient) till solution is bounded
        dtchf=-(t2-t1)*f1/(f2-f1)
 1001   CONTINUE
       ENDDO
! solution has been bounded, store bounding value in proper place
  175  IF (fcn.GT.0.d0) THEN
        t1=tchf
        f1=fcn
       ELSE
        t2=tchf
        f2=fcn
       ENDIF
! do initial pass after bounding using last tchf if its is possible
! solution, otherwise use bi-section.
       IF (t1.LT.tchfo.AND.tchfo.LT.t2) THEN
        tchf=tchfo
       ELSE
        tcnf=0.5d0*(t1+t2)
       ENDIF
       icbs=-1
! secient solution technique
       DO WHILE (.TRUE.)
        icount=icount+1
        icbs=icbs+1
        psat=satprs(tchf)
        fcn=fchf(tchf,tl,tsat,qppchf,ca,hforc,psat,p)
        IF (abs(fcn).LT.1.d-12*abs(qppchf)) GOTO 1000
        IF (fcn.GT.0.d0) THEN
```

```
          t1=tchf
          f1=fcn
        ELSE
          t2=tchf
          f2=fcn
        ENDIF
        dtchfo=dtchf
        dtchf=0.5d0*(t2-t1)
! if the secient method has been used isecmx-times, use bi-section once
        IF (icbs.GE.isecmx) THEN
          icbs=-1
        ELSEIF (f2.NE.f1) THEN
          dtchf=-(t2-t1)*f1/(f2-f1)
          IF (abs(dtchfo-dtchf).LT.1.d-3*epsol) THEN
            dtchf=0.5d0*(t2-t1)
          ENDIF
        ENDIF
        tchf=t1+dtchf
        IF (abs(dtchf).LT.epsol) GOTO 1000
        IF (icount.GT.imax) GOTO 900
      ENDDO
! failure to converge to tchf in imax iterations
  900 CONTINUE
      WRITE (itty,910)
      WRITE (iout,920) tw,tsat,tl,qppchf
      WRITE (itty,920) tw,tsat,tl,qppchf
      WRITE (iout,940) t1,tchf,t2
      WRITE (itty,940) t1,tchf,t2
      WRITE (iout,960) f1,fcn,f2
      WRITE (itty,960) f1,fcn,f2
  910 FORMAT (' ','warning tchf failed to converge')
  920 FORMAT (' ','tw =',e12.5,'tsat =',e12.5,'tl =',e12.5,'qchf =',  &
     &  e12.5)
  940 FORMAT (' ','t1 =',e12.5,'tchf =',e12.5,'t2 =',e12.5)
  960 FORMAT (' ','f1 =',e12.5,'fcn =',e12.5,'f2 =',e12.5)
      CALL error(1,'*chf* tchf failed to converge*')
! converged to tchf
    ENDIF
 1000 RETURN
    END SUBROUTINE chf

    SUBROUTINE chf1(gm,hdavg,p,qppchf,x,alp)
    USE HTPar
    IMPLICIT NONE
!
!   Declaration Generated by genImpDecs.pl 5/98
    REAL(sdk) alp,expnt,fpbar,gm,gmcgs,gmcx,hdavg,hdcgs,hinter,hpbar,  &
   & p,pbar,qanty,qchf2,qppchf,rhdcx,star,x,x1,x2,xpnt,xx,y1,y2
!
!
!   biasi correlation
!   ref-- l. biasi, et al.,"studies on burnout,"
!        energia nucleare,vol.14,n.9,sept.,1967.
!
!
    INCLUDE 'constant.h'
```

```
!
!    hinter linearly intepolates
!
     hinter(x1,x2,y1,y2,xx)=y1+(y2-y1)*(xx-x1)/(x2-x1)
!
!!   STATEMENT FUNCTION
     star(qanty,expnt)=exp(expnt*log(qanty))
!
!
     pbar=p*.00001d0
     hdcgs=hdavg*100.0d0
     xpnt=-0.4d0
     IF (hdcgs.LT.1.0d0) xpnt=-0.6d0
     fpbar=0.7249d0+0.099*pbar*exp(-0.032d0*pbar)
     hpbar=-1.159d0+0.149*pbar*exp(-0.019d0*pbar)+8.99d0*pbar/(10.0d0   &
    &+pbar*pbar)
     gmcgs=gm*.1d0
     gmcgs=  max(gmcgs,10.d0)
     rhdcx=star(hdcgs,xpnt)
     qppchf=3.78d07*hpbar*(one-x)*star(gmcgs,-.6d0)
     gmcgs=  max(gmcgs,30.d0)
     gmcx=star(gmcgs,-.16667d0)
     qchf2=1.883d07*(fpbar*gmcx-x)*gmcx
     qppchf=rhdcx*  max(qppchf,qchf2,0.0d0)
     IF (alp.GE.alpchf) THEN
       qppchf=hinter(alpchf,alpcut,qppchf,1.d0,alp)
       qppchf=  max(1.d0,qppchf)
!
     ENDIF
     RETURN
     END SUBROUTINE chf1

     SUBROUTINE ChfBWR(hfg,hd,p,alp,vlz,vvz,rol,rov,sig,cl,cpl,tw,tl    &
    &                  ,tsat,tchf,qppchf,ca,hforc,hnucb,gm,gma)
!
!    BEGIN MODULE USE
     USE Io
     USE EosNoInline
     USE Cise
!
     IMPLICIT NONE
!
!    Declaration Generated by genImpDecs.pl 5/98
     INTEGER(sik) icbs,icount,imax,isecmx
!
!    Declaration Generated by genImpDecs.pi 5/98
     REAL(sdk) alp,ca,cl,cpl,cpv,cv,dtchf,epsol,f1,f2,fchf,fcn,gm,gma,  &
    & grava,hd,hfg,hforc,hnucb,htc,htctmp,p,pdrat,psat,qppchf,rol,      &
    & rom,rov,sig,t1,t2,tchf,tchfo,tl,tsat,tv,tw,visl,visv,vlc,vlz,vmc, &
    & vmz,vvc,vvz,xchf,y1,y2,y3,y4,y5,y6,y7,y8
     REAL(sdk) gint,gcgs,pp,bl,a,b,brac,qppsub,qppfc,qpppb,qppb,dtsub   &
    &,tau,slip,tenm6,qint,gdum,x
!
     INCLUDE 'constant.h'
!
!
```

```
!
!*********          prologue
!
!*** title: ChfBWR  -- departure from nucleate boiling
!
!   authors--
!         lasl and inel trac development
!         groups
!
!   Purpose--
!
!   Decide whether the fluid conditions will support nucleate boiling
!   Call Chf1BWR to evaluate the Biasi correlation.
!   Evaluate CISE-GE if ichf is 3.
!   Interpolate to Zuber at low flow.
!   Calculate tchf using the same method as in subroutine chf..
!
!
!
!*** subroutine argument definition
!
!    ca     =  chen correlation coefficient
!    gma    =  local fluid mass flux
!    hd     =  local hydraulic diameter
!    hfg    =  local latent heat
!    hforc  =  forced convection ht coefficient
!    hnucb  =  nucleate boiling ht coefficient
!    p      =  local pressure
!    tl     =  local liquid temperature
!    tsat   =  local saturation temperature of the fluid
!    tw     =  local wall temperture
!    xe     =  equilibrium thermodynaic quality in CiseM
!    xc     =  critical quality in CiseM
!    cl     =  liquid conductivity
!    cpl    =  liquid specific heat
!    rol    =  liquid density
!    rov    =  vapor density
!    sig    =  surface tension
!    vlz    =  liquid velocity
!    vvz    =  vapor velocity
!    alp    =  void fraction
!    gm     =  abs(mass flux)
!    gma    =  mass flux (plus or minus)
!
!
!*** direct outputs --
!
!    tchf   =  peak heat flux temperature
!    qppchf =  peak heat flux at critical point
!
!    this subroutine evaluates the critical heat flux based on
!    a local conditions formulation
!    chf options
!      ichf=1 zuber/biase local only
!      ichf=2 =1 + biase critical quality above g.lt.-700,g.gt.300.
!      ichf=2 = 1 + biase critical quality if g>300.
```

```
!       ichf=3 = 1 + cise-ge critical quality if g>300.
!
!
!
!     local parameters
!
      PARAMETER (tenm6=1.d-06)
!
! added initialiations to keep compilers happy! - cgg 8/11/99
      DATA cpv,cv,dtchf,f1,f2,fcn,grava,htc,htctmp,pdrat,psat,          &
     & rom,t1,t2,tchfo,tv,visl,visv,vlc,vmc,                            &
     & vmz,vvc,xchf,y1,y2,y3,y4,y5,y6,y7,y8,                              &
     & gint,gcgs,pp,bl,a,b,brac,qppsub,qppfc,qpppb,qppb,dtsub,          &
     & tau,slip,gdum,x/47*0.0_sdk/
! end - cgg 8/11/99
!
      DATA epsol/0.5d0/,imax/35/,isecmx/7/
      fchf(y1,y2,y3,y4,y5,y6,y7,y8)=y4-(y6*(y1-y2)+y5*exp(.24d0*log(y1   &
     & -y3)+.75d0*log(  max(y7-y8,1.d-5)))*(y1-y3))
!
      qint(x) = 3.0d0*x*x - 2.0d0*x*x*x
!
!     standard rod chf package
!     zuber pool boiling chf correlation modified by void
!     fraction for low flow
!     biasi correlation for high flows
!
      IF (gma.GT.300.d0 .and. bls.GT.1.d0) THEN
        igrang = 1
      ELSE
        igrang = 0
      ENDIF
!
      IF (gma.LT.-700.d0 .OR. gma.GT.300.d0) THEN
!
!     high mass flux range
!
        CALL Chf1BWR(gm,hfg,hd,p,qppchf)
!
        IF (iichf.EQ.3 .and. igrang.EQ.1) THEN
!
!     limit cise-ge critical quality to g=1.e6 lbm/ft2-hr
!
          gcgs = gm*737.36d0*tenm6
          IF (gcgs.GT.1.d0) gcgs = 1.d0
          pp = p*145.04d-6 - 600.0d0
          bl = bls*3.281d0*12.0d0
          a = ((-0.285d0*gcgs+0.907d0)*gcgs-1.233d0)*gcgs + 1.055d0 -
     1       0.013d0* (pp/400.0d0)**2
          b = (-35.464d0*gcgs+78.873d0)*gcgs + 17.98d0
          IF (nrow.EQ.8) b = b/1.12d0
          xc = a*bl/ (b+bl)
          xc = xc* (1.24d0*rfi)
!
        ENDIF
!
```

```
      IF (xe.GT.xc) GOTO 1000
      GOTO 140
!
      ENDIF
!
!     Zuber low flow chf calculation
!
      qppsub = 0.0d0
      brac = (sig*gc* (rol-rov)/ (rov*rov))**0.25d0
      dtsub = tsat - tl
      IF (dtsub.LE.0.0d0) GOTO 110
      tau = 2.625d0*sqrt(sig/ (gc* (rol-rov)))/brac
      qppsub = 2.0d0*cl*dtsub/sqrt(pi*tau*cl/ (rol*cpl))
!
!     .1179 = .9 * .131
!
!      .9 is vertical surface correction. ref. j. h. lienhard and
!     v. k. dhir, hydrodynamic prediction of peak boiling heat
!     fluxes from finite bodies, j. heat transfer, vol 95, 1973,
!     pp. 152-158.
!
  110 CONTINUE
      qpppb = (1.0d0-alp)* (0.1179d0*hfg*rov*brac+qppsub)
      IF (gm.GT.100.d0) THEN
       IF (vlz.EQ.0.d0) GOTO 120
       slip = vvz/vlz
       IF (slip.GE.0.d0) GOTO 120
      ENDIF
!
!       counter current flow - use zuber chf correlation with the
!     1.-alp modification.
!
      qppchf = qpppb
      GOTO 140

  120 CONTINUE
!
!     low flow transition to high flow chf
!
      IF (gma.GT.0.d0) GOTO 130
       gdum = 700.d0
       CALL Chf1BWR(gdum,hfg,hd,p,qppfc)
       gint = (gm-100.d0)/600.d0
       qppchf = qint(gint)* (qppfc-qpppb) + qpppb
       GOTO 140
!
  130 CONTINUE
      gdum = 300.d0
      CALL Chf1BWR(gdum,hfg,hd,p,qppfc)
      gint = (gm-100.d0)/200.d0
      qppchf = qint(gint)* (qppfc-qpppb) + qpppb
!
!
  140 CONTINUE
!
!
```

```
      icount=1
! ensure that there is a positive solution for tchf
      tchf=tsat+0.5d0
      psat=satprs(tchf)
      fcn=fchf(tchf,tl,tsat,qppchf,ca,hforc,psat,p)
      IF (fcn.GT.0.d0) THEN
! generate first quess at the solution for tchf
       htctmp=hforc*(tw-tl)/(tw-tsat)
       htc=htctmp+hnucb
       tchf=  min(qppchf/htc,100.0d0)+tsat
       tchf=  max(tchf,tsat+.5d0)
       psat=satprs(tchf)
       fcn=fchf(tchf,tl,tsat,qppchf,ca,hforc,psat,p)
       t1=tchf
       f1=fcn
! generate second point for secient(newton's) method
       dtchf=1.d0
       DO WHILE (.TRUE.)
        tchf=tchf+dtchf
        psat=satprs(tchf)
        fcn=fchf(tchf,tl,tsat,qppchf,ca,hforc,psat,p)
        IF (fcn.EQ.0.d0) GOTO 1000
        t2=tchf
        f2=fcn
        IF (f2.NE.f1) GOTO 90
        dtchf=dtchf*2.0d0
       ENDDO
  90   dtchf=  min(-(t2-t1)*f1/(f2-f1),50.d0)
       DO WHILE (.TRUE.)
        icount=icount+1
        tchf=tchf+dtchf
        IF (abs(dtchf).LT.epsol) GOTO 1000
        IF (icount.GT.imax) GOTO 900
        psat=satprs(tchf)
        fcn=fchf(tchf,tl,tsat,qppchf,ca,hforc,psat,p)
        IF (fcn.EQ.0.d0) GOTO 1000
! this checks to see if the solution has been bounded yet
        IF (fcn/f1.LT.0.d0) GOTO 175
        IF (fcn.LT.0.d0) THEN
! solution still to the left, store & move backward
         t2=t1
         f2=f1
         t1=tchf
         f1=fcn
         IF (f2.EQ.f1) THEN
          dtchf=-epsol-5.d0
          GOTO 1001
         ENDIF
        ELSE
! solution still to the right, store & move forward
         t1=t2
         f1=f2
         t2=tchf
         f2=fcn
         IF (f2.EQ.f1) THEN
          dtchf=epsol+5.d0
```

```
            GOTO 1001
           ENDIF
          ENDIF
! use newton's method (secient) till solution is bounded
          dtchf=-(t2-t1)*f1/(f2-f1)
 1001     CONTINUE
        ENDDO
! solution has been bounded, store bounding value in proper place
  175   IF (fcn.GT.0.d0) THEN
          t1=tchf
          f1=fcn
        ELSE
          t2=tchf
          f2=fcn
        ENDIF
! do initial pass after bounding using last tchf if its is possible
! solution, otherwise use bi-section.
        IF (t1.LT.tchfo.AND.tchfo.LT.t2) THEN
          tchf=tchfo
        ELSE
          tchf=0.5d0*(t1+t2)
        ENDIF
        icbs=-1
! secient solution technique
        DO WHILE (.TRUE.)
          icount=icount+1
          icbs=icbs+1
          psat=satprs(tchf)
          fcn=fchf(tchf,tl,tsat,qppchf,ca,hforc,psat,p)
          IF (abs(fcn).LT.1.d-12*abs(qppchf)) GOTO 1000
          IF (fcn.GT.0.d0) THEN
            t1=tchf
            f1=fcn
          ELSE
            t2=tchf
            f2=fcn
          ENDIF
          dtchf=0.5d0*(t2-t1)
! if the secient method has been used isecmx-times, use bi-section once
          IF (icbs.GE.isecmx) THEN
            icbs=-1
          ELSEIF (f2.NE.f1) THEN
            dtchf=-(t2-t1)*f1/(f2-f1)
          ENDIF
          tchf=t1+dtchf
          IF (abs(dtchf).LT.epsol) GOTO 1000
          IF (icount.GT.imax) GOTO 900
        ENDDO
! failure to converge to tchf in imax iterations
  900   CONTINUE
        WRITE (itty,910)
        WRITE (iout,920) tw,tsat,tl,qppchf
        WRITE (itty,920) tw,tsat,tl,qppchf
        WRITE (iout,940) t1,tchf,t2
        WRITE (itty,940) t1,tchf,t2
        WRITE (iout,960) f1,fcn,f2
```

```
      WRITE (itty,960) f1,fcn,f2
  910  FORMAT (' ','warning tchf failed to converge')
  920  FORMAT (' ','tw =',e12.5,'tsat =',e12.5,'tl =',e12.5,'qchf =',  &
     &  e12.5)
  940  FORMAT (' ','t1 =',e12.5,'tchf =',e12.5,'t2 =',e12.5)
  960  FORMAT (' ','f1 =',e12.5,'fcn =',e12.5,'f2 =',e12.5)
      CALL error(1,'*chf* tchf failed to converge*')
!  converged to tchf
     ENDIF
 1000 RETURN
     END SUBROUTINE ChfBWR
!
     SUBROUTINE Chf1BWR(gmab,hfg,hd,p,qppchf)
!
!    BEGIN MODULE USE
     USE Cise
     IMPLICIT NONE
!
!    Declaration Generated by genImpDecs.pl 5/98
     REAL(sdk) a2,b2,xpnt,fpbar,gmcgs,gmcx,hd,hdcgs,hinter,hpbar      &
     & ,p,pbar,qchf1,qppchf,xc1,gmab
     REAL(sdk) d1p4,one6,seven6,blcgs,hfg,hfgcgs,hdcx,x1p6dz,p6dz
!
!
!    Purpose: Evaluate the biasi correlation or boiling length CHF
!    ref--  I. biasi, et al.,"studies on burnout,"
!         energia nucleare,vol.14,n.9,sept.,1967.
!
!
     INCLUDE 'constant.h'
!
     PARAMETER (x1p6dz=1.6_sdk)
     PARAMETER (p6dz=.6_sdk)
!    hinter linearly intepolates
!
!



     pbar = p/1.0d05
     hdcgs = hd*100.0d0
     gmcgs = gmab*0.1d0
     xpnt = 0.4d0
     IF (hdcgs.LT.1.d0) xpnt = 0.6d0
!
     fpbar = 0.7249d0 + 0.099d0*pbar*exp(-0.032d0*pbar)
     gmcx = gmcgs**0.16667d0
     hdcx = hdcgs**xpnt
     qppchf = 1.883d07* (fpbar/gmcx-xe)/ (hdcx*gmcx)
     hpbar = -1.159d0 + .149d0*pbar*exp(-.019d0*pbar) +
    1       8.99d0*pbar/ (10.d0+pbar*pbar)
     qchf1 = 3.78D7*hpbar* (1.d0-xe)/ (hdcx*gmcgs**.6d0)
!    biasi = .02d0
     IF (qchf1.GT.qppchf) THEN
!       biasi = .01d0
       qppchf = qchf1
     ENDIF
```

```
!
      IF (igrang.EQ.1 .AND. iichf.EQ.2) THEN
! Biasi in critical quality form
! Pillips, Shumway & Chu 20th ASME/AIChE National Heat Transfer
! Conference, Milwaukee, WI, August 2-5, 1981.
        d1p4 = hdcgs**1.4d0
        one6 = 1.d0/6.d0
        seven6 = 7.d0*one6
        blcgs = bls*100.d0
        hfgcgs = hfg/1000.d0
        a2 = fpbar/gmcgs**one6
        b2 = 1.328D-4*d1p4*hfgcgs*gmcgs**seven6
        xc = a2*blcgs/ (b2+blcgs)
        b2 = 6.614D-5*d1p4*hfgcgs*gmcgs**x1p6dz/hpbar
        xc1 = blcgs/ (b2+blcgs)
        xc = max(xc,xc1)
        xc = xc*per*sqrt(rfi)
!
      ENDIF
      RETURN
      END SUBROUTINE Chf1BWR

      SUBROUTINE htcor(tw,row,cpw,cw,emis,hdavg,p,tsat,hfg,cl,cv,cpl,    &
     & cpv,rol,rov,rom,visl,visv,alpc,sig,grava,vlz,vlzp,vlt,vvz,vvc,    &
     & vvt,vmz,vmc,vmt,tl,tv,ischan,isheatr,pdrat,ichf,el,ev,drovdt,     &
     & droldt,eva,rova,pa,tssn,hlsat,hssat,hgamq,hl,hv,qppchf,tchf,      &
     &ridreg,stnu,tld,htdfcl,htdfcv,dtube,dbaff,fracl,iLevF,dzLevF,      &
     &alppr,alpmr)
!
!     BEGIN MODULE USE
      USE Io
      USE GlobalDat
      USE CompTyp
      USE Flt
      USE EosNoInline
      USE Cise
      USE HTPar
      USE Limiters
!
      IMPLICIT NONE
!
!     Declaration Generated by genImpDecs.pl 5/98
      INTEGER(sik) iavgfg,ibrown,ichf,icon,inface,iqench,itralp,nfilm
!
!     Declaration Generated by genImpDecs.pl 5/98
      REAL(sdk) al0,alp,alp1,alpc,alpmin,alps,alpt,alpto,beta,ca,        &
     & cl,coef1,coef2,cpl,cpv,cpvf,cpw,cv,cvf,cw,deltv,dfhl,dmax,dmin,   &
     & droldt,drovdt,dtl1,dtv1,dtvs,el,emis,ev,eva,expnt,expnt1,expnt2,  &
     & factor,fracev,fsubc,fx,gamfac,gamma,gm,gm1xx,gma,gr,grava,gravg,  &
     & h2,hdavg,hf,hfg,hforc,hgamq,hgamqc,hgamqm,hgamqo,hgmax,hgmin,     &
     & hgmqto,hinter,hl,hlf,hllam,hlmax,hlmin,hlo,hlsat,hlt,hlto,hlturb, &
     & hnc1,hnc2,hnucb,hrl,hssat,htdfcl,htdfcv,htot,hv,hv2,hva,hvcond,   &
     & hvmax,hvmin,hvnc,hvo,hvsat,hvsav,hvt,hvto,hvturb,p,pa,pdrat,prl,  &
     & prm,prv,prvf,qanty,qanty1,qanty2,qchen,qchfo,qchmax,qchmin,ql1,   &
     & qppchf,qppmin,qt1,qtrans,qv1,ratio,rem,revf,ridreg,ridrg,ridrgo
!
```

```
      REAL(sdk) ridrgt,rof,rol,rom,rov,rova,rovf,rovw,row,salp,si,sig,  &
     & slp,ssav,star,stard,starsr,stnu,stnulm,svalow,svaup,tchf,       &
     & tchfo,tchfs,tchfxx,tfilm,tl,tld,tmin,tsat,tssn,tv,tvf,tw,visl,  &
     & vism,visv,visvf,visvw,vla,vlc,vlt,vlz,vlzmn,vlzp,vmc,vmt,vmz,vva, &
     & vvc,vvt,vvz,wa,wf,ws,x,x1,x2,x3,xchen,xchfx,xcon,xflow,xn,xnu,  &
     & xpe,xpelm,xx,y1,y2
!
      REAL(sdk) vdum,rfact,hfact,xfact,regl,dtube,froud,deltw,dbaff,    &
     & fracl,slip,hvap,hliq,qsteam
!
      REAL(sdk) :: dzLevF, alppr, alpmr
      INTEGER(sik) :: iLevF
!
! spent fuel pool heat transfer
      REAL(sdk) hvsfp
      LOGICAL ischan,isheatr
!
!
!     this subroutine has been rewritten in a more structured
!     manner to aid in 1) ease of understanding for problem
!     solving and 2) future vectorization.
!
!     r. nelson       july 1985
!
!     this subroutine computes an appropriate heat transfer coef.
!     based on local surface temperature, local coolant conditions,
!     geometry, and surface conditions
!
!*************** nomenclature ***********************************
!
! variable  i/o          definition
!
! tw       i   wall(i.e. surface) temperature, k
! row      i   wall density evaluated at tw, kg/m**3
! cpw      i   wall specific heat evaluated at tw, j/kg-k
! cw       i   wall thermal conductivity evaluated at tw, watt/m-k
! emis     i   wall emissivity, dimensionless
! hdavg    i   hydraulic diameter, m
! p        i   fluid pressure, nt/m**2
! tsat     i   saturation temperature, k
! hfg      i   latent heat, j/kg
! cl       i   liquid thermal conductivity evaluated at tl, watt/m-k
! cv       i   vapor thermal conductivity evaluated at tv, watt/m-k
! cpl      i   liquid specific heat evaluated at tl, j/kg-k
! cpv      i   vapor specific heat evaluated at tv, j/kg-k
! rol      i   liquid density evaluated at tl, kg/m**3
! rov      i   vapor density evaluated at tv, kg/m**3
! rom      i   mixture density, kg/m**3
! visl     i   liquid viscosity evaluated at tl, nt-s/m**2
! visv     i   vapor viscosity evaluated at tv, nt-s/m**2
! alpc     i   void fraction, dimensionless
! sig      i   surface tension, nt/m
! grava    i   average value of cos theta (angle pipe orientated)
! vlz      i   axial liquid velocity, m/s
! vlzp     i   axial liquid velocity associated with any drops
! vlt      i   not used
```

```
! vvz      i   axial vapor velocity, m/s
! vvc      i   not used
! tld      o   Liquid temperature at bubble departure, (K)
! vvt      i   not used
! vmz      i   mixture velocity, m/s
! vmc      i   not used
! vmt      i   not used
! tl       i   liquid temperature, k
! tv       i   vapor temperature, k
! ischan   l   true if htcor is performing for a CHAN component.
! pdrat    i   not used
! ichf     i   chf flag
!              = 0, always use single mixture correlation
!              = 1, use full boiling surface with Zuber/Biasi CHF
!              = 2, CHF use Zuber at low flow
!                   Biasi or Biasi critical quality for high flow
!              = 3, same as 2 but use CISE-GE critical quality
! boillen  o   height above the void=0 position.
! igrang   i   = 0, conditions out of range for BWR critical quality
!              = 1, check BWR critical quality
! el       i   liquid internal energy evaluated at tl
! ev       i   vapor internal energy evaluated at tv
! drovdt   i   derivative of vapor density w.r.t. temperature
! droldt   i   derivative of liquid density w.r.t. temperature
! eva      i   air internal energy evaluated at tv
! rova     i   air density evaluated at tv, kg/m**3
! pa       i   air pressure, nt/m**2
! tssn     i   saturation temperature of steam at partial p, k
! hlsat    i   saturation enthalpy of liquid at partial p, k
! hssat    i   saturation enthalpy of steam at partial p, k
!
! hgamq    i,o  subcooled boiling heat xfer coeff., watt/m**2-k
! hl       i,o  liquid heat transfer coefficient (htc), watt/m**2-k
! hv       i,o  vapor heat transfer coefficient, watt/m**2-k
! tchf     o   critical heat flux (chf) temperature, k
! ridreg   i,o  heat transfer regime flag,
!              = 1, forced convection to single phase liquid
!              = 2, nucleate boiling
!              = 3, transition boiling
!              = 4, film boiling
!              = 6, natural or forced convection to single phase vapor
!              = 7, convection to mixture (used only if ichf = 0 )
!              = 11, condensation
!              = 12, natural convection to single phase liquid
! stnu     o   Stanton Number
!
!***********************************************************************
!
      INCLUDE 'constant.h'
!
      DATA alpmin,xchen,alps/0.05d0,0.71d0,0.7d0/
      DATA ibrown/0/
      DATA gamfac/100.0d0/,al0/0.1d0/,slp/10.0d0/
      DATA stnulm/0.0065d0/,xpelm/70000.0d0/
      DATA iavgfg/1/
!
```

```
!!   STATEMENT FUNCTION
     star(qanty,expnt)=exp(expnt*log(qanty))
!
!!   STATEMENT FUNCTION
     stard(qanty1,expnt1,qanty2,expnt2)=exp(expnt1*log(qanty1)+expnt2  &
    & *log(qanty2))
!
!!   STATEMENT FUNCTION
     starsr(qanty)=sqrt(sqrt(qanty))
!
!    hinter is a function used to linearly interpolate htc's
!    between heat transfer regimes.
!
     hinter(x1,x2,y1,y2,xx)=y1+(y2-y1)*(xx-x1)/(x2-x1)
!
!    tchfxx is a simple expression to determine a
!    reasonable value for tchf for use in the nucleate
!    boiling regime when tchf is not used in the flow regimes
!    it is based upon a thom-type functional relationship,
!    i.e. q=c*(tw-tsat)**2, with respect to wall temperature
!
     tchfxx(x1,x2,x3)=x1*sqrt(x3/x2)
!
!    Initialize citical and equilibrium quaility.
     xc = one
     xe = zero
!    check for static calculation
!
!
     IF (isttc.EQ.1) THEN
      hl=0.d0
      hv=0.d0
      hgamq=0.d0
     ELSEIF (iconht.EQ.1) THEN
      hl=htcwl
      hv=htcwv
      hgamq=0.0d0
      ridreg=0.d0
     ELSEIF (ihtcn.EQ.0) THEN
      alp=  min(alpc,alp3-1.d-8)
      vlzmn=  max(hgvmn,abs(vlzp))
      IF (ihtav.NE.0) alp=alpc
!
!    save last time step values or values   which maybe changed
!    during the calculation
!
      ridrgo=ridreg
      svalow=alow
      svaup=aup
      hlo=hl
      hvo=hv
      hgamqo=hgamq
      tchfo=tchf
      IF (iavgfg.NE.0) qchfo=qppchf
!
!    initialize flags and values of some variables
```

```
!
      ridreg=0.d0
      iqench=0
      inface=0
      hvsav=zero
      hgamq=zero
      vlc=zero
      xflow=zero
      qppchf=zero
      slip=one
      tchf=tssn+0.5d0
      vla=abs(vlz)
      vva=abs(vvz)
      gma=vmz*rom
      vmz=abs(vmz)
      gravg=abs(grava)
      gm=abs(gma)
      IF (vlz.NE.zero) THEN
       slip=abs(vvz/vlz)
       IF (slip.LE.zero) slip=one
      ENDIF
      hliq=el+p/rol
      hvap=ev+p/rov
      htot=hliq
      wa=rova/rov
      ws=1.0d0-wa
      hva=eva+pa/(rova+1.0d-20)
      hf=hlsat
      hvsat=wa*hva+ws*hssat
! Calculate equalibrium quality for BWR CHF before DO WHILE loop
! which uses alp instead of alpc. Requirement CHAN 6.
!
     IF (ischan .OR. ichf .GT. 1) THEN
      IF (alpc.GT.zero) xflow=one/(one+(one-alpc)/alpc*rol/(rov     &
   &      *slip))
      IF (hvap.GT.hliq) htot=hliq+xflow*(hvap-hliq)
!
      IF (hvsat.GT.hlsat) xe=(htot-hf)/(hvsat-hlsat)
     ENDIF
     IF (iLevF.NE.1) THEN
!
!    check for an interface in the cell
!
       IF (aup.GE.alpchf) THEN
        IF (alow.LT.alpcut) THEN
         IF (.NOT.(alp.LE.alow.OR.alp.GE.aup)) THEN
          IF (aup-alow.GE.1.d-4) THEN
!
!    located an interface in the cell
!
            iqench=1
            x1=(aup-alpchf)/(alpcut-alpchf)
            x1= min(x1,1.d0)
            fx=(3.d0-2.d0*x1)*x1**2
            alow=fx*alow+(1.d0-fx)*alp
!
```

```
!     inface is the flag used to tell if an interface is present.
!     inface=0 indicates no interface.  inface=1 indicates an
!     interface is present and the first pass thru the logic.  this
!     initial pass thru the logic is made using alow or a modified alow
!     as the cell void fraction.  inface=2 indicates an interface
!     is present and the second pass thru the logic.  this second pass
!     thru the logic uses aup as the cell void fraction.  the two
!     results are then averaged in a weighted manner for the cell.
!
              inface=1
              alpto=alp
              alp=alow
              vlzmn=hgvmn
             ENDIF
            ENDIF
           ENDIF
          ENDIF
         ELSE
!
!     Level flag indicates that there is a level in this cell.
!
!
!     inface is the flag used to tell if a level is present.
!     inface=0 indicates no level.  inface=3 indicates a
!     level is present and the first pass thru the logic.  this
!     initial pass thru the logic is made using alpmr.
!     inface=4 indicates a level
!     is present and the second pass thru the logic.  this second pass
!     thru the logic uses alppr as the cell void fraction.  the two
!     results are then averaged in a weighted manner for the cell
!     based on dzLevF (i.e. the fraction of the heat transfer
!     cell below the level).
!
        alpto=alp
        alp=alpmr
        inface=3
       ENDIF
       DO WHILE (.TRUE.)
        itralp=0
        IF (.NOT.(alp.LE.alpchf.OR.alp.GE.alpcut)) THEN
         IF (.NOT.(tw.LT.tssn.OR.tw.LT.tl)) THEN
!
!     set the itralp flag to indicate iteration for an alp between
!     alpchf and alpcut and no condensation or wall-to-liquid
!     heat transfer.  set alp=alpchf for initial pass.
!
           itralp=1
           alpt=alp
           alp=alpchf
          ENDIF
         ENDIF
         DO WHILE (.TRUE.)
          alp1=  max(0.d0,alp3-alp)
!
!     the icon flag indicates whether condensation is present or not.
!     icon=1, condensation present.  icon=0, no condensation.
```

```
!
        icon=0
        IF (alp.GT.zero) xflow=one/(one+(one-alp)/alp*rol/(rov      &
    &     *slip))
!
!   calculate actual enthalpy
!
        IF (hvap.GT.hliq) htot=hliq+xflow*(hvap-hliq)
!   the peclet number = xpe
        xpe=  max(1.0d0,gm*hdavg*cpl/cl)
        tld=tssn
        stnu=0.0d0
!
!   calculate equilibrium quality
!
        x=0.d0
        IF (hvsat.GT.hlsat) x=(htot-hf)/(hvsat-hlsat)
!   note: the above calculation is still not correct, but it is lot
!       better than what was originally in the code.
        x=  max(0.0d0,  min(0.999999d0,x))
!
!
!   check for condensation (ridreg=11) or two-phase to wall
!   using chen with s=0 (ridreg=1) or natural circulation (ridreg=12)
!
        IF ((tw.LT.tssn.AND.alp.GT.alpmin).OR.isheatr) THEN
          IF (tw.GE.tv) THEN
            IF (iqench.EQ.0.AND.tw.GE.tl) GOTO 800
          ENDIF
          IF (tw.GE.tssn) GOTO 800
!
!   film condensation regimes
!
!   use chen correlation with s=0 for condensation hl
!
        x=alp*rov/(alp*rov+(1.0d0-alp)*rol)
        xc=x
!
!   if x.lt.xchen, no condensation and consider fluid-to-wall
!
        IF (isheatr) THEN
          deltw=ABS(tw-tssn)
          deltw=MAX(0.1_sdk,deltw)
!
!   correlation for liquid crossflow over tube banks
!   d.q.kern, process heat transfer, new york:mcgraw-hill,1960.
          vdum=MAX(vla,0.2_sdk)
          hl=0.36_sdk*cl/hdavg*(hdavg*rol*vdum/visl)**0.55_sdk*    &
    &       (cpl*visl/cl)**0.333_sdk
!
!   condensation on tube banks
!   get cross flow condensation parameters
!   t.fujii,h.uehara c.kurata,"laminar filmwise condensation on
!   a horizontal cylinder",int. j. h&mass transfer,peragmon press
!     1972,vol.15,pp235.
          rfact=SQRT(rol*visl/(rov*visv))
```

```
            hfact=cl*deltw/(visl*hfg)
            regl=rol*vmz*dtube/visl
            xfact=0.9_sdk*(one+one/(rfact*hfact))**0.333_sdk
!
            IF (grava.EQ.1.0_sdk) THEN
!
!   horizontal tube banks- cross flow and nusselt condensation
              froud=(vmz*vmz+0.01_sdk)/(gc*dtube)
              hv=xfact*cl/dtube                              &
     &         *SQRT(SQRT(one+0.276_sdk/(xfact**4*froud*hfact))*regl)
              ridreg=11.1_sdk
!
            ELSEIF (grava.EQ.0.0_sdk) THEN
!
!   vertical tube banks- cross flow and nusselt condensation
!   m.m.chen,"an analytical study of laminar film condensation
!   part 1- flat plates", trans of asme,feb 1961, pp48.
              hv=0.943_sdk*(hfg*cl**3*rol*rol*gc                &
     &         /(visl*dbaff*(deltw)))**0.25_sdk
              hv=hv+xfact*cl*SQRT(regl)/dtube
              ridreg=11.2_sdk
            ENDIF
!
!   redefine hl to include all condensation
!
            hl=fracl*hl+(one-fracl)*hv*(deltw)/MAX(1.0e-6_sdk,tl-tw)
            hv=zero
            GOTO 4000
!
          ELSEIF (x.GE.xchen) THEN
!
!   condensation present, set flag and evaluate
!
            xc=xchen
            icon=1
            hl=zero
!
!   nusselt analysis from bird, stewart, and lightfoot -- pp. 417,418
!
            coef1=cl*cl*cl*9.8d0*rol*rol
            coef2=(tssn-tw)*ylv
            coef2=  max(1.d-2,coef2)
            hvcond=.9428d0*sqrt(sqrt(coef1*hfg/(coef2*visl)))
            h2=.003d0*sqrt(coef1*coef2/(hfg*visl**3))
            wf=  min(1.0d0,  max(0.0d0,(ylv-.2d0)/1.8d0))
            hvcond=hvcond*(1.d0-wf)+  max(hvcond,h2)*wf
            hvcond=hvcond*(tssn-tw)/  max(abs(tv-tw),.01d0)
!
            prv=cpv*visv/cv
            deltv=  max(abs(tw-tv),1.0d-20)
!
!   calculate natural convection htc
!
            hvnc=0.13d0*cv*star(rov*rov*gc*deltv*prv/(visv*visv*tv), &
     &         0.333d0)
!
```

```
!   dittus-boelter htc, first evaluating vapor film properities.
!   note that the prandtl number coefficient is different for
!   cooling (0.3) and heating (0.4).
!
          tvf=0.5d0*(tw+tv)
          IF (tvf.LT.tssn) THEN
           cvf=cv
           prvf=prv
           revf=  max(rov*vva*hdavg/visv,1.0d-20)
          ELSE
           rovf=rov+drovdt*(tvf-tv)
           visvf=viscv(hvap,p,rovf,tvf,pa)
           cpvf=cpvv1(tvf,p,pa)
           cvf=thcv(hvap,p,rovf,tvf,pa)
           revf=  max(rovf*vva*hdavg/visvf,1.0d-20)
           prvf=cpvf*visvf/cvf
          ENDIF
          hvturb=0.023d0*cvf*stard(revf,0.8d0,prvf,0.3d0)/hdavg
          hv=  max(hvnc,hvturb)
          hvsav=hv
          qsteam=hvsav*(tw-tv)
          qsteam=  max(qsteam,0.0d0)
          hvcond=  max(hv,hvcond)
         ENDIF
!
         CALL chen(alp,cl,cpl,cpv,cv,gm,gravg,hdavg,hfg,hforc,     &
     &     hnucb,p,pdrat,rol,rom,rov,sig,tl,tsat,tv,tw,visl,visv,   &
     &     vlc,vla,vmc,vmz,vvc,vva,xc,ca,ridreg)
!
         ridreg=1.0d0
         hv=0.d0
!
!   natural convection (regime 12)
!   ref. j. p. holman, heat transfer, 3rd edition, p. 218.
!   (all properties evaluated at tl except rho in order
!   to save calls to thermo. rho evaluated from taylor series.)
!
         tfilm=.5d0*(tw+tl)
         rof=rol+droldt*(tfilm-tl)
         beta=-droldt/rof
         gr=  max(gc*beta*abs(tw-tl)*rof**2*hdavg**3/visl**2,1.0d  &
     &     -20)
         prl=cpl*visl/cl
!
!   laminar flow
!
         hnc1=.59d0*starsr(gr*prl)*cl/hdavg
!
!   turbulent flow
!
         hnc2=.10d0*star(gr*prl,.3333d0)*cl/hdavg
         hl=  max(hnc1,hnc2,hforc)
         IF (hl.NE.hforc) ridreg=12.0d0
!
!   condensation?  if not, liquid-to-wall has been used.
!
```

```
          IF (icon.EQ.1) THEN
!
!    setting final condensation values for hl and hv.
!
            ridreg=11.0d0
            xcon=x
            IF (x.GT.1.0d0) THEN
              hl=0.d0
              hv=hvcond
              xcon=1.0d0
            ELSE
              hl=hinter(xchen,1.0d0,hl,0.0d0,xcon)
              hv=hinter(xchen,1.0d0,hv,hvcond,xcon)
            ENDIF
          ENDIF
          GOTO 4000
        ENDIF
!
!    heat transfer from wall to fluid
!
  800     CONTINUE
        IF (alp.GE.alpchf) THEN
!
!    forced convection to vapor (regime 6)
!
          ridreg=6.0d0
          hl=zero
          hv=zero
          hgamq=zero
          prv=cpv*visv/cv
          deltv=  max(abs(tw-tv),1.0d-20)
!
!    calculate natural convection htc
!
          hvnc=0.13d0*cv*star(rov*rov*gc*deltv*prv/(visv*visv*tv),   &
     &       0.333d0)
!
!    sieder-tate htc, first evaluating vapor film properities
!
          tvf=0.5d0*(tw+tv)
          IF (tvf.LT.tssn) THEN
            factor=1.0d0
            xn=0.4d0
            cvf=cv
            prvf=prv
            revf=  max(rov*vva*hdavg/visv,1.0d-20)
          ELSE
            rovf=rov+drovdt*(tvf-tv)
            rovw=rov+drovdt*(tw-tv)
            visvf=viscv(hvap,p,rovf,tvf,pa)
            visvw=viscv(hvap,p,rovw,tw,pa)
            cpvf=cpvv1(tvf,p,pa)
            cvf=thcv(hvap,p,rovf,tvf,pa)
            revf=  max(rovf*vva*hdavg/visvf,1.0d-20)
            prvf=cpvf*visvf/cvf
            factor=star(visvf/visvw,0.14d0)
```

```
          xn=0.333d0
        ENDIF
        hvturb=0.023d0*cvf*stard(revf,0.8d0,prvf,xn)*factor/hdavg
        hvsfp=8.0d0*cvf/hdavg
        hv= max(hvnc,hvturb,hvsfp)
        hvsav=hv
        qsteam=hvsav*(tw-tv)
        qsteam= max(qsteam,0.0d0)
!
!   over the void range of alpchf to alp2 interpolate hv between
!   single-phase and two-phase results.  also interpolate hl
!   between zero and two-phase results.  loop back to the two-phase
!   coding to obtain two phase results and interpolate there.
!
        IF (alp.GE.alp2) GOTO 4000
      ENDIF
      IF (ichf.EQ.0) THEN
!
!   no boiling curve used
!   forced convection to a mixture (regime 7)
!   natural convection - mcadams
!   turbulent convection - dittus-boelter
!
        ridreg=7.0d0
        vism=one/(xflow/visv+(one-xflow)/visl)
        prm=cpl*visl/cl
        rem= max(gm*hdavg/vism,1.0d-20)
!
!   laminar forced convection htc
!
        hllam=4.0d0*cl/hdavg
!
!   turbulent forced convection htc
!
        hlturb=0.023d0*cl/hdavg*stard(rem,0.8d0,prm,0.4d0)
        hl= max(hllam,hlturb)
        hv=zero
        IF (alp.GT.alpcut) THEN
!   natural convection
          deltv= max(abs(tw-tv),1.0d-20)
          prv=cpv*visv/cv
          hvnc=0.13d0*cv*star(rov*rov*gc*deltv*prv/(visv*visv*tv), &
     &      0.333d0)
!   dittus-boelter htc, first evaluating vapor film properities
          tvf=0.5d0*(tw+tv)
          IF (tvf.LT.tssn) THEN
            cvf=cv
            prvf=prv
            revf= max(rov*vva*hdavg/visv,1.0d-20)
          ELSE
            rovf=rov+drovdt*(tvf-tv)
            visvf=viscv(hvap,p,rovf,tvf,pa)
            cpvf=cpvv1(tvf,p,pa)
            cvf=thcv(hvap,p,rovf,tvf,pa)
            revf= max(rovf*vva*hdavg/visvf,1.0d-20)
            prvf=cpvf*visvf/cvf
```

```
          ENDIF
          hvturb=0.023d0*cvf*stard(revf,0.8d0,prvf,0.4d0)/hdavg
          hv= max(hvnc,hvturb)
          hl=hinter(alpcut,alp3,hl,0.d0,alp)
          hv=hinter(alpcut,alp2,hv,hvsav,alp)
        ENDIF
!
!   check whether wall-to-fluid or fluid-to-wall
!
      ELSEIF (.NOT.(tw.GT.tsat.AND.tw.GT.tl)) THEN
!
!   convection to a single-phase liquid (regime 1)
!
        ridreg=1.0d0
        hv=0.d0
        CALL chen(alp,cl,cpl,cpv,cv,gm,gravg,hdavg,hfg,hforc,     &
     &     hnucb,p,pdrat,rol,rom,rov,sig,tl,tsat,tv,tw,visl,visv,   &
     &     vlc,vla,vmc,vmz,vvc,vva,x,ca,ridreg)
!
!   natural convection (regime 12)
!   ref. j. p. holman, heat transfer, 3rd edition, p. 218
!   (all properties evaluated at tl except rho in order
!   to save calls to thermo. rho evaluated from taylor series.)
!
        tfilm=.5d0*(tw+tl)
        rof=rol+droldt*(tfilm-tl)
        beta=-droldt/rof
        gr= max(gc*beta*abs(tw-tl)*rof**2*hdavg**3/visl**2,1.0d  &
     &     -20)
        prl=cpl*visl/cl
!
!   laminar flow
!
        hnc1=.59d0*starsr(gr*prl)*cl/hdavg
!
!   turbulent flow
!
        hnc2=.10d0*star(gr*prl,.3333d0)*cl/hdavg
        hl= max(hnc1,hnc2,hforc)
        IF (hl.NE.hforc) ridreg=12.0d0
!
!   estimate tchf for flow regime maps
!
        gm1xx= max(0.d0,abs(gma))
        xchfx= max(0.d0,x)
        CALL chf1(gm1xx,hdavg,p,qppchf,xchfx,alp)
        IF (tw.GT.tl) THEN
          tchf=tl+tchfxx(tw-tl,hl*(tw-tl),qppchf)
          tchf= max(tchf,tw+1.0d0)
        ENDIF
        IF (alp.GT.alpbr) THEN
          hl=hinter(alpbr,1.d0,hl,0.d0,alp)
          hv=hinter(alpbr,1.d0,0.d0,hvsav,alp)
        ENDIF
!
!   checking to see if in film boiling last time.  if so,
```

```
!    work from right to left on boiling.  if not, work from
!    left to right (standard procedure) on the boiling curve.
!
         ELSEIF (ridrgo.EQ.4.0d0) THEN
!
!    since film boiling existed last time step,
!    working from right to left on the boiling curve.
!    determine minimum film boiling temperature
!
           ratio=(cl*rol*cpl/(cw*row*cpw))
           tfilm=.5d0*(tw+tsat)
           rof=rov+drovdt*(tfilm-tv)
           CALL tmsfb(p,gm,tl,tmin,x,ratio,rof,cv,hfg,tsat,rov,rol,   &
     &       visv,sig,itmin)
           dtvs=  max((tv-tsat),zero)
!
!    determine if in film boiling
!
           IF (tw.GT.tmin.OR.xe.GT.xc) THEN
!
!    film boiling (regime 4):  calculate vapor htc at tw
!    Keep the logic in reg 4 when xe>xc and calculate hv at tfilm.
!    The code was in here when xe>xc but tw was close to tsat.
            ridreg=4.0d0
            IF (ischan) THEN
             tfilm=max(tw,tmin)
            ELSE
             tfilm = tw
            ENDIF
            CALL hvfilm(tfilm,tsat,tv,cpv,hfg,sig,rol,rov,cv,visv    &
     &         ,vva,alp,vla,hdavg,hv)
!
            alp1=  max(0.d0,alp3-alp)
!
!    radiative heat transfer htc to liquid - interface at tsat
!    Adding chan to the following logic helps meet requirement CHAN 7.9
!
            IF (nencl.GT.0.OR.ischan) THEN
             hrl=0.0d0
            ELSE
             hrl=alp1*5.6697d-08*emis*(tw**4-tsat**4)/(tw-tsat)
            ENDIF
!
!    liquid htc
!
            CALL hlfilm(alp,tfilm,tsat,tl,tv,rol,rov,cl,cv,cpl,cpv   &
     &         ,visl,visv,sig,hfg,vlz,vvz,hdavg,dfhl,hlf)
           dfhl=dfhl*fdfhl
           hrl=hrl+hlf
           hl=(hrl*(tw-tsat)/(tw-tl))+dfhl
           IF (alp.LT.0.10d0.AND.tl.LT.tsat) THEN
            hgamq=hl*  max(zero,  min(1.0d0,slp*(al0-alp)))
           ELSE
            hgamq=0.0d0
           ENDIF
           IF (alp.GT.alpcut) hv=hinter(alpcut,alp2,hv,hvsav,alp)
```

```
      ELSE
!
!   setup to determine nucleate boiling (regime 2) information
!   find s for alp = alps if alp .gt. alps
!
        IF (alp.GE.alps) THEN
         CALL chen(alps,cl,cpl,cpv,cv,gm,gravg,hdavg,hfg,      &
     &           hforc,hnucb,p,pdrat,rol,rom,rov,sig,tl,tsat,tv,tw,    &
     &           visl,visv,vlc,vla,vmc,vmz,vvc,vva,x,ca,ridreg)
         ssav=supres
        ENDIF
        CALL chen(alp,cl,cpl,cpv,cv,gm,gravg,hdavg,hfg,hforc,    &
     &         hnucb,p,pdrat,rol,rom,rov,sig,tl,tsat,tv,tw,visl,visv,  &
     &         vlc,vla,vmc,vmz,vvc,vva,x,ca,ridreg)
!
!   if alp gt. alps adjust the suppression factor, s
!
        IF (alp.GE.alps) THEN
         salp=supres
         IF (salp.GT.0.0d0) THEN
          si=  min(supres,ssav)
          supres=si*(alpcut-alp)/(alpcut-alps)
          supres=  max(supres,0.0d0)
          hnucb=hnucb*supres/salp
          ca=ca*supres/salp
         ENDIF
        ENDIF
        tfilm=.5d0*(tw+tl)
        rof=rol+droldt*(tfilm-tl)
        beta=-droldt/rof
        gr=  max(gc*beta*abs(tw-tl)*rof**2*hdavg**3/visl**2,1.d  &
     &         -20)
        prl=cpl*visl/cl
        hnc1=.59d0*starsr(gr*prl)*cl/hdavg
        hnc2=.1d0*star(gr*prl,.3333d0)*cl/hdavg
        hforc=  max(hnc1,hnc2,hforc)
        qppchf=hforc*(tw-tl)
!
!   test for dryout
!
        IF (alp.GE.alpcut) THEN
         tchf=tsat+0.5d0
         qppchf=0.5d0*hforc
         IF (tw.GT.tchf) GOTO 3200
        ENDIF
!
!   determine chf
!
        gm1xx=  max(200.d0,abs(gma))
        xchfx=x
!
!   if invan .ne. 0, calculate tchf.  invan is a flag used to
!   determine tw used in the flow regimes.  if invan=0, tsat
!   is used as the temp at which inverted annular flow begins.
!   if invan .ne. 0, tchf is used to define when inverted annular
!   flow begins.
```

```
!
            IF (invan.EQ.0) THEN
              qchen=hforc*(tw-tl)+hnucb*(tw-tsat)
              IF (.NOT.ischan) THEN
                CALL chf1(gm1xx,hdavg,p,qppchf,xchfx,alp)
!
!     test for chf by comparing heat fluxes when invan=0.
!
                IF (qchen.GE.qppchf) GOTO 3200
!
              ELSE
                bls = boillen
                iichf = ichf
                CALL ChfBWR(hfg,hdavg,p,alp,vla,vva,rol,rov,sig      &
     &                    ,cl,cpl,tw,tl,tsat,tchf,qppchf,ca          &
     &                    ,hforc,hnucb,gm,gma)
!
!     When igrang=1 conditions are OK for using a critical quality test
                IF (igrang.EQ.1) THEN
!
                 xc = xc*0.95
!
!     the 0.95 adds hysteresis to the return to nucleate boiling.
!
!     go  to film boiling if xe .gt. xc
!
!       WRITE(iout,777)tw,alpc,xflow,xe,xc
!  777 FORMAT ('htcor-1394; tw,alp,xflow,xe,xc',5g13.3)
                IF (xe.GE.xc) THEN
                  ridreg=4.0
                  CALL hvfilm(tw,tsat,tv,cpv,hfg,sig,rol,rov,cv    &
     &                       ,visv,vva,alp,vla,hdavg,hv)
!
                  alp1=  max(0.d0,alp3-alp)
!
!     radiative heat transfer htc to liquid - interface at tsat
!     Adding chan to the following logic helps meet requirement CHAN 7.9
!
                  IF (nencl.GT.0.OR.ischan) THEN
                    hrl=0.0d0
                  ELSE
                    hrl=alp1*5.6697d-08*emis*(tw**4-tsat**4)       &
     &                                  /(tw-tsat)
                  ENDIF
!
!     liquid htc
!
                  CALL hlfilm(alp,tw,tsat,tl,tv,rol,rov,cl,cv      &
     &                    ,cpl,cpv,visl,visv,sig,hfg,vlz,vvz,hdavg   &
     &                    ,dfhl,hlf)
                  dfhl=dfhl*fdfhl
                  hrl=hrl+hlf
                  hl=(hrl*(tw-tsat)/(tw-tl))+dfhl
                  IF (alp.LT.0.10d0.AND.tl.LT.tsat) THEN
                    hgamq=hl*  max(zero,  min(1.0d0,slp            &
     &                        *(al0-alp)))
```

```
                ELSE
                 hgamq=0.0d0
                ENDIF
                IF (alp.GT.alpcut) hv=hinter(alpcut,alp2,hv     &
     &                                ,hvsav,alp)
                GOTO 4000
               ENDIF
              ENDIF
              IF (tw.GE.tchf.OR.qchen.GT.qppchf) GOTO 3200
             ENDIF
!    define tchf in this manner to avoid calculation of tchf in
!    nucleate boiling.  this defination will create a slight
!    discontinuity in hv at chf.  used only when invan=0.
!
              tchf=tsat+tchfxx(tw-tsat,qchen,qppchf)
             ELSE
              tchf=tchfo
              CALL chf(alp,ca,cl,cpl,cpv,cv,gma,grava,hdavg,hfg,    &
     &          hforc,hnucb,ichf,p,pdrat,qppchf,rol,rom,rov,sig,     &
     &          tchf,tl,tsat,tv,tw,visl,visv,vlc,vla,vmc,vmz,vvc,    &
     &          vva,x)
              IF (tchf-tsat.LE.0.5d0) THEN
               tchf=tsat+0.5d0
               qppchf=0.5d0*hforc
              ENDIF
              IF (tw.GT.tchf) GOTO 3200
             ENDIF
!
!    complete nucleate boiling (regime 2) information since it
!    has now been determined to be the correct correlation
!
             ridreg=2.0d0
             fsubc=(tw-tsat)/(tw-tl)
             factor=  min(fsubc,one)
             hl=hforc+hnucb*factor
!
             IF (tw.GT.tsat.AND.tl.LT.tsat) THEN
!  the modified nusselt number = xnu
              xnu=hl*(tw-tl)*hdavg/cl
              CALL SubBoil(alp,hgamq,hl,stnulm,tl,tld,tssn,xnu,xpe  &
     &                ,xpelm)
             ELSE
              hgamq=0.0d0
             ENDIF
!
!    interpolate hv between 0 at tw = tsat and transition
!    boiling value at tw = tchf.
!
             CALL hvnb(tw,tsat,tv,cpv,hfg,sig,rol,rov,cv,visv,vva,   &
     &          alp,vla,hdavg,hv2)
             wf=  min(1.0d0,  max(0.0d0,((tw-tsat)/(tchf-tsat))**2))
             hv=wf*hv2
!
             IF (hv*(tw-tv).GT.0.d0) THEN
              IF (hv*(tw-tv).LT.0.5d0*hl*(tw-tl)) THEN
               hl=(hl*(tw-tl)-hv*(tw-tv))/(tw-tl)
```

```
              ELSE
                hl=0.5d0*hl
                hv=hl*(tw-tl)/(tw-tv)
              ENDIF
            ENDIF
            IF (alp.GT.alpcut) THEN
              hl=hinter(alpcut,alp3,hl,0.d0,alp)
              hv=hinter(alpcut,alp2,hv,hvsav,alp)
            ENDIF
            GOTO 4000
!
!   in transition boiling
!   if not already calculated, i.e. invan .ne. 0, calculate
!   tchf for transition boiling correlation. also, we have by-passed
!   the calculation of hgam in nucleate boiling so it must be done
!   here. you may wish to note the comments on this model above.
!
!   first the hgam work.
!
 3200         IF (tw.GT.tsat.AND.tl.LT.tsat) THEN
              factor= min((tw-tsat)/(tw-tl),one)
              hl=hforc+hnucb*factor
!  the modified nusselt number = xnu
              xnu=hl*(tw-tl)*hdavg/cl
              CALL SubBoil(alp,hgamqc,hl,stnulm,tl,tld,tssn,xnu,xpe   &
     &                    ,xpelm)
            ELSE
              hgamqc=0.0d0
            ENDIF
!
!   doing chf portion of work.
!
            IF (invan.EQ.0) THEN
              IF (.NOT.(alp.GE.alpcut.OR.qppchf.LE.0.0d0)) THEN
                tchf=tchfo
                CALL chf(alp,ca,cl,cpl,cpv,cv,gma,grava,hdavg,hfg,   &
     &                  hforc,hnucb,ichf,p,pdrat,qppchf,rol,rom,rov,sig,    &
     &                  tchf,tl,tsat,tv,tw,visl,visv,vlc,vla,vmc,vmz,vvc,   &
     &                  vva,x)
                IF (tchf-tsat.GE.0.5d0) GOTO 3400
              ENDIF
              tchf=tsat+0.5d0
              qppchf=0.5d0*hforc
            ENDIF
!
!   calculate vapor htc at tmin for transition boiling
!
 3400         CALL hvfilm(tmin,tsat,tv,cpv,hfg,sig,rol,rov,cv,visv,    &
     &         vva,alp,vla,hdavg,hv)
!
!   calculate radiative heat transfer contribution to the liquid
!   htc - interface assumed to be at tsat
!   Adding chan to the following logic helps meet requirement CHAN 7.9
!
            IF (ischan) THEN
              hrl = zero
```

```
            ELSE
              hrl=alp1*5.6697d-08*emis*(tmin**4-tsat**4)/(tmin-tsat)
            ENDIF
!
!   calculate liquid htc
!
            CALL hlfilm(alp,tmin,tsat,tl,tv,rol,rov,cl,cv,cpl,cpv,   &
     &        visl,visv,sig,hfg,vlz,vvz,hdavg,dfhl,hlf)
            dfhl=dfhl*fdfhl
!
!   calculate total heat flux at the minimum stable film
!   boiling point (tmin).
!
            hlmin=hrl*(tmin-tsat)/(tmin-tl)+dfhl
            hvmin=hv
!
!   calculate hgamq at tmin for transition boiling
!
            IF (alp.LT.0.10d0.AND.tl.LT.tsat) THEN
              hgamqm=hlmin* max(zero, min(1.0d0,slp*(al0-alp)))
            ELSE
              hgamqm=zero
            ENDIF
            IF (alp.GE.alpcut) THEN
              hvmin=hinter(alpcut,alp2,hvmin,hvsav,alp)
              hvmin= max(hvmin,0.0d0)
            ENDIF
            qppmin=hlmin*(tmin-tl)+hvmin*(tmin-tv)+hlf*(tmin-tsat)
            gamma=((tw-tmin)/(tchf-tmin))**2
            qtrans=(gamma*qppchf+(one-gamma)*qppmin)
!
!   transition boiling (regime 3)
!
            ridreg=3.0d0
            CALL hvfilm(tw,tsat,tv,cpv,hfg,sig,rol,rov,cv,visv,vva,  &
     &        alp,vla,hdavg,hv)
            IF (alp.GT.alpcut) THEN
              hv=hinter(alpcut,alp2,hv,hvsav,alp)
              qtrans=hinter(alpcut,alp3,qtrans,hvsav*(tw-tv),alp)
              hforc=hinter(alpcut,alp3,hforc,0.d0,alp)
            ENDIF
            IF (gamma.LE.1.0.OR.(.NOT.ischan)) THEN
              hl=(qtrans-hv*(tw-tv))/abs(tw-tl)
              hl= max(hl,0.0d0)
              hgamq= max(gamma*hgamqc+(one-gamma)*hgamqm,0.0d0)
              hgamq= min(hl,hgamq)
            ELSE
              hl=dfhl
              hgamq=zero
            ENDIF
          ENDIF
        ELSE
!
!   working from left to right on the boiling curve
!   setup to determine nucleate boiling (regime 2) information
!   find s for alp = alps if alp .gt. alps
```

```
!
        IF (alp.GE.alps) THEN
         CALL chen(alps,cl,cpl,cpv,cv,gm,gravg,hdavg,hfg,hforc,  &
     &     hnucb,p,pdrat,rol,rom,rov,sig,tl,tsat,tv,tw,visl,visv,  &
     &     vlc,vla,vmc,vmz,vvc,vva,x,ca,ridreg)
         ssav=supres
        ENDIF
        CALL chen(alp,cl,cpl,cpv,cv,gm,gravg,hdavg,hfg,hforc,    &
     &   hnucb,p,pdrat,rol,rom,rov,sig,tl,tsat,tv,tw,visl,visv,   &
     &   vlc,vla,vmc,vmz,vvc,vva,x,ca,ridreg)
!
!   if alp gt. alps adjust the suppression factor, s
!
        IF (alp.GE.alps) THEN
         salp=supres
         IF (salp.GT.0.0_sdk) THEN
          si= min(supres,ssav)
          supres=si*(alpcut-alp)/(alpcut-alps)
          supres= max(supres,0.0_sdk)
          hnucb=hnucb*supres/salp
          ca=ca*supres/salp
         ENDIF
        ENDIF
!
!   use natural convection correlations as a lower
!   bound on chen's flow term.
!   ref. j. p. holman, heat transfer, 3rd edition, p. 218.
!   (all properties evaluated at tl except rho in order
!   to save calls to thermo. rho evaluated from taylor series.)
!
        tfilm=.5d0*(tw+tl)
        rof=rol+droldt*(tfilm-tl)
        beta=-droldt/rof
        gr= max(gc*beta*abs(tw-tl)*rof**2*hdavg**3/visl**2,1.d   &
     &    -20)
        prl=cpl*visl/cl
!
!   laminar flow
!
        hnc1=.59d0*starsr(gr*prl)*cl/hdavg
!
!   turbulent flow
!
        hnc2=.1d0*star(gr*prl,.3333d0)*cl/hdavg
!
!   use the maximum of the three for the flow term
!
        hforc= max(hnc1,hnc2,hforc)
        qppchf=hforc*(tw-tl)
!
!   test for dryout
!
        IF (alp.GE.alpcut) THEN
         tchf=tsat+0.5d0
         qppchf=0.5d0*hforc
         IF (tw.GT.tchf) GOTO 1800
```

```
        ENDIF
!
!   determine chf
!
        gm1xx= max(200.d0,abs(gma))
        xchfx=x
!
!   if invan .ne. 0, calculate tchf.  invan is a flag used to
!   determine tw used in the flow regimes.  if invan=0, tsat
!   is used as the temp at which inverted annular flow begins.
!   if invan .ne. 0, tchf is used to define when inverted annular
!   flow begins.
!
        IF (invan.EQ.0) THEN
          qchen=hforc*(tw-tl)+hnucb*(tw-tsat)
          IF (.NOT.ischan) THEN
            CALL chf1(gm1xx,hdavg,p,qppchf,xchfx,alp)
!   test for chf by comparing heat fluxes when invan=0.
!
            IF (qchen.GE.qppchf) GOTO 1800
          ELSE
            bls = boillen
            iichf = ichf
            CALL ChfBWR(hfg,hdavg,p,alp,vla,vva,rol,rov,sig      &
     &                  ,cl,cpl,tw,tl,tsat,tchf,qppchf,ca        &
     &                  ,hforc,hnucb,gm,gma)
!
!   When igrang=1 conditions are OK for using a critical quality test
            IF (igrang.EQ.1) THEN
!
              IF (ridrgo.GE.3.d0) xc = xc*0.95
!
!   the 0.95 adds hysteresis to the return to nucleate boiling.
!
!   go  to film boiling if xe .gt. xc
!     WRITE(iout,778)tw,alpc,xflow,xe,xc
! 778 FORMAT ('htcor-1700; tw,alp,xflow,xe,xc',5g13.3)
!
              IF (xe.GE.xc) GOTO 1800
            ENDIF
            IF (tw.GT.tchf.OR.qchen.GT.qppchf) GOTO 1800
          ENDIF
!
!
!   define tchf in this manner to avoid calculation of tchf in
!   nucleate boiling.  this defination will create a slight
!   discontinuity in hv at chf.  used only when invan=0.
!
          tchf=tsat+tchfxx(tw-tsat,qchen,qppchf)
        ELSE
          tchf=tchfo
          CALL chf(alp,ca,cl,cpl,cpv,cv,gma,grava,hdavg,hfg,      &
     &       hforc,hnucb,ichf,p,pdrat,qppchf,rol,rom,rov,sig,tchf,  &
     &       tl,tsat,tv,tw,visl,visv,vlc,vla,vmc,vmz,vvc,vva,x)
          IF (tchf-tsat.LE.0.5d0) THEN
            tchf=tsat+0.5d0
```

```
            qppchf=0.5d0*hforc
           ENDIF
           IF (tw.GT.tchf) GOTO 1800
          ENDIF
!
!    complete nucleate boiling (regime 2) information since it
!    has now been determined to be the correct correlation
!
          ridreg=2.0d0
          fsubc=(tw-tsat)/(tw-tl)
          factor= min(fsubc,one)
          hl=hforc+hnucb*factor
!
          IF (tw.GT.tsat.AND.tl.LT.tsat) THEN
!  the modified nusselt number = xnu
           xnu=hl*(tw-tl)*hdavg/cl
           CALL SubBoil(alp,hgamq,hl,stnulm,tl,tld,tssn,xnu,xpe    &
    &                ,xpelm)
          ELSE
           hgamq=0.0d0
          ENDIF
!
!    interpolate hv between 0 at tw = tsat and transition
!    boiling value at tw = tchf.
!
          CALL hvnb(tw,tsat,tv,cpv,hfg,sig,rol,rov,cv,visv,vva,alp,  &
    &       vla,hdavg,hv2)
          wf= min(1.0d0,  max(0.0d0,((tw-tsat)/(tchf-tsat))**2))
          hv=wf*hv2
!
          IF (hv*(tw-tv).GT.0.d0) THEN
           IF (hv*(tw-tv).LT.0.5d0*hl*(tw-tl)) THEN
            hl=(hl*(tw-tl)-hv*(tw-tv))/(tw-tl)
           ELSE
            hl=0.5d0*hl
            hv=hl*(tw-tl)/(tw-tv)
           ENDIF
          ENDIF
          IF (alp.GT.alpcut) THEN
           hl=hinter(alpcut,alp3,hl,0.d0,alp)
           hv=hinter(alpcut,alp2,hv,hvsav,alp)
          ENDIF
          GOTO 4000
!
!    determine minimum film boiling temperature
!
 1800       ratio=(cl*rol*cpl/(cw*row*cpw))
          tfilm=.5d0*(tw+tsat)
          rof=rov+drovdt*(tfilm-tv)
          CALL tmsfb(p,gm,tl,tmin,x,ratio,rof,cv,hfg,tsat,rov,rol,   &
    &       visv,sig,itmin)
          dtvs= max((tv-tsat),zero)
!
!    determine if in film boiling
!    TRAC-B forces the logic to regime 4 when xe>xc
          IF (tw.GT.tmin.OR.xe.GT.xc) THEN
```

```
!
!    film boiling (regime 4):  calculate vapor htc at tw
!
           ridreg=4.0d0
           IF (ischan) THEN
            tfilm=max(tw,tmin)
           ELSE
            tfilm = tw
           ENDIF
           CALL hvfilm(tfilm,tsat,tv,cpv,hfg,sig,rol,rov,cv,visv    &
     &          ,vva,alp,vla,hdavg,hv)
!
           alp1=  max(0.d0,alp3-alp)
!
!    radiative heat transfer htc to liquid - interface at tsat
!    Adding chan to the following logic helps meet requirement CHAN 7.9
!
           IF (nencl.GT.0.OR.ischan) THEN
            hrl=0.0d0
           ELSE
            hrl=alp1*5.6697d-08*emis*(tw**4-tsat**4)/(tw-tsat)
           ENDIF
!
!    liquid htc
!
           CALL hlfilm(alp,tfilm,tsat,tl,tv,rol,rov,cl,cv,cpl,cpv   &
     &          ,visl,visv,sig,hfg,vlz,vvz,hdavg,dfhl,hlf)
           dfhl=dfhl*fdfhl
           hrl=hrl+hlf
           hl=(hrl*(tw-tsat)/(tw-tl))+dfhl
           IF (alp.LT.0.10d0.AND.tl.LT.tsat) THEN
            hgamq=hl*  max(zero,  min(1.0d0,slp*(al0-alp)))
           ELSE
            hgamq=zero
           ENDIF
           IF (alp.GT.alpcut) hv=hinter(alpcut,alp2,hv,hvsav,alp)
          ELSE
!
!    in transition boiling
!    if not already calculated, i.e. invan .ne. 0, calculate
!    tchf for transition boiling correlation. also, we have by-passed
!    the calculation of hgam in nucleate boiling so it must be done
!    here. you may wish to note the comments on this model above.
!
!    first the hgam work.
!
           IF (tw.GT.tsat.AND.tl.LT.tsat) THEN
            factor=  min((tw-tsat)/(tw-tl),one)
            hl=hforc+hnucb*factor
!  the modified nusselt number = xnu
            xnu=hl*(tw-tl)*hdavg/cl
            CALL SubBoil(alp,hgamqc,hl,stnulm,tl,tld,tssn,xnu,xpe  &
     &                  ,xpelm)
           ELSE
            hgamqc=0.0d0
           ENDIF
```

```
!
!    doing chf portion of work.
!
            IF (invan.EQ.0) THEN
             IF (.NOT.(alp.GE.alpcut.OR.qppchf.LE.0.0d0)) THEN
              tchf=tchfo
              CALL chf(alp,ca,cl,cpl,cpv,cv,gma,grava,hdavg,hfg,   &
     &          hforc,hnucb,ichf,p,pdrat,qppchf,rol,rom,rov,sig,    &
     &          tchf,tl,tsat,tv,tw,visl,visv,vlc,vla,vmc,vmz,vvc,   &
     &          vva,x)
              IF (tchf-tsat.GE.0.5d0) GOTO 2000
             ENDIF
             tchf=tsat+0.5d0
             qppchf=0.5d0*hforc
            ENDIF
!
!    calculate vapor htc at tmin for transition boiling
!
 2000        CALL hvfilm(tmin,tsat,tv,cpv,hfg,sig,rol,rov,cv,visv,    &
     &       vva,alp,vla,hdavg,hv)
!
!    calculate radiative heat transfer contribution to the liquid
!    htc - interface assumed to be at tsat
!    Adding chan to the following logic helps meet requirement CHAN 7.9
!
            IF (ischan) THEN
             hrl = zero
            ELSE
             hrl=alp1*5.6697d-08*emis*(tmin**4-tsat**4)/(tmin-tsat)
            ENDIF
!
!    calculate liquid htc
!
            CALL hlfilm(alp,tmin,tsat,tl,tv,rol,rov,cl,cv,cpl,cpv,   &
     &        visl,visv,sig,hfg,vlz,vvz,hdavg,dfhl,hlf)
            dfhl=dfhl*fdfhl
!
!    calculate total heat flux at the minimum stable film
!    boiling point (tmin).
!
            hlmin=hrl*(tmin-tsat)/(tmin-tl)+dfhl
            hvmin=hv
!
!    calculate hgamq at tmin for transition boiling
!
            IF (alp.LT.0.10d0.AND.tl.LT.tsat) THEN
             hgamqm=hlmin*  max(zero,  min(1.0d0,slp*(al0-alp)))
            ELSE
             hgamqm=zero
            ENDIF
            IF (alp.GE.alpcut) THEN
             hvmin=hinter(alpcut,alp2,hvmin,hvsav,alp)
             hvmin=  max(hvmin,0.0d0)
            ENDIF
            qppmin=hlmin*(tmin-tl)+hvmin*(tmin-tv)+hlf*(tmin-tsat)
            gamma=((tw-tmin)/(tchf-tmin))**2
```

```
            qtrans=(gamma*qppchf+(one-gamma)*qppmin)
!
!    transition boiling (regime 3)
!
            ridreg=3.0d0
            CALL hvfilm(tw,tsat,tv,cpv,hfg,sig,rol,rov,cv,visv,vva,  &
     &          alp,vla,hdavg,hv)
            IF (alp.GT.alpcut) THEN
             hv=hinter(alpcut,alp2,hv,hvsav,alp)
             qtrans=hinter(alpcut,alp3,qtrans,hvsav*(tw-tv),alp)
             hforc=hinter(alpcut,alp3,hforc,0.d0,alp)
            ENDIF
            IF (gamma.LE.1.0.OR.(.NOT.ischan)) THEN
             hl=(qtrans-hv*(tw-tv))/abs(tw-tl)
             hl= max(hl,0.0d0)
             hgamq= max(gamma*hgamqc+(one-gamma)*hgamqm,0.0d0)
             hgamq= min(hl,hgamq)
            ELSE
             hl=dfhl
             hgamq=zero
            ENDIF
           ENDIF
          ENDIF
!
!    ************************************************************
!    **  transfer here from all correlation evaluations.  **
!    ************************************************************
!
 4000      CONTINUE
!
!    check for interpolation on alp between alpchf and alpcut where
!    alp=alpchf initially as reflected by itralp=1.  if so, reset
!    alp=alpcut and itralp=2 then re-evaluate.  since hgamq is set to
!    zero before these alpha's, no work is done on it here.
!
         IF (itralp.EQ.0) GOTO 4200
         IF (itralp.EQ.2) GOTO 4100
         itralp=2
         hlt=hl
         hvt=hv
         ridrg=ridreg
         alp=alpcut
         hl=zero
         hv=zero
        ENDDO
!
!    now complete interpolation on non-zero itralp where alp
!    was between alpchf and alpcut.
!
 4100    alp=alpt
        hl=hinter(alpchf,alpcut,hlt,hl,alp)
        IF (ridrg.EQ.2.0d0.OR.ridrg.EQ.3.0d0) ridreg=ridrg
        hv=hinter(alpchf,alpcut,hvt,hv,alp)
 4200    hl= max(0.0d0,hl)
        hv= max(0.0d0,hv)
!
```

```
!    adjust htc's so a small vapor mass will not see a large
!    heat flux. the total heat flux is preserved.
!
       IF (.NOT.(alp.GT..15d0.OR.hv.LE.0.0d0)) THEN
        dtl1=tw-tl
        dtv1=tw-tv
        ql1=hl*dtl1
        qv1=hv*dtv1
        qt1=qv1+ql1
        qv1=hinter(.01d0,.15d0,0.0d0,qv1,alp)
        IF (alp.LE..01d0) qv1=0.0d0
        dtl1= max(1.d-10,abs(dtl1))*sign(1.d0,dtl1)
        dtv1= max(1.d-10,abs(dtv1))*sign(1.d0,dtv1)
        ql1=qt1-qv1
        hl=ql1/dtl1
        hv=qv1/dtv1
        hl= max(0.0d0,hl)
        hv= max(0.0d0,hv)
       ENDIF
!
!    is interface not present (inface=0) ?
!
       IF (inface.EQ.0) GOTO 4500
!
!    interface present. first or second pass? if first,
!    save info and make a second pass. if second, make
!    final evaluation.
!
       IF (inface.EQ.1) THEN
        inface=2
        alp=aup
        vlzmn= max(hgvmn,abs(vlzp))
        hlto=hl
        hvto=hv
        hgmqto=hgamq
        ridrgt=ridreg
        tchfs=tchf
        hl=zero
        hv=zero
        hgamq=zero
       ELSEIF (inface.EQ.2) THEN
        GOTO 4400
       ELSEIF (inface.EQ.3) THEN
        alp   = alppr
        hlto  = hl
        hvto  = hv
        hgmqto = hgamq
        ridrgt = ridreg
        tchfs  = tchf
        hl    = zero
        hv    = zero
        hgamq  = zero
        inface = 4
       ELSEIF (inface.EQ.4) THEN
        alp = alpto
        hl  = dzLevF*hlto+(1.0_sdk-dzLevF)*hl
```

```
          IF (icon.EQ.1.AND.genTab(cco)%type.EQ.prizrh) hl=0.0d0
          hv   = dzLevF*hvto   + (1.0_sdk-dzLevF)*hv
          hgamq = dzLevF*hgmqto + (1.0_sdk-dzLevF)*hgamq
          tchf  = dzLevF*tchfs  + (1.0_sdk-dzLevF)*tchf
          IF (ridrgt.EQ.2.0d0.OR.ridrgt.EQ.3.0d0) ridreg=ridrgt
          GOTO 4500
         ENDIF
        ENDDO
!
!    make final evaluation of hl and hv for cell with interface.
!
 4400   alp=alpto
        fx=(aup-alp)/(aup-alow)
        hl=fx*hlto+(1.d0-fx)*hl
        IF (icon.EQ.1.AND.genTab(cco)%type.EQ.prizrh) hl=0.0d0
        hv=fx*hvto+(1.d0-fx)*hv
        hgamq=fx*hgmqto+(1.d0-fx)*hgamq
        tchf=fx*tchfs+(1.d0-fx)*tchf
        IF (ridrgt.EQ.2.0d0.OR.ridrgt.EQ.3.0d0) ridreg=ridrgt
        alow=svalow
        aup=svaup
 4500   IF (.NOT.(ihtav.EQ.0.AND.iabs(istdy).NE.1)) THEN
!
!    average or limit new htc's
!
         IF (.NOT.(hvo.GT.1.d-6.OR.hlo.GT.1.d-6)) THEN
          hlo=hl
          hvo=hv
          hgamqo=hgamq
         ENDIF
         hl=hl*htdfcl
         hv=hv*htdfcv
         IF (iavgfg.EQ.0.OR.iabs(istdy).EQ.1) THEN
          hl=owhtd*hlo+(1.d0-owhtd)*hl
          hv=owhtd*hvo+(1.d0-owhtd)*hv
          hgamq=owhtd*hgamqo+(1.d0-owhtd)*hgamq
          hlt=hlo
          IF (hlt.EQ.0.d0.AND.hvo.EQ.0.d0) hlt=hv
          hvt=hvo
          IF (hvt.EQ.0.d0.AND.hlo.EQ.0.d0) hvt=hl
          IF (hv.GT.hvo) hv=  min(hv,  max(fhtcu*hvo,hlt))
          IF (hv.LT.hvo) hv=  max(hv,fhtcl*hvo)
          IF (hl.GT.hlo) hl=  min(hl,  max(fhtcu*hlo,hvt))
          IF (hl.LT.hlo) hl=  max(hl,fhtcl*hlo)
!     one should realize that hgamq can not be greater than hl.
          IF (hgamq.GT.hl) hgamq=hl
         ELSE
          dmax=fudge1**(  min(delt*freq1,explim))
          dmin=fudge2**(  min(delt*freq2,explim))
          IF (hgamqo.EQ.0.d0.AND.hgamq.NE.0.d0) hgamqo=hgamq
!
!     leave sub-cooled boiling on if it was on during the
!     last cycle and the conditions are still appropriate.
!
          IF (hgamqo.NE.0.d0.AND.hgamq.EQ.0.d0) THEN
           IF (ridrgo.NE.ridreg.OR.tw.LT.tsat.OR.tl.GT.tsat) THEN
```

```
            hgamqo=hgamq
            IF (idiag.GE.2) WRITE (imout,123) genTab(cco)%num,ridrgo &
     &       ,ridreg,tw,tsat,tl
    123      FORMAT (' Subcooled boiling turned off in component',   &
     &       i5,/,2x,'ridrgo ',f4.1,'ridreg ',f4.1,2x,'tw,tsat,tl ' &
     &       ,1pe13.4,1x,1pe13.4,1x,1pe13.4)
           ENDIF
          ENDIF
          hgmax=hgamqo*dmax
          hgmin=hgamqo*dmin
          hgamq= max(hgmin, min(hgmax,hgamq))
          IF (hlo.EQ.0.0d0.OR.hl.EQ.0.0d0) hlo=hl
          hlmax=hlo*dmax
          hlmin=hlo*dmin
          hl= max(hlmin, min(hlmax,hl))
          IF (hvo.EQ.0.0d0.OR.hv.EQ.0.0d0) hvo=hv
          hvmax=hvo*dmax
          hvmin=hvo*dmin
          hv= max(hvmin, min(hvmax,hv))
          IF (qchfo.EQ.0.0d0) qchfo=qppchf
          qchmax=qchfo*dmax
          qchmin=qchfo*dmin
          qppchf= max(qchmin, min(qchmax,qppchf))
         ENDIF
         IF (hv.LT.1.0d-10) hv=0.0d0
         IF (hl.LT.1.0d-10) hl=0.0d0
         IF (hgamq.LT.1.0d-10) hgamq=0.0d0
         alow=svalow
         aup=svaup
        ENDIF
        hgamq= min(hl,hgamq)
        IF (ibrown.EQ.1.AND.hgamq.NE.0.0d0) hgamq=gamfac
        xnu=hl* max(0.0d0,tw-tl)*hdavg/(cl* max(0.01d0,(tssn-tl)))
        stnu=xnu/xpe
       ENDIF
       RETURN
       END SUBROUTINE htcor

       SUBROUTINE hlfilm(alp,tw,tsat,tl,tv,rol,rov,cl,cv,cpl,cpv,visl,   &
      & visv,sig,hfg,vlz,vvz,hd,dfhl,hlf)
       USE HTPar
       IMPLICIT NONE
!
!      Declaration Generated by genImpDecs.pl 5/98
       REAL(sdk) alp,alp1,brac,c1,c2,cl,cpl,cpv,cv,delt,delta,deltw,dfhl, &
      & dfhlp,dro,expnt1,expnt2,hd,hfbb,hfg,hfgp,hlf,qanty,qanty1,qanty2, &
      & rol,rov,sig,stard,starsr,tl,tsat,tv,tw,vdif2,visl,visv,vlz,vvz,   &
      & we,wl,x1,x2
!
!
!      the purpose of this subroutine is to calculate the wall to
!      liquid htc in transition and film boiling.  this subroutine
!      combines the work originally done in subroutines dfht and
!      hvfilm into one subroutine.
!
!
```

```
!    written by
!             ralph nelson
!             q9, code development
!             lanl, june 1986
!
!
!  ***************  nomenclature  ***********************************
!
! variable  i/o           definition
!
! alp       i   cell void fraction
! tw        i   wall(i.e. surface) temperature, k
! tsat      i   saturation temperature, k
! tl        i   liquid temperature, k
! tv        i   vapor temperature, k
! rol       i   liquid density evaluated at tl, kg/m**3
! rov       i   vapor density evaluated at tv, kg/m**3
! cl        i   liquid thermal conductivity evaluated at tl, watt/m-k
! cv        i   vapor thermal conductivity evaluated at tv, watt/m-k
! cpl       i   liquid specific heat evaluated at tl, j/kg-k
! cpv       i   vapor specific heat evaluated at tv, j/kg-k
! visl      i   liquid viscosity evaluated at tl, nt-s/m**2
! visv      i   vapor viscosity evaluated at tv, nt-s/m**2
! sig       i   surface tension, nt/m
! hfg       i   latent heat, j/kg
! vlz       i   axial liquid velocity, m/s
! vvz       i   axial vapor velocity, m/s
! hd        i   hydraulic diameter, m
!
! dfhl      o   liquid heat transfer for dispersed flow, watt/m**2-k
! hl        o   liquid heat transfer coefficient not including
!               dispersed flow effects, watt/m**2-k
!
!  ***********************************************************************
!
!
      INCLUDE 'constant.h'

      DATA c1,c2,we/1.5d0,1.2760d0,4.0d0/
      REAL(sdk) limin,limax
      DATA limin,limax/1.0d-4,3.0d-3/
!
!!    STATEMENT FUNCTION
      stard(qanty1,expnt1,qanty2,expnt2)=exp(expnt1*log(qanty1)+expnt2   &
     & *log(qanty2))
!
!!    STATEMENT FUNCTION
      starsr(qanty)=sqrt(sqrt(qanty))
!
      hlf=0.0d0
      dfhl=0.0d0
!
!     bromley htc
!
      dro=rol-rov
      deltw=  max(tw-tsat,0.001d0)
```

```
      wl=2.0d0*pi*sqrt(sig/(gc*dro))
      hfgp=hfg+0.5d0*cpv*  max(tv-tsat,0.d0)
      hfbb=0.62d0*starsr(cv**3*dro*rov*gc*hfgp/(visv*deltw*wl))
      IF (alp.LT.aflmu) THEN
       hlf=hfbb
       IF (alp.LE.aflml) GOTO 900
      ENDIF
      alp1=alp3-alp
      IF (alp1.GT.0.0d0) THEN
!
!     forsland-rohsenow htc
!
       vdif2=abs(vvz-vlz)**2
       IF (vdif2.GT.0.0d0) THEN
        delta=  min(  max(we*sig/(rov*vdif2),limin),limax)
        delt=abs(tw-tsat)
        IF (delt.GT.0.0d0) THEN
         hfgp=hfg+0.35d0*cpv*(tw-tsat)
         brac=gc*rol*rov*hfgp*cv**3/(delt*visv*delta)
         dfhlp=  max(c1*c2*stard(alp1,0.6667d0,brac,0.25d0),1.0d-20)
!
         dfhl=dfhlp*(tw-tsat)/(tw-tl)
         dfhl=  max(  min(dfhl,hfbb),0.0d0)
         IF (alp.LT.aflmu) THEN
          x1=(alp-aflml)/(aflmu-aflml)
          x2=1.0d0-x1
          hlf=hfbb*(3.d0-2.d0*x2)*x2**2
          dfhl=dfhl*(3.d0-2.d0*x1)*x1**2
         ENDIF
        ENDIF
       ENDIF
      ENDIF
!
  900 CONTINUE
!
      RETURN
      END SUBROUTINE hlfilm

      SUBROUTINE hvfilm(tw,tsat,tv,cpv,hfg,sig,rol,rov,cv,visv,vva,alp,  &
     & vla,hdavg,hv)
      IMPLICIT NONE
!
!     Declaration Generated by genImpDecs.pl 5/98
      REAL(sdk) alp,cpv,cv,deltv,expnt,expnt1,expnt2,hdavg,hfbdr,hfg,hv, &
     & hvnc,prv,qanty,qanty1,qanty2,retp,rol,rov,sig,star,stard,tsat,tv, &
     & tw,visv,vla,vva
!
!
      INCLUDE 'constant.h'
!
!     the purpose of this subroutine is to calculate the vapor heat
!     transfer coefficient. this htc is the maximum of the
!     natural convection and dougall-rohsenow coefficients.
!
!     written by
!           d. a. mandell
```

```
!             safety code development, q-9
!             lasl
!             sept., 1980
!
!    modified by
!              ralph nelson
!              lanl, june 1986
!
!-----------------------------------------------------------------
!
!
! **************  nomenclature  ***********************************
!
! variable  i/o           definition
!
! tw        i   wall(i.e. surface) temperature, k
! tsat      i   saturation temperature, k
! tv        i   vapor temperature, k
! cpv       i   vapor specific heat evaluated at tv, j/kg-k
! hfg       i   latent heat, j/kg
! sig       i   surface tension, nt/m
! rol       i   liquid density evaluated at tl, kg/m**3
! rov       i   vapor density evaluated at tv, kg/m**3
! cv        i   vapor thermal conductivity evaluated at tv, watt/m-k
! visv      i   vapor viscosity evaluated at tv, nt-s/m**2
! vva       i   absolute value of axial vapor velocity, m/s
! alp       i   void fraction, dimensionless
! vla       i   absolute value of axial liquid velocity, m/s
! hdavg     i   hydraulic diameter, m
!
! hv        o   vapor heat transfer coefficient, watt/m**2-k
!
! ********************************************************************
!
!!   STATEMENT FUNCTION
     star(qanty,expnt)=exp(expnt*log(qanty))
!
!!   STATEMENT FUNCTION
     stard(qanty1,expnt1,qanty2,expnt2)=exp(expnt1*log(qanty1)+expnt2   &
    & *log(qanty2))
!
     prv=cpv*visv/cv
     deltv=  max(abs(tw-tv),1.0d-20)
!
!    natural convection htc
!
     hvnc=0.13d0*cv*star(rov*rov*gc*deltv*prv/(visv*visv*tv),0.333d0)
!
!    dougall-rohsenow htc
!
     retp=  max(rov*(vva*alp+vla*(one-alp))*hdavg/visv,1.0d-20)
     hfbdr=0.023d0*cv*stard(retp,0.8d0,prv,0.4d0)/hdavg
     hv=  max(hvnc,hfbdr)
     RETURN
     END SUBROUTINE hvfilm
```

```
      SUBROUTINE hvnb(tw,tsat,tv,cpv,hfg,sig,rol,rov,cv,visv,vva,alp,    &
     & vla,hdavg,hv)
      IMPLICIT NONE
!
!     Declaration Generated by genImpDecs.pl 5/98
      REAL(sdk) alp,alpmax,alpmin,cpv,cv,deltv,expnt,expnt1,expnt2,     &
     & hdavg,hfbdr,hfg,hv,hvnc,hvt,prv,qanty,qanty1,qanty2,retp,rol,rov, &
     & sig,star,stard,tsat,tv,tw,visv,vla,vva,x1
!
!
!
!     the purpose of this subroutine is to calculate the vapor heat
!     transfer coefficient for nucleate boiling. this subroutine is
!     basically the same as hvfilm but has been reproducted to apply
!     only to nucleate boiling due to the addition of the new
!     reflood model.
!
!     written by
!               ralph nelson
!               lanl, feb. 1990
!
!-----------------------------------------------------------------------
!
!
! **************  nomenclature  ***********************************
!
! variable  i/o            definition
!
! tw        i    wall(i.e. surface) temperature, k
! tsat      i    saturation temperature, k
! tv        i    vapor temperature, k
! cpv       i    vapor specific heat evaluated at tv, j/kg-k
! hfg       i    latent heat, j/kg
! sig       i    surface tension, nt/m
! rol       i    liquid density evaluated at tl, kg/m**3
! rov       i    vapor density evaluated at tv, kg/m**3
! cv        i    vapor thermal conductivity evaluated at tv, watt/m-k
! visv      i    vapor viscosity evaluated at tv, nt-s/m**2
! vva       i    absolute value of axial vapor velocity, m/s
! alp       i    void fraction, dimensionless
! vla       i    absolute value of axial liquid velocity, m/s
! hdavg     i    hydraulic diameter, m
!
! hv        o    vapor heat transfer coefficient, watt/m**2-k
!
! ***********************************************************************
      INCLUDE 'constant.h'
!
!!    STATEMENT FUNCTION
      star(qanty,expnt)=exp(expnt*log(qanty))
!
!!    STATEMENT FUNCTION
      stard(qanty1,expnt1,qanty2,expnt2)=exp(expnt1*log(qanty1)+expnt2   &
     & *log(qanty2))
!
      DATA alpmin,alpmax/0.50d0,0.75d0/
```

```
!
      prv=cpv*visv/cv
      deltv= max(abs(tw-tv),1.0d-20)
!
!     natural convection htc
!
      hvnc=0.13d0*cv*star(rov*rov*gc*deltv*prv/(visv*visv*tv),0.333d0)
!
!     dougall-rohsenow htc
!
      retp= max(rov*(vva*alp+vla*(one-alp))*hdavg/visv,1.0d-20)
      hfbdr=0.023d0*cv*stard(retp,0.8d0,prv,0.4d0)/hdavg
      hvt= max(hvnc,hfbdr)
      hv=0.0d0
      IF (alp.GT.alpmin) THEN
       hv=hvt
       IF (alp.LT.alpmax) THEN
        x1=(alpmax-alp)/(alpmax-alpmin)
        hv=hvt*(1.d0-x1)
       ENDIF
      ENDIF
      RETURN
      END SUBROUTINE hvnb
!
      SUBROUTINE SubBoil(alp,hgamq,hl,stnulm,tl,tld,tssn,xnu,xpe,xpelm)
!
!
!     Theshold liquid enthalpy for the onset of significant voids
!       given by the empirical model of Saha & Zuber (Proc. of the
!       Fifth International Heat Transfer Conference, 4, 1976)
!
!     Heat flux partititioning due to the mechanistic model of
!       Lahey (Proc. of the Sixth International Heat Transfer
!       Conference, 1, 1978)
!
      IMPLICIT NONE
!
! passed variables
      REAL(sdk) alp,hgamq,hl,stnulm,tl,tld,tssn,tw,xpe,xpelm,xnu
! internal variables
      REAL(sdk) alhgq,fracev,slphgq
!
      DATA alhgq/0.7d0/,slphgq/5.0d0/
!
      IF (xpe.LE.xpelm) THEN
       tld=tssn-xnu/(xpelm*stnulm)
      ELSE
       tld=tssn-xnu/(stnulm*xpe)
      ENDIF
      fracev=(tl-tld)/ max(1.0d0,tssn-tld)
      fracev= max(0.0d0, min(1.0d0,fracev))
      hgamq = max(0.0d0, min(1.0d0,(slphgq*(alhgq-alp))))*hl*fracev
      RETURN
      END SUBROUTINE SubBoil
!
      SUBROUTINE tmsfb(p,gmassi,tl,tmin,xe,ratio,rof,cf,hfg,tsat,rov,    &
```

```
     & rol,visf,sig,itmin)
      IMPLICIT NONE
!
!     Declaration Generated by genImpDecs.pl 5/98
      INTEGER(sik) itmin
!
!     Declaration Generated by genImpDecs.pl 5/98
      REAL(sdk) a1,a2,a3,a4,cf,cgmass,convp,dp,dtber,dtbera,dtmin,expnt, &
     & g,gc,gmass,gmassi,hfg,p,poly3,qanty,ratio,rof,rol,rov,sig,star,   &
     & tf,tf1,thn,thnf,tk,tk1,tl,tmin,tminb,tminbk,tmint,tsat,visf,x,xe, &
     & z1,z2,z3,z4,z5,z6
!
!
!     the purpose of this subroutine is to calculate the
!     minimum stable film boiling temperature (tmin).
!
!     ref.: o. c. iloeje et. al., an investigation of the collapse
!     and surface rewet in film boiling in forced vertical flow.,
!     j. heat transfer, may, 1975.
!
!     since the iloeje correlation gives too low a value of tmin at low
!     pressure, the homogeneous nucleation equation is used.
!
!     subroutine written by
!                   d. a. mandell
!                   safety code development, q-9
!                   lasl
!                   june, 1979
!
!-----------------------------------------------------------------------
!
!     note: properties are evaluated at tl or tv to avoid calling
!     thermo to get properties at tfilm=.5*(tw+tsat) etc.
!     except density evaluated from a taylor series.
!
!     if itmin <= 0, homogeneous nucleation tmin used
!     if tmin > 0, maximum of homogeneous nucleation and iloeje tmin
!     values used, with mass flux cutoff
!
      REAL(sdk) cthn(4)
!
      DATA cthn/705.44d0,-4.722d-2,2.3907d-5,-5.8193d-9/
!
!     1.45114e-4 = 14.7/1.013e5 (convert n/m**2 to psi)
!
      DATA convp/1.45114d-4/
!
!
!     gravitational constant in si units
!
      DATA gc,g/1.0d0,9.80665d0/
!
!     conversion factors for units
!
      DATA cgmass/737.35d0/
!
```

```
!!   STATEMENT FUNCTION
     star(qanty,expnt)=exp(expnt*log(qanty))
!
     poly3(a1,a2,a3,a4,x)=a1+x*(a2+x*(a3+x*a4))
     tf(tk)=1.8d0*(tk-273.d0)+32.d0
     tk1(tf1)=(tf1-32.d0)/1.8d0+273.d0
!
!    homogeneous nucleation tmin
!
!
!     calculate homogeneous nucleation temperature by using a curve fit
!     from cobra-tf.
!     dp, pressure difference, psi
!     thnf, homogeneous nucleation temperature, deg. f
!     thn, homogeneous temperature, deg. k
!
     dp=3203.6d0-p*convp
     thnf=poly3(cthn(1),cthn(2),cthn(3),cthn(4),dp)
     thn=tk1(thnf)
!
     tmint=thn+(thn-tl)*sqrt(ratio)
     IF (itmin.GT.0) THEN
!
!    iloeje tmin
!.
      z1=(rof/(cf))*hfg
      z2=star(g*(rol-rov)/(rol+rov),0.6666666667d0)
      z3=sqrt(gc*sig/((rol-rov)*g))
      z4=star(visf/(g*(rol-rov)),0.3333333333d0)
      dtber=.127d0*z1*z2*z3*z4
      dtbera=1.8d0*dtber
      gmass=gmassi*cgmass
!
!    cut-off iloeje tmin at g = 100,000 lbm/hr-ft**2
!
      IF (gmass.GT.1.d5) gmass=1.d5
!
      IF (xe.LT.0.d0) xe=.001d0
      z5=1.d0+star(gmass*1.d-4,.49d0)
      z6=1.d0-.295d0*star(xe,2.45d0)
      dtmin=.29d0*dtbera*z5*z6
      tminb=dtmin+tf(tsat)
      tminbk=tk1(tminb)
!
!    take maximum of trac value and iloeje value.
!
      tmin=  max(tmint,tminbk)
     ELSE
      tmin=tmint
     ENDIF
!
!
     tmin=  max(tmin,tsat+.0001d0)
     RETURN
     END SUBROUTINE tmsfb
```

```
END MODULE HeatCor
```

```
      MODULE Models

!     BEGIN MODULE USE
      USE IntrType

      CONTAINS

      SUBROUTINE fwall(cfz,wfv,fa,alp,rov,rol,vm,vl,vv,vol,dx,hd,rom    &
     &,visl,visv,nff,bd1,bd2,ncells,epsw,istrt,wfmfl,wfmfv)
!
!     BEGIN MODULE USE
      USE OneDDat
      USE CompTyp
      USE Bad
      USE Flt
!
      IMPLICIT NONE
!
!     Declaration Generated by genImpDecs.pl 5/98
      INTEGER(sik) istrt,j,jb,je,msctmp,ncells,ncp1,nfft
!
!     Declaration Generated by genImpDecs.pl 5/98
      REAL(sdk) aaa,alp15,alp5,alpjb,alpje,aream,areap,b,ccc,dxjb,dxje  &
     &,epsw,epsx,ff,fkfac,frp,gm,rdxm,rel,roljb,rolje,rolp5,romjb,romje &
     &,romp5,rovjb,rovje,rovp5,rvol,slip,tmp,va1,visbar,visljb,vislje   &
     &,vislp5,visvjb,visvje,visvp5,vj,vje,vl1,voljb,volje,vv1,x
!
!
!
!     this subroutine computes a two-phase friction factor
!
!*********************** nomenclature ****************************
!
! variable input/output         definition, units
!
! cfz(ncells+1)  o      wall friction coeff., dimensionless
! wfv(ncells+1)  o      wall friction for vapor, dimensionless
! fa(ncells+1)   i      flow area, m**2
! alp(ncells+1)  i      void fraction, dimensionless
! rov(ncells+1)  i      vapor density, kg/m**3
! rol(ncells+1)  i      liquid density, kg/m**3
! vm(ncells+1)   i      mixture velocity, m/s
! vl(ncells+1)   i      liquid velocity, m/s
! vv(ncells+1)   i      vapor velocity, m/s
! vol(ncells+1)  i      cell volume, m**3
! dx(ncells+1)   i      cell length, m
! hd(ncells+1)   i      cell hydraulic diameter, m
! rom(ncells+1)  i      mixture density, kg/m**3
! visl(ncells+1) i      liquid viscosity, nt*s/m**2
! visv(ncells+1) i      vapor viscosity, nt*s/m**2
! nff            i      wall friction correlation flag
! bd1(nbd)       i      left-side boundary array
! bd2(nbd)       i      right-side boundary array
! ncells         i      number of cells
! epsw           i      wall roughness, m
!
```

```
!-----------------------------------------------------------------
!
! note, numbers in brackets indicate dimension of variable
!
!*****************************************************************
!
!
!    nff=0 - constant friction factor, user input
!    nff=1 - homogeneous flow friction factor
!
     INCLUDE 'constant.h'
     DATA frp/0.5d0/
     TYPE(bdT) bd1,bd2
     REAL(sdk) nff(:)
     REAL(sdk) alp(:),rov(:),rol(:),vm(:),vol(:),dx(:),hd(:),rom(:)    &
    &,cfz(:),vl(:),vv(:),visl(:),visv(:),fa(:),wfv(:),wfmfl(:),wfmfv(:)
!
     alp5=0.d0
     ncp1=ncells+1
     msctmp=msc-istrt+1
     ccc=1.d0/12.0d0
!    Added as part of SJC task: #57
     IF(genTab(cco)%isSJC.NE.0) THEN
      j=0
     ENDIF
     DO j=1,ncp1
!     Added as part of SJC task: #57
      IF((genTab(cco)%isSJC.NE.0).AND.j.GT.1) EXIT
      nfft=iabs(int(nff(j)))
      cfz(j)=0.d0
      wfv(j)=0.d0
      epsx=epsw/hd(j)
      IF (nfft.NE.0) THEN
       ff=0.032d0
       je=j
       jb=j-1
       vj=vm(j)
       vje=vm(je)
!      Added as part of SJC task: #57
       IF(genTab(cco)%isSJC.NE.0) THEN
        dxje=bd2%dx
        dxjb=bd1%dx
        alpje=bd2%alpn
        alpjb=bd1%alpn
        rovje=bd2%rovn
        rovjb=bd1%rovn
        volje=bd2%vol
        voljb=bd1%vol
        IF (((bd1%type.EQ.plenh).OR.(bd1%type.EQ.vsslh)).          &
   &      AND.(bd1%faFrac.GT.0.0d0)) voljb=voljb/bd1%faFrac
        IF (((bd2%type.EQ.plenh).OR.(bd2%type.EQ.vsslh)).          &
   &      AND.(bd2%faFrac.GT.0.0d0)) volje=volje/bd2%faFrac
        rolje=bd2%roln
        roljb=bd1%roln
        romje=bd2%rom
        romjb=bd1%rom
```

```
            vislje=visl(je)
            visvje=visv(je)
            rdxm=1.d0/(.5d0*dxje+.5d0*dxjb)
            IF (bd1%visl.EQ.0.d0) THEN
              visljb=vislje
              visvjb=visvje
            ELSE
              visljb=bd1%visl
              visvjb=bd1%visv
            ENDIF
            IF (bd2%visl.EQ.0.d0) THEN
              vislje=visljb
              visvje=visvjb
            ELSE
              vislje=bd2%visl
              visvje=bd2%visv
            ENDIF
            GOTO 61
          ELSEIF (j.EQ.1) THEN
            dxje=dx(je)
            dxjb=bd1%dx
            alpje=alp(je)
            alpjb=bd1%alpn
            rovje=rov(je)
            rovjb=bd1%rovn
            volje=vol(je)
            voljb=bd1%vol
            IF (((bd1%type.EQ.plenh).OR.(bd1%type.EQ.vsslh)).AND      &
     &       .(bd1%faFrac.GT.0.0d0)) voljb=voljb/bd1%faFrac
            rolje=rol(je)
            roljb=bd1%roln
            romje=rom(je)
            romjb=bd1%rom
            vislje=visl(je)
            visvje=visv(je)
            rdxm=1.d0/(.5d0*dx(j)+.5d0*dxjb)
            IF (bd1%visl.EQ.0.d0) THEN
              visljb=vislje
              visvjb=visvje
              IF (ncp1.NE.1) GOTO 61
            ELSE
              visljb=bd1%visl
              visvjb=bd1%visv
              IF (ncp1.NE.1) GOTO 61
            ENDIF
          ELSEIF (j.NE.ncp1) THEN
            dxje=dx(je)
            dxjb=dx(jb)
            alpje=alp(je)
            alpjb=alp(j-1)
            rovje=rov(j)
            rovjb=rov(j-1)
            volje=vol(je)
            voljb=vol(jb)
            rolje=rol(je)
            roljb=rol(jb)
```

```
        romje=rom(je)
        romjb=rom(jb)
        vislje=visl(je)
        visljb=visl(jb)
        visvje=visv(je)
        visvjb=visv(jb)
        GOTO 61
       ENDIF
       dxje=bd2%dx
       dxjb=dx(jb)
       alpje=bd2%alpn
       alpjb=alp(j-1)
       rovje=bd2%rovn
       rovjb=rov(j-1)
       volje=bd2%vol
       IF (((bd2%type.EQ.plenh).OR.(bd2%type.EQ.vsslh)).AND          &
     &  .(bd2%faFrac.GT.0.0d0)) volje=volje/bd2%faFrac
       voljb=vol(jb)
       rolje=bd2%roln
       roljb=rol(jb)
       romje=bd2%rom
       romjb=rom(jb)
       visljb=visl(jb)
       visvjb=visv(jb)
       IF (bd2%visl.EQ.0.d0) THEN
        vislje=visljb
        visvje=visvjb
       ELSE
        vislje=bd2%visl
        visvje=bd2%visv
       ENDIF
 61    IF (nfft.LT.100) THEN
       va1=abs(vm(j))
       vl1=abs(vl(j))
       vv1=abs(vv(j))
       IF (va1.LT.1.0d-10) va1=zero
       IF (vl1.LT.1.0d-10) vl1=zero
       IF (vv1.LT.1.0d-10) vv1=zero
       rvol=one/(volje+voljb)
       alp5=(alpje*volje+voljb*alpjb)*rvol
       alp15=one-alp5
       rovp5=(rovje*volje+voljb*rovjb)*rvol
       romp5=(romje*volje+voljb*romjb)*rvol
       rolp5=(rolje*volje+roljb*voljb)*rvol
       vislp5=(vislje*volje+visljb*voljb)*rvol
       visvp5=(visvje*volje+visvjb*voljb)*rvol
!
!   homogeneous two phase friction factor
!
       x=zero
       slip=one
       IF (alp5.GE.0.001d0) x=one/(one+(one-alp5)/alp5*rolp5/(rovp5 &
     &   *slip))
       tmp=x/visvp5+(1.0d0-x)/vislp5
       visbar=1.0d0/tmp
       gm=romp5*va1
```

```
          rel=  max(100.d0,gm*hd(j)/visbar)
cjoe            aaa=(2.457d0*log(1.d0/((7.d0/rel)**0.9d0+.27*epsx)))**16.0d0
cjoe            b=(37530.d0/rel)**16.0d0
cjoe            ff=((8.0d0/rel)**12.0d0+1.d0/(aaa+b)**1.5d0)**ccc
cjoe            cfz(j)=4.d0*ff
        if(rel.gt.3000.d0)then
          cfz(j) = 2.*(0.0014+0.125/rel**0.32)
        else
          cfz(j) = 2.* 25./rel
        end if
!-----multiply the vapor and liquid friction factors by their input
!-----design factors (cfz's actual argument in preper is the liquid
!-----array).  Note that pre-design factor fwall had "wfv(j)=cfz(j)"
!-----at end of loop over faces, between statements 50 and 60.
          wfv(j)=cfz(j)*wfmfv(j)
          cfz(j)=cfz(j)*wfmfl(j)
        ENDIF
       ENDIF
      ENDDO
      RETURN
      END SUBROUTINE fwall

      SUBROUTINE mprop(t,row,cpw,cw,emis,matid,nodes,ndm1,ncells,ncm1,   &
     & idim)
!
      USE Materials
!
      IMPLICIT NONE
!
!     Declaration Generated by genImpDecs.pl 5/98
      INTEGER(sik) i,idim,j,k,ncells,ncm1,ndm1,nodes
!
!     Declaration Generated by genImpDecs.pl 5/98
      REAL(sdk) tav,tmss
!
!
!     orders structure property selection and evaluates
!     an average temperature for property evaluation.
!
      REAL(sdk) matid(ndm1)
      REAL(sdk) t(:,:),row(:,:),cpw(:,:),cw(:,:),emis(ncells)
!
!
      DO j=1,ncells
       DO i=1,ndm1
        tav=t(i,j)
        IF (nodes.NE.1) tav=0.5d0*(t(i,j)+t(i+1,j))
        k=int(matid(i))
!
!
        CALL mstrct(cw(i,j),cpw(i,j),emis(j),k,row(i,j),tav,tmss)
!
       ENDDO
      ENDDO
      RETURN
      END SUBROUTINE mprop
```

```
      SUBROUTINE FricIf(alp,rov,rol,visv,visl,vl,vv,vln,vvn,p,dx,hd,fa   &
     &,vol,wfv,wfl,cif,grav,bd1,bd2,ncells,sigm,nff,wfmfl,wfmfv,fric     &
     &,fricr,fricd,h1,h2,h3,regnm,bit,alpEdge,ialpEdge)
!
!     BEGIN MODULE USE
      USE CFaces
      USE Io
      USE OneDDat
      USE GlobalDat
      USE CompTyp
      USE Bits
      USE Flt
      USE Bad
      USE Gen1DArray
      USE InterfacePar

!
      IMPLICIT NONE
!
      INTEGER(sik) ihutze,ilvp,inhor,j,jm,jp,msc0,msc1,naltt,ncells,ncp
!
      REAL(sdk) agrav,agravm,agravp,aj,ajc,ajcm,ajcp,ajg,ajgdm,ajgmn     &
     &,ajgmx,ajle,alpa,alpam,alpb,alpc,alpd,alpf,alpfr,alpg,alpl,alpm    &
     &,alpmm,alpmm1,alpmsv,alppp,alppp1,alppsv,alpup,alpdn,alpv,altm     &
     &,altmf,area,areal,areag,aw,avr,avlj,avrd,avrf,avrm,avvj,bmax,bw,c0 &
     &,c1,cd,cifn,cimdm,citmax,citmin,cdmx,cfti,cfti1,cfti2,cfti3,cfhf   &
     &,cftmn,cftmx,cim,ciminc,cmul,cmulb,cmum,cmuv,cmuvb,cond,dcua,dftvr &
     &,dg,dl,dhldz,dhlt,dhltt,dia,diam,diamin,diap,dp,dro,droa,dxm,dxp    &
     &,eeq,fa1,fa2,fdf1,fdf2,ff,fi,fifd,fiff

      REAL(sdk) fricl,fricv,fricla,fricva,fwg3,fx,gav,gravm,gravp,gravr   &
     &,hdb,hdbp,hl,hlm,hlme,hlp,hlpe,rata,rdx,red,redmn,reg,regdm,rel     &
     &,rele,reyvb,rd,rddm,rl,rolm,rolp,rovm,rovp,rv,sig,sigma,sigmab,sg   &
     &,si,sip,sl,temp,ug,ul,ur,vcrit,vdis,vrfac,vscl,vscv,vlj,vvj,vlnj    &
     &,vvnj,wee,wfa,wfalp1,wfalp2,wfm,wfp,wfhf,wflj,wfvj,wl,wl1,wv,wv1    &
     &,wx,wxg,wxl,xjtot,xslug
!
!
!
!     evaluation of interfacial drag, separated from StbVel1D
!
!          Jennifer Uhle 11/98
!
!
!
!  interfacial shear modified by m. cappiello, spring 1988
!
!  References used in the determination of interfacial drag and
!  interfacial area:
!  1)Mishima and Ishii, "Theoretical Prediction of onset of
!    Horizontal slug flow",asme journal of fluids engineering,
!    vol.102,p.441,dec.1980
!  2)Ishii and Mishima, "study of two-fluid model and interfacial
!    area", nureg/cr-1873, anl-80-111.
!  3)Kataoka and ishii,"mechanism and correlation of droplet entrainmet
```

```
!   and deposition in annular two-phase flow", Nureg/cr-2885,anl-82-44.
!   july,1982.
!  4)ishii and mishima,"two-fluid model and hydrodynamic constitutive
!    relations", Nuclear Engineering and Design, 82 (1984) p.107.
!  5)Ishii and chawla,"local drag laws in dispersed two-phase flow", nu
!    nureg/cr-1230, anl-79-105, dec. 1979.
!  6)Bird,Stewart, and Lightfoot."Transport Phenomena".
!  7)Clift,Grace, and Weber, "Bubbles drops and particles"
!  8)Wallis, "One dimensional two-phase flow", 1969.
!  9)Ohnuki, adachi and murao, "scale-effects on counter-current
!   gas-liquid flow in horizontal tube connected to inclined riser ",
!   Ans 1987 national heat tx conf., pittsburgh Penn, aug. 1987,p.40.
!  10)Kataoka,Ishii,Mishima,"Generation and size distribution of drople
!    in annular two-phase flow", Trans. asme, vol105, june 1983, p.230
!  11)Ishii,"One-dimensional drift flux model and constitutive equation
!    for relative motion between phase in various two-phase flow regim
!    Anl-77-47, 1977.
!  12)Letter, Ishii to Nelson,July 28, 1987.
!  13)Chexal,Lellouche "A full range drift-flux correlation for vertica
!    flows (revision 1), epri np-3989-sr, rev 1, sept. 1986.
!  14)taitel and dukler, "a model for predicting flow regime transition
!    in horizontal and near horizontal gas-liquid flow", aiche Journal
!    vol.22,no.1,january,1976.
!
!  some variable definitions, in general l is liquid, v is vapor
!  sig: donor celled surface tension
!  vscl: donor celled liquid viscosity
!  vscv: donor celled vapor viscosity
!  altm: dx weighted void fraction
!  altmf: dx weighted void fraction for use in if shear calculations
!  rv:   dx weighted vapor density
!  rl:   dx weighted liquid density
!  dro:  rl-rv
!  vvnj: vapor velocity
!  vlnj: liquid velocity
!  cift1: bubbly-slug interfacial drag coeff.
!  cift2: annular-mist interfacial drag coeff.
!  cift3: transition regime if. drag coeff.
!  cfhf:  horizontal flow if. drag coeff.
!
!
!  constant definitions brought in through com decks:
!  com deck webnum:
!  bmin=0.0001
!  dmin=0.00001
!  com deck diddle:
!  alw1=0.5
!  alw2=0.75
!  vrtcut=0.01
!  alpbct=0.05
!  vecvct=0.01
!  veclct=0.001
!  vrbcut=1.
!  vlvcmx=20.
!  ihor=0, dhldz=0.,wfhf=0.
!  ihor=1, test dhldz,test wfhf (default)
```

```
!  ihor=2, test dhldz,wfhf=1.
!  ihor=3, dhldz=0.,test wfhf
!  inhor=0, normal comps
!  inhor=1, accum,prizer,dhldz=0.,wfhf=0.
!  com deck constant:
!  gc=9.80665
!  com deck ifcrs:
!  alpgs=0.15, flmin=0.0001, almin=0.00001, almax=0.9999
!  alpbcd=0.005,
!  hdmax=2.0,  rdmax=0.002, rdmin=0.000042,
!  fui1=2.0,  fui2=0.5,  fri1=20.0  fri2=20.0
!  stfrl=1.0,  stfru=2.0,  alptp=0.9,  alptm=0.1
!  d1x=0.09144, d2x=0.0381,  pc25=1.95e15,  pcrit=220.8e5
!  fifi=0.4, fifr=2.0, fdis1=10.0, fdis2=12.0
!
      INCLUDE 'strtnt.h'
      INCLUDE 'cflow.h'
      INCLUDE 'ciflim.h'
      INCLUDE 'vellim.h'
      INCLUDE 'diddle.h'
      INCLUDE 'tst3d.h'
      INCLUDE 'webnum.h'
      INCLUDE 'constant.h'
!
      REAL(sdk) nff(:)
      TYPE(bdT) bd1,bd2
      REAL(sdk) alp(:),rov(:),rol(:),visv(:),visl(:),vl(:),vv(:),vln(:)  &
     &,vvn(:),p(:),dx(:),hd(:),fa(:),vol(:),wfv(:),wfl(:),cif(:),grav(:) &
     &,sigm(:),wfmfl(:),wfmfv(:),fric(:),fricr(:),h1(:)                  &
     &,h2(:),h3(:),regnm(:),bit(:),fricd(:),alpEdge(:),ialpEdge(:)
      REAL(sdk) pl,pr,vmjm,dx1,dx2,vvap,vliq,vvt1,vlt1,vvold
!
      REAL(sdk), SAVE :: hdmax=2.0_sdk
      REAL(sdk), SAVE :: stfrl=1.0_sdk
      REAL(sdk), SAVE :: stfru=10.0_sdk
!
!
!  initialize various variables to save on special if tests.
!
      aw=1.d0/(alw2-alw1)
      bw=-aw*alw1
      ihutze=0
      ciminc=almin*gc/(vrcmin*vrcmin)
      ncp=ncells+1
!
      msc0=0
      msc1=0
      IF (genTab(cco)%type.NE.pumph) msc0=msc
      IF (msc0.NE.0) THEN
       msc1=msc+1
      ENDIF
!
      cmul=bd1%visl
      cmuv=bd1%visv
      sigma=bd1%sig
      inhor=0
```

```
!    IF (genTab(cco)%type.EQ.prizrh.OR.genTab(cco)%type.EQ.turbh.OR    &
!    &.(genTab(cco)%type.EQ.pipeh.AND.iacc2.NE.0)) inhor=1
     IF (genTab(cco)%type.EQ.prizrh.OR.(genTab(cco)%type.EQ.pipeh.AND.  &
    & iacc2.NE.0)) inhor=1
!
!
     DO j=jstart,ncp
!      Added as part of SJC task: #60
       IF((genTab(cco)%isSJC.NE.0).AND.j.GT.jstart) EXIT
       naltt=nalt
       jp=j+1
       jm=j-1
       vvj=vv(j)
       vlj=vl(j)
       vvnj=vvn(j)
       vlnj=vln(j)
!
!  donor cell weighing factors.
!
       wl=1.d0
       wv=1.d0
       IF (vvj.LT.0.d0) wv=0.d0
       IF (vlj.LT.0.d0) wl=0.d0
!
!  fluid properties
!
       cmulb=cmul
       cmuvb=cmuv
       sigmab=sigma
!
!      Added as part of SJC task: #60
!      BUG fix: found by DVF 6.0 out-of-bounds checking
       IF (j.NE.ncp) THEN
        cmul=visl(j)
        cmuv=visv(j)
        sigma=sigm(j)
       ENDIF
!
       IF (fa(j).LT.1.0d-10) GOTO 25
!      Added as part of SJC task: #60
       IF(genTab(cco)%isSJC.NE.0) THEN
        IF (nstep.EQ.1) THEN
         IF ((bd2%type.EQ.breakh.OR.bd2%type.EQ.vsslh).OR.          &
    &        (bd1%type.EQ.breakh.OR.bd1%type.EQ.vsslh).AND.         &
    &         fa(j).NE.0.d0)THEN
          fa2=bd2%vol/bd2%dx
          fa1=bd1%vol/bd1%dx
          rata=  max(1.0d-06,fa1)/  max(1.0d-06,fa2)
          IF (rata.GT.rvmax.OR.1.0d0/rata.GT.rvmax) THEN
           IF (nfrc1.EQ.1) THEN
            IF (fric(j).EQ.0.0d0.AND.int(nff(j)).GE.0) THEN
             jfat=2
             WRITE (iout,11) genTab(cco)%num,j,fa1,fa2
             WRITE (imout,11) genTab(cco)%num,j,fa1,fa2
             WRITE (itty,11) genTab(cco)%num,j,fa1,fa2
            ENDIF
```

```
          ELSEIF (fric(j).EQ.0.0d0.AND.int(nff(j)).GE.0.AND      &
     &     .fricr(j).EQ.0.0d0) THEN
           jfat=2
           WRITE (iout,11) genTab(cco)%num,j,fa1,fa2
           WRITE (imout,11) genTab(cco)%num,j,fa1,fa2
           WRITE (itty,11) genTab(cco)%num,j,fa1,fa2
          ENDIF
         ENDIF
        ENDIF
       ENDIF
       IF (isrb.EQ.2.OR.islb.EQ.2) naltt=0
       dp=bd2%pn-bd1%pn
       gravp=bd2%grav
       gravm=bd1%grav
       dxp=bd2%dx
       dxm=bd1%dx
       diap=sqrt(1.27323d0*bd2%vol/bd2%dx)
       diam=sqrt(1.27323d0*bd1%vol/bd1%dx)
       alppp=bd2%alpn
       alpmm=bd1%alpn
       alppsv=alppp
       alpmsv=alpmm
       IF (abs(vvnj).LT.1.1d-10) wv=.5d0-sign(.5d0,dp)
       IF (abs(vlnj).LT.1.1d-10) wl=.5d0-sign(.5d0,dp)
       wl1=1.d0-wl
       wv1=1.d0-wv
       rdx=2.d0/(bd1%dx+bd2%dx)
       wfm=.5d0*rdx*bd1%dx
       wfp=1.d0-wfm
       alpm=wfm*alpmm+wfp*alppp
       IF (isrb.EQ.2.AND.vlj.GT.0.0d0) alpm=bd1%alp
       IF (islb.EQ.2.AND.vlj.LT.0.0d0) alpm=bd2%alp
       alpg=alpm
       rovm=bd1%rovn
       rovp=bd2%rovn
       rv=wfm*rovm+wfp*rovp
       rolm=bd1%roln
       rolp=bd2%roln
       rl=wfm*rolm+wfp*rolp
       alpl=wl*alpmm+wl1*alppp
       alpv=wv*alpmm+wv1*alppp
       sigma=bd2%sig
       cmuv=bd2%visv
       cmul=bd2%visl

      ELSE IF (j.EQ.ncp) THEN
!
! special considerations for right boundary.
!
       IF (isrb.EQ.0) GOTO 25
       IF (nstep.EQ.1) THEN
        IF ((bd2%type.EQ.breakh.OR.bd2%type.EQ.vsslh).AND.fa(j).NE.0 &
     &    .d0)THEN
         fa2=bd2%vol/bd2%dx
         fa1=vol(j-1)/dx(j-1)
         rata=  max(1.0d-06,fa1)/  max(1.0d-06,fa2)
```

```
            IF (rata.GT.rvmax.OR.1.0d0/rata.GT.rvmax) THEN
              IF (nfrc1.EQ.1) THEN
                IF (fric(j).EQ.0.0d0.AND.int(nff(j)).GE.0) THEN
                  IF (.NOT.nofat) jfat=2
                  WRITE (iout,11) genTab(cco)%num,j,fa1,fa2
                  WRITE (imout,11) genTab(cco)%num,j,fa1,fa2
                  WRITE (itty,11) genTab(cco)%num,j,fa1,fa2
                ENDIF
              ELSEIF (fric(j).EQ.0.0d0.AND.int(nff(j)).GE.0.AND      &
     &          .fricr(j).EQ.0.0d0) THEN
                IF (.NOT.nofat) jfat=2
                WRITE (iout,11) genTab(cco)%num,j,fa1,fa2
                WRITE (imout,11) genTab(cco)%num,j,fa1,fa2
                WRITE (itty,11) genTab(cco)%num,j,fa1,fa2
              ENDIF
            ENDIF
          ENDIF
        ENDIF
        IF (isrb.EQ.2) naltt=0
        dp=bd2%pn-p(jm)
        gravp=bd2%grav
        gravm=grav(jm)
        dxp=bd2%dx
        dxm=dx(jm)
        diap=sqrt(1.27323d0*bd2%vol/bd2%dx)
        diam=sqrt(1.27323d0*vol(jm)/dx(jm))
        IF (bd2%type.EQ.vsslh) ilvp=1
        alppp=bd2%alpn
        alpmm=alp(jm)
        alppsv=alppp
        alpmsv=alpmm
        IF (abs(vvnj).LT.1.1d-10) wv=.5d0-sign(.5d0,dp)
        IF (abs(vlnj).LT.1.1d-10) wl=.5d0-sign(.5d0,dp)
        wl1=1.d0-wl
        wv1=1.d0-wv
        rdx=2.d0/(dx(jm)+bd2%dx)
        wfm=.5d0*rdx*dx(jm)
        wfp=1.d0-wfm
        alpm=wfm*alpmm+wfp*alppp
        IF (isrb.EQ.2.AND.vlj.GT.0.0d0) alpm=alp(jm)
        alpg=alpm
        rovm=rov(jm)
        rovp=bd2%rovn
        rv=wfm*rovm+wfp*rovp
        rolm=rol(jm)
        rolp=bd2%roln
        rl=wfm*rolm+wfp*rolp
        alpl=wl*alpmm+wl1*alppp
        alpv=wv*alpmm+wv1*alppp
        sigma=bd2%sig
        cmuv=bd2%visv
        cmul=bd2%visl

      ELSEIF (j.EQ.jstart) THEN
!
!  special considerations for left boundry.
```

```
!
      IF (nstep.EQ.1) THEN
       IF ((bd1%type.EQ.breakh.OR.bd1%type.EQ.vsslh).AND.fa(j).NE.0 &
     &    .d0)THEN
        fa1=bd1%vol/bd1%dx
        fa2=vol(j)/dx(j)
        rata= max(1.0d-06,fa1)/ max(1.0d-06,fa2)
        IF (rata.GT.rvmax.OR.1.0d0/rata.GT.rvmax) THEN
         IF (nfrc1.EQ.1) THEN
          IF (fric(j).EQ.0.0d0.AND.int(nff(j)).GE.0) THEN
           IF (.NOT.nofat) jfat=2
           WRITE (iout,11) genTab(cco)%num,j,fa1,fa2
           WRITE (imout,11) genTab(cco)%num,j,fa1,fa2
           WRITE (itty,11) genTab(cco)%num,j,fa1,fa2
          ENDIF
         ELSEIF (fric(j).EQ.0.0d0.AND.int(nff(j)).GE.0.AND       &
     &       .fricr(j).EQ.0.0d0) THEN
          IF (.NOT.nofat) jfat=2
          WRITE (iout,11) genTab(cco)%num,j,fa1,fa2
          WRITE (imout,11) genTab(cco)%num,j,fa1,fa2
          WRITE (itty,11) genTab(cco)%num,j,fa1,fa2
         ENDIF
        ENDIF
       ENDIF
      ENDIF
   11  FORMAT (' For component # ',i5,' the volume average flow ',    &
     &  'area change is too large ',/,' at cell edge ',i3,/,          &
     &  ' fa1 = ',1p,e14.4,' fa2 = ',1p,e14.4)
      IF (islb.EQ.0) GOTO 25
      IF (islb.EQ.2) naltt=0
      rdx=2.d0/(dx(j)+bd1%dx)
      wfm=.5d0*rdx*bd1%dx
      wfp=1.d0-wfm
      dp=p(j)-bd1%pn
      gravp=grav(j+1)
      gravm=bd1%grav
      dxp=dx(j)
      dxm=bd1%dx
      diam=sqrt(1.27323d0*bd1%vol/dxm)
      diap=sqrt(1.27323d0*vol(j)/dx(j))
      alppp=alp(j)
      alpmm=bd1%alpn
      alppsv=alppp
      alpmsv=alpmm
      IF (abs(vvnj).LT.1.1d-10) wv=.5d0-sign(.5d0,dp)
      IF (abs(vlnj).LT.1.1d-10) wl=.5d0-sign(.5d0,dp)
      wv1=1.d0-wv
      wl1=1.d0-wl
      alpm=wfm*alpmm+wfp*alppp
      IF (islb.EQ.2.AND.vlj.LT.0.0d0) alpm=alp(j)
      alpg=alpm
      rovm=bd1%rovn
      rovp=rov(j)
      rv=wfm*rovm+wfp*rovp
      rolm=bd1%roln
      rolp=rol(j)
```

```
        rl=wfm*rolm+wfp*rolp
        alpl=wl*alpmm+wl1*alppp
        alpv=wv*alpmm+wv1*alppp

      ELSE

        dp=p(j)-p(jm)
        gravp=grav(j+1)
        gravm=grav(jm)
        dxp=dx(j)
        dxm=dx(jm)
!
!  pipe diameter estimate(for stratified flow calculations)
!
        diap=sqrt(1.27323d0*vol(j)/dx(j))
        diam=sqrt(1.27323d0*vol(jm)/dx(jm))
        alppp=alp(j)
        IF (ialpEdge(j).GT.0.0_sdk) alppp=alpEdge(j)
        alpmm=alp(jm)
        IF (ialpEdge(j).LT.0.0_sdk) alpmm=alpEdge(j)
        alppsv=alppp
        alpmsv=alpmm
!
!  complete donor cell weighting calculations.
!
        IF (abs(vvnj).LT.1.1d-10) wv=.5d0-sign(.5d0,dp)
        IF (abs(vlnj).LT.1.1d-10) wl=.5d0-sign(.5d0,dp)
        wl1=1.d0-wl
        wv1=1.d0-wv
!
!  compute various averages needed at cell edges.
!
        rdx=2.d0/(dx(j)+dx(jm))
        wfm=.5d0*rdx*dx(jm)
        wfp=1.d0-wfm
        alpm=(wfm*alpmm+wfp*alppp)
        alpg=alpm
        rovm=rov(jm)
        rovp=rov(j)
        rv=wfm*rovm+wfp*rovp
        rolm=rol(jm)
        rolp=rol(j)
        rl=wfm*rolm+wfp*rolp
        alpl=wl*alpmm+wl1*alppp
        alpv=wv*alpmm+wv1*alppp
!
      ENDIF
!
!   evaluate the other momentum equation terms
!
      cond=0.d0
!
      IF ((genTab(cco)%type.EQ.pumph).AND.(j.EQ.msc)) THEN
!       Modified as part of SJC task: #3-1
        IF (genTab(cco)%isSJC.EQ.1) THEN
          diap=sqrt(1.27323d0*bd2%vol/bd2%dx)
```

```
          CALL level(hlp,bd2%alp,diap)
          temp= max(1.0d-06,1.0d0-(2.0d0*hlp/diap-1.0d0)**2)
          h2(j)=bd2%dx*diap*sqrt(temp)
         ELSE
          diap=sqrt(1.27323d0*vol(j)/dx(j))
          CALL level(hlp,alp(j),diap)
          temp= max(1.0d-06,1.0d0-(2.0d0*hlp/diap-1.0d0)**2)
          h2(j)=dx(j)*diap*sqrt(temp)
         ENDIF
         GOTO 781
        ELSE
!
!   do an input-data check for no flow-area change between jcell-1
!   and jcell and between jcell and jcell+1 of a tee component
!
         IF (nstep.EQ.0) THEN
          IF (j.EQ.msc0) THEN
           ajc=diap*diap*0.78540d0
           ajcm=diam*diam*0.78540d0
           IF ((j.EQ.jstart).AND.(bd1%faFrac.GT.0.0d0)) ajcm=ajcm    &
     &      /bd1%faFrac
           IF ((abs(ajcm-ajc).GT.1.0d-04*ajc).OR.(abs(fa(j)-ajc).GT.1 &
     &      .0d-04*ajc)) THEN
            WRITE (imout,301) genTab(cco)%num,ajcm,fa(j),ajc
            WRITE (iout,301) genTab(cco)%num,ajcm,fa(j),ajc
            WRITE (itty,301) genTab(cco)%num,ajcm,fa(j),ajc
  301        FORMAT(/' *StbVel1D* tee component',i4,' has a jcell-1', &
     &        ' flow area of',1p,e11.4/9x,'a jcell-1/2 flow',       &
     &        ' area of',e11.4,', and a jcell flow'/9x,'area',' of', &
     &        e11.4,' which involve a flow-area change'/9x,         &
     &        'not modeled by the jcell-1/2 motion equation')
            IF (.NOT.nofat) THEN
             CALL error(2,'*StbVel1D* jcell flow-area change')
            ENDIF
           ENDIF
          ENDIF
          IF (j.EQ.msc1) THEN
           ajc=diam*diam*0.78540d0
           ajcp=diap*diap*0.78540d0
           IF ((j.EQ.ncp).AND.(bd2%faFrac.GT.0.0d0)) ajcp=ajcp       &
     &      /bd2%faFrac
           IF ((abs(ajcp-ajc).GT.1.0d-04*ajc).OR.(abs(fa(j)-ajc).GT.1 &
     &      .0d-04*ajc)) THEN
            WRITE (imout,302) genTab(cco)%num,ajc,fa(j),ajcp
            WRITE (iout,302) genTab(cco)%num,ajc,fa(j),ajcp
            WRITE (itty,302) genTab(cco)%num,ajc,fa(j),ajcp
  302        FORMAT (/' *StbVel1D* tee component',i4,' has a jcell',  &
     &        ' flow area of',1p,e11.4/9x,'a jcell+1/2 flow',       &
     &        ' area of',e11.4,', and a jcell+1 flow'/9x,'area',    &
     &        ' of',e11.4,' which involve a flow-area change'/9x,   &
     &        'not modeled by the jcell+1/2 motion equation')
            IF (.NOT.nofat) THEN
             CALL error(2,'*StbVel1D* jcell flow-area change')
            ENDIF
           ENDIF
          ENDIF
```

```
      ENDIF
!
! interfacial drag coefficient
! initialize properties and alpha
!
      sig=wl*sigmab+wl1*sigma
      vscl=wl*cmulb+wl1*cmul
      vscv=wv*cmuvb+wv1*cmuv
      IF (alpm.LT.alpg) alpg=alpm
      alpg= max(alpg,1.d-6)
      alpg= min(alpg,0.999999d0)
!
! define alpfr (constrained) for interfacial drag calculations
! using the donor celled alpha (on vv)
!
      alpfr= max( min(alpv,almax),almin)
      altm= max(alpm,1.d-6)
      altmf=altm
      IF (naltt.NE.0) THEN
       altmf= min(alppp,alpmm)
       IF (grav(j).EQ.0.d0) altmf= max(alppp,alpmm)
       altmf= max(altmf,1.d-6)
       IF (grav(j).GE..3d0) THEN
        IF (alppp.LT.alpmm) altmf=alpmm
       ELSEIF (grav(j).LE.-.3d0) THEN
        IF (alpmm.LT.alppp) altmf=alppp
       ENDIF
      ENDIF
      altm= min(altm,.999999d0)
!
! avoid numerical diffusion in accum
! zero interfacial drag, and allow phase separation
!
      dhldz=0.0d0
      fricla=fric(j)
      fricva=fric(j)
      IF (nfrc1.NE.1) THEN
       IF (vlj.LT.0.0d0) fricla=fricr(j)
       IF (vvj.LT.0.0d0) fricva=fricr(j)
      ENDIF
      IF (abs(fricla).LE.1.0d20.AND.abs(fricva).LE.1.0d20) THEN
!
! more initialization of interfacial drag variables
!
       cfti1=0.d0
       cfti2=0.d0
       cfti3=0.d0
       cfhf=0.d0
       ur=0.d0
       ul=0.d0
       si=0.0d0
       sip=.001d0*diap
       hl=0.0d0
       wfhf=0.0d0
       reg=0.0d0
       dro= max(rl-rv,0.01d0)
```

```
          avr=abs(vvn(j)-vln(j))
          avr= max(avr,vrtcut)
          avvj= max(abs(vvnj),vecvct)
          avlj= max(abs(vlnj),veclct)
          diap= min(diap,hdmax)
          diam= min(diam,hdmax)
          hlpe=0.0d0
          hlme=0.0d0
          hlp=0.0d0
          hlm=0.0d0
          wfvj=wfv(j)
          wflj=wfl(j)
          agrav=abs(grav(j))
!
!  vertical stratified flow model for accumulators
!  droplet drag calculated above the interface
!  bubble drag calculated below the interface
!  transition as the cell empties
!
          xjtot=(1.d0-alpl)*avlj+alpv*avvj
          IF ((genTab(cco)%type.EQ.pipeh.AND.iacc2.NE.0).OR.(xjtot.LT  &
     &     .0.1d0.AND.agrav.GT.0.75d0)) THEN
           wfhf=0.0d0
           dhldz=0.0d0
           sip=diap
           cfhf=0.165d0*rv*(1.0d0-aupdrg)/rdmin
           cfti1=0.33d0*fifbs*alpbcd*rl/bmin
           alpup=alppp
           alpdn=alpmm
           IF (grav(j).LT.0.0d0) THEN
            alpup=alpmm
            alpdn=alppp
           ENDIF
           IF (alpup.GT.aupct) wfhf=1.0d0-  max(  min((1.0d0-alpup)   &
     &      /(1.0d0-aupct),1.0d0),0.0d0)
          ELSE
!
!  ************************************************
!  *****  horizontal-stratified flow regime    ****
!  ************************************************
!
          IF (.NOT.(ihor.EQ.0.OR.inhor.EQ.1)) THEN
           IF (.NOT.(vlj.GT.0.0d0.AND.grav(j).GT.0.174d0)) THEN
            IF (.NOT.(vlj.LT.0.0d0.AND.grav(j).LT.-0.174d0)) THEN
             dia=hd(j)
             diamin=  min(diam,diap)
             agravp=abs(gravp)
             agravm=abs(gravm)
             IF (agrav.LE.tgrav) THEN
              IF (.NOT.(agrav.GE.0.174d0.AND.altmf.LT.0        &
     &         .9d0)) THEN
              IF (.NOT.(gravm.LT.-0.174d0.AND.gravp.GT.0      &
     &         .174d0))THEN
               CALL level(hl,alpm,dia)
               IF (hl.GT.(.001d0*dia)) THEN
                gravr=sqrt(1.d0-grav(j)*grav(j))
```

```
                alpmm1=alpmm
                alppp1=alppp
                alpmm= min(alpmm,1.d0)
                alppp= min(alppp,1.d0)
                alpmm= max(alpmm,0.0d0)
                alppp= max(alppp,0.0d0)
                CALL level(hlp,alppp,diap)
                CALL level(hlm,alpmm,diam)
                hlpe=hlp-diap*.5d0
                alppp=alppp1
                alpmm=alpmm1
                hlme=hlm-diam*.5d0
                hlpe= min(diamin*.5d0,hlpe)
                hlme= min(diamin*.5d0,hlme)
                hlpe= max(-diamin*.5d0,hlpe)
                hlme= max(-diamin*.5d0,hlme)
                dhltt=hlpe-hlme
                dhldz=dhltt*rdx*delt*gc*gravr
!
! dhldz is a term which approximates the hydraulic head in 1-d
! stratified flow - the following test prevents the term from
! dominating at large time steps (it is added in explicitly)
!
                dhlt=.5d0/(rdx*delt)
                dhldz= min(abs(dhldz),dhlt)*sign(1.d0      &
   &             ,dhldz)
                temp=1.d0-(hlp*2.d0/diap-1.d0)**2
                temp= max(temp,1.d-6)
                fx=sqrt(temp)
!
! check for stratified conditions
! ucrit from ref. 1, eqn. 36
! fwgh from ref. 9
! critical liquid velocity (ul) is an ad hoc relation that is
! necessary to prevent stratified flow at unrealistic velocities
! this criterion will be changed when a suitable correlation is
! found in the literature
!
                sip=diap*fx
                IF (.NOT.(altm.GT..999d0.OR.altm.LT.       &
   &             .001d0))THEN
                 IF (ihor.NE.2) THEN
                  ul=0.2d0/(1.d0-alpfr)
                  ul= max(ul,veclct)
                  ug=((avlj*(1.d0-alpfr)/0.2d0)**(-       &
   &               .435d0))/alpfr
                  ug= max(ug,vecvct)
                 ENDIF
                 ff=hl*2.d0/dia-1.d0
                 si=dia*sqrt(1.d0-ff*ff)
                 sl=dia*(3.14159d0-acos(ff))
                 sg=3.14159d0*dia-sl
                 area=3.14159d0*.25d0*dia*dia
                 areal=area*(1.d0-altm)
                 areag=altm*area
                 dg=4.d0*areag/(si+sg)
```

```
                dl=4.d0*areal/sl
                dg= max(0.001d0, min(10.0d0,dg/dia))    &
     &           *hd(j)
                dl= max(0.001d0, min(10.0d0,dl/dia))    &
     &           *hd(j)
                reg= max(rv*avvj*dg/vscv,100.d0)
                rel= max(rl*avlj*dl/vscl,100.d0)
!
!  stratified flow wall drag (ref.14)
!
                wfvj=2.d0*.046d0*hd(j)/(dg*reg**0.2d0)
                IF (reg.LT.1502.d0) wfvj=2.0d0*(16.d0/reg) &
     &           *hd(j)/dg
                wflj=2.d0*.046d0*hd(j)/(dl*rel**0.2d0)
                IF (rel.LT.1502.d0) wflj=2.0d0*(16.d0/rel) &
     &           *hd(j)/dl
!
!
!    multiply the wall drag by a design-multipler factor
!
                wflj=wflj*wfmfl(j)
                wfvj=wfvj*wfmfv(j)
!  stratified flow interfacial friction (ref.9)
!  reg limits changed for smooth transitions
!
                fwg3=16.d0/reg
                IF (reg.GT.1189.d0) fwg3=0.079d0/(reg**0   &
     &           .25d0)
                IF (reg.GT.1.145d5) fwg3=0.0008d0+0.05525  &
     &           /(reg**0.237d0)
                cfhf=0.92d0*fwg3*rv*si*fifst/area
!
!  weighting factors between strat. flow regime and other regimes
!
                IF (ihor.NE.2) THEN
                  wxg=(stfru-avvj/ug)/(stfru-stfrl)
                  wxg= max( min(1.d0,wxg),0.d0)
                  wxl=(stflu-avlj/ul)/(stflu-stfll)
                  wxl= max( min(1.d0,wxl),0.d0)
                  wfalp1=(alpfr-0.5d0)/0.25d0
                  wfalp2=(0.75d0-alpfr)/0.25d0
                  wfhf=wxg*wfalp1+wxl*wfalp2
                  IF (alpfr.LE.0.5d0) wfhf=wxl
                  IF (alpfr.GE.0.75d0) wfhf=wxg
                  IF (ihutze.EQ.1) wfhf=1.0d0
                ENDIF
              ENDIF
            ENDIF
          ENDIF
        ENDIF
      ENDIF
    ENDIF
  ENDIF
  IF (ihor.EQ.0) wfhf=0.d0
  IF (ihor.EQ.2) wfhf=1.d0
```

```
        IF (ihor.EQ.3) dhldz=0.d0
        dhldz=wfhf*dhldz
!
! ************************************************
! ***** bubbly-slug regime              *****
! ************************************************
!
        IF (alpfr.LT.alw2) THEN
         gav=abs((1.0d0-alpl)*rl*vlnj+alpv*rv*vvnj)
         xslug=0.0d0
         alpb= max( min(altmf,alw1),alpbcd)
         IF (alpb.GT..3d0) xslug=(2700.d0-gav)/700.d0
         xslug= max( min(xslug,1.0d0),0.0d0)
         xslug=xslug*(5.d0*alpb-1.5d0)
         xslug= max( min(xslug,1.0d0),0.0d0)
!
! profile slip factor, ref.4, eqn.70
!
         c0=1.2d0-0.2d0*sqrt(rv/rl)
         c1=(1.0d0-c0*alpb)/(1.0d0-alpb)
         vrfac=((c1*avvj-c0*avlj)**2)/(avr**2)
         vrfac= max( min(vrfac,1.0d0),vrfmin)
         IF (alpfr.LT.alpbcd.OR.altmf.LT.alpbcd) vrfac=1.0d0
!
! bubble diameter, ref.12
!
         hdb=2.0d0/sqrt(gc*dro/sig)
         bmax= min(20.d0*hdb,0.9d0*hd(j))
         hdb= min(bmax,hdb)
         hdb= max(hdb,bmin)
         hdbp=hdb
         hdb=bmax*xslug+(1.d0-xslug)*hdb
!
! drag coefficient ref.7,table5.1, and ref.6
! cfi, ref. 5, eqn. 14,15
!
         cd=180.d0
         reyvb=avr*hdb*rl/vscl
         IF (reyvb.GT..1031d0.AND.reyvb.LT.989.d0) cd=18.d0/reyvb &
   &      *(1.0d0+0.15d0*reyvb**0.687d0)
         IF (reyvb.GE.989.d0) cd=.33d0
         cfti1=cd*vrfac*fifbs*alpb*rl/hdb
         IF (genTab(cco)%type.EQ.prizrh) cfti1=0.33d0*fifbs*alpb  &
   &      *rl/hdbp
         IF (alpfr.LE.alw1) GOTO 2021
        ENDIF
!
! ************************************************
! ***** annular-mist regime             ****
! ************************************************
!
        alpam= max(alpfr,alw2)
        vdis=(sig*gc*dro/(rv*rv))**0.25d0
        ajgmn=fdis1*vdis*alpam
        ajgmx=fdis2*vdis*alpam
        ajle=(1.d0-alpam)*avlj
```

```
          ajg=alpam*avvj
          ajgdm= max(ajg,ajgmx)
          aj=ajg+ajle
          vcrit=1.456d0*vdis*(vscv*vdis/sig)**(-0.1667d0)
          rele=rl*ajle*hd(j)/vscl
          wee=(rv*ajg*ajg*hd(j)/sig)*((dro/rv)**0.3333d0)
          eeq=tanh(7.25d-7*(wee**1.25d0)*(rele**0.25d0))
          eeq= max( min(1.0d0,eeq),0.0d0)
!
!  droplet radius, ref. 10, eqn. 42
!
          reg=hd(j)*ajg*rv/vscv
          regdm=reg*ajgdm/ajg
          droa=(rv/rl)**0.3333d0
          dcua=(vscv/vscl)**0.6666d0
          rd=(0.005d0*sig*dcua*(reg**0.6666d0))/(droa*rv*ajg*ajg)
          rd= max( min(rd,rdmax),rdmin)
          rddm=(0.005d0*sig*dcua*(regdm**0.6666d0))/(droa*rv*ajgdm  &
     &     *ajgdm)
          rddm= max( min(rddm,rdmax),rdmin)
!
!  the Kataoka and Ishii model was used.
!  relations between the entrainment and alpd, drift velocities,
!  and interfacial areas are given by Eq.(80)-Eq.(91) of Ref.3.
!
          alpd=eeq*ajle/ajg
          alpd= max( min(alpd,0.3333d0),almin)
          alpc=alpam/(1.d0-alpd)
          alpc= min(1.d0,alpc)
          alpf=1.d0-alpc
          IF ((1.d0-eeq).LT.1.d-10) alpf=0.d0
          alpf= max( min(1.d0-alw2,alpf),1.d-6)
          IF (aj.LT.vcrit) THEN
           avrd=abs(1.4142d0*vdis)*(1.d0-alpd)**1.5d0
          ELSE
           avrd=0.5d0*rd*((gc*dro)**2/(rv*vscv))**0.3333d0*(1.d0  &
     &      -alpd)**1.5d0
          ENDIF
          avrd= min(avvj-avlj,avrd)
          avrd= max(avrd,vrtcut)
          dftvr=sqrt((1.d0+75.d0*alpf)*rv/((sqrt(alpc)*rl)))
          avrf=abs((aj+sqrt(dro*gc*hd(j)*alpf/(0.015d0*rl)))/(alpc  &
     &     +dftvr))
          avrf= min(avvj-avlj,avrf)
          avrf= max(avrf,vrtcut)
          avrm=(alpf*avrf+alpd*avrd)/ max(1.d0-alpam,1.d-6)
!
!  fi for the liquid film, Ref. 8, Eq.(11-19)
!
          fi=0.005d0*(1.d0+75.d0*alpf)
          fiff=2.d0*fi*rv*avrf**2*sqrt(1.d0-alpf)/hd(j)
!
!  cim, ref.5,eqn.14,32-34
!
          cmum=vscv/((1.0d0-alpd)**2.5d0)
          red=2.d0*rv*rd*avrd/cmum
```

```
          red= max(red,1.d0)
          cd=24.d0*(1.d0+0.1d0*red**0.75d0)/red
          cim=0.45d0*(1.d0-alpf)*alpd*cd*rv/rd
          fifd=cim*avrd**2
          redmn=2.d0*rv*rdmin*veclct/cmum
          cdmx=24.d0/redmn
          cimdm=0.165d0*alpam*rv/rddm
          cimdm= max(cim,cimdm)
!
!  combine cim and cia into ci2 using momentum weighting
!
          cftmx=0.45d0*(1.d0-alpf)*cdmx
          cftmn=0.01d0*rv/hd(j)
          cfti2=(fifd+fiff)/(avrm*avrm+1.d-20)
          cfti2= max( min(cfti2,cftmx),cftmn)
          IF (genTab(cco)%type.EQ.prizrh) cfti2=cimdm
          IF (alpfr.LT.alw2) THEN
!
!  transition regime - linear weighting
!
            wx=aw*alpfr+bw
            wx= max( min(wx,1.0d0),0.0d0)
            cfti3=wx*cfti2+(1.d0-wx)*cfti1
            cfti1=0.0d0
            cfti2=0.0d0
          ENDIF
        ENDIF
!
!   Set the flow regime number, a(lregnm+j-1)
!
!      a(lregnm+j-1) is initialized to zero.
!
!      a(lregnm+j-1) = 1.0 is pure bubbly-slug flow.
!      Values of a(lregnm+j-1) between 1.0 and 2.0 signify flow
!      becoming horizontally stratified from the bubbly-slug regime.
!
!      a(lregnm+j-1) = 3.0 is "pure" transition flow.
!      Values of a(lregnm+j-1) between 3.0 and 4.0 signify flow
!      becoming horizontally stratified from the transition regime.
!
!      a(lregnm+j-1) = 5.0 is pure annular-mist flow.
!      Values of a(lregnm+j-1) between 5.0 and 6.0 signify flow
!      becoming horizontally stratified from the annular-mist regime.
!
!      a(lregnm+j-1) = -1.0 is an indication of an error.
!
 2021     IF ((cfti1.GT.0.0d0).AND.(cfti2.EQ.0.0d0).AND.(cfti3.EQ.0   &
    &    .0d0))THEN
          regnm(j)=1.0d0
        ELSEIF ((cfti2.GT.0.0d0).AND.(cfti1.EQ.0.0d0).AND.(cfti3.EQ  &
    &    .0.0d0)) THEN
          regnm(j)=5.0d0
        ELSEIF ((cfti3.GT.0.0d0).AND.(cfti1.EQ.0.0d0).AND.(cfti2.EQ  &
    &    .0.0d0)) THEN
          regnm(j)=3.0d0
        ELSE
```

```
          regnm(j)=-1.0d0
        ENDIF
!
        IF (regnm(j).NE.-1.0d0) regnm(j)=regnm(j)+wfhf
!
!  total interfacial drag
!
        cfti=(1.0d0-wfhf)*(cfti1+cfti2+cfti3)+wfhf*cfhf
        cfti=  max(ciminc*dro,cfti)
        IF (stdyst.EQ.0) THEN
!
!  average coefficients and limit changes over the
!  last time step
!
          IF (wfhf.GT.0.0d0) THEN
            fdf1=  min(fui1**(  min(delt*fri1,explim)),1.1d0)
            fdf2=  max(fui2**(  min(delt*fri2,explim)),0.9d0)
          ELSE
            fdf1=  min(fui1**(  min(delt*fri1,explim)),fifr)
            fdf2=  max(fui2**(  min(delt*fri2,explim)),fifi)
          ENDIF
!
          citmax=cif(j)*fdf1
          citmin=cif(j)*fdf2
          cifn=  max(  min(cfti,citmax),citmin)
          IF ((cif(j).LE.1.0d-05).OR.(btestc(bit(j),changeVapVel).EQ &
     &       .1)) cifn=cfti
        ELSE
          cifn=0.1d0*cfti+0.9d0*cif(j)
        ENDIF
      ELSE
!  this is the else from the if test on abs(fricla).le.100
        cifn=0.d0
        wfhf=1.0d0
        sip=diap
      ENDIF
      cif(j)=cifn
      IF (nifsh.EQ.1) cif(j)=ccif
      h1(j)=wfhf
      h2(j)=sip*dxp
      h3(j)=dhldz
!
!    calculate additive friction losses
!
      alpa=0.5d0*(alppsv+alpmsv)
      fricl=  max(fricla,0.0d0)
      fricv=  max(-fricva,0.0d0)
      IF (abs(fricva).LE.1.0d20) fricv=fricl
      IF (nwsh.NE.0) fricv=fricva
      wfl(j)=wfl(j)+fricl+fricd(j)
      wfv(j)=wfv(j)+fricv
    ENDIF
    GOTO 781
!
!    here is flow area is about zero or dead end
  25  CONTINUE
```

```
      h1(j)=1.0d0
      IF (j.LE.ncells) THEN
!       Modified as part of SJC task: #3-1
        IF (genTab(cco)%isSJC.EQ.1) THEN
          diap=sqrt(1.27323d0*bd2%vol/bd2%dx)
          CALL level(hlp,bd2%alp,diap)
          temp=  max(1.0d-06,1.0d0-(2.0d0*hlp/diap-1.0d0)**2)
          h2(j)=bd2%dx*diap*sqrt(temp)
        ELSE
          diap=sqrt(1.27323d0*vol(j)/dx(j))
          CALL level(hlp,alp(j),diap)
          temp=  max(1.0d-06,1.0d0-(2.0d0*hlp/diap-1.0d0)**2)
          h2(j)=dx(j)*diap*sqrt(temp)
        ENDIF
      ENDIF
!
  781  CONTINUE
     ENDDO
     RETURN
     END SUBROUTINE FricIf
!
     SUBROUTINE FLoss(cfz,wfv,istrt,nff,fa,bd1,bd2,vm,hd,ncells,dx,vol)
!
     USE OneDDat
     USE IntrType
     USE CompTyp
     USE Flt
     USE Bad

     IMPLICIT NONE
!
     REAL(sdk) areap,aream,fkfac,volje,voljb,dxje,dxjb
     INTEGER(sik) j,ncp1,msctmp,nfft,ncells,istrt
     REAL(sdk) cfz(:),wfv(:),fa(:),vm(:),hd(:),dx(:),vol(:),nff(:)
     TYPE(bdT) bd1,bd2
!
!
!    when nff.lt.0, evaluate the irreversible form-loss k-factor
!    for an abrupt contraction or expansion between mesh cells
!    except for the three interfaces of jcell for a tee component
!    where an input check warns if their nff values are negative and
!    a bypass test is done below in case the warning was ignored
     ncp1=ncells+1
     msctmp=msc-istrt+1
     DO j=1,ncp1
!      Added as part of SJC task: #2-6
       nfft=iabs(int(nff(j)))
       IF (nfft.NE.0) THEN
         IF(genTab(cco)%isSJC.NE.0) THEN
           dxjb=bd1%dx
           dxje=bd2%dx
           voljb=bd1%vol
           volje=bd2%vol
           IF (((bd1%type.EQ.plenh).OR.(bd1%type.EQ.vsslh)).AND       &
    &       .(bd1%faFrac.GT.0.0d0)) voljb=voljb/bd1%faFrac
           IF (((bd2%type.EQ.plenh).OR.(bd2%type.EQ.vsslh)).AND       &
```

```
&     .(bd2%faFrac.GT.0.0d0)) volje=volje/bd2%faFrac
      ELSEIF ((j.EQ.1) .AND. (ncp1.NE.1)) THEN
       dxje=dx(j)
       dxjb=bd1%dx
       volje=vol(j)
       voljb=bd1%vol
       IF (((bd1%type.EQ.plenh).OR.(bd1%type.EQ.vsslh)).AND       &
&     .(bd1%faFrac.GT.0.0d0)) voljb=voljb/bd1%faFrac
      ELSEIF (j.NE.ncp1) THEN
       dxje=dx(j)
       dxjb=dx(j-1)
       volje=vol(j)
       voljb=vol(j-1)
      ELSE
       dxje=bd2%dx
       dxjb=dx(j-1)
       volje=bd2%vol
       IF (((bd2%type.EQ.plenh).OR.(bd2%type.EQ.vsslh)).AND       &
&     .(bd2%faFrac.GT.0.0d0)) volje=volje/bd2%faFrac
       voljb=vol(j-1)
      ENDIF
      IF (int(nff(j)).LT.0) THEN
       IF (.NOT.((((genTab(cco)%type.EQ.teeh).OR.(genTab(cco)%type  &
&     .EQ.jetph).OR.(genTab(cco)%type.EQ.turbh).OR.              &
&     (genTab(cco)%type.EQ.heatrh)).AND.((j.EQ.                  &
&     msctmp).OR.(j.EQ.msctmp+1)).AND.(msctmp.GE.1)).OR.         &
&     (((bd1%type.EQ.teeh).OR.(bd1%type.EQ.heatrh)).AND.(j.EQ.1)  &                    &
&     .AND.(bd1%ellFlag.GT.0.0d0)).OR.(((bd2%type.EQ.teeh).OR.    &
&     (bd2%type.EQ.heatrh)).AND.(j.EQ                            &
&     .ncp1).AND.(bd2%ellFlag.GT.0.0d0)).OR.((j.EQ.1).AND.(msctmp  &
&     .LE.-1)))) THEN
        areap=volje/dxje
        aream=voljb/dxjb
        IF ((aream.GT.0.0d0).AND.(fa(j).GT.0.0d0).AND.(areap.GT.0  &
&       .0d0))THEN
         CALL fwkf(aream,fa(j),areap,vm(j),fkfac)
         cfz(j)=cfz(j)+hd(j)*fkfac/(dxjb+dxje)
         wfv(j)=wfv(j)+hd(j)*fkfac/(dxjb+dxje)
        ENDIF
       ENDIF
      ENDIF
     ENDIF
    ENDDO
    RETURN
    END SUBROUTINE FLoss

    SUBROUTINE fwkf(aream,fa,areap,vm,fkfac)
!
!   BEGIN MODULE USE
    USE IntrType
    USE OneDDat
!
    IMPLICIT NONE
!
!   Declaration Generated by genImpDecs.pl 5/98
    REAL(sdk) a1,a2,aream,areap,fa,fkfac,vm
```

```
!
!
!     subroutine fwkf evaluates the form-loss k-factor for an
!     abrupt contraction or expansion between mesh cells
!
!     arg-list
!     variable i/o description
!
!     aream    i  average flow area of the mesh cell to the left, m**2
!     fa       i  flow area of the interface between mesh cells, m**2
!     areap    i  average flow area of the mesh cell to the right, m**2
!     vm       i  mixture velocity used to define flow direction, m/s
!     fkfac    o  k-factor to be applied to the squared velocity at
!                 the mesh-cell interface
!
!     area ratios are applied to the momentum-convection term
!     when ary=1.0 and not applied when ary=0.0
!
!     to obtain the form-loss k-factor that is to be multi-
!     plied by the interface velocity squared, the form-loss
!     k-factor that is multiplied by the minimum flow area
!     mesh-cell velocity squared needs to be multiplied
!     by a2*a2 where a2 is the ratio of the interface flow
!     area (fa) to the minimum mesh-cell averaged flow area
!
!     for an abrupt contraction, the form-loss k-factor that
!     is multiplied by the minimum flow area mesh-cell vel-
!     ocity squared is 0.5-0.7*a1+0.2*a1*a1 where a1 is the
!     ratio of the minimum to maximum mesh-cell averaged
!     flow areas.  when ary=1.0, the momentum-convection term
!     evaluates the reversible form loss of bernoulli and
!     needs the above irreversible form-loss k-factor added
!     in.  when ary=0.0, the momentum-convection term deter-
!     mines the reversible form loss of bernoulli and an
!     irreversible form-loss k-factor of (1-a1)**2 = 1.0
!     -2.0*a1+a1*a1.  subtracting an irreversible form-loss
!     k-factor of 0.5-1.3*a1+0.8*a1*a1 results in the correct
!     irreversible form-loss k-factor getting applied.
!
!     for an abrupt expansion, the form-loss k-factor that is
!     multiplied by the minimum flow area mesh-cell velocity
!     squared is (1-a1)**2 = 1.0-2.0*a1+a1*a1 where a1 is the
!     ratio of the minimum to maximum mesh-cell averaged flow
!     areas.  when ary=1.0, the momentum-convection term
!     evaluates the reversible form loss of bernoulli and
!     needs the above irreversible form-loss k-factor added
!     in.  when ary=0.0, the momentum-convection term deter-
!     mines the reversible form loss of bernoulli and an
!     irreversible form-loss k-factor of (1-a1)**2 and thus
!     needs no form-loss k-factor added in to apply the cor-
!     rect irreversible form-loss k-factor.
!
      IF (vm.LT.0.0d0) THEN
!
!     flow is in the negative direction
!
```

```
      IF (aream.LT.areap) THEN
!
!     flow is into an abrupt contraction
!
        a1=aream/areap
        a2=fa/aream
        IF (ary.GE.0.5d0) THEN
          fkfac=(0.5d0-a1*(0.7d0-a1*0.2d0))*a2*a2
        ELSE
          fkfac=-(0.5d0-a1*(1.3d0-a1*0.8d0))*a2*a2
        ENDIF
      ELSE
!
!     flow is into an abrupt expansion
!
        a1=areap/aream
        a2=fa/areap
        IF (ary.GE.0.5d0) THEN
          fkfac=(1.0d0-a1)*(1.0d0-a1)*a2*a2
        ELSE
          fkfac=0.0d0
        ENDIF
      ENDIF
!
!     flow is in the positive direction
!
      ELSEIF (areap.LT.aream) THEN
!
!     flow is into an abrupt contraction
!
       a1=areap/aream
       a2=fa/areap
       IF (ary.GE.0.5d0) THEN
         fkfac=(0.5d0-a1*(0.7d0-a1*0.2d0))*a2*a2
       ELSE
         fkfac=-(0.5d0-a1*(1.3d0-a1*0.8d0))*a2*a2
       ENDIF
      ELSE
!
!     flow is into an abrupt expansion
!
       a1=aream/areap
       a2=fa/aream
       IF (ary.GE.0.5d0) THEN
         fkfac=(1.0d0-a1)*(1.0d0-a1)*a2*a2
       ELSE
         fkfac=0.0d0
!
       ENDIF
      ENDIF
      RETURN
      END SUBROUTINE fwkf
      SUBROUTINE level(hl,alp,d)
!
!     BEGIN MODULE USE
!
```

```
      IMPLICIT NONE
!
!     Declaration Generated by genImpDecs.pl 5/98
      REAL(sdk) al,alp,d,hl
!
!
!     uses a curve fit to obtain the water level in
!     a cylindrical pipe as a function of void fraction.
!
      al=alp
      IF (alp.GT..5d0) al=1.d0-alp
      IF (al.GT..001d0) THEN
        hl=(1.d0-.70269591d0*al**.6666666667d0-.034146667*al-          &
     &  .161023911d0*al**2)*d
      ELSE
        hl=(1.d0-7.612668d0*al)*d
      ENDIF
      IF (alp.GE..5d0) hl=d-hl
      RETURN
      END SUBROUTINE level
!
      END MODULE Models
```

```
      SUBROUTINE wdrag(length,alp,rol,rov,visv,visl,vl1,vv1,hd,vl,vv,    &
     & eps,cfwl,cfwv,romk,gavw,chi,rcmu,reyg,a,b)
!
!     BEGIN MODULE USE
      USE IntrType
!
      IMPLICIT NONE
!
!     Declaration Generated by genImpDecs.pl 5/98
      INTEGER(sik) ii,length
!
!
!     r. nelson        feb '86            q9
!
!     this subroutine calculates the coefficient of friction
!     for the liquid and vapor at the wall.
!     the coefficients have been lifted from trac 3d vessel.
!
!******************** arguement list nomenclature ***********************
!
! variable  i/o              definition
!
! length     i    length of vector
! alp        i    void fraction
! rol        i    liquid density
! rov        i    vapor density
! visv       i    vapor viscosity
! visl       i    liquid viscosity
! vl1        i    liquid velocity for the mass flux calculation
! vv1        i    vapor velocity for the mass flux calculation
! hd         i    hydraulic diameter
! vl         i    liquid velocity
! vv         i    vapor velocity
! eps        i    wall surface roughness divided by hd
! cfwl       o    coefficient of friction for liquid to wall
! cfwv       o    coefficient of friction for vapor to wall
! romk      dum   dummy vector, used internally as mixture density
! gavw      dum   dummy vector, used internally as mass flux
! chi       dum   dummy vector, used internally
! rcmu      dum   dummy vector, used internally
! reyg      dum   dummy vector, used internally as vapor reynolds #
! a         dum   dummy vector, used internally
! b         dum   dummy vector, used internally
!
!
!
      REAL(sdk) a(*),alp(*),b(*),cfwl(*),cfwv(*),chi(*),eps(*),gavw(*),  &
     & hd(*),rcmu(*),reyg(*),rol(*),romk(*),rov(*),visl(*),visv(*),      &
     & vl(*),vl1(*),vv(*),vv1(*)
!
!
      DO ii=1,length
        romk(ii)=(1.0d0-alp(ii))*rol(ii)+alp(ii)*rov(ii)
        gavw(ii)=(1.d0-alp(ii))*rol(ii)*vl1(ii)+alp(ii)*rov(ii)*vv1(ii)
        chi(ii)=alp(ii)*rov(ii)/romk(ii)
        rcmu(ii)=chi(ii)/visv(ii)+(1.d0-chi(ii))/visl(ii)
```

```
      reyg(ii)=  max(100.0d0,gavw(ii)*hd(ii)*rcmu(ii))
cjoe       a(ii)=(2.457d0*log(1.d0/((7.d0/reyg(ii))**0.9d0+0.27d0*        &
cjoe     & eps(ii))))**16.0d0
cjoe       b(ii)=(37530.d0/reyg(ii))**16.0d0
cjoe       cfwl(ii)=4.d0*((8.0d0/reyg(ii))**12.0d0+1.d0/(a(ii)+b(ii))**    &
cjoe     & 1.5d0)**0.083333d0
      if(reyg(ii).gt.3000.d0)then
        cfwl(ii) = 2.*(0.0014+0.125/reyg(ii)**0.32)
      else
        cfwl(ii) = 2.* 25./reyg(ii)
      end if
      cfwv(ii)=cfwl(ii)
    ENDDO
!
    DO ii=1,length
      IF (abs(vl(ii)).GT.1.d-5) cfwl(ii)=cfwl(ii)*vl1(ii)*vl1(ii)/(vl  &
    &   (ii)*vl(ii))
      IF (abs(vv(ii)).GT.1.d-5) cfwv(ii)=cfwv(ii)*vv1(ii)*vv1(ii)/(vv  &
    &   (ii)*vv(ii))
!
    ENDDO
    RETURN
    END SUBROUTINE wdrag
```

```
      MODULE Materials
!
!     Begin Module Use
      USE IntrType
!
      IMPLICIT NONE
!
      TYPE matPropT
        INTEGER(sik) :: matb
        INTEGER(sik) :: ptbln
        REAL(sdk), DIMENSION(:),POINTER :: temp
        REAL(sdk), DIMENSION(:),POINTER :: rho
        REAL(sdk), DIMENSION(:),POINTER :: spHt
        REAL(sdk), DIMENSION(:),POINTER :: cond
        REAL(sdk), DIMENSION(:),POINTER :: emis
      END TYPE matPropT
!
      INTEGER(sik) nMtrls
!     nMtrls is the former variable nmat, this new name assures that
!     the variable is correct in all places and is more distinguished
!     from matrices, which are also 'mats'
!
      TYPE (matPropT),ALLOCATABLE,DIMENSION(:) :: matProp
      TYPE (matPropT),ALLOCATABLE,DIMENSION(:) :: matPropTemp
!
      LOGICAL :: rdMatDmp = .TRUE.
!     rdMatDmp is a flag used by the restart system to indicate that
!     this structure has been read, and to skip over it...
      LOGICAL :: useDmpMatProps = .FALSE.
!     useDmpMatProps is a flag for input that the nMtrls variable has
!     been applied to the dumpfile and should not try to read them
!     from input (tracin)
!
      CONTAINS
!
!     Allocation Routines
!
      SUBROUTINE AllocMProp
!     This subroutine performs the Equivalent of TracAllo for
!     Type MatPropT. It allocates the array of materials. Subroutine
!     AllocPrpTb allocates the property arrays for each material.
      IMPLICIT NONE
!
      INTEGER(sik) errorStatus
!
      ALLOCATE(matProp(nMtrls),STAT=errorStatus)
!
      IF (errorStatus.ne.0) then
        WRITE (*,*) 'Got allocate error code ',errorStatus
        WRITE (*,*) 'AllocMProp: Trouble allocating matProp'
        CALL tracstop()
      ENDIF
!
      END SUBROUTINE AllocMProp

      SUBROUTINE ReAllocMProp
```

```
!
    INTEGER(sik) nMtrlsOld,i,nPoints,errorStatus
!
    nMtrlsOld=SIZE(matProp)
    ALLOCATE(matPropTemp(nMtrlsOld),STAT=errorStatus)
    IF (errorStatus.NE.0) THEN
     WRITE (*,*) 'Got allocate error code ',errorStatus
     WRITE (*,*) 'AllocMProp: Trouble allocating matPropTemp'
     CALL tracstop()
    ENDIF
!
    DO i=1,nMtrlsOld
     nPoints=SIZE(matProp(i)%temp)
     CALL AllocPropTb(matPropTemp(i),nPoints)
     matPropTemp(i)%temp=matProp(i)%temp
     matPropTemp(i)%rho=matProp(i)%rho
     matPropTemp(i)%spHt=matProp(i)%spHt
     matPropTemp(i)%cond=matProp(i)%cond
     matPropTemp(i)%emis=matProp(i)%emis
    ENDDO
    DEALLOCATE(matProp)
!
    ALLOCATE(matProp(nMtrls),STAT=errorStatus)
    IF (errorStatus.ne.0) then
     WRITE (*,*) 'Got allocate error code ',errorStatus
     WRITE (*,*) 'AllocMProp: Trouble allocating matProp'
     CALL tracstop
    ENDIF
!
    DO i=1,nMtrlsOld
     CALL AllocPropTb(matPropTemp(i),nPoints)
     matProp(i)%temp=matPropTemp(i)%temp
     matProp(i)%rho=matPropTemp(i)%rho
     matProp(i)%spHt=matPropTemp(i)%spHt
     matProp(i)%cond=matPropTemp(i)%cond
     matProp(i)%emis=matPropTemp(i)%emis
    ENDDO
    DEALLOCATE(matPropTemp)
!
    RETURN
    END SUBROUTINE ReAllocMProp
!
    SUBROUTINE AllocPropTb(mPrpTb,nPoints)
!   This subroutine allocates the individual property arrays for each
!   material by way of the TracAllo Interface.
!
!   BEGIN Module USE
    USE Alloc
    USE IntrType
!
    IMPLICIT NONE
!
    TYPE(matPropT), INTENT(INOUT):: mPrpTb
    INTEGER(sik), INTENT(IN) :: nPoints
!
    CALL TracAllo(mPrpTb%temp,nPoints,'matProp%temp',0.0d0)
```

```
      CALL TracAllo(mPrpTb%rho,nPoints,'matProp%rho',0.0d0)
      CALL TracAllo(mPrpTb%spHt,nPoints,'matProp%spHt',0.0d0)
      CALL TracAllo(mPrpTb%cond,nPoints,'matProp%cond',0.0d0)
      CALL TracAllo(mPrpTb%emis,nPoints,'matProp%emis',0.0d0)
!
      END SUBROUTINE AllocPropTb
!
!     Materials property routines. Routines give the basic properties
!      of structural materials, heaters and fuels.
!
      SUBROUTINE mstrct(cw,cpw,emis,k,row,t,tmss)
!
!     BEGIN MODULE USE
      USE IntrType
      USE Io
!
      IMPLICIT NONE
!
!     Declaration Generated by genImpDecs.pl 5/98
      INTEGER(sik) itab,k,kmat,l,lng
!
!     Declaration Generated by genImpDecs.pl 5/98
      REAL(sdk) a1,a2,a3,a4,a5,a6,a7,a8,cpw,cw,emis,fctr,poly2,poly3,    &
     & poly4,poly5,poly7,row,t,tf,tfss,tmss,x
!
!
!     mstrct calculates values for density,specific heat,
!     thermal conductivity,and emissivity for stainless
!     steel types 304,316,347, a508 medium carbon steel,
!     and inconel 600 and 718
!
!     cpw    - specific heat, j/kg k
!     cw     - thermal conductivity, w/m k
!     emis   - emissivity
!     k      - material type
!              k=6 - stainless steel type 304
!              k=7 - stainless steel type 316
!              k=8 - stainless steel type 347
!              k=9 - medium carbon steel a508
!              k=10 - inconel 718
!              k=12 - inconel 600
!     row    - density, kg/m**3
!     t      - temperature, degrees kelvin
!     tc     - temperature, degrees celsius
!     tf     - temperature, degrees fahrenheit
!     tmss   - melting temperature of stainless steel, k
!
!
      REAL(sdk) c508(8),c600(6),cp508(4),cp600(8),cpss(5),               &
     & r304(3),r316(3),r508(4),r600(3),r718(4)
!
!
      DATA tfss/1700.0d0/
      DATA r600/5.261008d2,-1.345453d-2,-1.194367d-7/
      DATA cp600/.1014456d0,4.378952d-5,-2.046138d-8,3.418111d-11,       &
     & -2.060318d-13,3.682836d-16,-2.458648d-19,5.597571d-23/
```

```
      DATA c600/8.011332d0,4.643719d-3,1.872857d-6,-3.914512d-9,3       &
     &.475513d-12,-9.936696d-16/
      DATA cpss/426.17d0,0.43816d0,-6.3759d-04,4.4803d-07,-1.0729d-10/
      DATA r304/7984.0d0,-2.651d-01,-1.158d-04/
      DATA r316/8084.0d0,-4.209d-01,-3.894d-05/
      DATA cp508/400.48d0,0.4582d0,-6.5532d-04,5.3706d-07/
      DATA c508/66.1558d0,-1.4386d-02,-2.6987d-04,1.8306d-06,-6.0673d-09 &
     &,1.0524d-11,-9.1603d-15,3.1597d-18/
      DATA r508/7859.82d0,-2.6428d-02,-4.5471d-04,3.3111d-07/
      DATA r718/8233.4d0,-1.8351d-01,-9.8415d-06,-6.5343d-09/
      poly2(a1,a2,a3,x)=a1+x*(a2+x*a3)
      poly3(a1,a2,a3,a4,x)=a1+x*(a2+x*(a3+x*a4))
      poly4(a1,a2,a3,a4,a5,x)=a1+x*(a2+x*(a3+x*(a4+x*a5)))
      poly5(a1,a2,a3,a4,a5,a6,x)=a1+x*(a2+x*(a3+x*(a4+x*(a5+x*a6))))
      poly7(a1,a2,a3,a4,a5,a6,a7,a8,x)=a1+x*(a2+x*(a3+x*(a4+x*(a5+x*(a6  &
     & +x*(a7+x*a8))))))
!
      tf=(t-273.15d0)*1.8d0+32.0
      tmss=tfss
!
      IF (k.GE.50) THEN
!
!       tabular input for structural material
!
       kmat=1
       itab=1
       DO WHILE (k.NE.matProp(kmat)%matb)
        kmat=kmat+1
        IF (kmat.GT.nMtrls) GOTO 1001
       ENDDO
       GOTO 205
 1001  WRITE (imout,202) k,(matProp(kmat)%matb,kmat=1,nMtrls)
       WRITE (iout,202) k,(matProp(kmat)%matb,kmat=1,nMtrls)
       WRITE (itty,202) k,(matProp(kmat)%matb,kmat=1,nMtrls)
 202   FORMAT (/' *mstrct* material-property id number',i4,          &
     &  ' could not',' be found'/10x,                                &
     &  'in the list of user-specified material-',                   &
     &  'property id numbers:'/(10x,10i5))
       CALL error(1,'*mstrct* incorrect tabular mat. i.d.')
!
 205   CONTINUE
       lng=matprop(kmat)%ptbln
       IF (t.GE.matProp(kmat)%temp(1).AND.t.LE.                      &
     &   matProp(kmat)%temp(lng)) THEN
!
        DO WHILE ((itab.LT.lng).AND.(t.GE.matProp(kmat)%temp(itab+1)))
         itab=itab+1
        ENDDO
!
        IF (t.EQ.matProp(kmat)%temp(itab)) THEN
         row =matProp(kmat)%rho(itab)
         cpw =matProp(kmat)%spHt(itab)
         cw  =matProp(kmat)%cond(itab)
         emis=matProp(kmat)%emis(itab)
        ELSE
         fctr=(t-matProp(kmat)%temp(itab))/                          &
```

```
     &       (matProp(kmat)%temp(itab+1)-matProp(kmat)%temp(itab))
           row=matProp(kmat)%rho(itab)+(matProp(kmat)%rho(itab+1)      &
     &       -matProp(kmat)%rho(itab))*fctr
           cpw=matProp(kmat)%spHt(itab)+(matProp(kmat)%spHt(itab+1)    &
     &       -matProp(kmat)%spHt(itab))*fctr
           cw=matProp(kmat)%cond(itab)+(matProp(kmat)%cond(itab+1)     &
     &       -matProp(kmat)%cond(itab))*fctr
           emis=matProp(kmat)%emis(itab)+(matProp(kmat)%emis(itab+1)   &
     &       -matProp(kmat)%emis(itab))*fctr
         ENDIF
        ELSE
         WRITE (imout,2345) k,t
         WRITE (iout,2345) k,t
         WRITE (itty,2345) k,t
2345     FORMAT (/,'*material id=',i3,'; temp=',g12.2)
         CALL error(2,'*mstrct* temperature outside table range')
        ENDIF
       ELSE
        l=k-5
        IF (k.EQ.12) l=k-6
!
        IF (l.NE.1) THEN
         IF (l.EQ.2) THEN
!
!       stainless steel type 316
!
!       ref-- l. leibowitz,et al.,"properties for lmfbr safety analysis,"
!       argonne national laboratory report anl-cen-rsd-76-1 (1976).
!
           row=poly2(r316(1),r316(2),r316(3),t)
           cpw=poly4(cpss(1),cpss(2),cpss(3),cpss(4),cpss(5),tf)
           cw=9.248d0+1.571d-02*t
           emis=0.84d0
           IF (cpw.LT.0.0d0) THEN
            WRITE (imout,6) k,t,cpw
            WRITE (iout,6) k,t,cpw
            WRITE (itty,6) k,t,cpw
            CALL error(1,'*mstrct*illegal negative cpwall ')
           ENDIF
!
           GOTO 1000
         ELSEIF (l.EQ.3) THEN
!
!       stainless steel type 347
!
!       ref-- w.l. kirchner,"reflood heat transfer in a light
!           water reactor,"u.s. nuclear regulatory commission
!           report nureg-0106 (1976).
!
           row=7913.0d0
           cpw=502.416d0+0.0984*(tf-240.0d0)
           cw=14.1926d0+7.269d-03*tf
           emis=0.84d0
           GOTO 1000
         ELSEIF (l.EQ.4) THEN
!
```

```
!    medium carbon steel a508
!
!    ref-- "nuclear systems materials handbook,"vol. 1 design data,
!         hanford engineering development laboratory,tid 26666.
!
          row=poly3(r508(1),r508(2),r508(3),r508(4),tf)
          cpw=poly3(cp508(1),cp508(2),cp508(3),cp508(4),tf)
          cw=poly7(c508(1),c508(2),c508(3),c508(4),c508(5),c508(6),    &
     &      c508(7),c508(8),tf)
          emis=0.84d0
          IF (cpw.LT.0.0d0) THEN
            WRITE (imout,6) k,t,cpw
            WRITE (iout,6) k,t,cpw
            WRITE (itty,6) k,t,cpw
            CALL error(1,'*mstrct*illegal negative cpwall ')
          ENDIF
!
          GOTO 1000
        ELSEIF (I.EQ.5) THEN
!
!    inconel 718
!
!    ref-- "nuclear systems materials handbook,"vol. 1 design data,
!         hanford engineering development laboratory,tid 26666.
!
          row=poly3(r718(1),r718(2),r718(3),r718(4),tf)
          cpw=418.18d0+0.1204*tf
          cw=10.8046d0+8.829d-03*tf
          emis=0.84d0
          IF (cpw.LT.0.0d0) THEN
            WRITE (imout,6) k,t,cpw
            WRITE (iout,6) k,t,cpw
            WRITE (itty,6) k,t,cpw
            CALL error(1,'*mstrct*illegal negative cpwall ')
          ENDIF
!
          GOTO 1000
        ELSEIF (I.EQ.6) THEN
!
!
!    inconel 600 - nuclear systems materials handbook, vol. i,
!              hedl, tid 26666.
!
!    row equation good to 1033 k
!    cpw and cw equations good to 1144 k
!
          row=16.01846d0*poly2(r600(1),r600(2),r600(3),tf)
          cpw=4.1868d3*poly7(cp600(1),cp600(2),cp600(3),cp600(4),      &
     &      cp600(5),cp600(6),cp600(7),cp600(8),tf)
          cw=1.729577d0*poly5(c600(1),c600(2),c600(3),c600(4),c600(5), &
     &      c600(6),tf)
          emis=0.84d0
          IF (cpw.LT.0.0d0) THEN
            WRITE (imout,6) k,t,cpw
            WRITE (iout,6) k,t,cpw
            WRITE (itty,6) k,t,cpw
```

```
          CALL error(1,'*mstrct*illegal negative cpwall ')
         ENDIF
!
!
         GOTO 1000
        ELSE
         WRITE (imout,5) k
         WRITE (iout,5) k
         WRITE (itty,5) k
    5     FORMAT (/' *mstrct* material-property id number',i4,        &
    &      'is not 6,7,8,9,10,12 or .ge.50')
         CALL error(1,                                    &
    &      '*mstrct*illegal index in computed go to statement')
        ENDIF
       ENDIF
!
!    stainless steel type 304
!
!    ref-- l. leibowitz,et al.,"properties for lmfbr safety analysis,"
!    argonne national laboratory report anl-cen-rsd-76-1 (1976).
!
      row=poly2(r304(1),r304(2),r304(3),t)
      cpw=poly4(cpss(1),cpss(2),cpss(3),cpss(4),cpss(5),tf)
      cw=8.116d0+1.618d-02*t
      emis=0.84d0
      IF (cpw.LT.0.0d0) THEN
         WRITE (imout,6) k,t,cpw
         WRITE (iout,6) k,t,cpw
         WRITE (itty,6) k,t,cpw
    6     FORMAT (/' *mstrct* material-property id number',i4,        &
    &      ' negative cpwall at Tw= ',f10.2,' K',/,25x,'cpwall= '     &
    &      ,g14.3,' J/kg-K')
         CALL error(1,'*mstrct*illegal negative cpwall ')
       ENDIF
      ENDIF
 1000 RETURN
      END SUBROUTINE mstrct
!
      SUBROUTINE mfuel(burn,cw,cpw,emis,fpuo2,ftd,row,t,tmuo2)
      IMPLICIT NONE
!
!    Declaration Generated by genImpDecs.pl 5/98
      REAL(sdk) beta,burn,c1,c2,c3,c4,corr,cp1,cp2,cp3,cp4,cpw,cu1,cu10, &
     & cu11,cu12,cu13,cu14,cu2,cu3,cu4,cu5,cu6,cul1,cul2,cul3,cul4,cw,   &
     & d1,d2,d3,d4,d5,d6,dens,dlol,emis,fpuo2,ftd,pu1,pu2,pu3,pu4,pu5,   &
     & pu6,row,rpuo2,ruo2,specht,t,tc,thexp,tmuo2
!
!
!    mfuel calculates values for density,specific heat,thermal
!    conductivity, emissivity, elastic modulus, thermal
!    expansion, and melting temperature for uo2 and uo2-puo2 fuels
!
!    ref-- d.l. hagrman, et al.,"matpro - version 11,"tree-nureg-
!         1280, rev 1,idaho national engineering laboratory (1980).
!
!    beta   - porosity coefficient
```

```
!    burn   - burnup, mwd/mtu
!    cpw    - specific heat, j/kg k
!    cw     - thermal conductivity, w/m k
!    emis   - emissivity
!    fpuo2  - fraction of plutonium-oxide in fuel
!    ftd    - fraction of theoretical density
!    row    - density, kg/m**3
!    t      - temperature, k
!    tc     - temperature, c
!    thexp  - thermal expansion
!    tmuo2  - melting point temperature of uo2, k
!
!
!
     DATA cu1,cu2,cu3,cu4,cu5,cu6/19.145d0,7.8473d-04,5.6437d06,535    &
    &.285d0,37694.6d0,1.987d0/
     DATA cu10,cu11,cu12,cu13,cu14/40.4d0,464.0d0,1.216d-04,1.867d-03, &
    & 0.0191d0/
     DATA cul1,cul2,cul3,cul4/1.0d-5,3.0d-3,4.0d-4,5000.d0/
     DATA cp1,cp2,cp3,cp4/9.0d-6,2.7d-3,7.0d-2,5072.5d0/
     DATA pu1,pu2,pu3,pu4,pu5,pu6/19.53d0,9.25d-04,6.02d06,539.0d0,    &
    & 40100.0d0,1.987d0/
     DATA ruo2,rpuo2/1.097d04,1.146d04/
!
     specht(t,d1,d2,d3,d4,d5,d6)=15.496d0*(d1*d4**2*exp(d4/t)/(t**2    &
    &*(exp(d4/t)-1.0d0)**2)+2.0d0*d2*t+d3*d5/(d6*t**2)*exp(-d5/(d6*t)))
!
!    c4 = ed/k, dlol = delta l over l
!
     dlol(c1,c2,c3,c4,t)=c1*t-c2+c3*exp(-c4/t)
!
!
     tc=t-273.15d0
!
!    melting temperature of uo2
     tmuo2=3113.0d0-(0.0032d0*burn)
!
!
!    emissivity of uo2 and uo2-puo2
!
     IF (t.LE.1000.0d0) THEN
       emis=0.8707d0
     ELSEIF (t.GT.2050.0d0) THEN
       emis=0.4083d0
     ELSE
       emis=1.311d0-4.404d-04*t
     ENDIF
!
     IF (fpuo2.GT.0.0d0) THEN
!
!    mixed oxide uo2-puo2 fuel properties
!
!    thermal expansion
       thexp=dlol(cp1,cp2,cp3,cp4,t)
!
!    density of uo2-puo2 corrected for temperature and fraction
```

```
!    of theoretical density and fraction of puo2
!
      dens=ftd*((1.0d0-fpuo2)*ruo2+fpuo2*rpuo2)
!
      row=dens/(1.0d0+3.0*thexp)
!
!    specific heat
      cpw=specht(t,pu1,pu2,pu3,pu4,pu5,pu6)
!
!    thermal conductivity
      beta=1.43d0
      corr=100.0d0*(ftd/(1.0d0+beta*(1.0d0-ftd)))*(1.0d0+0.04*beta)/0  &
     & .96d0
      IF (tc.GT.1550.0d0) THEN
        cw=corr*(0.0171d0+1.54d-04*exp(1.71d-03*tc))
!
!
!
      ELSE
        cw=corr*(33.0d0/(375.0d0+tc)+1.54d-04*exp(1.71d-03*tc))
      ENDIF
    ELSE
!
!    uranium dioxide properties
!
!    density of uo2 corrected for temperature and fraction of
!    theoretical density
!
!    thermal expansion for uo2
!
      thexp=dlol(cul1,cul2,cul3,cul4,t)
!
      row=ruo2*ftd/(1.0d0+3.0*thexp)
!
!
!    specific heat
      cpw=specht(t,cu1,cu2,cu3,cu4,cu5,cu6)
!
!    thermal conductivity
      beta=2.58d0-5.8d-04*tc
      corr=100.0d0*(1.0d0-beta*(1.0d0-ftd))/(1.0d0-0.05*beta)
      IF (tc.GT.1650.0d0) THEN
        cw=corr*(cu14+cu12*exp(cu13*tc))
!
      ELSE
        cw=corr*(cu10/(cu11+tc)+cu12*exp(cu13*tc))
      ENDIF
    ENDIF
    RETURN
    END SUBROUTINE mfuel
!
    SUBROUTINE mgap(cgap,drgap,gmix,pgap,tgap)
!
!    BEGIN MODULE USE
    USE EosData
!
```

```
      IMPLICIT NONE
!
!     Declaration Generated by genImpDecs.pl 5/98
      INTEGER(sik) i,j
!
!     Declaration Generated by genImpDecs.pl 5/98
      REAL(sdk) a1,a2,a3,a4,altg,cc5,cc6,cc7,cc8,cc9,cgap,drgap,factor,  &
     & pgap,phi,poly2,poly3,psi,rc,rmw,rough,sum,tgap,x
!
!
!     mgap calculates values for the thermal conductivity
!     of the gap gas mixture
!
!     c     - individual gas thermal conductivities, w/m k
!     cgap  - output thermal conductivity of gap gas mixture,
!             w/m k
!     drgap - gap thickness, m
!     gmix  - mole fraction of gas mixture normalized to one
!             constituents
!             1 - helium
!             2 - argon
!             3 - krypton
!             4 - xenon
!             5 - hydrogen
!             6 - air/nitrogen
!             7 - water vapor
!     pgap  - gap gas pressure, n/m**2
!     rough - rms roughness for fuel rod, m
!     tgap  - temperature of gap gas, k
!
!     ref-- d.l. hagrman,et al.,"matpro - version 11,"inel
!          report nureg/cr-0497 tree 1280 (1979).
!
!     reference for helium-- J. Goodman et al.,"The Thermodynamic
!          and Transport Properties of Helium," GA-A13400 (1975).
!
      REAL(sdk) c(7),cc1(6),cc2(6),cc3(3),cc4(4),gmix(7),gmw(7)
!
!
      DATA gmw/4.003d0,39.944d0,83.80d0,131.30d0,2.016d0,28.8d0,18.016d0 &
     &/
      DATA cc1/3.366d-03,3.421d-04,4.0288d-05,4.726d-05,1.6355d-03,     &
     & 2.091d-04/
      DATA cc2/0.668d0,0.701d0,0.872d0,0.923d0,0.8213d0,0.846d0/
      DATA cc3/-2.8516d-8,9.424d-10,-6.004d-14/
      DATA cc4/8.4063d-3,1.19998d-5,6.706d-8,4.51d-11/
      DATA cc5,cc6/0.2103d0,1.0d-09/
      DATA cc7,cc8,cc9/2.8284d0,2.41d0,0.142d0/
      DATA rough/4.389d-06/
      poly2(a1,a2,a3,x)=a1+x*(a2*x*a3)
      poly3(a1,a2,a3,a4,x)=a1+x*(a2+x*(a3+x*a4))
!
!     constants if nonideal helium properties selected
      IF (igas.EQ.4) THEN
        cc1(1)=pkhe1
        cc2(1)=pkhe2
```

```
      ENDIF
!
!     calculate constituent conductivities and light gas correction
!     factors
!
      factor=0.0d0
      rc=0.0d0
      rmw=0.0d0
      phi=0.0d0
      psi=0.0d0
      altg=log(tgap)
!     knudsen domain correction factor for light gases
      IF (drgap.LE.rough) factor=cc5*sqrt(tgap)/(pgap*rough)
      DO i=1,6
       IF (gmix(i).GE.cc6) THEN
        c(i)=cc1(i)*exp(cc2(i)*altg)
        IF (i.EQ.1.OR.i.EQ.5) c(i)=c(i)/(1.0d0+factor*c(i))
       ENDIF
      ENDDO
!
      IF (gmix(7).GE.cc6) c(7)=poly2(cc3(1),cc3(2),cc3(3),tgap)*(pgap    &
     &/tgap)+1.009d0*((pgap/tgap))**2*exp(-4.2d0*log(tgap-273.d0))      &
     &+poly3(cc4(1),cc4(2),cc4(3),cc4(4),tgap)
!
!     calculate mixture thermal conductivity
!
      cgap=0.0d0
      DO i=1,7
       IF (gmix(i).GE.cc6) THEN
        sum=0.0d0
        DO j=1,7
         IF (j.NE.i) THEN
          IF (gmix(j).GE.cc6) THEN
           rc=c(i)/c(j)
           rmw=gmw(i)/gmw(j)
           phi=(1.d0+sqrt(rc*sqrt(rmw)))**2/(cc7*sqrt(1.0d0+rmw))
           psi=1.0d0+cc8*(rmw-1.0d0)*(rmw-cc9)/(gmw(j)*(1.0d0+rmw)* &
     &      *2)
           sum=sum+phi*psi*gmix(j)
          ENDIF
         ENDIF
        ENDDO
        cgap=cgap+c(i)*gmix(i)/(gmix(i)+sum)
       ENDIF
      ENDDO
      RETURN
      END SUBROUTINE mgap
!
      SUBROUTINE mhtr(cw,cpw,emis,row,t,tmhtr)
      IMPLICIT NONE
!
!     Declaration Generated by genImpDecs.pl 5/98
      REAL(sdk) cpw,cw,emis,expnt,qanty,row,star,t,tf,tmhtr
!
!
!     mhtr calculates values for density,specific heat,
```

```
!     thermal conductivity, and emissivity for electrically
!     heated fuel rod heater coil properties
!
!     cpw    - specific heat, j/kg k
!     cw     - thermal conductivity, w/m k
!     emis   - emissivity
!     row    - density, kg/m**3
!     t      - temperature, k
!     tf     - temperature, f
!     tmhtr  - melting temperature of heater coil, k
!
!
!     constantan heater coil
!
!     ref--  w.l. kirchner,"reflood heat transfer in a light
!            water reactor,"u.s. nuclear regulatory commission
!            report nureg-0106 (1976).
!
!!    STATEMENT FUNCTION
      star(qanty,expnt)=exp(expnt*log(qanty))
!
      tmhtr=1500.0d0
!
      tf=1.8d0*(t-273.15d0)+32.0d0
!
      row=8393.4d0
      cpw=110.0d0*star(tf,0.2075d0)
      cw=29.18d0+2.683d-03*(tf-100.0d0)
      emis=1.0d0
!
!
      RETURN
      END SUBROUTINE mhtr
!
      SUBROUTINE mwrx(drz,drzn,dt,ncrzp1,nodes,ncraz,qwrx,matrd,radrn,   &
     & rdtn,rnd,id,z,rnrdx,amh2,jflg)
      IMPLICIT NONE
!
!     Declaration Generated by genImpDecs.pl 5/98
      INTEGER(sik) j,jc,jflg,ncraz,ncrzp1,nf1,nodes
!
!     Declaration Generated by genImpDecs.pl 5/98
      REAL(sdk) amh2,amp,c,cwr1,cwr2,cwr3,dt,dzm,dzp,h4rzr,rmwrx,rzr
!
!
!     mwrx calculates the zircaloy steam reaction in the
!     cladding at high temperatures
!     the energy is assumed to be distributed uniformly in
!     the cladding
!
      REAL(sdk) matrd(:),id(:),rnd(:,:)
      REAL(sdk) rnrdx
      REAL(sdk) drz(:),drzn(:),qwrx(:),radrn(:,:),rdtn(:,:),z(:)
!
!        dimensions:
!     dz(ncrz)
```

```
!   drz(ncrzp1),drzn(ncrzp1),qwrx(ncrzp1),radrn(nodes,ncraz),
!   rdtn(nodes,ncraz),rnd(nodes,ncrzp1),id(ncraz),matrd(nodes-1)
!
      INCLUDE 'constant.h'
!
! reaction model based on eq. (b-14.11) and eq. (b-14.1) matpro-11
! cwr1=2.*a/(1.5*0.9*.26)**2
! a from table b-14.i  [weight gain: a = 16.80 (kg/m2)**2]
! 1.5 is volume expansion zr to zro2 (radial only for 1-d model)
! 0.9 is density zro2/density zr assuming 50 per cent volume
!  expansion
! 0.26 is weight fraction o in zro2
! cwr2 = b from table b-14.i divided by the rate constant r
! cwr3 is heat of reaction (j/kg) per kg of zr  eq. (b-14.10)
!for steam      DATA cwr1,cwr2,cwr3/2.727d02,-2.007d04,6.45d06/
!air constants
      DATA cwr1,cwr2,cwr3/100.65,-1.4634d04,12.052d06/
      DATA h4rzr/0.0d0/
!
!    determine the clad inner-radius node nf1
!
      IF (nodes.GE.2) THEN
       nf1=nodes-1
       DO WHILE (nf1.NE.1)
        IF (int(matrd(nf1)).NE.int(matrd(nf1-1))) GOTO 110
        nf1=nf1-1
       ENDDO
  110  CONTINUE
!
       j=0
       jc=0
       DO WHILE (.TRUE.)
        jc=jc+1
        IF (int(id(jc)).GE.1000) THEN
         j=j+1
         IF (rdtn(nodes,jc).GE.273.15d0) THEN
          rzr=radrn(nodes,j)-drz(j)
          IF (rzr.GE.radrn(nf1,j)) THEN
           c=cwr1/rnd(nodes,j)**2
           drzn(j)=sqrt(drz(j)*drz(j)+c*dt*exp(cwr2/rdtn(nodes,jc)  &
     &       ))
           rmwrx=rnd(nodes,j)*(rzr**2-(radrn(nodes,j)-drzn(j))**2)
           qwrx(j)=rmwrx*cwr3/(dt*(radrn(nodes,j)**2-radrn(nf1,j)*  &
     &       *2))
!
!    determine hydrogen generated by the average rod's cladding
!
           IF (jflg.GE.1) THEN
            IF (jc.EQ.1) THEN
             dzm=0.0d0
            ELSE
            dzm=z(j)-z(j-1)
            ENDIF
            IF (jc.EQ.ncraz) THEN
             dzp=0.d0
            ELSE
```

```
            dzp=z(j+1)-z(j)
            ENDIF
            amp=rnrdx*h4rzr*pi*rmwrx
            amh2=amh2+0.5d0*(dzm+dzp)*amp
           ENDIF
          ENDIF
         ENDIF
         IF (jc.GE.ncraz) GOTO 140
        ENDIF
       ENDDO
      ENDIF
  140 CONTINUE
!
      RETURN
      END SUBROUTINE mwrx
!
      SUBROUTINE MWRxN(dt,ncr)

      USE HSArray
      USE Global
      USE RodVlt
      USE Flt

      IMPLICIT NONE
!
      INTEGER(sik), INTENT(in) :: ncr
      INTEGER(sik) j,jf,ncraz,nf1,nodes
!
      REAL(sdk), INTENT(in) :: dt
      REAL(sdk) amp,c,cwr1,cwr2,cwr3,h4rzr,rmwrx,rzr
!
!     mwrxN calculates the zircaloy-steam reaction in the cladding at
!     high temperatures for the HTSTR component
!
      REAL(sdk), POINTER, DIMENSION(:) :: dz
      REAL(sdk), POINTER, DIMENSION(:,:) :: rho
!
      INCLUDE 'constant.h'
!
      DATA cwr1,cwr2,cwr3/2.727d02,-2.007d04,6.45d06/
      DATA h4rzr/.0442d0/

      nodes=genTab(cco)%nodes
      ncraz=hsAr(cco)%noht(ncr)
      IF (.NOT.rodTab(cco)%fmOn) THEN
       dz=>hsAr(cco)%dhtstrz(1:ncraz)
      ELSE
       dz=>hsAr(cco)%dhtstrzf(1:ncraz,ncr)
      ENDIF
      rho=>hsAr(cco)%rnd(1:nodes,1:ncraz,ncr)

!     determine the clad inner-radius node nf1
!
      IF (nodes.GE.2) THEN
       nf1=nodes-1
       DO WHILE (nf1.NE.1)
```

```
        IF (int(hsAr(cco)%matrd(nf1)).NE.                    &
     &      int(hsAr(cco)%matrd(nf1-1))) GOTO 110
        nf1=nf1-1
       ENDDO
  110  CONTINUE
!
       DO jf=1,ncraz
        j=hsAr(cco)%ncoarse(jf,ncr)
        IF (hsAr(cco)%rft(nodes,jf,ncr).GE.1273.15d0) THEN
         rzr=hsAr(cco)%radrn(nodes,j,ncr)-hsAr(cco)%drz(jf,ncr)
         IF (rzr.GE.hsAr(cco)%radrn(nf1,j,ncr)) THEN
          c=cwr1/rho(nodes,jf)**2
          hsAr(cco)%drzn(jf,ncr)=sqrt(hsAr(cco)%drz(jf,ncr)**2+c*dt* &
     &                exp(cwr2/hsAr(cco)%rft(nodes,jf,ncr)))
          rmwrx=rho(nodes,jf)*                               &
     &          (rzr**2-(hsAr(cco)%radrn(nodes,j,ncr)-       &
     &                   hsAr(cco)%drzn(jf,ncr))**2)
          hsAr(cco)%qwrx(jf,ncr)=rmwrx*cwr3/                 &
     &      (dt*(hsAr(cco)%radrn(nodes,j,ncr)**2-            &
     &                hsAr(cco)%radrn(nf1,j,ncr)**2))
!
!         determine hydrogen generated by the average rod's cladding
!
          IF (ncr.EQ.1) THEN
!           average rod only
            amp=hsAr(cco)%nrdx(1)*h4rzr*pi*rmwrx
            rodTab(cco)%amh2=rodTab(cco)%amh2+0.5d0*dz(jf)*amp
          ENDIF
         ENDIF
        ENDIF
        IF (jf.GE.ncraz) GOTO 140
       ENDDO
      ENDIF
  140 CONTINUE
!
      RETURN
      END SUBROUTINE MWRxN
!
!
      SUBROUTINE mzirc(cw,cpw,emis,nclox,row,t,tmzirc)
!     BEGIN MODULE USE
      USE Util
      IMPLICIT NONE
!
!     Declaration Generated by genImpDecs.pl 5/98
      INTEGER(sik) id2,k1,nclox
!
!     Declaration Generated by genImpDecs.pl 5/98
      REAL(sdk) a1,a2,a3,a4,cpw,cw,dvol,emis,poly3,row,t,tc,tfzirc,     &
     & thexpr,thexpz,tmzirc,x
!
      REAL(sdk) xd(1)
!
!     mzirc calculates values for density specific heat,thermal
!     conductivity, emissivity,elastic modulus,axial thermal
!     expansion, radial thermal expansion and melting temperature
```

```
!     for zircaloy-4.  zircaloy-2 properties are assumed
!     equivalent
!
!     ref--  p.e. macdonald, et al.,"matpro - version 09,"tree-
!          nureg-1005,idaho national engineering laboratory (1976).
!
!     cpw    - specific heat, j/kg k
!     cw     - thermal conductivity, w/m k
!     emis   - emissivity
!     nclox  - zircaloy type,  nclox=0 - zircaloy
!                              nclox=1 - zro2
!     row    - density, kg/m**3
!     t      - temperature, degrees kelvin
!     tc     - temperature, degrees centigrade
!     thexpr - thermal expansion, radial
!     thexpz - thermal expansion, axial
!     tmzirc - melting point temperature of zr-4, k
!     ttzirc - transition point temperature, alpha-beta phases, k
!
      REAL(sdk) cpzirc(26),cwzirc(4),cwzro2(4)
!
!
      DATA tfzirc/2098.15d0/
      DATA cpzirc/300.0d0,281.0d0,400.0d0,302.0d0,640.0d0,331.0d0,1090   &
     &.0d0,375.0d0,1093.0d0,502.0d0,1113.0d0,590.0d0,1133.0d0,615.0d0    &
     &,1153.0d0,719.0d0,1173.0d0,816.0d0,1193.0d0,770.0d0,1213.0d0       &
     &,619.0d0,1233.0d0,469.0d0,1248.0d0,356.0d0/
      DATA cwzirc/7.51d0,2.09d-02,-1.45d-05,7.67d-09/
      DATA cwzro2/1.96d0,-2.41d-04,6.43d-07,-1.95d-10/
      poly3(a1,a2,a3,a4,x)=a1+x*(a2+x*(a3+x*a4))
!
      tc=t-273.15d0
      tmzirc=tfzirc
!
!     thermal expansion of zircaloy
!
!     alpha phase thermal expansion
!
      IF (t.LE.1073.15d0) THEN
        thexpr=-2.373d-04+6.721d-06*tc
        thexpz=-2.506d-05+4.441d-06*tc
!
!     transition regime thermal expansion
!
      ELSEIF (t.GT.1273.15d0) THEN
!
!     beta phase thermal expansion
!
        thexpr=-6.8d-03+9.7d-06*tc
        thexpz=-8.3d-03+9.7d-06*tc
      ELSE
        thexpr=5.1395d-03-1.12d-05*(t-1073.15d0)
        thexpz=3.5277d-03-1.06385d-05*(t-1073.15d0)
      ENDIF
!
!     density of zircaloy corrected for thermal expansion
```

```
!
      dvol=(2.0d0*thexpr+thexpz)
      row=6551.4d0/(1.0d0+dvol)
!
!     emissivity of zircaloy and zro2
!     an average value is used as emissivity of zro2 is also
!     time dependent
      emis=0.75d0
!
!     specific heat of zircaloy
!
      IF (t.GT.1248.0d0) THEN
       cpw=356.0d0
      ELSE
       k1=1
       xd(1)=cpw
       CALL linint0(cpzirc,t,1,13,xd,k1,id2)
       cpw=xd(1)
      ENDIF
!
!     thermal conductivity
!
      IF (nclox.EQ.1) THEN
!
!     zro2 thermal conductivity
       cw=poly3(cwzro2(1),cwzro2(2),cwzro2(3),cwzro2(4),t)
      ELSE
!
!     zircaloy thermal conductivity
       cw=poly3(cwzirc(1),cwzirc(2),cwzirc(3),cwzirc(4),t)
      ENDIF
!
      RETURN
      END SUBROUTINE mzirc

      END MODULE Materials
```