INTERNAL REPORT SDL-89-05

LISTINGS OF FORTRAN PROGRAMS: SURFGEN, SSHIST, UNLOAD, REVCNC

J.B. AIDUN

December 1989

Shock Dynamics Laboratory

Department of Physics

Washington State University

Pullman, WA 99164

SDL Internal Report 89-5 - John B. Aidun

LISTINGS OF FORTRAN PROGRAMS: SURFGEN, SSHIST, UNLOAD, REVCNC

This report contains the listings and sample input data for four fortran computer programs developed by J. B. Aidun: "Study of Shear and Compression Waves in Shocked Calcium Carbonate," Ph.D. Thesis, Department of Physics, Washington State University, Pullman, WA 99164-2814 (1989).

The program SURFGEN generates the particle velocity surface described in Appendix G of the thesis from the values of the various parameters entered as input.

SSHIST integrates this particle velocity surface along particle paths to calculate the density, and along isochrons to calculate the longitudinal stress. The input data for surfgen and sshist are identical. The role of each of the parameters is described in Appendix G. The values used for each of the reported fits are also given in that appendix.

The program UNLOAD calculates the phase velocity, C_{u1}, during unloading by linearly interpolating between the points of the digitized particle velocity histories. Two density histories are then calculated from

$$\mu(\tilde{h}, t) = - \int_{u_1(\tilde{h}, t_o)}^{u_1(\tilde{h}, t)} \frac{du'}{C_u(\tilde{h}, u')} \tag{1}$$

using C_{u1} values determined from the first two histories, and from the second and third histories, respectively. In this equation, h is the Lagrangian depth, t is time, u is longitudinal particle velocity, and μ is a volumetric strain, defined as

$$\mu = 1 - V/V_o = 1 - \rho_o/\rho \, , \tag{2}$$

where V is the specific volume and V_o is its zero-pressure value.

Two longitudinal stress histories are calculated from

$$\sigma(\tilde{h}, t) - \sigma(\tilde{h}, t_o) = \rho_o \int_{u(\tilde{h}, t_o)}^{u(\tilde{h}, t)} C_\sigma(\tilde{h}, u') \, du' \qquad (3)$$

using these same two sets of C_{u1} values and assuming that they equal the phase velocity, $C_{\sigma 1}$. When this assumption is correct, the stress-density paths calculated from different pairs of particle velocity histories are identical.

The program REVCNC is the stress concentration model for the mean stress response of calcite rocks. This model is described in Section 6.3.1 and in Appendix H of the thesis.

The UNIX utility, AWK, was used to generate plot files from the data files output by these programs. The Awk commands are not included in this report.

```
      program surfgen

c     Generate a surface from the input parameters
c     for comparison with measured particle velocity histories.
c
c     John Aidun, WSU SDL, Pullman WA, 8/89

      parameter (mm=500, mgage=30, npts=306)
      implicit real*8 (a-h,o-z)
      real*8 lam, nexp
      character*30 title

      dimension t(mgage,mm), u(mgage,mm), uk(mgage), tinf(mgage),
     1 tbrk(mgage), tpk(mgage), trl(mgage), upk(mgage), h(mgage),
     2 ubrk(mgage), tk(mgage), uinf(mgage), tm(mgage)

      ngage = 3
      read(5,900) title
      read (5,*) (h(i), i=1,ngage)
      read (5,*) d, aych       ! loading wave speed, H
      read (5,*) uko, lam      ! intial knee amplitude, lambda
      read (5,*) um            ! amplitude at foot of second shock
      read (5,*) tmo, cm       ! offset for TOA of second shock and
c                        ! wave speed at foot of second shock
      read (5,*) alpha1, gamma1
      read (5,*) eps, epso     ! epsilon at deepest gauge, and front
      read (5,*) tdel          ! tdelta at deepest gauge
      read (5,*) ulin          ! p.v. at inflection at deepest gauge
      read (5,*) cd            ! 1st dispersive wave shape
      read (5,*) alpha2, gamma2
      read (5,*) beta, eta
      read (5,*) ub, ten1      ! p.v. at break in slope at 1st
c                        ! gauge & decrease to last gauge
      read (5,*) nexp          ! exponent on 2nd dispersive wave
      read (5,*) up, ten2      ! p.v. at foot of final slope 1st
c                        ! gauge & decrease to last gauge
      read (5,*) tp, cpk       ! time of peak at first gauge,
c                        ! and wave speed
      read (5,*) tauo, vp      ! front surface pulse width,
c                        ! and release speed
      read (5,*) urel          ! peak particle velocity

      ukfac = uko/(1.d+00 + aych)
      h3 = h(ngage)
      h(5) = h(ngage)
      h(3) = h(2)
      h(2) = (h(1) + h(3))*0.5d+00
      h(4) = (h(3) + h(5))*0.5d+00
      hmax = d*(tauo + h3/vp)
      dh = 1.d-01*(hmax - h3)           ! 10 increments

      do 110 k = 1, 10
      j = k + 5
      j1 = j - 1
      h(j) = h(j1) + dh
  110 continue

      ngage = ngage + 12
      attn1 = ten1 / ub / (h3 - h(1))     ! omega in notes
      attn2 = ten2 / up / (h3 - h(1))     ! omega2 in notes
      carg1 = -cd/cm
      arg = -h3/lam
c     uk3 = 0.5d+00*uko*(1.d+00 + dexp(arg))
      uk3 = ukfac*(1.d+00 + aych*dexp(arg))
      peps = epso*eps/(epso - eps)

      do 510 j = 1, ngage
      to = h(j)/d
      arg = -h(j)/lam

c epsilon varies with depth unless epso = 0.
      if (eps .eq. 0.d+00) then
            elon = eps
      else
            elon = 1.d+00/epso + h(j)/h3/peps
            elon = 1.d+00/elon
      endif
c     uk(j) = 0.5d+00*uko*(1.d+00 + dexp(arg))
      uk(j) = ukfac*(1.d+00 + aych*dexp(arg))
      tknee = alpha1*dlog((1.d+00 + gamma1*(1.d+00 - elon))/elon)
      tk(j) = tknee + to
      uinf(j) = uk(j) + (ulin - uk3)*h(j)/h3
      tdelta = tdel*h(j)/h3            !grows linearly with depth
      tdsp = tk(j) + tdelta
      carg2 = cd*(tdsp - tmo)/h(j)
      ampa = (um - uk(j))/(1.d+00 - dexp(carg1 + carg2))
      if (uinf(j) .le. uk(j)) then
            tinf(j) = tk(j)
      else
            tinf(j) = h(j)*dlog((ampa + uk(j) - uinf(j))/ampa)/cd
            tinf(j) = tdsp - tinf(j)
      endif
      tm(j) = tmo + h(j)/cm
      if (tm(j) .le. tk(j)) then
            tm(j) = tk(j)
            um = uk(j)
      endif
      ubrk(j) = ub*(1.d+00 - attn1 * (h(j) - h(1)))
      ampb = (ubrk(j) - um)/(1.d+00 - eta)
      temp = alpha2*dlog((1.d+00 + gamma2*h(j)*(1.d+00 - eta))/eta)
      if (tm(j) .le. tk(j)) then
            tbrk(j) = tk(j) + (1.d+00 + beta*h(j))*temp
      else
            tbrk(j) = tm(j) + (1.d+00 + beta*h(j))*temp
      endif
      if (alpha2 .eq. 0.0d+00) then
            tbrk(j) = tm(j)
            ubrk(j) = um
      endif
      upk(j) = up*(1.d+00 - attn2*(h(j) - h(1)))
      tpk(j) = tp + (h(j) - h(1))/cpk
      if (tpk(j) .le. tbrk(j)) tpk(j) = tbrk(j)
      trat = (tbrk(j)/tpk(j))**nexp
      deltat = (tpk(j) - to)/(dble(npts) - 6.d+00)
      trl(j) = tauo + h(j)/vp
      icnt = 0

      do 500 i = 1, npts
      tincr = (dble(i)-1.d+00)*deltat
      t(j,i) = to + tincr

      if (t(j,i) .le. tk(j)) then
            etoal = dexp(tincr/alpha1)
            u(j,i) = uk(j)*(etoal - 1.d+00)/(etoal + gamma1)
            u(j,i) = u(j,i)/(1.d+00 - elon)

      elseif (t(j,i) .le. tinf(j)) then
            utemp = (uinf(j) - uk(j))*(t(j,i) - tk(j))
            u(j,i) = uk(j) + utemp/(tinf(j) - tk(j))

      elseif (t(j,i) .le. tm(j)) then
            darg = -cd*(t(j,i)-tdsp)/h(j)
            u(j,i) = uk(j) + ampa*(1.d+00 - dexp(darg))

      elseif (t(j,i) .le. tbrk(j)) then
            earg = alpha2*(1.d+00 + beta*h(j))
            fexp = dexp((t(j,i)-tm(j))/earg)
            u(j,i) = um + ampb*(fexp -1.d+00)/(fexp + gamma2*h(j))

      elseif (t(j,i) .le. tpk(j)) then
            crat = (tbrk(j)/t(j,i))**nexp
            u(j,i) = (upk(j) - ubrk(j))*(1.d+00 - crat)
            u(j,i) = u(j,i)/(1.d+00 - trat)  + ubrk(j)

      else
            icnt = icnt + 1
            t(j,i) = tpk(j) +  icnt*(trl(j) - tpk(j))/5.d+00
            utemp = (urel - upk(j))*(t(j,i) - tpk(j))
            u(j,i) = upk(j) + utemp/(trl(j) - tpk(j))
      endif

      if (t(j,i) .gt. trl(j)) then
            t(j,i) = t(j,i-1)
            u(j,i) = u(j,i-1)
      endif
  500 continue
  510 continue

c     OUTPUT >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>

      write(6,901) title, d, uko, lam, aych, eps, epso, beta,
     1          alpha1, gamma1, eta, alpha2, gamma2, tdel,
  2          um, cm, tmo, ulin, cd, ten1, ub, nexp, ten2,
  3          up, cpk, tp, urel, vp, tauo

      write(6,903) h(1),h(3),h(5),uk(1),uk(3),uk(5),
  1   tk(1),tk(3),tk(5),uinf(1),uinf(3),uinf(5),
  2   tinf(1),tinf(3),tinf(5),ubrk(1),ubrk(3),ubrk(5),
  3   tm(1), tm(3), tm(5), tbrk(1),tbrk(3),tbrk(5),
  4   upk(1), upk(3),upk(5), tpk(1),tpk(3),tpk(5),
  5   trl(1),trl(3),trl(5)

      write(6,905) (h(j), j=1,ngage)
      write(6,909) ((t(j,k), u(j,k), j = 1,ngage), k = 1,npts)

900   format(a30/)
901   format(/10x,'Surface generating parameters -'a30,///
  1   5x,'D       =',f5.2,5x,'uko     =',f7.4,3x,'lambda =',f6.3,
  1   3x,'H  =',f6.3/
  2   5x,'epsilon =',f8.6,2x,'epso    =',f8.6,2x,'beta    =',f5.3/
  3   5x,'alpha1  =',f6.4,4x,'gamma1 =',f6.2,4x,'eta     =',f8.6/
  4   5x,'alpha2  =',f6.4,4x,'gamma2 =',f5.2,5x,'tdelay =',f5.3/
  5   5x,'Um      =',f7.5,3x,'Cm      =',f6.3,4x,'tmo     =',f6.3/
  6   5x,'UL      =',f7.5,3x,'Cdsp1  =',f5.2,5x,'attn1  =',f6.4/
  7   5x,'Ub      =',f7.5,3x,'nexp   =',f6.3,4x,'attn2  =',f6.4/
  8   5x,'Up      =',f7.5,3x,'Cpk    =',f6.3,4x,'tp      =',f5.3/
  9   5x,'Ur      =',f7.5,3x,'Vp     =',f5.2,5x,'tau0   =',f6.3/)
903   format(5x,'  h    =',3f10.3//
  1        5x,'Uknee(h)=',3f10.4/5x,'tk(h)   =',3f10.4/
  1        5x,'Uinf(h) =',3f10.4/5x,'tinf(h) =',3f10.4/
  1        5x,'Ubrk(h) =',3f10.4/5x,'tm(h)   =',3f10.4/
  2        5x,'tbrk(h) =',3f10.4/5x,'Upk(h)  =',3f10.4/
  2        5x,'tpk(h)  =',3f10.4/5x,'trl(h)  =',3f10.4/)
905   format(' h =',3x,15(f7.3,9x)/x,15(4x,'t',7x,'u',x,'| ')/)
909   format(15(f7.3,f8.4,x))

1000       stop
      end
      program sshist

c     Calculate stress and strain histories by integrating partial
c     derivatives determined from a piecewise continuous surface
c     that was fit to longitudinal particle velocity histories.
c
c     John Aidun, WSU SDL, Pullman, WA 9/89
c
c     mu = 1 - V/Vo

      parameter (mgage=5, npts=3000)
      implicit real*8 (a-h,o-z)
      real*8 lam, nexp, mu(mgage,npts), mucu(mgage,npts)
      character*30 title

      dimension nt(mgage), nh(mgage,npts), delt(mgage), mrem(mgage),
     2 delh(mgage,npts), sigma(mgage,npts), uc(mgage,npts), mt(mgage),
     3 t(mgage,npts), tc(mgage,npts), puph(mgage,npts), u(mgage,npts),
     4 cu(mgage,npts), sigcu(mgage,npts)

      common /face/ carg1, tincr, hincr, h(mgage), h3, uk3, d,
     2 ukfac, aych, lam, um, tmo, cm, alpha1, gamma1, tdel, ulin,
     3 cd, alpha2, gamma2, beta, eta, ub, attn1, attn2,
     4 nexp, up, tp, cpk, tauo, vp, urel, peps, eps, epso

      ngage = 4
      read (5,900) title
      write(6,901) title
c density, time step, # of time steps between output times, depth step
      read (5,*) denso, tincr, kedit, hincr
      read (5,*) (h(i), i=1,ngage)
      read (5,*) d, aych      ! loading wave speed, H
      read (5,*) uko, lam     ! knee amplitude at impact face, lambda
      read (5,*) um           ! amplitude at foot of second shock
      read (5,*) tmo, cm      ! offset for TOA of second shock and
c                        ! wave speed at foot of second shock
      read (5,*) alpha1, gamma1
      read (5,*) eps, epso    ! epsilon at deepest gauge and at front
      read (5,*) tdel         ! tdelta at deepest gauge
      read (5,*) ulin         ! p.v. at inflection at deepest gauge
      read (5,*) cd           ! 1st dispersive wave shape parameter
      read (5,*) alpha2, gamma2
      read (5,*) beta, eta
      read (5,*) ub, ten1     ! p.v. at break in slope at 1st gauge and
                         ! its decrease between 1st and last gauges
      read (5,*) nexp         ! exponent on 2nd dispersive wave
      read (5,*) up, ten2     ! p.v. at foot of final slope of 1st gauge
                         ! its decrease between 1st and last gauges
      read (5,*) tp, cpk      ! time of peak at first gauge, wave speed
      read (5,*) tauo, vp     ! front surface pulse width, release speed
      read (5,*) urel         ! peak particle velocity

      h3 = h(3)
      ukfac = uko/(1.d+00 + aych)
      if (h(4) .eq. h(3)) ngage = 3

      write(6,903) denso, h(1), h(2), h(3), h(4), tincr, hincr, kedit
      write(6,905) d, uko, lam, aych, eps, epso, beta,
     1            alpha1, gamma1, eta, alpha2, gamma2, tdel,
     2            um, cm, tmo, ulin, cd, ten1, ub, nexp, ten2,
     3            up, cpk, tp, urel, vp, tauo

c  initialization

      tnear = 1.0d-03*tincr
     do 30 j = 1, ngage
      t(j,1) = h(j)/d
      tc(j,1) = t(j,1)
      trlt = trl(h(j))
      nt(j) = (trlt - t(j,1))/tincr
      if (nt(j) .ge. npts-4) then
        write(6,*) ' tincr too small for current value of npts.'
            goto 1000
      endif
      mt(j) = nt(j)/kedit
      mrem(j) = nt(j) - kedit*mt(j)
      mt(j) = mt(j) + mrem(j) + 3
      nh(j,1) = 0
      delh(j,1) = 0

     do 20 i = 1, nt(j)
      ip = i + 1
      t(j,ip) = t(j,i) + tincr
      tc(j,ip) = 0.5d+00*(t(j,ip) + t(j,i))
      htemp = (d*tc(j,ip) - h(j))
      nh(j,ip) = htemp/hincr
      delh(j,ip) = htemp - nh(j,ip)*hincr
 20   continue

      nt1 = nt(j) + 1
      nt2 = nt1 + 1
      nt3 = nt2 + 1
      t(j,nt2) = trlt
      delt(j) = 0.5d+00*( t(j,nt2) - t(j,nt1) )
      tc(j,nt2) = 0.5d+00*( t(j,nt2) + t(j,nt1) )
      tc(j,nt3) = trlt
      htemp = (d*tc(j,nt2) - h(j))
      nh(j,nt2) = htemp/hincr
      delh(j,nt2) = htemp - nh(j,nt2)*hincr
      htemp = (d*tc(j,nt3) - h(j))
      nh(j,nt3) = htemp/hincr
      delh(j,nt3) = htemp - nh(j,nt3)*hincr
 30   continue

      attn1 = ten1 / ub / (h3 - h(1))
      attn2 = ten2 / up / (h3 - h(1))
      carg1 = -cd/cm
      arg = -h3/lam
```

```
       uk3 = ukfac*(1.d+00 + aych*dexp(arg))
       peps = epso*eps/(epso - eps)

      do 210 j = 1, ngage
       to = h(j)/d
       arg = -h(j)/lam
       elont = elon(h(j))
       ukt   = uk(h(j))
       tkt   = tk(h(j))
       tinft = tinf(h(j))
       tdspt = tdsp(h(j))
       tmt   = tm(h(j))
       ubrkt = ubrk(h(j))
       tbrkt = tbrk(h(j))
       upkt  = upk(h(j))
       tpkt  = tpk(h(j))
       trlt  = trl(h(j))

       carg2 = cd*(tdspt - tmo)/h(j)
       aa = (um - ukt)/(1.d+00 - dexp(carg1 + carg2))
       ampb = (ubrkt - um)/(1.d+00 - eta)
       trat = (tbrkt/tpkt)**nexp

c   calculate particle velocity histories at centered times

       do 200 i = 1, nt(j)+3

        if (tc(j,i) .le. tkt) then
              tstep = tc(j,i) - to
              etoal = dexp(tstep/alpha1)
              uc(j,i) = ukt*(etoal - 1.d+00)/(etoal + gamma1)
              uc(j,i) = uc(j,i)/(1.d+00 - elont)

        elseif (tc(j,i) .le. tinft) then
              utemp = (uinf(h(j)) - ukt)*(tc(j,i) - tkt)
              uc(j,i) = ukt + utemp/(tinft - tkt)

        elseif (tc(j,i) .le. tmt) then
              darg = -cd*(tc(j,i)-tdspt)/h(j)
              uc(j,i) = ukt + aa*(1.d+00 - dexp(darg))

        elseif (tc(j,i) .le. tbrkt) then
              earg = alpha2*(1.d+00 + beta*h(j))
              fexp = dexp((tc(j,i)-tmt)/earg)
              uc(j,i) = um + ampb*(fexp -1.d+00)/(fexp + gamma2*h(j))

        elseif (tc(j,i) .le. tpkt) then
              crat = (tbrkt/tc(j,i))**nexp
              uc(j,i) = (upkt - ubrkt)*(1.d+00 - crat)
              uc(j,i) = uc(j,i)/(1.d+00 - trat)  + ubrkt

        else
              utemp = (urel - upkt)*(tc(j,i) - tpkt)
              uc(j,i) = upkt + utemp/(trlt - tpkt)
      endif

 200  continue

c   calculate phase velocity and particle velocity histories at border times

      do 205 i = 1, nt(j)+2

      if (t(j,i) .le. tkt) then
            tstep = t(j,i) - to
            etoal = dexp(tstep/alpha1)
            u(j,i) = ukt*(etoal - 1.d+00)/(etoal + gamma1)
            u(j,i) = u(j,i)/(1.d+00 - elont)
            dudt = d1udt(h(j),t(j,i))

      elseif (t(j,i) .le. tinft) then
            utemp = (uinf(h(j)) - ukt)*(t(j,i) - tkt)
            u(j,i) = ukt + utemp/(tinft - tkt)
            dudt = d2udt(h(j),t(j,i))

      elseif (t(j,i) .le. tmt) then
            darg = -cd*(t(j,i)-tdspt)/h(j)
            u(j,i) = ukt + aa*(1.d+00 - dexp(darg))
            dudt = d3udt(h(j),t(j,i))

      elseif (t(j,i) .le. tbrkt) then
            earg = alpha2*(1.d+00 + beta*h(j))
            fexp = dexp((t(j,i) - tmt)/earg)
            u(j,i) = um + ampb*(fexp -1.d+00)/(fexp + gamma2*h(j))
            dudt = d4udt(h(j),t(j,i))

      elseif (t(j,i) .le. tpkt) then
            crat = (tbrkt/t(j,i))**nexp
            u(j,i) = (upkt - ubrkt)*(1.d+00 - crat)
            u(j,i) = u(j,i)/(1.d+00 - trat)  + ubrkt
            dudt = d5udt(h(j),t(j,i))

      else
            utemp = (urel - upkt)*(t(j,i) - tpkt)
            u(j,i) = upkt + utemp/(trlt - tpkt)
            dudt = d6udt(h(j),t(j,i))
      endif

      if (t(j,i) .eq. to) then
      ttemp = t(j,i) + tnear
            ugrad =     dudh(h(j),ttemp)
      else
            ugrad = dudh(h(j),t(j,i))
      endif
      if (abs(dudt) .lt. 1.d-25 .or. abs(ugrad) .lt. 1.d-25) then
            cu(j,i) = 0.d+00
      else
            cu(j,i) = -dudt/ugrad
      endif
 205  continue
 210  continue

c   Grand loop for calculating stress and strain

      ierr = 0

      do 800 j = 1, ngage
c debugging code
c     write(6,*) 'nh(j,nt2) =', nh(j,nt2)
c     write(6,*)
c    1 '    sum        stem          tval         jj   m  flag'
      iflag = 1
      nflag = 1
      kcnt = 1
      mu(j,1) = 0.d+00
      mucu(j,1) = 0.d+00
      sigcu(j,1) = 0.d+00

      tkt   = tk(h(j))
      tinft = tinf(h(j))
      tmt   = tm(h(j))
      tbrkt = tbrk(h(j))
      tpkt  = tpk(h(j))

c   Calculate longitudinal stress

      sigma(j,1) = 0.d+00
      nt2 = nt(j) + 2
      nt3 = nt2 + 1

      ip = nt2
      call ssumh(sum,j,tc(j,ip),nh(j,ip),ierr,m)
      if (ierr .gt. 0) goto 888
      mm = nh(j,nt2) + 1
      hvm = h(j) + (dble(mm) - 1.d+00)*hincr
      hvp = d*(tc(j,nt2) - tnear)
      stem = d1udt(hvm,tc(j,nt2)) + d1udt(hvp,tc(j,nt2))
      sigma(j,nt2) = (sum + 0.5d+00*delh(j,nt2)*stem)*denso
c  debugging code
c     write(6,*) 'sigma(1,nt2) =', sigma(1,nt2)
c     stop

      ip = nt3
      call ssumh(sum,j,tc(j,ip),nh(j,ip),ierr,m)
      if (ierr .gt. 0) goto 888
      mm = nh(j,nt3) + 1
      hvm = h(j) + (dble(mm) - 1.d+00)*hincr
      hvp = d*(tc(j,nt3) - tnear)
      stem = d1udt(hvm,tc(j,nt3)) + d1udt(hvp,tc(j,nt3))
      sigma(j,nt3) = (sum + 0.5d+00*delh(j,nt3)*stem)*denso

      do 700 i = 1, nt(j)
       icnt = 2
```

```
        ip = i + 1
        if (ip .gt. mrem(j) + 2) then
              knum = mrem(j) + 2 + kcnt*kedit
              if (ip .lt. knum) then
                    goto 705
              else
                    kcnt = kcnt + 1
              endif
        endif
        call ssumh(sum,j,tc(j,ip),nh(j,ip),ierr,m)
        if (ierr .gt. 0) goto 888
        mm = nh(j,ip) + 1
        hvm = h(j) + (dble(mm) - 1.d+00)*hincr
        hvp = d*(tc(j,ip) - tnear)
        stem = d1udt(hvm,tc(j,ip)) + d1udt(hvp,tc(j,ip))
        sigma(j,ip) = (sum + 0.5d+00*delh(j,ip)*stem)*denso

c   Calculate  strain, mu = 1 - V/Vo

 705  m = 0
      goto (710, 720, 730, 740, 750, 780), iflag

 710  if (tkt .lt. t(j,ip)) then
         if (tinft .ge. t(j,ip)) then
            iflag = 2
            icnt = 0
            dudt = d2udt(h(j),tkt)
         endif
         if(tkt .ge. tinft .or. iflag .eq. 2) then
            iflag = iflag + icnt
            if (icnt .eq. 2) dudt = d3udt(h(j),tkt)
            if (tkt .ge. tmt .and. tkt .lt. tbrkt) then
                   iflag = 4
                   dudt = d4udt(h(j),tkt)
            elseif (tkt .ge. tbrkt .and. tkt .lt. tpkt) then
                   iflag = 5
                   dudt = d5udt(h(j),tkt)
            elseif (tkt .ge. tpkt) then
                   iflag = 6
                   dudt = d6udt(h(j),tkt)
            endif
            tkp = tkt + tnear
      tempmu = (dudh(h(j),tkt) + dudh(h(j),t(j,i)))*(tkt - t(j,i))
      mu(j,ip) = (dudh(h(j),t(j,ip)) + dudh(h(j),tkp))*(t(j,ip) - tkt)
            mu(j,ip) = tempmu + mu(j,ip)
            cux = -d1udt(h(j),tkt)/dudh(h(j),tkt)
            ugrad = dudh(h(j),tkp)
            udiff = u(j,ip) - u(j,i)

          if (abs(dudt) .lt. 1.d-25 .or. abs(ugrad) .lt. 1.d-25
     1        .or. udiff .eq. 0.d+00) then
                  avcu = 0.d+00
                  avcup = 0.d+00
                  avcinv = 0.d+00
                  avcinvp = 0.d+00
              else
              cuy = -dudt/ugrad
              avcu = (uk(h(j)) - u(j,i))*(cux + cu(j,i))/udiff
              avcup = (u(j,ip) - uk(h(j)))*(cu(j,ip) + cuy)/udiff
              avcinv = (1.d+00/cux + 1.d+00/cu(j,i))/udiff
              avcinv = (uk(h(j)) - u(j,i))*avcinv
              avcinvp = (1.d+00/cu(j,ip) + 1.d+00/cuy)
              avcinvp = (u(j,ip) - uk(h(j)))*avcinvp/udiff
              endif
              goto 790
          else
              ierr = 5
              goto 888
          endif
      endif
720   if (tinft .lt. t(j,ip)) then
          if (tmt .ge. t(j,ip)) then
              iflag = 3
              icnt = 0
              dudt = d3udt(h(j),tinft)
          endif
          if (tinft .ge. tmt .or. iflag .eq. 3) then
              iflag = iflag + icnt
              if (icnt .eq. 2) dudt = d4udt(h(j),tinft)
              if (tinft .ge. tbrkt .and. tinft .lt. tpkt) then
                    iflag = 5
                    dudt = d5udt(h(j),tinft)
              elseif (tinft .ge. tpkt) then
                    iflag = 6
                    dudt = d6udt(h(j),tinft)
              endif
              tip = tinft + tnear
      tempmu = (dudh(h(j),tinft) + dudh(h(j),t(j,i)))*(tinft - t(j,i))
      mu(j,ip) = (dudh(h(j),t(j,ip)) + dudh(h(j),tip))*(t(j,ip) - tinft)
              mu(j,ip) = tempmu + mu(j,ip)
              cux = -d2udt(h(j),tinft)/dudh(h(j),tinft)
              ugrad = dudh(h(j),tip)
              udiff = u(j,ip) - u(j,i)

            if (abs(dudt) .lt. 1.d-25 .or. abs(ugrad) .lt. 1.d-25
     1           .or. udiff .eq. 0.d+00) then
                    avcu = 0.d+00
                    avcup = 0.d+00
                    avcinv = 0.d+00
                    avcinvp = 0.d+00
              else
              cuy = -dudt/ugrad
              avcu = (uinf(h(j)) - u(j,i))*(cux + cu(j,i))/udiff
              avcup = (u(j,ip) - uinf(h(j)))*(cu(j,ip) + cuy)/udiff
              avcinv = (1.d+00/cux + 1.d+00/cu(j,i))/udiff
              avcinv = (uinf(h(j)) - u(j,i))*avcinv
              avcinvp = (1.d+00/cu(j,ip) + 1.d+00/cuy)/udiff
              avcinvp = (u(j,ip) - uinf(h(j)))*avcinvp
            endif
            goto 790
         else
            ierr = 6
            goto 888
         endif
      endif
 730  if (tmt .lt. t(j,ip)) then
         if (tbrkt .ge. t(j,ip)) then
            iflag = 4
            icnt = 0
            dudt = d4udt(h(j),tmt)
         endif
         if(tmt .ge. tbrkt .or. iflag .eq. 4) then
            iflag = iflag + icnt
            if (icnt .eq.2) dudt = d5udt(h(j),tmt)
            if(tmt .ge. tpkt) then
                  iflag = 6
                  dudt = d6udt(h(j),tmt)
            endif
            tmp = tmt + tnear
      tempmu = (dudh(h(j),tmt) + dudh(h(j),t(j,i)))*(tmt - t(j,i))
      mu(j,ip) = (dudh(h(j),t(j,ip)) + dudh(h(j),tmp))*(t(j,ip) - tmt)
            mu(j,ip) = tempmu + mu(j,ip)
            cux = -d3udt(h(j),tmt)/dudh(h(j),tmt)
            ugrad = dudh(h(j),tmp)
            udiff = u(j,ip) - u(j,i)

          if (abs(dudt) .lt. 1.d-25 .or. abs(ugrad) .lt. 1.d-25
     1         .or. udiff .eq. 0.d+00) then
                  avcu = 0.d+00
                  avcup = 0.d+00
                  avcinv = 0.d+00
                  avcinvp = 0.d+00
            else
            cuy = -dudt/ugrad
            avcu = (um - u(j,i))*(cux + cu(j,i))/udiff
            avcup = (u(j,ip) - um)*(cu(j,ip) + cuy)/udiff
            avcinv = (1.d+00/cux + 1.d+00/cu(j,i))/udiff
            avcinv = (um - u(j,i))*avcinv
            avcinvp = (1.d+00/cu(j,ip) + 1.d+00/cuy)/udiff
            avcinvp = (u(j,ip) - um)*avcinvp
            endif
            goto 790
         else
            ierr = 7
            goto 888
         endif
      endif
 740  if (tbrkt .lt. t(j,ip)) then
         if (tpkt .ge. t(j,ip)) then
            iflag = 5
            icnt = 0
            dudt = d5udt(h(j),tbrkt)
          endif
          if(tbrkt .ge. tpkt .or. iflag .eq. 5) then
             iflag = iflag + icnt
             if (icnt .eq. 2 ) dudt = d6udt(h(j),tbrkt)
             tbp = tbrkt + tnear
       tempmu = (dudh(h(j),tbrkt) + dudh(h(j),t(j,i)))*(tbrkt - t(j,i))
       mu(j,ip) = (dudh(h(j),t(j,ip)) + dudh(h(j),tbp))*(t(j,ip) - tbrkt)
             mu(j,ip) = tempmu + mu(j,ip)
             cux = -d4udt(h(j),tbrkt)/dudh(h(j),tbrkt)
             ugrad = dudh(h(j),tbp)
             udiff = u(j,ip) - u(j,i)

           if (abs(dudt) .lt. 1.d-25 .or. abs(ugrad) .lt. 1.d-25
     1         .or. udiff .eq. 0.d+00) then
                    avcu = 0.d+00
                    avcup = 0.d+00
                    avcinv = 0.d+00
                    avcinvp = 0.d+00
             else
             cuy = -dudt/ugrad
             avcu = (ubrk(h(j)) - u(j,i))*(cux + cu(j,i))/udiff
             avcup = (u(j,ip) - ubrk(h(j)))*(cu(j,ip) + cuy)/udiff
             avcinv = (1.d+00/cux + 1.d+00/cu(j,i))/udiff
             avcinv = (ubrk(h(j)) - u(j,i))*avcinv
             avcinvp = (1.d+00/cu(j,ip) + 1.d+00/cuy)/udiff
             avcinvp = (u(j,ip) - ubrk(h(j)))*avcinvp
             endif
             goto 790
          else
             ierr = 8
             goto 888
          endif
       endif
 750   if (tpkt .lt. t(j,ip)) then
             iflag = 6
             tpp = tpkt + tnear
       tempmu = (dudh(h(j),tpkt) + dudh(h(j),t(j,i)))*(tpkt - t(j,i))
       mu(j,ip) = (dudh(h(j),t(j,ip)) + dudh(h(j),tpp))*(t(j,ip) - tpkt)
             mu(j,ip) = tempmu + mu(j,ip)
             cux = -d5udt(h(j),tpkt)/dudh(h(j),tpkt)
             dudt = d6udt(h(j),tpkt)
             ugrad = dudh(h(j),tpp)
             udiff = u(j,ip) - u(j,i)

           if (abs(dudt) .lt. 1.d-25 .or. abs(ugrad) .lt. 1.d-25
     1         .or. udiff .eq. 0.d+00) then
                    avcu = 0.d+00
                    avcup = 0.d+00
                    avcinv = 0.d+00
                    avcinvp = 0.d+00
             else
             cuy = -dudt/ugrad
             avcu = (upk(h(j)) - u(j,i))*(cux + cu(j,i))/udiff
             avcup = (u(j,ip) - upk(h(j)))*(cu(j,ip) + cuy)/udiff
          avcinv = (1.d+00/cux + 1.d+00/cu(j,i))/udiff
          avcinv = (upk(h(j)) - u(j,i))*avcinv
          avcinvp = (1.d+00/cu(j,ip) + 1.d+00/cuy)
          avcinvp = (u(j,ip) - upk(h(j)))*avcinvp/udiff

          endif
          goto 790
      endif

780   mu(j,ip) = ( dudh(h(j),t(j,i)) + dudh(h(j),t(j,ip)) )*tincr
790   puph(j,ip) = 0.5d+00*mu(j,ip)/tincr
      mu(j,ip) = -0.5d+00*mu(j,ip) + mu(j,i)

      if (iflag .eq. nflag) then
          avcu = cu(j,i)
          avcup = cu(j,ip)
          if (cu(j,i) .eq. 0.d+00) then
              avcinv = 0.d+00
          else
              avcinv = 1.d+00/cu(j,i)
          endif
          if (cu(j,ip) .eq. 0.d+00) then
              avcinvp = 0.d+00
          else
              avcinvp = 1.d+00/cu(j,ip)
          endif
      endif
      nflag = iflag
      udiff = u(j,ip) - u(j,i)
      mucu(j,ip) = 0.5d+00*(avcinvp + avcinv)*udiff + mucu(j,i)
      sigcu(j,ip) = denso*0.5d+00*(avcup + avcu)*udiff + sigcu(j,i)

700   continue

c     the last two points are not centered because I want the last
c     point to be associated with the release wave T.O.A.
      nt1 = nt(j) + 1
      nt2 = nt1 + 1
      nt3 = nt2 + 1
      mu(j,nt2) = (dudh(h(j),t(j,nt2)) + dudh(h(j),t(j,nt1)))*delt(j)
      puph(j,nt2) = 0.5d+00*mu(j,nt2)/delt(j)
      puph(j,nt3) = puph(j,nt2)
      mu(j,nt3) = mu(j,nt1) - mu(j,nt2)
      mu(j,nt2) = mu(j,nt1) - 0.5d+00*mu(j,nt2)
      udiff = u(j,nt2) - u(j,nt1)
      if (cu(j,nt1) .eq. 0.d+00) then
          cuinv = 0.d+00
      else
          cuinv = 1.d+00/cu(j,nt1)
      endif
      if (cu(j,nt2) .eq. 0.d+00) then
          cuinvp = 0.d+00
      else
          cuinvp = 1.d+00/cu(j,nt2)
      endif
      mucu(j,nt2) = (cuinvp + cuinv)*udiff
      mucu(j,nt3) = mucu(j,nt1) + mucu(j,nt2)
      mucu(j,nt2) = mucu(j,nt1) + 0.5d+00*mucu(j,nt2)
      sigcu(j,nt2) = denso*(cu(j,nt2) + cu(j,nt1))*udiff
      sigcu(j,nt3) = sigcu(j,nt1) + sigcu(j,nt2)
      sigcu(j,nt2) = sigcu(j,nt1) + 0.5d+00*sigcu(j,nt2)

  800 continue
      goto 890

  888 write(6,889) ierr, j, m, ip
      goto 1000
  890 continue

c ******** OUTPUT ***********

      write(6,911) (mt(j), j = 1,ngage)    ! change 911 with ngage
      do 850 j = 1, ngage
      nt2 = nt(j) + 2
      nt3 = nt2 + 1
      kcnt = 1
      write(6,907) h(j), mt(j)

      do 845 i = 1, nt3
      if (i .gt. mrem(j) + 2 .and. i .lt. nt2) then
            knum = mrem(j) + 2 + kcnt*kedit
            if (i .lt. knum) then
                  goto 845
            else
                  kcnt = kcnt + 1
            endif
      endif
      dcomp = 1.d+00/(1.d+00 - mu(j,i))
      dencu =  1.d+00/(1.d+00 - mucu(j,i))
      test = (dcomp - dencu)/dcomp
      if (abs(test) .gt. 1.0d-04) then
            test = 1.1d+00           ! Inconsistent density calculations
      else
            test = 0.d+00            ! Consistent density calculations
      endif
      write(6,909) tc(j,i), uc(j,i), dcomp, puph(j,i), sigma(j,i),
     1 sigcu(j,i), dencu, cu(j,i), test
  845 continue
  850 continue

  889 format(/' **** Execution halted due to 2 special times being',
     1 ' less than tincr apart.'//
     2 5x,'ierr =',i3,' j =',i3,' m =',i5,' ip =',i5)
  900 format(a30)
  901 format(/'Load path calculated from surface fit: ',a30/)
  903 format(2x,'Density (g/cm3) =',f7.4,' Gauge Depths (mm) =',
     1 4f7.3/2x,'Integration Steps =',f6.4,' microsec, ',f6.4,
     2 ' mm',5x,'kedit = ',i3/)
```

```
905 format(/15x,'Surface generating parameters',//
   1 5x,'D       =',f5.2,5x,'uko    =',f7.4,3x,'lambda =',f6.3,
   1 3x,'H  =',f6.3/
   2 5x,'epsilon =',f8.6,2x,'epso   =',f8.6,2x,'beta   =',f5.3/
   3 5x,'alpha1  =',f6.4,4x,'gamma1 =',f6.2,4x,'eta    =',f8.6/
   4 5x,'alpha2  =',f6.4,4x,'gamma2 =',f5.2,5x,'tdelay =',f5.3/
   5 5x,'Um      =',f7.5,3x,'Cm     =',f6.3,4x,'tmo    =',f6.3/
   6 5x,'UL      =',f7.5,3x,'Cdsp1  =',f5.2,5x,'attn1  =',f6.4/
   7 5x,'Ub      =',f7.5,3x,'nexp   =',f6.3,4x,'attn2  =',f6.4/
   8 5x,'Up      =',f7.5,3x,'Cpk    =',f6.3,4x,'tp     =',f5.3/
   9 5x,'Ur      =',f7.5,3x,'Vp     =',f5.2,5x,'tau0   =',f6.3/)
907 format(/2x,'Depth =',f7.3,' mm ',i5,' Time steps'/
   1 3x,'time',3x,'part. velo.',x,'rho/rho0',3x,'(dudh)t',x,
   2 'stress',2x,'sig(Cu)',x,'rho(Cu)   Cu  test'/x,'microsec',4x,
   3 'km/s',25x,'GPa',6x,'GPa',11x,'km/s',/)
909 format(f7.4,4x,f6.4,5x,f7.5,2x,e9.3,x,f6.3,x,f6.3,2x,f7.5,
   1 2x,f5.2,2x,f3.1)
911 format(4i7)

 1000      stop
      end
C...|....1....|....2....|....3....|....4....|....5....|....6....|....7

      subroutine hoft(dp, nf, gk, tv)

c     find the depth at which a special time equals the current time

      parameter (mgage=5, npts=3000)
      implicit real*8 (a-h,o-z)
      real*8 lam, nexp
      common /face/ carg1, tincr, hincr, h(mgage), h3, uk3, d,
     2 ukfac, aych, lam, um, tmo, cm, alpha1, gamma1, tdel, ulin,
     3 cd, alpha2, gamma2, beta, eta, ub, attn1, attn2,
     4 nexp, up, tp, cpk, tauo, vp, urel, peps, eps, epso

      cnvrg = 1.0d+01
      icnt = 0
      gj = gk + hincr
      goto (310, 320, 330, 340, 350), nf
      goto 360

 310  testa = tk(gk)
      testb = tk(gj)

      if (tv .le. testa .or. tv .ge. testb) then
            nf = nf + 100
            goto 360
      endif
      add = 0.5d+00*hincr
      gi = gk + add
 315  testa = tk(gi)
      dtest = abs(tv - testa)
      if (dtest .le. tincr/cnvrg) then
            dp = gi
```

```
              return
        endif
        if (tv .lt. testa) then
              gj = gi
              gi = gi - add
        endif
        add = 0.5d+00*(gj - gi)
        gi = gi + add
        icnt = icnt + 1
        if (icnt .gt. 100) goto 360
        goto 315

  320   testa = tinf(gk)
        testb = tinf(gj)

        if (tv .le. testa .or. tv .ge. testb) then
              nf = nf + 100
              goto 360
        endif
        add = 0.5d+00*hincr
        gi = gk + add
  325   testa = tinf(gi)
        dtest = abs(tv - testa)
        if (dtest .le. tincr/cnvrg) then
              dp = gi
              return
        endif
        if (tv .lt. testa) then
              gj = gi
              gi = gi - add
        endif
        add = 0.5d+00*(gj - gi)
        gi = gi + add
        icnt = icnt + 1
        if (icnt .gt. 100) goto 360
        goto 325

  330   testa = tm(gk)
        testb = tm(gj)

        if (tv .le. testa .or. tv .ge. testb) then
              nf = nf + 100
              goto 360
        endif
        dp = cm*(tv - tmo)
        return

  340   testa = tbrk(gk)
        testb = tbrk(gj)

        if (tv .le. testa .or. tv .ge. testb) then
              nf = nf + 100
              goto 360
        endif
```

```
      add = 0.5d+00*hincr
      gi = gk + add
345   testa = tbrk(gi)
      dtest = abs(tv - testa)
      if (dtest .le. tincr/cnvrg) then
            dp = gi
            return
      endif
      if (tv .lt. testa) then
            gj = gi
            gi = gi - add
      endif
      add = 0.5d+00*(gj - gi)
      gi = gi + add
      icnt = icnt + 1
      if (icnt .gt. 100) goto 360
      goto 345

350   testa = tpk(gk)
      testb = tpk(gj)

      if (tv .le. testa .or. tv .ge. testb) then
            nf = nf + 100
            goto 360
      endif
      dp = cpk*(tv - tp) + h(1)
      return

360   write(6,99) nf, inct, gk, tv
99    format(/'*** Execution halted in subroutine HOFT.'/
     1 5x,'iflag =',i5,' icnt =',i4,' hv =',e11.4,' tv =',e11.4)
      stop
      end
C...|....1....|....2....|....3....|....4....|....5....|....6....|....7....|

      subroutine ssumh(sum, jj, tv, nh, ierr, mk)

c     integrate the particle acceleration along an isochron to
c     calculate stress.

      parameter (mgage=5, npts=3000)
      implicit real*8 (a-h,o-z)
      real*8 lam, nexp
      common /face/ carg1, tincr, hincr, h(mgage), h3, uk3, d,
     2 ukfac, aych, lam, um, tmo, cm, alpha1, gamma1, tdel, ulin,
     3 cd, alpha2, gamma2, beta, eta, ub, attn1, attn2,
     4 nexp, up, tp, cpk, tauo, vp, urel, peps, eps, epso

      sum = 0
      jflag = 1
      kflag = 1

      do 600 m = 1, nh
       jcnt = 2
      mp =  nh + 2 - m
      mm = mp - 1
      hvm = h(jj) + (dble(mm) - 1.d+00)*hincr
      hvp = h(jj) + (dble(mp) - 1.d+00)*hincr
      goto (610, 620, 630, 640, 650, 680), jflag

 610  stem = d1udt(hvm,tv) + d1udt(hvp,tv)
      tkt = tk(hvm)
      if (tkt.lt. tv) then
         tinft = tinf(hvm)
         if (tinft .ge. tv) then
            jflag = 2
            jcnt = 0
         endif
         if(tkt .ge. tinft .or. jflag .eq. 2) then
            call hoft(dpth, kflag, hvm, tv)
c debugging code
c     write(6,*) 'flag, h(Tspec) =',kflag, dpth
            jflag = jflag + jcnt
            if(tkt .ge. tm(hvm))   jflag = 4
            if(tkt .ge. tbrk(hvm)) jflag = 5
            if(tkt .ge. tpk(hvm))  jflag = 6
            kflag = jflag
            stema = (d1udt(dpth,tv) + d1udt(hvm,tv))*(dpth - hvm)
            goto (699, 612, 613, 614, 615, 616), jflag
 612        stemb = (d2udt(hvp,tv) + d2udt(dpth,tv))*(hvp - dpth)
      goto 619
 613        stemb = (d3udt(hvp,tv) + d3udt(dpth,tv))*(hvp - dpth)
      goto 619
 614        stemb = (d4udt(hvp,tv) + d4udt(dpth,tv))*(hvp - dpth)
      goto 619
 615        stemb = (d5udt(hvp,tv) + d5udt(dpth,tv))*(hvp - dpth)
      goto 619
 616        stemb = (d6udt(hvp,tv) + d6udt(dpth,tv))*(hvp - dpth)
 619        stem = (stema + stemb)/hincr
         else
            ierr = 1
            mk = m
            return
         endif
      endif
      goto 690

 620  stem = d2udt(hvm,tv) + d2udt(hvp,tv)
      tinft = tinf(hvm)
      if (tinft .lt. tv) then
         tmt = tm(hvm)
         if (tmt .ge. tv) then
            jflag = 3
            jcnt = 0
         endif
         if(tinft .ge. tmt .or. jflag .eq. 3) then
            call hoft(dpth, kflag, hvm, tv)
c debugging code
```

```
c     write(6,*) 'flag, h(Tspec) =',kflag, dpth
            jflag = jflag + jcnt
            if(tinft .ge. tbrk(hvm)) jflag = 5
            if(tinft .ge. tpk(hvm))  jflag = 6
            kflag = jflag
            stema = (d2udt(dpth,tv) + d2udt(hvm,tv))*(dpth - hvm)
            goto (699, 699, 623, 624, 625, 626), jflag
 623        stemb = (d3udt(hvp,tv) + d3udt(dpth,tv))*(hvp - dpth)
      goto 629
 624        stemb = (d4udt(hvp,tv) + d4udt(dpth,tv))*(hvp - dpth)
      goto 629
 625        stemb = (d5udt(hvp,tv) + d5udt(dpth,tv))*(hvp - dpth)
      goto 629
 626        stemb = (d6udt(hvp,tv) + d6udt(dpth,tv))*(hvp - dpth)
 629        stem = (stema + stemb)/hincr
         else
            ierr = 2
            mk = m
            return
         endif
      endif
      goto 690

 630  stem = d3udt(hvm,tv) + d3udt(hvp,tv)
      tmt = tm(hvm)
      if (tmt .lt. tv) then
         tbrkt = tbrk(hvm)
         if (tbrkt .ge. tv) then
            jflag = 4
            jcnt = 0
         endif
         if(tmt .ge. tbrkt .or. jflag .eq. 4) then
            call hoft(dpth, kflag, hvm, tv)
c debugging code
c     write(6,*) 'flag, h(Tspec) =',kflag, dpth
            jflag = jflag + jcnt
            if(tmt .ge. tpk(hvm))  jflag = 6
            kflag = jflag
            stema = (d3udt(dpth,tv) + d3udt(hvm,tv))*(dpth - hvm)
            goto (699, 699, 699, 634, 635, 636), jflag
 634        stemb = (d4udt(hvp,tv) + d4udt(dpth,tv))*(hvp - dpth)
      goto 639
 635        stemb = (d5udt(hvp,tv) + d5udt(dpth,tv))*(hvp - dpth)
      goto 639
 636        stemb = (d6udt(hvp,tv) + d6udt(dpth,tv))*(hvp - dpth)
 639        stem = (stema + stemb)/hincr
         else
            ierr = 3
            mk = m
            return
         endif
      endif
      goto 690
 640 stem = d4udt(hvm,tv) + d4udt(hvp,tv)
     tbrkt = tbrk(hvm)
     if (tbrkt .lt. tv) then
        tpkt = tpk(hvm)
        if (tpkt .ge. tv) then
           jflag = 5
           jcnt = 0
        endif
        if(tbrkt .ge. tpkt .or. jflag .eq. 5) then
           call hoft(dpth, kflag, hvm, tv)
c debugging code
c      write(6,*) 'flag, h(Tspec) =',kflag, dpth
           jflag = jflag + jcnt
           stema = (d4udt(dpth,tv) + d4udt(hvm,tv))*(dpth - hvm)
           goto (699, 699, 699, 699, 645, 646), jflag
 645       stemb = (d5udt(hvp,tv) + d5udt(dpth,tv))*(hvp - dpth)
     goto 649
 646       stemb = (d6udt(hvp,tv) + d6udt(dpth,tv))*(hvp - dpth)
 649       stem = (stema + stemb)/hincr
        else
           ierr = 4
           mk = m
           return
        endif
     endif
     goto 690

 650 stem = d5udt(hvm,tv) + d5udt(hvp,tv)
     if (tpk(hvm) .lt. tv) then
           call hoft(dpth, jflag, hvm, tv)
c debugging code
c      write(6,*) 'flag, h(Tspec) =',jflag, dpth
           jflag = 6
           stema = (d5udt(dpth,tv) + d5udt(hvm,tv))*(dpth - hvm)
           stemb = (d6udt(hvp,tv) + d6udt(dpth,tv))*(hvp - dpth)
           stem = (stema + stemb)/hincr
     endif
     goto 690

 680 stem = d6udt(hvm,tv) + d6udt(hvp,tv)
 690 sum = sum + 0.5d+00*stem*hincr
c debugging code
c      write(6,601) sum, stem, tv, jj, m, jflag
 600 continue
     return

 699 ierr = 7
     mk = m
     return
 601 format(2x,3e14.6,2x,3i5)

     end
C...|....1....|....2....|....3....|....4....|....5....|....6....|....7
```

```
      real*8 function daadh(hv)

      parameter (mgage=5, npts=3000)
      implicit real*8 (a-h,o-z)
      real*8 lam, nexp
      common /face/ carg1, tincr, hincr, h(mgage), h3, uk3, d,
     2 ukfac, aych, lam, um, tmo, cm, alpha1, gamma1, tdel, ulin,
     3 cd, alpha2, gamma2, beta, eta, ub, attn1, attn2,
     4 nexp, up, tp, cpk, tauo, vp, urel, peps, eps, epso

      ukt = uk(hv)
      tdspt = tdsp(hv)
      carg2 = cd*(tdspt - tmo)/hv
      aa = (um - uk(hv))/(1.d+00 - dexp(carg1 + carg2))
      terma = cd*(aa + ukt - um)/hv
      termb = tdel/h3 + dtkdh(hv) + (tmo - tdspt)/hv
      termc = ukfac*aych*dexp(-hv/lam)/lam
      daadh = aa*(termc + terma*termb)/(um - ukt)

      return
      end
C...|....1....|....2....|....3....|....4....|....5....|....6....|....7....|

      real*8 function dudh(hv,tv)

c     spatial partial derivative of the particle velocity field

      parameter (mgage=5, npts=3000)
      implicit real*8 (a-h,o-z)
      real*8 lam, nexp
      common /face/ carg1, tincr, hincr, h(mgage), h3, uk3, d,
     2 ukfac, aych, lam, um, tmo, cm, alpha1, gamma1, tdel, ulin,
     3 cd, alpha2, gamma2, beta, eta, ub, attn1, attn2,
     4 nexp, up, tp, cpk, tauo, vp, urel, peps, eps, epso

      sexp = aych*dexp(-hv/lam)

  110 if (tv .gt. tk(hv)) goto 120

      if (tv .eq. hv/d) then
            dudh = 0.d+00
            return
      endif
      elont = elon(hv)
      terma = sexp/lam/(1.d+00 + sexp)
      termb = elont*elont/h3/peps/(1.d+00 - elont)
      qexp = dexp((tv - hv/d)/alpha1)
      termc = qexp*qexp + (gamma1 - 1.d+00)*qexp - gamma1
      termc = (1.d+00 + gamma1)*qexp/termc/d/alpha1
      dudh = uchron(hv,tv)*(termb - terma - termc)
      return

  120 if (tv .gt. tinf(hv)) goto 130
      tinft = tinf(hv)
      tkt = tk(hv)
      if (tinft .eq. tkt) then
           write(6,*) 'HALTED in dudh, 120'
           stop
      endif
      terma = dtidh(hv)/(tinft - tkt)
      termb = (tinft - tv)/(tv - tkt)/(tinft -tkt)
      termc = 1.d+00 - hv*(dtkdh(hv)*termb + terma)
      terma = (tv - tkt)*(ulin - uk3)/h3/(tinft - tkt)
      dudh = -ukfac*sexp/lam + terma*termc
      return

  130 if (tv .gt. tm(hv)) goto 140
      tdspt = tdsp(hv)
      carg2 = cd*(tdspt - tmo)/hv
      aa = (um - uk(hv))/(1.d+00 - dexp(carg1 + carg2))
      terma = aa*cd*( (tv - tdspt)/hv + dtkdh(hv) + tdel/h3 )/hv
      aexp = dexp(-cd*(tv - tdspt)/hv)
      terma = terma*aexp
      termb = daadh(hv)*(1.d+00 - aexp)
      dudh = -ukfac*sexp/lam - terma + termb
      return

  140 if (tv .gt. tbrk(hv)) goto 150
      tmt = tm(hv)
      if (cm*(tv - tmo) .eq. hv) then
           denom = cm*alpha2*(1.d+00 + gamma2*hv)
           denom = denom*(1.d+00 - eta)*(1.d+00 + beta*hv)
           dudh = (um - ubrk(hv))/denom
           return
      endif
      bexp = dexp( (tv - tmt)/alpha2/(1.d+00 + beta*hv) )
      dbdh = -bexp*(1.d+00/cm + beta*(tv - tmt)/(1.d+00 + beta*hv))
      dbdh = dbdh/alpha2/(1+beta*hv)
      terma = gamma2/(bexp + gamma2*hv)
      termb = gamma2*hv + 1.d+00
      termb = dbdh*termb/(bexp - 1.d+00)/(bexp + gamma2*hv)
      termc = attn1*ub/(ubrk(hv) - um)
      dudh = (uchron(hv,tv) - um)*(termb -termc - terma)
      return

  150 if (tv .gt. tpk(hv)) goto 160
      ubrkt = ubrk(hv)
      tbrkt = tbrk(hv)
      upkt = upk(hv)
      tpkt = tpk(hv)
      if(upkt .eq. ubrkt .or. tbrkt .eq. tpkt) then
           write(6,*) 'HALTED in dudh, 150'
           stop
      endif
      dtbt = dtbdh(hv)
      power = (tbrkt/tv)**nexp
      powerb = (tbrkt/tpkt)**nexp
      bb = (upkt - ubrkt)/(1.d+00 - powerb)
      berma = nexp*(dtbt/tbrkt - 1.d+00/tpkt/cpk)
      berma = berma*powerb/(1.d+00 - powerb)
      bermb = (attn1*ub - attn2*up)/(upkt - ubrkt)
      dbbdh = bb*(bermb + berma)
      terma = nexp*bb*power*dtbt/tbrkt
      termb = dbbdh*(1.d+00 - power)
      dudh = termb - terma - attn1*ub
      return

  160 upkt = upk(hv)
      tpkt = tpk(hv)
      trlt = trl(hv)
      if (trlt .eq. tpkt) then
            write(6,*) 'HALTED in dudh, 160'
            stop
      endif
      terma = attn2*up*((tv - tpkt)/(trlt - tpkt) - 1.d+00)
      termb = (vp - cpk)*(tv-tpkt)/cpk/vp
      termc = cpk*(trlt - tpkt)
      dudh = terma + (urel - upkt)*(termb - 1.d+00/termc)

      return
      end
C...|....1....|....2....|....3....|....4....|....5....|....6....|....7....|

      real*8 function dtkdh(hv)

      parameter (mgage=5, npts=3000)
      implicit real*8 (a-h,o-z)
      real*8 lam, nexp
      common /face/ carg1, tincr, hincr, h(mgage), h3, uk3, d,
     2 ukfac, aych, lam, um, tmo, cm, alpha1, gamma1, tdel, ulin,
     3 cd, alpha2, gamma2, beta, eta, ub, attn1, attn2,
     4 nexp, up, tp, cpk, tauo, vp, urel, peps, eps, epso

      elont = elon(hv)
      terma = alpha1*elont/(1.d+00 + gamma1*(1.d+00 - elont))
      termb = (1.d+00 + gamma1)/h3/peps
      dtkdh = 1.d+00/d + termb*terma

      return
      end
C...|....1....|....2....|....3....|....4....|....5....|....6....|....7....|

      real*8 function dtidh(hv)

      parameter (mgage=5, npts=3000)
      implicit real*8 (a-h,o-z)
      real*8 lam, nexp
      common /face/ carg1, tincr, hincr, h(mgage), h3, uk3, d,
     2 ukfac, aych, lam, um, tmo, cm, alpha1, gamma1, tdel, ulin,
     3 cd, alpha2, gamma2, beta, eta, ub, attn1, attn2,
     4 nexp, up, tp, cpk, tauo, vp, urel, peps, eps, epso
```

```
      tdspt = tdsp(hv)
      terma = tdel/h3 + dtkdh(hv)
      terma = terma + (tinf(hv) - tdspt)/hv
      carg2 = cd*(tdsp(hv) - tmo)/hv
      aa = (um - uk(hv))/(1.d+00 - dexp(carg1 + carg2))
      termb = hv*(uk3 - ulin)*(1.d+00 - hv*daadh(hv)/aa)
      termb = termb/cd/(aa*h3 + (uk3 - ulin)*hv)
      dtidh = terma - termb

      return
      end
C...|....1....|....2....|....3....|....4....|....5....|....6....|....7....|

      real*8 function dtbdh(hv)

      parameter (mgage=5, npts=3000)
      implicit real*8 (a-h,o-z)
      real*8 lam, nexp
      common /face/ carg1, tincr, hincr, h(mgage), h3, uk3, d,
     2 ukfac, aych, lam, um, tmo, cm, alpha1, gamma1, tdel, ulin,
     3 cd, alpha2, gamma2, beta, eta, ub, attn1, attn2,
     4 nexp, up, tp, cpk, tauo, vp, urel, peps, eps, epso

      terma = alpha2*gamma2*(1.d+00 + beta*hv)*(1.d+00 - eta)
      terma = terma/(1.d+00 + gamma2*hv*(1.d+00 - eta))
      termb = (1.d+00 + gamma2*hv*(1.d+00 - eta))/eta
      termb = alpha2*beta*dlog(termb)
      dtbdh = 1.d+00/cm + termb + terma

      return
      end
C...|....1....|....2....|....3....|....4....|....5....|....6....|....7

      real*8 function d1udt(hv,tv)

      parameter (mgage=5, npts=3000)
      implicit real*8 (a-h,o-z)
      real*8 lam, nexp
      common /face/ carg1, tincr, hincr, h(mgage), h3, uk3, d,
     2 ukfac, aych, lam, um, tmo, cm, alpha1, gamma1, tdel, ulin,
     3 cd, alpha2, gamma2, beta, eta, ub, attn1, attn2,
     4 nexp, up, tp, cpk, tauo, vp, urel, peps, eps, epso

      if (tv .lt. hv/d) then
            write(6,*) 'HALTED in d1udt: hv, tv =', hv, tv
            stop
      endif
      if (tv .eq. hv/d) then
         d1udt = uk(hv)/(1.d+00 - elon (hv))/(1.d+00 + gamma1)/alpha1
      else
            qexp = dexp((tv - hv/d)/alpha1)
            terma = qexp*qexp + (gamma1 - 1.d+00)*qexp - gamma1
            uval = uchron(hv,tv)
            termb = uchron(hv,tv)*(1.d+00 + gamma1)*qexp/alpha1
            d1udt = termb/terma
         endif
         return
         end
C...|....1....|....2....|....3....|....4....|....5....|....6....|....7....|

         real*8 function d2udt(hv,tv)

         parameter (mgage=5, npts=3000)
         implicit real*8 (a-h,o-z)
         real*8 lam, nexp
         common /face/ carg1, tincr, hincr, h(mgage), h3, uk3, d,
     2  ukfac, aych, lam, um, tmo, cm, alpha1, gamma1, tdel, ulin,
     3  cd, alpha2, gamma2, beta, eta, ub, attn1, attn2,
     4  nexp, up, tp, cpk, tauo, vp, urel, peps, eps, epso

         d2udt = hv*(ulin - uk3)
         d2udt = d2udt/(tinf(hv) - tk(hv))/h3

         return
         end
C...|....1....|....2....|....3....|....4....|....5....|....6....|....7....|

         real*8 function d3udt(hv,tv)

         parameter (mgage=5, npts=3000)
         implicit real*8 (a-h,o-z)
         real*8 lam, nexp
         common /face/ carg1, tincr, hincr, h(mgage), h3, uk3, d,
     2  ukfac, aych, lam, um, tmo, cm, alpha1, gamma1, tdel, ulin,
     3  cd, alpha2, gamma2, beta, eta, ub, attn1, attn2,
     4  nexp, up, tp, cpk, tauo, vp, urel, peps, eps, epso

         carg2 = cd*(tdsp(hv) - tmo)/hv
         aa = (um - uk(hv))/(1.d+00 - dexp(carg1 + carg2))
         termx = dexp(-cd*(tv - tdsp(hv))/hv)
         d3udt = aa*cd*termx/hv

         return
         end
C...|....1....|....2....|....3....|....4....|....5....|....6....|....7....|

         real*8 function d4udt(hv,tv)

         parameter (mgage=5, npts=3000)
         implicit real*8 (a-h,o-z)
         real*8 lam, nexp
         common /face/ carg1, tincr, hincr, h(mgage), h3, uk3, d,
     2  ukfac, aych, lam, um, tmo, cm, alpha1, gamma1, tdel, ulin,
     3  cd, alpha2, gamma2, beta, eta, ub, attn1, attn2,
     4  nexp, up, tp, cpk, tauo, vp, urel, peps, eps, epso

         if (tv .eq. tm(hv)) then
            d4udt = 0.d+00
```

```
             return
        endif
        if (cm*(tv - tmo) .eq. hv) then
             denom = alpha2*(1.d+00 - eta)*(1.d+00 + beta*hv)
             denom = denom*(1.d+00 +gamma2*hv)
             d4udt = (ubrk(hv) - um)/denom
             return
        endif
        bexp = dexp( (tv - tm(hv))/alpha2/(1.d+00 + beta*hv) )
        termb = gamma2*hv + 1.d+00
        termb = termb*bexp/(bexp - 1.d+00)/(bexp + gamma2*hv)
        d4udt = (uchron(hv,tv) - um)*termb/alpha2/(1.d+00 + beta*hv)

        return
        end
C...|....1....|....2....|....3....|....4....|....5....|....6....|....7....|

        real*8 function d5udt(hv,tv)

        parameter (mgage=5, npts=3000)
        implicit real*8 (a-h,o-z)
        real*8 lam, nexp
        common /face/ carg1, tincr, hincr, h(mgage), h3, uk3, d,
     2  ukfac, aych, lam, um, tmo, cm, alpha1, gamma1, tdel, ulin,
     3  cd, alpha2, gamma2, beta, eta, ub, attn1, attn2,
     4  nexp, up, tp, cpk, tauo, vp, urel, peps, eps, epso

        tbrkt = tbrk(hv)
        tpkt = tpk(hv)
        if (tbrkt .eq. tpkt) then
             write(6,*) 'HALTED in d5udt'
             stop
        endif
        bb = (upk(hv) - ubrk(hv))
        bb = bb/(1.d+00 - (tbrkt/tpkt)**nexp)
        power = (tbrkt/tv)**nexp
        d5udt = bb*nexp*power/tv

        return
        end
C...|....1....|....2....|....3....|....4....|....5....|....6....|....7....|

        real*8 function d6udt(hv,tv)

        parameter (mgage=5, npts=3000)
        implicit real*8 (a-h,o-z)
        real*8 lam, nexp
        common /face/ carg1, tincr, hincr, h(mgage), h3, uk3, d,
     2  ukfac, aych, lam, um, tmo, cm, alpha1, gamma1, tdel, ulin,
     3  cd, alpha2, gamma2, beta, eta, ub, attn1, attn2,
     4  nexp, up, tp, cpk, tauo, vp, urel, peps, eps, epso

        tpkt = tpk(hv)
        trlt = trl(hv)
```

```
      if (tpkt .eq. trlt) then
            d6udt = 0.d+00
      else
            d6udt = (urel - upk(hv))/(trlt - tpkt)
      endif
      return
      end
C...|....1....|....2....|....3....|....4....|....5....|....6....|....7....|

      real*8 function elon(hv)

      parameter (mgage=5, npts=3000)
      implicit real*8 (a-h,o-z)
      real*8 lam, nexp
      common /face/ carg1, tincr, hincr, h(mgage), h3, uk3, d,
     2 ukfac, aych, lam, um, tmo, cm, alpha1, gamma1, tdel, ulin,
     3 cd, alpha2, gamma2, beta, eta, ub, attn1, attn2,
     4 nexp, up, tp, cpk, tauo, vp, urel, peps, eps, epso

c epsilon varies with depth unless epso = 0.
      if(eps .eq. 0.d+00) then
            elon = eps
      else
            elon = 1.d+00/epso + hv/h3/peps
            elon = 1.d+00/elon
      endif

      return
      end
C...|....1....|....2....|....3....|....4....|....5....|....6....|....7....|

      real*8 function tk(hv)

      parameter (mgage=5, npts=3000)
      implicit real*8 (a-h,o-z)
      real*8 lam, nexp
      common /face/ carg1, tincr, hincr, h(mgage), h3, uk3, d,
     2 ukfac, aych, lam, um, tmo, cm, alpha1, gamma1, tdel, ulin,
     3 cd, alpha2, gamma2, beta, eta, ub, attn1, attn2,
     4 nexp, up, tp, cpk, tauo, vp, urel, peps, eps, epso

      elont = elon(hv)
      tknee = alpha1*dlog((1.d+00 + gamma1*(1.d+00 - elont))/elont)
      tk = tknee + hv/d
      return
      end
C...|....1....|....2....|....3....|....4....|....5....|....6....|....7....|

      real*8 function tinf(hv)

      parameter (mgage=5, npts=3000)
      implicit real*8 (a-h,o-z)
      real*8 lam, nexp
      common /face/ carg1, tincr, hincr, h(mgage), h3, uk3, d,
     2 ukfac, aych, lam, um, tmo, cm, alpha1, gamma1, tdel, ulin,
     3 cd, alpha2, gamma2, beta, eta, ub, attn1, attn2,
     4 nexp, up, tp, cpk, tauo, vp, urel, peps, eps, epso

       tdspt = tdsp(hv)
       carg2 = cd*(tdsp(hv) - tmo)/hv
       aa = (um - uk(hv))/(1.d+00 - dexp(carg1 + carg2))
       if (uinf(hv) .le. uk(hv)) then
             tinf = tk(hv)
       else
             tinf = dlog(1.d+00 + (uk3 - ulin)*hv/aa/h3)*hv/cd
             tinf = tdspt - tinf
       endif
       return
       end
C...|....1....|....2....|....3....|....4....|....5....|....6....|....7....|

       real*8 function tdsp(hv)

       parameter (mgage=5, npts=3000)
       implicit real*8 (a-h,o-z)
       real*8 lam, nexp
       common /face/ carg1, tincr, hincr, h(mgage), h3, uk3, d,
     2 ukfac, aych, lam, um, tmo, cm, alpha1, gamma1, tdel, ulin,
     3 cd, alpha2, gamma2, beta, eta, ub, attn1, attn2,
     4 nexp, up, tp, cpk, tauo, vp, urel, peps, eps, epso

       tdsp = tk(hv) + tdel*hv/h3
       return
       end
C...|....1....|....2....|....3....|....4....|....5....|....6....|....7....|

       real*8 function tm(hv)

       parameter (mgage=5, npts=3000)
       implicit real*8 (a-h,o-z)
       real*8 lam, nexp
       common /face/ carg1, tincr, hincr, h(mgage), h3, uk3, d,
     2 ukfac, aych, lam, um, tmo, cm, alpha1, gamma1, tdel, ulin,
     3 cd, alpha2, gamma2, beta, eta, ub, attn1, attn2,
     4 nexp, up, tp, cpk, tauo, vp, urel, peps, eps, epso

       tm = tmo + hv/cm
       if (tm .le. tk(hv)) then
             tm = tk(hv)
             um = uk(hv)      ! will this work?
       endif
       return
       end
C...|....1....|....2....|....3....|....4....|....5....|....6....|....7....|

       real*8 function tbrk(hv)

       parameter (mgage=5, npts=3000)
```

```
      implicit real*8 (a-h,o-z)
      real*8 lam, nexp
      common /face/ carg1, tincr, hincr, h(mgage), h3, uk3, d,
     2 ukfac, aych, lam, um, tmo, cm, alpha1, gamma1, tdel, ulin,
     3 cd, alpha2, gamma2, beta, eta, ub, attn1, attn2,
     4 nexp, up, tp, cpk, tauo, vp, urel, peps, eps, epso

      tmt = tm(hv)
      tkt = tk(hv)
      temp = alpha2*dlog((1.d+00 + gamma2*hv*(1.d+00 - eta))/eta)
      if (tmt .le. tkt) then
            tbrk = tkt + (1.d+00 + beta*hv)*temp
      else
            tbrk = tmt + (1.d+00 + beta*hv)*temp
      endif
      if (alpha2 .eq. 0.0d+00) tbrk = tmt
      return
      end
C...|....1....|....2....|....3....|....4....|....5....|....6....|....7....|

      real*8 function tpk(hv)

      parameter (mgage=5, npts=3000)
      implicit real*8 (a-h,o-z)
      real*8 lam, nexp
      common /face/ carg1, tincr, hincr, h(mgage), h3, uk3, d,
     2 ukfac, aych, lam, um, tmo, cm, alpha1, gamma1, tdel, ulin,
     3 cd, alpha2, gamma2, beta, eta, ub, attn1, attn2,
     4 nexp, up, tp, cpk, tauo, vp, urel, peps, eps, epso

      tbrkt = tbrk(hv)
      tpk = tp + (hv - h(1))/cpk
      if (tpk .le. tbrkt) tpk = tbrkt
      return
      end
C...|....1....|....2....|....3....|....4....|....5....|....6....|....7....|

      real*8 function trl(hv)

      parameter (mgage=5, npts=3000)
      implicit real*8 (a-h,o-z)
      real*8 lam, nexp
      common /face/ carg1, tincr, hincr, h(mgage), h3, uk3, d,
     2 ukfac, aych, lam, um, tmo, cm, alpha1, gamma1, tdel, ulin,
     3 cd, alpha2, gamma2, beta, eta, ub, attn1, attn2,
     4 nexp, up, tp, cpk, tauo, vp, urel, peps, eps, epso

      trl = tauo + hv/vp
      return
      end
C...|....1....|....2....|....3....|....4....|....5....|....6....|....7....|

      real*8 function uk(hv)
      parameter (mgage=5, npts=3000)
      implicit real*8 (a-h,o-z)
      real*8 lam, nexp
      common /face/ carg1, tincr, hincr, h(mgage), h3, uk3, d,
     2 ukfac, aych, lam, um, tmo, cm, alpha1, gamma1, tdel, ulin,
     3 cd, alpha2, gamma2, beta, eta, ub, attn1, attn2,
     4 nexp, up, tp, cpk, tauo, vp, urel, peps, eps, epso

      uk = ukfac*(1.d+00 + aych*dexp(-hv/lam))
      return
      end
C...|....1....|....2....|....3....|....4....|....5....|....6....|....7....|

      real*8 function uinf(hv)

      parameter (mgage=5, npts=3000)
      implicit real*8 (a-h,o-z)
      real*8 lam, nexp
      common /face/ carg1, tincr, hincr, h(mgage), h3, uk3, d,
     2 ukfac, aych, lam, um, tmo, cm, alpha1, gamma1, tdel, ulin,
     3 cd, alpha2, gamma2, beta, eta, ub, attn1, attn2,
     4 nexp, up, tp, cpk, tauo, vp, urel, peps, eps, epso

      uinf = uk(hv) + (ulin - uk3)*hv/h3
      return
      end
C...|....1....|....2....|....3....|....4....|....5....|....6....|....7....|

      real*8 function ubrk(hv)

      parameter (mgage=5, npts=3000)
      implicit real*8 (a-h,o-z)
      real*8 lam, nexp
      common /face/ carg1, tincr, hincr, h(mgage), h3, uk3, d,
     2 ukfac, aych, lam, um, tmo, cm, alpha1, gamma1, tdel, ulin,
     3 cd, alpha2, gamma2, beta, eta, ub, attn1, attn2,
     4 nexp, up, tp, cpk, tauo, vp, urel, peps, eps, epso

      ubrk = ub*(1.d+00 - attn1*(hv - h(1)))
      if (alpha2 .eq. 0.d+00) then
            temp = tm(hv)           ! reset um value
            ubrk = um
      endif
      return
      end
C...|....1....|....2....|....3....|....4....|....5....|....6....|....7....|

      real*8 function upk(hv)

      parameter (mgage=5, npts=3000)
      implicit real*8 (a-h,o-z)
      real*8 lam, nexp
      common /face/ carg1, tincr, hincr, h(mgage), h3, uk3, d,
     2 ukfac, aych, lam, um, tmo, cm, alpha1, gamma1, tdel, ulin,
     3 cd, alpha2, gamma2, beta, eta, ub, attn1, attn2,
     4 nexp, up, tp, cpk, tauo, vp, urel, peps, eps, epso

      upk = up*(1.d+00 - attn2*(hv - h(1)))
      return
      end
C...|....1....|....2....|....3....|....4....|....5....|....6....|....7....|

      real*8 function uchron(hv,tv)

      parameter (mgage=5, npts=3000)
      implicit real*8 (a-h,o-z)
      real*8 lam, nexp
      common /face/ carg1, tincr, hincr, h(mgage), h3, uk3, d,
     2 ukfac, aych, lam, um, tmo, cm, alpha1, gamma1, tdel, ulin,
     3 cd, alpha2, gamma2, beta, eta, ub, attn1, attn2,
     4 nexp, up, tp, cpk, tauo, vp, urel, peps, eps, epso

      to = hv/d
      arg = -hv/lam
      ukt = uk(hv)

      carg2 = cd*(tdsp(hv) - tmo)/hv
      aa = (um - ukt)/(1.d+00 - dexp(carg1 + carg2))
      ampb = (ubrk(hv) - um)/(1.d+00 - eta)
      trat = (tbrk(hv)/tpk(hv))**nexp

      if (tv .le. tk(hv)) then
            tstep = tv - to
            etoal = dexp(tstep/alpha1)
      uchron = ukt*(etoal-1.d+00)/(etoal + gamma1)/(1.d+00 - elon(hv))

      elseif (tv .le. tinf(hv)) then
            utemp = (uinf(hv)-ukt)*(tv - tk(hv))
            uchron = ukt + utemp/(tinf(hv) - tk(hv))

      elseif (tv .le. tm(hv)) then
            darg = -cd*(tv-tdsp(hv))/hv
            uchron = ukt + aa*(1.d+00 - dexp(darg))

      elseif (tv .le. tbrk(hv)) then
            earg = alpha2*(1.d+00 + beta*hv)
            fexp = dexp((tv-tm(hv))/earg)
            uchron = um + ampb*(fexp -1.d+00)/(fexp + gamma2*hv)

      elseif (tv .le. tpk(hv)) then
            crat = (tbrk(hv)/tv)**nexp
            uchron = (upk(hv) - ubrk(hv))*(1.d+00 - crat)
            uchron = uchron/(1.d+00 - trat)
            uchron = uchron + ubrk(hv)

      else
            utemp = (urel - upk(hv))*(tv - tpk(hv))
            uchron = upk(hv) + utemp/(trl(hv) - tpk(hv))
      endif

      return
      end
C...|....1....|....2....|....3....|....4....|....5....|....6....|....7
```

Input for SURFGEN and SSHIST:

```
  Carr12.ia   10/5/89
2.712 .001 10 .002      density, tincr, tedit, hincr
1.355 2.942 4.510 3.967    gauge depths
6.28 1.75   shock speed, H
0.11 3.65   uko, lambda
0.115       um
-0.006 3.716      tmo, Cm
0.007 11.0 alpha1, gamma1
0.010 0.150      eps, epso
0.020       tdelay
0.07        UL
10.0        Cdsp1
0.021 3.00 alpha2, gamma2
0.24 0.150 beta, eta
0.185 0.01 Ub, attn1
10.0        nexp
0.2045 0.01      Up, attn2
0.720 3.411      tp, Cpk
1.837 5.63 tau0, Vp
0.2155           Ur
```

```
      program unload
c
c This routine is used for extracting constant particle velocity
c wave speeds, Cu from particle velocity unloading histories.
c I use a two-point, linear interpolation to determine the time
c corresponding to a chosen amplitude.
c
c J.B. Aidun, WSU SDL, Pullman WA 10/89.  Modification of the program CUHT.
c
c   ut(i,j,k) = particle velocity or stress history
c           i = gage index
c           j = abscissa (1) or ordinate (2) of a data pair
c           k = data pair index
c   h(i) = depth of gage i
c   datht(i,1,k) = array of times corresponding to chosen levels of
c              of particle velocity or stress for each gage.
c   ndata(i) = number of data pairs in the record of gage i
c           neighboring the end of the jth region on the
c           late-time side
c   nbtwn(i,j) = the value of the data pair index of gage i for the
c             point neighboring the jth chosen level on the
c             late-time side.
c   cu(i,k) = wave speed at constant amplitude.

      parameter (mdim = 1000, mgage = 3, m2gage = 6)

      real max(mgage)
      character*15 nam1(6),nam2,nam4

      dimension ut(mgage,5,mdim), ndata(mgage), nbtwn(mgage,mdim),
     1 h(m2gage),datht(mgage,2,mdim), cu(mgage,mdim), npk(mgage),
     2 strs(m2gage,mdim), strn(m2gage,mdim), pkstt(3,2), npt

      read(5,*)(h(i),i=1,3)

      rho = 2.712              ! calcite density
      ijk = 6
      n2gage = ijk - 1
      do 10 i= 4, n2gage
      j = i - 3
      j1 = j + 1
 10   h(i) = (h(j1) + h(j))*0.5
      h(ijk) = (h(3) + h(1))*0.5

      read(5,908) nam2
      nam4 = nam2(:index(nam2,' ')-1)//'cu'
      read(5,*) pkstt(1,1)             ! peak u
      read(5,*) (pkstt(2,i), i = 2,3) ! peak rho/rho0
      read(5,*) (pkstt(3,i), i = 2,3) ! peak stress
      pkstt(2,1) = pkstt(2,2)         ! average Cu, stress, and
      pkstt(3,1) = pkstt(3,2)         ! strain are stored in
                                      ! place of the first gauge

c Read in each time history ..................................
```

```
      do 58 i=1,3

      read(5,908)nam1(i)
      open(1,file=nam1(i),status='old')
      j=1
 25     read(1,*,end=30) ut(i,1,j),ut(i,2,j)
      j = j + 1
      goto 25
 30   ndata(i) = j - 1
      close(1)

c Find the peak of each record

      max(i)=0.
      do 50 j = 1, ndata(i)
      if (ut(i,2,j) .lt. max(i)) goto 50
      max(i) = ut(i,2,j)
      npk(i) = j
 50     continue
 58   continue

      mnd = 1
      if (ut(2,2,ndata(2)) .gt. ut(1,2,ndata(1))) mnd = 2
      if (ut(3,2,ndata(3)) .gt. ut(mnd,2,ndata(mnd))) mnd = 3

c Determine the chosen particle velocities within each region.
c These velocities are the same at each gauge plane. 0.5 m/s
c increments are used.

      npt = (pkstt(1,1) - ut(mnd,2,ndata(mnd)))*2000

      do 140 i = 1, 3

      k = 1

      do 120 j = 1, npt
      datht(i,2,j) = pkstt(1,1) - (j-1)*0.0005
 120  continue

c Interpolate to find the times at the chosen particle velocities

      do 132 j = npk(i), ndata(i)
  33  if (k .gt. npt) goto 132
      if (ut(i,2,j) .gt. datht(i,2,k)) goto 132
      nbtwn(i,k) = j
      k = k + 1
      if (ut(i,2,j) .le. datht(i,2,k)) goto 33
 132  continue

      do 135 j= 1, npt

      nup = nbtwn(i,j)
      ndn = nup -1
      deltat = (ut(i,2,nup)-datht(i,2,j))*(ut(i,1,nup)-ut(i,1,ndn))
      deltat = deltat/ ( ut(i,2,nup) - ut(i,2,ndn) )
      datht(i,1,j) = ut(i,1,nup) - deltat
 135  continue
 140  continue

c  calculate the phase velocities and integrate for stress
c  and volume.  velocities in km/s.
c     strn(i,j) = compression strain (dimensionless)
c     strs(i,j) = Vo * [T(h,t) - T(h,0)], T = normal stress
c                (+ve in compression)
c  Multiply strs by the appropriate density (gm/cm**3) to
c  obtain GPa:   GPa  [=]  (g/cm**3)  (km/s)**2

      do 250 i = 1, 3
      im = i - 1
      ii = i
      dlth = h(ii) - h(im)
      if (i .eq. 1) then
            ii = 3
            im = 1
            dlth = h(3) - h(1)
      endif
      cu(i,1) = dlth/(datht(ii,1,1) - datht(im,1,1))
      strn(i,1) = 1. - 1./pkstt(2,i)
      strs(i,1) = pkstt(3,i)

      do 220 k = 2, npt
      strs(i,k) = 0.
      strn(i,k) = 0.
      k1 = k - 1
      cu(i,k) = dlth/(datht(ii,1,k) - datht(im,1,k))
      base = (datht(ii,2,k) - datht(ii,2,k1))*0.5
      strs(i,k) = base*(cu(i,k) + cu(i,k1))*rho + strs(i,k1)
      strn(i,k) = base*(1./cu(i,k) + 1./cu(i,k1)) + strn(i,k1)
 220  continue
 250  continue

c     Output -------------------------------------

      open(10,file=nam4,status='new')
      open(11,file=nam2,status='new')

      do 310 i= 1, 3
      do 300 k = 1, npt
      ut(i,1,k) = datht(i,1,k)        ! in microsec.
      ut(i,2,k) = datht(i,2,k)        ! in km/s
      ut(i,3,k) = cu(i,k)                   ! in km/s
      ut(i,4,k) = 1./(1. - strn(i,k))       ! rho/rho0
 300  ut(i,5,k) = strs(i,k)                 ! GPa
 310  continue

      write(10,801) npt, (h(i),h(i+3),i=1,3)
      write(10,811)
      write(11,905) npt, h(6), h(2), h(3)
      write(11,911)

      do 320 k = 1, npt
      write(10,803) k,ut(1,2,k),ut(1,1,k),ut(1,3,k),
     1 (ut(i,1,k),ut(i,3,k),i=2,3)
      write(11,903) k,ut(1,2,k),(ut(i,1,k), ut(i,4,k), ut(i,5,k),
     1 i = 1, 3)
  320 continue

      write(10,805) h(6), h(2), h(3)
      write(10,815)

      do 330 k = 1, npt
  330 write(10,807) k,ut(1,2,k),(ut(i,4,k),ut(i,5,k),i=1,3)

  355 write(10,983) (nam1(i), i=1,3)
      write(10,985) rho, npt

C...|....1....|....2....|....3....|....4....|....5....|....6....|....7
  801 format('   3',i4/'Lagr. Position: ',5(f5.3,' mm',3x),f5.3,
     1 ' mm')
  803 format(i3,7e11.4)
  805 format(///'Lagr. Position:',5x,2(f5.3,' mm',11x),2x,f5.3,
     1 ' mm')
  807 format(i3,e11.4,x,3(f8.5,3x,f6.3,3x))
  811    format(/4x,'Part. Velo   T.O.A.  Phase Velo.   T.O.A.  ',
     1 'Phase Velo.  T.O.A.  Phase Velo.'/)
  815    format(/4x,'Part. Velo  N.Dens. Stress GPa  N.Dens. ',
     1 'Stress GPa  N.Dens. Stress GPa'/)
  903 format(i3,f7.4,x,3(x,f7.4,f8.5,x,f6.3))
  905 format(i4/'Lagr. Position:',6x,2(f5.3,' mm',14x),f5.3,' mm')
  908    format(a14)
  911 format(/3x,'Part. Velo  T.O.A. N.Dens. Stress T.O.A. ',
     1 'N.Dens. Stress T.O.A. N.Dens. Stress'/)
  983 format(/2x,'INPUT FILES'/5x,3a20)
  985 format(/'DENSITY =',f6.4,'    No. of Levels =',i5)
      stop
      end
C...|....1....|....2....|....3....|....4....|....5....|....6....|....7
c             input file
c
 1.284 3.38 5.438        gauge locations
 rlse.21              output file name  (use no tabs)
 0.407                   peak p. v. (km/s)
 1.10911 1.10929         peak rho/rho0 at gauges 2, 3, resp.
 4.698 4.700             peak stress (GPa) at gauges  2, 3, resp.
 u14g1.21c             filename of record 1 (use no tabs)
 u8g2.21c             filename of record 2 (use no tabs)
 u10g3.21c             filename of record 3 (use no tabs)
```

```
      program revcnc

c Calculation of density compression, bulk and shear modulus,
c and extent of transformation in a random calcite polycrystal
c versus applied hydrostatic pressure, pmin <= P <= pmax.
c This calculation was developed by elaborating on Grady's stress
c concentration model. Its additional features are
c inclusion of the pressure dependence of the bulk modulus,
c and using a more physically reasonable distribution function.
c The distribution function has a finite width, limited to positive
c pressures, and, in phase 2, it becomes bimodal rather than be
c the continuation of the distribution from the low-pressure phase.
c
c The values of the bulk modulus are taken from Singh and Kennedy's
c hydrostatic compression curve. Beyond 17.5 kbar, the bulk
c modulus is linearly extraploated with pressure. The variable
c dk2 is the slope, dK/dP for this extrapolation. Singh and
c Kennedy's phase 2 bulk modulus and its pressure derivative at
c 17.5 kb are 424 kb and 47.4, respectively.
c
c The shear modulus in calcite is the mean of the Hashin-Shtrikman
c bounds otained from Vo Thanh's elastic constants. Because these
c values only range from 314 kb to 319 kb, which is less than their
c uncertainties, I take calcite's shear modulus to be constant,
c G(I) = 317 kb. Based on the shock measurements, the shear modulus
c in phase 2 is constant, G(II) = 255 kb. The shear modulus is phase
c 3 is calculated from the bulk modulus assuming Poisson's ratio is
c a constant, input value.
c
c Stresses and moduli are calculated in kilobars and output in GPa.
c Velocities are in km/s.
c
c     John B. Aidun, July 1989.
c-----------------------------------------------------------------

      parameter (mm=100)
      implicit real*8 (a-h,o-z)

      dimension fbmod(mm), rbmod(mm), rho(mm), tcmpr(mm), press(mm),
     1  shmod(mm), vpfrz(mm), vprlx(mm), vs(mm), extnt(mm),
     2  rholn(mm), pmean(mm)
      common /const/ a, ab, alpha, anorm, crit, cosm, fm, hh, iflag,
     1             pold, ptrns, sincm, z, bnorm, icnt

c     The integration step size, dw, must be a factor of ptrns,
c     (17.5 - ptrns), and (pmax + z*hwidth - 17.5) so that the
c     integration covers the entire pressure range.
      data  dw, fm,
     1      a1,a2, b0,b1,b2,
     2      g1,g2/
     3      1.0d-02, 4.4934095d+00,
     4      -1.407d-03, 5.107d-06, 0.0382d+00, -6.78d-03, 1.291d-04,
     5      317.0d+00,255.d+00/
```

```
      jflag = 0
      rt2 = sqrt(2.)
      pi = 4.0d+00 * datan(1.0d+00)
      rt2pi = rt2 * sqrt(pi)
      pold = 0.0d+00
      sincm = sinc(fm)                        ! = -0.2172336, yes?
      cosm = sincm
      hh = sumten(fm,0.0d+00,1.0d+00)                  ! = 1.65557, yes?
      anorm = 2.*(hh - fm*cosm)         ! = 5.26338, yes?
c     write(6,*) 'sincm, hh, anorm =',sincm, hh, anorm

c    distribution half-width = fm/a, with a = 1/(alpha * P)
c                       ---->        half-width = fm * alpha * P.
c    alpha < 1/fm for no tensions.
c    ENTER alpha as the constant ratio halfwidth/P < 1.

      read(5,*) alpha
      read(5,*) z
      read(5,*) ptrns
      read(5,*) vtrns
      read(5,*) pmin
      read(5,*) pmax
      read(5,*) dp
      read(5,*) dk2
      read(5,*) hwidth         ! maximum half-width in phase 1

      write(6,901) alpha, z, vtrns, ptrns, g1, g2, dw, dk2, hwidth
      write(6,903)

      crit = hwidth/alpha
      alpha = alpha/fm
      rho(1) = 1. + a1*pmin + a2*pmin*pmin
      rho(1) = 1./rho(1)   ! Singh and Kennedy's calcite isotherm
      rholn(1) = log(rho(1))

      nmax = (pmax - pmin)/dp + 1
      do 500 n = 1, nmax
      press(n) = pmin + (n-1)*dp
      w = 0.0
      pmean(n) = 0.0
      fbmod(n) = 0.0

      amax = ptrns/dw
      imax = amax + 0.1
      bmax = imax
      if (amax - bmax .gt. 1.0d-04) then
            jflag = 100
            goto 750
      endif

      do 100 idw = 1, imax
      w = (idw -1)*dw
      w2 = w + dw
      bmod1 = -(1. + a1*w + a2*w*w)/(a1 + 2.0*a2*w)
      bmod2 = -(1. + a1*w2 + a2*w2*w2)/(a1 + 2.0*a2*w2)
      wght = dstrb(w,press(n))
      wght2 = dstrb(w2,press(n))
      pmean(n) = pmean(n) + 0.5*dw*(w*wght + w2*wght2)
      fbmod(n) = fbmod(n) + 0.5*dw*(wght/bmod1 + wght2/bmod2)
c     write(6,701) n,w,press(n),bmod1,fbmod(n)
c 701       format('n, w, press, bmod1, fbmod = ',i3,4f9.4)
 100  continue

      amax = (17.5 - ptrns)/dw
      imax = amax + 0.1
      bmax = imax
      if (amax - bmax .gt. 1.0d-04) then
            jflag = 200
            goto 750
      endif

      do 200 idw = 1, imax
      w = (idw -1)*dw + ptrns
      w2 = w + dw
      bmod1 = -(1. + b0+ b1*w + b2*w*w)/(b1 + 2.0*b2*w)
      bmod2 = -(1. + b0+ b1*w2 + b2*w2*w2)/(b1 + 2.0*b2*w2)
      wght = dstrb(w,press(n))
      wght2 = dstrb(w2,press(n))
      pmean(n) = pmean(n) + 0.5*dw*(w*wght + w2*wght2)
      fbmod(n) = fbmod(n) + 0.5*dw*(wght/bmod1 + wght2/bmod2)
 200  continue

      amax = (pmax + z*hwidth - 17.5)/dw
      imax = amax + 0.1
      bmax = imax
      if (amax - bmax .gt. 1.0d-04) then
            jflag = 300
c           write(6,*) 'amax,bmax =',amax,bmax
            goto 750
      endif

c stop integration when dstrb(w,pmax) = 0.0
      do 300 idw = 1, imax
      w = (idw -1)*dw + 17.5
      w2 = w + dw
      bmod1 = 424.0 + dk2*(w - 17.5)
      bmod2 = 424.0 + dk2*(w2 - 17.5)
      wght = dstrb(w,press(n))
      wght2 = dstrb(w2,press(n))
      pmean(n) = pmean(n) + 0.5*dw*(w*wght + w2*wght2)
      fbmod(n) = fbmod(n) + 0.5*dw*(wght/bmod1 + wght2/bmod2)
 300  continue

      tcmpr(n) = vtrns * beta(press(n))

c     Calculate the density compression and extent of
c     transformation for the case of hydrostatic loading.
```

```
      rbmod(n) = tcmpr(n) + fbmod(n) !compressibilities, temporarily
      if (n .eq. 1) go to 450
      rholn(n) = rholn(n-1) + 0.5*dp*(rbmod(n) + ctemp)
      rho(n) = exp(rholn(n))
  450 fbmod(n) = 1./fbmod(n)
      ctemp = rbmod(n)
      rbmod(n) = 1. / rbmod(n)
      extnt(n) = textnt(press(n))          !fraction of phase 2

c     Calculate the frozen-composition and relaxed longitudinal
c      wave speeds, and the shear wave speed.    The shear modulus
c      is the Reuss average for the two-phase aggregate.

      shmod(n) = (1. - extnt(n))/g1 + extnt(n)/g2
      shmod(n) = 1./shmod(n)
      dens = 2.712*rho(n)
      vpfrz(n) = ( fbmod(n) + shmod(n)/0.75 )*1.0d+09/dens
      vpfrz(n) = sqrt(vpfrz(n))*1.0d-05
      vprlx(n) = ( rbmod(n) + shmod(n)/0.75 )*1.0d+09/dens
      vprlx(n) = sqrt(vprlx(n))*1.0d-05
      vs(n) = sqrt(shmod(n)*1.0d+09/dens) * 1.0d-05

c     Output  (convert kbar to GPa) ----------------------------

      press(n) = press(n)*0.1
      pmean(n) = pmean(n)*0.1
      fbmod(n) = fbmod(n)*0.1
      rbmod(n) = rbmod(n)*0.1
      shmod(n) = shmod(n)*0.1
      write(6,905) press(n), pmean(n), rho(n), extnt(n),fbmod(n),
     1          rbmod(n), shmod(n), vpfrz(n), vprlx(n), vs(n)

  500 continue

 750  if (jflag .ne. 0) write(6,975) jflag

  901       format ('Hydrostatic compression of a calcite rock:'/
     1 'Variable moduli and a bimodal local mean stress distribution'
     2 //'alpha =',f6.3,'  z = a/b =',f6.3,'  (1 - Vtrns/Vo) =',
     3 f6.4,'  Ptrns =',f5.1,' kbar'/
     4 'G(I)  =',f5.0,' kbar    G(II) =',f5.0,' kbar  ',
     5 'Integration step =',f5.3/
     6 'dK(II)/dP =',f5.1,10x,
     7 'distribution half-width limited to +/-',f4.1,' kbar')
  903  format (/x,'Press/Pmean',2x,'rho/rho0',4x,'extent',3x,
     1 'Kforz',4x,'Krlxd',5x,'G',3x,' Vpfrz',' Vprlx',3x,'Vs'/
     2 6x,'GPa',27x,'GPa',6x,'GPa',6x,'GPa',3(3x,'km/s')/)
  905 format (x,f5.3,x,f5.3,2x,f8.5,x,f8.4,3f9.2,x,3f6.2)
  975  format (/5x,'Integration step does not fill range of do'
     1  ' loop',i4,'. Program halted.')

       stop
       end
C...|....1....|....2....|....3....|....4....|....5....|....6....|....7....|....8
```

```
      real*8 function beta(p)

c     Calculation of effective compressibility due to the
c     transformation under applied pressure p, divided by the
c     fractional volume change of the transformation.

      implicit real*8 (a-h,o-z)
      common /const/ a, ab, alpha, anorm, crit, cosm, fm, hh,
     1           iflag, pold, ptrns, sincm, z, bnorm, icnt

      y = a*(ptrns - p)/z
      b = a/z
      if (p + fm/a .le. ptrns) then
            beta = 0.0
      elseif (p .le. ptrns) then
            pomega = wintg(p-fm/a,ptrns,a,p)
            dbnorm = (domega(p) - 1.)/(p - pomega)
            dbnorm = (dbnorm + iflag/p)*bnorm   ! b* dR/dP
            beta = textnt(p)*(iflag - p*dbnorm/bnorm)/p
      elseif (p .le. ptrns + fm/b) then
            siny = sin(y)
            sumb = sumten(y,0.0d+00,1.0d+00)
            pomega = wintg(p-fm/a,ptrns,a,p)
            dbnorm = iflag*(cos(y) - 1.)/b/p - (1. + iflag)*sumb
            dbnorm = dbnorm + (iflag*ptrns/p + 1.)*siny
            dbnorm = dbnorm  + b*p*(sinc(y) - cosm)
            dbnorm=(dbnorm + 0.5*anorm*(1. + iflag))/(p-pomega)
            dbnorm= dbnorm + bnorm*(domega(p) -1.)/(p - pomega)
            beta = p*b*sinc(y) + iflag*siny
            beta = (beta - b*cosm*(p + iflag*(ptrns -p)))/p/bnorm
            beta = beta + textnt(p)*(iflag - p*dbnorm/bnorm)/p
      else
            beta = 0.0
      endif

      return
      end
C...|....1....|....2....|....3....|....4....|....5....|....6....|....7....|....8

      real*8 function domega(p)

c     Calculation of the pressure derivative of pomega, which is
c     the contribution to the mean value of the local pressure from
c     the low-pressure-phase fraction of the sample.  Evaluated at
c     applied pressure p.

      implicit real*8 (a-h, o-z)
      common /const/ a, ab, alpha, anorm, crit, cosm, fm, hh, iflag,
     1           pold, ptrns, sincm, z, bnorm, icnt

      y = a*(ptrns-p)
      siny = sin(y)
      domega = 0.5*a*cosm*( ptrns*ptrns - (p-fm/a)*(p-fm/a) )/p
      domega = domega + hh + sumten(y,0.0d+00,1.0d+00)
      domega = domega - (1. + iflag)*(fm*cosm + siny)
      domega = domega - a*p*(sinc(y) - cosm)
      domega = domega - (cos(y) + y*siny - (1. + fm*fm)*cosm)/a/p
      domega = domega/anorm

      return
      end
C...|....1....|....2....|....3....|....4....|....5....|....6....|....7....|....8

      real*8 function dstrb(w,p)

c   Distribution density of mass fraction of the sample under
c   local mean stress w when the pressure p is applied.

      implicit real*8 (a-h, o-z)
      common /const/ a, ab, alpha, anorm, crit, cosm, fm, hh,
     1          iflag, pold, ptrns, sincm, z, bnorm, icnt

c  set the value of a.  The half-width of the distribution is fm/a.

      if (p .eq. pold) goto 810
      pold = p
      icnt = 0
      if (p .le. crit) then
            ab = 1.0/alpha/p
            a = ab
            iflag = 1
      else
            a = ab
            iflag = 0
      endif

 810  x = a*(w - p)
      b = a/z
      if (p + fm/a .le. ptrns) then

            if (w .le. p-fm/a .or. w .ge. p+fm/a) then
                  dstrb = 0.0
            else
                  dstrb = a*(sinc(x) - cosm)/anorm
            endif
      elseif (p .le. ptrns) then

            if (w .le. p-fm/a .or. w .ge. ptrns+fm/b) then
                  dstrb = 0.0
            elseif (w .le. ptrns) then
                  dstrb = a*(sinc(x) - cosm)/anorm
            else
                  if (p .eq. pold .and. icnt .gt. 0) goto 820
                  icnt = 1
                  pomega = wintg(p-fm/a,ptrns,a,p)
                  bnorm = anorm*wintg(ptrns,ptrns+fm/b,b,ptrns)
                  bnorm = bnorm/(p - pomega)
 820              dstrb = b*(sinc(b*(w-ptrns)) - cosm)/bnorm
```

```
              endif
         elseif (p .le. ptrns + fm/b) then

              if (w .le. p-fm/a .or. w .ge. p+fm/b) then
                   dstrb = 0.0
              elseif (w .le. ptrns) then
                   dstrb = a*(sinc(x) - cosm)/anorm
              else
                   if (p .eq. pold .and. icnt .gt. 0) goto 830
                   icnt = 1
                   pomega = wintg(p-fm/a,ptrns,a,p)
                   bnorm = anorm*wintg(ptrns,p+fm/b,b,p)
                   bnorm = bnorm/(p - pomega)
 830               dstrb = b*(sinc(x/z) - cosm)/bnorm
              endif
         else

              if (w .le. p-fm/b .or. w .ge. p+fm/b) then
                   dstrb = 0.0
              else
                   dstrb = b*(sinc(x/z) - cosm)/anorm
              endif
         endif

         return
         end
C...|....1....|....2....|....3....|....4....|....5....|....6....|....7....|....8

         real*8 function sinc(x)

c   Calculation of zero-order spherical Bessel function using a
c   series approximation for small arguments.

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

         if (abs(x) .le. 0.25) then
              sinc = 1. - x*x/6.
         else
              sinc = sin(x)/x
         endif

         return
         end
C...|....1....|....2....|....3....|....4....|....5....|....6....|....7....|....8

         real*8 function sumten(x,y,c)

c   Sum of the first 11 terms in the series solution for the
c   integral of the zero-order spherical Bessel function.
c   For the limits within the smallest magnitude minima of the
c   function, this partial sum is converged to better than 1 ppm.

         implicit real*8 (a-h, o-z)
```

```
      asign = -1.0
      sumten = 0.0
      fact = 1.

      do 870 n1 = 1, 11
      n = n1 - 1
      nn = 2*n + 1
      fact = fact * 2 * n * nn
      if (n1 .eq. 1) fact = 1.
      asign = -1.0*asign
      sumten = sumten + asign*((c*(x-y))**nn)/nn/fact
 870  continue

      return
      end
C..|....1....|....2....|....3....|....4....|....5....|....6....|....7....|....8

      real*8 function textnt(p)

c   Mass or mole fraction of high-pressure phase under applied
c   pressure p.

      implicit real*8 (a-h, o-z)
      common /const/ a, ab, alpha, anorm, crit, cosm, fm, hh, iflag,
     1          pold, ptrns, sincm, z, bnorm, icnt

      temp = p + fm /a
      if (p + fm/a .le. ptrns) then
            textnt = 0.0
      elseif (p .le. ptrns) then
            textnt = (hh - fm*cosm)/bnorm
      elseif (p .le. ptrns + fm*z/a) then
            b = a/z
            textnt = hh + sumten(p,ptrns,b)
            textnt = (textnt - b*cosm*(p - ptrns + fm/b))/bnorm
      else
            textnt = 1.0
      endif

      return
      end
C..|....1....|....2....|....3....|....4....|....5....|....6....|....7....|....8

      real*8 function wintg(x,y,c,q)

c   Integral from x to y of local mean stress, weighted by the
c   distribution density, whose parameters are c and q.

      implicit real*8 (a-h, o-z)
      common /const/ a, ab, alpha, anorm, crit, cosm, fm, hh, iflag,
     1          pold, ptrns, sincm, z, bnorm, icnt

      arg1 = c*(x-q)
      arg2 = c*(y-q)
      wintg = q*(sumten(arg2,0.0d+00,1.0d+00) -
     1        sumten(arg1,0.0d+00,1.0d+00))
      wintg = wintg + (cos(arg1) - cos(arg2))/c
      wintg = wintg - 0.5*c*cosm*(y*y -x*x)
      wintg = wintg/anorm

      return
      end
C...|....1....|....2....|....3....|....4....|....5....|....6....|....7
```

Input:

0.5	alpha = hwidth/P in revcnc
0.66	z = a/b
14.5	ptrns in kbar
0.013	volume change of transformation
5.0	pmin in kbar
23.0	pmax in kbar
0.25	dp in kbar
0.0	dK/dP in phase 2 beyond 17.5 kbar
3.0	maximum halfwidth of distribution, kbar (H)

INTERNAL REPORTS - 1990

1. J.B. Aidun, "Notes on the New Controller for the Anac Two-Axis Magnet", Internal Report 90-01, May, 1990.

2. X.A. Shen, "Efficiency of Stimulated R-Line Emission in Ruby", Internal Report 90-02, June, 1990.

3. X.A. Shen, "Use of Fast Fourier Transform for the Analysis of Ruby R-Line Spectrum", Internal Report 90-03, July, 1990.

4. G.E. Duvall, "Pressure Buildup at a Shock Front Which Initiates an Exothermic Reaction", Internal Report 90-04, August, 1990.

5. X.A. Shen, "Data Reduction in Ruby Emission Experiments", Internal Report 90-05, September, 1990.

6. R. Gustavsen, "Characterization of the WSU Shock Dynamics Laboratory's Time Resolved Raman Spectroscopy System", Internal Report 90-06, September, 1990.