strs3d - 3D Stress Transformations

A Fortran code for three-dimensional stress transformations as outlined in Module 10 is listed below. A PC-executable version is also available in the readings section, which can be saved to disk and run as a console program.

Fortran Source

```
c
c                     s t r s 3 d
c
c     an instructional code for transformations and principal values
c     of three-dimensional stress states
c
      dimension sig(3,3),sigp(3),temp(3,3),signew(3,3),
     1   t(3,3),tt(3,3)
      common /inv/a(4)
      data  rad/57.29578/
c
c     read stress matrix
c
   10 write (6,15)
   15 format (5x,'enter stress matrix by rows'/)
      read (5,*) ((sig(i,j),j=1,3),i=1,3)
c
c     read branch option
c
   20 write (6,30)
   30 format (/5x,'enter problem option:',
     1        /7x,'1 - transformation',
     2        /7x,'2 - principal values',
     3        /7x,'3 - restart',
     4        /7x,'4 - stop'/)
      read (5,*) ibr
      go to (100,200,10,35),ibr
c
   35 stop 'program terminated'
  100 continue
c
c**********************************************************************
c
c     compute transformed stresses
c
c**********************************************************************
c
      write (6,120)
  120 format (/5x,'enter euler angles psi, theta, phi (degrees):'/)
      read (5,*) psi, theta, phi
c
c     compute transformation matrix t(i,j)
c
      spsi=sin(psi/rad)
```

```
      cpsi=cos(psi/rad)
      stheta=sin(theta/rad)
      ctheta=cos(theta/rad)
      sphi=sin(phi/rad)
      cphi=cos(phi/rad)
c
      t(1,1)=cpsi*cphi-sphi*ctheta*spsi
      t(2,1)=-sphi*cpsi-spsi*ctheta*cphi
      t(3,1)=stheta*spsi
      t(1,2)=cphi*spsi+sphi*ctheta*cpsi
      t(2,2)=-sphi*spsi+cphi*ctheta*cpsi
      t(3,2)=-stheta*cphi
      t(1,3)=stheta*sphi
      t(2,3)=stheta*cphi
      t(3,3)=ctheta
c
c     inverse (transpose) transformation matrix tt(i,j)
c
      do 130 i=1,3
      do 130 j=1,3
  130 tt(i,j)=t(j,i)
c
c     compute transformed stresses signew = t*(sig*tt) = t*temp
c
      call mult (sig,tt,3,3,3,temp)
      call mult (t,temp,3,3,3,signew)
c
c     print results, then branch for new option
c
      write (6,140)
  140 format (//1x,'transformation matrix:',18x,'stress:'/)

      do 150 i=1,3
  150 write (6,160) (t(i,j),j=1,3),(signew(i,j),j=1,3)
  160 format (1x,3f10.4,10x,3f10.4)
c
      go to 20

c
c*********************************************************************
c
c     compute principal values and directions of principal planes
c
c*********************************************************************
c
c     compute invariants
c
  200 a(1)=1.
      a(2)=-1.*(sig(1,1)+sig(2,2)+sig(3,3))
      a(3)= sig(1,1)*sig(2,2) + sig(1,1)*sig(3,3) + sig(2,2)*sig(3,3)
     1     -sig(1,2)*sig(1,2) - sig(2,3)*sig(2,3) - sig(1,3)*sig(1,3)
      a(4)=-1.*(sig(1,1)*(sig(2,2)*sig(3,3) - sig(3,2)*sig(2,3))
     1         -sig(1,2)*(sig(2,1)*sig(3,3) - sig(3,1)*sig(2,3))
     2         +sig(1,3)*(sig(2,1)*sig(3,2) - sig(3,1)*sig(2,2)))
c
c     compute principal stresses as roots of cubic equation
c
```

```
c
c     interactive evaluation of characteristic equation
c
  205 write (6,210)
  210 format (/5x,'enter solution option:',
     1        /7x,'1 - evaluate characteristic equation',
     2        /7x,'2 - newton-raphson interation',
     3        /7x,'3 - compute direction cosines',
     4        /7x,'4 - return for new problem option'/)
      read (5,*) ibr
      go to (215,220,245,20),ibr
c
c
  215 write (6,216)
  216 format (' Enter sigma (999 to stop)'/)
      read (5,*) sigg
      if (sigg.eq.999) go to 205
      ysg=ysig(sigg)
      write (6,217)  sigg,ysg
  217 format (' y(',g12.4,') = ',g12.4/)
      go to 215
c
c     Newton-Raphson iteration to refine roots
c
  220 kmax=10
  222 write (6,223)
  223 format (' Enter initial guess (999 to stop)'/)
      read (5,*) sigpi
      if (sigpi.eq.999) go to 205
         do 230 k=1,kmax
            ysg=ypsig(sigpi)
            if (ysg.ne.0.) go to 226
               write (6,225)
  225          format (' Zero slope - try another guess')
               go to 222
  226       delta=ysig(sigpi)/ysg
            sigpi=sigpi-delta
             if (sigpi.ne.0) go to 228
               if (delta.lt.0.01) go to 240
               go to 230
  228       if (abs(delta/sigpi).lt.0.01) go to 240
  230    continue
         write (6,*) ' No convergence for root'
         go to 222
  240 write (6,241) sigpi,k
  241 format (' root =',g12.4,' (',i3,' iterations)')
      go to 222
c
c
c     compute direction cosines and print results
c
  245 write (6,*) ' Enter 3 principal stresses'
      read (5,*) sigp
      write (6,250)
  250 format (//5x,'stress',12x,'l',9x,'m',9x,'n'/)
c
      do 270 i=1,3
```

```
c
        a2=(sig(2,2)-sigp(i))*(sig(3,3)-sigp(i))-(sig(2,3)*sig(2,3))
        b= sig(1,2)*(sig(3,3)-sigp(i))-(sig(1,3)*sig(2,3))
        c=(sig(1,2)*sig(2,3))-sig(1,3)*(sig(2,2)-sigp(i))
c
        akk=sqrt (a2*a2 + b*b + c*c)
        ak=0.
        if (abs(akk).gt. 0.001) ak=1./akk
c
        al=a2*ak
        am=b*ak
        an=c*ak
c
        write (6,260) sigp(i),al,am,an
  260   format (1x,f10.4,5x,3f10.4)
  270 continue
c
      go to 20
      end
c
c
      function ysig(sig)
      common /inv/ a(4)
      ysig=a(1)*sig**3+a(2)*sig*sig+a(3)*sig+a(4)
      return
      end
c
c
      function ypsig(sig)
      common /inv/ a(4)
      ypsig=3.*a(1)*sig*sig+2.*a(2)*sig+a(3)
      return
      end
c
c
      subroutine mult (a,b,l,m,n,c)
c
c     returns the matrix product c = a * b
c
      dimension a(l,m),b(m,n),c(l,n)
c
      do 20 i=1,l
      do 20 j=1,n
        cc=0.
        do 10 k=1,m
            cc=cc+a(i,k)*b(k,j)
   10   continue
   20 c(i,j)=cc
      return
      end
```