```
      program stat2pot
c program converts CHAMP or GOCE state-vectors to "potential" /kinetic
c energy.
c programmed 2001-10-09 by CCT/eva, last update: 2004-06-04
c
c input of lchamp, true if CHAMP and false if GOCE.
c input for CHAMP:
c  name of file with accelerations (output from readacc). 10 s sampling.
c   and attitude angles. If mode=3 (see below) the accelerations are
c   in the inertial coordinate system, otherwise in the instrument frame.
c  name of file with state-vector, 30 s sampling.
c  name of file with output data (one gravsoft ready).
c  value of pole position
c  mode (=1 no correction, = 2 correction for integral(velocity*
c     acc(y)*30(s)), = 3 correction for integral (vecocity vector
c     inner product acceleration vector * 30 sec.).
c input for GOCE:
c  maximal number of records - maxrec
c  name of file with state-vector.
c  name of file with output data (one gravsoft ready).
c
c output:
c  for first 4 records, all parmaters.
c***********************************************************************
      implicit none
      integer imax,i,k,mode,itime,maxrec,imode
      parameter(imax=257000)
      logical lchamp
      real*8 x,y,z,vx,vy,vz,v2,v0,pi,pot,j2,GM,epot,lx2,lx0
c
c pot is the kinetic energy, epot is the normal potential
c of the Earth without centrifucal terms.
c
      real*8 ax2,e22,vex,vey,p,w,d0,d1,d2,radeg,azim,dazim,head,
     *       dhead,dazimma,dheadma,dazimmi,dheadmi,sdazim,ssdazim,
     *       sdhead,ssdhead,gmean,gss,gmin,gmax
c ax2=semi major axis (m), e22=second excentricity,p distance from Z-axis,
c vex, vey velocity vector of the Earth seen from inertial system,
c w=omega=rotational velocity (rad/sec).
      real*8 g,Vr,Vla,cosla,sin2la,j4,j6,pmean,dif(imax,25)
      real*8 xp,yp,A(3,3),V(3),tide
      real*8 time, psu,pmo,tig,gmo,gs,fpot,fpot0,sfpot,dpot,dpot0
      real*8 vxi,vyi,vzi,Jday,vxc,vyc,vzc,vxa,vya,vza
      real*8 t,GMST,Om,L,L1,dp,de,GAST,att(4),st,std
      real*8 meanx,meany,sxy,s12,k1,theta,phi,psi
      real*8 corr,scorr,ssa,sa,ssb,sb,rsec_i
      real*8 spot,sspot,spsu,sspsu,spmo,sspmo,sepot,ssepot,sdpot,
     *ssdpot,cpot,scpot,sscpot,data(11)
c
      real*8 sinlap,coslap,rlatp,sinlop,coslop,rlongp,hp
c cos and sin of geodetic latitude and longitude and ellipsoidal
c height.
      integer*8 ix,iy,iz,ivx,ivy,ivz,irec

      integer*4 day, sec_i
c variables to read header for CHAMP orbit format.
      character*72 ifila,ifile,ofile,ofilg
      character*5 orbit
      COMMON /ITRANC/ax2,e22
c
c constants:
      ax2=6378136.42d0
```



```
      e22=0.006694380d0
c coefficients of Earth-potential.
      j2=1082.63d-6
      j4=-2.371d-6
      j6=0.006d-6
      GM=398600.5d9
c revolution speed.
      w=7.292115d-5
c constants
      pi=4.0d0*atan(1.0d0)
      d1=1.0d0
      d0=0.0d0
      d2=2.0d0
      radeg=180.0d0/pi
c initialisations.
      dazimma=-1.0d6
      dazimmi= 1.0d6
      dheadma=-1.0d6
      dheadmi= 1.0d6
      gmean=d0
      gss=d0
      gmax=-1.0d6
      gmin= 1.0d6
      spot=d0
      sspot=d0
      spsu=d0
      sspsu=d0
      spmo=d0
      sspmo=d0
      sepot=d0
      ssepot=d0
      sdpot=d0
      ssdpot=d0
      scpot=d0
      sscpot=d0
c
      write(*,*)' Program stat2pot, ver. 2004-07-08. '
c
      write(*,*)' input t for CHAMP and f for GOCE '
      read(*,*)lchamp
      write(*,*)lchamp
      if (lchamp) then
       write(*,*)' Input names of output from readacc, '
       write(*,*)' CHAMP orbit file and 2 output file (four lines) '
       read(*,'(a)')ifila
       read(*,'(a)')ifile
       read(*,'(a)')ofile
       read(*,'(a)')ofilg
       write(*,17)ifila,ifile,ofile,ofilg
 17    format(' 2 Input and  2 output files: ',/,4(1x,A72/))
      else
       write(*,*)' input maximal number of records '
       read(*,*)maxrec
       write(*,*)maxrec
       write(*,*)' Input name of GOCE statevector file, '
       write(*,*)' output file (2 lines) '
       read(*,'(a)')ifile
       read(*,'(a)')ofilg
       write(*,16)ifile,ofilg
 16    format(' Input and  output files: ',/,4(1x,A72/))
      end if
c
```

```
      write(*,*)' Input pole position xp and yp in  (two lines) '
      read(*,*) xp
      read(*,*) yp
      write(*,18)xp,yp
 18   format(' Pole-position ',2f10.3)
c
      imode=0
      if (lchamp) then
       write(*,*)' input value of mode; = 1, no correction '
       write(*,*)' = 2, int(v*ay*dt), = 3, int(v*va*dt) '
       read(*,*)mode
       if (mode.gt.3) then
        write(*,*)' mode not implemented '
        mode=1
       end if
      else
       write(*,*)' input input-data mode ',imode
       read(*,*)imode
       write(*,*)' input mode= ',imode
c for imode=1 we have end-to-end simulator data
c for imode = 2 we have IAG SC/ data (in Earth fixed frame).
       mode=1
      end if
      write(*,*)' mode = ',mode
      irec=0
c
      if (lchamp) open(11,file=ifila)
c     open(11,file='goce999.inp')
      open(12,file=ifile)
      open(15,file=ofilg)
      if (lchamp) then
       open(14,file=ofile)
c
c read header, stop at 'ORBIT'
c
 100   read(12,*)orbit
       irec=irec+1
       if (orbit.ne.'ORBIT') go to 100
      else
c day is 2006-10-10.0.0.00 minus J2000 in days.
c first obs is at 47361.0d0/864000.0d0
       day=54023.0d0
      end if
c
c initializing counters and variables for accumulation.
      irec=0
      pmean=0.0d0
      fpot=0.0d0
      sfpot=d0
      meanx=0.0d0
      meany=0.0d0
      sxy=0.0d0
      s12=0.0d0
      corr=0.0d0
      scorr=0.0d0
      sa=0.0d0
      ssa=sa
      sb=sa
      ssb=sa
      sdazim=d0
      ssdazim=d0
      sdhead=d0
```



```
      ssdhead=d0
      write(*,*)' first 5 records (time,position and velocity).'
c
c read CHAMP state-vector records.
 200  if (lchamp) then
       read(12,10,end=999)day,sec_i,ix,iy,iz,ivx,ivy,ivz
       time=day/10.0d0+(sec_i*1.0d-6-64184d-3)/86400.0d0
       if (irec.lt.5) write(*,95)time,sec_i,ix,iy,iz,ivx,ivy,ivz
  95   format(f14.5,4i16,/,16x,4i16)
      else
c GOCE.
       if (irec.gt.maxrec) go to 999
c time is seconds after 2006-10-10:0.0.00, coordinates and velocity
c in IRS.
       if (imode.eq.1) then
        read(12,*,end=999)itime,x,y,z,vx,vy,vz
        time=day+itime/86400.0d0
       else
        read(12,*,end=999)time,x,y,z,vx,vy,vz,(data(k),k=1,9)
        time=time-51740.5d0
        st=time*86400.0d0*7.29211585531d-5+5.133658456d0
        std=st*180.0d0/pi
        call r180(std)
        time=irec+1
       end if
       irec=irec+1
       dif(irec,21)=time
       if (irec.lt.5) write(*,91)itime,x,y,z,vx,vy,vz
 91    format(i8,3f13.2,3f10.3)
c      if (mod(irec,10).eq.0) write(11,91)itime,x,y,z,vx,vy,vz
      end if
  10  format(i6,i11,6i12)
c
      if (lchamp) then
c Read the accelerations from the output file of readacc.
c in inertial frame.
       read(11,*,end=999)Jday,vxc,vyc,vzc,theta,phi,psi
c the acceleration file contains 10 s data, while we only
c use 30 s data. We therefor skip data as below.
c only when Jday=time
 300   if ((Jday-time).gt.0.00005d0) then
        read(12,27,end=999)day,sec_i,ix,iy,iz,ivx,ivy,ivz
  27    format(i6,i11,6i12)
        time=day/10.0d0+(sec_i*1.0d-6-64184d-3)/86400.0d0
        go to 300
       end if
       if ((time-Jday).gt.0.00005d0) then
c Read the accelerations and Euler-attitude angles
c from the accelerometer.
        read(11,*,end=999)Jday,vxc,vyc,vzc,theta,phi,psi
        go to 300
       end if
       if (ABS(time-Jday).le.0.00005d0) then
        if (irec.lt.5)write(*,*)' time,Jday ',time,Jday
        irec=irec+1
        dif(irec,9)=time
c
c The time marker for the outfiles is calculated in messurements after 1.1.
        dif(irec,21)=(((day+5)/10.d0-365)*24*3600
     .  +(sec_i*1.0d-6-64184d-3))/30
c
c conversion to m.
```

```
        x=ix*1.0d-3
        y=iy*1.0d-3
        z=iz*1.0d-3
        vx=ivx*1.0d-7
        vy=ivy*1.0d-7
        vz=ivz*1.0d-7
       end if
      end if
c
c Calculation of the normal earth potential without centrifucal
c terms from the cordinates
c
      lx2=(x**2+y**2+z**2)
      lx0=sqrt(lx2)
      sin2la=z**2/lx2
      epot=(GM/sqrt(lx2))*(1-(j2*(ax2**2/lx2)*((1.5d0*sin2la)-0.5d0)
     .+j4*(ax2**4/lx2**2)*(((35.0/8.0)*sin2la**2)-((30.0/8.0)*sin2la)
     .+(3.0/8.0))+j6*(ax2**6/lx2**3)*(((231.0/16.0)*sin2la**3)
     .-((315.0/16.0)*sin2la**2)+((105.0/16.0)*sin2la)-(5.0/16.0))))
C
C Calculation of normal gravity.
C first the radial and spherical component and then the size
C
      cosla=sqrt((x**2+y**2)/lx2)
      Vr=(GM/lx2)*(-1-(3.0*j2*(ax2**2/lx2)*((1.5d0*sin2la)-0.5d0)
     .-5.0*j4*(ax2**4/lx2**2)*(((35.0/8.0)*sin2la**2)
     .-((30.0/8.0)*sin2la)
     .+(3.0/8.0))-7.0*j6*(ax2**6/lx2**3)*(((231.0/16.0)*sin2la**3)
     .-((315.0/16.0)*sin2la**2)+((105.0/16.0)*sin2la)-(5.0/16.0))))
c
      Vla=(GM/lx2)*((ax2**2/lx2)*j2*3*cosla*sqrt(sin2la)+j4
     .*(ax2**4/lx2**2)*((35.0/2.0)*cosla*sin2la**1.5-(30.0/4.0)
     .*cosla*sqrt(sin2la))+j6*(ax2**6/lx2**3)*((6.0*231.0/16.0)
     .*cosla*sin2la**2.5-(315.0/4.0)*cosla*sin2la**1.5+(105.0/8.0)
     .*cosla*sqrt(sin2la)))
c
      g=sqrt(Vr**2+Vla**2)
      cpot=w**2/d2*(x**2+y**2)
      dif(irec,25)=cpot
      gmean=gmean+g
      gss=gss+g**2
      if (g.gt.gmax) gmax=g
      if (g.lt.gmin) gmin=g
      if (irec.lt.5) write(*,66)epot,g
  66  format(' potential ',d17.9,' m**2/s**2 ,gravity ',
     *f10.7)
      dif(irec,10)=g
c
c transformation from cartesian to geodetic coordinates.
c
      call TRANS(sinlap,coslap,rlatp,sinlop,coslop,rlongp,hp,
     *x,y,z)
      if ((.not.lchamp).and.imode.eq.2) rlongp=rlongp-st
      rlongp=rlongp*180.0/pi
      call r180(rlongp)
c
      dif(irec,1)=rlongp
      dif(irec,2)=rlatp*180.0/pi
      dif(irec,3)=hp
      if (hp.lt.20000.0) then
       write(*,*)' hp ',irec,hp,x,y,z
      end if
```



```
      dif(irec,12)=theta
      dif(irec,13)=phi
      dif(irec,14)=psi
c Save the accelerations as spacecraft system
      dif(irec,17)=vyc
      dif(irec,18)=-vzc
      dif(irec,19)=-vxc
c
c Calculation of the kinetic energy including transformation to CIS.
c it is used that sin(xp) almost is egual to xp for small angels.
c First the earth rotation term
      p=sqrt(x**2+y**2)
      vex=-p*w*sinlop
      vey=p*w*coslop
      t=time/36525
c Greenwich mean siderial time.
      GMST= mod(280.46061837d0+360.98564736629d0*time,360.0d0)
      Om=mod(125.04452d0-1934.136261d0*t,360.0d0)
      L=mod(280.4665d0+36000.7698d0*t,360.0d0)
      L1=mod(218.3465d0+481267.8813d0*t,360.0d0)
      dp=(-17.2d0*sin(Om*pi/180.0)-1.32d0*sin(2*L*pi/180.0)
     .-0.23d0*sin(2*L1*pi/180.0)+0.21d0*sin(2*Om*pi/180.0))/3600d0
      de=(9.2d0*cos(Om*pi/180.0)+0.57d0*cos(2*L*pi/180.0)
     .+0.1d0*cos(2*L1*pi/180.0)-0.9d0*cos(2*Om*pi/180.0))/3600d0
c Greenwich  apparent siderial time.
      GAST=GMST+dp*cos(de*pi/180.0)
      If ((.not.lchamp).and.imode.eq.1) dif(irec,1)=dif(irec,1)+GAST
c
c prepare rotation matrix.
      A(1,1)=cos(GAST*pi/180.0)
      A(2,2)=cos(GAST*pi/180.0)
      A(3,3)=1.0
      A(1,2)=-sin(GAST*pi/180.0)
      A(2,1)=sin(GAST*pi/180.0)
      A(1,3)=-cos(GAST*pi/180.0)
     .*(xp/3600d0)*pi/180.0-sin(GAST*pi/180.0)
     .*(yp/3600d0)*pi/180.0
      A(3,1)=(xp/3600d0)*pi/180.0
      A(2,3)=cos(GAST*pi/180.0)
     .*(yp/3600d0)*pi/180.0-sin(GAST*pi/180.0)
     .*(xp/3600d0)*pi/180.0
      A(3,2)=(-yp/3600d0)*pi/180.0
      V(1)=vx+vex
      V(2)=vy+vey
      V(3)=vz
      if (lchamp) then
C Transformation to CIS
       call AXV(A,V)
       vxi=V(1)
       vyi=V(2)
       vzi=V(3)
       v2=(vxi**2+vyi**2+vzi**2)
      else
       if (imode.eq.5) then
        vx=vx+vex
        vy=vy+vey
       end if
       v2=((vx)**2+(vy)**2+vz**2)
      end if
      v0=sqrt(v2)
      if (irec.lt.5) write(*,59)GAST,v0
  59  format(' GAST ',D15.8,' v ',f10.3, ' m/s ')
```

```
c
      dif(irec,22)=jday
      dif(irec,23)=lx0
      dif(irec,24)=v0

c
      if (lchamp) then
       azim=atan2(vyi,vxi)*radeg
       head=atan2(vzi,sqrt(vxi**2+vyi**2))*radeg
       dazim=azim-psi
       if (abs(dazim).gt.10.0) then
        if (dazim.gt.300.0) dazim=azim+psi
        if (dazim.lt.-300.0) dazim=-azim-psi
        if (abs(dazim).gt.10.0) write(*,98)irec,azim,psi,dazim
  98    format(i6,3f8.2)
       end if
       dif(irec,16)=dazim
       dhead=head+theta
       dif(irec,20)=dhead
       sdhead=sdhead+dhead
       ssdhead=ssdhead+dhead**2
       sdazim=sdazim+dazim
       ssdazim=ssdazim+dazim**2
       if (dazim.gt.dazimma) dazimma=dazim
       if (dazim.lt.dazimmi) dazimmi=dazim
       if (dhead.gt.dheadma) dheadma=dhead
       if (dhead.lt.dheadmi) dheadmi=dhead
       if (irec.lt.5) then
        write(*,96)irec,azim,head,theta,phi,psi
 96     format(i5,5f10.3)
       end if
      end if
c
c kinetic energy, m**2/s**2.
      if (lchamp) then
       pot=0.5d0*v2
      else
       pot=(vx**2+vy**2+vz**2)/d2
      end if
c
      if (lchamp) then
c Calculate the potential of the friction force. The factor 30 occurs
c because we have one observation per 30 seconds.
       if (mode.eq.3) then
        fpot0=(vxi*vxc+vyi*vyc+vzi*vzc)*30.0
        if (irec.le.5)
     *  write(*,57)irec,vxi,vyi,vzi,vxc*1000,vyc*1000,vzc*1000,
     *  fpot0
   57   format(i7,3f9.1,3f7.3,f12.1)
        fpot=fpot+fpot0
       else
        fpot0=v0*vyc*30.0d0
        fpot=fpot+fpot0
        if (irec.le.5)write(*,*)' v,vyc,fpot0 ',v0,vyc,fpot0,fpot
       end if
       sfpot=sfpot+fpot
       dif(irec,11)=fpot0
       dif(irec,15)=fpot
       sa=sa+fpot0
       ssa=ssa+fpot0**2
      else
       fpot=d0
```



```
       sfpot=d0
       sa=d0
       ssa=d0
      end if
c
c Calculate the rotation potential using equation (17) from C.Jekeli,
c The determination of gravitational potential differences from sat.-to-sat
king,1999
      if (lchamp) then
       dpot=w*(x*(vy+vey)-y*(vx+vex))
      else
       dpot=w*(x*(vy)-y*(vx))
      end if
      if (irec.lt.5) write(*,*)' dpot,pot ',dpot,pot
c
c calculate potential contribution of Sun and Moon.

      tig=tide(dif(irec,9),dif(irec,2),dif(irec,1),dif(irec,3),
     .psu,pmo,gmo,gs)
c
c all units  m**2/s**2
       pmean=pmean+pot-psu-pmo-epot-dpot
       dif(irec,6)=pot-pmo-psu-epot-dpot
       if (irec.lt.5) write(*,*)' pot-all ',dif(irec,6)
      spot=spot+pot
      sspot=sspot+pot**2
      spsu=spsu+psu
      sspsu=sspsu+psu**2
      spmo=spmo+pmo
      sspmo=sspmo+pmo**2
      sepot=sepot+epot
      ssepot=ssepot+epot**2
      sdpot=sdpot+dpot
      ssdpot=ssdpot+dpot**2
      scpot=scpot+cpot
      sscpot=sscpot+cpot**2
      dif(irec,4)=pot
      dif(irec,5)=epot
      dif(irec,7)=fpot
      dif(irec,8)=dpot
      if (irec.eq.imax) then
       write(*,*)' number of data too large '
       go to 999
      end if
      go to 200
c
  999 sfpot=sfpot/irec
      pmean=pmean/irec
      sspot=sqrt((sspot-spot**2/irec)/(irec-1))
      spot=spot/irec
      sspsu=sqrt((sspsu-spsu**2/irec)/(irec-1))
      spsu=spsu/irec
      sspmo=sqrt((sspmo-spmo**2/irec)/(irec-1))
      spmo=spmo/irec
      ssepot=sqrt((ssepot-sepot**2/irec)/(irec-1))
      sepot=sepot/irec
      ssdpot=sqrt((ssdpot-sdpot**2/irec)/(irec-1))
      sdpot=sdpot/irec
      sscpot=sqrt((sscpot-scpot**2/irec)/(irec-1))
      scpot=scpot/irec
      write(*,92)spot,sspot,spsu,sspsu,spmo,sspmo,
     *sepot,ssepot,sdpot,ssdpot,scpot,sscpot
```

```
 92   format(' type      mean stdv. ',/,
     *' pot  ',2d15.7,/,
     *' psu  ',2d15.7,/,
     *' pmo  ',2d15.7,/,
     *' epot ',2d15.7,/,
     *' cpot ',2d15.7,/
     *' dpot ',2d15.7,/)
c Calculate the trend. using regression.
      do i=1,irec
       meany=meany+dif(i,6)
       meanx=meanx+i
      end do
      meanx=meanx/irec
      meany=meany/irec
      do i=1,irec
       sxy=sxy+(dif(i,6)-meany)*(i-meanx)
       s12=s12+(i-meanx)**2
      end do
      sxy=sxy*1.0d0/(irec-1.0d0)
      s12=s12*1.0d0/(irec-1.0d0)
      k1=sxy/s12
      if (lchamp) write(*,20)sxy,s12,k1
  20  format(' sxy, s12, slope (m**2/s**2/30s) ',/,3d12.4)
      do i=1,irec
c calculate potential contribution of Sun and Moon.

      tig=tide(dif(i,9),dif(i,2),dif(i,1),dif(i,3),psu,pmo,gmo,gs)
c
c output in GMT format (long, lat, data) and GRAVSOFT format.
c trend is being subtracted.
       if (lchamp) then
        write(14,50)dif(i,22),dif(i,23),dif(i,24),dif(i,8)
   50   format(f13.8,f16.6,2f14.6)
        write(15,55)
     *  nint(dif(i,21)),dif(i,2),dif(i,1),dif(i,3),
     *  ((dif(i,6)-k1*(i-meanx)-pmean)/dif(i,10)),dif(i,11)
C HER  ER SLETTET NOGET
       else
c output of the height anomaly equal to potential minus mean and
c tilt contribution divided by gravity.
        if (imode.eq.1) then
         itime=(dif(i,21)-day)*86400.0d0
        else
         itime=dif(i,21)
        end if
        write(15,55)
     *  itime,dif(i,2),dif(i,1),dif(i,3),
     *  ((dif(i,6)-k1*(i-meanx)-pmean)/dif(i,10)),(dif(i,6)-pmean)
     *  /dif(i,10),(dif(i,4)-spot)/dif(i,10),
     *  (dif(i,5)-sepot)/dif(i,10),(dif(i,8)-sdpot)/dif(i,10)
       end if
c sfpot is the mean of the friction-energy integral contribution.
  55   format(i10,f10.5,f11.5,f10.2,f13.3
     *  ,f13.3,f13.3,2f13.3)
      scorr=scorr+(dif(i,6)-k1*(i-meanx)-pmean)/dif(i,10)*(dif(i,11)
     * -sa/irec)
      sb=sb+(dif(i,6)-k1*(i-meanx)-pmean)/dif(i,10)
      ssb=ssb+((dif(i,6)-k1*(i-meanx)-pmean)/dif(i,10))**2
      end do
c
       write(*,60)pmean,meanx,k1
  60   format(' pmean ',D16.8,' meanx ',f10.1,', slope= ',D12.5)
```



```
       write(*,*)' number of records written ',irec
c
       if (lchamp) then
        write(*,*)' Correlation between friction and heights. '
        write(*,*)' corr  mean fric. std.  mean h  stdv. '
        ssa=sqrt((ssa-sa**2/irec)/(irec-1))
        ssb=sqrt((ssb-sb**2/irec)/(irec-1))
        sa=sa/irec
        sb=sb/irec
        corr=scorr/(ssa*ssb*irec)
        write(*,65)corr,sa,ssa,sb,ssb
  65    format(5d12.5)
c
        ssdazim=sqrt((ssdazim-sdazim**2/irec)/(irec-1))
        ssdhead=sqrt((ssdhead-sdhead**2/irec)/(irec-1))
        sdazim=sdazim/irec
        sdhead=sdhead/irec
        write(*,97)sdazim,ssdazim,dazimmi,dazimma,
     *  sdhead,ssdhead,dheadmi,dheadma
  97    format(' mean, stdv. min and max diff azimuth and heading ',
     *  /,4f8.2,/4f8.2,/)
       end if
c
       gss = sqrt((gss-gmean**2/irec)/(irec-1))
       gmean=gmean/irec
       write(*,93)gmean,gss,gmin,gmax
  93   format(' mean , stdv, min, max gravity ',/,4f10.7)
c
      stop
      end
c
      SUBROUTINE TRANS(SINLAP,COSLAP,RLATP,SINLOP,COSLOP,RLONGP,HP,
     *X,Y,Z)
C ORIGINAL VERSION PROGRAMMED IN 1974 BY C.C.TSCHERNING, GEODAETISK
C INSTITUT. LATEST UPDATE 2001-12-05.
C
C THE SUBROUTINE TRANSFORMS THE CARTESIAN COORDINATES
C TO GEODETIC.
C
C IF DOUBLE PRECISION IS NEEDED ACTIVATE:
      IMPLICIT INTEGER(I,J,K,N,M),LOGICAL(L),REAL *8(A-H,O-Z)
C AND USE DSIN, DCOS, DATAN2, DABS BELOW.
      COMMON /ITRANC/AX2,E22
      PI=3.1415926535d0
      D0=0.0D0
      D1=1.0D0
      X0=X
      Y0=Y
      Z0=Z
      XY20= X0*X0+Y0*Y0
      XY0 =  SQRT(XY20)
      DIST20 = XY20+Z0*Z0
      DISTO0 =  SQRT(DIST20)
C LONGITUDE.
      RLONG =  ATAN2(Y0,X0)
C
C  COMPUTATION OF THE NEW GEODETIC LATITUDE, CF REF(C) PAGE 183.
      S = AX2
      DH = D0
      RLAT1 = D0
      COSLA=1.0D0
   70 RLAT = RLAT1
```

```
C
      RLAT1 =  ATAN2(Z0,XY0-E22*S*COSLA)
      COSLA =  COS(RLAT1)
      S = AX2/ SQRT(D1-E22*(D1-COSLA**2))
      DH = XY0/COSLA-S
      IF ( ABS(RLAT1-RLAT).GT.1.0D-10) GO TO 70
C
      RLONGP = RLONG
      if (rlongp.gt.2*pi)write(*,*)' warning long ',rlong
      RLATP = RLAT1
      SINLOP= SIN(RLONG)
      COSLOP= COS(RLONG)
      COSLAP=COSLA
      SINLAP= SIN(RLATP)
      HP=DH
C
      RETURN
      END
C
      SUBROUTINE AXV(A,V)
C THE SUBRIUTINE WILL COMPUTE THE PRODUCT OF THE MATRIX A AND THE
C VECTOR V AND RETURN IT IN V. PROGRAMMED 1990.11.03 BY CCT.
      IMPLICIT NONE
      INTEGER I,J
      REAL*8 A(3,3),V(3),Y(3)
      DO 10 I=1,3
      Y(I)=V(I)
   10 V(I)=0.0D0
      DO 20 I=1,3
      DO 20 J=1,3
   20 V(I)=A(I,J)*Y(J)+V(I)
      RETURN
      END
c
      real*8 function tide(time,lat,lon,height,psu,pmo,gm,gs)
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
c                      t i d e
c
c  the procedure compute the tidal acceleration due to
c  the moon and the sum.
c
c  Parameters.
c
c  tide     (return value). the tidal acceleration corresponding
c                           to a rigid earth in mgal.
c  psu      (   -     -  ). the potential of the sun (m**2/s**2).
c
c  pmo      (   -     -  ). the potential of the moon (m**2/s**2)
c
c  gs       (   -     -  ). the attraction of the sun (gal).
c
c  gm       (   -     -  ). the attraction of the moon (gal).
c
c  time     (call   value). time in days from epoch 2000,
c                           i.e. Greenwich mean noon on December
c                           31, 1999.
c  latitude (   -     -  ). Terrestrial latitude of point of computa-
c                           tion, degrees.
c  longitude(   -     -  ). Terrestrial longitude of point of
c                           computation, positive east, degrees.
c  height   (   -     -  ). Height above sea level (or above ellip-
```



```
c                            soid) of point of computation, meter.
c
c  Reference:
c
c  I.M.Longman: Tidal accelerations due to the moon and the sun.
c  Journal of Geophysical Research, Volume 64, No 12, December 1959.
c
c  Explanatory supplement to
c  The Astronomical Ephemeris and
c  The American Ephemeris and Nautical Almanac.
c  Her Majestys Stationery Office, 1961
c
c  algol program by Willy Weng, 1976.
c  fortran version jan 1992, Rene Forsberg
c  (c) KMS, modified 2001-10-26 by cct/eva
c
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      implicit none
      real*8 lat,lon,time,height,psu,pmo
      real*8 omega, N, sinii, cosii, pi,
     .alfa, l, sigma, coslambda, sinlambda, cosphi,
     .tt, tt2, tt3, r, e, e1, p, h, c, c1, aprim, a1prim,
     .resd, resdd, i, ii, m, mm, ss, my, ksi, ksi1, ll,
     .p1, ny, t, l1, costheta, g0, gm, gs, s
c
      pi     = 3.1415926536d0
      e      = 0.054899720d0
      c      = 3.84402d10
      c1     = 1.495d13
      aprim  = 1/(c*(1-e*e))
      i      = 0.08979719
      omega  = 0.4093146162
      ss      = 1.993d33
      mm      = 7.3537d25
      my     = 6.670d-8
      m      = 0.074804d0
c
c  computation point
c
      coslambda  = cos(lat/180*pi)
      sinlambda  = sin(lat/180*pi)
      r      = 6.378270d8/sqrt(1+0.006738d0*sinlambda**2)
     .          + height * 100
      ll      = lon/180*pi
c
c  Julian centuries and series in tt,
c  Longman (10), (11), (12), (19), (26) and (27).
c
c  Sun:
c
c  Longman     Exp. Sup (p.98)
c  h  (12)     ll
c  p1 (26)     GAMMA
c  e1 (27)     e
c
      tt  = time/36525+1.0d0
      tt2 = tt*tt
      tt3 = tt2*tt
      s   = 4.720023438 + 8399.7093*tt + 4.40695d-5*tt2 + 3.29d-8*tt3
      p   = 5.835124721 + 71.018009*tt - 1.80546d-4*tt2 - 2.181d-7*tt3
      h   = 4.881627934 + 628.33195*tt + 5.2796 d-6*tt2
      N   = 4.523588570 - 33.757153*tt + 3.67488d-5*tt2 + 3.87 d-8*tt3
```

```
      p1  = 4.908229467 + 3.0005264d-2*tt + 7.9024d-6*tt2 + 5.81d-8*tt3
      e1  = 0.01675104  - 4.18d-5 *tt - 1.26d-7 * tt2
c
c  reciproc distances
c
      a1prim= 1/(c1*(1-e1*e1))
      resd  = 1/c + aprim*e*cos(s-p)
     .        + aprim*e*e*cos(2*(s-p))
     .        + 15d0/8*aprim*m*e*cos(s-2*h+p)
     .        + aprim*m*m*cos(2*(s-h))
      resdd  = 1/c1 + a1prim*e1*cos(h-p1)
c
c  longitude of moons ascending node
c
      cosii  = cos(omega)*cos(i)
     .        - sin(omega)*sin(i)*cos(N)
      sinii  = sqrt(1-cosii**2)
      ii     = atan(sinii/cosii)
      ny    = asin(sin(i)*sin(N)/sinii)
c
c  longitude and rigth ascension
c

      t      = 2 * pi * (time - int(time)) + ll
      ksi1   = t + h
      ksi    = ksi1 - ny
      l1     = h + 2*e1*sin(h-p1)
      alfa   = 2 * atan((sin(omega)*sin(N)/sinii)  /
     .                  (1 + cos(N)*cos(ny)
     .                   + sin(N)*sin(ny)*cos(omega)))
      sigma  = s - N + alfa
      l      = sigma + 2*e*sin(s-p)
     .       + 5d0/4*e*e*sin(2*(s-p))
     .       + 15d0/4*m*e*sin(s - 2*h + p)
     .       + 11d0/8*m*m*sin(2*(s -h))
c
c  zenith angles
c
      costheta = sinlambda*sinii*sin(l)
     .           + coslambda*(cos(ii/2)**2*cos(l-ksi) +
     .                        sin(ii/2)**2*cos(l+ksi))
      cosphi   = sinlambda*sin(omega)*sin(l1)
     .           + coslambda*(cos(omega/2)**2*cos(l1-ksi1) +
     .                        sin(omega/2)**2*cos(l1+ksi1))
c
c  gravities
c
      gs    = my*ss*r*resdd**3*(3*cosphi**2 - 1)
      gm    = my*mm*r*resd**3*(3*costheta**2 - 1)
     .        + 3/2*my*mm*r**2*resd**4*
     .          (5*costheta**3 - 3*costheta)
      psu=1.0d-4*(gs*r/2.0)
c +my*ss*r*resdd**2*cosphi)
      pmo=1.0d-4*(gm*r/2.0)
c +my*mm*r*resd**2*costheta)
      g0    = gm + gs
c
c  transformation from the cgs unit gal to mgal
      tide   = g0 * 1000
      return
      end
```



```
      subroutine r180(ang)
c the subroutine will trans an angle to the interval -180.0 to 180.0
c degrees. 2002-09-19.
      implicit none
      real*8 ang,ang1
 10   if (ang.gt.180.0d0) then
       ang=ang-360.0d0
       go to 10
      end if
 20   if (ang.lt.-180.0d0) then
       ang=ang+360.0d0
       go to 20
      end if
      return
      end
```