```fortran
      program marsg
c
c     This program provides a Mars-GRAM type interface to the Mars Database
c     developed jointly in Atmospheric, Oceanic and Planetary Physics, Oxford
c     and at the Laboratoire de Meteorologie Dynamique, Paris, under
c     ESTEC Contract 11369/95/NL/JG.
c
c     Fortran channel numbers:
c
c     5 Standard input
c     6 Standard output
c     10 list file
c     11 output file
c     12 DENSLO
c     13 DENSAV
c     14 DENSHI
c     15 DENSRP
c     16 TEMP
c     17 PRESS
c     18 EWWIND
c     19 NSWIND
c     20 TMAX
c     21 TMIN
c     22 TAVG
c     23 HGTrho
c     24 DENSRM
c     25 DENSWA
c     26 Tbase
c     27 Zbase
c     28 TRAJDATA
c     30 cospar atmosphere
c     60 DRS dic file: mean data
c     61 DRS dat file: mean data
c     62 DRS dic file: std dev data
c     63 DRS dat file: std dev data
c     64 DRS dic file: orography data
c     65 DRS dat file: orography data
c     66 DRS dic file: eof data
c     67 DRS dat file: eof data
c
      integer nout /6/           !Output channel number
      integer nin  /5/           !Input channel number
      character*255 list
      character*255 outf
      character*16 name
      integer month
      integer day
      integer year
      integer npos
      integer hour
      integer minute
      integer second
      real*8  date
      real    marsau
      integer ierr
      integer udrsm /60/         !DRS mean file unit number, udrsm+1 is also used
      integer udrssd /62/        !DRS std dev file unit number, ditto
      integer ueof /66/          !DRS eof file unit number, ditto
      integer noutf /11/         !output file
      integer dust               !dust scenario 1=Viking 2=lowdust hight for storms
      integer nvx
      integer nvy
      integer lgs
      integer seed
      integer varflag
      real    ls
      real    sunlon
      real    localtime
      real    ilat
      real    ilon
      real    dlat
      real    dlon
      real    dheight
      real    dtime
      real    oroheight
```

```
      real    iheight
      real    stormls
      real    intens
      real    raddust
      real    dustlat
      real    dustlon
      real    f107
      real    sdf107
c
      write(nout,*) 'Mars Climate Database Version 1.0 - MarsGRAM',
     :               ' style interface'
      write(nout,*) 'Enter file name for full output listing',
     :       ' (CON for screen listing): '
      read(nin,'(A)') list
      write(nout,*) 'Enter file name for tabulated output:'
      read(nin,'(A)') outf
      ierr=1
      do while (ierr.ne.0) !make sure month etc. is in correct range
        write(nout,*) 'Enter month, day of month, 4-digit year,',
     : ' and maximum number of positions (all integers) - Enter zero',
     : ' number of positions to read positions from TRAJDATA file'
        read(nin,*) month,day,year,npos
        write(nout,*) 'Enter initial GMT Time in Hours, Minutes,',
     :                 ' Seconds (all integers)'
        read(nin,*) hour,minute,second
c
c       Compute Ls
c
        call ctime(month,day,year,hour,minute,second,ierr,
     :             date,ls,marsau)
        if (ierr.ne.0) write(nout,*) 'Re-enter date -',
     :                               ' field out of range'
      enddo
c
c     If npos=0 then lon, lat etc. are read from TRAJDATA file
c
      if (npos.eq.0)
     :      open(28,file='TRAJDATA',form='formatted',status='old')
    8 format(' Date = ',i2,'/',i2,'/',i4,'  Julian Date = ',f9.1,
     &  '  GMT Time = ',i2,':',i2,':',f4.1)
      write(nout,8) month,day,year,date,hour,minute,second
   10 format(' Ls = ',F6.1,' degrees for this date.')
      write(nout,10) ls
c
c     Select dust amount
c
      write(nout,*) 'Select dust scenario',
     : '  (1) Viking dust scenario',
     : '  (2) Low dust scenario',
     : '  (3) Dust storm, tau=2',
     : '  (4) Dust storm, tau=5'
      read(nin,*) dust
      if (dust.eq.0) dust=1 !for some compatability with Mars-Gram
      if ((dust.lt.1).or.(dust.gt.4)) then
        dust=1
        write(nout,*) 'Viking scenario assumed'
      endif
      if (((dust.eq.3).or.(dust.eq.4)).and.
     :    ((ls.lt.210.).or.(ls.gt.300))) then
        dust=1
      write(nout,*) 'No data exists for dust storm scenarios at this',
     : 'Ls: Viking scenario assumed'
      endif
      if (dust.eq.1) intens=1.0
      if (dust.eq.3) intens=2.0
      if (dust.eq.4) intens=3.0
      stormls=210.
c
c     Open appropriate DRS file
c
      call opend(udrsm,udrssd,ueof,ls,dust)
c
c     Select 10.7 cm solar radio flux (units 10**-22 W/cm**2) for
c     thermosphere model
c
```

```
c   14 continue
c      write(nout,*)'Enter mean F10.7 flux at 1AU (nominal value = 150)',
c     :  ' and +/- number of std. deviations for thermosphere variation'
c      read(nin,*) f107,sdf107
c      if ((f107.lt.50.).or.(f107.gt.450.)) then
c        write(nout,*) 'F10.7 must be between 50 and 450'
c        goto 14
c      endif
c      if ((sdf107.lt.-3.).or.(sdf107.gt.3.)) then
c        write(nout,*) 'Std. deviations must be between -3 and +3'
c        goto 14
c      endif
      f107=150.
      sdf107=2.
c
c     Select perturbation model
c
      write(nout,*) 'Enter perturbation model: 1=none, 2=large scale',
     :' 3=small scale, 4=large scale and small scale'
      read(nin,*) varflag
c
c     Select seed for random perturbation
c
   15 continue
      write(nout,*)'Enter seed for random number generator (any ',
     :  ' positive integer < 30,000)'
      read(nin,*) seed
      if ((seed.lt.0).or.(seed.ge.30000)) goto 15
c
c     Select output
c
      write(nout,19)
   19 format(/' Select x-code and y-code for plotable output versus',
     :  ' desired parameter(s):'//
     :  ' Code              Parameter'/
     :  ' ----   -------------------------------------------------'/
     :  '   1    Height (above reference ellipsoid, km)'/
     :  '   2    Height (above local terrain, km)'/
     :  '   3    Latitude (deg.)'/
     :  '   4    Longitude (deg.)'/
     :  '   5    Time from start (Earth seconds)'/
     :  '   6    Time from start (Martian Sols)'/
     :  '   7    Areocentric Longitude of Sun, Ls (deg.)'/
     :  '   8    Hour Angle for Local Time (Mars hours * 15)'//
     :  ' Use y-code = 0 for plotable output vs x-code variable only')
      read(nin,*) nvx,nvy
      if ((nvx.lt.1).or.(nvx.gt.8)) nvx=1
      if ((nvy.lt.0).or.(nvy.gt.8)) nvy=0
      write(nout,*)'Enter 0 for normal (MKS) units for density,',
     : ' 1 for log-base-10 or 2 for % difference from COSPAR'
      read(nin,*) lgs
      if (lgs.ne.1) lgs=0
c
c     If npos=0 then lon,lat etc. read from TRAJDATA so jum to dstep
c
      if (npos.eq.0) goto 60
      ierr=1
      do while (ierr.ne.0) !make sure longitude and latitude are in range
        write(nout,*) 'Enter initial latitude and west longitude in',
     :                 ' degrees'
        read(nin,*) ilat,ilon
        if ((ilat.ge.-90.).and.(ilat.le.90.)
     :     .and.(ilon.ge.0.).and.(ilon.le.360.)) ierr=0
        if (ierr.ne.0) write(nout,*) 'Invalid coordinates: re-enter'
      enddo
      name='orography'
      call mount(ilon,ilat,oroheight,name)
   20 format(' Surface elevation = ',F10.6,' km at this location')
      write(nout,20) oroheight/1000.
      write(nout,*) 'Enter Initial Height relative to zero datum (km)'
      read(nin,*) iheight
      write(nout,*) 'Enter Increments in Height (km), Latitude (deg.),',
     :  ' West Longitude (deg.),  and Time (sec.)'
      read(nin,*) dheight,dlat,dlon,dtime
c
```

```
      if ((list.ne.'CON').and.(list.ne.'con')) then
        nout=10
        open(nout,file=list,form='formatted',status='unknown')
        write(nout,*) 'Mars Climate Database Version 1.0 - MarsGRAM',
     :               ' style interface'
        write(nout,30) month,day,year,date,hour,minute,second
   30   format(' Date = ',I2,'/',I2,'/',I4,'  Julian Date = ',F9.1,
     : '  GMT Time = ',I2,':',I2,':',F4.1)
c        write(nout,*) 'Height of 1.26 nbar level from hydrostatic',
c     :  ' interpolation'
c        write(nout,40) f107,f107/marsau**2,sdf107
c   40   format(' F10.7 flux = ',F5.1,' (1 AU)  ',F5.1,
c     &    ' (Mars),  standard deviation = ',F4.1)
        write(nout,50) seed
   50   format(' Random seed =',I6)
      endif
   60 continue
      open(noutf,file=outf,form='formatted',status='unknown')
      open(noutf+1,file='DENSLO',form='formatted',status='unknown')
      open(noutf+2,file='DENSAV',form='formatted',status='unknown')
      open(noutf+3,file='DENSHI',form='formatted',status='unknown')
      open(noutf+4,file='DENSRP',form='formatted',status='unknown')
      open(noutf+5,file='TEMP',form='formatted',status='unknown')
      open(noutf+6,file='PRES',form='formatted',status='unknown')
      open(noutf+7,file='EWWIND',form='formatted',status='unknown')
      open(noutf+8,file='NSWIND',form='formatted',status='unknown')
      open(noutf+9,file='TMAX',form='formatted',status='unknown')
      open(noutf+10,file='TMIN',form='formatted',status='unknown')
      open(noutf+11,file='TAVG',form='formatted',status='unknown')
      open(noutf+12,file='HGTrho',form='formatted',status='unknown')
      open(noutf+13,file='DENSRM',form='formatted',status='unknown')
      open(noutf+14,file='DENSWA',form='formatted',status='unknown')
      open(noutf+15,file='Tbase',form='formatted',status='unknown')
      open(noutf+16,file='Zbase',form='formatted',status='unknown')
      call dstep(udrsm,udrssd,ueof,seed,nout,noutf,npos,nvx,nvy,lgs,
     :  date,ilon,ilat,dlon,dlat,iheight,dheight,dtime,f107,sdf107,
     :  stormls,intens,dust,varflag)
      end
```

```
      subroutine atime(lt,ut,lonw)
c
c     Convert local time, lt (1..24) at west longitude=lonw to
c     universal time, ut (1..12). Note that it is midnight at
c     east longitude=0.0 at universal time=12.
c
c     input
c
      real lt         !local time
      real lonw       !longitude west
c
c     output
c
      real ut         !universal time
c
c     local
c
      real lon        !longitude east
      integer iut
      lon=360.-lonw
      iut=mod(nint(-lon/30.+lt/2.+12.),12)+1
      ut=float(iut)
      if (iut.eq.13) ut=1.0
      return
      end
```

```
      Subroutine cospar(z,t,p,rho)
c
c     Calculate COSPAR northern hemisphere mean temperature, pressure
c     and density at height=z
c
c     input
c
      real z        !Height (km)
c
c     output
c
      real t        !Temperature
      real p        !Pressure
      real rho      !Density
c
c     local
c
      real zc(159)
      real tc(159)
      real pc(159)
      real dc(159)
      integer i
      open(30,file='../data/cospar.dat',
     :     form='formatted',status='old')
      do i=1,159
        read(30,*) zc(i),tc(i),pc(i),dc(i)
      enddo
      close(30)
      if (z.lt.130.) then
        iz = Int(z+6.)
      else
        iz = 136 + Int((z-130.)/10.)
      endif
      if ((iz.lt.1).or.(iz.gt.159)) then
        t = 0.
        p = 0.
        rho = 0.
        return
      endif
      if (iz.gt.158) iz=158
c
c     Linear interpolation on temperature
c
      dz = (z - zc(iz))/(zc(iz+1) - zc(iz))
      t = tc(iz) + (tc(iz+1) - tc(iz))*dz
c
c     Pressure from hydrostatic relation (with special isothermal case)
c
      if (abs(tc(iz+1)-tc(iz)).gt.0.01) then
        aexp = Alog(pc(iz+1)/pc(iz))/Alog(tc(iz+1)/tc(iz))
        p = 100.*pc(iz)*(t/tc(iz))**aexp
      else
        H = (zc(iz+1) - zc(iz))/Alog(pc(iz)/pc(iz+1))
        p = 100.*pc(iz)*exp(-(z-zc(iz))/H)
      endif
c
c     Linear interpolation on gas constant
c
      R1 = pc(iz)/(dc(iz)*tc(iz))
      R2 = pc(iz+1)/(dc(iz+1)*tc(iz+1))
      R = R1 + (R2-R1)*dz
c
c     density from perfect gas law (and convert units to kg/m**3)
c
      rho = 10.*p/(R*t)
c
      return
      end
```

```
      subroutine ctime(month,day,year,hour,minute,second,ierr,
     :                 date,ls,marsau)
c
c     Given Earth date and time compute Ls and local time on Mars
c
c     Inputs
c
      integer month
      integer day
      integer year
      integer hour       !All Earth GMT values
      integer minute
      integer second
c
c     Output
c
      integer ierr       !0 if ok >0 if there is a problem
      real    ls         !Ls
      real*8  date       !Julian date
      real    marsau     !Mars distance from sun in AU
c
c     Local
c
      integer nday
      integer daynumber(12)   !days for months of the year
      data    daynumber/0,31,59,90,120,151,181,212,243,273,304,334/
      real*8  xyear
      real*8  ddate
      real    sunlat
      real    sunlon
c
c     Check range of month etc.
c
      ierr=0
      if ((month.lt.1).or.(month.gt.12)) then
        ierr=1
        return
      endif
      if ((day.lt.1).or.(day.gt.31)) then
        ierr=2
        return
      endif
      if (year.lt.1) then
        ierr=3
        return
      endif
      if ((hour.lt.0).or.(hour.gt.23)) then
        ierr=4
        return
      endif
      if ((minute.lt.0).or.(minute.gt.59)) then
        ierr=5
        return
      endif
      if ((second.lt.0).or.(second.gt.23)) then
        ierr=6
        return
      endif
c
c     Calculate Julian date and Ls
c
      nday=daynumber(month)+day
c
c     Correct for leap year
c
      if ((mod(year,4).eq.0).and.(month.gt.2)) nday=nday+1
c
c     Compute Julian date
c
      xyear=(year-1.9665d3)/4.0d0
      date=2.439856d6+3.65d2*(year-1.968d3)+nday+
     :     int(xyear+dsign(0.5d0,xyear))-0.5d0
      date=date+hour/24.0d0+minute/1.440d3+second/8.6400d4
c
c     Compute Ls
```

```
c
      call orbit(date,sunlat,sunlon,ls,marsau)
c
      return
      end
```

```fortran
      subroutine dstep(udrsm,udrssd,ueof,seed,nout,noutf,maxp,nvx,nvy,
     :lgs,idate,ilon,ilat,dlon,dlat,iheight,dheight,dtime,f107,sdf107,
     :stormls,intens,dust,varflag)
c
c     Loop over longitude, latitude, height and time to extract data
c     from the database. Add large and small scale variability to if
c     required. Calculate various other quantaties and output data
c     in the same format as MarsGRAM version 3.5.
c
#include "drsdef.h"
c
c     Input
c
      integer udrsm    !DRS channel number - mean fields
      integer udrssd   !DRS channel number - std dev fields
      integer ueof     !DRS channel number - eof fields
      integer seed     !seed for random number generator
      integer nout     !Output channel number
      integer noutf    !Oupput file channel number
      integer maxp     !Maximum number of outputs
      integer nvx      !Controls formatted outout
      integer nvy      !Controls formatted outout
      integer lgs      !logscale on formatted output
      real*8  idate    !Initial Julian date
      real    ilon     !Initial longitude (west)
      real    ilat     !Initial latitude
      real    dlon     !longitude step
      real    dlat     !latitude step
      real    iheight  !initial height above surface
      real    dheight  !height step
      real    dtime    !time step
      real    f107     !10.7 solar flux
      real    sdf107   !Standard deviation of f107
      real    stormls  !Starting ls for dust storm
      real    intens   !Intensity of dust storm
      integer dust     !Dust flag
      integer varflag  !Variability model flag
c
c     Local
c
      real    pi /3.14159265359/
      real    beta /1.578e-6/
      real    sval /240./
      real    sollength /88775.245/
      real    maxheight /300./
      real    R /191.2/
      real    g /3.72/
      integer nl
      parameter(nl=25)   !Number of levels
      integer np
      integer ierr
      integer i
      real*8  date
      real    utime
      real    time
      real    height
      real    sheight    !height relative to zero datum
      real    oroheight  !orographic height
      real    absheight  !height above surface
      real    outheight
      real    tmpheight
      real    ls
      real    sunlon
      real    sunlat
      real    marsau
      real    localtime
      real    lon        !longitude west
      real    lat
      real    prevlon
      real    prevlat
      real    t
      real    tl         !l indicates low level variable from database
      real    tu         !u indicates upper level var from MarsGRAM thermosphere
      real    Tmax
      real    Tmin
```

```
      real    Tsurf
      real    ps
      real    p
      real    pl
      real    pu
      real    rho
      real    rhol
      real    rhou
      real    rholo
      real    rholol
      real    rholoh
      real    rhohi
      real    rhohil
      real    rhohiu
      real    sd
      real    sdrho
      real    sdrhol
      real    sdrhou
      real    pertm
      real    pertr
      real    pertmrho
      real    pertrrho
      real    pertrrhol
      real    pertrrhou
      real    u
      real    ul
      real    uu
      real    v
      real    vl
      real    vu
      real    sigma(nl)
      real    s1
      integer levhi
      integer levlow
      real    levweight
      real    pscaleheight
      real    rscaleheight
      real    vx
      real    vy
      real    t75
      real    t50
      real    t5
      real    platp
      real    plonp
      real    platm
      real    plonm
      real    tmplat
      real    tmplon
      real    delns
      real    delew
      real    f
      real    fug
      real    fvg
      real    fugp
      real    fvgp
      real    rsc
      real    dtr
      real    pcos
      real    tcos
      real    rcos
      real    rcosav
      real    rcoslo
      real    rcoshi
      real    texos
      real    tbase
      real    zbase
      real    visc
      real    viscfac
      real    gasd(6)
      real    dvardz
      real    dev
      character*16 name
c
c     Get sigma from DRS file
c
```

```
      ierr = cluvdb()
      ierr = setname(' ','sigma',' ',' ',' ')
      ierr = getdat(udrsm,sigma,IDRS_BYTES_PER_WORD*nl)
c
      np=1
      date=idate
      time=0.
      zbase=0.
      do i=1,6
        gasd(i)=gasdev(seed)
      enddo
      dev=ran(seed)
      do while ((np.le.maxp).or.(maxp.eq.0))
       height=iheight
       do while (((height.le.maxheight).and.(np.le.maxp))
     :            .or.(maxp.eq.0))
        lat=ilat
        prevlat=lat
        do while (((lat.le.90.).and.(np.le.maxp)).or.(maxp.eq.0))
         lon=ilon
         prevlon=lon
         do while (((lon.le.360.).and.(np.le.maxp)).or.(maxp.eq.0))
c
c        if maxp=0 then read time,height,lat and lon from TRAJDATA
c
         if (maxp.eq.0) then
           read(28,*) time,height,lat,lon
           date=idate+time/8.6400d4
         endif
c
c         calculate absolute time and localtime
c
          call orbit(date,sunlat,sunlon,ls,marsau)
c
c         call ltime with initial longitude to keep local time fixed
c
          call ltime(ilon,sunlon,localtime)
c
c         call ltime with current longitude for correct local time
c
c          call ltime(lon,sunlon,localtime)
          call atime(localtime,utime,lon)
c
c         open appropriate DRS file
c
          call opend(udrsm,udrssd,ueof,ls,dust)
          if (height.lt.85.) then
c
c           get data from the database
c
c
c           calculate sigma level at height=absheight
c           (absheight=height above ground)
c
            name='orography'
            call mount(lon,lat,oroheight,name)
            sheight=height
            absheight=sheight-oroheight/1000.
            call getsi(udrsm,sigma,lon,lat,absheight,utime,
     :                 levhi,levlow,levweight)
            if (levweight.lt..5) then
              s1=sigma(levlow) !nearest sigma level
            else
              s1=sigma(levhi)  !nearest sigma level
            endif
c
c           read data from DRS file.
c
            name='u'
            call var3d(udrsm,ul,lon,lat,levhi,levlow,levweight,utime,
     :                 name)
            name='v'
            call var3d(udrsm,vl,lon,lat,levhi,levlow,levweight,utime,
     :                 name)
            name='t'
```

```
      call var3d(udrsm,tl,lon,lat,levhi,levlow,levweight,utime,
     :           name)
      name='rho'
      call var3d(udrsm,rhol,lon,lat,levhi,levlow,levweight,utime,
     :           name)
      name='sdrho'
      call var3d(udrssd,sdrhol,lon,lat,levhi,levlow,levweight,1.0,
     :           name)
      name='ps'
      call var2d(udrsm,ps,lon,lat,utime,name)
      name='tsurf'
      call var2d(udrsm,Tsurf,lon,lat,utime,name)
      call max2d(udrsm,Tmax,lon,lat,name)
      call min2d(udrsm,Tmin,lon,lat,name)
      if ((varflag.eq.2).or.(varflag.eq.4)) then
c
c       add large scale variability to u,v,t and ps
c       if different lon or lat from previous recomputed gaussian
c       deviates for eof model and square dist. random number for
c       gravity wave model.
c
        if ((abs(prevlon-lon).gt.1.e-20).or.
     :      (abs(prevlat-lat).gt.1.e-20)) then
          do i=1,6
            gasd(i)=gasdev(seed)
          enddo
          dev=ran(seed)
        endif
        name='sdu'
        call var3d(udrssd,sd,lon,lat,levhi,levlow,levweight,
     :             1.0,name)
        name='u'
        call eofpb(ueof,pertm,pertr,gasd,lon,lat,s1,
     :             ls,sd,name)
        ul=ul+pertr
        name='sdv'
        call var3d(udrssd,sd,lon,lat,levhi,levlow,levweight,
     :             1.0,name)
        name='v'
        call eofpb(ueof,pertm,pertr,gasd,lon,lat,s1,
     :             ls,sd,name)
        vl=vl+pertr
        name='sdt'
        call var3d(udrssd,sd,lon,lat,levhi,levlow,levweight,
     :             1.0,name)
        name='t'
        call eofpb(ueof,pertm,pertr,gasd,lon,lat,s1,
     :             ls,sd,name)
        tl=tl+pertr
        name='sdps'
        call var2d(udrssd,sd,lon,lat,1.0,name)
        name='ps'
        call eofpb(ueof,pertm,pertr,gasd,lon,lat,s1,
     :             ls,sd,name)
        ps=ps+pertr
      endif
      if ((varflag.eq.3).or.(varflag.eq.4)) then
c
c       add small scale (gravity wave) variability
c
        name='u'
        call vgrad(udrsm,lon,lat,absheight,utime,sigma,dvardz,
     :             name)
        call grwpb(udrsm,lon,lat,absheight,rhol,ul,vl,utime,
     :             dvardz,sigma,dev,pertr)
        ul=ul+pertr
        name='v'
        call vgrad(udrsm,lon,lat,absheight,utime,sigma,dvardz,
     :             name)
        call grwpb(udrsm,lon,lat,absheight,rhol,ul,vl,utime,
     :             dvardz,sigma,dev,pertr)
        vl=vl+pertr
        name='t'
        call vgrad(udrsm,lon,lat,absheight,utime,sigma,dvardz,
     :             name)
```

```
            call grwpb(udrsm,lon,lat,absheight,rhol,ul,vl,utime,
     :                 dvardz,sigma,dev,pertr)
            tl=tl+pertr
          endif
c
c         calculate some other things
c
          pl=ps*sigma(levlow)+(sigma(levhi)-sigma(levlow))*levweight
          pscaleheight=R*Tl/g
          pscaleheight=pscaleheight/1000.
          rscaleheight=pscaleheight
c
c         calculate density perturbation using eof method
c
          name='rho'
          call eofpb(ueof,pertmrho,pertrrhol,gasd,lon,lat,s1,
     :               ls,sdrhol,name)
c
c         adding the mean perturbation may be a bad idea as variables
c         can become discontinuous across season boundaries.
c
c          rhol=rhol+pertmrho
          rholol=rhol-sdrhol
          rhohil=rhol+sdrhol
c
c         calculate density perturbation due to gravity waves
c
          name='rho'
          call vgrad(udrsm,lon,lat,absheight,utime,sigma,dvardz,
     :               name)
          call grwpb(udrsm,lon,lat,absheight,rhol,ul,vl,utime,
     :               dvardz,sigma,dev,pertr)
          rholol=rhol-abs(pertr)
          rhohil=rhol+abs(pertr)
c
          t=tl
          p=pl
          rho=rhol
          rholo=rholol
          rhohi=rhohil
          pertrrho=pertrrhol
          sdrho=sdrhol
          u=ul
          v=vl
c
        endif
        if (height.gt.75.) then
c
c         Use Mars-GRAM thermosphere model.
c
c         Compute wind using geostrophic balance. Difference across
c         5 degrees of longitude and latitude
c
          tmplon=lon+2.5
          if (tmplon.gt.360.) tmplon=tmplon-360.
          name='orography'
          call mount(lon,lat,oroheight,name)
          sheight=height
          absheight=sheight-oroheight/1000.
          tmpheight=50.-oroheight/1000.
          call getsi(udrsm,sigma,tmplon,lat,tmpheight,utime,
     :               levhi,levlow,levweight)
          name='t'
          call var3d(udrsm,t50,tmplon,lat,levhi,levlow,levweight,
     :               utime,name)
          tmpheight=75.-oroheight/1000.
          call getsi(udrsm,sigma,tmplon,lat,tmpheight,utime,
     :               levhi,levlow,levweight)
          name='t'
          call var3d(udrsm,t75,tmplon,lat,levhi,levlow,levweight,
     :               utime,name)
          name='tsurf'
          call var2d(udrsm,tsurf,tmplon,lat,utime,name)
          name='ps'
          call var2d(udrsm,ps,tmplon,lat,utime,name)
```

```
      tmpheight=5.-oroheight/1000.
      call getsi(udrsm,sigma,tmplon,lat,tmpheight,utime,
     :           levhi,levlow,levweight)
      name='t'
      call var3d(udrsm,t5,tmplon,lat,levhi,levlow,levweight,
     :           utime,name)
      call mgthm(height,lat,tmplon,f107,sdf107,
     :         sunlat,sunlon,als,marsau,t50,t75,tsurf,ps,t5,
     :         stormls,intens,pscaleheight,tu,rhou,
     :         rhohiu,rholou,plonp,rsc,rscaleheight,
     :         texos,tbase,zbase)
      tmplon=lon-2.5
      if (tmplon.lt.0.) tmplon=tmplon+360.
      tmpheight=50.-oroheight/1000.
      call getsi(udrsm,sigma,tmplon,lat,tmpheight,utime,
     :           levhi,levlow,levweight)
      name='t'
      call var3d(udrsm,t50,tmplon,lat,levhi,levlow,levweight,
     :           utime,name)
      tmpheight=75.-oroheight/1000.
      call getsi(udrsm,sigma,tmplon,lat,tmpheight,utime,
     :           levhi,levlow,levweight)
      name='t'
      call var3d(udrsm,t75,tmplon,lat,levhi,levlow,levweight,
     :           utime,name)
      name='tsurf'
      call var2d(udrsm,tsurf,tmplon,lat,utime,name)
      name='ps'
      call var2d(udrsm,ps,tmplon,lat,utime,name)
      tmpheight=5.-oroheight/1000.
      call getsi(udrsm,sigma,tmplon,lat,tmpheight,utime,
     :           levhi,levlow,levweight)
      name='t'
      call var3d(udrsm,t5,tmplon,lat,levhi,levlow,levweight,
     :           utime,name)
      call mgthm(height,lat,tmplon,f107,sdf107,
     :         sunlat,sunlon,als,marsau,t50,t75,tsurf,ps,t5,
     :         stormls,intens,pscaleheight,tu,rhou,
     :         rhohiu,rholou,plonm,rsc,rscaleheight,
     :         texos,tbase,zbase)
      tmplat=lat+2.5
      tmplon=lon
      if (tmplat.gt.90.) then
        tmplat=180.-tmplat
        tmplon=tmplon+180.
      endif
      tmpheight=50.-oroheight/1000.
      call getsi(udrsm,sigma,tmplon,tmplat,tmpheight,utime,
     :           levhi,levlow,levweight)
      name='t'
      call var3d(udrsm,t50,tmplon,tmplat,levhi,levlow,levweight,
     :           utime,name)
      tmpheight=75.-oroheight/1000.
      call getsi(udrsm,sigma,tmplon,tmplat,tmpheight,utime,
     :           levhi,levlow,levweight)
      name='t'
      call var3d(udrsm,t75,tmplon,tmplat,levhi,levlow,levweight,
     :           utime,name)
      name='tsurf'
      call var2d(udrsm,tsurf,tmplon,tmplat,utime,name)
      name='ps'
      call var2d(udrsm,ps,tmplon,tmplat,utime,name)
      tmpheight=5.-oroheight/1000.
      call getsi(udrsm,sigma,tmplon,tmplat,tmpheight,utime,
     :           levhi,levlow,levweight)
      name='t'
      call var3d(udrsm,t5,tmplon,tmplat,levhi,levlow,levweight,
     :           utime,name)
      call mgthm(height,tmplat,tmplon,f107,sdf107,
     :         sunlat,sunlon,als,marsau,t50,t75,tsurf,ps,t5,
     :         stormls,intens,pscaleheight,tu,rhou,
     :         rhohiu,rholou,platp,rsc,rscaleheight,
     :         texos,tbase,zbase)
      tmplat=lat-2.5
      tmplon=lon
```

```
          if (tmplat.lt.-90.) then
            tmplat=-180.+tmplat
            tmplon=tmplon+180.
          endif
          tmpheight=50.-oroheight/1000.
          call getsi(udrsm,sigma,tmplon,tmplat,tmpheight,utime,
     :                levhi,levlow,levweight)
          name='t'
          call var3d(udrsm,t50,tmplon,tmplat,levhi,levlow,levweight,
     :                utime,name)
          tmpheight=75.-oroheight/1000.
          call getsi(udrsm,sigma,tmplon,tmplat,tmpheight,utime,
     :                levhi,levlow,levweight)
          name='t'
          call var3d(udrsm,t75,tmplon,tmplat,levhi,levlow,levweight,
     :                utime,name)
          name='tsurf'
          call var2d(udrsm,tsurf,tmplon,tmplat,utime,name)
          name='ps'
          call var2d(udrsm,ps,tmplon,tmplat,utime,name)
          tmpheight=5.-oroheight/1000.
          call getsi(udrsm,sigma,tmplon,tmplat,tmpheight,utime,
     :                levhi,levlow,levweight)
          name='t'
          call var3d(udrsm,t5,tmplon,tmplat,levhi,levlow,levweight,
     :                utime,name)
          call mgthm(height,tmplat,tmplon,f107,sdf107,
     :              sunlat,sunlon,als,marsau,t50,t75,tsurf,ps,t5,
     :              stormls,intens,pscaleheight,tu,rhou,
     :              rhohiu,rholou,platm,rsc,rscaleheight,
     :              texos,tbase,zbase)
c
c         get data from Mars-GRAM stratosphere/thermosphere model
c
          tmpheight=50.-oroheight/1000.
          call getsi(udrsm,sigma,lon,lat,tmpheight,utime,
     :                levhi,levlow,levweight)
          name='t'
          call var3d(udrsm,t50,lon,lat,levhi,levlow,levweight,
     :                utime,name)
          tmpheight=75.-oroheight/1000.
          call getsi(udrsm,sigma,lon,lat,tmpheight,utime,
     :                levhi,levlow,levweight)
          name='t'
          call var3d(udrsm,t75,lon,lat,levhi,levlow,levweight,
     ;                utime,name)
          name='tsurf'
          call var2d(udrsm,tsurf,lon,lat,utime,name)
          name='ps'
          call var2d(udrsm,ps,lon,lat,utime,name)
          tmpheight=5.-oroheight/1000.
          call getsi(udrsm,sigma,lon,lat,tmpheight,utime,
     :                levhi,levlow,levweight)
          name='t'
          call var3d(udrsm,t5,lon,lat,levhi,levlow,levweight,
     ;                utime,name)
          call mgthm(height,lat,lon,f107,sdf107,
     :              sunlat,sunlon,als,marsau,t50,t75,tsurf,ps,t5,
     :              stormls,intens,pscaleheight,tu,rhou,
     :              rhohiu,rholou,pu,rsc,rscaleheight,
     :              texos,tbase,zbase)
          pertrrhou=gasdev(seed)*rhou*(rhohiu-rholou)*.5
          rholou=rholou*rhou
          rhohiu=rhohiu*rhou
          sdrhou=abs(rhohiu-rholou)
          dtr=pi/180.
          delns=dtr*rsc*5.e3
          tmplat=lat
          if (abs(lat).gt.87.5) tmplat=sign(87.5,lat)
          if (abs(lat).lt.2.5) tmplat=sign(2.5,lat)
          delew=-dtr*rsc*cos(dtr*tmplat)*5.e3
          f=2.*7.08822e-5*sin(dtr*tmplat)
          visc=beta*tu**1.5/(tu+sval)
          viscfac=0.04*visc/(1.e6*rhou*pscaleheight**2)
          fug=-(platp-platm)/(delns*rhou)
```

```
          fvg=(plonp-plonm)/(delew*rhou)
          if (abs(tmplat).lt.15.) then
            fugp=-4000.*rsc*sin(dtr*tmplat)*(platp+platm-2.*pu)/
     :        (rhou*cos(dtr*tmplat)*delns*delns)
            fvgp=0.0
            if (abs(tmplat).gt.7.5) then
              fug=fugp+(abs(tmplat)-7.5)*(fug-fugp)/7.5
              fvg=fvgp+(abs(tmplat)-7.5)*(fvg-fvgp)/7.5
            else
              fug=fugp
              fvg=fvgp
            endif
          endif
          if (abs(tmplat).gt.75.) then
              fug=fug*cos(tmplat)*cos(tmplat)/cos(dtr*75.)
              fvg=fvg*cos(tmplat)*cos(tmplat)/cos(dtr*75.)
          endif
          uu=(f*fug-viscfac*fvg)/(f**2+viscfac**2)
          vu=(f*fvg+viscfac*fug)/(f**2+viscfac**2)
c
          t=tu
          p=pu
          rho=rhou
          rholo=rholou
          rhohi=rhohiu
          pertrrho=pertrrhou
          sdrho=sdrhou
          u=uu
          v=vu
c
        endif
        if ((height.gt.75.).and.(height.lt.85.)) then
c
c         Blend database and MarsGRAM together
c
          t=.1*(85.-height)*tl+.1*(height-75.)*tu
          p=.1*(85.-height)*pl+.1*(height-75.)*pu
          rho=.1*(85.-height)*rhol+.1*(height-75.)*rhou
          rholo=.1*(85.-height)*rholol+.1*(height-75.)*rholou
          rhohi=.1*(85.-height)*rhohil+.1*(height-75.)*rhohiu
          pertrrho=.1*(85.-height)*pertrrhol+.1*(height-75.)*pertrrhou
          sdrho=.1*(85.-height)*sdrhol+.1*(height-75.)*sdrhou
          u=.1*(85.-height)*ul+.1*(height-75.)*uu
          v=.1*(85.-height)*vl+.1*(height-75.)*vu
        endif
c
c       compute percentage deviation from COSPAR atmosphere
c
        call cospar(height,tcos,pcos,rcos)
        if (abs(rcos).le.1.e-20) then
          rcoslo=-99.9
          rcosav=-99.9
          rcoshi=-99.9
        else
          rcoslo=100.*(rholo-rcos)/rcos
          rcoshi=100.*(rhohi-rcos)/rcos
          rcosav=100.*(rho-rcos)/rcos
        endif
c
c       output (as in MarsGRAM 3.5)
c
       write(nout,10) time,time/sollength,ls,sheight,absheight,
     :         pscaleheight,rscaleheight,lat,lon,sunlat,marsau,sunlon,
     :         localtime
  10   format(' Time (rel. to T0) =',F10.1,' sec. (',F8.3,' sols)  ',
     : '  Ls = ',F6.1,' deg.'/' Height =',F7.2,' km (',F7.2,' km)   ',
     : 'Scale Hgt H(p) =',F6.2,' H(rho) =',F6.2,' km'/' Latitude = ',
     : F8.3,' degrees   ','    West Longitude = ',F8.3,' degrees'/
     : ' Sun Latitude =',F9.2,' deg.      Mars Orbital Radius =',F6.3,
     : ' AU'/' Sun Longitude =',F8.2,' deg.      Local Time = ',F6.2,
     : ' Mars hours')
       if (zbase.gt.0.) write(nout,15) texos,tbase,zbase
  15   format(' Exospheric Temp. = ',F6.1,' K',7X,'Tbase = ',F6.1,
     : ' K',3X,' Zbase = ',F6.1,' km')
       write(nout,20) T,p,rholo,rho,rhohi,rcoslo,rcosav,rcoshi,
```

```
     :          100.*pertrrho/rho,u,v
   20    format(' Temperature = ',F7.1,' K',10X,' Pressure = ',1p,E12.3,
     : ' N/m**2'/ ' Density (Low, Avg., High) =   ',3E12.3,' kg/m**3'/
     : ' Departure, COSPAR NH Mean =',0p,F12.1,' %',2(F10.1,' %')/
     : ' Density perturbation =        ',F7.2,' % of mean value'/
     : ' Eastward Wind = ',F7.1,' m/s       Northward Wind = ',F7.1,
     : ' m/s')
          write(nout,30)
   30     format(' -----------------------------------------------------',
     :   '----------------------')
c
c         Write some things to the output files
c
          if (nvx.eq.1) then
            outheight=sheight
          else
            outheight=absheight
          endif
          if (lgs.eq.0) then
            write(noutf,40) time,outheight,lat,lon,rho,t,u,v,
     :                 100.*sdrho/rho,pertr,ls
   40       format(f10.0,3f7.2,1pe9.2,0p3f7.1,3f6.1)
          endif
          if (lgs.eq.1) then
            write(noutf,41) time,outheight,lat,lon,alog10(rho),t,u,v,
     :                 100.*sdrho/rho,pertr,ls
          endif
          if (lgs.eq.2) then
            write(noutf,41) time,outheight,lat,lon,rcosav,t,u,v,
     :                 100.*sdrho/rho,pertr,ls
   41       format(f10.0,3f7.2,f9.2,3f7.1,3f6.1)
          endif
          if (nvx.eq.1) vx=sheight
          if (nvx.eq.2) vx=absheight
          if (nvx.eq.3) vx=lat
          if (nvx.eq.4) vx=lon
          if (nvx.eq.5) vx=time
          if (nvx.eq.6) vx=time/sollength
          if (nvx.eq.7) vx=ls
          if (nvx.eq.8) vx=localtime*15.
          if (lgs.eq.1) then
            rholo=alog10(rholo)
            rho=alog10(rho)
            rhohi=alog10(rhohi)
            p=alog10(p)
            pertr=alog10(pertr)
          endif
          if (lgs.eq.2) then
            rholo=rcoslo
            rho=rcosav
            rhohi=rcoshi
            pertr=pertr/rcos
          endif
          if (nvy.eq.0) then
            if (nvx.gt.2) then
              write(noutf+1,50) vx,rholo
              write(noutf+2,50) vx,rho
              write(noutf+3,50) vx,rhohi
              write(noutf+4,50) vx,pertrrho
              write(noutf+5,50) vx,T
              write(noutf+6,50) vx,p
              write(noutf+7,50) vx,u
              write(noutf+8,50) vx,v
              write(noutf+9,50) vx,Tmax
              write(noutf+10,50) vx,Tmin
              write(noutf+11,50) vx,Tsurf
              write(noutf+12,50) vx,rscaleheight
              write(noutf+13,50) vx,100.*sdrho/rho
              write(noutf+14,50) vx,pertr
              if (zbase.gt.0.) write(noutf+15,50) vx,tbase
              if (zbase.gt.0.) write(noutf+16,50) vx,zbase
           else
              write(noutf+1,50) rholo,vx
              write(noutf+2,50) rho,vx
              write(noutf+3,50) rhohi,vx
```

```
            write(noutf+4,50) sdrho,vx
            write(noutf+5,50) T,vx
            write(noutf+6,50) p,vx
            write(noutf+7,50) u,vx
            write(noutf+8,50) v,vx
            write(noutf+9,50) Tmax,vx
            write(noutf+10,50) Tmin,vx
            write(noutf+11,50) Tsurf,vx
            write(noutf+12,50) rscaleheight,vx
            write(noutf+13,50) 100.*sdrho/rho,vx
            write(noutf+14,50) pertr,vx
            if (zbase.gt.0.) write(noutf+15,50) tbase,vx
            if (zbase.gt.0.) write(noutf+16,50) zbase,vx
          endif
        else
          if(nvy.EQ.1) vy=sheight
          if(nvy.EQ.2) vy=absheight
          if(nvy.EQ.3) vy=lat
          if(nvy.EQ.4) vy=lon
          if(nvy.EQ.5) vy=time
          if(nvy.EQ.6) vy=time/88775.245
          if(nvy.EQ.7) vy=ls
          if(nvy.EQ.8) vy=localtime*15.
          write(noutf+1,60) vx,vy,rholo
          write(noutf+2,60) vx,vy,rho
          write(noutf+3,60) vx,vy,rhohi
          write(noutf+4,60) vx,vy,sdrho
          write(noutf+5,60) vx,vy,T
          write(noutf+6,60) vx,vy,p
          write(noutf+7,60) vx,vy,u
          write(noutf+8,60) vx,vy,v
          write(noutf+9,60) vx,vy,Tmax
          write(noutf+10,60) vx,vy,Tmin
          write(noutf+11,60) vx,vy,Tsurf
          write(noutf+12,60) vx,vy,rscaleheight
          write(noutf+13,60) vx,vy,100.*sdrho/rho
          write(noutf+14,60) vx,vy,pertr
          if (zbase.gt.0.) write(noutf+15,50) vx,vy,tbase
          if (zbase.gt.0.) write(noutf+16,50) vx,vy,zbase
        endif
 50     format(2g12.4)
 60     format(3g12.4)
c
        np=np+1
        prevlon=lon
        lon=lon+dlon
        if (abs(dlon).le.1.0e-20) lon=999. !terminate loop
       enddo
       prevlat=lat
       lat=lat+dlat
       if (abs(dlat).le.1.0e-20) lat=999. !terminate loop
      enddo
      height=height+dheight
      if (abs(dheight).le.1.0e-20) height=999. !terminate loop
     enddo
     date=date+dtime/8.6400d4
     time=time+dtime
    enddo
c
    return
    end
```

```
      subroutine eofpb(ueof,pertm,pertr,gasd,lon,lat,sigma,ls,sd,name)
c
c     Calculate an EOF perturbation on the 3d variable=name at
c     longitude=lon, latitude=lat sigma=sigma and day=day.
c
#include "drsdef.h"
c
c     input
c
      integer       ueof   !drs unit number for eof file
      real          gasd(6)!Gaussian deviates for a single profile
      real          lon    !longitude of point
      real          lat    !latitude of point
      real          sigma  !sigma of point
      real          ls     !Ls
      real          sd     !standard deviation of variable
      character*16  name   !Name of variable
c
c     output
c
      real    pertm        !perturbation corresponding to trend
      real    pertr        !perturbation corresponding to random component
c
c     local variables
c
      integer ierr,i,j
      real    norm
      character*16 dumname
      real    evals(6)
      real    dum
      real    ne
      real    pccof(4,6)
      real    nc
      real    evecs(6)
      real    pc
      real    day
      real    sumevals
      real    tmplat
      real    tmplon
c
c     calculate non-dim day [0,1] in the season
c
      day=amod(ls,30.)/30.
c
c     normalisation factor is the standard deviation
c
      norm=sd
c
c     check for latitude and longitude outside range
c
      tmplat=lat
      if (lat.le.-82.5) tmplat=-82.5
      if (lat.ge.77.5) tmplat=77.5
      tmplon=lon
      if (lon+180..ge.340.) tmplon=340.-180.
c
c     read eigenvalues
c
      do i=1,6
        ne=float(i)
        dumname='evals'
        call ner2d(ueof,dum,tmplon,tmplat,ne,dumname)
        evals(i)=dum
      enddo
c
c     read PC coefficients
c
      do i=1,6
        do j=1,4
          ne=float(i)
          nc=float(j)
          dumname='cofs'
          call ner3d(ueof,dum,tmplon,tmplat,nc,ne,dumname)
          pccof(j,i)=dum
        enddo
```

```
      enddo
c
c     read 6 eofs
c
      do i=1,6
        ne=float(i)
        if (name.eq.'ps') then
          call ner2d(ueof,dum,tmplon,tmplat,ne,name)
        else
          call ner3d(ueof,dum,tmplon,tmplat,sigma,ne,name)
        endif
        evecs(i)=dum
      enddo
c
c     calculate perturbation
c
      pertm=0.0
      pertr=0.0
      do i=1,6
c
c       polynomial fit
c
        pc=pccof(1,i)+pccof(2,i)*day+pccof(3,i)*day**2
        pertm=pertm+pc*evecs(i)
c
c       Gaussian random variable
c
        pc=pccof(4,i)*gasd(i)
        pertr=pertr+pc*evecs(i)
      enddo
c
c     note that only sum(eigenvalues) of the variance is explained by
c     the EOFs, so could add in some random noise.
c
c      sumevals=0.
c      do i=1,6
c        sumevals=sumevals+evals(i)
c      enddo
c      pertr=pertr+(1.-sumevals)*gasdev(seed+999)
c
c     renormalise
c
      pertm=pertm*norm
      pertr=pertr*norm
c
c     that's it!
c
      return
      end
```

```fortran
      function gasdev(idum)
c
c     Return a Gaussian deviate with zero mean and unit standard deviation
c
c     input
c
      integer idum    !Seed for random number generator
c
c     local
c
      data iset/0/
      real v1
      real v2
      real r
      real fac
      real gset
      if (iset.eq.0) then
1       v1=2.*ran(idum)-1.
        v2=2.*ran(idum)-1.
        r=v1**2+v2**2
        if (r.ge.1.) goto 1
        fac=sqrt(-2.*log(r)/r)
        gset=v1*fac
        gasdev=v2*fac
        iset=1
      else
        gasdev=gset
        iset=0
      endif
      return
      end
```

```
      subroutine getsi(udrs,sigma,lonw,lat,height,time,
     :                 levhi,levlow,levweight)
c
c     Find the nearest sigma levels and interpolation weights at
c     longitude=lonw, latitude=lat and height=height.
c
c     inputs
c
      integer udrs        !drs unit number
      real    sigma(25)   !sigma levels
      real    height      !height of point
      real    lonw        !west longitude
      real    lat         !latitude
      real    time        !Universal time
c
c     outputs
c
      integer levhi     !database level upper bound
      integer levlow    !database level lower bound
      real    levweight !level weight for interpolation
c
c     locals
c
      integer l
      real    dsigma(25)
      real    hsigma(26)
      real    sheight(25)
      real    g /3.72/
      real    R /191.1/
      character*16  name /'t'/
      real    t(25)
      real    lon
c
c     convert longitude west to longitude east
c
      lon=360.-lonw
c
c     Compute delta sigma
c
      hsigma(1)=1.
      do l=2,25
        hsigma(l)=.5*(sigma(l-1)+sigma(l))
      enddo
      hsigma(26)=0.
      do l=1,25
        dsigma(l)=hsigma(l)-hsigma(l+1)
      enddo
c
c     get temperature profile
c
      call profi(udrs,t,lon,lat,time,name)
c
c     Integrate hydrostatic equation (use km)
c
      sheight(1)=(R*t(1)*dsigma(1))/(1000.*g*sigma(1))
      do l=2,25
        sheight(l)=sheight(l-1)+(R*t(l)*dsigma(l))/(1000.*g*sigma(l))
      enddo
c
c     find levhi, levlow and levweight
c
      if (height.lt.sheight(1)) then
        levhi=1
        levlow=1
        levweight=1.
      endif
      do l=1,24
        if ((height.ge.sheight(l)).and.(height.lt.sheight(l+1))) then
          levhi=l+1
          levlow=l
          levweight=(height-sheight(levlow))
     :              /(sheight(levhi)-sheight(levlow))
        endif
      enddo
      if (height.ge.sheight(25)) then
```

```
  levhi=25
  levlow=25
  levweight=1.
endif
return
end
```

```
      subroutine grid4(lon,lat,dlon,dlat,t,u)
c
c     Given longitude=lon and latitude=lat find the nearest 4 horizontal
c     gridpoints in the database and bilinear interpolation weights.
c
c     Grid points are arranged as in Numerical Recipies page 117
c
c     4 3
c     1 2
c
c     inputs
c
      real lon       !east longitude of point
      real lat       !latitude of point
c
c     outputs
c
      real dlon(4)  !longitudes of database points
      real dlat(4)  !latitudes of database points
      real t        !weight
      real u        !weight
c
c     local
c
      real alon,alat
      integer i
      real deltalon /3.75/
      real deltalat /3.75/
      real ilon /0./
      real ilat /-88.125/
      integer nlon /96/
      integer nlat /48/
c
      alon=ilon
      do i=1,nlon-1
        if ((lon.ge.alon).and.(lon.lt.alon+deltalon)) then
          dlon(4)=alon
          dlon(3)=alon+deltalon
          dlon(1)=alon
          dlon(2)=alon+deltalon
          t=(lon-alon)/deltalon
        endif
        alon=alon+deltalon
      enddo
      if (lon.ge.alon) then
        dlon(4)=alon
        dlon(3)=ilon
        dlon(1)=alon
        dlon(2)=ilon
        t=(lon-alon)/deltalon
      endif
      alat=ilat
      if (lat.le.alat) then
        dlat(4)=alat
        dlat(3)=alat
        dlat(1)=alat
        dlat(2)=alat
        u=0.
      endif
      do i=1,nlat-1
        if ((lat.ge.alat).and.(lat.lt.alat+deltalat)) then
          dlat(4)=alat+deltalat
          dlat(3)=alat+deltalat
          dlat(1)=alat
          dlat(2)=alat
          u=(lat-alat)/deltalat
        endif
        alat=alat+deltalat
      enddo
      if (lat.ge.alat) then
        dlat(4)=alat
        dlat(3)=alat
        dlat(1)=alat
        dlat(2)=alat
        u=0.
```

```
endif
return
end
```

```
      subroutine grwpb(udrsm,lonw,lat,height,rho,u,v,time,dvardz,sigma,
     :                 dev,pert)
c
c     Small scale gravity wave perturbation model. Computes perturbation
c     on u,v,t or rho.
c
c     input
c
      integer udrsm !DRS file unit number
      real lonw     !west longitude of point
      real lat      !latitude of point
      real height   !height of point
      real rho      !density at height=height
      real u        !zonal wind at height=height
      real v        !meridional wind at height=height
      real time     !database time
      real dvardz   !d(variable)/dz in units/km
      real sigma(25)!sigma levels
      real dev      !random number between 0 and 1
c
c     output
c
      real pert     !perturbation
c
c     local
c
      real rho0
      real u0
      real v0
      real tmpheight
      integer levlow
      integer levhi
      real levweight
      real dz
      real sig
      real lamda /8./     !Vertical wavelength (km)
      real iphase         !Surface phase
      real pi /3.14159265359/
      character*16 name
      real lon
c
c     convert longitude west to longitude east
c
      lon=360.-lonw
c
c     get rho0, u0, v0
c
      tmpheight=0.
      call getsi(udrsm,sigma,lon,lat,tmpheight,time,
     :           levhi,levlow,levweight)
      name='u'
      call var3d(udrsm,u0,lon,lat,levhi,levlow,levweight,time,
     :           name)
      name='v'
      call var3d(udrsm,v0,lon,lat,levhi,levlow,levweight,time,
     :           name)
      name='rho'
      call var3d(udrsm,rho0,lon,lat,levhi,levlow,levweight,time,
     :           name)
c
c     get sub-grid scale variance
c
      name='substd'
      call mount(lon,lat,sig,name)
      sig=sig/1000.
c
c     compute delta z (N=N0 is implicit)
c
      dz=(rho0*sqrt(u0**2+v0**2)*sig)/(rho*sqrt(u**2+v**2))
      dz=sqrt(dz)
c
c     compute wave
c
      iphase=2.*pi*dev
      dz=dz*sin(iphase+(2.*pi*height)/lamda)
```

```
c
c     compute perturbation
c
      pert=dz*dvardz
c
      return
      end
```

```
      subroutine ltime(longitude,sunlon,localtime)
c
c     Compute local time at longitude=longitude when the sun is
c     at longitude=sunlon.
c
c     input
c
      real longitude !west longitude
      real sunlon    !west longitude of sun
c
c     output
c
      real localtime !local time
c
      localtime=12.+(sunlon-longitude)/15.
      if (localtime.lt.0.) localtime=localtime+24.
      if (localtime.gt.24.) localtime=localtime-24.
c
      return
      end
```

```
      subroutine max2d(udrs,a,lon,lat,name)
c
c     Retrieve the daily maximum value of 2-d variable=name at
c     longitude=lon, latitude=lat
c
#include "drsdef.h"
c
c     input
c
      integer       udrs   !drs unit number
      real          lon    !west longitude of point
      real          lat    !latitude of point
      character*16  name   !Name of variable
c
c     output
c
      real a               !maximum of variable
c
c     local variables
c
      real    x
      real    xmax
      real    time
c
      xmax=-1.e20
      do i=1,12
        time=float(i)
        call var2d(udrs,x,lon,lat,time,name)
        if (x.gt.xmax) xmax=x
      enddo
      a=xmax
      return
      end
```

```
      subroutine min2d(udrs,a,lon,lat,name)
c
c     Retrieve the daily minimum value of 2-d variable=name at
c     longitude=lon, latitude=lat
c
#include "drsdef.h"
c
c     input
c
      integer       udrs   !drs unit number
      real          lon    !longitude of point
      real          lat    !latitude of point
      character*16  name   !Name of variable
c
c     output
c
      real a               !minimum of variable
c
c     local variables
c
      real    x
      real    xmin
      real    time
      xmin=1.e20
      do i=1,12
        time=float(i)
        call var2d(udrs,x,lon,lat,time,name)
        if (x.lt.xmin) xmin=x
      enddo
      a=xmin
      return
      end
```

```
      subroutine mount(lon,lat,height,name)
c
c     Return orographic height or sub-grid scale standard deviation at
c     west longitiude=lon, latitude=lat.
c
#include "drsdef.h"
c
c     Inputs
c
      real lon           !longitude
      real lat           !latitude
      character*16 name !name of variable
c
c     Outputs
c
      real height        !value of variable
c
c     Local
c
      integer       udrs /64/
      integer       ierr
      real          time /1.0/
      character*256 mean_dic,mean_dat
c
      mean_dic='../data/mountain.dic'
      mean_dat='../data/mountain.dat'
      ierr = aslun(udrs,mean_dic,udrs+1,mean_dat,IDRS_READ)
      call var2d(udrs,height,lon,lat,time,name)
      ierr = cllun(udrs)
      return
      end
```

```
      subroutine ner2d(udrs,a,lonw,lat,time,name)
c
c     Retrieve the value of 2-d variable=name at the nearest MCD grid point
c     to west longitude=lon, latitude=lat and universal time=time.
c
#include "drsdef.h"
c
c     input
c
      integer       udrs   !drs unit number
      real          lonw   !west longitude of point
      real          lat    !latitude of point
      real          time   !Global time
      character*16  name   !Name of variable
c
c     output
c
      real a(1,1,1)        !Value of variable at nearest grid point
c
c     local variables
c
      integer ierr,i
      integer order(3)
      integer dim(3)
      real    fe(3)
      real    le(3)
      real    cycle(3)
      real    lon
c
c     convert longitude west to longitude east
c
      lon=360.-lonw
c
      do i=1,3
        order(i)=i
        cycle(i)=0.0
        dim(i)=1
      enddo
      cycle(1)=360.0
      fe(1)=lon
      le(1)=lon
      fe(2)=lat
      le(2)=lat
      if (lat.le.-88.125) then
        fe(2)=-88.125
        le(2)=-88.125
      endif
      if (lat.ge.88.125) then
        fe(2)=88.125
        le(2)=88.125
      endif
      fe(3)=time
      le(3)=time
      ierr = cluvdb()
      ierr = setname(' ',name,' ',' ',' ')
      ierr = getslab(udrs,3,order,fe,le,cycle,a,dim)
      end
```

```fortran
      subroutine ner3d(udrs,a,lonw,lat,sigma,time,name)
c
c     Retrieve the value of 3-d variable=name at the nearest MCD grid point
c     to west longitude=lon, latitude=lat
c     sigma=sigma and universal time=time.
c
#include "drsdef.h"
c
c     input
c
      integer       udrs   !drs unit number
      real          lonw   !west longitude of point
      real          lat    !latitude of point
      real          sigma  !sigma of point
      real          time   !Global time
      character*16  name   !Name of variable
c
c     output
c
      real    a(1,1,1,1)   !Value of variable at nearest grid point
c
c     local variables
c
      integer ierr,i
      integer order(4)
      integer dim(4)
      real    fe(4)
      real    le(4)
      real    cycle(4)
      real    tmplat
      real    lon
c
c     convert longitude west to longitude east
c
      lon=360.-lonw
c
      do i=1,4
        order(i)=i
        cycle(i)=0.0
        dim(i)=1
      enddo
      cycle(1)=360.0
      fe(1)=lon
      le(1)=lon
      fe(2)=lat
      le(2)=lat
      if (lat.le.-88.125) then
        fe(2)=-88.125
        le(2)=-88.125
      endif
      if (lat.ge.88.125) then
        fe(2)=88.125
        le(2)=88.125
      endif
      fe(3)=sigma
      le(3)=sigma
      fe(4)=time
      le(4)=time
      ierr = cluvdb()
      ierr = setname(' ',name,' ',' ',' ')
      ierr = getslab(udrs,4,order,fe,le,cycle,a,dim)
      end
```

```fortran
      subroutine opend(udrsm,udrssd,ueof,ls,dust)
c
c     Open the appropriate DRS file corrseponding to Ls=ls and
c     dust senario=dust.
c
#include "drsdef.h"
c
c     Input
c
      integer udrsm    !DRS channel number for mean fields
      integer udrssd   !DRS channel number for standard deviation fields
      integer ueof     !DRS channel number for EOF fields
      real    ls       !Ls
      integer dust     !Dust senario
c
c     Local
c
      integer ierr
      character*255 datfile
      character*255 dicfile
c
c     Close files
c
      ierr=cllun(udrsm)
      ierr=cllun(udrssd)
      ierr=cllun(ueof)
c
c     Mean file
c
      if (dust.eq.1) then
        if ((ls.ge.0.0).and.(ls.le.30)) then
          datfile='../data/viks01me.dat'
          dicfile='../data/viks01me.dic'
        endif
        if ((ls.ge.30.0).and.(ls.le.60)) then
          datfile='../data/viks02me.dat'
          dicfile='../data/viks02me.dic'
        endif
        if ((ls.ge.60.0).and.(ls.le.90)) then
          datfile='../data/viks03me.dat'
          dicfile='../data/viks03me.dic'
        endif
        if ((ls.ge.90.0).and.(ls.le.120)) then
          datfile='../data/viks04me.dat'
          dicfile='../data/viks04me.dic'
        endif
        if ((ls.ge.120.0).and.(ls.le.150)) then
          datfile='../data/viks05me.dat'
          dicfile='../data/viks05me.dic'
        endif
        if ((ls.ge.150.0).and.(ls.le.180)) then
          datfile='../data/viks06me.dat'
          dicfile='../data/viks06me.dic'
        endif
        if ((ls.ge.180.0).and.(ls.le.210)) then
          datfile='../data/viks07me.dat'
          dicfile='../data/viks07me.dic'
        endif
        if ((ls.ge.210.0).and.(ls.le.240)) then
          datfile='../data/viks08me.dat'
          dicfile='../data/viks08me.dic'
        endif
        if ((ls.ge.240.0).and.(ls.le.270)) then
          datfile='../data/viks09me.dat'
          dicfile='../data/viks09me.dic'
        endif
        if ((ls.ge.270.0).and.(ls.le.300)) then
          datfile='../data/viks10me.dat'
          dicfile='../data/viks10me.dic'
        endif
        if ((ls.ge.300.0).and.(ls.le.330)) then
          datfile='../data/viks11me.dat'
          dicfile='../data/viks11me.dic'
        endif
        if ((ls.ge.330.0).and.(ls.le.360)) then
```

```
    datfile='../data/viks12me.dat'
    dicfile='../data/viks12me.dic'
  endif
endif
if (dust.eq.2) then
  if ((ls.ge.0.0).and.(ls.le.30)) then
    datfile='../data/lows01me.dat'
    dicfile='../data/lows01me.dic'
  endif
  if ((ls.ge.30.0).and.(ls.le.60)) then
    datfile='../data/lows02me.dat'
    dicfile='../data/lows02me.dic'
  endif
  if ((ls.ge.60.0).and.(ls.le.90)) then
    datfile='../data/lows03me.dat'
    dicfile='../data/lows03me.dic'
  endif
  if ((ls.ge.90.0).and.(ls.le.120)) then
    datfile='../data/lows04me.dat'
    dicfile='../data/lows04me.dic'
  endif
  if ((ls.ge.120.0).and.(ls.le.150)) then
    datfile='../data/lows05me.dat'
    dicfile='../data/lows05me.dic'
  endif
  if ((ls.ge.150.0).and.(ls.le.180)) then
    datfile='../data/lows06me.dat'
    dicfile='../data/lows06me.dic'
  endif
  if ((ls.ge.180.0).and.(ls.le.210)) then
    datfile='../data/lows07me.dat'
    dicfile='../data/lows07me.dic'
  endif
  if ((ls.ge.210.0).and.(ls.le.240)) then
    datfile='../data/lows08me.dat'
    dicfile='../data/lows08me.dic'
  endif
  if ((ls.ge.240.0).and.(ls.le.270)) then
    datfile='../data/lows09me.dat'
    dicfile='../data/lows09me.dic'
  endif
  if ((ls.ge.270.0).and.(ls.le.300)) then
    datfile='../data/lows10me.dat'
    dicfile='../data/lows10me.dic'
  endif
  if ((ls.ge.300.0).and.(ls.le.330)) then
    datfile='../data/lows11me.dat'
    dicfile='../data/lows11me.dic'
  endif
  if ((ls.ge.330.0).and.(ls.le.360)) then
    datfile='../data/lows12me.dat'
    dicfile='../data/lows12me.dic'
  endif
endif
if (dust.eq.3) then
  if ((ls.ge.210).and.(ls.le.240)) then
    datfile='../data/ds2s08me.dat'
    dicfile='../data/ds2s08me.dic'
  endif
  if ((ls.ge.240).and.(ls.le.270)) then
    datfile='../data/ds2s09me.dat'
    dicfile='../data/ds2s09me.dic'
  endif
  if ((ls.ge.270).and.(ls.le.300)) then
    datfile='../data/ds2s10me.dat'
    dicfile='../data/ds2s10me.dic'
  endif
endif
if (dust.eq.4) then
  if ((ls.ge.210).and.(ls.le.240)) then
    datfile='../data/ds5s08me.dat'
    dicfile='../data/ds5s08me.dic'
  endif
  if ((ls.ge.240).and.(ls.le.270)) then
    datfile='../data/ds5s09me.dat'
```

```
          dicfile='../data/ds5s09me.dic'
        endif
        if ((ls.ge.270).and.(ls.le.300)) then
          datfile='../data/ds5s10me.dat'
          dicfile='../data/ds5s10me.dic'
        endif
      endif
      ierr=aslun(udrsm,dicfile,udrsm+1,datfile,IDRS_READ)
c
c     Standard Deviation file
c
      if (dust.eq.1) then
        if ((ls.ge.0.0).and.(ls.le.30)) then
          datfile='../data/viks01sd.dat'
          dicfile='../data/viks01sd.dic'
        endif
        if ((ls.ge.30.0).and.(ls.le.60)) then
          datfile='../data/viks02sd.dat'
          dicfile='../data/viks02sd.dic'
        endif
        if ((ls.ge.60.0).and.(ls.le.90)) then
          datfile='../data/viks03sd.dat'
          dicfile='../data/viks03sd.dic'
        endif
        if ((ls.ge.90.0).and.(ls.le.120)) then
          datfile='../data/viks04sd.dat'
          dicfile='../data/viks04sd.dic'
        endif
        if ((ls.ge.120.0).and.(ls.le.150)) then
          datfile='../data/viks05sd.dat'
          dicfile='../data/viks05sd.dic'
        endif
        if ((ls.ge.150.0).and.(ls.le.180)) then
          datfile='../data/viks06sd.dat'
          dicfile='../data/viks06sd.dic'
        endif
        if ((ls.ge.180.0).and.(ls.le.210)) then
          datfile='../data/viks07sd.dat'
          dicfile='../data/viks07sd.dic'
        endif
        if ((ls.ge.210.0).and.(ls.le.240)) then
          datfile='../data/viks08sd.dat'
          dicfile='../data/viks08sd.dic'
        endif
        if ((ls.ge.240.0).and.(ls.le.270)) then
          datfile='../data/viks09sd.dat'
          dicfile='../data/viks09sd.dic'
        endif
        if ((ls.ge.270.0).and.(ls.le.300)) then
          datfile='../data/viks10sd.dat'
          dicfile='../data/viks10sd.dic'
        endif
        if ((ls.ge.300.0).and.(ls.le.330)) then
          datfile='../data/viks11sd.dat'
          dicfile='../data/viks11sd.dic'
        endif
        if ((ls.ge.330.0).and.(ls.le.360)) then
          datfile='../data/viks12sd.dat'
          dicfile='../data/viks12sd.dic'
        endif
      endif
      if (dust.eq.2) then
        if ((ls.ge.0.0).and.(ls.le.30)) then
          datfile='../data/lows01sd.dat'
          dicfile='../data/lows01sd.dic'
        endif
        if ((ls.ge.30.0).and.(ls.le.60)) then
          datfile='../data/lows02sd.dat'
          dicfile='../data/lows02sd.dic'
        endif
        if ((ls.ge.60.0).and.(ls.le.90)) then
          datfile='../data/lows03sd.dat'
          dicfile='../data/lows03sd.dic'
        endif
        if ((ls.ge.90.0).and.(ls.le.120)) then
```

```
          datfile='../data/lows04sd.dat'
          dicfile='../data/lows04sd.dic'
        endif
        if ((ls.ge.120.0).and.(ls.le.150)) then
          datfile='../data/lows05sd.dat'
          dicfile='../data/lows05sd.dic'
        endif
        if ((ls.ge.150.0).and.(ls.le.180)) then
          datfile='../data/lows06sd.dat'
          dicfile='../data/lows06sd.dic'
        endif
        if ((ls.ge.180.0).and.(ls.le.210)) then
          datfile='../data/lows07sd.dat'
          dicfile='../data/lows07sd.dic'
        endif
        if ((ls.ge.210.0).and.(ls.le.240)) then
          datfile='../data/lows08sd.dat'
          dicfile='../data/lows08sd.dic'
        endif
        if ((ls.ge.240.0).and.(ls.le.270)) then
          datfile='../data/lows09sd.dat'
          dicfile='../data/lows09sd.dic'
        endif
        if ((ls.ge.270.0).and.(ls.le.300)) then
          datfile='../data/lows10sd.dat'
          dicfile='../data/lows10sd.dic'
        endif
        if ((ls.ge.300.0).and.(ls.le.330)) then
          datfile='../data/lows11sd.dat'
          dicfile='../data/lows11sd.dic'
        endif
        if ((ls.ge.330.0).and.(ls.le.360)) then
          datfile='../data/lows12sd.dat'
          dicfile='../data/lows12sd.dic'
        endif
      endif
      if (dust.eq.3) then
        if ((ls.ge.210).and.(ls.le.240)) then
          datfile='../data/ds2s08sd.dat'
          dicfile='../data/ds2s08sd.dic'
        endif
        if ((ls.ge.240).and.(ls.le.270)) then
          datfile='../data/ds2s09sd.dat'
          dicfile='../data/ds2s09sd.dic'
        endif
        if ((ls.ge.270).and.(ls.le.300)) then
          datfile='../data/ds2s10sd.dat'
          dicfile='../data/ds2s10sd.dic'
        endif
      endif
      if (dust.eq.4) then
        if ((ls.ge.210).and.(ls.le.240)) then
          datfile='../data/ds5s08sd.dat'
          dicfile='../data/ds5s08sd.dic'
        endif
        if ((ls.ge.240).and.(ls.le.270)) then
          datfile='../data/ds5s09sd.dat'
          dicfile='../data/ds5s09sd.dic'
        endif
        if ((ls.ge.270).and.(ls.le.300)) then
          datfile='../data/ds5s10sd.dat'
          dicfile='../data/ds5s10sd.dic'
        endif
      endif
      ierr=aslun(udrssd,dicfile,udrssd+1,datfile,IDRS_READ)
c
c     eof file
c
      if (dust.eq.1) then
        if ((ls.ge.0.0).and.(ls.le.30)) then
          datfile='../data/viks01eo.dat'
          dicfile='../data/viks01eo.dic'
        endif
        if ((ls.ge.30.0).and.(ls.le.60)) then
          datfile='../data/viks02eo.dat'
```

```
        dicfile='../data/viks02eo.dic'
      endif
      if ((ls.ge.60.0).and.(ls.le.90)) then
        datfile='../data/viks03eo.dat'
        dicfile='../data/viks03eo.dic'
      endif
      if ((ls.ge.90.0).and.(ls.le.120)) then
        datfile='../data/viks04eo.dat'
        dicfile='../data/viks04eo.dic'
      endif
      if ((ls.ge.120.0).and.(ls.le.150)) then
        datfile='../data/viks05eo.dat'
        dicfile='../data/viks05eo.dic'
      endif
      if ((ls.ge.150.0).and.(ls.le.180)) then
        datfile='../data/viks06eo.dat'
        dicfile='../data/viks06eo.dic'
      endif
      if ((ls.ge.180.0).and.(ls.le.210)) then
        datfile='../data/viks07eo.dat'
        dicfile='../data/viks07eo.dic'
      endif
      if ((ls.ge.210.0).and.(ls.le.240)) then
        datfile='../data/viks08eo.dat'
        dicfile='../data/viks08eo.dic'
      endif
      if ((ls.ge.240.0).and.(ls.le.270)) then
        datfile='../data/viks09eo.dat'
        dicfile='../data/viks09eo.dic'
      endif
      if ((ls.ge.270.0).and.(ls.le.300)) then
        datfile='../data/viks10eo.dat'
        dicfile='../data/viks10eo.dic'
      endif
      if ((ls.ge.300.0).and.(ls.le.330)) then
        datfile='../data/viks11eo.dat'
        dicfile='../data/viks11eo.dic'
      endif
      if ((ls.ge.330.0).and.(ls.le.360)) then
        datfile='../data/viks12eo.dat'
        dicfile='../data/viks12eo.dic'
      endif
    endif
    if (dust.eq.2) then
      if ((ls.ge.0.0).and.(ls.le.30)) then
        datfile='../data/lows01eo.dat'
        dicfile='../data/lows01eo.dic'
      endif
      if ((ls.ge.30.0).and.(ls.le.60)) then
        datfile='../data/lows02eo.dat'
        dicfile='../data/lows02eo.dic'
      endif
      if ((ls.ge.60.0).and.(ls.le.90)) then
        datfile='../data/lows03eo.dat'
        dicfile='../data/lows03eo.dic'
      endif
      if ((ls.ge.90.0).and.(ls.le.120)) then
        datfile='../data/lows04eo.dat'
        dicfile='../data/lows04eo.dic'
      endif
      if ((ls.ge.120.0).and.(ls.le.150)) then
        datfile='../data/lows05eo.dat'
        dicfile='../data/lows05eo.dic'
      endif
      if ((ls.ge.150.0).and.(ls.le.180)) then
        datfile='../data/lows06eo.dat'
        dicfile='../data/lows06eo.dic'
      endif
      if ((ls.ge.180.0).and.(ls.le.210)) then
        datfile='../data/lows07eo.dat'
        dicfile='../data/lows07eo.dic'
      endif
      if ((ls.ge.210.0).and.(ls.le.240)) then
        datfile='../data/lows08eo.dat'
        dicfile='../data/lows08eo.dic'
```

```
      endif
      if ((ls.ge.240.0).and.(ls.le.270)) then
        datfile='../data/lows09eo.dat'
        dicfile='../data/lows09eo.dic'
      endif
      if ((ls.ge.270.0).and.(ls.le.300)) then
        datfile='../data/lows10eo.dat'
        dicfile='../data/lows10eo.dic'
      endif
      if ((ls.ge.300.0).and.(ls.le.330)) then
        datfile='../data/lows11eo.dat'
        dicfile='../data/lows11eo.dic'
      endif
      if ((ls.ge.330.0).and.(ls.le.360)) then
        datfile='../data/lows12eo.dat'
        dicfile='../data/lows12eo.dic'
      endif
    endif
    if (dust.eq.3) then
      if ((ls.ge.210).and.(ls.le.240)) then
        datfile='../data/ds2s08eo.dat'
        dicfile='../data/ds2s08eo.dic'
      endif
      if ((ls.ge.240).and.(ls.le.270)) then
        datfile='../data/ds2s09eo.dat'
        dicfile='../data/ds2s09eo.dic'
      endif
      if ((ls.ge.270).and.(ls.le.300)) then
        datfile='../data/ds2s10eo.dat'
        dicfile='../data/ds2s10eo.dic'
      endif
    endif
    if (dust.eq.4) then
      if ((ls.ge.210).and.(ls.le.240)) then
        datfile='../data/ds5s08eo.dat'
        dicfile='../data/ds5s08eo.dic'
      endif
      if ((ls.ge.240).and.(ls.le.270)) then
        datfile='../data/ds5s09eo.dat'
        dicfile='../data/ds5s09eo.dic'
      endif
      if ((ls.ge.270).and.(ls.le.300)) then
        datfile='../data/ds5s10eo.dat'
        dicfile='../data/ds5s10eo.dic'
      endif
    endif
    ierr=aslun(ueof,dicfile,ueof+1,datfile,IDRS_READ)
    return
    end
```

```
      subroutine orbit(date,sunlat,sunlon,ls,marsau)
c
c     Given Julian date solves the equation of time to give
c     solar longitude and latitude, Ls and distance of Mars
c     from the sun in AU
c
c     inputs
c
      real*8 date   !Julian date
c
c     outputs
c
      real   sunlat !subsolar latitude
      real   sunlon !subsolar longitude
      real   ls     !Ls
      real   marsau !Sun-Mars distance in AU
c
c     local
c
      real*8 modelday
      real*8 marsday /88775.245/
      real*8 earthday /86400./
      real   dummy
      real*8 mdoffset /222.4/
      real   slonoff /197.2/
      real   inclin /25.3/
      real   pi /3.14159/
c
c     convert Julian day to model day
c
      modelday=date*earthday/marsday
      modelday=dmod(modelday,669.d0)+mdoffset
      dummy=modelday
      call sol2ls(dummy,ls)
      sunlon=360.*(dummy-float(int(dummy)))+slonoff
      sunlon=amod(sunlon,360.)
      sunlat=inclin*sin(ls*pi/180.)
      marsau=1.5237*7.73396*((1.-0.934**2))
     :       /(1.+0.0934*cos((ls+109.)*pi/180.))
      return
      end
```

```
      subroutine profi(udrs,a,lon,lat,time,name)
c
c     Retrieve a profile of 3-d variable=name at longitude=lon, latitude=lat
c     and universal time=time.
c
#include "drsdef.h"
c
c     input
c
      integer       udrs   !drs unit number
      real          lon    !east longitude of point
      real          lat    !latitude of point
      real          time   !Global time
      character*16  name   !Name of variable
c
c     output
c
      real    a(25)        !Profile
c
c     local variables
c
      real    b(1,1,25,1)
      integer ierr,i,l
      integer order(4)
      integer dim(4)
      real    fe(4)
      real    le(4)
      real    cycle(4)
      real    dlon(4)
      real    dlat(4)
      real    y(25,4)
      real    t
      real    u
c
c     find nearest 4 grid points
c
      call grid4(lon,lat,dlon,dlat,t,u)
c
c     get 4 profiles at those points
c
      do i=1,4
        order(i)=i
        cycle(i)=0.0
        dim(i)=1
      enddo
      dim(3)=25
      cycle(1)=360.0
      fe(3)=0.9995000
      le(3)=0.0000563
      fe(4)=time
      le(4)=time
      do i=1,4
        fe(1)=dlon(i)
        le(1)=dlon(i)
        fe(2)=dlat(i)
        le(2)=dlat(i)
        if (dlat(i).le.-88.125) then
          fe(2)=-88.125
          le(2)=-88.125
        endif
        if (dlat(i).ge.88.125) then
          fe(2)=88.125
          le(2)=88.125
        endif
        ierr = cluvdb()
        ierr = setname(' ',name,' ',' ',' ')
        ierr = getslab(udrs,4,order,fe,le,cycle,b,dim)
        do l=1,25
          y(l,i)=b(1,1,l,1)
        enddo
      enddo
c
c     bilinear interpolation
c
      do l=1,25
```

```
      a(l)=(1.-t)*(1.-u)*y(l,1)
:          +t*(1.-u)*y(l,2)
:          +t*u*y(l,3)
:          +(1.-t)*u*y(l,4)
 enddo
 return
 end
```

```
      subroutine sol2ls(sol,ls)
c
c  Convert day number, sol, to solar longitude, Ls, where sol=0 is the
c  spring equinox.

      implicit none

c  Arguments:
      real sol
      real ls

c  Local:
      real year_day /668.6/
      real peri_day /485.0/
      real timeperi /1.905637/
      real e_elips  /0.093358/
      real pi       /3.1415927/
      real degrad   /57.295779/
      real zanom,xref,zx0,zdx,zteta,zz
      integer iter

      zz=(sol-peri_day)/year_day
      zanom=2.*pi*(zz-nint(zz))
      xref=abs(zanom)

c  The equation of time zx0 - e * sin (zx0) = xref, solved by Newton
      zx0=xref+e_elips*sin(xref)
      do 110 iter=1,10
         zdx=-(zx0-e_elips*sin(zx0)-xref)/(1.-e_elips*cos(zx0))
         if(abs(zdx).le.(1.e-7)) goto 120
         zx0=zx0+zdx
  110 continue
  120 continue
      zx0=zx0+zdx
      if(zanom.lt.0.) zx0=-zx0

      zteta=2.*atan(sqrt((1.+e_elips)/(1.-e_elips))*tan(zx0/2.))
      ls=zteta-timeperi
      if(ls.lt.0.) ls=ls+2.*pi
      if(ls.gt.2.*pi) ls=ls-2.*pi
      ls=degrad*ls
      return
      end
```

```
      subroutine var2d(udrs,a,lonw,lat,time,name)
c
c     Retrieve the value of 2-d variable=name at longitude=lon, latitude=lat
c     and universal time=time. Use bilinear interpolation to get user
c     specified longitude and latitude.
c
#include "drsdef.h"
c
c     input
c
      integer       udrs   !drs unit number
      real          lonw   !longitude (west) of point
      real          lat    !latitude of point
      real          time   !Global time
      character*16  name   !Name of variable
c
c     output
c
      real a(1,1,1)
c
c     local variables
c
      integer ierr,i
      integer order(3)
      integer dim(3)
      real    fe(3)
      real    le(3)
      real    cycle(3)
      real    dlon(4)
      real    dlat(4)
      real    y(4)
      real    t
      real    u
      real    lon          !longitdue east
c
c     convert longitude west to longitude east
c
      lon=360.-lonw
c
c     find nearest 4 grid points
c
      call grid4(lon,lat,dlon,dlat,t,u)
c
      do i=1,3
        order(i)=i
        cycle(i)=0.0
        dim(i)=1
      enddo
      cycle(1)=360.0
      fe(3)=time
      le(3)=time
      do i=1,4
        fe(1)=dlon(i)
        le(1)=dlon(i)
        fe(2)=dlat(i)
        le(2)=dlat(i)
        if (dlat(i).le.-88.125) then
          fe(2)=-88.125
          le(2)=-88.125
        endif
        if (dlat(i).ge.88.125) then
          fe(2)=88.125
          le(2)=88.125
        endif
        ierr = cluvdb()
        ierr = setname(' ',name,' ',' ',' ')
        ierr = getslab(udrs,3,order,fe,le,cycle,a,dim)
        y(i)=a(1,1,1)
      enddo
c
c     bilinear interpolation
c
      a(1,1,1)=(1.-t)*(1.-u)*y(1)
     :        +t*(1.-u)*y(2)
     :        +t*u*y(3)
```

```
:        +(1.-t)*u*y(4)
 return
 end
```

```fortran
      subroutine var3d(udrs,a,lonw,lat,levhi,levlow,levweight,time,name,
     :                 varflag)
c
c     Retrieve the value of 3-d variable=name at longitude=lon, latitude=lat
c     and universal time=time. The height of the variable is controlled by
c     levhi, levlow and levweight.  Use bilinear interpolation to get user
c     specified longitude and latitude.
c
#include "drsdef.h"
c
c     input
c
      integer       udrs      !drs unit number
      real          lonw      !longitude (west) of point
      real          lat       !latitude of point
      integer       levhi     !database level upper bound
      integer       levlow    !database level lower bound
      real          levweight !level weight for interpolation
      real          time      !Global time
      character*16  name      !Name of variable
      integer       varflag   !Variability model flag
c
c     output
c
      real    a               !value of variable
c
c     local variables
c
      real    profile(25)
      real    lon
c
c     convert longitude west to longitude east
c
      lon=360.-lonw
c
      call profi(udrs,profile,lon,lat,time,name)
      a=profile(levlow)+(profile(levhi)-profile(levlow))*levweight
      return
      end
```

```fortran
      subroutine vgrad(udrsm,lon,lat,height,time,sigma,dvardz,name)
c
c     Estimate the vertical gradient of a variable using simple finite
c     difference
c
c     inputs
c
      integer udrsm     !DRS file unit number
      real lon          !longitude of point
      real lat          !latitude of point
      real height       !height of point
      real time         !database time
      real sigma(25)    !sigma levels
      character*16 name !variable name
c
c     outputs
c
      real dvardz       !d(variable)/dz in units/km
c
c     local
c
      real varhi
      real varlow
      integer levlow
      integer levhi
      real levweight
      real tmpheight
c
c     get value of variable at height+2 and height-2
c
      tmpheight=height-2.
      call getsi(udrsm,sigma,lon,lat,tmpheight,time,
     :           levhi,levlow,levweight)
      call var3d(udrsm,varlow,lon,lat,levhi,levlow,levweight,time,
     :           name)
      tmpheight=height+2.
      call getsi(udrsm,sigma,lon,lat,tmpheight,time,
     :           levhi,levlow,levweight)
      call var3d(udrsm,varhi,lon,lat,levhi,levlow,levweight,time,
     :           name)
      dvardz=(varhi-varlow)/2.
      return
      end
```

```
#
# Make file for Mars Climat Database MarsGRAM style interface - MCDGM
#
MCDGM   =       mcdgm.f \
                atime.f \
                cospa.f \
                ctime.f \
                dstep.f \
                eofpb.f \
                gasdv.f \
                getsi.f \
                grid4.f \
                grwpb.f \
                ltime.f \
                max2d.f \
                mgthm.f \
                min2d.f \
                mount.f \
                ner2d.f \
                ner3d.f \
                opend.f \
                orbit.f \
                profi.f \
                solls.f \
                var2d.f \
                var3d.f \
                vgrad.f

DRS_INC = /home/nergal/gfd/lgm/lib

LIB_DIR = /home/nergal/gfd/lgm/lib

LIBS    = -L$(LIB_DIR) -ldrs

INCS    = -I$(DRS_INC)

FC      = f77

FLAGS   = -cpp

mcdgm:          $(MCDGM)
        $(FC) $(FLAGS) $(LIBS) $(INCS) -o $@ $(MCDGM)
```