```fortran
      program Mass

cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
c Chiara Ferrari
c April 2003
c
c Mvir=3*pi*Rh*(sigmaV)**2/G (expressed in Msun*h^(-1))
c
c sigmaV = line of sight velocity dispersion of the galaxies taken as the
c          biweight scale estimate (expressed in km/s)
c
c Rh = mean projected harmonic radius (expressed in km) =
c (Ngal*(Ngal-1)*Dlum)/(sum(i<j)1/(angular distance between galaxies))
c *(2*(1+z)^2)
c N.B.B. angular distance has to be expresses in radians!
c
c OR:
c
c Rh = ring-wise projected harmonic mean radius (expressed in km)
c
c Mpm=(10.2/G*(N-1.5))*(sum(i)(deltav)**2*Ri
c
c Ri=distance from the cluster center expressed in km
c
c G has to be expressed as km^3*(kg Msun)^(-1)*s^(-2)
c
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

      real*8 pi, G, cvelo, Mpc
      parameter (pi=3.14159265358979)
      parameter (G=6.67259*1.9891E+10)
      parameter (cvelo=2.99792458E+05)
      parameter (Mpc=3.0857E+19)

      integer Ntot, nin
      real*8 al(1000), del(1000), cz(1000), errcz(1000)
      real*8 xdata(1000), xerr(1000)
      real*8 XLBIWT,XSBIWT,XLBIWT1,XSBIWT1
      real*8 vmean, sigma

      real*8 sommaR, distance(1000,1000), dist, Rmh
      real H0,OmegaM,OmegaL,w,z,Dlum,Vol
      character*40 input, output

      real*8 fi
      parameter (fi=3.14159265358979/2.)
      real*8 alpha0, delta0, Ri(1000), k(1000,1000), kappa(1000,1000)
      real*8 ellf, Sk

      real *8 Spm, Rikm(1000)

      real*8 Mv, Mvrw, Mpm
```

```
c Input and output files:

      write (6,*) 'Introduce the name of the input file:'
      write (6,*) '(alpha [deg] - delta [deg]- cz - error_cz)'
      read (5,*) input

      write(6,*) 'Introduce the name of the output file:'
      write(6,*) '(with the different mass estimates)'
      read (5,*) output

      write(6,*) 'Introduce the coordinates of the cluster center:'
      write (6,*) '(alpha [deg] - delta [deg])'
      read(5,*) alpha0, delta0

      open (unit=10,file=input,status='old')

      nin=0
      do 100, i=1,1000

         read (10,*,end=15) al(i), del(i), cz(i), errcz(i)
         nin=nin+1

  100  continue

c Velocity location using the biweight estimator.

15      xlbiwt = zero

      Ntot=0
      do 200, j=1,nin

         Ntot=Ntot+1
         xerr(Ntot)=errcz(j)
         xdata(Ntot)=cz(j)

200  continue

      CALL XBIWT (xdata,Ntot,XLBIWT,XSBIWT,XLBIWT1,XSBIWT1)

c vmean is the sample location obtained with the biweight estimator

      vmean=XLBIWT

c Estimate of the distance of each galaxy from the cluster center:

      do 220, i=1,nin

         Ri(i)=dist(alpha0,delta0,al(i),del(i))
         Ri(i)=abs(Ri(i))

220  continue

c Estimate of Luminosity Distance:
```

```
      write (6,9)
  9   format (/,
     @  ' NOTE: THIS IS A BETA VERSION! ',/,
     @  ' Please for any comments, bugs or problems report to:',/,
     @  ' cappi@bo.astro.it',/)

       write (6,10)
10    format (' H0, Omega Matter, Omega Lambda: ',$)
       read (5,*) H0, OmegaM, OmegaL

       w=0
       if (OmegaL.gt.0) then
        write (6,16)
16     format (' w [w=-1 for Lambda, -1<w<0 for quintessence]: ',$)
        read (5,*) w
       end if

       z=vmean/cvelo

      CALL CosmoDist (H0,OmegaM,OmegaL,w,z,Dlum,Vol)

      write (2,36) H0, OmegaM,OmegaL,w
36   format (' H0: ',f6.2,' OmegaM: ',f7.3,' OmegaL: ',f7.3,
     @         ' w: ',f6.2,/,
     @         '      z           Dlum          Volume')

      write (2,*) z, Dlum, Vol
      write (6,*) z, Dlum, Vol
      write (6,*) ' Output file name: fort.2 '

c Here we make the cosmological and relativistic correction for radial
c velocities.

     do 300 i=1,nin
300    xdata(i)=(xdata(i)-xlbiwt)/(1.+xlbiwt/cvelo)

     CALL XBIWT (xdata,Ntot,XLBIWT,XSBIWT,XLBIWT1,XSBIWT1)

c sigma is the sample scale obtained with the biweight estimator

     sigma=XSBIWT

c We begin the estimate of the mean harmonic radius:

     sommaR=0

     do 400, i=1,nin
        do 500, j=i+1,nin

           distance(i,j)=dist(al(j),del(j),al(i),del(i))
           distance(i,j)=abs(distance(i,j))

           sommaR=sommaR+1/distance(i,j)
```

```
500      continue
400   continue

c Mean harmonic radius expressed in km:

      Rmh=Mpc*(nin*(nin-1)*Dlum)/(sommaR*2*(1+vmean/cvelo)**2)

c Compute the ring-wise projected harmonic mean radius:

      Sk=0

      do 450, i=1,nin
         do 550, j=i+1,nin

            k(i,j)=sqrt(4*Ri(i)*Ri(j)/(Ri(i)+Ri(j))**2)
            kappa(i,j)=ellf(fi,k(i,j))
            Sk=Sk+2*kappa(i,j)/(pi*(Ri(i)+Ri(j)))

550      continue
450   continue

c Ring-wise projected harmonic mean radius expressed in km:

      Rrw=Mpc*(nin*(nin-1)*Dlum)/(Sk*2*(1+vmean/cvelo)**2)

c Estimate of the virial mass in solar mass with:
c     - mean harmonic radius
c     - ring-wise radius

      Mv=3*pi*sigma*sigma*Rmh/G
      Mvrw=3*pi*sigma*sigma*Rrw/G

c Projected mass estimator (the difference between the radial velocities and
c the mean cluster redshift have been relativistically and cosmologically
c corrected):

      Spm=0
      do 600, i=1,nin
         Rikm(i)=Mpc*Ri(i)*Dlum/((1+vmean/cvelo)**2)
         Spm=Spm+(xdata(i)*xdata(i)*Rikm(i))
600   continue

      Mpm=(10.2*Spm)/(G*(nin-1.5))

      write(*,*) Rmh
      open(unit=20,file=output,status='unknown')
      write(20,46) Mv, Mvrw, Mpm
46    format (' Virial Mass (pairwaise estimator)= ',e15.9,//
     @ ' Virial Mass (ring-wise estimator)= ',e15.9,//
     @ ' Projected Mass= ',e15.9)

      stop
      end
```

```fortran
c-----------------------------------------------------------------------------
      FUNCTION DIST (alpha1,delta1,alpha2,delta2)
c-----------------------------------------------------------------------------

c--- This function calculates the distance between two points on the
c    celestial sphere:
c    - both alphas and deltas are expressed in degrees
c    - the distance in output is expressed in radiants
c
c*****************************************************************************
      real*8 alpha1,delta1,alpha2,delta2

      dist=acos(sin(delta1*pi/180)*sin(delta2*pi/180)+cos
     A(delta1*pi/180)*cos(delta2*pi/180)*cos((alpha1-alpha2)*pi/180))

      return
      end



c-----------------------------------------------------------------------------
      FUNCTION ellf (phi,ak)
c-----------------------------------------------------------------------------

c
c Compute the Legendre elliptic integral of the 1st kind evaluated using
c Carlson's function RF.
c To obtain the complete elliptic integral of the first kind phi=pi/2
c

      REAL*8 ak, phi
      REAL*8 s, rf, ellf, zed

      zed=1.
      s=sin(phi)
      ellf=s*rf(cos(phi)**2,(1.-s*ak)*(1.+s*ak),zed)

      return
      end

c-----------------------------------------------------------------------------
c

c-----------------------------------------------------------------------------
      FUNCTION rf (x,y,z)
c-----------------------------------------------------------------------------
c
c Compute Carlson's elliptic integral of the first kind, RF(x,y,z). x, y and z
c must be nonnegative, and at most one can be zero. TINY must be at least 5
c times the machine underflow limit, BIG at most one fifth the machine
c overflow limit.
c
```

```fortran
      REAL*8 rf, x, y, z, ERRTOL, TINY, BIG, THIRD, C1, C2, C3, C4

      PARAMETER (ERRTOL=0.08,TINY=1.5E-38,BIG=3.E37,THIRD=1./3.,
     @C1=1./24.,C2=0.1,C3=3./44.,C4=1./14.)

      REAL*8 alamb, ave, delx, dely, delz, e2, e3, sqrtx, sqrty, sqrtz
      REAL*8 xt, xy, xz

c       if (min(x,y,z).lt.0.or.min(x+y,x+z,y+z)).lt.TINY.or.
c      @max(x,y,z).gt.BIG) pause 'invalide argument in rf'

      xt=x
      yt=y
      zt=z
1     continue
          sqrtx=sqrt(x)
          sqrty=sqrt(y)
          sqrtz=sqrt(z)
          alamb=sqrtx*(sqrty+sqrtz)+sqrty*sqrtz
          xt=0.25*(xt+alamb)
          yt=0.25*(yt+alamb)
          zt=0.25*(zt+alamb)
          ave=THIRD*(xt+yt+zt)
          delx=(ave-xt)/ave
          dely=(ave-yt)/ave
          delz=(ave-zt)/ave
      if (max(abs(delx),abs(dely),abs(delz)).gt.ERRTOL) goto 1
      e2=delx*dely-delz**2
      e3=delx*dely*delz
      rf=(1.+(C1*e2-C2-C3*e3)*e2+C4*e3)/sqrt(ave)

      return
      end


c-----------------------------------------------------------------------------
        SUBROUTINE XBIWT (xdata,N,XLBIWT,XSBIWT,XLBIWT1,XSBIWT1)
c-----------------------------------------------------------------------------

c--- The subroutine XBIWT provides an estimator of the location and
c    scale of the data set XDATA.  The scale uses the Biweight function
c    in the general formula of "A-estimators." This formula is given
c    on page of 416 in UREDA (formula 4). The BIWEIGHT scale estimate
c    is returned as the value XSBIWT. The BIWEIGHT function is given
c    by:
c
c                                 u((1-u*u)**2)     abs(u) <= 1
c                        f(u) =
c                                 0                 abs(u) >  1
c
c    where u is defined by
c
c                        u = (XDATA(I) - M) / c*MAD  .
c
```

```
c    M, MAD, and c are the median, the median absolute deviation from
c    the median, and the tuning constant respectively. The tuning
c    constant is a parameter which is chosen depending on the sample
c    size and the specific function being used for the scale estimate.
c    (See page 417 in UREDA).  Here we take c = 9.0.
c
c--- The biweght location is found using the formula:
c
c                         T = M + (sums)
c
c                         where M is the sample median and sums are
c                         as given on page 421 in UREDA
c
c                         the tuning constant c is set to 6.0 for calculation
c                         of the location as reccommended by Tukey ()
c
c--- NOTE that the biweight is meant to be an iterated estimator, but one
c    commonly only takes the first step of the iteration.  Here we report
c    both the one-step estimators (XLBIWT1, XSBIWT1) and the preferred
c    fully iterated versions (XLBIWT, XSBIWT).
c
c***************************************************************************


      implicit real*8 (a-h,o-z)
      dimension xdata(n),u1(10000),u2(10000),
     +  xlb(11),xsb(11)
      dimension depths(15),xletter(15,2),mletter(8),sletter(15)
      dimension  rletter(8),cletter(8)
      data zero,d6,d1,d5,d9/0.0,6.0,1.0,5.0,9.0/

c---    sort the data and find the median

      CALL MDIAN1(XDATA,N,XM)

c---    call xmad to find the median absolute deviation

      CALL XMAD(XDATA,N,XM,XMADM)

c---    must choose value of the tuning constant "c"
c       here c = 6.0 for the location estimator and
c       9.0 for the scale estimator

      c1 = d6
      c2 = d9

      if (xmadm.le..0001) then
      xlbiwt=xm
      xlbiwt1=xm
      xsbiwt=xmadm
      xsbiwt1=xmadm
      goto 20
      endif
```

```
      do 11 i = 1,n
      u1(i) = (xdata(i) - xm)/(c1*xmadm)
      u2(i) = (xdata(i) - xm)/(c2*xmadm)
11    continue

      s1 = zero
      s2 = zero
      s3 = zero
      s4 = zero

      do 12 i = 1,n
      if (abs(u2(i)) .lt. d1) then
          s1 = s1+(((xdata(i)-xm)**2)*(d1-(u2(i)*u2(i)))**4)
          s2 = s2+((d1-u2(i)*u2(i))*(d1-(d5*u2(i)*u2(i))))
      endif
      if (abs(u1(i)) .lt. d1) then
          s3 = s3+(xdata(i)-xm)*(d1-u1(i)*u1(i))**2
          s4 = s4+(d1-u1(i)*u1(i))**2
      endif
12    continue

c--- here are the one-step estimators

      xlbiwt1 = xm+s3/s4
      xsbiwt1 = float(n)/(float(n-1))**0.5*s1**0.5/abs(s2)

c--- now obtain the fully-iterated versions

c--- solve for new estimates of u1 and u2

      xlb(1) = xlbiwt1
      xsb(1) = xsbiwt1

      do 15 j = 2,11   !assuming 10 iterations is sufficient

      xmm = xlb(j-1)
      do 13 i = 1,n
      u1(i) = (xdata(i) - xmm)/(c1*xmadm)
      u2(i) = (xdata(i) - xmm)/(c2*xmadm)
13    continue

      s1 = zero
      s2 = zero
      s3 = zero
      s4 = zero

      do 14 i = 1,n
      if (abs(u2(i)) .lt. d1) then
          s1 = s1+(((xdata(i)-xmm)**2)*(d1-(u2(i)*u2(i)))**4)
          s2 = s2+((d1-u2(i)*u2(i))*(d1-(d5*u2(i)*u2(i))))
      endif
      if (abs(u1(i)) .lt. d1) then
          s3 = s3+(xdata(i)-xmm)*(d1-u1(i)*u1(i))**2
          s4 = s4+(d1-u1(i)*u1(i))**2
```

```
      endif
14    continue

      xlb(j) = xlb(j-1)+s3/s4
      xsb(j) = float(n)/(float(n-1))**0.5*s1**0.5/abs(s2)

15    continue

      xlbiwt = xlb(11)
      xsbiwt = xsb(11)

20    continue
      return
      end

c-----------------------------------------------------------------------------
      SUBROUTINE XMAD (XDATA,N,XMED,XMADM)
c-----------------------------------------------------------------------------

c--- The XMAD subroutine calculates the Median Absolute Deviation from
c    the sample median. The median, M , is subtracted from each
c    ORDERED statistic and then the absolute value is taken. This new
c    set of of statistics is then resorted so that they are ORDERED
c    statistics. The MAD is then defined to be the median of this
c    new set of statistics and is returned as XMADM. The MAD can
c    be defined:
c
c                  XMADM = median{ abs(x(i) - M) }
c
c    where the x(i) are the values passed in the array XDATA, and
c    the median, M, is passed in the array XLETTER. The set of stats
c    in the brackets is assumed to be resorted. For more information
c    see page 408 in UREDA.
c
c
c****************************************************************************


      implicit real*8 (a-h,o-z)
      dimension xdata2(10000),xdata(n)
      data dhalf,n1,n2/0.5,1,2/

      do 11 i = 1,n
      xdata2(i) = abs(xdata(i) - xmed)
11    continue

      CALL SORT(N,XDATA2)

      if (float(n)/float(n2) - int(n/n2) .eq. 0) then
              i1 = n/n2
              i2 = n/n2 + n1
              xmadm = dhalf*(xdata2(i1) + xdata2(i2))
      else
              i1 = int(n/n2) + n1
```

```
                xmadm = xdata2(i1)
        endif

        return
        end

c-----------------------------------------------------------------------------
        SUBROUTINE MDIAN1 (X,N,XMED)
c-----------------------------------------------------------------------------

c---  Taken from Numerical Recipes, page 460.
c     Given an array X of N numbers, returns their median value XMED, and
c     the array is sorted low to high
c
c******************************************************************************

        implicit real*8 (a-h,o-z)
        dimension x(n)
        call sort(n,x)
        n2=n/2
        if (2*n2.eq.n) then
                xmed=0.5*(x(n2)+x(n2+1))
        else
                xmed=x(n2+1)
        endif
        return
        end

c-----------------------------------------------------------------------------
        SUBROUTINE SORT(N,RA)
c-----------------------------------------------------------------------------

c--- Routine to do a heapsort of a data array RA
c    Stolen (unabashedly) from NUMERICAL RECIPES
c
c*****************************************************************************

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

      l=n/2+1
      ir=n
10    continue
        if(l.gt.1)then
          l=l-1
          rra=ra(l)
        else
          rra=ra(ir)
          ra(ir)=ra(1)
          ir=ir-1
          if(ir.eq.1)then
            ra(1)=rra
            return
          endif
```

```
      endif
      i=l
      j=l+l
20    if(j.le.ir)then
        if(j.lt.ir)then
          if(ra(j).lt.ra(j+1))j=j+1
        endif
        if(rra.lt.ra(j))then
          ra(i)=ra(j)
          i=j
          j=j+j
        else
          j=ir+1
        endif
      go to 20
      endif
      ra(i)=rra
    go to 10
    end

c
c ****************************************************************************

c
c ****************************************************************************

      SUBROUTINE CosmoDist (H0,OmegaM,OmegaL,w,z,Dlum,Vol)

c
c ----------------------------------------------------------------------------

c Author:      Alberto Cappi, Osservatorio Astronomico di Bologna
c E-Mail:      cappi@bo.astro.it
c Home Page:   http://www.bo.astro.it/~cappi

c Description: this subroutine calculates cosmological distances and volumes
c              from redshifts. Redshifts must be in a vector.

c Version:     1.0 beta, compiled with f77, alpha Unix
c Date:        04/04/2000


c Input:
c H0           - Hubble constant in km/s/Mpc
c OmegaM       - rho/rho_c of the matter component
c OmegaL       - rho/rho_c of the cosmological constant/quintessence component
c w            - P/(rho c^2); w=-1 for Lambda; -1<w<0 for quintessence
c zv           - redshift
c
c Output:
c Dl(ngal)     - luminosity distance (vector)
c Volume(ngal) - cosmological volume (vector)

c Notes:       A simple, web-based tool for calculating cosmological distances
```

```
c               is available at
c http://boas5.bo.astro.it/~cappi/cosmotools.html.

c
c-----------------------------------------------------------------------------

        REAL z, z1, w, w1, wQ, O, absO, Vol
        REAL H0, q0, cvel, h, Tnorm, HRC, cs, DC, a
        REAL pig, pig43, rad
        REAL R0, r, R0r, tau, cdiff, Dcom, Dlum
        REAL OmegaM, OmegaL, Omega_M, Omega_L, Omega_k
        INTEGER k
        PARAMETER (eps=1.E-9,jmax=55)
        EXTERNAL midinf
        COMMON /MathConst/pig,pig43
        COMMON /PhysConst/cvel
        COMMON /Cosmology/Omega_M, Omega_L, Omega_k, wQ

        if (Omega_L.gt.0.and.w.gt.0) then
                write (6,*) ' w is outside permitted range -1=<w<0 '
                stop
        end if

        cvel=299792.458
        pig=3.1415926536
        pig43=pig*4./3.
        rad=pig/180.
        h=H0/100.
        Tnorm=9.77810945/h

        w1=1.+w
        wQ=3.*w1

        Omega_M=OmegaM
        Omega_L=OmegaL

        Omega_k=1.0-Omega_M-Omega_L
        q0=0.5*(1-Omega_k)+1.5*(w*Omega_L)
        CH0=cvel/H0

        absO=abs(Omega_k)
        if (absO.lt.1.0E-06) then
                          Omega_k=0.0
                          k=0
                          HRC=1.
                          R0=cvel/H0
                    else
                          absO=abs(Omega_k)
                          k=-absO/Omega_k
                          cs=sqrt(absO)
                          HRC=1./cs
                          R0=HRC*cH0
        end if
```

```
        DC=1.0
        a=1.0

          z1=1.+z

          IF (Omega_L.eq.0.) THEN                ! Friedmann-Lemaitre models
                CALL Mattig (CH0,q0,z,Dcom,Dlum)
                CALL FLRVol (CH0,q0,z,Vol)
          END IF

          IF (Omega_M.eq.0.and.Omega_L.eq.1.and.w.eq.-1.) THEN ! Flat Lambda
                Dcom=CH0*z
                Dlum=Dcom*z1
                r=Dcom/R0
                CALL Volumes (R0,r,k,Vol)
          END IF

          IF (Omega_L.gt.0.) THEN                               ! General case
                call qromo (a,z1,omega,eps,jmax,1)
                if (k.ne.0) omega=omega*cs
                if (k.eq.-1) r=sinh(omega)
                if (k.eq.0) r=omega
                if (k.eq.1) r=sin(omega)
                R0r=R0*r
                Dlum=R0r*z1
                CALL Volumes (R0,r,k,Vol)
          END IF

      Return
      END



        SUBROUTINE Mattig (CH0,q0,z,DC,DL)
        REAL CH0, q0, z, DC, DL

c CH0 = c / H0
c DC: Comoving distance = R0*r
c DL: Luminosity distance = DC*(1+z)

        z1=1.0+z
        if (q0.eq.0.) then
           DL=CH0*z*(1.+z/2.)
           DC=DL/(1.+z)
                      else
           AAA=1.-q0+q0*z+(q0-1.)*(SQRT(2.*q0*z+1.))
           DL=AAA*CH0/(q0*q0)
           DC=DL/(1.+z)
        end if

        Return
        END
```

```
SUBROUTINE FLRVol (CH0,q0,z,Vol)

REAL z,z1,Vol
REAL CH0,q0,CH3
REAL A,B,O,PSI0,PSI1

COMMON /MathConst/pig,pig43
COMMON /PhysConst/cvel

CH3=CH0**3

z1=1.+z
IF (q0.EQ.0.0) THEN
        VOL=2.*pig*CH3*((Z1**4-1.)/(Z1*Z1)/4.-LOG(Z1))
END IF
IF (q0.EQ.0.5) THEN
        VOL=32./3.*pig*CH3*(1-1/SQRT(1.+Z))**3
END IF
IF (q0.EQ.1) THEN
        VOL=2.*pig*CH3*(ASIN(Z/Z1)-Z*SQRT(1.+2.*Z)/Z1**2)
END IF

IF (q0.GT.0.AND.q0.LT.0.5) THEN
        R0=CH0/SQRT(1.-2.*q0)
        A=(1.-q0)/q0
        PSI0=LOG(A+SQRT(A*A-1))
        B=(A+Z)/(1.+Z)
        PSI1=LOG(B+SQRT(B*B-1))
        O=PSI0-PSI1
        VOL=2.*pig*R0**3*(0.5*SINH(2.*O)-O)
END IF

IF (q0.GT.0.5.AND.q0.LT.1.) THEN
        R0=CH0/SQRT(2.*q0-1.)
        A=(1.-q0)/q0
        PSI0=ACOS(A)
        B=(A+Z)/(1.+Z)
        PSI1=ACOS(B)
        O=PSI0-PSI1
        VOL=2.*pig*R0**3*(O-0.5*SIN(2.*O))
END IF

Return
END



SUBROUTINE Volumes (R0,r,k,Vol)

REAL R0,r,Vol
REAL rsq,ash,V0,V1
INTEGER k
```

```fortran
        COMMON /MathConst/pig,pig43

c Parametric form

        V0=pig43*(R0*r)**3
        if (k.eq.0) then
          Vol=V0
        end if

        if (k.eq.1) then
          rsq=r*r
          V1=1.5*(asin(r)/r**3-sqrt(1.-rsq)/rsq)
          Vol=V0*V1
        end if

        if (k.eq.-1) then
          rsq=r*r
          ash=log(r+sqrt(rsq+1.))                 !  This is arcsinh(r)
          V1=1.5*(sqrt(1.+rsq)/rsq-ash/r**3)
          Vol=V0*V1
        end if

        Return
        END



        SUBROUTINE fun(t,f,g)
        REAL t, f, g, v2, v3
        REAL Omega_m, Omega_L, Omega_Q, Omega_k, wQ
        COMMON /Cosmology/Omega_m, Omega_L, Omega_k, wQ

         v2=t*t
         v3=v2*t
         if (wQ.ne.0.) then
          vQ=t**wQ
                     else
          vQ=1.0
         end if
         f=sqrt(Omega_M*v3+Omega_k*v2+Omega_L*vQ)
         f=1./f
         g=f/t

        Return
        END



c
*****************************************************************************
c
c  Routines for numerical integration, from Numerical Recipes.
c  (C) Copr. 1986-92 Numerical Recipes Software ]2#1.
```

```
      SUBROUTINE qromo(a,b,ss,eps,jmax,isel)

      INTEGER JMAX,K,KM
      REAL a,b,ss,EPS
      EXTERNAL midinf
      PARAMETER (K=5, KM=K-1)
CU    USES polint
      INTEGER j
      REAL dss,h(JMAX+1),s(JMAX+1)
      h(1)=1.
      do 11 j=1,JMAX
        call midinf (a,b,s(j),j,isel)
        if (j.ge.K) then
          call polint(h(j-KM),s(j-KM),K,0.,ss,dss)
          if (abs(dss).le.EPS*abs(ss)) return
        endif
        s(j+1)=s(j)
        h(j+1)=h(j)/9.
11    continue
      pause 'too many steps in qromo'
      END
C  (C) Copr. 1986-92 Numerical Recipes Software ]2#1.


      SUBROUTINE midinf(aa,bb,s,n,isel)
c
      INTEGER n
      REAL aa,bb,s
      INTEGER it,j
      REAL a,b,ddel,del,sum,tnm,x
      b=1./aa
      a=1./bb
      if (n.eq.1) then
        d=0.5*(a+b)
        dinv=1./d
        call fun(dinv,aaa,bbb)
        if (isel.eq.1) fun3=aaa
        if (isel.eq.2) fun3=bbb
        s=(b-a)*fun3/d**2
      else
        it=3**(n-2)
        tnm=it
        del=(b-a)/(3.*tnm)
        ddel=del+del
        x=a+0.5*del
       sum=0.
        do 11 j=1,it
          xinv=1./x
          call fun(xinv,aaa,bbb)
          if (isel.eq.1) fun3=aaa
          if (isel.eq.2) fun3=bbb
          sum=sum+fun3/x**2
          x=x+ddel
```

```
          xinv=1./x
          call fun(xinv,aaa,bbb)
          if (isel.eq.1) fun3=aaa
          if (isel.eq.2) fun3=bbb
          sum=sum+fun3/x**2
          x=x+del
11      continue
        s=(s+(b-a)*sum/tnm)/3.
      endif
      return
      END
C  (C) Copr. 1986-92 Numerical Recipes Software ]2#1.
c
c
      SUBROUTINE polint(xa,ya,n,x,y,dy)
      INTEGER n,NMAX
      REAL dy,x,y,xa(n),ya(n)
      PARAMETER (NMAX=10)
      INTEGER i,m,ns
      REAL den,dif,dift,ho,hp,w,c(NMAX),d(NMAX)
      ns=1
      dif=abs(x-xa(1))
      do 11 i=1,n
        dift=abs(x-xa(i))
        if (dift.lt.dif) then
          ns=i
          dif=dift
        endif
        c(i)=ya(i)
        d(i)=ya(i)
11    continue
      y=ya(ns)
      ns=ns-1
      do 13 m=1,n-1
        do 12 i=1,n-m
          ho=xa(i)-x
          hp=xa(i+m)-x
          w=c(i+1)-d(i)
          den=ho-hp
          if(den.eq.0.)pause 'failure in polint'
          den=w/den
          d(i)=hp*den
         c(i)=ho*den
12      continue
        if (2*ns.lt.n-m)then
          dy=c(ns+1)
        else
          dy=d(ns)
          ns=ns-1
        endif
        y=y+dy
13    continue
      return
      END
```