```
cc To use this to integrate func(x)=sin(x) for 0.1 to 1.0, for example:
        implicit double precision (a-h,o-z)
        common h,amp
        external func
        a=0.0d0
        tol=1.0d-12
        write(*,*) 'Enter kmax,h,amp'
        read(*,*) b,h,amp
        result=rombint(func,a,b,tol)
c       write(*,*) 'Q=',sqrt(result)*2.73d6
        write(*,*) 'Vrms=',sqrt(result)*100.*h
        stop
        end
c
        double precision function func(x)
        implicit double precision (a-h,o-z)
        dimension ajl(3)
        common h,amp
        if (x.eq.0) then
          func=0.d0
          return
        end if
        xc=0.024*h
        c=(x/xc)**2.4
        pdelta=amp*x/sqrt(1+c)
c       y=x*8.d0/h
c       w=3.0*(sin(y)-y*cos(y))/(y*y*y)
c       func=4.0d0*3.141592654d0*x*x*pdelta*w*w
cc      rh=2.998d3/h
cc      y=2.d0*x*rh
c       w=((3.d0-y*y)*sin(y)-3.d0*y*cos(y))/(y*y*y)
cc      call spherbess(y,2,ajl)
cc      w=ajl(3)
cc      func=5.0d0*3.141592654d0*pdelta/(x*x*rh**4)*w*w
        y=x*50.d0/h
        w=3.0*(sin(y)-y*cos(y))/(y*y*y)
        func=4.0d0*3.141592654d0*pdelta*w*w
c
        return
        end
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
ccccccccc
        function rombint(f,a,b,tol)
c  Rombint returns the integral from a to b of f(x)dx using Romberg
integration.
c  The method converges provided that f(x) is continuous in (a,b).  The
function
c  f must be double precision and must be declared external in the
calling
c  routine.  tol indicates the desired relative accuracy in the
integral.
c
        parameter (MAXITER=17,MAXJ=5)
        implicit double precision (a-h,o-z)
        dimension g(MAXJ+1)
        external f
c
```

```
      h=0.5d0*(b-a)
      gmax=h*(f(a)+f(b))
      g(1)=gmax
      nint=1
      error=1.0d20
      i=0
10      i=i+1
        if (i.gt.MAXITER.or.(i.gt.15.and.abs(error).lt.tol))
     2    go to 40
c  Calculate next trapezoidal rule approximation to integral.
        g0=0.0d0
          do 20 k=1,nint
          g0=g0+f(a+(k+k-1)*h)
20      continue
        g0=0.5d0*g(1)+h*g0
        h=0.5d0*h
        nint=nint+nint
        jmax=min(i,MAXJ)
        fourj=1.0d0
          do 30 j=1,jmax
c  Use Richardson extrapolation.
          fourj=4.0d0*fourj
          g1=g0+(g0-g(j))/(fourj-1.0d0)
          g(j)=g0
          g0=g1
30      continue
        if (abs(g0).gt.tol) then
          error=1.0d0-gmax/g0
        else
          error=gmax
        end if
        gmax=g0
        g(jmax+1)=g0
      go to 10
40    rombint=g0
      if (i.gt.MAXITER.and.abs(error).gt.tol)
     2  write(*,*) 'Rombint failed to converge; integral, error=',
     3   rombint,error
      return
      end
```