```
c=============================================================
c     newtsqrt:  Uses Newton's method to find (positive)
c     square root of number supplied on command line, i.e.
c     solves
c
c     f(x) = x^2 - a = 0
c
c     for given 'a'.  Optional second argument specifies
c     convergence criteria (relative dx). Tracing output
c     includes iteration number, estimated root (xn),
c     change in estimate (dxn), log(dxn), residual and
c     log(residual).  Optional second argument
c     specifies convergence criteria (relative dx).
c=============================================================
      program         newtsqrt

      implicit        none

      integer         iargc
      real*8          r8arg,        drelabs

      real*8          r8_never
      parameter     ( r8_never = -1.0d-60 )

      integer         mxiter
      parameter     ( mxiter = 50 )

      real*8          a,            xtol

      integer         iter
      real*8          xn,           resn,           dxn
c-------------------------------------------------------------
c     Argument parsing.
c-------------------------------------------------------------
      if( iargc() .lt. 1 ) go to 900
      a     = r8arg(1,r8_never)
      if( a .eq. r8_never .or. a .lt. 0.0d0 ) go to 900
      xtol  = r8arg(2,1.0d-8)
      if( xtol .le. 0.0d0 ) xtol = 1.0d-8
c-------------------------------------------------------------
```

```
c     Un-inspired initial guess: x^(0) = a / 2.
c-------------------------------------------------------------
      xn = 0.5d0 * a
c-------------------------------------------------------------
c     Newton loop.
c-------------------------------------------------------------
      write(0,*) 'Iter          xn                 '//
     &           'dxn       log(dxn)    rn         log(rn)'
      write(0,*)
      do iter = 1 , mxiter
         resn = xn**2 - a
         dxn  = resn / (2.0d0 * xn)
         xn   = xn - dxn
         write(0,1000) iter, xn, dxn, log(abs(dxn)),
     &                 resn, log(abs(resn))
1000     format(i2,1p,e26.16,e12.3,0p,f10.2,1p,e12.3,0p,f10.2)
c-------------------------------------------------------------
c        Check for convergence.
c-------------------------------------------------------------
         if( drelabs(dxn,dxn,1.0d-6) .le. xtol ) go to 100
      end do
c-------------------------------------------------------------
c     No-convergence exit.
c-------------------------------------------------------------
      write(0,*) 'No convergence after ', mxiter,
     &           ' iterations'
      stop
c-------------------------------------------------------------
c     Normal exit, write input and estimated square root
c     to standard output.
c-------------------------------------------------------------
 100  continue
      write(0,*)
      write(*,*) a, xn
      stop

 900  continue
         write(0,*) 'usage: newtsqrt <a> [<xtol>]'
      stop
      end
```

```
c-------------------------------------------------------------
c     drelabs:  Function useful for 'relativizing' quantity
c     being monitored for detection of convergence.
c-------------------------------------------------------------
      real*8 function drelabs(dx,x,xfloor)

         implicit      none

         real*8        dx,     x,    xfloor

         if( abs(x) .lt. abs(xfloor) ) then
            drelabs = abs(dx)
         else
            drelabs = abs(dx/x)
         end if

         return

      end
```

```
#################################################################
# Building 'newtsqrt' and sample output on SGIs
#################################################################

einstein% pwd; ls
/usr2/people/phy329/nonlin/ex2
Makefile     newtsqrt.f

einstein% make
        f77 -g -c newtsqrt.f
        f77 -g -L/usr/local/lib newtsqrt.o -lp329f -o newtsqrt

einstein% newtsqrt
 usage: newtsqrt <a> [<xtol>]

einstein% newtsqrt 10.0
 Iter        xn                  dxn     log(dxn)    rn       log(rn)

 1  3.5000000000000000E+00   1.500E+00    0.41   1.500E+01    2.71
 2  3.1785714285714284E+00   3.214E-01   -1.13   2.250E+00    0.81
 3  3.1623194221508828E+00   1.625E-02   -4.12   1.033E-01   -2.27
 4  3.1622776604441363E+00   4.176E-05  -10.08   2.641E-04   -8.24
 5  3.1622776601683795E+00   2.758E-10  -22.01   1.744E-09  -20.17

    10.00000000000000       3.162277660168380

einstein% newtsqrt 10.0 1.0e-15
 Iter        xn                  dxn     log(dxn)    rn       log(rn)

 1  3.5000000000000000E+00   1.500E+00    0.41   1.500E+01    2.71
 2  3.1785714285714284E+00   3.214E-01   -1.13   2.250E+00    0.81
 3  3.1623194221508828E+00   1.625E-02   -4.12   1.033E-01   -2.27
 4  3.1622776604441363E+00   4.176E-05  -10.08   2.641E-04   -8.24
 5  3.1622776601683795E+00   2.758E-10  -22.01   1.744E-09  -20.17
 6  3.1622776601683791E+00   2.809E-16  -35.81   1.776E-15  -33.96

    10.00000000000000       3.162277660168379
```

```
einstein% newtsqrt 0.5
 Iter        xn                   dxn     log(dxn)    rn        log(rn)

 1  1.1250000000000000E+00  -8.750E-01   -0.13  -4.375E-01   -0.83
 2  7.8472222222222221E-01   3.403E-01   -1.08   7.656E-01   -0.27
 3  7.1094518190757128E-01   7.378E-02   -2.61   1.158E-01   -2.16
 4  7.0711714297003669E-01   3.828E-03   -5.57   5.443E-03   -5.21
 5  7.0710678126246607E-01   1.036E-05  -11.48   1.465E-05  -11.13
 6  7.0710678118654757E-01   7.592E-11  -23.30   1.074E-10  -22.95

   0.5000000000000000      0.707106781186547
```