Source file: bisect.f

```
c===========================================================
c     bisect: Uses bisection to find approximate root
c     of f(x) on interval [xmin .. xmax].  Return value is
c     root located to (relative) tolerance 'xtol'.  Return code
c     'rc' is set to 0 on success, non-zero on failure
c     and routine succeeds (by definition) as long as initial
c     interval *does* bracket at least one root.  Routine
c     performs tracing of algorithm (on stderr) if input
c     argument 'trace' is .true.
c===========================================================
      real*8 function bisect(f,xmin,xmax,xtol,trace,rc)

         implicit     none

         real*8       drelabs

         real*8       f
         external     f

         real*8       xmin,          xmax,         xtol
         logical      trace
         integer      rc
c-----------------------------------------------------------
c        Other variables needed for search.
c-----------------------------------------------------------
         integer      mxiter
         parameter  ( mxiter = 50 )

         real*8       xlo,           dx,           sgn
         integer      iter
c-----------------------------------------------------------
c        Check that input interval is specified correctly
c        and that it manifestly brackets at least one root:
c        (i.e. the fcn changes sign).
c-----------------------------------------------------------
         if( xmax .le. xmin  .or.
     &       f(xmin) * f(xmax) .gt. 0.0d0 ) then
            write(0,*) 'bisect:  Input interval is not '//
     &                 'bracketing'
            rc = 1
c-----------------------------------------------------------
c           Returned value is meaningless in this case,
c           but have to return *some* value.
c-----------------------------------------------------------
            bisect = xmin
            return
         end if
c-----------------------------------------------------------
c        Compute 'sgn' such that sgn * f(xmin) < 0, and
c        intialize bracketing interval
c-----------------------------------------------------------
         sgn  = 1.0d0
         if( f(xmin) .le. 0.0d0 ) then
            sgn =  1.0d0
         else
            sgn = -1.0d0
         end if
         xlo  = xmin
         dx   = xmax - xmin
c-----------------------------------------------------------
c        Bisection loop: continue until root found to
c        specfied tolerance or until maximum number of
c        iterations taken
c-----------------------------------------------------------
         do iter = 1 , mxiter
            bisect = xlo + 0.5d0 * dx
            if( trace ) then
               write(0,*) xlo, xlo + dx, f(bisect)
            end if
            if( sgn * f(bisect) .lt. 0.0d0 ) then
               xlo = bisect
            end if
            if( drelabs(dx,bisect,1.0d-10) .le. xtol ) go to 900
            dx = 0.5d0 * dx
         end do

900      continue
         rc = 0
         if( trace ) write(0,*)
         return

      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
```

Source file: tbisect.f

```
c===========================================================
c     tbisect: Illustrates root finding using bisection
c     routine 'bisect'.
c
c     Initial bracketing interval must be specified via the
c     command-line, along with optional convergence criteria
c     and output option.
c
c     This program also illustrates the general Fortran
c     techniques (briefly discussed previously) for:
c
c     (1) Writing and using routines which take other routines
c     as arguments.
c     (2) Using a COMMON block to communicate information to
c     a routine in cases where the information cannot be
c     passed via the argument list.
c     (3) Using an "INCLUDE" file (in this case 'comf.inc')
c     to ensure that the same common block structure is defined
c     in all program units.
c
c     Currently set up for computing square roots i.e.
c     solves
c
c         f(x; a) = x**2 - a = 0
c
c     for 'a' specifed on command-line
c
c     Outputs a, approximate root (x*) and f(x*; a) on stdout.
c===========================================================
      program         tbisect
      implicit        none
c-----------------------------------------------------------
c     Declaration of the bisection routine.
c-----------------------------------------------------------
      real*8          bisect
c-----------------------------------------------------------
c     Name of the specific function whose root we seek.
c     Note use of 'external' to let compiler know 'fsqr'
c     is the name of a function, not a variable.
c-----------------------------------------------------------
      real*8          fsqr
      external        fsqr

      integer         i4arg,         iargc
      real*8          r8arg
c-----------------------------------------------------------
c     For use in detecting bad real*8 command-line value.
c-----------------------------------------------------------
      real*8          r8_never
      parameter     ( r8_never = -1.0d-60 )
c-----------------------------------------------------------
c     Use a common block to pass number whose square root
c     is sought to external function 'fsqr'.
c-----------------------------------------------------------
      include        'comf.inc'
c-----------------------------------------------------------
```

```
c     Initial bracket, convergence tolerance and output
c     option from command-line; default value for conv.
c     tolerance.
c-----------------------------------------------------------
      real*8          xmin,           xmax,         xtol
      logical         trace

      real*8          default_xtol
      parameter     ( default_xtol = 1.0d-8 )
c-----------------------------------------------------------
c     Root and return code from bisection routine.
c-----------------------------------------------------------
      real*8          root
      integer         rc
c-----------------------------------------------------------
c     Argument parsing.
c-----------------------------------------------------------
      if( iargc() .lt. 3 ) go to 900
      a     = r8arg(1,r8_never)
      xmin  = r8arg(2,r8_never)
      xmax  = r8arg(3,r8_never)
      if( a .eq. r8_never  .or.  xmin .eq. r8_never  .or.
     &    xmax .eq. r8_never ) go to 900

      xtol  = r8arg(4,default_xtol)
      trace = iargc() .gt. 4
c-----------------------------------------------------------
c     Invoke root finder then write a, sqrt(a), and residual
c     to standard output.
c-----------------------------------------------------------
      root = bisect(fsqr,xmin,xmax,xtol,trace,rc)
      if( rc .eq. 0 ) then
         write(*,*) a, root, fsqr(root)
      else
         write(0,*) 'tbisect: Bisection failed.'
      end if
c-----------------------------------------------------------
c     Normal exit.
c-----------------------------------------------------------
      stop

c-----------------------------------------------------------
c     Usage exit.
c-----------------------------------------------------------
 900  continue
         write(0,*) 'usage: tbisect <a> <xmin> <xmax> '//
     &              '[<xtol> <trace>]'
      stop

      end

c===========================================================
c     Function whose root is sought.  Again, note use of
c     COMMON block to pass additional information (in this
c     case 'a') to the routine.
c===========================================================
      real*8 function fsqr(x)
         implicit      none

         real*8        x

         include      'comf.inc'

         fsqr = x**2 - a

         return
      end
```

Source file: comf.inc

```
c-----------------------------------------------------------
c     Common block for communicating value of 'a' from main
c     to 'fsqr'.
c-----------------------------------------------------------
      real*8              a
      common    / comf /   a
```

Source file: sgi-output

```
############################################################
# Building 'tbisect' and sample output on sgi1
#
# 'tbisect' is set up to compute sqrt(a) via bisection.
############################################################

sgi1% pwd ; ls
/usr/people/phys410/nonlin/ex1

Makefile    bisect.f    comf.inc    tbisect.f
sgi1% make
f77 -g -64 -c tbisect.f
f77 -g -64 -c bisect.f
f77 -g -64 -L/usr/local/lib tbisect.o bisect.o -lp410f -o tbisect

sgi1% tbisect
 usage: tbisect <a> <xmin> <xmax> [<xtol> <trace>]

############################################################
# Compute +sqrt(2) to default tolerance (1.0d-8)
#
# Note: Exact value to 16 digits is 1.414 2135 6237 3095
############################################################
sgi1% tbisect 2.0 1.0 2.0
    2.000000000000000       1.414213564246893      5.2999009625409599E-09

############################################################
# Recompute with higher tolerance (1.0d-12)
############################################################
sgi1% tbisect 2.0 1.0 2.0 1.0e-12
    2.000000000000000       1.414213562372879     -6.1084470814876113E-13

############################################################
# Enable tracing output by supplying 5th argument.  Note
# supplying a '.' as an argument parsed by 'i4arg' or 'r8arg'
# is equivalent to specifying the default value.
############################################################
sgi1% tbisect 2.0 1.0 2.0 . 1
    1.000000000000000       2.000000000000000      0.2500000000000000
    1.000000000000000       1.500000000000000     -0.4375000000000000
    1.250000000000000       1.500000000000000     -0.1093750000000000
    1.375000000000000       1.500000000000000      6.6406250000000000E-02
    1.375000000000000       1.437500000000000     -2.2460937500000000E-02
    1.406250000000000       1.437500000000000      2.1728515625000000E-02
    1.406250000000000       1.421875000000000     -4.2724609375000000E-04
    1.414062500000000       1.421875000000000      1.0635375976562500E-02
    1.414062500000000       1.417968750000000      5.1002502441406250E-03
    1.414062500000000       1.416015625000000      2.3355484008789063E-03
    1.414062500000000       1.415039062500000      9.5391273498535156E-04
    1.414062500000000       1.414550781250000      2.6327371597290039E-04
    1.414062500000000       1.414306640625000     -8.2001090049743652E-05
    1.414184570312500       1.414306640625000      9.0632587671279907E-05
    1.414184570312500       1.414245605468750      4.3148174881935120E-06
    1.414184570312500       1.414215087890625     -3.8843369111418724E-05
    1.414199829101563       1.414215087890625     -1.7264334019273520E-05
    1.414207458496094       1.414215087890625     -6.4747728174552321E-06
    1.414211273193359       1.414215087890625     -1.0799813026096672E-06
    1.414213180541992       1.414215087890625      1.6174171832972206E-06
    1.414213180541992       1.414214134216309      2.6871771297010127E-07
    1.414213180541992       1.414213657379150     -4.0563185166320181E-07
    1.414213418960571       1.414213657379150     -6.8457083557404985E-08
    1.414213538169861       1.414213657379150      1.0013031115363447E-07
    1.414213538169861       1.414213597774506      1.5836612909936321E-08
    1.414213538169861       1.414213567972183     -2.6310235545778937E-08
    1.414213553071022       1.414213567972183     -5.2368114289436107E-09
    1.414213560521603       1.414213567972183      5.2999009625409599E-09

    2.000000000000000       1.414213564246893      5.2999009625409599E-09
```