8.1 Find a Zero of a Univariate Function

A. Purpose

Find a zero of a univariate function, $f(x)$.

B. Usage

This subroutine uses reverse communication, *i.e.*, it returns to the calling program each time it needs to have $f()$ evaluated at a new value of x.

B.1 Program Prototype, Single Precision

```
REAL   X1, F1, X2, F2, TOL
INTEGER   MODE
```

Assign values to X1, X2, and TOL.

```
   MODE = 0
   F2 = The value of f() evaluated at X2.
10 F1 = The value of f() evaluated at X1.

CALL SZERO(X1, F1, X2, F2, MODE, TOL)

   IF ( MODE .EQ. 1) go to 10
```

Computed quantities are returned in MODE, X1, F1, X2, and F2.

B.2 Argument Definitions

X1, F1, X2, F2 [inout] On the initial entry, F1 = f(X1) and F2 = f(X2). If F1×F2 > 0, and X2 ≠ X1 then |X2 − X1| will be used in setting the step size for the initial search. If X2 = X1 on entry, then one must also have F2 = F1, and the search is started as described in Section D.

On returns with MODE = 1, X1 is a new trial abscissa. The calling program must set F1 = f(X1) and call SZERO again.

On return with MODE = 2 or 3, X1 is the abscissa at which $|f()|$ had the smallest value, and F1 = f(X1). If F1 ≠ 0, X2 is the nearest abscissa to X1 at which $f()$ was evaluated and found to have the opposite sign from F1, and F2 = f(X2).

MODE [inout] Current state of computation. The user must initially set MODE = 0, and after that should not change its value, except to control detailed printing as explained in Section D. On each return, MODE will have one of the following values:

- = 1 The calling program is to compute F1 = f(X1) and call SZERO again.
- = 2 Normal termination. Error tolerance criterion is satisfied.
- = 3 Normal termination. Error tolerance criterion is not satisfied.
- = 4 Apparently f has a discontinuity between X1 and X2. No zero has been identified.
- = 5 Set when F1×F2 > 0, and SZERO was unable to find function values with different signs.
- = 6 SZERO was called with MODE > 1, or called initially with MODE > 0. This causes execution to stop.

TOL [in] Error tolerance.

- > 0 the zero is to be isolated to an interval of length less than TOL.
- < 0 an x is desired for which $|f(x)| \le |\text{TOL}|$.
- = 0 the iteration continues until the zero of $f()$ is isolated as accurately as possible. In this case SZERO will never set MODE = 3.

B.3 Modifications for Double Precision

For double-precision usage change the name SZERO to DZERO, and change the REAL declaration to DOUBLE PRECISION.

C. Examples and Remarks

The program DRSZERO illustrates the use of SZERO to compute the root of the function, $f(x) = 2^x - 8$, for which the exact answer is $x = 3$. Output is shown in ODSZERO.

D. Functional Description

When F1×F2 > 0 at the initial point, iterates are generated according to the formula $x = x_{\min} + (x_{\min} - x_{\max}) \times \rho$, where the subscript "min" is associated with the (x, f) pair that has the smallest value for $|f|$, and ρ is 8 if $r = f_{\min} / (f_{\max} - f_{\min}) \ge 8$, else $\rho = \max(\kappa/4, r)$, where κ is a count of the number of iterations that have been taken without finding f's with opposite signs. If X1 and X2 have the same value initially (and F1 and F2 equal F(X1)), then the next x is a distance $0.008 + |x_{\min}|/4$ from $x_{\min}$ taken toward 0. (If X1 = X2 = 0, the next x is −.008.)

Let x_1 and x_2 denote the first two x values that give f values with different signs. Let $A < B$ be

©1997 Calif. Inst. of Technology, 2009 Math à la Carte, Inc.

the two values of x that bracket the zero as tightly as is known. Thus $A = x_1$ or $A = x_2$ and B is the other when computing x_3. The next point x_3 is generated treating x as the linear function $q(f)$ that interpolates the points $(f(x_1), x_1)$ and $(f(x_2), x_2)$, and computing $x_3 = q(0)$, subject to the condition that $A + \varepsilon \le x_3 \le B - \varepsilon$, where $\varepsilon = 0.875$ times the accuracy to which the root has been requested. (This condition on x_3 with updated values for A and B is also applied to future iterates.)

Let x_4, x_5, ..., x_m denote the abscissae on the following iterations. Let $a = x_m$, $b = x_{m-1}$, and $c = x_{m-2}$. Either A or B (defined as above) will coincide with a, and B will frequently coincide with either b or c. Let $p(x)$ be the quadratic polynomial in x that passes through the values of f evaluated at a, b, and c. Let $q(f)$ be the quadratic polynomial in f that passes through the points $(f(a), a)$, $(f(b), b)$, and $(f(c), c)$.

Let $C = A$ or B, selected so that $C \ne a$. If the sign of f has changed in the last 4 iterations and $p'(a) \times q'(f(a))$ and $p'(C) \times q'(f(C))$ are both in the interval [1/4, 4], then x is set to $q(0)$. (Note that if p is replaced by f and q is replaced by x, then both products have the value 1.) Otherwise x is set to $a - (a - C)(\varphi/(1 + \varphi))$, where φ is selected based on past behavior and on the ratio of $f(a)$ with values of $f()$ having the same sign as $f(a)$, and different sign from $f(a)$, evaluated at nearby values of x. The method of selecting φ is sufficiently complicated that we simply refer interested readers to the code. The algorithm is such that $0 < \varphi$, and if the sign of $f()$ does not change for an extended period, φ gets large.

Reference [1] compares a number of algorithms for finding a zero of a continuous function. Dr. Shi has kindly used the same test program, ENCL0FX, on DZERO. With Dr. Shi's permission, results from Table II of that paper are given on the next page with an additional column added for the results he obtained with DZERO. With the exception of DE and M, all codes solved all of the problems. See [1] for more details.

Detailed printing

Before the initial call with MODE = 0, or at any time during the iterative process, the user may set a counter for detailed output by calling SZERO with a negative value of MODE. There is no problem-solving action on such a call. A saved counter is set so detailed output will be written using the message processor described in Chapter 19.3 on the next |MODE|−1 normal calls. To resume normal computation with the detailed print on, the calling program must set MODE = 0 if a new problem sequence is being started or MODE = 1 if an iterative sequence is being continued. (Detailed print can be turned off by setting MODE = −1 as is implied by the above.)

The detailed output consists of X1, F1, KTYP, DIV, and KS. KTYP is 0 if φ above was used on the previous iteration, and KTYP is 1 if x was set to $q(0)$. DIV is the name of the program variable corresponding to φ, and KS is the number of iterations since there has been a change in the sign of $f()$.

References

1. G. E. Alefeld, F. A. Potra, and Yixun Shi, *Algorithm 748: Enclosing zeros of continuous functions*, **ACM Trans. on Math. Software 21**, 3 (Sept. 1995) 327–344.

E. Error Procedures and Restrictions

The user must initially set MODE = 0 and must not alter MODE after that while iterating on the same problem (except as described in Section D for detailed output.) Entering SZERO with MODE > 0 when SZERO has not set MODE to 1 results in an error condition and an error message will be issued using the system error handler SMESS (or DMESS). In such a case, if MODE ≠ 6, SZERO will set MODE = 6 and return, whereas if MODE = 6, SZERO will STOP.

SZERO uses the Fortran 77 SAVE statement to save values of internal variables between the successive calls needed to solve a problem. Thus SZERO cannot be used to work on more than one problem at a time. In particular, calling SZERO with MODE = 0 will always initialize it for a new problem and no data will be retained from a previous problem (except for the internal detailed print counter described in Section D.)

F. Supporting Information

The source language is ANSI Fortran 77.

Entry	Required Files
DZERO	AMACH, DMESS, DZERO, MESS
SZERO	AMACH, MESS, SMESS, SZERO

Algorithm and code due to F. T. Krogh, JPL, April 1972. Revised to improve portability, April 1974. Name changed from SFABZ/DFABZ to SZERO/DZERO, and minor improvements to the algorithm, September 1987.

Revised to allow F1×F2 > 0 on initial entry, and to change error handling, November 1991.

Total Number of Function Evaluations in Solving All the Problems Listed in Table I of [1]

tol	BR	DE	M	R	LE	2.1	2.2	2.3	2.4	2.5	4.1	4.2	DZERO
10^{-7}	2804	2808	2839	7630	2694	3154	2950	2645	2791	2687	2696	2650	2100
10^{-10}	2905	2963	2992	7768	2821	3338	3060	2789	2922	2819	2835	2786	2177
10^{-15}	2975	3196	3261	8014	3061	3448	3151	2948	3015	2914	2908	2859	2236
0	3008	2998	3146	8230	3165	3509	3219	3029	3060	2954	2950	2884	2255

DRSZERO

```
c     program DRSZERO
c>> 1995-05-28 DRSZERO  Krogh  Changes to use M77CON
c>> 1993-05-05 DRSZERO  Krogh  Adjusted to simplify conversion to C.
c>> 1992-03-24 DRSZERO  Krogh  Added "," to format statmement.
c>> 1991-11-25 DRSZERO  Krogh  Cleaned up Fortran version.
c>> 1987-12-09 DRSZERO  Krogh  Initial Code.
c--S replaces "?": DR?ZERO, ?ZERO
c     Demo driver for SZERO.  Univariate zero finder.
c     F. T. Krogh, Sept. 1987.
c     ------------------------------------------------------------------
      real               X1, X2, F1, F2, A, B, TOL, TWO, EIGHT
      parameter(A = 0.0E0, B = 4.0E0, TOL = 0.0E0)
      parameter(TWO = 2.0E0, EIGHT = 8.0E0)
      integer MODE
   10 format(' Results from subroutine SZERO:'/'    MODE = ', I3/
     *'     Solution:        X1 = ',F11.8/
     *'                    f(X1) = ',1P,G11.3/
     *'     Accuracy:    X1 - X2 = ',G11.3/
     *'                    f(X2) = ',G11.3)
c     ------------------------------------------------------------------
      write(*,'(1x,a)')
     * 'DRSZERO.. Demo driver for SZERO, univariate zero finder.',
     * 'Problem: Find zero of 2**X - 8.   Exact result: X = 3.'
      X2 = B
      F2 = TWO**X2 - EIGHT
      MODE = 0
      X1 = A
   20 F1 = TWO**X1 - EIGHT
      call SZERO(X1, F1, X2, F2, MODE, TOL)
      if ( MODE .eq. 1) go to 20
      write(*,10) MODE, X1, F1, X1-X2, F2
      stop
      end
```

ODSZERO

```
DRSZERO.. Demo driver for SZERO, univariate zero finder.
Problem: Find zero of 2**X - 8.   Exact result: X = 3.
Results from subroutine SZERO:
   MODE =    2
    Solution:        X1 =  3.00000024
                   f(X1) =   9.537E-07
    Accuracy:    X1 - X2 =   4.768E-07
                   f(X2) =  -1.431E-06
```