7.2 Roots of a Quadratic Polynomial

A. Purpose

Compute the two roots of a quadratic polynomial having real coefficients.

B. Usage

B.1 Program Prototype, Single Precision

REAL A(≥ 3)

COMPLEX Z(≥ 2)

Assign values to A().

CALL SPOLZ2(A, Z)

The roots are returned in Z().

B.2 Argument Definitions

A() [in] Contains the coefficients of the polynomial

$$a_1x^2 + a_2x + a_3$$

Require A(1) $\neq 0$. The contents of the array A() are not modified by the subroutine.

Z() [out] Array in which the two roots are returned. The roots will be stored as complex numbers even in cases in which they are real.

B.3 Modifications for Double Precision

For double precision usage change the subroutine name to **DPOLZ2** and change the type statements to:

DOUBLE PRECISION A(≥ 3)

DOUBLE PRECISION Z($2, \geq 2$), or

COMPLEX*16 Z(≥ 2))

On return the real and imaginary parts of the j^{th} root will be stored in Z$(1, j)$ and Z$(2, j)$ respectively.

C. Examples and Remarks

The program, DRSPOLZ2, uses SPOLZ2 to compute the roots of the polynomials, $x^2 + x - 1$, $2x^2 - 12x + 26$, and $x^2 - 4x + 4$. The output is shown in ODSPOLZ2.

See the remark in Chapter 15.1, Section C, concerning the use of COMPLEX*16.

D. Functional Description

Method

We are given the real coefficients of a quadratic polynomial:

$$a_1x^2 + a_2x + a_3$$

The case of $a_1 = 0$ is regarded as an error. The subroutine issues an error message and returns, setting both roots to zero.

Compute $p = a_2/a_1$ and $q = a_3/a_1$. The cases of $p = 0$ or $q = 0$ are given special treatment. Otherwise we compute the roots of $x^2 + px + q$, knowing that p and q are both nonzero.

Compute $u = -p/2$. Direct use of the quadratic formula would give the roots as $r_1 = u + z$ and $r_2 = u - z$, where $z = \sqrt{u^2 - q}$. To extend the range of data for which results can be obtained without overflow or underflow, the expression for z is evaluated differently depending on the magnitude of u. We use thresholds, c_1 and c_2, such that $c_1^2/16$ is the underflow limit and $16c_2^2$ is the overflow limit. The values of c_1 and c_2 are set on the first call to this subroutine by use of the subprograms R1MACH or D1MACH from Chapter 19.1.

Rather than computing z, we compute $f = |z|$ and a quantity, d, having the same sign as $u^2 - q$:

If $|u| > c_2$
  $d = 1 - (q/u)/u$
  $f = |u|\sqrt{|d|}$
Elseif $|u| < c_1$
  $d = u(u/|q|) - \operatorname{sign} q$
  $f = \sqrt{|q|}\ \sqrt{|d|}$
Else
  $d = u^2 - q$
  $f = \sqrt{|d|}$
Endif

If $d = 0$, the polynomial has the real root, u, with multiplicity two. If $d < 0$, the roots are the complex numbers, (u, f) and $(u, -f)$.

If $d > 0$, the roots are the two real numbers, $u + f$ and $u - f$, however use of these expressions would cause unnecessary loss of accuracy in the root of smaller magnitude when the magnitudes of u and f are nearly the same. Instead we use the expressions, $r_1 = u + f \operatorname{sign} u$, and $r_2 = q/r_1$.

Accuracy tests

A test program ran cases exercising all of the branches of the subroutine. Accuracy was consistent with the computer being used.

E. Error Procedures and Restrictions

If the high-order coefficient, A(1), is zero, the subroutine issues an error message via ERMSG of Chapter 19.2,

with an error level of 0, and returns with both roots set to zero.

F. Supporting Information

The Source Language is ANSI Fortran 77.

Designed by C. L. Lawson, JPL, May 1986.

Programmed by C. L. Lawson and S. Y. Chiu, JPL, May 1986, Feb. 1987.

Entry	Required Files
DPOLZ2	AMACH, DPOLZ2, ERFIN, ERMSG
SPOLZ2	AMACH, ERFIN, ERMSG, SPOLZ2

DRSPOLZ2

```
c      Program DRSPOLZ2
c>> 2009-10-28 DRZPOLZ2 Krogh Mods to get comples used in single prec.
c>> 1996-07-03 DRSPOLZ2 Krogh Set for deriving single precision C vers.
c>> 1995-05-28 DRSPOLZ2 Krogh Moved formats up.
c>> 1994-08-09 DRSPOLZ2 WVS Removed '0' in format
c>> 1991-11-20 DRSPOLZ2 CLL Editing for Fortran 90.
c>> 1987-12-09 DRSPOLZ2 Lawson  Initial Code.
c Conversion should only be done from "D" to "S" for processing to C.
c--S replaces "?": DR?POLZ2, ?POLZ2
c      Demonstration driver for SPOLZ2.
c
      real               DC(3,3)
c++ CODE for .D. | .C. is inactive
C      real               Z(2,2)
c++ CODE for .S. & ~.C. is active
      complex Z(2)
c++ END

      integer  I, J, K, N
c
      data (DC(I,1),I=1,3) / 1.E0, 1.E0, -1.E0 /
      data (DC(I,2),I=1,3) / 2.E0, -12.E0, 26.E0 /
      data (DC(I,3),I=1,3) / 1.E0, -4.E0, 4.E0 /
      data N / 2 /
c
  100 format(' ','Degree =',I2/' ',
     *       'Coefficients =',6X,3(F10.4,1X))
  200 format(' ','Roots ='/(2(1X:'('',1X,F8.5,',',1X,F8.5,2X,')',2X)))
  300 format(//' ')
c
      do 40 J = 1,3
         print 100, N, (DC(I, J),I=1,3)
        call SPOLZ2(DC(1,J),Z)
c++ CODE for .D. & ~.C. is inactive
C         print '('' Roots ='')/(2(1X:'')('',1X,F8.5,'','',1X,F8.5,2X,
C     *     '')'',2X)))', (Z(1,K),Z(2,K),K=1,2)
c++ CODE for .S. & ~.C. is active
        print '('' Roots ='')/(2(1X:'')('',1X,F8.5,'','',1X,F8.5,2X,
     *     '')'',2X)))', (Z(K),K=1,2)
c++ CODE for .C. is inactive
c%%         printf( "\n Roots =\n  ( %8.5f, %8.5f )  ( %8.5f, %8.5f ) ",
c%%            z[0][0], z[0][1], z[1][0], z[1][1] );
c++ End
        print 300
   40 continue
      end
```

ODSPOLZ2

```
Degree = 2
Coefficients =          1.0000      1.0000     -1.0000
Roots =
( -1.61803,   0.00000  )   (  0.61803,   0.00000  )


Degree = 2
Coefficients =          2.0000    -12.0000     26.0000
Roots =
(  3.00000,   2.00000  )   (  3.00000,  -2.00000  )


Degree = 2
Coefficients =          1.0000     -4.0000      4.0000
Roots =
(  2.00000,   0.00000  )   (  2.00000,   0.00000  )
```