7.3 Compute Polynomial Coefficients from Roots

A. Purpose

Given n complex numbers, z_i, compute (complex) coefficients, c_i, such that the monic polynomial defined by

$$p(z) = c_1 z^n + ... + c_n z + c_{n+1}$$

has the numbers z_i as its roots. The coefficients will be computed from the relation

$$c_1 z^n + . . . + c_n z + c_{n+1} = (z - z_1)(z - z_2)^{\cdots}(z - z_n)$$

Note that c_1 is always set to 1.

B. Usage

B.1 Program Prototype, Single Precision

INTEGER NDEG

COMPLEX ROOTS($\geq$NDEG),**COEFS**($\geq$NDEG+1)

Assign values to NDEG and ROOTS().

$$\boxed{\textbf{CALL CCOEF (NDEG, ROOTS, COEFS)}}$$

Computed quantities are returned in COEFS().

B.2 Argument Definitions

NDEG [in] Number of roots given in ROOTS(), and thus the degree of the polynomial whose coefficients are to be computed.

ROOTS() [in] Roots, given as complex numbers.

COEFS() [out] Computed coefficients, stored as complex numbers. The arrays COEFS() and ROOTS() must be distinct. The coefficient, c_1, will be the coefficient of z^{NDEG} and will be set to 1.

B.3 Modifications for Double Precision

For double precision usage change the subroutine name from CCOEF to ZCOEF. Recall that the Fortran 77 standard does not support a double precision complex data type, although many Fortran compilers do, using the declaration, COMPLEX*16. To remain within the Fortran 77 standard, use the declarations

DOUBLE PRECISION ROOTS(2, $\geq$ NDEG), **COEFFS**(2, $\geq$NDEG+1)

and use the convention that real and imaginary parts of complex numbers are associated with the values 1 and 2, respectively, of the first subscript. This usage is compatible with the Fortran 90 standard. Alternatively, if the COMPLEX*16 declaration is available and is compatible with this storage convention, one may use the nonstandard declaration

COMPLEX*16 ROOTS($\geq$ NDEG), **COEFFS**($\geq$ NDEG+1)

C. Examples and Remarks

The program, DRZCOEF, with its output, ODZCOEF, illustrates the use of ZCOEF to compute the coefficients of a quadratic and a cubic polynomial.

If this subroutine is used to assess the accuracy of a polynomial root finder we suggest use of the double precision version, even if it is a single precision root finder, to reduce the introduction of errors from the process of computing the polynomial coefficients.

D. Functional Description

Method

The degree, NDEG, and roots, z_1, ..., z_{NDEG}, are given. The coefficients, c_i, are computed by the following algorithm. The quantities z_i and c_i are complex. In the double precision version the complex arithmetic is coded in-line in terms of operations on the real and imaginary parts to conform to the Fortran 77 standard.

```
c_1 = 1.0
if( NDEG .le. 0 ) return
c_2 = -z_1
do i = 2, NDEG
    c_{i+1} = -c_i * z_i
    do j = i, 2, -1
        c_j = c_j - c_{j-1} * z_i
    enddo
enddo
return
```

Accuracy tests

The logic and the accuracy of this code were checked by use with a root finder. The accuracy was consistent with the computer system being used.

E. Error Procedures and Restrictions

If NDEG $\leq$ 0 the subroutine returns, setting COEFS(1) = 1. The arrays ROOTS() and COEFS() must occupy distinct storage locations.

F. Supporting Information

The source language for these subroutines is ANSI Fortran 77.

Entry	**Required Files**
CCOEF	CCOEF
ZCOEF	ZCOEF

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

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

DRZCOEF

```
c       program DRZCOEF
c>> 1996-06-25 DRZCOEF Krogh Set for deriving C vers.
c>> 1994-07-15 CLL
c>> 1987-12-09 DRZCOEF  Lawson  Initial Code.
c Conversion should only be done from "Z" to "C" for processing to C.
c--Z replaces "?": DR?COEF, ?COEF
c       Demo driver for ZCOEF
c       C. L. Lawson & S. Chiu, JPL, 1987 Feb 17.
c       ------------------------------------------------------------------
      integer I, NDEG1, NDEG2
      double precision RT1(2,3), RT2(2,2)
      double precision ZC(2,4)
c       ------------------------------------------------------------------
c
      data (RT1(1,I),I=1,3) / 1.D0, 1.D0, 3.D0 /
      data (RT1(2,I),I=1,3) / 1.D0, -1.D0, 0.D0 /
      data (RT2(1,I),I=1,2) / 2.D0, 3.D0 /
      data (RT2(2,I),I=1,2) / 1.D0, 2.D0 /
      data NDEG1, NDEG2 / 3, 2 /
c
      call ZCOEF(NDEG1,RT1,ZC)
c++ CODE for ~.C. is active
  100 format(1X/1X,A,I3)
  200 format(1x,A/ (1X,'(',F12.9,',',F12.9,')':
     *                 '  (',F12.9,',',F12.9,')'))
      print 100,'Degree   =',NDEG1
      print 200,'Roots    =',(RT1(1,I),RT1(2,I),I=1,NDEG1)
      print 200,'Coeffs   =',(ZC(1,I),ZC(2,I),I=1,NDEG1+1)
      print '(/)'
c
      call ZCOEF(NDEG2,RT2,ZC)
      print 100,'Degree   =',NDEG2
      print 200,'Roots    =',(RT2(1,I),RT2(2,I),I=1,NDEG2)
      print 200,'Coeffs   =',(ZC(1,I),ZC(2,I),I=1,NDEG2+1)
c++ CODE for .C. is inactive
c%%    printf( " \n Degree   =%3ld", ndeg1 );
c%%    printf( " \n Roots    =\n" );
c%%    for (i = 0; i < ndeg1; i+=2){
c%%       printf( " (%12.9f,%12.9f )", rt1[i][0], rt1[i][1] );
c%%       if (i < ndeg1-1) printf( "  (%12.9f,%12.9f )", rt1[i+1][0],
c%%          rt1[i+1][1] );
c%%       printf( "\n");}
c%%    printf( " \n  Coeffs   =\n" );
c%%    for (i = 0; i <= ndeg1; i+=2){
c%%       printf( " (%12.9f,%12.9f )", zc[i][0], zc[i][1] );
c%%       if (i < ndeg1) printf( "  (%12.9f,%12.9f )", zc[i+1][0],
c%%          zc[i+1][1] );
c%%       printf( "\n");}
c%%
c%%    printf( "\n\n" );
c%%    zcoef( ndeg2, rt2, zc );
c%%    printf( " \n Degree   =%3ld", ndeg2 );
c%%    printf( " \n Roots    =\n" );
c%%    for (i = 0; i < ndeg2; i+=2){
c%%       printf( " (%12.9f,%12.9f )", rt2[i][0], rt2[i][1] );
c%%       if (i < ndeg2-1) printf( "  (%12.9f,%12.9f )", rt2[i+1][0],
```

```
c%%            rt2[i+1][1] );
c%%          printf( "\n");}
c%%     printf( " \n  Coeffs   =\n" );
c%%     for (i = 0; i <= ndeg2; i+=2){
c%%          printf( " (%12.9f,%12.9f )", zc[i][0], zc[i][1] );
c%%          if (i < ndeg2) printf( "  (%12.9f,%12.9f )", zc[i+1][0],
c%%            zc[i+1][1] );
c%%          printf( "\n");}
c++ END
      end
```

ODZCOEF

```
Degree    =  3
Roots     =
(  1.000000000,  1.000000000)  (  1.000000000,-1.000000000)
(  3.000000000,  0.000000000)
Coeffs    =
(  1.000000000,  0.000000000)  (-5.000000000,  0.000000000)
(  8.000000000,-0.000000000)  (-6.000000000,-0.000000000)


Degree    =  2
Roots     =
(  2.000000000,  1.000000000)  (  3.000000000,  2.000000000)
Coeffs    =
(  1.000000000,  0.000000000)  (-5.000000000,-3.000000000)
(  4.000000000,  7.000000000)
```