

C02AFF – NAG Fortran Library Routine Document

Note. Before using this routine, please read the Users' Note for your implementation to check the interpretation of bold italicised terms and other implementation-dependent details.

1 Purpose

C02AFF finds all the roots of a complex polynomial equation, using a variant of Laguerre's Method.

2 Specification

```
SUBROUTINE C02AFF(A, N, SCALE, Z, W, IFAIL)
  INTEGER          N, IFAIL
  real            A(2,N+1), Z(2,N), W(4*(N+1))
  LOGICAL          SCALE
```

3 Description

The routine attempts to find all the roots of the n th degree complex polynomial equation

$$P(z) = a_0 z^n + a_1 z^{n-1} + a_2 z^{n-2} + \dots + a_{n-1} z + a_n = 0.$$

The roots are located using a modified form of Laguerre's Method, originally proposed by Smith [2].

The method of Laguerre [3] can be described by the iterative scheme

$$L(z_k) = z_{k+1} - z_k = \frac{-n \times P(z_k)}{P'(z_k) \pm \sqrt{H(z_k)}},$$

where $H(z_k) = (n-1) \times [(n-1) \times (P'(z_k))^2 - n \times P(z_k)P''(z_k)]$, and z_0 is specified.

The sign in the denominator is chosen so that the modulus of the Laguerre step at z_k , viz. $|L(z_k)|$, is as small as possible. The method can be shown to be cubically convergent for isolated roots (real or complex) and linearly convergent for multiple roots.

The routine generates a sequence of iterates z_1, z_2, z_3, \dots , such that $|P(z_{k+1})| < |P(z_k)|$ and ensures that $z_{k+1} + L(z_{k+1})$ 'roughly' lies inside a circular region of radius $|F|$ about z_k known to contain a zero of $P(z)$; that is, $|L(z_{k+1})| \leq |F|$, where F denotes the Féjer bound (see Marden [1]) at the point z_k . Following Smith [2], F is taken to be $\min(B, 1.445 \times n \times R)$, where B is an upper bound for the magnitude of the smallest zero given by

$$B = 1.0001 \times \min(\sqrt{n} \times L(z_k), |r_1|, |a_n/a_0|^{1/n}),$$

r_1 is the zero X of smaller magnitude of the quadratic equation

$$(P''(z_k)/(2 \times n \times (n-1)))X^2 + (P'(z_k)/n)X + \frac{1}{2}P(z_k) = 0$$

and the Cauchy lower bound R for the smallest zero is computed (using Newton's Method) as the positive root of the polynomial equation

$$|a_0|z^n + |a_1|z^{n-1} + |a_2|z^{n-2} + \dots + |a_{n-1}|z - |a_n| = 0.$$

Starting from the origin, successive iterates are generated according to the rule $z_{k+1} = z_k + L(z_k)$ for $k = 1, 2, 3, \dots$ and $L(z_k)$ is 'adjusted' so that $|P(z_{k+1})| < |P(z_k)|$ and $|L(z_{k+1})| \leq |F|$. The iterative procedure terminates if $P(z_{k+1})$ is smaller in absolute value than the bound on the rounding error in $P(z_{k+1})$ and the current iterate $z_p = z_{k+1}$ is taken to be a zero of $P(z)$. The deflated polynomial $\tilde{P}(z) = P(z)/(z - z_p)$ of degree $n-1$ is then formed, and the above procedure is repeated on the deflated polynomial until $n < 3$, whereupon the remaining roots are obtained via the 'standard' closed formulae for a linear ($n = 1$) or quadratic ($n = 2$) equation.

To obtain the roots of a quadratic polynomial, C02AHF can be used.

4 References

- [1] Marden M (1966) Geometry of polynomials *Mathematical Surveys, no. 3* American Mathematical Society, Providence, RI
- [2] Smith B T (1967) ZERPOL: A zero finding algorithm for polynomials using Laguerre's method *Technical Report* Department of Computer Science, University of Toronto, Canada
- [3] Wilkinson J H (1965) *The Algebraic Eigenvalue Problem* Oxford University Press, London

5 Parameters

- 1: A(2,N+1) — *real* array *Input*
On entry: if A is declared with bounds (2,0:N), then A(1, i) and A(2, i) must contain the real and imaginary parts of a_i (i.e., the coefficient of z^{n-i}), for $i = 0, 1, \dots, n$.
Constraint: A(1,0) \neq 0.0 or A(2,0) \neq 0.0.
- 2: N — INTEGER *Input*
On entry:
Constraint: N \geq 1.
- 3: SCALE — LOGICAL *Input*
On entry: indicates whether or not the polynomial is to be scaled. See Section 8 for advice on when it may be preferable to set SCALE = .FALSE. and for a description of the scaling strategy.
Suggested value: SCALE = .TRUE..
- 4: Z(2,N) — *real* array *Output*
On exit: the real and imaginary parts of the roots are stored in Z(1, i) and Z(2, i) respectively, for $i = 1, 2, \dots, n$.
- 5: W(4*(N+1)) — *real* array *Workspace*
- 6: IFAIL — INTEGER *Input/Output*
On entry: IFAIL must be set to 0, -1 or 1. For users not familiar with this parameter (described in Chapter P01) the recommended value is 0.
On exit: IFAIL = 0 unless the routine detects an error (see Section 6).

6 Error Indicators and Warnings

If on entry IFAIL = 0 or -1, explanatory error messages are output on the current error message unit (as defined by X04AAF).

Errors detected by the routine:

IFAIL = 1

On entry, A(1,0) = 0.0 and A(2,0) = 0.0,
 or N < 1.

IFAIL = 2

The iterative procedure has failed to converge. This error is very unlikely to occur. If it does, please contact NAG immediately, as some basic assumption for the arithmetic has been violated. See also Section 8.

IFAIL = 3

Either overflow or underflow prevents the evaluation of $P(z)$ near some of its zeros. This error is very unlikely to occur. If it does, please contact NAG immediately. See also Section 8.

7 Accuracy

All roots are evaluated as accurately as possible, but because of the inherent nature of the problem complete accuracy cannot be guaranteed.

8 Further Comments

If `SCALE = .TRUE.`, then a scaling factor for the coefficients is chosen as a power of the base B of the machine so that the largest coefficient in magnitude approaches `THRESH = BEMAX-P`. Users should note that no scaling is performed if the largest coefficient in magnitude exceeds `THRESH`, even if `SCALE = .TRUE.`. (For definition of B , $EMAX$ and P see the X02 Chapter Introduction.)

However, with `SCALE = .TRUE.`, overflow may be encountered when the input coefficients $a_0, a_1, a_2, \dots, a_n$ vary widely in magnitude, particularly on those machines for which $B^{(4 \times P)}$ overflows. In such cases, `SCALE` should be set to `.FALSE.` and the coefficients scaled so that the largest coefficient in magnitude does not exceed $B^{(EMAX-2 \times P)}$.

Even so, the scaling strategy used in C02AFF is sometimes insufficient to avoid overflow and/or underflow conditions. In such cases, the user is recommended to scale the independent variable (z) so that the disparity between the largest and smallest coefficient in magnitude is reduced. That is, use the routine to locate the zeros of the polynomial $d \times P(cz)$ for some suitable values of c and d . For example, if the original polynomial was $P(z) = 2^{-100}i + 2^{100}z^{20}$, then choosing $c = 2^{-10}$ and $d = 2^{100}$, for instance, would yield the scaled polynomial $i + z^{20}$, which is well-behaved relative to overflow and underflow and has zeros which are 2^{10} times those of $P(z)$.

If the routine fails with `IFAIL = 2` or `3`, then the real and imaginary parts of any roots obtained before the failure occurred are stored in `Z` in the reverse order in which they were found. Let n_R denote the number of roots found before the failure occurred. Then `Z(1, n)` and `Z(2, n)` contain the real and imaginary parts of the first root found, `Z(1, n-1)` and `Z(2, n-1)` contain the real and imaginary parts of the second root found, ..., `Z(1, nR)` and `Z(2, nR)` contain the real and imaginary parts of the n_R th root found. After the failure has occurred, the remaining $2 \times (n - n_R)$ elements of `Z` contain a large negative number (equal to $-1/\text{X02AMF}(\sqrt{2})$).

9 Example

To find the roots of the polynomial

$$a_0 z^5 + a_1 z^4 + a_2 z^3 + a_3 z^2 + a_4 z + a_5 = 0,$$

where $a_0 = (5.0+6.0i)$, $a_1 = (30.0+20.0i)$, $a_2 = -(0.2+6.0i)$, $a_3 = (50.0+100000.0i)$, $a_4 = -(2.0-40.0i)$ and $a_5 = (10.0+1.0i)$.

9.1 Program Text

Note. The listing of the example program presented below uses bold italicised terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*      C02AFF Example Program Text
*      Mark 14 Release.  NAG Copyright 1989.
*      .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5, NOUT=6)
      INTEGER          MAXDEG
      PARAMETER        (MAXDEG=100)
      LOGICAL          SCALE
      PARAMETER        (SCALE=.TRUE.)
*      .. Local Scalars ..
      INTEGER          I, IFAIL, N
```

```

*    .. Local Arrays ..
      real          A(2,0:MAXDEG), W(4*MAXDEG+4), Z(2,MAXDEG)
*    .. External Subroutines ..
      EXTERNAL      CO2AFF
*    .. Executable Statements ..
      WRITE (NOUT,*) 'CO2AFF Example Program Results'
*    Skip heading in data file
      READ (NIN,*)
      READ (NIN,*) N
      IF (N.GT.0 .AND. N.LE.MAXDEG) THEN
          READ (NIN,*) (A(1,I),A(2,I),I=0,N)
          IFAIL = 0
*
          CALL CO2AFF(A,N,SCALE,Z,W,IFAIL)
*
          WRITE (NOUT,*)
          WRITE (NOUT,99999) 'Degree of polynomial = ', N
          WRITE (NOUT,*)
          WRITE (NOUT,*) 'Roots of polynomial'
          WRITE (NOUT,*)
          DO 20 I = 1, N
              WRITE (NOUT,99998) 'z = ', Z(1,I), Z(2,I), '*i'
20      CONTINUE
          ELSE
              WRITE (NOUT,*) 'N is out of range'
          END IF
          STOP
*
99999 FORMAT (1X,A,I4)
99998 FORMAT (1X,A,1P,e12.4,SP,e14.4,A)
      END

```

9.2 Program Data

C02AFF Example Program Data

```

5
  5.0      6.0
 30.0     20.0
 -0.2     -6.0
 50.0    100000.0
 -2.0     40.0
 10.0     1.0

```

9.3 Program Results

C02AFF Example Program Results

Degree of polynomial = 5

Roots of polynomial

```

z = -2.4328E+01 -4.8555E+00*i
z =  5.2487E+00 +2.2736E+01*i
z =  1.4653E+01 -1.6569E+01*i
z = -6.9264E-03 -7.4434E-03*i
z =  6.5264E-03 +7.4232E-03*i

```