NAG Fortran Library Routine Document

D03EAF

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

D03EAF solves Laplace's equation in two dimensions for an arbitrary domain bounded internally or externally by one or more closed contours, given the value of either the unknown function or its normal derivative (into the domain) at each point of the boundary.

2 Specification

```
 SUBROUTINE D03EAF(STAGE1, EXT, DORM, N, P, Q, X, Y, N1P1, PHI, PHID,
1                  ALPHA, C, IC, NP4, ICINT, NP1, IFAIL)
 INTEGER           N, N1P1, IC, NP4, ICINT(NP1), NP1, IFAIL
 real              P, Q, X(N1P1), Y(N1P1), PHI(N), PHID(N), ALPHA,
1                  C(IC,NP4)
 LOGICAL           STAGE1, EXT, DORM
```

3 Description

The routine uses an integral equation method, based upon Green's formula, which yields the solution, ϕ, within the domain, given its value or that of its normal derivative at each point of the boundary (except possibly at a finite number of discrete points). The solution is obtained in two stages. The first, which is executed once only, determines the complementary boundary values, i.e., ϕ, where its normal derivative is known and vice versa. The second stage is entered once for each point at which the solution is required.

The boundary is divided into a number of intervals in each of which ϕ and its normal derivative are approximated by constants. Of these half are evaluated by applying the given boundary conditions at one 'nodal' point within each interval while the remainder are determined (in stage 1) by solving a set of simultaneous linear equations. Here this is achieved by means of auxiliary routines F02WDF and F04JGF, which will yield the least-squares solution of an overdetermined system of equations as well as the unique solution of a square non-singular system.

In exterior domains the solution behaves as $c + s \log r + O(1/r)$ as r tends to infinity, where c is a constant, s is the total integral of the normal derivative around the boundary and r is the radial distance from the origin of co-ordinates. For the Neumann problem (when the normal derivative is given along the whole boundary) s is fixed by the boundary conditions whilst c is chosen by the user. However, for a Dirichlet problem (when ϕ is given along the whole boundary) or for a mixed problem, stage 1 produces a value of c for which $s = 0$; then as r tends to infinity the solution tends to the constant c.

4 References

Symm G T and Pitfield R A (1974) Solution of Laplace's equation in two dimensions *NPL Report NAC 44* National Physical Laboratory

5 Parameters

1: STAGE1 – LOGICAL *Input*

On entry: indicates whether the routine call is for stage 1 of the computation as defined in Section 3. If STAGE1 = .TRUE., then the call is for stage 1. If STAGE1 = .FALSE., then the call is for stage 2.

2: EXT – LOGICAL *Input*

On entry: the form of the domain. If EXT = .TRUE., the domain is unbounded. Otherwise the domain is an interior one.

3: DORM – LOGICAL *Input*

On entry: the form of the boundary conditions. If DORM = .TRUE., then the problem is a Dirichlet or mixed boundary value problem. Otherwise it is a Neumann problem.

4: N – INTEGER *Input*

On entry: the number of intervals into which the boundary is divided (see Section 7 and Section 8).

5: P – ***real*** *Input*
6: Q – ***real*** *Input*

On entry: to stage 2, P and Q must specify the x and y co-ordinates respectively of a point at which the solution is required.

When STAGE1 is .TRUE., P and Q are ignored.

7: X(N1P1) – ***real*** array *Input*
8: Y(N1P1) – ***real*** array *Input*

On entry: the x and y co-ordinates respectively of points on the one or more closed contours which define the domain of the problem.

Note: each contour is described in such a manner that the subscripts of the co-ordinates increase when the domain is kept on the left. The final point on each contour coincides with the first and, if a further contour is to be described, the co-ordinates of this point are repeated in the arrays. In this way each interval is defined by three points, the second of which (the nodal point) always has an even subscript. In the case of the interior Neumann problem, the outermost boundary contour must be given first, if there is more than one.

9: N1P1 – INTEGER *Input*

On entry: the value $2(\mathrm{N} + M) - 1$, where M denotes the number of closed contours which make up the boundary.

10: PHI(N) – ***real*** array *Input/Output*

On entry: for stage 1, PHI must contain the nodal values of ϕ or its normal derivative (into the domain) as prescribed in each interval. For stage 2 it must retain its output values from stage 1.

On exit: from stage 1, it contains the constants which approximate ϕ in each interval. It remains unchanged on exit from stage 2.

11: PHID(N) – ***real*** array *Input/Output*

On entry: for stage 1, PHID(i) must hold the value 0.0 or 1.0 according as PHI(i) contains a value of ϕ or its normal derivative, for $i = 1, 2, \ldots, \mathrm{N}$. For stage 2 it must retain its output values from stage 1.

On exit: from stage 1, PHID contains the constants which approximate the normal derivative of ϕ in each interval. It remains unchanged on exit from stage 2.

12: ALPHA – ***real*** *Input/Output*

On entry: for stage 1, the use of ALPHA depends on the nature of the problem:

if DORM = .TRUE., ALPHA need not be set.

if DORM = .FALSE. and EXT = .TRUE., ALPHA must contain the prescribed constant c (see Section 3).

if DORM = .FALSE. and EXT = .FALSE., ALPHA must contain an appropriate value (often zero) for the integral of ϕ around the outermost boundary.

For stage 2, on every call ALPHA must contain the value returned at stage 1.

On exit: from stage 1:

if EXT = .FALSE., ALPHA contains 0.0.

if EXT = .TRUE. and DORM = .FALSE., ALPHA is unchanged.

if EXT = .TRUE. and DORM = .TRUE., ALPHA contains a computed estimate for c.

From stage 2:

ALPHA contains the computed value of ϕ at the point (P,Q).

13: C(IC,NP4) – ***real*** array *Workspace*

14: IC – INTEGER *Input*

On entry: the first dimension of the array C as declared in the (sub)program from which D03EAF is called.

Constraint: IC $\geq$ N + 1.

15: NP4 – INTEGER *Input*

On entry: the value N + 4.

16: ICINT(NP1) – INTEGER array *Workspace*

17: NP1 – INTEGER *Input*

On entry: the value N + 1.

18: IFAIL – INTEGER *Input/Output*

On entry: IFAIL must be set to 0, −1 or 1. Users who are unfamiliar with this parameter should refer to Chapter P01 for details.

On exit: IFAIL = 0 unless the routine detects an error (see Section 6).

For environments where it might be inappropriate to halt program execution when an error is detected, the value −1 or 1 is recommended. If the output of error messages is undesirable, then the value 1 is recommended. Otherwise, for users not familiar with this parameter the recommended value is 0. **When the value −1 or 1 is used it is essential to test the value of IFAIL on exit.**

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 or warnings detected by the routine:

IFAIL = 1

Invalid tolerance used in an internal call to an auxiliary routine:

ICINT(1) = 0

indicates too large a tolerance.

ICINT(1) > 0

indicates too small a tolerance.

Note: this error is only possible in stage 1, and the circumstances under which it may occur cannot be foreseen. In the event of a failure, it is suggested that the user change the scale of the domain of the problem and apply the routine again.

IFAIL $= 2$

Incorrect rank obtained by an auxiliary routine; ICINT(1) contains the computed rank.

7 Accuracy

The accuracy of the computed solution depends upon how closely ϕ and its normal derivative may be approximated by constants in each interval of the boundary and upon how well the boundary contours are represented by polygons with vertices at the selected points $(\mathrm{X}(i), \mathrm{Y}(i))$, $i = 1, 2, \ldots, 2(\mathrm{N} + \mathrm{M}) - 1$.

Consequently, in general, the accuracy increases as the boundary is subdivided into smaller and smaller intervals and by comparing solutions for successive subdivisions one may obtain an indication of the error in these solutions.

Alternatively, since the point of maximum error always lies on the boundary of the domain, an estimate of the error may be obtained by computing ϕ at a sufficient number of points on the boundary where the true solution is known. The latter method (not applicable to the Neumann problem) is most useful in the case where ϕ alone is prescribed on the boundary (the Dirichlet problem).

8 Further Comments

The time taken by the routine for stage 1, which is executed once only, is roughly proportional to N^2, being dominated by the time taken to compute the coefficients. The time for each stage 2 application of the routine is proportional to N.

The intervals into which the boundary is divided need not be of equal lengths.

For many practical problems useful results may be obtained with 20 to 40 intervals per boundary contour.

9 Example

An interior Neumann problem to solve Laplace's equation in the domain bounded externally by the triangle with vertices (3,0), $(-3, 0)$ and (0,4), and internally by the triangle with vertices (2,1), $(-2, 1)$ and (0,3), given that the normal derivative of the solution ϕ is zero on each side of each triangle and, for uniqueness that the total integral of ϕ around the outer triangle is 16 (the length of the contour).

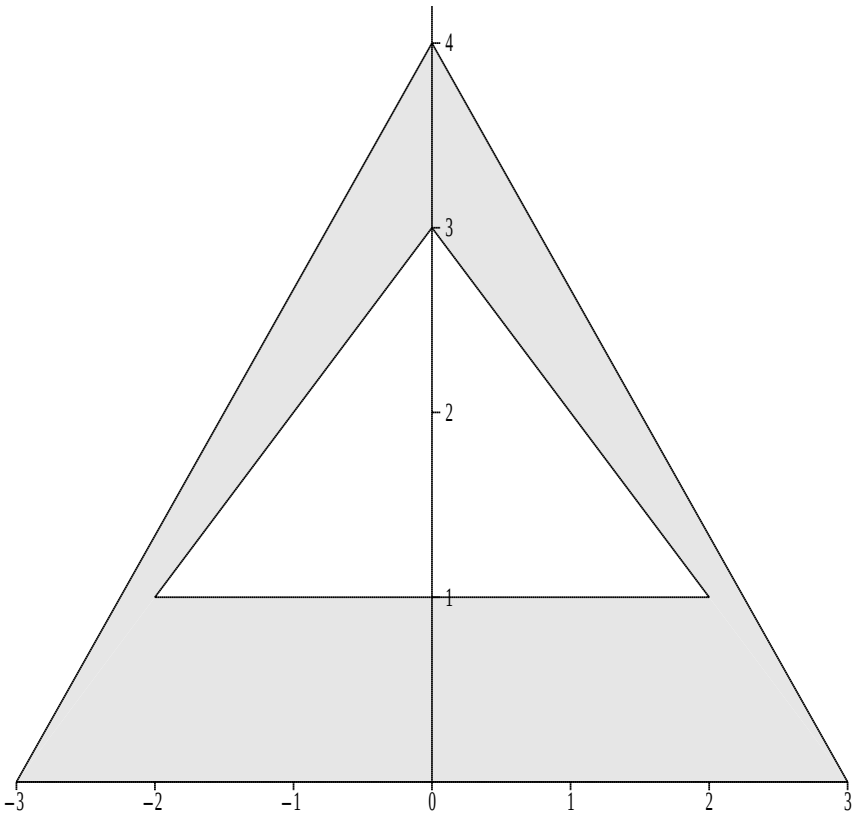

Figure 1

This trivial example has the obvious solution $\phi = 1$ throughout the domain. However it provides a useful illustration of data input for a doubly-connected domain. The solution is computed at one corner of each triangle and at one point inside the domain.

The program is written to handle any of the different types of problem that the routine can solve. The array dimensions must be increased for larger problems.

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.

```
*     D03EAF Example Program Text
*     Mark 14 Revised.  NAG Copyright 1989.
*     .. Parameters ..
      INTEGER          N, M, NP1, IC, NP4, N1, N1P1
      PARAMETER        (N=6,M=2,NP1=N+1,IC=N+1,NP4=N+4,N1=2*(N+M)-2,
     +                 N1P1=N1+1)
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
*     .. Local Scalars ..
      real             ALPHA, C, P, Q
      INTEGER          I, IFAIL, J, NPTS
      LOGICAL          DORM, EXT, STGONE
*     .. Local Arrays ..
      real             C1(IC,NP4), PHI(N), PHID(N), X(N1P1), Y(N1P1)
      INTEGER          ICINT(NP1)
*     .. External Subroutines ..
      EXTERNAL         D03EAF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'D03EAF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
```

```
      READ (NIN,*) EXT, DORM
      STGONE = .TRUE.
      WRITE (NOUT,*)
      IF ( .NOT. EXT .AND. .NOT. DORM) THEN
         READ (NIN,*) ALPHA
         WRITE (NOUT,*) 'Interior Neumann problem'
         WRITE (NOUT,*)
         WRITE (NOUT,99999) 'Total integral =', ALPHA
      ELSE
         IF (EXT .AND. .NOT. DORM) THEN
            READ (NIN,*) ALPHA
            WRITE (NOUT,*) 'Exterior Neumann problem'
            WRITE (NOUT,*)
            WRITE (NOUT,99998) 'C=', ALPHA
         END IF
      END IF
      DO 20 I = 1, N1 + 1
         READ (NIN,*) X(I), Y(I)
   20 CONTINUE
      DO 40 I = 1, N
         READ (NIN,*) PHI(I), PHID(I)
   40 CONTINUE
      IFAIL = 1
*
      CALL D03EAF(STGONE,EXT,DORM,N,P,Q,X,Y,N1P1,PHI,PHID,ALPHA,C1,IC,
     +            NP4,ICINT,NP1,IFAIL)
*
      IF (IFAIL.NE.0) THEN
         WRITE (NOUT,*)
         WRITE (NOUT,99996) 'Error in D03EAF IFAIL = ', IFAIL
         WRITE (NOUT,*)
         WRITE (NOUT,99996) 'The value of RANK is ', ICINT(1)
         STOP
      END IF
      C = ALPHA
      IF (EXT .AND. DORM) THEN
         WRITE (NOUT,*)
         WRITE (NOUT,*) 'Exterior problem'
         WRITE (NOUT,*)
         WRITE (NOUT,99999) 'Computed C =', C
      END IF
      J = 2
      WRITE (NOUT,*)
      WRITE (NOUT,*) 'Nodes'
      WRITE (NOUT,*)
     +  '           X              Y              PHI            PHID'
      DO 60 I = 1, N
         IF (X(J).EQ.X(J-1) .AND. Y(J).EQ.Y(J-1)) J = J + 2
         WRITE (NOUT,99997) X(J), Y(J), PHI(I), PHID(I)
         J = J + 2
   60 CONTINUE
      STGONE = .FALSE.
      WRITE (NOUT,*)
      WRITE (NOUT,*) 'Selected points'
      WRITE (NOUT,*) '           X              Y              PHI'
      READ (NIN,*) NPTS
      DO 80 I = 1, NPTS
         READ (NIN,*) P, Q
         ALPHA = C
*
         CALL D03EAF(STGONE,EXT,DORM,N,P,Q,X,Y,N1P1,PHI,PHID,ALPHA,C1,
     +               IC,NP4,ICINT,NP1,IFAIL)
*
         WRITE (NOUT,99997) P, Q, ALPHA
   80 CONTINUE
      STOP
*
99999 FORMAT (1X,A,F15.2)
99998 FORMAT (1X,A,1P,E15.4)
99997 FORMAT (1X,4F15.2)
99996 FORMAT (1X,A,I2)
```

```
      END
```

9.2 Program Data

```
D03EAF Example Program Data
F F
 16.0
  3.0  0.0
  1.5  2.0
  0.0  4.0
 -1.5  2.0
 -3.0  0.0
  0.0  0.0
  3.0  0.0
  3.0  0.0
  2.0  1.0
  0.0  1.0
 -2.0  1.0
 -1.0  2.0
  0.0  3.0
  1.0  2.0
  2.0  1.0
  0.0  1.0
  0.0  1.0
  0.0  1.0
  0.0  1.0
  0.0  1.0
  0.0  1.0
3
  2.0  1.0
  2.5  0.5
  3.0  0.0
```

9.3 Program Results

```
 D03EAF Example Program Results

 Interior Neumann problem

 Total integral =          16.00

 Nodes
            X              Y              PHI            PHID
          1.50           2.00           1.00           0.00
         -1.50           2.00           1.00           0.00
          0.00           0.00           1.00           0.00
          0.00           1.00           1.00           0.00
         -1.00           2.00           1.00           0.00
          1.00           2.00           1.00           0.00

 Selected points
            X              Y              PHI
          2.00           1.00           1.00
          2.50           0.50           1.00
          3.00           0.00           1.00
```