NAG Library Routine Document

D03MAF

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

D03MAF places a triangular mesh over a given two-dimensional region. The region may have any shape, including one with holes.

2 Specification

```
SUBROUTINE D03MAF (H, M, N, NB, NPTS, PLACES, INDX, SDINDX, ISIN, DIST,    &
                   SDDIST, IFAIL)
INTEGER            M, N, NB, NPTS, INDX(4,SDINDX), SDINDX, ISIN,           &
                   SDDIST, IFAIL
REAL (KIND=nag_wp) H, PLACES(2,SDINDX), DIST(4,SDDIST)
EXTERNAL           ISIN
```

3 Description

D03MAF begins with a uniform triangular grid as shown in Figure 1 and assumes that the region to be triangulated lies within the rectangle given by the inequalities

$$0 < x < \sqrt{3}(m-1)h, \quad 0 < y < (n-1)h.$$

This rectangle is drawn in bold in Figure 1. The region is specified by the ISIN which must determine whether any given point (x, y) lies in the region. The uniform grid is processed column-wise, with (x_1, y_1) preceding (x_2, y_2) if $x_1 < x_2$ or $x_1 = x_2$, $y_1 < y_2$. Points near the boundary are moved onto it and points well outside the boundary are omitted. The direction of movement is chosen to avoid pathologically thin triangles. The points accepted are numbered in exactly the same order as the corresponding points of the uniform grid were scanned. The output consists of the x, y coordinates of all grid points and integers indicating whether they are internal and to which other points they are joined by triangle sides.

The mesh size h must be chosen small enough for the essential features of the region to be apparent from testing all points of the original uniform grid for being inside the region. For instance if any hole is within $2h$ of another hole or the outer boundary then a triangle may be found with all vertices within $\frac{1}{2}h$ of a boundary. Such a triangle is taken to be external to the region so the effect will be to join the hole to another hole or to the external region.

Further details of the algorithm are given in the references.

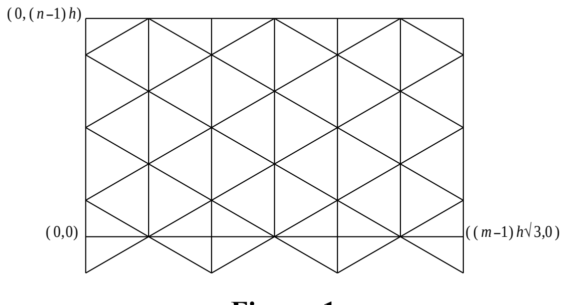


Figure 1

4 References

Reid J K (1970) Fortran subroutines for the solutions of Laplace's equation over a general routine in two dimensions *Harwell Report TP422*

Reid J K (1972) On the construction and convergence of a finite-element solution of Laplace's equation *J. Instr. Math. Appl.* **9** 1–13

5 Arguments

1: H – REAL (KIND=nag_wp) *Input*

On entry: h, the required length for the sides of the triangles of the uniform mesh.

2: M – INTEGER *Input*
3: N – INTEGER *Input*

On entry: values m and n such that all points (x, y) inside the region satisfy the inequalities

$$0 \leq x \leq \sqrt{3}(m-1)h,$$
$$0 \leq y \leq (n-1)h.$$

Constraint: $\mathrm{M} = \mathrm{N} > 2$.

4: NB – INTEGER *Input*

On entry: the number of times a triangle side is bisected to find a point on the boundary. A value of 10 is adequate for most purposes (see Section 7).

Constraint: $\mathrm{NB} \geq 1$.

5: NPTS – INTEGER *Output*

On exit: the number of points in the triangulation.

6: PLACES$(2, \mathrm{SDINDX})$ – REAL (KIND=nag_wp) array *Output*

On exit: the x and y coordinates respectively of the ith point of the triangulation.

7: INDX$(4, \mathrm{SDINDX})$ – INTEGER array *Output*

On exit: $\mathrm{INDX}(1, i)$ contains i if point i is inside the region and $-i$ if it is on the boundary. For each triangle side between points i and j with $j > i$, $\mathrm{INDX}(k, i)$, $k > 1$, contains j or $-j$ according to whether point j is internal or on the boundary. There can never be more than three such points. If there are less, then some values $\mathrm{INDX}(k, i)$, $k > 1$, are zero.

8: SDINDX – INTEGER *Input*

On entry: the second dimension of the arrays PLACES and INDX as declared in the (sub) program from which D03MAF is called.

Constraint: $\mathrm{SDINDX} \geq \mathrm{NPTS}$.

9: ISIN – INTEGER FUNCTION, supplied by the user. *External Procedure*

ISIN must return the value 1 if the given point (X,Y) lies inside the region, and 0 if it lies outside.

The specification of ISIN is:

```
FUNCTION ISIN (X, Y)
INTEGER ISIN

REAL (KIND=nag_wp) X, Y
```

1: X – REAL (KIND=nag_wp) *Input*
2: Y – REAL (KIND=nag_wp) *Input*

On entry: the coordinates of the given point.

ISIN must either be a module subprogram USEd by, or declared as EXTERNAL in, the (sub) program from which D03MAF is called. Arguments denoted as *Input* must **not** be changed by this procedure.

10: DIST(4, SDDIST) – REAL (KIND=nag_wp) array *Workspace*
11: SDDIST – INTEGER *Input*

On entry: the second dimension of the array DIST as declared in the (sub)program from which D03MAF is called.

Constraint: SDDIST $\geq$ 4N.

12: IFAIL – INTEGER *Input/Output*

On entry: IFAIL must be set to 0, -1 or 1. If you are unfamiliar with this argument you should refer to Section 3.4 in How to Use the NAG Library and its Documentation for details.

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, if you are not familiar with this argument, the recommended value is 0. **When the value -1 or 1 is used it is essential to test the value of IFAIL on exit.**

On exit: IFAIL $= 0$ unless the routine detects an error or a warning has been flagged (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 or warnings detected by the routine:

IFAIL $= 1$

SDINDX is too small.

IFAIL $= 2$

A point inside the region violates one of the constraints (see arguments M and N).

IFAIL $= 3$

SDDIST is too small.

IFAIL $= 4$

M ≤ 2.

IFAIL $= 5$

N ≤ 2.

IFAIL $= 6$

NB ≤ 0.

IFAIL $= -99$

An unexpected error has been triggered by this routine. Please contact NAG.

See Section 3.9 in How to Use the NAG Library and its Documentation for further information.

IFAIL $= -399$

Your licence key may have expired or may not have been installed correctly.

See Section 3.8 in How to Use the NAG Library and its Documentation for further information.

IFAIL $= -999$

Dynamic memory allocation failed.

See Section 3.7 in How to Use the NAG Library and its Documentation for further information.

7 Accuracy

Points are moved onto the boundary by bisecting a triangle side NB times. The accuracy is therefore $h \times 2^{-\mathrm{NB}}$.

8 Parallelism and Performance

D03MAF is not threaded in any implementation.

9 Further Comments

The time taken is approximately proportional to $m \times n$.

10 Example

The following program triangulates the circle with centre $(7.0, 7.0)$ and radius 6.0 using a basic grid size $h = 4.0$.

10.1 Program Text

```
!   D03MAF Example Program Text
!   Mark 26 Release. NAG Copyright 2016.

    Module d03mafe_mod

!     D03MAF Example Program Module:
!            Parameters and User-defined Routines

!     .. Use Statements ..
      Use nag_library, Only: nag_wp
!     .. Implicit None Statement ..
      Implicit None
!     .. Accessibility Statements ..
      Private
      Public                           :: isin
!     .. Parameters ..
      Real (Kind=nag_wp), Parameter    :: rad = 6.0_nag_wp
      Real (Kind=nag_wp), Parameter    :: xmid = 7.0_nag_wp
      Real (Kind=nag_wp), Parameter    :: ymid = 7.0_nag_wp
      Integer, Parameter, Public       :: nin = 5, nout = 6
    Contains
      Function isin(x,y)
!       Circular domain

!       .. Function Return Value ..
        Integer                        :: isin
!       .. Scalar Arguments ..
```

```
        Real (Kind=nag_wp), Intent (In) :: x, y
!       .. Executable Statements ..
        If ((x-xmid)**2+(y-ymid)**2<=rad**2) Then
          isin = 1
        Else
          isin = 0
        End If
        Return
      End Function isin
    End Module d03mafe_mod

    Program d03mafe

!     D03MAF Example Main Program

!     .. Use Statements ..
      Use nag_library, Only: d03maf, nag_wp
      Use d03mafe_mod, Only: isin, nin, nout
!     .. Implicit None Statement ..
      Implicit None
!     .. Local Scalars ..
      Real (Kind=nag_wp)                :: h
      Integer                           :: i, ifail, m, n, nb, npts, sddist,    &
                                           sdindx
!     .. Local Arrays ..
      Real (Kind=nag_wp), Allocatable   :: dist(:,:), places(:,:)
      Integer, Allocatable              :: indx(:,:)
!     .. Executable Statements ..
      Write (nout,*) 'D03MAF Example Program Results'
      Write (nout,*)
!     Skip heading in data file
      Read (nin,*)
      Read (nin,*) sddist, sdindx
      Allocate (dist(4,sddist),places(2,sdindx),indx(4,sdindx))

      Read (nin,*) h
      Read (nin,*) m, n, nb

!     ifail: behaviour on error exit
!            =0 for hard exit, =1 for quiet-soft, =-1 for noisy-soft
      ifail = 0
      Call d03maf(h,m,n,nb,npts,places,indx,sdindx,isin,dist,sddist,ifail)

      Write (nout,*) '  I    X(I)      Y(I)'
      Do i = 1, npts
        Write (nout,99999) i, places(1,i), places(2,i)
      End Do
      Write (nout,*)
      Write (nout,*) 'INDX'
      Write (nout,99998)(indx(1:4,i),i=1,npts)

99999 Format (1X,I3,2F10.6)
99998 Format (1X,4I5)
    End Program d03mafe
```

10.2 Program Data

```
D03MAF Example Program Data
  20 100              : sddist, sdindx
  4.0                 : h
  3 5 10              : m, n, nb
```

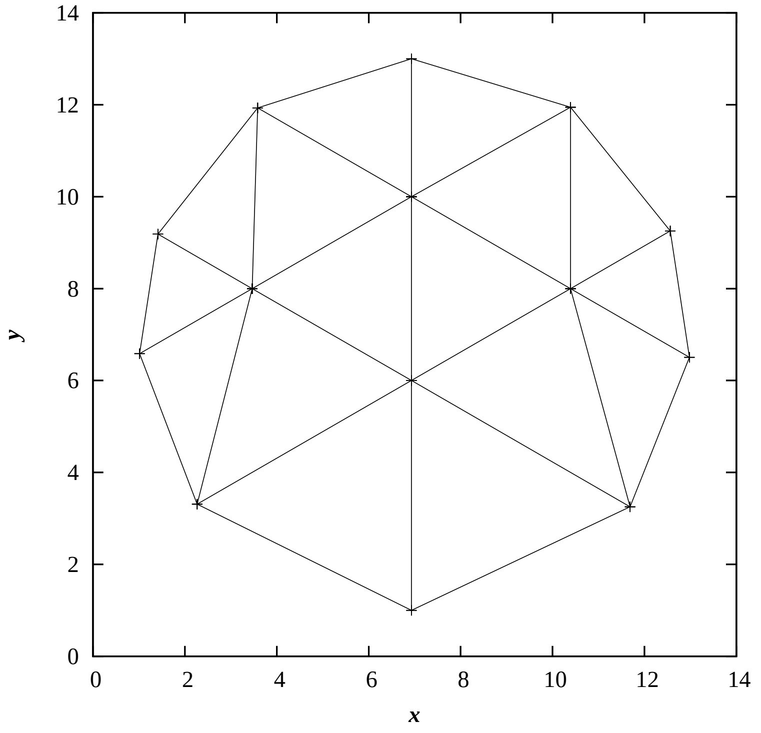
10.3 Program Results

```
 D03MAF Example Program Results

   I    X(I)      Y(I)
   1  1.013182  6.584961
   2  1.412366  9.184570
   3  2.268242  3.309570
   4  3.464102  8.000000
   5  3.584195 11.930664
   6  6.928203  1.001953
   7  6.928203  6.000000
   8  6.928203 10.000000
   9  6.928203 12.998047
  10 11.686269  3.252930
  11 10.392305  8.000000
  12 10.392305 11.947266
  13 12.978541  6.506836
  14 12.562443  9.252930

 INDX
    -1    -3     4    -2
    -2     4    -5     0
    -3    -6     7     4
     4     7     8    -5
    -5     8    -9     0
    -6     0   -10     7
     7   -10    11     8
     8    11   -12    -9
    -9   -12     0     0
   -10     0   -13    11
    11   -13   -14   -12
   -12   -14     0     0
   -13     0     0   -14
   -14     0     0     0
```

Example Program
Triangulation of a Circle
with centre (7,7) and radius 6 using grid size=4


14
12
10
8
6
4
2
0
y
0
2
4
6
8
10
12
14
x