

# NAG Library Routine Document

## D06BAF

**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

D06BAF generates a boundary mesh on a closed connected subdomain  $\Omega$  of  $\mathbb{R}^2$ .

### 2 Specification

```

SUBROUTINE D06BAF (NLINES, COORCH, LINED, FBND, CRUS, SDCRUS, RATE,      &
                  NCOMP, NLCOMP, LCOMP, NVMAX, NEDMX, NVB, COOR, NEDGE,  &
                  EDGE, ITRACE, RUSER, IUSER, RWORK, LRWORK, IWORK,     &
                  LIWORK, IFAIL)
INTEGER            NLINES, LINED(4,NLINES), SDCRUS, NCOMP,              &
                  NLCOMP(NCOMP), LCOMP(NLINES), NVMAX, NEDMX, NVB,     &
                  NEDGE, EDGE(3,NEDMX), ITRACE, IUSER(*), LRWORK,     &
                  IWORK(LIWORK), LIWORK, IFAIL
REAL (KIND=nag_wp) COORCH(2,NLINES), FBND, CRUS(2,SDCRUS),           &
                  RATE(NLINES), COOR(2,NVMAX), RUSER(*),              &
                  RWORK(LRWORK)
EXTERNAL          FBND

```

### 3 Description

Given a closed connected subdomain  $\Omega$  of  $\mathbb{R}^2$ , whose boundary  $\partial\Omega$  is divided by characteristic points into  $m$  distinct line segments, D06BAF generates a boundary mesh on  $\partial\Omega$ . Each line segment may be a straight line, a curve defined by the equation  $f(x, y) = 0$ , or a polygonal curve defined by a set of given boundary mesh points.

This routine is primarily designed for use with either D06AAF (a simple incremental method) or D06ABF (Delaunay–Voronoi method) or D06ACF (Advancing Front method) to triangulate the interior of the domain  $\Omega$ . For more details about the boundary and interior mesh generation, consult the D06 Chapter Introduction as well as George and Borouchaki (1998).

This routine is derived from material in the MODULEF package from INRIA (Institut National de Recherche en Informatique et Automatique).

### 4 References

George P L and Borouchaki H (1998) *Delaunay Triangulation and Meshing: Application to Finite Elements* Editions HERMES, Paris

### 5 Arguments

1: NLINES – INTEGER *Input*

*On entry:*  $m$ , the number of lines that define the boundary of the closed connected subdomain (this equals the number of characteristic points which separate the entire boundary  $\partial\Omega$  into lines).

*Constraint:*  $NLINES \geq 1$ .

2: COORCH(2,NLINES) – REAL (KIND=nag\_wp) array *Input*

*On entry:* COORCH(1,  $i$ ) contains the  $x$  coordinate of the  $i$ th characteristic point, for  $i = 1, 2, \dots, NLINES$ ; while COORCH(2,  $i$ ) contains the corresponding  $y$  coordinate.

3: LINED(4,NLINES) – INTEGER array *Input*

*On entry:* the description of the lines that define the boundary domain. The line  $i$ , for  $i = 1, 2, \dots, m$ , is defined as follows:

LINED(1,  $i$ )

The number of points on the line, including two end points.

LINED(2,  $i$ )

The first end point of the line. If LINED(2,  $i$ ) =  $j$ , then the coordinates of the first end point are those stored in COORCH(:,  $j$ ).

LINED(3,  $i$ )

The second end point of the line. If LINED(3,  $i$ ) =  $k$ , then the coordinates of the second end point are those stored in COORCH(:,  $k$ ).

LINED(4,  $i$ )

This defines the type of line segment connecting the end points. Additional information is conveyed by the numerical value of LINED(4,  $i$ ) as follows:

- (i) LINED(4,  $i$ ) > 0, the line is described in FBND with LINED(4,  $i$ ) as the index. In this case, the line must be described in the trigonometric (anticlockwise) direction;
- (ii) LINED(4,  $i$ ) = 0, the line is a straight line;
- (iii) if LINED(4,  $i$ ) < 0, say  $(-p)$ , then the line is a polygonal arc joining the end points and interior points specified in CRUS. In this case the line contains the points whose coordinates are stored in  
 COORCH(:,  $j$ ),  
 CRUS(:,  $p$ ),  
 CRUS(:,  $p + 1$ ), ..., CRUS(:,  $p + r - 3$ ),  
 COORCH(:,  $k$ ),  
 where  $z \in \{1, 2\}$ ,  $r = \text{LINED}(1, i)$ ,  $j = \text{LINED}(2, i)$  and  $k = \text{LINED}(3, i)$ .

*Constraints:*

- $2 \leq \text{LINED}(1, i)$ ;
- $1 \leq \text{LINED}(2, i) \leq \text{NLINES}$ ;
- $1 \leq \text{LINED}(3, i) \leq \text{NLINES}$ ;
- $\text{LINED}(2, i) \neq \text{LINED}(3, i)$ , for  $i = 1, 2, \dots, \text{NLINES}$ .

For each line described by FBND (lines with LINED(4,  $i$ ) > 0, for  $i = 1, 2, \dots, \text{NLINES}$ ) the two end points (LINED(2,  $i$ ) and LINED(3,  $i$ )) lie on the curve defined by index LINED(4,  $i$ ) in FBND, i.e.,

FBND(LINED(4,  $i$ ), COORCH(1, LINED(2,  $i$ )), COORCH(2, LINED(2,  $i$ )), RUSER, IUSER) = 0;

FBND(LINED(4,  $i$ ), COORCH(1, LINED(3,  $i$ )), COORCH(2, LINED(3,  $i$ )), RUSER, IUSER) = 0, for  $i = 1, 2, \dots, \text{NLINES}$ .

For all lines described as polygonal arcs (lines with LINED(4,  $i$ ) < 0, for  $i = 1, 2, \dots, \text{NLINES}$ ) the sets of intermediate points (i.e.,  $[-\text{LINED}(4, i) : -\text{LINED}(4, i) + \text{LINED}(1, i) - 3]$  for all  $i$  such that LINED(4,  $i$ ) < 0) are not overlapping. This can be expressed as:

$$-\text{LINED}(4, i) + \text{LINED}(1, i) - 3 = \sum_{\{i, \text{LINED}(4, i) < 0\}} \{\text{LINED}(1, i) - 2\}$$

or

$$-\text{LINED}(4, i) + \text{LINED}(1, i) - 2 = -\text{LINED}(4, j),$$

for a  $j$  such that  $j = 1, 2, \dots, \text{NLINES}$ ,  $j \neq i$  and LINED(4,  $j$ ) < 0.

4: FBND – REAL (KIND=nag\_wp) FUNCTION, supplied by the user. *External Procedure*

FBND must be supplied to calculate the value of the function which describes the curve  $\{(x, y) \in \mathbb{R}^2; \text{ such that } f(x, y) = 0\}$  on segments of the boundary for which LINED(4,  $i$ ) > 0. If

there are no boundaries for which  $\text{LINED}(4, i) > 0$  FBND will never be referenced by D06BAF and FBND may be the dummy function D06BAD. (D06BAD is included in the NAG Library.)

The specification of FBND is:

```
FUNCTION FBND (I, X, Y, RUSER, IUSER)
REAL (KIND=nag_wp) FBND
INTEGER          I, IUSER(*)
REAL (KIND=nag_wp) X, Y, RUSER(*)
```

1: I – INTEGER *Input*

*On entry:*  $\text{LINED}(4, i)$ , the reference index of the line (portion of the contour)  $i$  described.

2: X – REAL (KIND=nag\_wp) *Input*

3: Y – REAL (KIND=nag\_wp) *Input*

*On entry:* the values of  $x$  and  $y$  at which  $f(x, y)$  is to be evaluated.

4: RUSER(\*) – REAL (KIND=nag\_wp) array *User Workspace*

5: IUSER(\*) – INTEGER array *User Workspace*

FBND is called with the arguments RUSER and IUSER as supplied to D06BAF. You should use the arrays RUSER and IUSER to supply information to FBND.

FBND must either be a module subprogram USED by, or declared as EXTERNAL in, the (sub) program from which D06BAF is called. Arguments denoted as *Input* must **not** be changed by this procedure.

5: CRUS(2, SDCRUS) – REAL (KIND=nag\_wp) array *Input*

*On entry:* the coordinates of the intermediate points for polygonal arc lines. For a line  $i$  defined as a polygonal arc (i.e.,  $\text{LINED}(4, i) < 0$ ), if  $p = -\text{LINED}(4, i)$ , then  $\text{CRUS}(1, k)$ , for  $k = p, \dots, p + \text{LINED}(1, i) - 3$ , must contain the  $x$  coordinate of the consecutive intermediate points for this line. Similarly  $\text{CRUS}(2, k)$ , for  $k = p, \dots, p + \text{LINED}(1, i) - 3$ , must contain the corresponding  $y$  coordinate.

6: SDCRUS – INTEGER *Input*

*On entry:* the second dimension of the array CRUS as declared in the (sub)program from which D06BAF is called.

*Constraint:*  $\text{SDCRUS} \geq \sum_{\{i, \text{LINED}(4, i) < 0\}} \{\text{LINED}(1, i) - 2\}$ .

7: RATE(NLINES) – REAL (KIND=nag\_wp) array *Input*

*On entry:*  $\text{RATE}(i)$  is the geometric progression ratio between the points to be generated on the line  $i$ , for  $i = 1, 2, \dots, m$  and  $\text{LINED}(4, i) \geq 0$ .

If  $\text{LINED}(4, i) < 0$ ,  $\text{RATE}(i)$  is not referenced.

*Constraint:* if  $\text{LINED}(4, i) \geq 0$ ,  $\text{RATE}(i) > 0.0$ , for  $i = 1, 2, \dots, \text{NLINES}$ .

8: NCOMP – INTEGER *Input*

*On entry:*  $n$ , the number of separately connected components of the boundary.

*Constraint:*  $\text{NCOMP} \geq 1$ .

- 9: NLCOMP(NCOMP) – INTEGER array *Input*  
*On entry:*  $|\text{NLCOMP}(k)|$  is the number of line segments in component  $k$  of the contour. The line  $i$  of component  $k$  runs in the direction  $\text{LINED}(2, i)$  to  $\text{LINED}(3, i)$  if  $\text{NLCOMP}(k) > 0$ , and in the opposite direction otherwise; for  $k = 1, 2, \dots, n$ .  
*Constraints:*  

$$1 \leq |\text{NLCOMP}(k)| \leq \text{NLINES}, \text{ for } k = 1, 2, \dots, \text{NCOMP};$$

$$\sum_{k=1}^n |\text{NLCOMP}(k)| = \text{NLINES}.$$
- 10: LCOMP(NLINES) – INTEGER array *Input*  
*On entry:* LCOMP must contain the list of line numbers for the each component of the boundary. Specifically, the line numbers for the  $k$ th component of the boundary, for  $k = 1, 2, \dots, \text{NCOMP}$ , must be in elements  $l1 - 1$  to  $l2 - 1$  of LCOMP, where  $l2 = \sum_{i=1}^k |\text{NLCOMP}(i)|$  and  $l1 = l2 + 1 - |\text{NLCOMP}(k)|$ .  
*Constraint:* LCOMP must hold a valid permutation of the integers  $[1, \text{NLINES}]$ .
- 11: NVMAX – INTEGER *Input*  
*On entry:* the maximum number of the boundary mesh vertices to be generated.  
*Constraint:*  $\text{NVMAX} \geq \text{NLINES}$ .
- 12: NEDMX – INTEGER *Input*  
*On entry:* the maximum number of boundary edges in the boundary mesh to be generated.  
*Constraint:*  $\text{NEDMX} \geq 1$ .
- 13: NVB – INTEGER *Output*  
*On exit:* the total number of boundary mesh vertices generated.
- 14: COOR(2, NVMAX) – REAL (KIND=nag\_wp) array *Output*  
*On exit:*  $\text{COOR}(1, i)$  will contain the  $x$  coordinate of the  $i$ th boundary mesh vertex generated, for  $i = 1, 2, \dots, \text{NVB}$ ; while  $\text{COOR}(2, i)$  will contain the corresponding  $y$  coordinate.
- 15: NEDGE – INTEGER *Output*  
*On exit:* the total number of boundary edges in the boundary mesh.
- 16: EDGE(3, NEDMX) – INTEGER array *Output*  
*On exit:* the specification of the boundary edges.  $\text{EDGE}(1, j)$  and  $\text{EDGE}(2, j)$  will contain the vertex numbers of the two end points of the  $j$ th boundary edge.  $\text{EDGE}(3, j)$  is a reference number for the  $j$ th boundary edge and  

$$\text{EDGE}(3, j) = \text{LINED}(4, i), \text{ where } i \text{ and } j \text{ are such that the } j\text{th edges is part of the } i\text{th line of the boundary and } \text{LINED}(4, i) \geq 0;$$

$$\text{EDGE}(3, j) = 100 + |\text{LINED}(4, i)|, \text{ where } i \text{ and } j \text{ are such that the } j\text{th edges is part of the } i\text{th line of the boundary and } \text{LINED}(4, i) < 0.$$
- 17: ITRACE – INTEGER *Input*  
*On entry:* the level of trace information required from D06BAF.  
 $\text{ITRACE} = 0$  or  $\text{ITRACE} < -1$   
 No output is generated.

ITRACE = 1

Output from the boundary mesh generator is printed on the current advisory message unit (see X04ABF). This output contains the input information of each line and each connected component of the boundary.

ITRACE = -1

An analysis of the output boundary mesh is printed on the current advisory message unit. This analysis includes the orientation (clockwise or anticlockwise) of each connected component of the boundary. This information could be of interest to you, especially if an interior meshing is carried out using the output of this routine, calling either D06AAF, D06ABF or D06ACF.

ITRACE > 1

The output is similar to that produced when ITRACE = 1, but the coordinates of the generated vertices on the boundary are also output.

You are advised to set ITRACE = 0, unless you are experienced with finite element mesh generation.

18: RUSER(\*) – REAL (KIND=nag\_wp) array *User Workspace*

19: IUSER(\*) – INTEGER array *User Workspace*

RUSER and IUSER are not used by D06BAF, but are passed directly to FBND and should be used to pass information to this routine.

20: RWORK(LRWORK) – REAL (KIND=nag\_wp) array *Workspace*

21: LRWORK – INTEGER *Input*

*On entry:* the dimension of the array RWORK as declared in the (sub)program from which D06BAF is called.

*C o n s t r a i n t :*

$LRWORK \geq 2 \times (NLINES + SDCRUS) + 2 \times \max_{i=1,2,\dots,m} \{LINED(1, i)\} \times NLINES.$

22: IWORK(LIWORK) – INTEGER array *Workspace*

23: LIWORK – INTEGER *Input*

*On entry:* the dimension of the array IWORK as declared in the (sub)program from which D06BAF is called.

*Constraint:*

$LIWORK \geq \sum_{\{i, LINED(4,i) < 0\}} \{LINED(1, i) - 2\} + 8 \times NLINES + NVMAX + 3 \times NEDMX + 2 \times SDCRUS.$

24: 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:

$IFAIL = 1$

- On entry,  $NLINES < 1$ ;
- or  $NVMAX < NLINES$ ;
- or  $NEDMX < 1$ ;
- or  $NCOMP < 1$ ;
- or  $LRWORK < 2 \times (NLINES + SDCRUS) + 2 \times \max_{i=1,2,\dots,m} \{LINED(1, i)\} \times NLINES$ ;
- or  $LIWORK < \sum_{\{i, LINED(4,i) < 0\}} \{LINED(1, i) - 2\} + 8 \times NLINES + NVMAX + 3 \times NEDMX + 2 \times SDCRUS$ ;
- or  $SDCRUS < \sum_{\{i, LINED(4,i) < 0\}} \{LINED(1, i) - 2\}$ ;
- or  $RATE(i) < 0.0$  for some  $i = 1, 2, \dots, NLINES$  with  $LINED(4, i) \geq 0$ ;
- or  $LINED(1, i) < 2$  for some  $i = 1, 2, \dots, NLINES$ ;
- or  $LINED(2, i) < 1$  or  $LINED(2, i) > NLINES$  for some  $i = 1, 2, \dots, NLINES$ ;
- or  $LINED(3, i) < 1$  or  $LINED(3, i) > NLINES$  for some  $i = 1, 2, \dots, NLINES$ ;
- or  $LINED(2, i) = LINED(3, i)$  for some  $i = 1, 2, \dots, NLINES$ ;
- or  $NLCOMP(k) = 0$ , or  $|NLCOMP(k)| > NLINES$  for a  $k = 1, 2, \dots, NCOMP$ ;
- or  $\sum_{k=1}^n |NLCOMP(k)| \neq NLINES$ ;
- or  $LCOMP$  does not represent a valid permutation of the integers in  $[1, NLINES]$ ;
- or one of the end points for a line  $i$  described by the user-supplied function (lines with  $LINED(4, i) > 0$ , for  $i = 1, 2, \dots, NLINES$ ) does not belong to the corresponding curve in  $FBND$ ;
- or the intermediate points for the lines described as polygonal arcs (lines with  $LINED(4, i) < 0$ , for  $i = 1, 2, \dots, NLINES$ ) are overlapping.

$IFAIL = 2$

An error has occurred during the generation of the boundary mesh. It appears that  $NEDMX$  is not large enough, so you are advised to increase the value of  $NEDMX$ .

$IFAIL = 3$

An error has occurred during the generation of the boundary mesh. It appears that  $NVMAX$  is not large enough, so you are advised to increase the value of  $NVMAX$ .

$IFAIL = 4$

An error has occurred during the generation of the boundary mesh. Check the definition of each line (the argument  $LINED$ ) and each connected component of the boundary (the arguments  $NLCOMP$ , and  $LCOMP$ , as well as the coordinates of the characteristic points. Setting  $ITRACE > 0$  may provide more details.

$IFAIL = -99$

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

Not applicable.

## 8 Parallelism and Performance

D06BAF is not threaded in any implementation.

## 9 Further Comments

The boundary mesh generation technique in this routine has a ‘tree’ structure. The boundary should be partitioned into geometrically simple segments (straight lines or curves) delimited by characteristic points. Then, the lines should be assembled into connected components of the boundary domain.

Using this strategy, the inputs to that routine can be built up, following the requirements stated in Section 5:

the characteristic and the user-supplied intermediate points:

NLINES, SDCRUS, COORCH and CRUS;

the characteristic lines:

LINED, FBND, RATE;

finally the assembly of lines into the connected components of the boundary:

NCOMP, and

NLCOMP, LCOMP.

The example below details the use of this strategy.

## 10 Example

The NAG logo is taken as an example of a geometry with holes. The boundary has been partitioned in 40 lines characteristic points; including 4 for the exterior boundary and 36 for the logo itself. All line geometry specifications have been considered, see the description of LINED, including 4 lines defined as polygonal arc, 4 defined by FBND and all the others are straight lines.

### 10.1 Program Text

```
! D06BAF Example Program Text
! Mark 26 Release. NAG Copyright 2016.
! Module d06baf_mod

! D06BAF 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 :: fbnd
! .. Parameters ..
! Integer, Parameter, Public :: nin = 5, nout = 6
! Contains
! Function fbnd(i,x,y,ruser,iuser)
```

```

! .. Function Return Value ..
Real (Kind=nag_wp)          :: fbnd
! .. Scalar Arguments ..
Real (Kind=nag_wp), Intent (In) :: x, y
Integer, Intent (In)          :: i
! .. Array Arguments ..
Real (Kind=nag_wp), Intent (Inout) :: ruser(*)
Integer, Intent (Inout)          :: iuser(*)
! .. Local Scalars ..
Real (Kind=nag_wp)          :: radius2, x0, xa, xb, y0
! .. Executable Statements ..
xa = ruser(1)
xb = ruser(2)
x0 = ruser(3)
y0 = ruser(4)

fbnd = 0.0_nag_wp

Select Case (i)
Case (1)

!     line 1,2,3, and 4: ellipse centred in (X0,Y0) with
!     XA and XB as coefficients

    fbnd = ((x-x0)/xa)**2 + ((y-y0)/xb)**2 - 1.0_nag_wp
Case (2)

!     line 7, lower arc on letter n, is a circle centred in (X0,Y0)
!     with radius SQRT(RADIUS2)

    x0 = 0.5_nag_wp
    y0 = 6.25_nag_wp
    radius2 = 20.3125_nag_wp
    fbnd = (x-x0)**2 + (y-y0)**2 - radius2
Case (3)

!     line 11, upper arc on letter n, is a circle centred in (X0,Y0)
!     with radius SQRT(RADIUS2)

    x0 = 1.0_nag_wp
    y0 = 4.0_nag_wp
    radius2 = 9.0_nag_wp + (11.0_nag_wp-y0)**2
    fbnd = (x-x0)**2 + (y-y0)**2 - radius2
Case (4)

!     line 15, upper arc on letter a, is a circle centred in (X0,Y0)
!     with radius SQRT(RADIUS2) touching point (5,11).

    x0 = 8.5_nag_wp
    y0 = 2.75_nag_wp
    radius2 = (x0-5.0_nag_wp)**2 + (11.0_nag_wp-y0)**2
    fbnd = (x-x0)**2 + (y-y0)**2 - radius2
Case (5)

!     line 25, lower arc on hat of 'a', is a circle centred in (X0,Y0)
!     with radius SQRT(RADIUS2) touching point (11,10).

    x0 = 8.5_nag_wp
    y0 = 4.0_nag_wp
    radius2 = 2.5_nag_wp**2 + (10.0_nag_wp-y0)**2
    fbnd = (x-x0)**2 + (y-y0)**2 - radius2
Case (6)

!     lines 20, 21 and 22, belly of letter a, is an ellipse centered
!     in (X0, Y0) with semi-axes 3.5 and 2.75.

    x0 = 8.5_nag_wp
    y0 = 5.75_nag_wp
    fbnd = ((x-x0)/3.5_nag_wp)**2 + ((y-y0)/2.75_nag_wp)**2 - 1.0_nag_wp
Case (7)

```



```

!      lines 43, 44 and 45, outer curve on bottom of 'g', is an ellipse
!      centered in (X0, Y0) with semi-axes 3.5 and 2.5.

      x0 = 17.5_nag_wp
      y0 = 2.5_nag_wp
      fbnd = ((x-x0)/3.5_nag_wp)**2 + ((y-y0)/2.5_nag_wp)**2 - 1.0_nag_wp
Case (8)

!      lines 28, 29 and 30, inner curve on bottom of 'g', is an ellipse
!      centered in (X0, Y0) with semi-axes 2.0 and 1.5.

      x0 = 17.5_nag_wp
      y0 = 2.5_nag_wp
      fbnd = ((x-x0)/2.0_nag_wp)**2 + ((y-y0)/1.5_nag_wp)**2 - 1.0_nag_wp
Case (9)

!      line 42, inner curve on lower middle of 'g', is an ellipse
!      centered in (X0, Y0) with semi-axes 1.5 and 0.5.

      x0 = 17.5_nag_wp
      y0 = 5.5_nag_wp
      fbnd = ((x-x0)/1.5_nag_wp)**2 + ((y-y0)/0.5_nag_wp)**2 - 1.0_nag_wp
Case (10)

!      line 31, outer curve on lower middle of 'g', is an ellipse
!      centered in (X0, Y0) with semi-axes 2.0 and 1.5.

      x0 = 17.5_nag_wp
      y0 = 5.5_nag_wp
      fbnd = ((x-x0)/3.0_nag_wp)**2 + ((y-y0)/1.5_nag_wp)**2 - 1.0_nag_wp
Case (11)

!      line 41, inner curve on upper middle of 'g', is an ellipse
!      centered in (X0, Y0) with semi-axes 1.0 and 1.0.

      x0 = 17.0_nag_wp
      y0 = 5.5_nag_wp
      fbnd = ((x-x0)/1.0_nag_wp)**2 + ((y-y0)/1.0_nag_wp)**2 - 1.0_nag_wp
Case (12)

!      line 32, outer curve on upper middle of 'g', is an ellipse
!      centered in (X0, Y0) with semi-axes 1.5 and 1.1573.

      x0 = 16.0_nag_wp
      y0 = 5.5_nag_wp
      fbnd = ((x-x0)/1.5_nag_wp)**2 + ((y-y0)/1.1573_nag_wp)**2 -      &
      1.0_nag_wp
Case (13)

!      lines 33, 33, 34, 39 and 40, upper portion of 'g', is an ellipse
!      centered in (X0, Y0) with semi-axes 3.0 and 2.75.
      x0 = 17.0_nag_wp
      y0 = 9.25_nag_wp
      fbnd = ((x-x0)/3.0_nag_wp)**2 + ((y-y0)/2.75_nag_wp)**2 - 1.0_nag_wp
End Select

      Return

      End Function fbnd
End Module d06baf_mod
Program d06baf

!      D06BAF Example Main Program

!      .. Use Statements ..
      Use nag_library, Only: d06abf, d06acf, d06baf, f16dnf, nag_wp
      Use d06baf_mod, Only: fbnd, nin, nout
!      .. Implicit None Statement ..
      Implicit None
!      .. Local Scalars ..

```

```

Real (Kind=nag_wp)           :: x0, xa, xb, xmax, xmin, y0, ymax,    &
                             ymin
Integer                       :: i, ifail, itrace, j, k, liwork,    &
                             lrwork, maxind, maxval, ncomp,      &
                             nedge, nedmx, nelt, nlines, npropa, &
                             nv, nvb, nvint, nvmax, reftk, sdcrus
Character (1)                 :: pmesh
! .. Local Arrays ..
Real (Kind=nag_wp), Allocatable :: coor(:,,:), coorch(:,,:), crus(:,,:),    &
                             rate(:), rwork(:), weight(:)
Real (Kind=nag_wp)           :: ruser(4)
Integer, Allocatable         :: conn(:,,:), edge(:,,:), iwork(:),    &
                             lcomp(:), lined(:,,:), nlcomp(:)
Integer                       :: iuser(1)
! .. Intrinsic Procedures ..
Intrinsic                    :: abs
! .. Executable Statements ..
Write (nout,*) 'D06BAF Example Program Results'
Flush (nout)

! Skip heading in data file
Read (nin,*)

! Initialize boundary mesh inputs:
! the number of line and of the characteristic points of
! the boundary mesh

Read (nin,*) nlines, nvmax, nedmx
Allocate (coor(2,nvmax),coorch(2,nlines),rate(nlines),edge(3,nedmx),    &
         lcomp(nlines),lined(4,nlines))

! The Lines of the boundary mesh

Read (nin,*)(lined(1:4,j),rate(j),j=1,nlines)

sdcrus = 0

Do i = 1, nlines
  If (lined(4,i)<0) Then
    sdcrus = sdcrus + lined(1,i) - 2
  End If
End Do

liwork = 8*nlines + nvmax + 3*nedmx + 3*sdcrus

! Get max(LINED(1,:)) for computing LRWORK
Call f16dnf(nlines,lined,4,maxind,maxval)

lrwork = 2*(nlines+sdcrus) + 2*maxval*nlines

Allocate (crus(2,sdcrus),iwork(liwork),rwork(lrwork))

! The ellipse boundary which envelops the NAG Logo
! the N, the A and the G

Read (nin,*) coorch(1,1:nlines)
Read (nin,*) coorch(2,1:nlines)

Read (nin,*) crus(1,1:sdcrus)
Read (nin,*) crus(2,1:sdcrus)

! The number of connected components to the boundary
! and their information

Read (nin,*) ncomp
Allocate (nlcomp(ncomp))

j = 1

```

```

Do i = 1, ncomp
  Read (nin,*) nlcomp(i)
  k = j + abs(nlcomp(i)) - 1
  Read (nin,*) lcomp(j:k)
  j = k + 1
End Do

! Data passed to the user-supplied function

xmin = coorch(1,4)
xmax = coorch(1,2)
ymin = coorch(2,1)
ymax = coorch(2,3)

xa = (xmax-xmin)/2.0_nag_wp
xb = (ymax-ymin)/2.0_nag_wp

x0 = (xmin+xmax)/2.0_nag_wp
y0 = (ymin+ymax)/2.0_nag_wp

ruser(1:4) = (/xa,xb,x0,y0/)
iuser(1) = 0

itrace = -1

Write (nout,*)
Flush (nout)

! Call to the boundary mesh generator

ifail = 0
Call d06baf(nlines,coorch,lined,fbnd,crus,sdcrus,rate,ncomp,nlcomp,      &
  lcomp,nvmax,nedmx,nvb,coor,nedge,edge,itrace,ruser,iuser,rwork,lrwork, &
  iwork,liwork,ifail)

Read (nin,*) pmesh

Select Case (pmesh)
Case ('N')
  Write (nout,*) 'Boundary mesh characteristics'
  Write (nout,99999) 'NVB =', nvb
  Write (nout,99999) 'NEDGE =', nedge
Case ('Y')

! Output the mesh

Write (nout,99998) nvb, nedge

Do i = 1, nvb
  Write (nout,99997) i, coor(1:2,i)
End Do

Do i = 1, nedge
  Write (nout,99996) i, edge(1:3,i)
End Do

Flush (nout)

Case Default
  Write (nout,*) 'Problem with the printing option Y or N'
  Go To 100
End Select

Deallocate (rwork,iwork)

! Initialize mesh control parameters

itrace = 0
npropa = 1
nvint = 0

```

```

lrwork = 12*nvmax + 15
liwork = 6*nedge + 32*nvmax + 2*nvb + 78
Allocate (weight(nvint),rwork(lrwork),iwork(liwork),conn(3,2*nvmax+5))

! Call to the 2D Delaunay-Voronoi mesh generator

ifail = 0
Call d06abf(nvb,nvint,nvmax,nedge,edge,nv,nelt,coor,conn,weight,npropa, &
  itrace,rwork,lrwork,iwork,liwork,ifail)

Select Case (pmesh)
Case ('N')
  Write (nout,*) 'Complete mesh (via the 2D Delaunay-Voronoi'
  Write (nout,*) 'mesh generator) characteristics'
  Write (nout,99999) 'NV (rounded to nearest 10) =', 10*((nv+5)/10)
  Write (nout,99999) 'NELT (rounded to nearest 10) =', 10*((nelt+5)/10)
Case ('Y')

! Output the mesh

Write (nout,99998) nv, nelt

Do i = 1, nv
  Write (nout,99995) coor(1:2,i)
End Do

reftk = 0

Do k = 1, nelt
  Write (nout,99994) conn(1:3,k), reftk
End Do

Flush (nout)

End Select

Deallocate (rwork,iwork)
lrwork = 12*nvmax + 30015
liwork = 8*nedge + 53*nvmax + 2*nvb + 10078
Allocate (rwork(lrwork),iwork(liwork))

! Call to the 2D Advancing front mesh generator

ifail = 0
Call d06acf(nvb,nvint,nvmax,nedge,edge,nv,nelt,coor,conn,weight,itrace, &
  rwork,lrwork,iwork,liwork,ifail)

Select Case (pmesh)
Case ('N')
  Write (nout,*) 'Complete mesh (via the 2D Advancing front mesh'
  Write (nout,*) 'generator) characteristics'
  Write (nout,99999) 'NV (rounded to nearest 10) =', 10*((nv+5)/10)
  Write (nout,99999) 'NELT (rounded to nearest 10) =', 10*((nelt+5)/10)
Case ('Y')

! Output the mesh

Write (nout,99998) nv, nelt

Do i = 1, nv
  Write (nout,99995) coor(1:2,i)
End Do

reftk = 0

Do k = 1, nelt
  Write (nout,99994) conn(1:3,k), reftk
End Do

End Select

```

```

100  Continue

99999 Format (1X,A,I6)
99998 Format (1X,2I10)
99997 Format (2X,I4,2(2X,E13.6))
99996 Format (1X,4I4)
99995 Format (2(2X,E13.6))
99994 Format (1X,4I10)
      End Program d06baf
    
```

**10.2 Program Data**

```

D06BAF Example Program Data
45  5000  1000
15  1  2  1  0.9500  15  2  3  1  1.0500      :NLINEs (m), NVMAX, NEDMX
15  3  4  1  0.9500  15  4  1  1  1.0500
  4  6  5 -1  1.0000  10 10  6  0  1.0000
10 14 10  2  1.0000  10  7 14  0  1.0000
  4  8  7  0  1.0000  10 13  8  0  1.0000
10 13  9  3  1.0000  10 12  9  0  1.0000
  4 11 12  0  1.0000  15  5 11  0  1.0000
15 26 15  4  1.0000  10 26 25  0  1.0000
  4 25 24  0  1.0000   4 24 23  0  1.0000
  4 23 22  0  1.0000  10 21 22  6  1.0000
10 20 21  6  1.0000  10 19 20  6  1.0000
  4 19 18  0  1.0000   5 18 17  0  1.0000
15 17 16  5  1.0000   4 16 15  0  1.0000
  4 27 28  0  1.0000   7 28 30  8  1.0000
  7 30 32  8  1.0000   7 32 34  8  1.0000
  6 36 34 10  1.0000   6 38 36 12  1.0000
10 40 38 13  1.0000  10 42 40 13  1.0000
  8 44 42 13  1.0000   4 44 45  0  1.0000
  4 45 43  0  1.0000   4 43 41  0  1.0000
  6 39 41 13  1.0000  10 37 39 13  1.0000
  6 37 35 11  1.0000   6 35 33  9  1.0000
10 31 33  7  1.0000  10 29 31  7  1.0000
10 27 29  7  1.0000      : (LINE(:,j),RATE(j),j=1,m)
  9.5000 33.0000  9.5000 -14.0000
 -4.0000 -2.0000  2.0000  4.0000 -2.0000 -2.0000 -4.0000
 -2.0000  4.0000  2.0000
  5.0000  6.0000 11.0000 11.0000  8.5000  5.0000  8.5000
11.5000 13.0000 14.0000 13.0000 13.0000
14.0000 15.5000 17.5000 17.5000 21.0000 19.5000 17.5000
17.5000 16.0000 14.5000 17.0000 16.0000 20.0000 14.0000
19.3142 17.0000 20.5000 18.7249 19.5000      : End of X coords
 -3.0000  6.5000 16.0000  6.5000
  3.0000  3.0000  3.0000  3.0000 11.0000 10.0000 11.5000
12.0000 11.0000 10.5000
11.0000 10.0000 10.0000  8.5000  8.5000  5.7500  3.0000
  4.3333  3.0000  3.7500  4.7500 10.5000
  2.5000  2.5000  0.0000  1.0000  2.5000  2.5000  5.0000
  4.0000  5.5000  5.5000  6.5000  6.6573  9.2500  9.2500
11.0000 12.000  11.5000 11.5000 12.0000      : End of Y coords
 -2.6667 -3.3333  3.3333  2.6667      : (Poly (X))
  3.0000  3.0000  3.0000  3.0000      : (Poly (Y))
  4      :NCOMP (n, number of contours)
  4      :number of lines in contour 1
  1  2  3  4      :lines of contour 1 (Ellipse)
10      :number of lines in contour 2
14 13 12 11 10  9  8  7  6  5      :lines of contour 2 (Letter N)
12      :number of lines in contour 3
18 19 20 21 22 23 24 25 26 15 16 17      :lines of contour 3 (Letter A)
19      :number of lines in contour 4
27 28 29 30 31 32 33 34 35 36 37 38
39 40 41 42 43 44 45      :lines of contour 4 (Letter G)
'N'      :Printing option 'Y' or 'N'
    
```

### 10.3 Program Results

#### D06BAF Example Program Results

Analysis of the boundary created:

The boundary mesh contains 332 vertices and 332 edges

There are 4 components comprising the boundary:

The 1-st component contains 4 lines in anticlockwise orientation

The 2-nd component contains 10 lines in clockwise orientation

The 3-rd component contains 12 lines in anticlockwise orientation

The 4-th component contains 19 lines in clockwise orientation

Boundary mesh characteristics

NVB = 332

NEDGE = 332

Complete mesh (via the 2D Delaunay-Voronoi

mesh generator) characteristics

NV (rounded to nearest 10) = 900

NELT (rounded to nearest 10) = 1480

Complete mesh (via the 2D Advancing front mesh

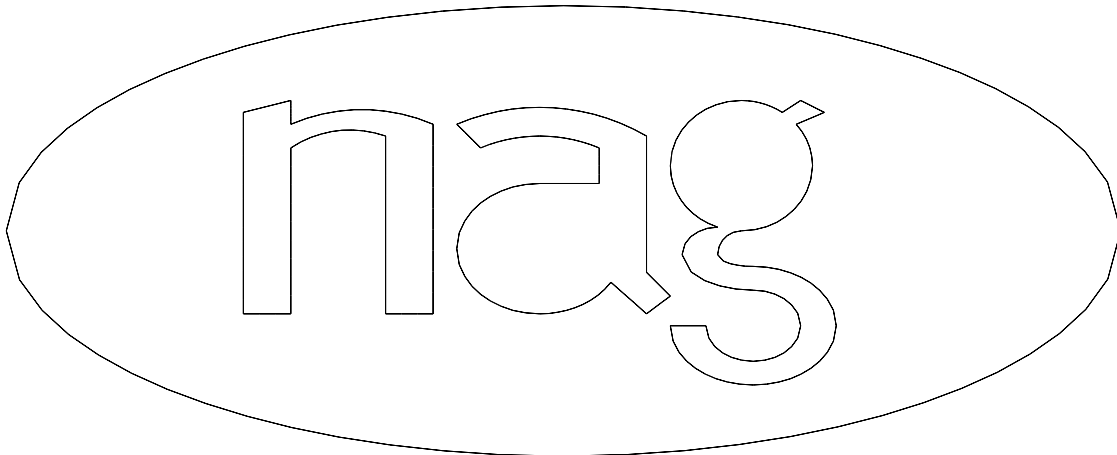
generator) characteristics

NV (rounded to nearest 10) = 920

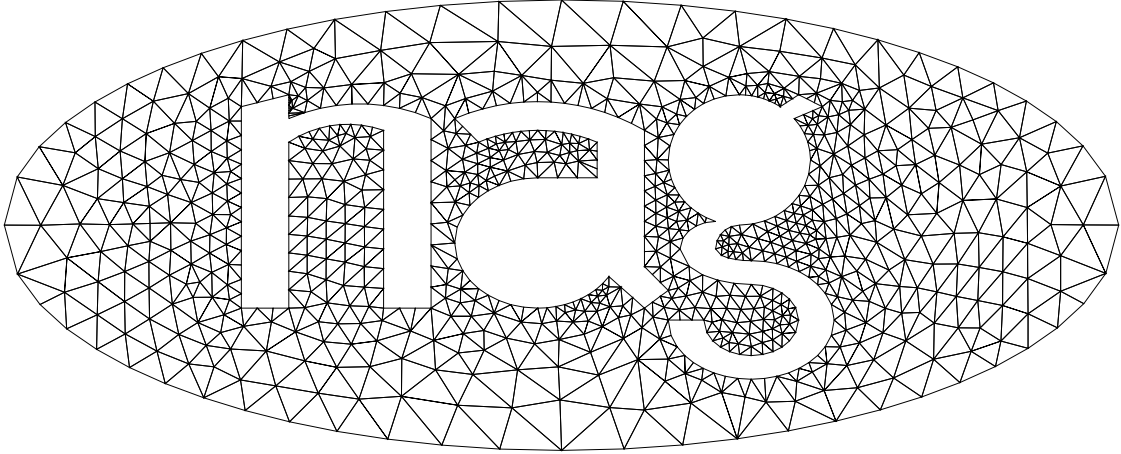
NELT (rounded to nearest 10) = 1520

#### Example Program

Boundary Mesh of the NAG Logo with 259 Nodes and 259 Edges



Final Mesh Built Using the Delaunay-Voronoi Method



Final Mesh Built Using the Advancing Front Method

