NAG Library Routine Document

D06ACF

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

D06ACF generates a triangular mesh of a closed polygonal region in $\mathbb{R}^2$, given a mesh of its boundary. It uses an Advancing Front process, based on an incremental method.

2 Specification

```
SUBROUTINE D06ACF (NVB, NVINT, NVMAX, NEDGE, EDGE, NV, NELT, COOR, CONN,    &
                   WEIGHT, ITRACE, RWORK, LRWORK, IWORK, LIWORK, IFAIL)

INTEGER            NVB, NVINT, NVMAX, NEDGE, EDGE(3,NEDGE), NV, NELT,       &
                   CONN(3,2*NVMAX+5), ITRACE, LRWORK, IWORK(LIWORK),        &
                   LIWORK, IFAIL
REAL (KIND=nag_wp) COOR(2,NVMAX), WEIGHT(*), RWORK(LRWORK)
```

3 Description

D06ACF generates the set of interior vertices using an Advancing Front process, based on an incremental method. It allows you to specify a number of fixed interior mesh vertices together with weights which allow concentration of the mesh in their neighbourhood. For more details about the triangulation method, 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: NVB – INTEGER *Input*

On entry: the number of vertices in the input boundary mesh.

Constraint: $\mathrm{NVB} \geq 3$.

2: NVINT – INTEGER *Input*

On entry: the number of fixed interior mesh vertices to which a weight will be applied.

Constraint: $\mathrm{NVINT} \geq 0$.

3: NVMAX – INTEGER *Input*

On entry: the maximum number of vertices in the mesh to be generated.

Constraint: $\mathrm{NVMAX} \geq \mathrm{NVB} + \mathrm{NVINT}$.

4: NEDGE – INTEGER *Input*

On entry: the number of boundary edges in the input mesh.

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

5: EDGE$(3,\text{NEDGE})$ – INTEGER array *Input*

On entry: the specification of the boundary edges. EDGE$(1,j)$ and EDGE$(2,j)$ contain the vertex numbers of the two end points of the jth boundary edge. EDGE$(3,j)$ is a user-supplied tag for the jth boundary edge and is not used by D06ACF.

Constraint: $1 \leq \text{EDGE}(i,j) \leq \text{NVB}$ and $\text{EDGE}(1,j) \neq \text{EDGE}(2,j)$, for $i = 1,2$ and $j = 1,2,\ldots,\text{NEDGE}$.

6: NV – INTEGER *Output*

On exit: the total number of vertices in the output mesh (including both boundary and interior vertices). If $\text{NVB} + \text{NVINT} = \text{NVMAX}$, no interior vertices will be generated and $\text{NV} = \text{NVMAX}$.

7: NELT – INTEGER *Output*

On exit: the number of triangular elements in the mesh.

8: COOR$(2,\text{NVMAX})$ – REAL (KIND=nag_wp) array *Input/Output*

On entry: COOR$(1,i)$ contains the x coordinate of the ith input boundary mesh vertex, for $i = 1,2,\ldots,\text{NVB}$. COOR$(1,i)$ contains the x coordinate of the $(i - \text{NVB})$th fixed interior vertex, for $i = \text{NVB} + 1,\ldots,\text{NVB} + \text{NVINT}$. For boundary and interior vertices, COOR$(2,i)$ contains the corresponding y coordinate, for $i = 1,2,\ldots,\text{NVB} + \text{NVINT}$.

On exit: COOR$(1,i)$ will contain the x coordinate of the $(i - \text{NVB} - \text{NVINT})$th generated interior mesh vertex, for $i = \text{NVB} + \text{NVINT} + 1,\ldots,\text{NV}$; while COOR$(2,i)$ will contain the corresponding y coordinate. The remaining elements are unchanged.

9: CONN$(3, 2 \times \text{NVMAX} + 5)$ – INTEGER array *Output*

On exit: the connectivity of the mesh between triangles and vertices. For each triangle j, CONN(i,j) gives the indices of its three vertices (in anticlockwise order), for $i = 1,2,3$ and $j = 1,2,\ldots,\text{NELT}$.

10: WEIGHT$(*)$ – REAL (KIND=nag_wp) array *Input*

Note: the dimension of the array WEIGHT must be at least $\max(1,\text{NVINT})$.

On entry: the weight of fixed interior vertices. It is the diameter of triangles (length of the longer edge) created around each of the given interior vertices.

Constraint: if $\text{NVINT} > 0$, $\text{WEIGHT}(i) > 0.0$, for $i = 1,2,\ldots,\text{NVINT}$.

11: ITRACE – INTEGER *Input*

On entry: the level of trace information required from D06ACF.

$\text{ITRACE} \leq 0$
No output is generated.

$\text{ITRACE} \geq 1$
Output from the meshing solver is printed on the current advisory message unit (see X04ABF). This output contains details of the vertices and triangles generated by the process.

You are advised to set $\text{ITRACE} = 0$, unless you are experienced with finite element mesh generation.

12: RWORK(LRWORK) – REAL (KIND=nag_wp) array *Workspace*

13: LRWORK – INTEGER *Input*

On entry: the dimension of the array RWORK as declared in the (sub)program from which D06ACF is called.

Constraint: LRWORK $\geq 12 \times \text{NVMAX} + 30015$.

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

15: LIWORK – INTEGER *Input*

On entry: the dimension of the array IWORK as declared in the (sub)program from which D06ACF is called.

Constraint: LIWORK $\geq 8 \times \text{NEDGE} + 53 \times \text{NVMAX} + 2 \times \text{NVB} + 10078$.

16: 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, NVB < 3,
or NVINT < 0,
or NVB + NVINT $>$ NVMAX,
or NEDGE < 1,
or EDGE$(i,j) < 1$ or EDGE$(i,j) >$ NVB, for some $i = 1,2$ and $j = 1,2,\ldots,\text{NEDGE}$,
or EDGE$(1,j) =$ EDGE$(2,j)$, for some $j = 1,2,\ldots,\text{NEDGE}$,
or if NVINT > 0, WEIGHT$(i) \leq 0.0$, for some $i = 1,2,\ldots,\text{NVINT}$;
or LRWORK $< 12 \times \text{NVMAX} + 30015$,
or LIWORK $< 8 \times \text{NEDGE} + 53 \times \text{NVMAX} + 2 \times \text{NVB} + 10078$.

IFAIL $= 2$

An error has occurred during the generation of the interior mesh. Check the definition of the boundary (arguments COOR and EDGE) as well as the orientation of the boundary (especially in the case of a multiple connected component boundary). 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

D06ACF makes calls to BLAS and/or LAPACK routines, which may be threaded within the vendor library used by this implementation. Consult the documentation for the vendor library for further information.

Please consult the X06 Chapter Introduction for information on how to control and interrogate the OpenMP environment used within this routine. Please also consult the Users' Note for your implementation for any additional implementation-specific information.

9 Further Comments

The position of the internal vertices is a function position of the vertices on the given boundary. A fine mesh on the boundary results in a fine mesh in the interior. During the process vertices are generated on edges of the mesh $\mathcal{T}_i$ to obtain the mesh $\mathcal{T}_{i+1}$ in the general incremental method (consult the D06 Chapter Introduction or George and Borouchaki (1998)).

You are advised to take care to set the boundary inputs properly, especially for a boundary with multiply connected components. The orientation of the interior boundaries should be in **clockwise** order and opposite to that of the exterior boundary. If the boundary has only one connected component, its orientation should be **anticlockwise**.

10 Example

In this example, a geometry with two holes (two wings inside an exterior circle) is meshed using a Delaunay–Voronoi method. The exterior circle is centred at the point $(1.5, 0.0)$ with a radius 4.5, the first wing begins at the origin and it is normalized, finally the last wing is also normalized and begins at the point $(0.8, -0.3)$. To be able to carry out some realistic computation on that geometry, some interior points have been introduced to have a finer mesh in the wake of those airfoils.

The boundary mesh has 120 vertices and 120 edges (see Figure 1 top). Note that the particular mesh generated could be sensitive to the ***machine precision*** and therefore may differ from one implementation to another.

10.1 Program Text

```
!   D06ACF Example Program Text
!   Mark 26 Release. NAG Copyright 2016.
    Module d06acfe_mod

!     D06ACF 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)             :: c, radius, x0, x1, y0, y1
!       .. Intrinsic Procedures ..
        Intrinsic                      :: sqrt
!       .. Executable Statements ..
        fbnd = 0.0_nag_wp

        Select Case (i)
        Case (1)

!         upper NACA0012 wing beginning at the origin

          c = 1.008930411365_nag_wp
          fbnd = 0.6_nag_wp*(0.2969_nag_wp*sqrt(c*x)-0.126_nag_wp*c*x-         &
            0.3516_nag_wp*(c*x)**2+0.2843_nag_wp*(c*x)**3-                     &
            0.1015_nag_wp*(c*x)**4) - c*y
        Case (2)

!         lower NACA0012 wing beginning at the origin

          c = 1.008930411365_nag_wp
          fbnd = 0.6_nag_wp*(0.2969_nag_wp*sqrt(c*x)-0.126_nag_wp*c*x-         &
            0.3516_nag_wp*(c*x)**2+0.2843_nag_wp*(c*x)**3-                     &
            0.1015_nag_wp*(c*x)**4) + c*y
        Case (3)
          x0 = ruser(1)
          y0 = ruser(2)
          radius = ruser(3)
          fbnd = (x-x0)**2 + (y-y0)**2 - radius**2
        Case (4)

!         upper NACA0012 wing beginning at (X1;Y1)

          c = 1.008930411365_nag_wp
          x1 = ruser(4)
          y1 = ruser(5)
          fbnd = 0.6_nag_wp*(0.2969_nag_wp*sqrt(c*(x-                          &
            x1))-0.126_nag_wp*c*(x-x1)-0.3516_nag_wp*(c*(x-                    &
            x1))**2+0.2843_nag_wp*(c*(x-x1))**3-0.1015_nag_wp*(c*(x-x1))**4) - &
            c*(y-y1)
        Case (5)

!         lower NACA0012 wing beginning at (X1;Y1)

          c = 1.008930411365_nag_wp
          x1 = ruser(4)
          y1 = ruser(5)
          fbnd = 0.6_nag_wp*(0.2969_nag_wp*sqrt(c*(x-                          &
            x1))-0.126_nag_wp*c*(x-x1)-0.3516_nag_wp*(c*(x-                    &
            x1))**2+0.2843_nag_wp*(c*(x-x1))**3-0.1015_nag_wp*(c*(x-x1))**4) + &
            c*(y-y1)
        End Select

        Return
      End Function fbnd
    End Module d06acfe_mod
    Program d06acfe

!     D06ACF Example Main Program

!     .. Use Statements ..
      Use nag_library, Only: d06acf, d06baf, f16dnf, nag_wp
      Use d06acfe_mod, Only: fbnd, nin, nout
!     .. Implicit None Statement ..
      Implicit None
!     .. Local Scalars ..
      Real (Kind=nag_wp)                :: dnvint, radius, x0, x1, y0, y1
      Integer                           :: i, ifail, itrace, j, k, liwork,      &
                                           lrwork, maxind, maxval, ncomp,       &
                                           nedge, nedmx, nelt, nlines, nv, nvb, &
                                           nvint, nvint2, nvmax, reftk, sdcrus
      Character (1)                     :: pmesh
!     .. Local Arrays ..
      Real (Kind=nag_wp), Allocatable  :: coor(:,:), coorch(:,:), crus(:,:),   &
                                           rate(:), rwork(:), weight(:)
      Real (Kind=nag_wp)                :: ruser(5)
      Integer, Allocatable              :: conn(:,:), edge(:,:), iwork(:),      &
                                           lcomp(:), lined(:,:), nlcomp(:)
      Integer                           :: iuser(1)
!     .. Intrinsic Procedures ..
      Intrinsic                         :: abs, real
!     .. Executable Statements ..
      Write (nout,*) 'D06ACF Example Program Results'

!     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))

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

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

      Read (nin,*) ncomp
      Allocate (crus(2,sdcrus),nlcomp(ncomp),iwork(liwork),rwork(lrwork))
      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

      x0 = 1.5_nag_wp
      y0 = 0.0_nag_wp
      radius = 4.5_nag_wp
      x1 = 0.8_nag_wp
      y1 = -0.3_nag_wp

      ruser(1:5) = (/x0,y0,radius,x1,y1/)
      iuser(1) = 0

      itrace = 0

!     Call to the 2D 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)

      Write (nout,*)
      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
      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

!     Generation of interior vertices
!     for the wake of the first NACA

      nvint = 40
      lrwork = 12*nvmax + 30015
      liwork = 8*nedge + 53*nvmax + 2*nvb + 10078
      Allocate (weight(nvint),rwork(lrwork),conn(3,2*nvmax+5),iwork(liwork))

      nvint2 = 20
      dnvint = 5.0_nag_wp/real(nvint2+1,kind=nag_wp)

      Do i = 1, nvint2
        reftk = nvb + i
        coor(1,reftk) = 1.0_nag_wp + real(i,kind=nag_wp)*dnvint
        coor(2,reftk) = 0.0_nag_wp
      End Do

      weight(1:nvint2) = 0.05_nag_wp

!     for the wake of the second one

      dnvint = 4.19_nag_wp/real(nvint2+1,kind=nag_wp)

      Do i = nvint2 + 1, nvint
        reftk = nvb + i
        coor(1,reftk) = 1.8_nag_wp + real(i-nvint2,kind=nag_wp)*dnvint
        coor(2,reftk) = -0.3_nag_wp
      End Do

      weight((nvint2+1):nvint) = 0.05_nag_wp

!     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 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 d06acfe
```

10.2 Program Data

```
D06ACF Example Program Data
 8  2000  200                                          :NLINES (m), NVMAX, NEDMX
  0.0000   1.0000  -3.0000   6.0000   0.8000
  1.8000   1.5000   1.5000                             :(COORCH(1,1:m))
  0.0000   0.0000   0.0000   0.0000  -0.3000
 -0.3000   4.5000  -4.5000                             :(COORCH(2,1:m))
21  2  1  1   1.0000 21  1  2  2   1.0000
```

```
11  3  8  3   1.0000 11  4  7  3   1.0000
21  6  5  4   1.0000 21  5  6  5   1.0000
11  7  3  3   1.0000 11  8  4  3   1.0000 :(LINE(:,j),RATE(j),j=1,m)
 3                                     :NCOMP (n, number of contours)
-2                                      :number of lines in contour 1
 1  2                                             :lines of contour 1
 4                                      :number of lines in contour 2
 3  8  4  7                                       :lines of contour 2
-2                                      :number of lines in contour 3
 5  6                                             :lines of contour 3
'N'                                       :Printing option 'Y' or 'N'
```

10.3 Program Results

```
 D06ACF Example Program Results

 Boundary mesh characteristics
 NVB   =   120
 NEDGE =   120
 Complete mesh characteristics
 NV (rounded to nearest 10)   =  1890
 NELT (rounded to nearest 10) =  3660
```

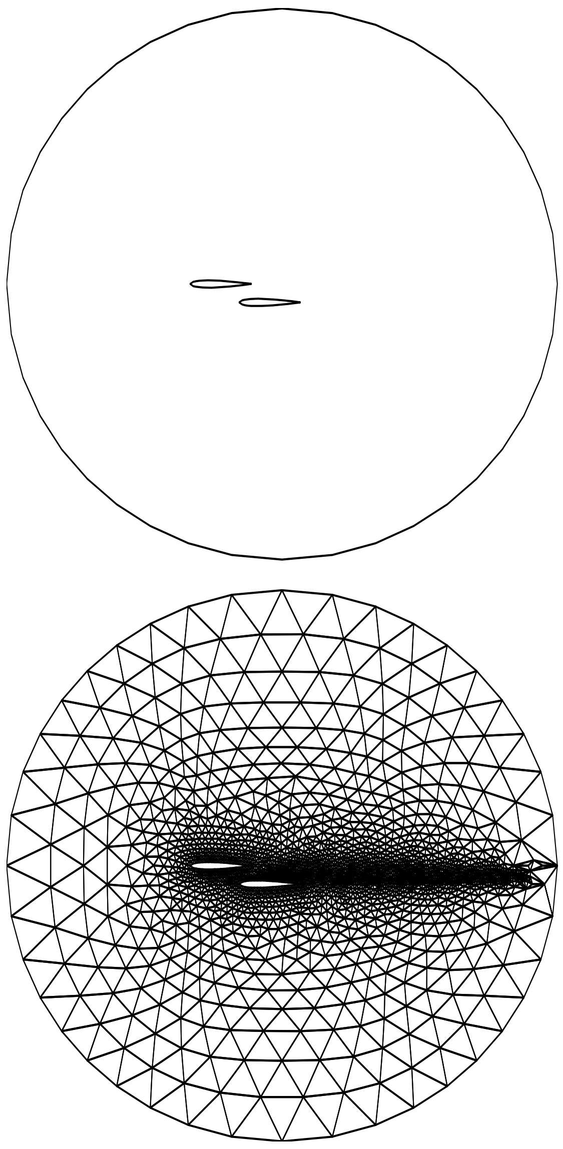

Figure 1
The boundary mesh (top), the interior mesh (bottom) of a double wing inside a circle geometry

---