# NAG Library Routine Document

# D06CAF

**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

D06CAF uses a barycentering technique to smooth a given mesh.

## 2 Specification

```
SUBROUTINE D06CAF (NV, NELT, NEDGE, COOR, EDGE, CONN, NVFIX, NUMFIX,           &
                   ITRACE, NQINT, IWORK, LIWORK, RWORK, LRWORK, IFAIL)

INTEGER            NV, NELT, NEDGE, EDGE(3,NEDGE), CONN(3,NELT), NVFIX,        &
                   NUMFIX(*), ITRACE, NQINT, IWORK(LIWORK), LIWORK,            &
                   LRWORK, IFAIL
REAL (KIND=nag_wp) COOR(2,NV), RWORK(LRWORK)
```

## 3 Description

D06CAF uses a barycentering approach to improve the smoothness of a given mesh. The measure of quality used for a triangle $K$ is

$$Q_K = \alpha \frac{h_K}{\rho_K};$$

where $h_K$ is the diameter (length of the longest edge) of $K$, $\rho_K$ is the radius of its inscribed circle and $\alpha = \frac{\sqrt{3}}{6}$ is a normalization factor chosen to give $Q_K = 1$ for an equilateral triangle. $Q_K$ ranges from 1, for an equilateral triangle, to $\infty$, for a totally flat triangle.

D06CAF makes small perturbation to vertices (using a barycenter formula) in order to give a reasonably good value of $Q_K$ for all neighbouring triangles. Some vertices may optionally be excluded from this process.

For more details about the smoothing method, especially with regard to differing quality, 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 Parameters

1: NV – INTEGER *Input*

*On entry*: the total number of vertices in the input mesh.

*Constraint*: NV $\geq$ 3.

2: NELT – INTEGER *Input*

*On entry*: the number of triangles in the input mesh.

*Constraint*: NELT $\leq 2 \times$ NV $- 1$.

3: NEDGE – INTEGER *Input*

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

*Constraint*: NEDGE $\geq 1$.

4: COOR(2,NV) – REAL (KIND=nag_wp) array *Input/Output*

*On entry*: $\mathrm{COOR}(1,i)$ contains the $x$ coordinate of the $i$th input mesh vertex, for $i = 1,2,\ldots,\mathrm{NV}$; while $\mathrm{COOR}(2,i)$ contains the corresponding $y$ coordinate.

*On exit*: $\mathrm{COOR}(1,i)$ will contain the $x$ coordinate of the $i$th smoothed mesh vertex, for $i = 1,2,\ldots,\mathrm{NV}$; while $\mathrm{COOR}(2,i)$ will contain the corresponding $y$ coordinate. Note that the coordinates of boundary and interface edge vertices, as well as those specified by you (see the description of NUMFIX), are unchanged by the process.

5: EDGE(3,NEDGE) – INTEGER array *Input*

*On entry*: the specification of the boundary or interface edges. $\mathrm{EDGE}(1,j)$ and $\mathrm{EDGE}(2,j)$ contain the vertex numbers of the two end points of the $j$th boundary edge. $\mathrm{EDGE}(3,j)$ is a user-supplied tag for the $j$th boundary or interface edge: $\mathrm{EDGE}(3,j) = 0$ for an interior edge and has a nonzero tag otherwise.

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

6: CONN(3,NELT) – INTEGER array *Input*

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

*Constraint*: $1 \leq \mathrm{CONN}(i,j) \leq \mathrm{NV}$ and $\mathrm{CONN}(1,j) \neq \mathrm{CONN}(2,j)$ and $\mathrm{CONN}(1,j) \neq \mathrm{CONN}(3,j)$ and $\mathrm{CONN}(2,j) \neq \mathrm{CONN}(3,j)$, for $i = 1,2,3$ and $j = 1,2,\ldots,\mathrm{NELT}$.

7: NVFIX – INTEGER *Input*

*On entry*: the number of fixed vertices in the input mesh.

*Constraint*: $0 \leq \mathrm{NVFIX} \leq \mathrm{NV}$.

8: NUMFIX($*$) – INTEGER array *Input*

**Note**: the dimension of the array NUMFIX must be at least $\max(1,\mathrm{NVFIX})$.

*On entry*: the indices in COOR of fixed interior vertices of the input mesh.

*Constraint*: if $\mathrm{NVFIX} > 0$, $1 \leq \mathrm{NUMFIX}(i) \leq \mathrm{NV}$, for $i = 1,2,\ldots,\mathrm{NVFIX}$.

9: ITRACE – INTEGER *Input*

*On entry*: the level of trace information required from D06CAF.

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

$\mathrm{ITRACE} = 1$
: A histogram of the triangular element qualities is printed on the current advisory message unit (see X04ABF) before and after smoothing. This histogram gives the lowest and the highest triangle quality as well as the number of elements lying in each of the NQINT equal intervals between the extremes.

$\mathrm{ITRACE} > 1$
: The output is similar to that produced when $\mathrm{ITRACE} = 1$ but the connectivity between vertices and triangles (for each vertex, the list of triangles in which it appears) is given.

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

10: NQINT – INTEGER *Input*

*On entry*: the number of intervals between the extreme quality values for the input and the smoothed mesh.

If ITRACE $= 0$, NQINT is not referenced.

11: IWORK(LIWORK) – INTEGER array *Workspace*
12: LIWORK – INTEGER *Input*

*On entry*: the dimension of the array IWORK as declared in the (sub)program from which D06CAF is called.

*Constraint*: LIWORK $\geq 8 \times$ NELT $+ 2 \times$ NV.

13: RWORK(LRWORK) – REAL (KIND=nag_wp) array *Workspace*
14: LRWORK – INTEGER *Input*

*On entry*: the dimension of the array RWORK as declared in the (sub)program from which D06CAF is called.

*Constraint*: LRWORK $\geq 2 \times$ NV $+$ NELT.

15: IFAIL – INTEGER *Input/Output*

*On entry*: IFAIL must be set to 0, $-1$ or 1. If you are unfamiliar with this parameter you should refer to Section 3.3 in the Essential Introduction 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 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.**

*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, NV $< 3$,
or NELT $> 2 \times$ NV $- 1$,
or NEDGE $< 1$,
or EDGE$(i,j) < 1$ or EDGE$(i,j) >$ NV for some $i = 1,2$ and $j = 1,2,\ldots,$NEDGE,
or EDGE$(1,j) =$ EDGE$(2,j)$ for some $j = 1,2,\ldots,$NEDGE,
or CONN$(i,j) < 1$ or CONN$(i,j) >$ NV for some $i = 1,2,3$ and $j = 1,2,\ldots,$NELT,
or CONN$(1,j) =$ CONN$(2,j)$ or CONN$(1,j) =$ CONN$(3,j)$ or CONN$(2,j) =$ CONN$(3,j)$ for some $j = 1,2,\ldots,$NELT,
or NVFIX $< 0$ or NVFIX $>$ NV,
or NUMFIX$(i) < 1$ or NUMFIX$(i) >$ NV for some $i = 1,2,\ldots,$NVFIX if NVFIX $> 0$,
or LIWORK $< 8 \times$ NELT $+ 2 \times$ NV,
or LRWORK $< 2 \times$ NV $+$ NELT.

IFAIL $= 2$

A serious error has occurred in an internal call to an auxiliary routine. Check the input mesh, especially the connectivity between triangles and vertices (the parameter CONN). Setting ITRACE $> 1$ may provide more information. If the problem persists, contact NAG.

## 7 Accuracy

Not applicable.

## 8 Further Comments

None.

## 9 Example

In this example, a uniform mesh on the unit square is randomly distorted using routines from Chapter G05. D06CAF is then used to smooth the distorted mesh and recover a uniform mesh.

### 9.1 Program Text

```
    Program d06cafe

!     D06CAF Example Program Text

!     Mark 24 Release. NAG Copyright 2012.

!     .. Use Statements ..
      Use nag_library, Only: d06caf, g05kff, g05sqf, nag_wp, x01aaf
!     .. Implicit None Statement ..
      Implicit None
!     .. Parameters ..
      Integer, Parameter               :: nin = 5, nout = 6
!     .. Local Scalars ..
      Real (Kind=nag_wp)               :: delta, hx, hy, pi2, r, rad, sk,      &
                                          theta, x1, x2, x3, y1, y2, y3
      Integer                          :: genid, i, ifail, imax, ind, itrace,  &
                                          j, jmax, k, liwork, lrwork, lseed,   &
                                          lstate, me1, me2, me3, nedge, nelt,  &
                                          nqint, nv, nvfix, reftk, subid
      Character (1)                    :: pmesh
!     .. Local Arrays ..
      Real (Kind=nag_wp), Allocatable  :: coor(:,:), rwork(:), variates(:)
      Integer, Allocatable             :: conn(:,:), edge(:,:), iwork(:),      &
                                          numfix(:), seed(:), state(:)
!     .. Intrinsic Procedures ..
      Intrinsic                        :: cos, min, real, sin
!     .. Executable Statements ..
      Write (nout,*) 'D06CAF Example Program Results'
      Flush (nout)

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

!     Read IMAX and JMAX, the number of vertices
!     in the x and y directions respectively.

      Read (nin,*) imax, jmax
      nv = imax*jmax
      nelt = 2*(imax-1)*(jmax-1)
      nedge = 2*(imax-1) + 2*(jmax-1)
      liwork = 8*nelt + 2*nv
      lrwork = 2*nv + nelt

!     The array VARIATES will be used when distorting the mesh

      Allocate (variates(2*nv),coor(2,nv),conn(3,nelt),edge(3,nedge), &
        iwork(liwork),rwork(lrwork))

!     Read distortion percentage and calculate radius
!     of distortion neighbourhood so that cross-over
!     can only occur at 100% or greater.
```

```
      Read (nin,*) delta

      hx = 1.0E0_nag_wp/real(imax-1,kind=nag_wp)
      hy = 1.0E0_nag_wp/real(jmax-1,kind=nag_wp)
      rad = 0.005E0_nag_wp*delta*min(hx,hy)
      pi2 = 2.0E0_nag_wp*x01aaf(pi2)

!     GENID identifies the base generator

      genid = 1
      subid = 1

!     For GENID = 1 only one seed is required
!     The initialiser is first called in query mode to get the value of
!     LSTATE for the chosen base generator

      lseed = 1
      lstate = -1
      Allocate (seed(lseed),state(lstate))

!     Initialise the seed

      seed(1:lseed) = (/1762541/)

      ifail = 0
      Call g05kff(genid,subid,seed,lseed,state,lstate,ifail)

      Deallocate (state)
      Allocate (state(lstate))

!     Initialise the generator to a repeatable sequence

      ifail = 0
      Call g05kff(genid,subid,seed,lseed,state,lstate,ifail)

!     Generate two sets of uniform random variates

      ifail = 0
      Call g05sqf(nv,0.0E0_nag_wp,rad,state,variates,ifail)

      ifail = 0
      Call g05sqf(nv,0.0E0_nag_wp,pi2,state,variates(nv+1),ifail)

!     Generate a simple uniform mesh and then distort it
!     randomly within the distortion neighbourhood of each
!     node.

      k = 0
      ind = 0

      Do j = 1, jmax

        Do i = 1, imax
          k = k + 1
          r = variates(k)
          theta = variates(nv+k)

          If (i==1 .Or. i==imax .Or. j==1 .Or. j==jmax) Then
            r = 0.E0_nag_wp
          End If

          coor(1,k) = real(i-1,kind=nag_wp)*hx + r*cos(theta)
          coor(2,k) = real(j-1,kind=nag_wp)*hy + r*sin(theta)

          If (i<imax .And. j<jmax) Then
            ind = ind + 1
            conn(1,ind) = k
            conn(2,ind) = k + 1
            conn(3,ind) = k + imax + 1
            ind = ind + 1
            conn(1,ind) = k
```

```
            conn(2,ind) = k + imax + 1
            conn(3,ind) = k + imax
          End If

        End Do

      End Do

      Read (nin,*) pmesh

      Write (nout,*)

      Select Case (pmesh)
      Case ('N')
        Write (nout,*) 'The complete distorted mesh characteristics'
        Write (nout,99999) 'Number of vertices =', nv
        Write (nout,99999) 'Number of elements =', nelt
      Case ('Y')

!       Output the mesh

        Write (nout,99998) nv, nelt

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

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

      reftk = 0

      Do k = 1, nelt
        me1 = conn(1,k)
        me2 = conn(2,k)
        me3 = conn(3,k)

        x1 = coor(1,me1)
        x2 = coor(1,me2)
        x3 = coor(1,me3)
        y1 = coor(2,me1)
        y2 = coor(2,me2)
        y3 = coor(2,me3)

        sk = ((x2-x1)*(y3-y1)-(y2-y1)*(x3-x1))/2.E0_nag_wp

        If (sk<0.E0_nag_wp) Then
          Write (nout,*) 'Error: the surface of the element is negative'
          Write (nout,99999) 'element number  = ', k
          Write (nout,99995) 'element surface = ', sk
          Go To 100
        End If

        If (pmesh=='Y') Then
          Write (nout,99996) conn(1,k), conn(2,k), conn(3,k), reftk
        End If

      End Do
      Flush (nout)

!     Boundary edges

      Do i = 1, imax - 1
        edge(1,i) = i
        edge(2,i) = i + 1
      End Do

      Do i = 1, jmax - 1
        edge(1,imax-1+i) = i*imax
```

```
          edge(2,imax-1+i) = (i+1)*imax
        End Do

        Do i = 1, imax - 1
          edge(1,imax-1+jmax-1+i) = imax*jmax - i + 1
          edge(2,imax-1+jmax-1+i) = imax*jmax - i
        End Do

        Do i = 1, jmax - 1
          edge(1,2*(imax-1)+jmax-1+i) = (jmax-i)*imax + 1
          edge(2,2*(imax-1)+jmax-1+i) = (jmax-i-1)*imax + 1
        End Do

        edge(3,1:nedge) = 0

        nvfix = 0
        Allocate (numfix(nvfix))

        itrace = 1
        nqint = 10

!       Call the smoothing routine

        ifail = 0
        Call d06caf(nv,nelt,nedge,coor,edge,conn,nvfix,numfix,itrace,nqint, &
          iwork,liwork,rwork,lrwork,ifail)

        Select Case (pmesh)
        Case ('N')
          Write (nout,*)
          Write (nout,*) 'The complete smoothed mesh characteristics'
          Write (nout,99999) 'Number of vertices =', nv
          Write (nout,99999) 'Number of elements =', nelt
        Case ('Y')

!         Output the mesh

          Write (nout,99998) nv, nelt

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

          reftk = 0

          Do k = 1, nelt
            Write (nout,99996) conn(1,k), conn(2,k), conn(3,k), reftk
          End Do

        End Select

100     Continue

99999 Format (1X,A,I6)
99998 Format (1X,2I10)
99997 Format (2(2X,E13.6))
99996 Format (1X,4I10)
99995 Format (1X,A,E13.6)
    End Program d06cafe
```

## 9.2 Program Data

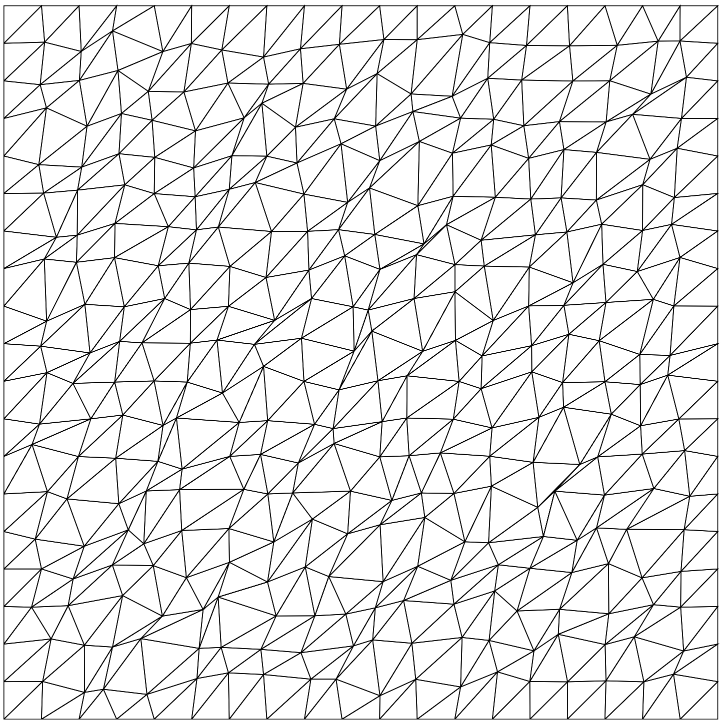
```
D06CAF Example Program Data
20 20    :IMAX JMAX
87.0     :DELTA
'N'      :Printing option 'Y' or 'N'
```

## 9.3 Program Results

```
D06CAF Example Program Results

The complete distorted mesh characteristics
Number of vertices =   400
Number of elements =   722
BEFORE SMOOTHING
Minimum smoothness measure:       1.0060557
Maximum smoothness measure:      45.7310387
Distribution interval            Number of elements
     1.0060557 -       5.4785540       715
     5.4785540 -       9.9510523         4
     9.9510523 -      14.4235506         1
    14.4235506 -      18.8960489         0
    18.8960489 -      23.3685472         0
    23.3685472 -      27.8410455         0
    27.8410455 -      32.3135438         0
    32.3135438 -      36.7860421         0
    36.7860421 -      41.2585404         0
    41.2585404 -      45.7310387         1

AFTER SMOOTHING
Minimum smoothness measure:       1.3377832
Maximum smoothness measure:       1.4445226
Distribution interval            Number of elements
     1.3377832 -       1.3484572         0
     1.3484572 -       1.3591311        13
     1.3591311 -       1.3698050        42
     1.3698050 -       1.3804790       104
     1.3804790 -       1.3911529       162
     1.3911529 -       1.4018268       159
     1.4018268 -       1.4125008       122
     1.4125008 -       1.4231747        74
     1.4231747 -       1.4338486        31
     1.4338486 -       1.4445226        14

The complete smoothed mesh characteristics
Number of vertices =   400
Number of elements =   722
```

**Example Program**

Unform mesh on the unit square randomly distorted using routines from Chapter G05



Distorted mesh smoothed and a uniform mesh recovered