# NAG Toolbox

# nag_mesh_2d_gen_front (d06ac)

## 1 Purpose

nag_mesh_2d_gen_front (d06ac) 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 Syntax

```
[nv, nelt, coor, conn, ifail] = nag_mesh_2d_gen_front(nvb, edge, coor, weight, itrace, 'nvint', nvint, 'nvmax', nvmax, 'nedge', nedge)
[nv, nelt, coor, conn, ifail] = d06ac(nvb, edge, coor, weight, itrace, 'nvint', nvint, 'nvmax', nvmax, 'nedge', nedge)
```

## 3 Description

nag_mesh_2d_gen_front (d06ac) 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 function 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

### 5.1 Compulsory Input Parameters

1: **nvb** – INTEGER

The number of vertices in the input boundary mesh.

*Constraint*: **nvb** $\geq 3$.

2: **edge**$(\mathbf{3}, \mathbf{nedge})$ – INTEGER array

The specification of the boundary edges. **edge**$(1, j)$ and **edge**$(2, j)$ contain the vertex numbers of the two end points of the $j$th boundary edge. **edge**$(3, j)$ is a user-supplied tag for the $j$th boundary edge and is not used by nag_mesh_2d_gen_front (d06ac).

*Constraint*: $1 \leq$ **edge**$(i, j) \leq$ **nvb** and **edge**$(1, j) \neq$ **edge**$(2, j)$, for $i = 1, 2$ and $j = 1, 2, \ldots,$ **nedge**.

3: **coor**$(\mathbf{2}, \mathbf{nvmax})$ – REAL (KIND=nag_wp) array

**coor**$(1, i)$ contains the $x$ coordinate of the $i$th input boundary mesh vertex, for $i = 1, 2, \ldots,$ **nvb**. **coor**$(1, i)$ contains the $x$ coordinate of the $(i - \mathbf{nvb})$th fixed interior vertex, for $i =$ **nvb** $+ 1, \ldots,$ **nvb** $+$ **nvint**. For boundary and interior vertices, **coor**$(2, i)$ contains the corresponding $y$ coordinate, for $i = 1, 2, \ldots,$ **nvb** $+$ **nvint**.

4: **weight**(:) – REAL (KIND=nag_wp) array

The dimension of the array **weight** must be at least $\max(1, \mathbf{nvint})$

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 $\mathbf{nvint} > 0$, $\mathbf{weight}(i) > 0.0$, for $i = 1, 2, \ldots, \mathbf{nvint}$.

5: **itrace** – INTEGER

The level of trace information required from nag_mesh_2d_gen_front (d06ac).

**itrace** $\leq 0$
No output is generated.

**itrace** $\geq 1$
Output from the meshing solver is printed on the current advisory message unit (see nag_file_set_unit_advisory (x04ab)). This output contains details of the vertices and triangles generated by the process.

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

## 5.2 Optional Input Parameters

1: **nvint** – INTEGER

*Default*: the dimension of the array **weight**.

The number of fixed interior mesh vertices to which a weight will be applied.

*Constraint*: $\mathbf{nvint} \geq 0$.

2: **nvmax** – INTEGER

*Default*: the dimension of the array **coor**.

The maximum number of vertices in the mesh to be generated.

*Constraint*: $\mathbf{nvmax} \geq \mathbf{nvb} + \mathbf{nvint}$.

3: **nedge** – INTEGER

*Default*: the dimension of the array **edge**.

The number of boundary edges in the input mesh.

*Constraint*: $\mathbf{nedge} \geq 1$.

## 5.3 Output Parameters

1: **nv** – INTEGER

The total number of vertices in the output mesh (including both boundary and interior vertices). If $\mathbf{nvb} + \mathbf{nvint} = \mathbf{nvmax}$, no interior vertices will be generated and $\mathbf{nv} = \mathbf{nvmax}$.

2: **nelt** – INTEGER

The number of triangular elements in the mesh.

3: **coor**$(\mathbf{2}, \mathbf{nvmax})$ – REAL (KIND=nag_wp) array

$\mathbf{coor}(1, i)$ will contain the $x$ coordinate of the $(i - \mathbf{nvb} - \mathbf{nvint})$th generated interior mesh vertex, for $i = \mathbf{nvb} + \mathbf{nvint} + 1, \ldots, \mathbf{nv}$; while $\mathbf{coor}(2, i)$ will contain the corresponding $y$ coordinate. The remaining elements are unchanged.

4: **conn**$(\mathbf{3}, \mathbf{2} \times \mathbf{nvmax} + \mathbf{5})$ – INTEGER array

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 $j = 1, 2, \ldots,$ **nelt**.

5: **ifail** – INTEGER

**ifail** $= 0$ unless the function detects an error (see Section 5).

## 6 Error Indicators and Warnings

Errors or warnings detected by the function:

**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 $j = 1, 2, \ldots,$ **nedge**,
or **edge**$(1, j) =$ **edge**$(2, j)$, for some $j = 1, 2, \ldots,$ **nedge**,
or if **nvint** $> 0$, **weight**$(i) \leq 0.0$, for some $i = 1, 2, \ldots,$ **nvint**;
or $lrwork < 12 \times$ **nvmax** $+ 30015$,
or $liwork < 8 \times$ **nedge** $+ 53 \times$ **nvmax** $+ 2 \times$ **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.

**ifail** $= -399$

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

**ifail** $= -999$

Dynamic memory allocation failed.

## 7 Accuracy

Not applicable.

## 8 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**.

# 9 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. Note that the particular mesh generated could be sensitive to the ***machine precision*** and therefore may differ from one implementation to another.

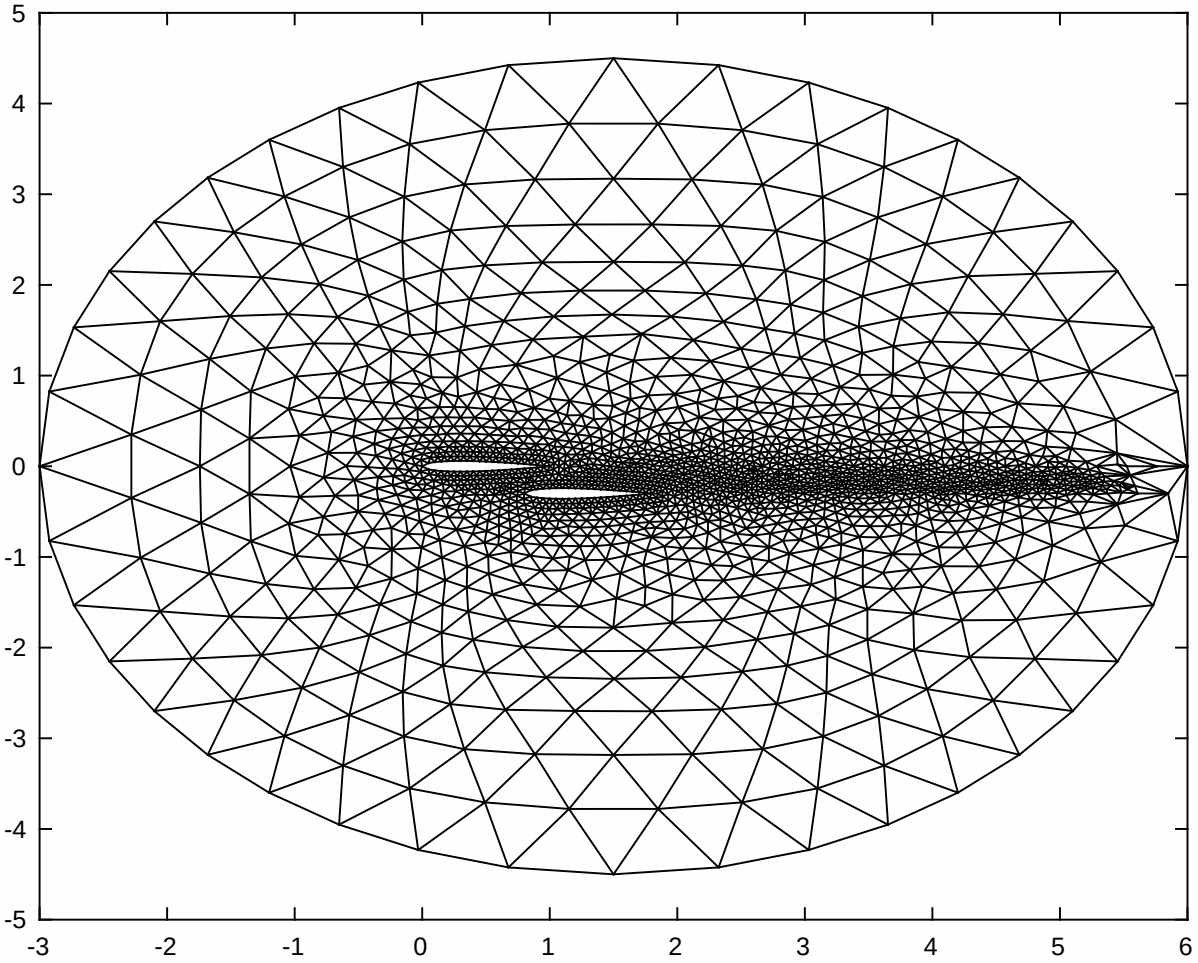
## 9.1 Program Text

```
function d06ac_example

fprintf('d06ac example results\n\n');

% The characteristic points of the boundary mesh
coorch = [0, 1, -3, 6,  0.8,  1.8, 1.5,  1.5;
          0, 0,  0, 0, -0.3, -0.3, 4.5, -4.5];
coorus = zeros(2,100);
% The lines of the boundary mesh
blines = [nag_int(21), 21, 11, 11, 21, 21, 11, 11;
                  2,   1,  3,  4,  6,  5,  7,  8;
                  1,   2,  8,  7,  5,  6,  3,  4;
                  1,   2,  3,  3,  4,  5,  3,  3];
rate = ones(8,1);

% The number of connected components to the boundary
ncomp = nag_int(3);
% Number and direction of lines per contour
nlcomp = [nag_int(-2), 4, -2];
% List of line numbers
lcomp = [nag_int(1), 2, 3, 8, 4, 7, 5, 6];

user = struct('x0', 1.5, 'y0', 0, 'radius', 4.5, 'x1', 0.8, 'y1', -0.3);

nvmax = nag_int(2000);
nedmx = nag_int(200);
itrace = nag_int(0);

[nvb, coor, nedge, edge, user, ifail] = ...
d06ba( ...
       coorch, blines, @fbnd, coorus, rate, nlcomp, lcomp, nvmax, ...
       nedmx, itrace, 'user', user);
fprintf('\nBoundary mesh characteristics:\n');
fprintf('  nvb   = %d\n', nvb);
fprintf('  nedge = %d\n', nedge);

% Generation of interior vertices for the wake of the first naca
nvint  = 40;
nvint2 = 20;
dnvint = 5/(nvint2+1);
weight = ones(nvint,1)/nvint2;
for i=1:nvint2
  coor(1, nvb+i) = 1+i*dnvint;
end
% ... for the wake of the second naca
for i = nvint2+1:nvint
  coor(1, double(nvb)+i) = 1.8 + (i-nvint2)*dnvint;
  coor(2, double(nvb)+i) = -0.3;
end

% Call the 2D advancing front mesh generator.  Note only pass relevant
% portion of edge
[nv, nelt, coor, conn, ifail] = ...
d06ac( ...
       nvb, edge(:,1:nedge), coor, weight, itrace);
```

```
fprintf('\nComplete mesh characteristics:\n');
fprintf('  nv    = %d\n', nv);
fprintf('  nelt  = %d\n', nelt);

% Plot mesh
fig1 = figure;
triplot(transpose(double(conn(:,1:nelt))), coor(1,:), coor(2,:));

function [result, user] = fbnd(i, x, y, user)

  result = 0;
  c = 1.008930411365;
  p4 = @(x) 0.6*( 0.2969*sqrt(c*x) - 0.126*c*x - 0.3516*(c*x)^2 + ...
   0.2843*(c*x)^3 - 0.1015*(c*x)^4 );
  if (i==1)
    % upper naca0012 wing beginning at the origin
    result = p4(x) -c*y;
  elseif (i==2)
    % lower naca0012 wing beginning at the origin
    result = p4(x) + c*y;
  elseif (i==3)
    result = (x-user.x0)^2 + (y-user.y0)^2 - user.radius^2;
  elseif (i==4)
    % upper naca0012 wing beginning at (user.x1;user.y1)
    result = p4(x-user.x1) - c*(y-user.y1);
  elseif (i==5)
    % lower naca0012 wing beginning at (user.x1;user.y1)
    result = p4(x-user.x1) + c*(y-user.y1);
  end
```

## 9.2 Program Results

```
     d06ac example results

Boundary mesh characteristics:
  nvb   = 120
  nedge = 120

Complete mesh characteristics:
  nv    = 1894
  nelt  = 3664
```


5
4
3
2
1
0
-1
-2
-3
-4
-5
-3
-2
-1
0
1
2
3
4
5
6