NAG Toolbox

nag_mesh_2d_gen_boundary (d06ba)

1 Purpose

nag_mesh_2d_gen_boundary (d06ba) generates a boundary mesh on a closed connected subdomain Ω of $\mathbb{R}^2$.

2 Syntax

```
[nvb, coor, nedge, edge, user, ifail] = nag_mesh_2d_gen_boundary(coorch, lined, fbnd, crus, rate, nlcomp, lcomp, nvmax, nedmx, itrace, 'nlines', nlines, 'sdcrus', sdcrus, 'ncomp', ncomp, 'user', user)
[nvb, coor, nedge, edge, user, ifail] = d06ba(coorch, lined, fbnd, crus, rate, nlcomp, lcomp, nvmax, nedmx, itrace, 'nlines', nlines, 'sdcrus', sdcrus, 'ncomp', ncomp, 'user', user)
```

3 Description

Given a closed connected subdomain Ω of $\mathbb{R}^2$, whose boundary $\partial\Omega$ is divided by characteristic points into m distinct line segments, nag_mesh_2d_gen_boundary (d06ba) 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 function is primarily designed for use with either nag_mesh_2d_gen_inc (d06aa) (a simple incremental method) or nag_mesh_2d_gen_delaunay (d06ab) (Delaunay–Voronoi method) or nag_mesh_2d_gen_front (d06ac) (Advancing Front method) to triangulate the interior of the domain Ω. For more details about the boundary and interior mesh generation, 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: **coorch**$(\mathbf{2}, \mathbf{nlines})$ – REAL (KIND=nag_wp) array

coorch$(1, i)$ contains the x coordinate of the ith characteristic point, for $i = 1, 2, \ldots,$ **nlines**; while **coorch**$(2, i)$ contains the corresponding y coordinate.

2: **lined**$(\mathbf{4}, \mathbf{nlines})$ – INTEGER array

The description of the lines that define the boundary domain. The line i, for $i = 1, 2, \ldots, m$, 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$(:,p+1),\ldots,$ **crus**$(:,p+r-3)$,
coorch$(:,k)$,
where $z\in\{1,2\}$, $r=$ **lined**$(1,i)$, $j=$ **lined**$(2,i)$ and $k=$ **lined**$(3,i)$.

Constraints:

$2\leq$ **lined**$(1,i)$;
$1\leq$ **lined**$(2,i)\leq$ **nlines**;
$1\leq$ **lined**$(3,i)\leq$ **nlines**;
lined$(2,i)\neq$ **lined**$(3,i)$, for $i=1,2,\ldots,$ **nlines**.

For each line described by **fbnd** (lines with **lined**$(4,i)>0$, for $i=1,2,\ldots,$ **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**$(2,i))$, **user**, **user**$)=0$;

fbnd(**lined**$(4,i)$, **coorch**$(1,$ **lined**$(3,i))$, **coorch**$(2,$ **lined**$(3,i))$, **user**, **user**$)=0$, for $i=1,2,\ldots,$ **nlines**.

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

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

or

$$-\textbf{lined}(4,i)+\textbf{lined}(1,i)-2=-\textbf{lined}(4,j),$$

for a j such that $j=1,2,\ldots,$ **nlines**, $j\neq i$ and **lined**$(4,j)<0$.

3: **fbnd** – REAL (KIND=nag_wp) FUNCTION, supplied by the user.

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 **lined**$(4,i)>0$ **fbnd** will never be referenced by nag_mesh_2d_gen_boundary (d06ba) and **fbnd** may be the string `'d06bad'`. (nag_mesh_2d_gen_boundary_dummy_fbnd (d06bad) is included in the NAG Toolbox.)

```
[result, user] = fbnd(ii, x, y, user)
```

Input Parameters

1: **ii** – INTEGER

lined$(4,i)$, the reference index of the line (portion of the contour) i described.

2: **x** – REAL (KIND=nag_wp)

3: **y** – REAL (KIND=nag_wp)

The values of x and y at which $f(x,y)$ is to be evaluated.

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

fbnd is called from nag_mesh_2d_gen_boundary (d06ba) with the object supplied to nag_mesh_2d_gen_boundary (d06ba).

Output Parameters

1: **result**

The value of $f(x,y)$ at the specified point.

2: **user** – REAL (KIND=nag_wp) array

4: **crus**(**2**, **sdcrus**) – REAL (KIND=nag_wp) array

The coordinates of the intermediate points for polygonal arc lines. For a line i defined as a polygonal arc (i.e., **lined**$(4,i) < 0$), if $p = -$**lined**$(4,i)$, then **crus**$(1,k)$, for $k = p, \ldots, p +$ **lined**$(1,i) - 3$, must contain the x coordinate of the consecutive intermediate points for this line. Similarly **crus**$(2,k)$, for $k = p, \ldots, p +$ **lined**$(1,i) - 3$, must contain the corresponding y coordinate.

5: **rate**(**nlines**) – REAL (KIND=nag_wp) array

rate(i) is the geometric progression ratio between the points to be generated on the line i, for $i = 1, 2, \ldots, m$ and **lined**$(4,i) \geq 0$.

If **lined**$(4,i) < 0$, **rate**(i) is not referenced.

Constraint: if **lined**$(4,i) \geq 0$, **rate**$(i) > 0.0$, for $i = 1, 2, \ldots,$ **nlines**.

6: **nlcomp**(**ncomp**) – INTEGER array

|**nlcomp**(k)| is the number of line segments in component k of the contour. The line i of component k runs in the direction **lined**$(2,i)$ to **lined**$(3,i)$ if **nlcomp**$(k) > 0$, and in the opposite direction otherwise; for $k = 1, 2, \ldots, n$.

Constraints:

$1 \leq |\mathbf{nlcomp}(k)| \leq \mathbf{nlines}$, for $k = 1, 2, \ldots,$ **ncomp**;

$\sum_{k=1}^{n} |\mathbf{nlcomp}(k)| = \mathbf{nlines}$.

7: **lcomp**(**nlines**) – INTEGER array

lcomp must contain the list of line numbers for the each component of the boundary. Specifically, the line numbers for the kth component of the boundary, for $k = 1, 2, \ldots,$ **ncomp**, must be in elements $l1 - 1$ to $l2 - 1$ of **lcomp**, where $l2 = \sum_{i=1}^{k} |\mathbf{nlcomp}(i)|$ and $l1 = l2 + 1 - |\mathbf{nlcomp}(k)|$.

Constraint: **lcomp** must hold a valid permutation of the integers $[1,$ **nlines**$]$.

8: **nvmax** – INTEGER

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

Constraint: **nvmax** $\geq$ **nlines**.

9: **nedmx** – INTEGER

The maximum number of boundary edges in the boundary mesh to be generated.

Constraint: **nedmx** ≥ 1.

10: **itrace** – INTEGER

The level of trace information required from nag_mesh_2d_gen_boundary (d06ba).

itrace $= 0$ or **itrace** < -1
No output is generated.

itrace $= 1$
Output from the boundary mesh generator is printed on the current advisory message unit (see nag_file_set_unit_advisory (x04ab)). 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 function, calling either nag_mesh_2d_gen_inc (d06aa), nag_mesh_2d_gen_delaunay (d06ab) or nag_mesh_2d_gen_front (d06ac).

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.

5.2 Optional Input Parameters

1: **nlines** – INTEGER

Default: the dimension of the arrays **coorch**, **lined**, **rate**, **lcomp**. (An error is raised if these dimensions are not equal.)

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** ≥ 1.

2: **sdcrus** – INTEGER

Default: the second dimension of the array **crus**.

The second dimension of the array **crus**.

Constraint: **sdcrus** $\geq \sum_{\{i,\mathbf{lined}(4,i)<0\}} \{\mathbf{lined}(1,i) - 2\}$.

3: **ncomp** – INTEGER

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

n, the number of separately connected components of the boundary.

Constraint: **ncomp** ≥ 1.

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

user is not used by nag_mesh_2d_gen_boundary (d06ba), but is passed to **fbnd**. Note that for large objects it may be more efficient to use a global variable which is accessible from the m-files than to use **user**.

5.3 Output Parameters

1: **nvb** – INTEGER

The total number of boundary mesh vertices generated.

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

coor$(1, i)$ will contain the x coordinate of the ith boundary mesh vertex generated, for $i = 1, 2, \ldots,$ **nvb**; while **coor**$(2, i)$ will contain the corresponding y coordinate.

3: **nedge** – INTEGER

The total number of boundary edges in the boundary mesh.

4: **edge**$(\mathbf{3}, \mathbf{nedmx})$ – INTEGER array

The specification of the boundary edges. **edge**$(1, j)$ and **edge**$(2, j)$ will contain the vertex numbers of the two end points of the jth boundary edge. **edge**$(3, j)$ is a reference number for the jth boundary edge and

edge$(3, j) =$ **lined**$(4, i)$, where i and j are such that the jth edges is part of the ith line of the boundary and **lined**$(4, i) \geq 0$;

edge$(3, j) = 100 +$ **lined**$(4, i)|$, where i and j are such that the jth edges is part of the ith line of the boundary and **lined**$(4, i) < 0$.

5: **user** – REAL (KIND=nag_wp) array

6: **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, **nlines** < 1;
or **nvmax** $<$ **nlines**;
or **nedmx** < 1;
or **ncomp** < 1;
or $lrwork < 2 \times ($**nlines** $+$ **sdcrus**$) + 2 \times \max_{i=1,2,\ldots,m}\{$**lined**$(1, i)\} \times$ **nlines**;
or $liwork <$ $\sum_{\{i,\mathbf{lined}(4,i)<0\}} \{$**lined**$(1, i) - 2\} + 8 \times$ **nlines** $+$ **nvmax** $+ 3 \times$ **nedmx** $+ 2 \times$ **sdcrus**;
or **sdcrus** $<$ $\sum_{\{i,\mathbf{lined}(4,i)<0\}} \{$**lined**$(1, i) - 2\}$;
or **rate**$(i) < 0.0$ for some $i = 1, 2, \ldots,$ **nlines** with **lined**$(4, i) \geq 0$;
or **lined**$(1, i) < 2$ for some $i = 1, 2, \ldots,$ **nlines**;
or **lined**$(2, i) < 1$ or **lined**$(2, i) >$ **nlines** for some $i = 1, 2, \ldots,$ **nlines**;
or **lined**$(3, i) < 1$ or **lined**$(3, i) >$ **nlines** for some $i = 1, 2, \ldots,$ **nlines**;
or **lined**$(2, i) =$ **lined**$(3, i)$ for some $i = 1, 2, \ldots,$ **nlines**;
or **nlcomp**$(k) = 0$, or $|$**nlcomp**$(k)| >$ **nlines** for a $k = 1, 2, \ldots,$ **ncomp**;
or $\sum_{k=1}^{n} |$**nlcomp**$(k)| \neq$ **nlines**;

or **lcomp** does not represent a valid permutation of the integers in $[1, \mathbf{nlines}]$;
or one of the end points for a line i described by the user-supplied function (lines with $\mathbf{lined}(4, i) > 0$, for $i = 1, 2, \ldots, \mathbf{nlines}$) does not belong to the corresponding curve in **fbnd**;
or the intermediate points for the lines described as polygonal arcs (lines with $\mathbf{lined}(4, i) < 0$, for $i = 1, 2, \ldots, \mathbf{nlines}$) 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.

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 boundary mesh generation technique in this function 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 function 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.

9 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.

9.1 Program Text

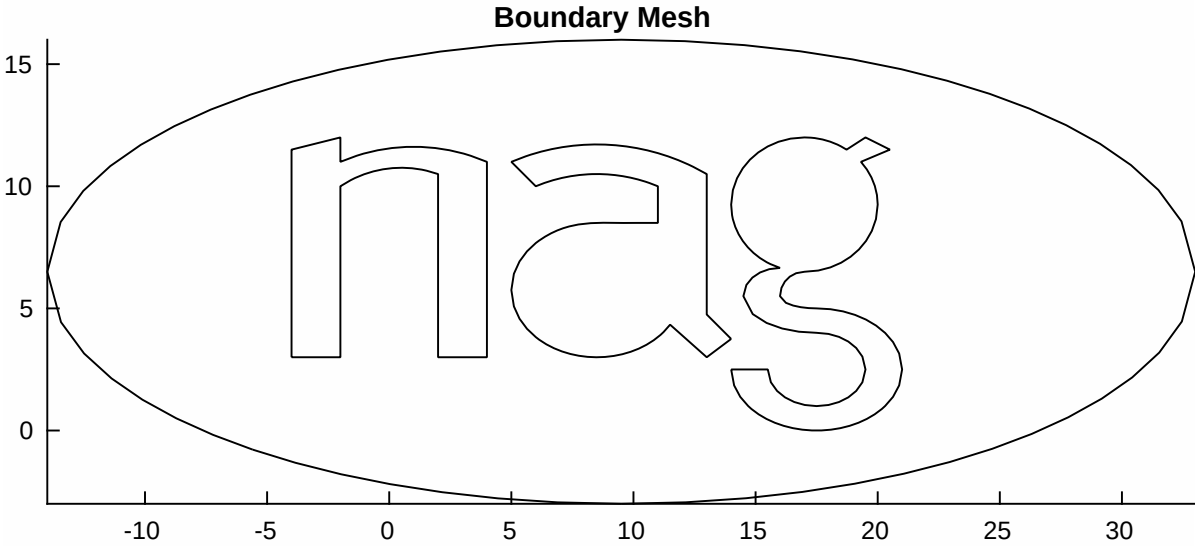
```
     function d06ba_example

fprintf('d06ba example results\n\n');

% Desciption of Nag Logo boundary in terms of arguments to d06ba
nlines = 45;
line = zeros(4,nlines,nag_int_name);
line(1:4,1:nlines) = [ 15  1  2  1;    15  2  3  1;    15  3  4  1;
        15  4  1  1;     4  6  5 -1;    10 10  6  0;
        10 14 10  2;    10  7 14  0;     4  8  7  0;
        10 13  8  0;    10 13  9  3;    10 12  9  0;
         4 11 12  0;    15  5 11  0;    15 26 15  4;
        10 26 25  0;     4 25 24  0;     4 24 23  0;
         4 23 22  0;    10 21 22  6;    10 20 21  6;
        10 19 20  6;     4 19 18  0;     5 18 17  0;
        15 17 16  5;     4 16 15  0;     4 27 28  0;
         7 28 30  8;     7 30 32  8;     7 32 34  8;
         6 36 34 10;     6 38 36 12;    10 40 38 13;
        10 42 40 13;     8 44 42 13;     4 44 45  0;
         4 45 43  0;     4 43 41  0;     6 39 41 13;
        10 37 39 13;     6 37 35 11;     6 35 33  9;
        10 31 33  7;    10 29 31  7;    10 27 29  7]';
coorch = zeros(2,nlines);
coorch(1,:) = ...
   [ 9.5 33.0  9.5 -14.0                                                ...
    -4.0 -2.0  2.0   4.0 -2.0    -2.0  -4.0 -2.0     4.0   2.0          ...
     5.0  6.0 11.0  11.0  8.5     5.0   8.5  11.5    13.0 14.0 13.0 13.0 ...
    14.0 15.5 17.5  17.5 21.0    19.5  17.5  17.5    16.0 14.5           ...
                    17.0 16.0    20.0  14.0  19.3142 17.0 20.5 18.7249 19.5];
coorch(2,:) = ...
   [-3.0  6.5 16.0   6.5                                                 ...
     3.0  3.0  3.0   3.0 11.0    10.0  11.5  12.0    11.0 10.5           ...
    11.0 10.0 10.0   8.5  8.5     5.75  3.0   4.3335  3.0  3.75 4.75 10.5 ...
     2.5  2.5  0.0   1.0  2.5     2.5   5.0   4.0     5.5  5.5            ...
                     6.5  6.6573  9.25  9.25 11.0    12.0 11.5  11.5 12.0];

crus = zeros(2, 2);
crus(1,1:2)=[-8,-10]/3;
crus(2,1:2) = [3,3];
rate = ones(nlines,1);
rate(1:4,1) = [0.95;1.05;0.95;1.05];
nlcomp = nag_int([4 10 12 19]);
lcomp = zeros(nlines,1,nag_int_name);
lcomp(1:nlines,1) = [  1  2  3  4                        ...
             14 13 12 11 10  9  8  7  6  5        ...
                    18 19 20 21 22 23 24 25 26 15 16 17 ...
                    27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45];
nvmax = nag_int(5000);
nedmx = nag_int(1000);
itrace = nag_int(-1);
user = [23.5; 9.5; 9.5; 6.5];

% Generate boundary mesh
[nvb, coor, nedge, edge, user, ifail] = ...
d06ba( ...
       coorch, line, @fbnd, crus, rate, nlcomp, lcomp, nvmax, ...
       nedmx, itrace, 'user', user);

i = nag_int(0);
k = nag_int(0);
```

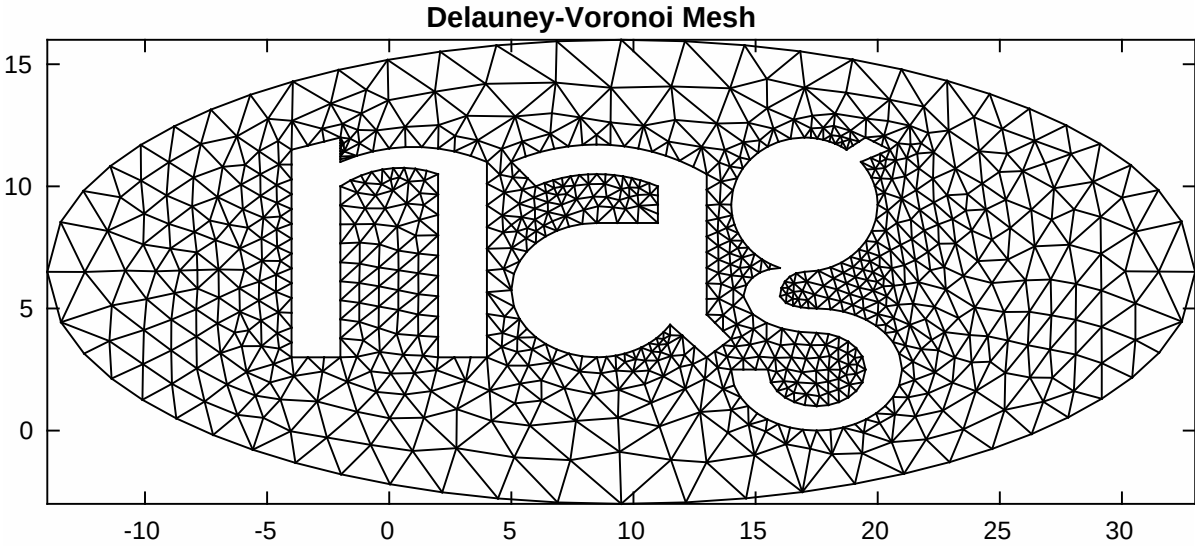
```
fig1 = figure;
hold on;
title('Boundary Mesh');

for seg = 1:4
  i0 = i;
  for j = k+1:k+nlcomp(seg)
    i = i + line(1,lcomp(j,1));
  end
  k = k + nlcomp(seg);
  i = i - nlcomp(seg);
  segl = i - i0;
  for j = 1:segl
    segx(j) = coor(1,i0+j);
    segy(j) = coor(2,i0+j);
  end
  segx(segl+1) = segx(1);
  segy(segl+1) = segy(1);
  plot(segx(1:segl+1),segy(1:segl+1));
end
axis equal tight;

% generate Delauney-Voronoi mesh using logo boundary
npropa = nag_int(1);
itrace = nag_int(0);
weight = [];
[nv, nelt, coor, conn, ifail] = ...
d06ab( ...
       nvb, edge, coor, weight, npropa, itrace,'nedge',nedge);

% Plot mesh
fig2 = figure;
triplot(transpose(double(conn(:,1:nelt))), coor(1,:), coor(2,:));
title('Delauney-Voronoi Mesh');
axis equal tight;

fprintf('\nComplete mesh characteristics for Delauney-Voronoi mesh:\n');
fprintf('Number of vertices    = %4d\n',nv);
fprintf('Number of elements    = %4d\n',nelt);

% generate 2D advancing front mesh on logo
[nv, nelt, coor, conn, ifail] = ...
d06ac( ...
       nvb, edge(:,1:nedge), coor, weight, itrace);

% Plot mesh
fig3 = figure;
triplot(transpose(double(conn(:,1:nelt))), coor(1,:), coor(2,:));
title('Advancing Front Mesh');
axis equal tight;

fprintf('\nComplete mesh characteristics for advancing front mesh:\n');
fprintf('Number of vertices    = %4d\n',nv);
fprintf('Number of elements    = %4d\n',nelt);

function [result, user] = fbnd(i, x, y, user)
  xa = user(1);
  xb = user(2);
  x0 = user(3);
  y0 = user(4);

  result = 0;
  if (i == 1)
    % line 1,2,3, and 4: ellipse centred in (x0,y0) with
    % xa and xb as coefficients
    result = ((x-x0)/xa)^2 + ((y-y0)/xb)^2 - 1;
  elseif (i == 2)
    % line 24, 27, 33 and 38 are a circle centred in (x0,y0)
    % with radius sqrt(radius2)
    x0 = 0.5;
    y0 = 6.25;
```

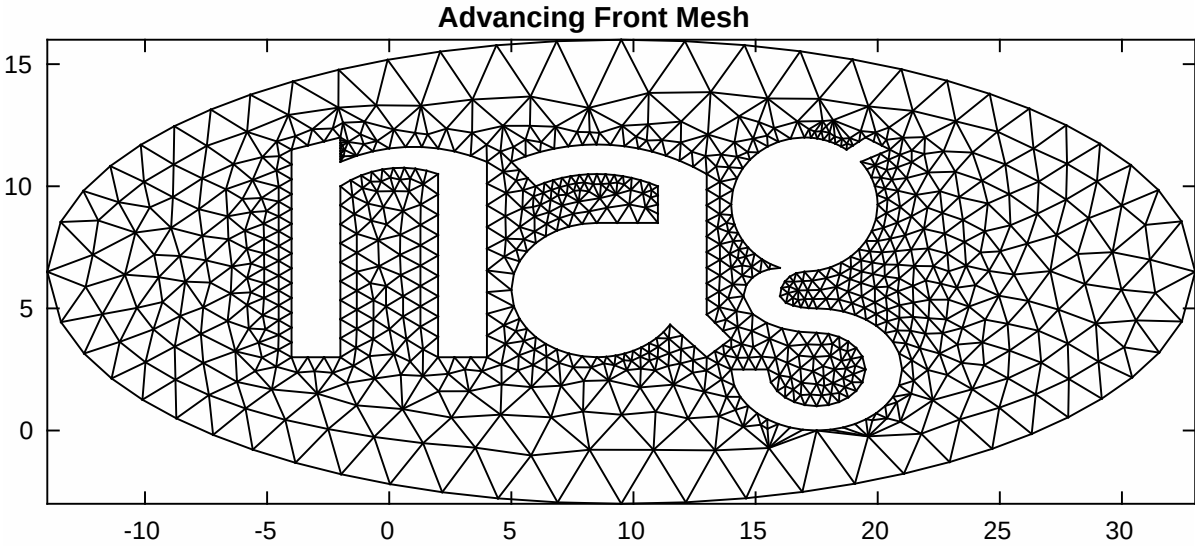
```
    radius2 = 20.3125;
    result = (x-x0)^2 + (y-y0)^2 - radius2;
  elseif (i == 3)
    x0 = 1;
    y0 = 4;
    radius2 = 9 + (11-y0)^2;
    result = (x-x0)^2 + (y-y0)^2 - radius2;
  elseif (i == 4)
    x0 = 8.5;
    y0 = 2.75;
    radius2 = (x0-5)^2 + (11-y0)^2;
    result = (x-x0)^2 + (y-y0)^2 - radius2;
  elseif (i == 5)
    x0 = 8.5;
    y0 = 4;
    radius2 = 2.5^2 + (10-y0)^2;
    result = (x-x0)^2 + (y-y0)^2 - radius2;
  elseif (i == 6)
    x0 = 8.5;
    y0 = 5.75;
    result = ((x-x0)/3.5)^2 + ((y-y0)/2.75)^2 - 1;
  elseif (i == 7)
    x0 = 17.5;
    y0 = 2.5;
    result = ((x-x0)/3.5)^2 + ((y-y0)/2.5)^2 - 1;
  elseif (i == 8)
    x0 = 17.5;
    y0 = 2.5;
    result = ((x-x0)/2)^2 + ((y-y0)/1.5)^2 - 1;
  elseif (i == 9)
    x0 = 17.5;
    y0 = 5.5;
    result = ((x-x0)/1.5)^2 + ((y-y0)/0.5)^2 - 1;
  elseif (i == 10)
    x0 = 17.5;
    y0 = 5.5;
    result = ((x-x0)/3)^2 + ((y-y0)/1.5)^2 - 1;
  elseif (i == 11)
    x0 = 17.0;
    y0 = 5.5;
    result = ((x-x0))^2 + ((y-y0))^2 - 1;
  elseif (i == 12)
    x0 = 16;
    y0 = 5.5;
    result = ((x-x0)/1.5)^2 + ((y-y0)/1.1573)^2 - 1;
  elseif (i == 13)
    x0 = 17;
    y0 = 9.25;
    result = ((x-x0)/3)^2 + ((y-y0)/2.75)^2 - 1;
  end
```

9.2 Program Results

```
     d06ba example 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

Complete mesh characteristics for Delauney-Voronoi mesh:
Number of vertices    =  904
Number of elements    = 1480

Complete mesh characteristics for advancing front mesh:
Number of vertices    =  924
Number of elements    = 1520
```


Boundary Mesh
15
10
5
0
-10
-5
0
5
10
15
20
25
30


Delauney-Voronoi Mesh
15
10
5
0
-10
-5
0
5
10
15
20
25
30


Advancing Front Mesh
15
10
5
0
-10
-5
0
5
10
15
20
25
30