NAG Toolbox

nag_mesh_2d_gen_delaunay (d06ab)

1 Purpose

nag_mesh_2d_gen_delaunay (d06ab) generates a triangular mesh of a closed polygonal region in $\mathbb{R}^2$, given a mesh of its boundary. It uses a Delaunay–Voronoi process, based on an incremental method.

2 Syntax

```
[nv, nelt, coor, conn, ifail] = nag_mesh_2d_gen_delaunay(nvb, edge, coor, weight, npropa, itrace, 'nvint', nvint, 'nvmax', nvmax, 'nedge', nedge)
[nv, nelt, coor, conn, ifail] = d06ab(nvb, edge, coor, weight, npropa, itrace, 'nvint', nvint, 'nvmax', nvmax, 'nedge', nedge)
```

3 Description

nag_mesh_2d_gen_delaunay (d06ab) generates the set of interior vertices using a Delaunay–Voronoi 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** ≥ 3.

2: **edge**$(3, \mathbf{nedge})$ – 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 jth boundary edge. **edge**$(3, j)$ is a user-supplied tag for the jth boundary edge and is not used by nag_mesh_2d_gen_delaunay (d06ab).

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

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

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

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: **npropa** – INTEGER

The propagation type and coefficient, the argument **npropa** is used when the internal points are created. They are distributed in a geometric manner if **npropa** is positive and in an arithmetic manner if it is negative. For more details see Section 9.

Constraint: $\mathbf{npropa} \neq 0$.

6: **itrace** – INTEGER

The level of trace information required from nag_mesh_2d_gen_delaunay (d06ab).

itrace ≤ 0
No output is generated.

itrace ≥ 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

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 **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 $j = 1, 2, \ldots, \mathbf{nelt}$.

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 $j = 1, 2, \ldots, \mathbf{nedge}$,
or **edge**$(1, j) =$ **edge**$(2, j)$, for some $j = 1, 2, \ldots, \mathbf{nedge}$,
or **npropa** $= 0$;
or if **nvint** > 0, **weight**$(i) \leq 0.0$, for some $i = 1, 2, \ldots, \mathbf{nvint}$;
or $lrwork < 12 \times \mathbf{nvmax} + 15$,
or $liwork < 6 \times \mathbf{nedge} + 32 \times \mathbf{nvmax} + 2 \times \mathbf{nvb} + 78$.

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 $= 3$

An error has occurred during the generation of the boundary mesh. It appears that **nvmax** is not large enough.

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. To dilute the influence of the data on the interior of the domain, the value of **npropa** can be changed. The propagation coefficient is calculated as: $\omega = 1 + \dfrac{a - 1.0}{20.0}$, where a is the absolute value of **npropa**. 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)). This generation uses the coefficient ω, and it is geometric if **npropa** > 0, and arithmetic otherwise. But increasing the value of a may lead to failure of the process, due to precision, especially in geometries with holes. So you are advised to manipulate the argument **npropa** with care.

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 $(1.0, 0.0)$ with a radius 3. The main wing, using aerofoil RAE 2822 data, lies between the origin and the centre of the circle, while the secondary aerofoil is produced from the first by performing a translation, a scale reduction and a rotation. 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 aerofoils.

9.1 Program Text

```
function d06ab_example

fprintf('d06ab example results\n\n');

% Boundary mesh sizes
nrae = 128;
nvc = 40;
nvb = nag_int(2*nrae + nvc);
edge = zeros(3, nvb, nag_int_name);
% Maximum number of vertices in final mesh
nvmax = 6000;
coor = zeros(2, nvmax);

% Circular boundary
centre = [1.0,0.0];
radius = 3.0;
for i = 1:nvc
  theta = 2*pi*(i-1)/nvc;
  coor(1,i) = radius*cos(theta) + centre(1);
  coor(2,i) = radius*sin(theta) + centre(2);
  edge(1,i) = i;
  edge(2,i) = i + 1;
  edge(3,i) = 1;
end
edge(2,nvc) = 1;

% data for aerofoil RAE 2822
rae = [ 0.000000  0.000000; 0.000602  0.003165; 0.002408  0.006306;
        0.005412  0.009416; 0.009607  0.012480; 0.014984  0.015489;
        0.021530  0.018441; 0.029228  0.021348; 0.038060  0.024219;
        0.048005  0.027062; 0.059039  0.029874; 0.071136  0.032644;
        0.084265  0.035360; 0.098396  0.038011; 0.113495  0.040585;
        0.129524  0.043071; 0.146447  0.045457; 0.164221  0.047729;
        0.182803  0.049874; 0.202150  0.051885; 0.222215  0.053753;
        0.242949  0.055470; 0.264302  0.057026; 0.286222  0.058414;
        0.308658  0.059629; 0.331555  0.060660; 0.354858  0.061497;
```

```
        0.378510  0.062133; 0.402455  0.062562; 0.426635  0.062779;
        0.450991  0.062774; 0.475466  0.062530; 0.500000  0.062029;
        0.524534  0.061254; 0.549009  0.060194; 0.573365  0.058845;
        0.597545  0.057218; 0.621490  0.055344; 0.645142  0.053258;
        0.668445  0.050993; 0.691342  0.048575; 0.713778  0.046029;
        0.735698  0.043377; 0.757051  0.040641; 0.777785  0.037847;
        0.797850  0.035017; 0.817197  0.032176; 0.835779  0.029347;
        0.853553  0.026554; 0.870476  0.023817; 0.886505  0.021153;
        0.901604  0.018580; 0.915735  0.016113; 0.928864  0.013769;
        0.940961  0.011562; 0.951995  0.009508; 0.961940  0.007622;
        0.970772  0.005915; 0.978470  0.004401; 0.985016  0.003092;
        0.990393  0.002001; 0.994588  0.001137; 0.997592  0.000510;
        0.999398  0.000128; 1.000000  0.000000; 0.999398  0.000035;
        0.997592  0.000137; 0.994588  0.000296; 0.990393  0.000497;
        0.985016  0.000719; 0.978470  0.000935; 0.970772  0.001112;
        0.961940  0.001212; 0.951995  0.001197; 0.940961  0.001033;
        0.928864  0.000694; 0.915735  0.000157; 0.901604 -0.000600;
        0.886505 -0.001592; 0.870476 -0.002829; 0.853553 -0.004314;
        0.835779 -0.006048; 0.817197 -0.008027; 0.797850 -0.010244;
        0.777785 -0.012690; 0.757051 -0.015357; 0.735698 -0.018232;
        0.713778 -0.021289; 0.691342 -0.024495; 0.668445 -0.027814;
        0.645142 -0.031207; 0.621490 -0.034631; 0.597545 -0.038043;
        0.573365 -0.041397; 0.549009 -0.044642; 0.524534 -0.047719;
        0.500000 -0.050563; 0.475466 -0.053099; 0.450991 -0.055257;
        0.426635 -0.056979; 0.402455 -0.058224; 0.378510 -0.058974;
        0.354858 -0.059236; 0.331555 -0.059046; 0.308658 -0.058459;
        0.286222 -0.057547; 0.264302 -0.056376; 0.242949 -0.054994;
        0.222215 -0.053427; 0.202150 -0.051694; 0.182803 -0.049805;
        0.164221 -0.047773; 0.146447 -0.045610; 0.129524 -0.043326;
        0.113495 -0.040929; 0.098396 -0.038431; 0.084265 -0.035843;
        0.071136 -0.033170; 0.059039 -0.030416; 0.048005 -0.027586;
        0.038060 -0.024685; 0.029228 -0.021722; 0.021530 -0.018707;
        0.014984 -0.015649; 0.009607 -0.012559; 0.005412 -0.009443;
        0.002408 -0.006308; 0.000602 -0.003160];

% Transform RAE 2822 for secondary foil by rotating (theta),
% contracting (0.4) and translating (Ttrans).
theta = pi/12.0;
c = cos(theta);
s = sin(theta);
Trot = [c s;-s,c];
Ttrans = [0.75+0.25*c; -0.25*s];
coor(1:2,nvc+1:nvc+nrae) = transpose(rae);
coor(1:2,nvc+nrae+1:nvb) = 0.4*Trot*transpose(rae);
for i = nvc+nrae+1:nvb
  coor(1:2,i) = coor(1:2,i) + Ttrans;
end

% Edges
for i = 1:nvb
  edge(1,i) = i;
  edge(2,i) = i + 1;
  edge(3,i) = 0;
end
edge(2,nvc) = 1;
edge(2,nvc+nrae) = nvc + 1;
edge(2,nvb) = nvc + nrae + 1;

% Interior vertices on the wake of the aerofiols
d_interior = 2.5/(nvc+1);
for i = 1:nvc
  i1 = nvc + i;
  coor(1,nvb+i) = 1.38 + i*d_interior;
  coor(2,nvb+i) = -tan(theta)*(coor(1,nvb+i)-0.75);
end

% Weights for interior vertices
weight(1:nvc) = 0.01;

% Generate mesh using arithmetic propgation coefficient 1.2
npropa = nag_int(-5);
```

```
itrace = nag_int(0);
[nv, nelt, coor, conn, ifail] = ...
d06ab( ...
       nvb, edge, coor, weight, npropa, itrace);

fprintf('\nnv   = %d\n', nv);
fprintf('nelt = %d\n', nelt);
% Plot mesh
fig1 = figure;
triplot(transpose(double(conn(:,1:nelt))), coor(1,:), coor(2,:));
axis equal; % To ensure that circles look like circles
title('Triangulation for Aerofoils RAE 2822 and wake vertices');
```

9.2 Program Results

```
     d06ab example results

nv   = 2327
nelt = 4360
```