NAG C Library Function Document

nag_mesh2d_delaunay (d06abc)

1 Purpose

nag_mesh2d_delaunay (d06abc) 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 Specification

```
void nag_mesh2d_delaunay (Integer nvb, Integer nvint, Integer nvmax,
     Integer nedge, const Integer edge[], Integer *nv, Integer *nelt, double coor[],
     Integer conn[], const double weight[], Integer npropa, Integer itrace,
     const char *outfile, NagError *fail)
```

3 Description

nag_mesh2d_delaunay (d06abc) generates the set of interior vertices using a Delaunay-Voronoi process, based on an incremental method. It allows the user 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

1: **nvb** – Integer *Input*

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

Constraint: **nvb** ≥ 3.

2: **nvint** – Integer *Input*

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

Constraint: **nvint** ≥ 0.

3: **nvmax** – Integer *Input*

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

Constraint: **nvmax** $\geq$ **nvb** $+$ **nvint**.

4: **nedge** – Integer *Input*

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

Constraint: **nedge** ≥ 1.

5: **edge**[3 × **nedge**] – const Integer *Input*

Note: where **EDGE**(i, j) appears in this document it refers to the array element **edge**$[3 \times (j-1) + i - 1]$. We recommend using a #define to make the same definition in your calling program.

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 this routine. Note that the edge vertices are numbered from 1 to **nvb**.

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

6: **nv** – Integer * *Output*

On exit: 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 $\mathbf{nv} = \mathbf{nvmax}$.

7: **nelt** – Integer * *Output*

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

8: **coor**[2 × **nvmax**] – double *Input/Output*

Note: where **COOR**(i, j) appears in this document it refers to the array element **coor**$[2 \times (j-1) + i - 1]$. We recommend using a #define to make the same definition in your calling program.

On entry: **COOR**$(1, i)$ contains the x-coordinate of the ith input boundary mesh vertex, for $i = 1, \ldots, \mathbf{nvb}$. **COOR**$(1, i)$ contains the x-coordinate of the $(i - \mathbf{nvb})$th fixed interior vertex, for $i = \mathbf{nvb} + 1, \ldots, \mathbf{nvb} + \mathbf{nvint}$. While **COOR**$(2, i)$ contains the corresponding y-coordinate, for $i = 1, \ldots, \mathbf{nvb} + \mathbf{nvint}$.

On exit: **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.

9: **conn**[6 × **nvmax** + 15] – Integer *Output*

Note: where **CONN**(i, j) appears in this document it refers to the array element **conn**$[3 \times (j-1) + i - 1]$. We recommend using a #define to make the same definition in your calling program.

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 $j = 1, \ldots, \mathbf{nelt}$. Note that the mesh vertices are numbered from 1 to **nv**.

10: **weight**[dim] – const double *Input*

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

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 $\mathbf{nvint} > 0$, $\mathbf{weight}[i-1] > 0.0$ for $i = 1, 2, \ldots, \mathbf{nvint}$.

11: **npropa** – Integer *Input*

On entry: the propagation type and coefficient, the parameter **npropa** is used when the internal points are created. They are distributed in a geometric manner if **npropa** is positive and in a arithmetic manner if it is negative. For more details see Section 8.

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

12: **itrace** – Integer *Input*

On entry: the level of trace information required from nag_mesh2d_delaunay (d06abc) as follows:

if **itrace** ≤ 0, no output is generated;

if **itrace** ≥ 1, then output from the meshing solver is printed. This output contains details of the vertices and triangles generated by the process.

Users are advised to set **itrace** $= 0$, unless they are experienced with Finite Element meshes.

13: **outfile** – char * *Input*

On entry: the name of a file to which diagnostic output will be directed. If **outfile** is NULL the diagnostic output will be directed to standard output.

14: **fail** – NagError * *Input/Output*

The NAG error parameter (see the Essential Introduction).

6 Error Indicators and Warnings

NE_INT

On entry, **nedge** $= \langle value \rangle$.
Constraint: **nedge** ≥ 1.

On entry, **nvb** $= \langle value \rangle$.
Constraint: **nvb** ≥ 3.

On entry, **nvint** $= \langle value \rangle$.
Constraint: **nvint** ≥ 0.

On entry, **npropa** $= 0$.

NE_INT_2

On entry, the endpoints of the edge j have the same index i: $j = \langle value \rangle$, $i = \langle value \rangle$.

NE_INT_3

On entry, **nvb** $= \langle value \rangle$, **nvint** $= \langle value \rangle$, **nvmax** $= \langle value \rangle$.
Constraint: **nvmax** $\geq$ **nvb** $+$ **nvint**.

NE_INT_4

On entry, **edge**$(i, j) < 1$ or **edge**$(i, j) >$ **nvb**, where **edge**(i, j) denotes **edge**$[3 \times (j - 1) + i - 1]$: **edge**$(i, j) = \langle value \rangle$, $i = \langle value \rangle$, $j = \langle value \rangle$, **nvb** $= \langle value \rangle$.

NE_MESH_ERROR

An error has occurred during the generation of the interior mesh. Check the inputs of the boundary.

NE_REAL_ARRAY_INPUT

On entry, **weight**$[i - 1] \leq 0.0$: **weight**$[i - 1] = \langle value \rangle$, $i = \langle value \rangle$.

NE_ALLOC_FAIL

Memory allocation failed.

NE_BAD_PARAM

On entry, parameter $\langle value \rangle$ had an illegal value.

NE_NOT_WRITE_FILE

Cannot open file ⟨*value*⟩ for writing.

NE_NOT_CLOSE_FILE

Cannot close file ⟨*value*⟩.

NE_INTERNAL_ERROR

An internal error has occurred in this function. Check the function call and any array sizes. If the call is correct then please consult NAG for assistance.

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 the user is advised to manipulate the argument **npropa** with care.

The user is 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 first RAE wing begins at the origin and it is normalised, and the last wing is a result from the first one after a translation, a scale reduction and a rotation. To enable a realistic computation on that geometry, some interior points have been introduced to cause a finer mesh in the wake of the airfoils.

The boundary mesh has 296 vertices and 296 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. The interior meshes for different values of **npropa** are given in Figure 1.

9.1 Program Text

```
/* nag_mesh2d_delaunay (d06abc) Example Program.
 *
 * Copyright 2001 Numerical Algorithms Group.
 *
 * Mark 7, 2001.
 */

#include <stdio.h>
#include <nag.h>
#include <nag_stdlib.h>
#include <nagd06.h>

#define EDGE(I,J) edge[3*((J)-1)+(I)-1]
#define CONN(I,J) conn[3*((J)-1)+(I)-1]
```

```
#define COOR(I,J) coor[2*((J)-1)+(I)-1]

int main(void)
{
  const Integer nvmax=6000, nvint=40;
  double dnvint;
  Integer exit_status, i, itrace, j, k, nedge, nelt,
    npropa, nv, nvb, reftk, i1;
  NagError fail;
  char pmesh[2];
  double *coor=0, *weight=0;
  Integer *conn=0, *edge=0;

  INIT_FAIL(fail);
  exit_status = 0;

  Vprintf("d06abc Example Program Results\n\n");

  /* Skip heading in data file */

  Vscanf("%*[^\n] ");

  /* Reading of the geometry */

  Vscanf("%ld%ld%*[^\n] ", &nvb, &nedge);

  if (nvb > nvmax)
    {
      Vprintf("Problem with the array dimensions\n");
      Vprintf(" nvb nvmax %6ld%6ld\n", nvb, nvmax);
      Vprintf(" Please increase the value of nvmax\n");
      exit_status = -1;
      goto END;
    }

  /* Allocate memory */

  if ( !(coor = NAG_ALLOC(2*nvmax, double)) ||
       !(weight = NAG_ALLOC(nvint, double)) ||
       !(conn = NAG_ALLOC(3*(2*nvmax + 5), Integer)) ||
       !(edge = NAG_ALLOC(3*nedge, Integer)) )
    {
      Vprintf("Allocation failure\n");
      exit_status = -1;
      goto END;
    }

  /* Coordinates of the boundary mesh vertices and boundary edges */

  for (i = 1; i <= nvb; ++i)
    {
      Vscanf("%ld", &i1);
      Vscanf("%lf", &COOR(1,i));
      Vscanf("%lf", &COOR(2,i));
      Vscanf("%*[^\n] ");
    }

  for (i = 1; i <= nedge; ++i)
    {
      Vscanf("%ld", &i1);
      Vscanf("%ld", &EDGE(1,i));
      Vscanf("%ld", &EDGE(2,i));
      Vscanf("%ld", &EDGE(3,i));
      Vscanf("%*[^\n] ");
    }

  Vscanf(" ' %1s '%*[^\n]", pmesh);

  /* Initialise mesh control parameters */

  itrace = 0;
```

```
  /* Generation of interior vertices on the RAE airfoils wake */

  dnvint = 2.5/(double)(nvint + 1);

  for (i = 1; i <= nvint; ++i)
    {
      i1 = nvb + i;
      COOR(1, i1) = (double)i*dnvint + 1.38;
      COOR(2, i1) = -0.27*COOR(1, i1) + 0.2;
      weight[i-1] = 0.01;
    }

  /* Loop on the propagation coef */

  for (j = 0; j < 4; ++j)
    {
      switch (j)
        {
        case 0:
          npropa = -5;
          break;
        case 1:
          npropa = -1;
          break;
        case 2:
          npropa = 1;
          break;
        default:
          npropa = 5;
        }

      /* Call to the 2D Delaunay-Voronoi mesh generator */

      d06abc(nvb, nvint, nvmax, nedge, edge, &nv, &nelt, coor,
             conn, weight, npropa, itrace, 0, &fail);

      if (fail.code == NE_NOERROR)
        {
          if (pmesh[0] == 'N')
            {
              Vprintf(" Mesh characteristics with npropa =%6ld\n", npropa);
              Vprintf(" nv   =%6ld\n", nv);
              Vprintf(" nelt =%6ld\n", nelt);
            }
          else if (pmesh[0] == 'Y')
            {
              /* Output the mesh to view it using the NAG Graphics Library */

              Vprintf(" %10ld %10ld\n", nv, nelt);

              for (i = 1; i <= nv; ++i)
                {
                  Vprintf("  %12.6e  %12.6e  \n", COOR(1,i), COOR(2,i));
                }

              reftk = 0;
              for (k = 1; k <= nelt; ++k)
                {
                  Vprintf(" %10ld %10ld %10ld %10ld\n",
                          CONN(1,k), CONN(2,k), CONN(3,k), reftk);
                }
            }
          else
            {
              Vprintf("Problem with the printing option Y or N\n");
              exit_status = -1;
              goto END;
            }
        }
      else
```

```
        {
          Vprintf("Error from d06abc.\n%s\n",   fail.message);
          exit_status = 1;
          goto END;
        }
    }

 END:
  if (coor) NAG_FREE(coor);
  if (weight) NAG_FREE(weight);
  if (conn) NAG_FREE(conn);
  if (edge) NAG_FREE(edge);

  return exit_status;
}
```

9.2 Program Data

Note: since the data file for this example is quite large only a section of it is reproduced in this document. The full data file is distributed with your implementation.

```
D06ABF Example Program Data
       296       296     :NVB NEDGE
     1  0.400000E+01  0.000000E+00
              .
              .
              .
   296  0.991387E+00  -.659880E-01 :(I1, COOR(:,I),I=1,...,NVB)
   1   1   2   0
              .
              .
              .
 296 296 169   0  :(I1, EDGE(:,I), I=1,...,NEDGE)
'N'               :Printing option 'Y' or 'N'
```

9.3 Program Results

```
d06abc Example Program Results

 Mesh characteristics with npropa =    -5
 nv   =  2317
 nelt =  4340
 Mesh characteristics with npropa =    -1
 nv   =  4420
 nelt =  8546
 Mesh characteristics with npropa =     1
 nv   =  5081
 nelt =  9868
 Mesh characteristics with npropa =     5
 nv   =  2015
 nelt =  3736
```

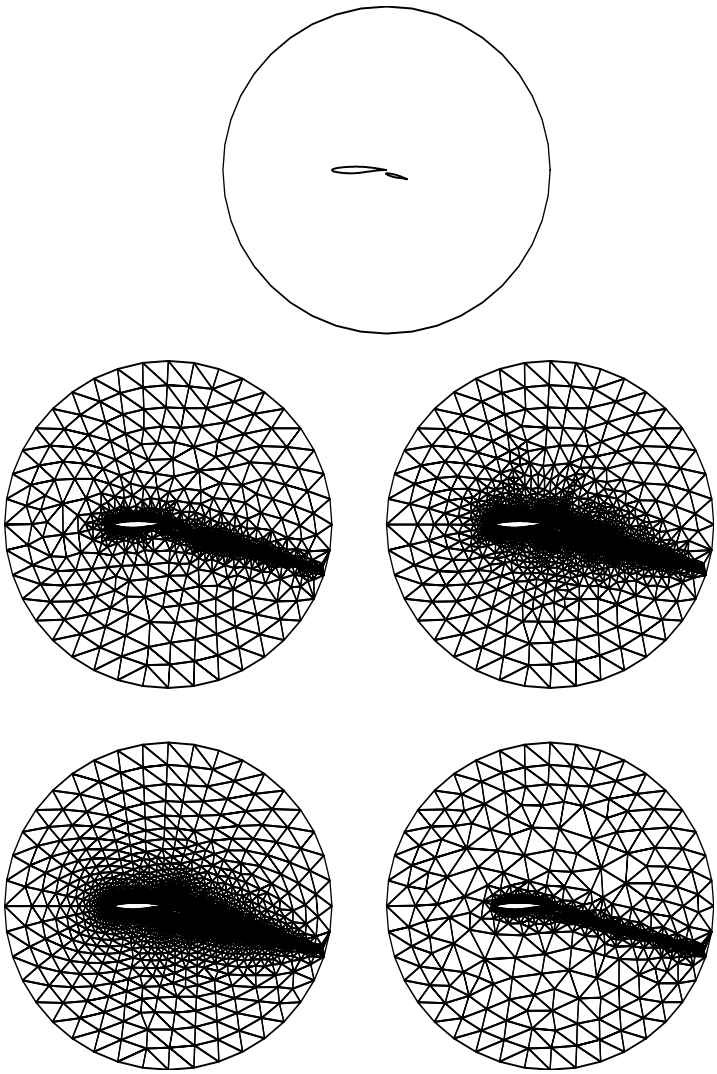

Figure 1
The boundary mesh (top), the interior mesh with **npropa** $= -5$ (middle left), -1 (middle right), 1 (bottom left) and 5 (bottom right) of a double RAE wings inside a circle geometry