

NAG Toolbox for MATLAB

d06aa

1 Purpose

d06aa generates a triangular mesh of a closed polygonal region in $\mathbb{R}^2$, given a mesh of its boundary. It uses a simple incremental method.

2 Syntax

```
[nv, nelt, coor, conn, ifail] = d06aa(edge, coor, bspace, smooth,
itrace, 'nvb', nvb, 'nvmax', nvmax, 'nedge', nedge, 'coef', coef,
'power', power)
```

3 Description

d06aa generates the set of interior vertices using a process based on a simple incremental method. A smoothing of the mesh is optionally available. 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: **edge(3,nedge)** – int32 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 d06aa.

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

2: **coor(2,nvmax)** – double array

coor(1,i) contains the x co-ordinate of the i th input boundary mesh vertex; while **coor(2,i)** contains the corresponding y co-ordinate, for $i = 1, \dots, \mathbf{nvb}$.

3: **bspace(nvb)** – double array

The desired mesh spacing (triangle diameter, which is the length of the longer edge of the triangle) near the boundary vertices.

Constraint: **bspace(i)** > 0.0 , for $i = 1, 2, \dots, \mathbf{nvb}$.

4: **smooth** – logical scalar

Indicates whether or not mesh smoothing should be performed.

If **smooth** = **true**, the smoothing is performed; otherwise no smoothing is performed.

5: **itrace – int32 scalar**

The level of trace information required from d06aa.

$$\text{itrace} \leq 0$$

No output is generated.

$$\text{itrace} \geq 1$$

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

You are advised to set **itrace** = 0, unless you are experienced with finite element meshes generation.

5.2 Optional Input Parameters

1: **nvb – int32 scalar**

Default: The dimension of the array **bspace**.

the number of vertices in the input boundary mesh.

Constraint: $3 \leq \text{nvb} \leq \text{nvmax}$.

2: **nvmax – int32 scalar**

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

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

3: **nedge – int32 scalar**

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

the number of boundary edges in the input mesh.

Constraint: **nedge** ≥ 1 .

4: **coef – double scalar**

The coefficient in the stopping criteria for the generation of interior vertices. This parameter controls the triangle density and the number of triangles generated is in $O(\text{coef}^2)$. The mesh will be finer if **coef** is greater than 0.7165 and 0.75 is a good value.

Suggested value: 0.75.

Default: 0.75

5: **power – double scalar**

Controls the rate of change of the mesh size during the generation of interior vertices. The smaller the value of **power**, the faster the decrease in element size away from the boundary.

Suggested value: 0.25.

Default: 0.25

Constraint: $0.1 \leq \text{power} \leq 10.0$.

5.3 Input Parameters Omitted from the MATLAB Interface

rwork, lrwork, iwork, liwork

5.4 Output Parameters

- 1: **nv** – **int32 scalar**
The total number of vertices in the output mesh (including both boundary and interior vertices). If **nvb** = **nvmax**, no interior vertices will be generated and **nv** = **nvb**.
- 2: **nelt** – **int32 scalar**
The number of triangular elements in the mesh.
- 3: **coor(2,nvmax)** – **double array**
coor(1,i) will contain the x co-ordinate of the $(i - \mathbf{nvb})$ th generated interior mesh vertex; while **coor(2,i)** will contain the corresponding y co-ordinate, for $i = \mathbf{nvb} + 1, \dots, \mathbf{nv}$. The remaining elements are unchanged.
- 4: **conn(3,2 × (nvmax – 1))** – **int32 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 $i = 1, 2$ and 3 , and $j = 1, \dots, \mathbf{nelt}$.
- 5: **ifail** – **int32 scalar**
0 unless the function detects an error (see Section 6).

6 Error Indicators and Warnings

Errors or warnings detected by the function:

ifail = 1

On entry, **nvb** < 3 or **nvb** > **nvmax**,
 or **nedge** < 1,
 or **edge(i,j)** < 1 or **edge(i,j)** > **nvb**, for some $i = 1, 2$ and $j = 1, \dots, \mathbf{nedge}$,
 or **edge(1,j)** = **edge(2,j)**, for some $j = 1, \dots, \mathbf{nedge}$,
 or **bspace(i)** ≤ 0.0, for some $i = 1, \dots, \mathbf{nvb}$,
 or **power** < 0.1 or **power** > 10.0,
 or **liwork** < $16 \times \mathbf{nvmax} + 2 \times \mathbf{nedge} + \max(4 \times \mathbf{nvmax} + 2, \mathbf{nedge}) - 14$,
 or **lrwork** < **nvmax**.

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.

7 Accuracy

Not applicable.

8 Further Comments

The position of the internal vertices is a function of the positions of the vertices on the given boundary. A fine mesh on the boundary results in a fine mesh in the interior. The algorithm allows you to obtain a denser interior mesh by varying **nvmax**, **bspace**, **coef** and **power**. But you are advised to manipulate the last two parameters 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

```
edge = zeros(3, 100, 'int32');
edge(1, 1) = 1;
edge(1, 2) = 2;
edge(1, 3) = 3;
edge(1, 4) = 4;
edge(1, 5) = 5;
edge(1, 6) = 6;
edge(1, 7) = 7;
edge(1, 8) = 8;
edge(1, 9) = 9;
edge(1, 10) = 10;
edge(1, 11) = 11;
edge(1, 12) = 12;
edge(1, 13) = 13;
edge(1, 14) = 14;
edge(1, 15) = 15;
edge(1, 16) = 16;
edge(1, 17) = 17;
edge(1, 18) = 18;
edge(1, 19) = 19;
edge(1, 20) = 20;
edge(1, 21) = 21;
edge(1, 22) = 22;
edge(1, 23) = 23;
edge(1, 24) = 24;
edge(1, 25) = 25;
edge(1, 26) = 26;
edge(1, 27) = 27;
edge(1, 28) = 28;
edge(1, 29) = 29;
edge(1, 30) = 30;
edge(1, 31) = 31;
edge(1, 32) = 32;
edge(1, 33) = 33;
edge(1, 34) = 34;
edge(1, 35) = 35;
edge(1, 36) = 36;
edge(1, 37) = 37;
edge(1, 38) = 38;
edge(1, 39) = 39;
edge(1, 40) = 40;
edge(1, 41) = 41;
edge(1, 42) = 42;
edge(1, 43) = 43;
edge(1, 44) = 44;
edge(1, 45) = 45;
edge(1, 46) = 46;
edge(1, 47) = 47;
edge(1, 48) = 48;
edge(1, 49) = 49;
edge(1, 50) = 50;
edge(1, 51) = 51;
edge(1, 52) = 52;
edge(1, 53) = 53;
edge(1, 54) = 54;
edge(1, 55) = 55;
edge(1, 56) = 56;
edge(1, 57) = 57;
edge(1, 58) = 58;
edge(1, 59) = 59;
edge(1, 60) = 60;
edge(1, 61) = 61;
edge(1, 62) = 62;
edge(1, 63) = 63;
edge(1, 64) = 64;
edge(1, 65) = 65;
```

```
edge(1, 66) = 66;  
edge(1, 67) = 67;  
edge(1, 68) = 68;  
edge(1, 69) = 69;  
edge(1, 70) = 70;  
edge(1, 71) = 71;  
edge(1, 72) = 72;  
edge(1, 73) = 73;  
edge(1, 74) = 74;  
edge(1, 75) = 75;  
edge(1, 76) = 76;  
edge(1, 77) = 77;  
edge(1, 78) = 78;  
edge(1, 79) = 79;  
edge(1, 80) = 80;  
edge(1, 81) = 81;  
edge(1, 82) = 82;  
edge(1, 83) = 83;  
edge(1, 84) = 84;  
edge(1, 85) = 85;  
edge(1, 86) = 86;  
edge(1, 87) = 87;  
edge(1, 88) = 88;  
edge(1, 89) = 89;  
edge(1, 90) = 90;  
edge(1, 91) = 91;  
edge(1, 92) = 92;  
edge(1, 93) = 93;  
edge(1, 94) = 94;  
edge(1, 95) = 95;  
edge(1, 96) = 96;  
edge(1, 97) = 97;  
edge(1, 98) = 98;  
edge(1, 99) = 99;  
edge(1, 100) = 100;  
edge(2, 1) = 2;  
edge(2, 2) = 3;  
edge(2, 3) = 4;  
edge(2, 4) = 5;  
edge(2, 5) = 6;  
edge(2, 6) = 7;  
edge(2, 7) = 8;  
edge(2, 8) = 9;  
edge(2, 9) = 10;  
edge(2, 10) = 11;  
edge(2, 11) = 12;  
edge(2, 12) = 13;  
edge(2, 13) = 14;  
edge(2, 14) = 15;  
edge(2, 15) = 16;  
edge(2, 16) = 17;  
edge(2, 17) = 18;  
edge(2, 18) = 19;  
edge(2, 19) = 20;  
edge(2, 20) = 21;  
edge(2, 21) = 22;  
edge(2, 22) = 23;  
edge(2, 23) = 24;  
edge(2, 24) = 25;  
edge(2, 25) = 26;  
edge(2, 26) = 27;  
edge(2, 27) = 28;  
edge(2, 28) = 29;  
edge(2, 29) = 30;  
edge(2, 30) = 31;  
edge(2, 31) = 32;  
edge(2, 32) = 33;  
edge(2, 33) = 34;  
edge(2, 34) = 35;  
edge(2, 35) = 36;
```

```
edge(2, 36) = 37;  
edge(2, 37) = 38;  
edge(2, 38) = 39;  
edge(2, 39) = 40;  
edge(2, 40) = 1;  
edge(2, 41) = 42;  
edge(2, 42) = 43;  
edge(2, 43) = 44;  
edge(2, 44) = 45;  
edge(2, 45) = 46;  
edge(2, 46) = 47;  
edge(2, 47) = 48;  
edge(2, 48) = 49;  
edge(2, 49) = 50;  
edge(2, 50) = 51;  
edge(2, 51) = 52;  
edge(2, 52) = 53;  
edge(2, 53) = 54;  
edge(2, 54) = 55;  
edge(2, 55) = 56;  
edge(2, 56) = 57;  
edge(2, 57) = 58;  
edge(2, 58) = 59;  
edge(2, 59) = 60;  
edge(2, 60) = 61;  
edge(2, 61) = 62;  
edge(2, 62) = 63;  
edge(2, 63) = 64;  
edge(2, 64) = 65;  
edge(2, 65) = 66;  
edge(2, 66) = 67;  
edge(2, 67) = 68;  
edge(2, 68) = 69;  
edge(2, 69) = 70;  
edge(2, 70) = 41;  
edge(2, 71) = 72;  
edge(2, 72) = 73;  
edge(2, 73) = 74;  
edge(2, 74) = 75;  
edge(2, 75) = 76;  
edge(2, 76) = 77;  
edge(2, 77) = 78;  
edge(2, 78) = 79;  
edge(2, 79) = 80;  
edge(2, 80) = 81;  
edge(2, 81) = 82;  
edge(2, 82) = 83;  
edge(2, 83) = 84;  
edge(2, 84) = 85;  
edge(2, 85) = 86;  
edge(2, 86) = 87;  
edge(2, 87) = 88;  
edge(2, 88) = 89;  
edge(2, 89) = 90;  
edge(2, 90) = 91;  
edge(2, 91) = 92;  
edge(2, 92) = 93;  
edge(2, 93) = 94;  
edge(2, 94) = 95;  
edge(2, 95) = 96;  
edge(2, 96) = 97;  
edge(2, 97) = 98;  
edge(2, 98) = 99;  
edge(2, 99) = 100;  
edge(2, 100) = 71;  
edge(3, 1) = 1;  
edge(3, 2) = 1;  
edge(3, 3) = 1;  
edge(3, 4) = 1;  
edge(3, 5) = 1;
```

```
edge(3, 6) = 1;  
edge(3, 7) = 1;  
edge(3, 8) = 1;  
edge(3, 9) = 1;  
edge(3, 10) = 1;  
edge(3, 11) = 1;  
edge(3, 12) = 1;  
edge(3, 13) = 1;  
edge(3, 14) = 1;  
edge(3, 15) = 1;  
edge(3, 16) = 1;  
edge(3, 17) = 1;  
edge(3, 18) = 1;  
edge(3, 19) = 1;  
edge(3, 20) = 1;  
edge(3, 21) = 1;  
edge(3, 22) = 1;  
edge(3, 23) = 1;  
edge(3, 24) = 1;  
edge(3, 25) = 1;  
edge(3, 26) = 1;  
edge(3, 27) = 1;  
edge(3, 28) = 1;  
edge(3, 29) = 1;  
edge(3, 30) = 1;  
edge(3, 31) = 1;  
edge(3, 32) = 1;  
edge(3, 33) = 1;  
edge(3, 34) = 1;  
edge(3, 35) = 1;  
edge(3, 36) = 1;  
edge(3, 37) = 1;  
edge(3, 38) = 1;  
edge(3, 39) = 1;  
edge(3, 40) = 1;  
edge(3, 41) = 1;  
edge(3, 42) = 1;  
edge(3, 43) = 1;  
edge(3, 44) = 1;  
edge(3, 45) = 1;  
edge(3, 46) = 1;  
edge(3, 47) = 1;  
edge(3, 48) = 1;  
edge(3, 49) = 1;  
edge(3, 50) = 1;  
edge(3, 51) = 1;  
edge(3, 52) = 1;  
edge(3, 53) = 1;  
edge(3, 54) = 1;  
edge(3, 55) = 1;  
edge(3, 56) = 1;  
edge(3, 57) = 1;  
edge(3, 58) = 1;  
edge(3, 59) = 1;  
edge(3, 60) = 1;  
edge(3, 61) = 1;  
edge(3, 62) = 1;  
edge(3, 63) = 1;  
edge(3, 64) = 1;  
edge(3, 65) = 1;  
edge(3, 66) = 1;  
edge(3, 67) = 1;  
edge(3, 68) = 1;  
edge(3, 69) = 1;  
edge(3, 70) = 1;  
edge(3, 71) = 1;  
edge(3, 72) = 1;  
edge(3, 73) = 1;  
edge(3, 74) = 1;  
edge(3, 75) = 1;
```

```
edge(3, 76) = 1;
edge(3, 77) = 1;
edge(3, 78) = 1;
edge(3, 79) = 1;
edge(3, 80) = 1;
edge(3, 81) = 1;
edge(3, 82) = 1;
edge(3, 83) = 1;
edge(3, 84) = 1;
edge(3, 85) = 1;
edge(3, 86) = 1;
edge(3, 87) = 1;
edge(3, 88) = 1;
edge(3, 89) = 1;
edge(3, 90) = 1;
edge(3, 91) = 1;
edge(3, 92) = 1;
edge(3, 93) = 1;
edge(3, 94) = 1;
edge(3, 95) = 1;
edge(3, 96) = 1;
edge(3, 97) = 1;
edge(3, 98) = 1;
edge(3, 99) = 1;
edge(3, 100) = 1;
coor = zeros(2, 250);
coor(1, 1) = 1;
coor(1, 2) = 0.987688;
coor(1, 3) = 0.951057;
coor(1, 4) = 0.891007;
coor(1, 5) = 0.809017;
coor(1, 6) = 0.707107;
coor(1, 7) = 0.587785;
coor(1, 8) = 0.45399;
coor(1, 9) = 0.309017;
coor(1, 10) = 0.156434;
coor(1, 11) = 6.1230300000000001e-17;
coor(1, 12) = -0.156434;
coor(1, 13) = -0.309017;
coor(1, 14) = -0.45399;
coor(1, 15) = -0.587785;
coor(1, 16) = -0.707107;
coor(1, 17) = -0.809017;
coor(1, 18) = -0.891007;
coor(1, 19) = -0.951057;
coor(1, 20) = -0.987688;
coor(1, 21) = -1;
coor(1, 22) = -0.987688;
coor(1, 23) = -0.951057;
coor(1, 24) = -0.891007;
coor(1, 25) = -0.809017;
coor(1, 26) = -0.707107;
coor(1, 27) = -0.587785;
coor(1, 28) = -0.45399;
coor(1, 29) = -0.309017;
coor(1, 30) = -0.156434;
coor(1, 31) = -1.83691e-16;
coor(1, 32) = 0.156434;
coor(1, 33) = 0.309017;
coor(1, 34) = 0.45399;
coor(1, 35) = 0.587785;
coor(1, 36) = 0.707107;
coor(1, 37) = 0.809017;
coor(1, 38) = 0.891007;
coor(1, 39) = 0.951057;
coor(1, 40) = 0.987688;
coor(1, 41) = -0.01;
coor(1, 42) = -0.0207077;
coor(1, 43) = -0.0523627;
coor(1, 44) = -0.103582;
```

```
coor(1, 45) = -0.172126;  
coor(1, 46) = -0.255;  
coor(1, 47) = -0.348582;  
coor(1, 48) = -0.448781;  
coor(1, 49) = -0.551219;  
coor(1, 50) = -0.6514180000000001;  
coor(1, 51) = -0.745;  
coor(1, 52) = -0.827874;  
coor(1, 53) = -0.896418;  
coor(1, 54) = -0.947637;  
coor(1, 55) = -0.9792920000000001;  
coor(1, 56) = -0.99;  
coor(1, 57) = -0.9792920000000001;  
coor(1, 58) = -0.947637;  
coor(1, 59) = -0.896418;  
coor(1, 60) = -0.827874;  
coor(1, 61) = -0.745;  
coor(1, 62) = -0.6514180000000001;  
coor(1, 63) = -0.551219;  
coor(1, 64) = -0.448781;  
coor(1, 65) = -0.348582;  
coor(1, 66) = -0.255;  
coor(1, 67) = -0.172126;  
coor(1, 68) = -0.103582;  
coor(1, 69) = -0.0523627;  
coor(1, 70) = -0.0207077;  
coor(1, 71) = -0.35;  
coor(1, 72) = -0.353278;  
coor(1, 73) = -0.362968;  
coor(1, 74) = -0.378647;  
coor(1, 75) = -0.39963;  
coor(1, 76) = -0.425;  
coor(1, 77) = -0.453647;  
coor(1, 78) = -0.484321;  
coor(1, 79) = -0.515679;  
coor(1, 80) = -0.546353;  
coor(1, 81) = -0.575;  
coor(1, 82) = -0.60037;  
coor(1, 83) = -0.621353;  
coor(1, 84) = -0.637032;  
coor(1, 85) = -0.646722;  
coor(1, 86) = -0.65;  
coor(1, 87) = -0.646722;  
coor(1, 88) = -0.637032;  
coor(1, 89) = -0.621353;  
coor(1, 90) = -0.60037;  
coor(1, 91) = -0.575;  
coor(1, 92) = -0.546353;  
coor(1, 93) = -0.515679;  
coor(1, 94) = -0.484321;  
coor(1, 95) = -0.453647;  
coor(1, 96) = -0.425;  
coor(1, 97) = -0.39963;  
coor(1, 98) = -0.378647;  
coor(1, 99) = -0.362968;  
coor(1, 100) = -0.353278;  
coor(2, 2) = 0.156434;  
coor(2, 3) = 0.309017;  
coor(2, 4) = 0.45399;  
coor(2, 5) = 0.587785;  
coor(2, 6) = 0.707107;  
coor(2, 7) = 0.809017;  
coor(2, 8) = 0.891007;  
coor(2, 9) = 0.951057;  
coor(2, 10) = 0.987688;  
coor(2, 11) = 1;  
coor(2, 12) = 0.987688;  
coor(2, 13) = 0.951057;  
coor(2, 14) = 0.891007;  
coor(2, 15) = 0.809017;
```

```
coor(2, 16) = 0.707107;  
coor(2, 17) = 0.587785;  
coor(2, 18) = 0.45399;  
coor(2, 19) = 0.309017;  
coor(2, 20) = 0.156434;  
coor(2, 21) = 1.22461e-16;  
coor(2, 22) = -0.156434;  
coor(2, 23) = -0.309017;  
coor(2, 24) = -0.45399;  
coor(2, 25) = -0.587785;  
coor(2, 26) = -0.707107;  
coor(2, 27) = -0.809017;  
coor(2, 28) = -0.891007;  
coor(2, 29) = -0.951057;  
coor(2, 30) = -0.987688;  
coor(2, 31) = -1;  
coor(2, 32) = -0.987688;  
coor(2, 33) = -0.951057;  
coor(2, 34) = -0.891007;  
coor(2, 35) = -0.809017;  
coor(2, 36) = -0.707107;  
coor(2, 37) = -0.587785;  
coor(2, 38) = -0.45399;  
coor(2, 39) = -0.309017;  
coor(2, 40) = -0.156434;  
coor(2, 42) = -0.101877;  
coor(2, 43) = -0.199301;  
coor(2, 44) = -0.288015;  
coor(2, 45) = -0.364141;  
coor(2, 46) = -0.424352;  
coor(2, 47) = -0.466018;  
coor(2, 48) = -0.487316;  
coor(2, 49) = -0.487316;  
coor(2, 50) = -0.466018;  
coor(2, 51) = -0.424352;  
coor(2, 52) = -0.364141;  
coor(2, 53) = -0.288015;  
coor(2, 54) = -0.199301;  
coor(2, 55) = -0.101877;  
coor(2, 56) = -6.00057e-17;  
coor(2, 57) = 0.101877;  
coor(2, 58) = 0.199301;  
coor(2, 59) = 0.288015;  
coor(2, 60) = 0.364141;  
coor(2, 61) = 0.424352;  
coor(2, 62) = 0.466018;  
coor(2, 63) = 0.487316;  
coor(2, 64) = 0.487316;  
coor(2, 65) = 0.466018;  
coor(2, 66) = 0.424352;  
coor(2, 67) = 0.364141;  
coor(2, 68) = 0.288015;  
coor(2, 69) = 0.199301;  
coor(2, 70) = 0.101877;  
coor(2, 71) = 0.65;  
coor(2, 72) = 0.6188129999999999;  
coor(2, 73) = 0.58899;  
coor(2, 74) = 0.561832;  
coor(2, 75) = 0.538528;  
coor(2, 76) = 0.520096;  
coor(2, 77) = 0.507342;  
coor(2, 78) = 0.500822;  
coor(2, 79) = 0.500822;  
coor(2, 80) = 0.507342;  
coor(2, 81) = 0.520096;  
coor(2, 82) = 0.538528;  
coor(2, 83) = 0.561832;  
coor(2, 84) = 0.58899;  
coor(2, 85) = 0.6188129999999999;  
coor(2, 86) = 0.65;
```