NAG Toolbox for MATLAB

d06ac

1 Purpose

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] = d06ac(nvb, edge, coor, weight, itrace,
'nvint', nvint, 'nvmax', nvmax, 'nedge', nedge)
```

3 Description

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 – int32 scalar**

The number of vertices in the input boundary mesh.

Constraint: **nvb** ≥ 3.

2: **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 jth boundary edge. **edge**$(3,j)$ is a user-supplied tag for the jth boundary edge and is not used by 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**(**2**,**nvmax**) **– double array**

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

4: **weight**$(*)$ **– double array**

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

5: **itrace – int32 scalar**

The level of trace information required from d06ac.

itrace ≤ 0

No output is generated.

itrace ≥ 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: **nvint – int32 scalar**

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

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

Constraint: **nvint** ≥ 0.

2: **nvmax – int32 scalar**

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

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

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

3: **nedge – int32 scalar**

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

the number of boundary edges in the input mesh.

Constraint: **nedge** ≥ 1.

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** $+$ **nvint** $=$ **nvmax**, no interior vertices will be generated and **nv** $=$ **nvmax**.

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

4: **conn**(**3**,**2** × **nvmax** + **5**) – **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 $j = 1, \ldots,$ **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 **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 $j = 1, \ldots,$ **nedge**,
or **edge**$(1,j) =$ **edge**$(2,j)$, for some $j = 1, \ldots,$ **nedge**,
or if **nvint** > 0, **weight**$(i) \leq 0.0$, for some $i = 1, \ldots,$ **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.

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

```
nvb = int32(120);
edge = zeros(3, 120, '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(1, 101) = 101;
edge(1, 102) = 102;
edge(1, 103) = 103;
edge(1, 104) = 104;
edge(1, 105) = 105;
edge(1, 106) = 106;
edge(1, 107) = 107;
edge(1, 108) = 108;
edge(1, 109) = 109;
edge(1, 110) = 110;
edge(1, 111) = 111;
edge(1, 112) = 112;
edge(1, 113) = 113;
edge(1, 114) = 114;
edge(1, 115) = 115;
edge(1, 116) = 116;
edge(1, 117) = 117;
edge(1, 118) = 118;
edge(1, 119) = 119;
edge(1, 120) = 120;
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) = 71;
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) = 41;
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) = 101;
edge(2, 101) = 102;
edge(2, 102) = 103;
edge(2, 103) = 104;
edge(2, 104) = 105;
edge(2, 105) = 106;
edge(2, 106) = 107;
edge(2, 107) = 108;
edge(2, 108) = 109;
edge(2, 109) = 110;
edge(2, 110) = 111;
edge(2, 111) = 112;
edge(2, 112) = 113;
edge(2, 113) = 114;
edge(2, 114) = 115;
edge(2, 115) = 116;
edge(2, 116) = 117;
edge(2, 117) = 118;
edge(2, 118) = 119;
edge(2, 119) = 120;
edge(2, 120) = 81;
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) = 2;
edge(3, 22) = 2;
edge(3, 23) = 2;
edge(3, 24) = 2;
edge(3, 25) = 2;
edge(3, 26) = 2;
edge(3, 27) = 2;
edge(3, 28) = 2;
edge(3, 29) = 2;
edge(3, 30) = 2;
edge(3, 31) = 2;
edge(3, 32) = 2;
edge(3, 33) = 2;
edge(3, 34) = 2;
edge(3, 35) = 2;
edge(3, 36) = 2;
edge(3, 37) = 2;
edge(3, 38) = 2;
edge(3, 39) = 2;
edge(3, 40) = 2;
edge(3, 41) = 3;
edge(3, 42) = 3;
```

```
edge(3, 43) = 3;
edge(3, 44) = 3;
edge(3, 45) = 3;
edge(3, 46) = 3;
edge(3, 47) = 3;
edge(3, 48) = 3;
edge(3, 49) = 3;
edge(3, 50) = 3;
edge(3, 51) = 3;
edge(3, 52) = 3;
edge(3, 53) = 3;
edge(3, 54) = 3;
edge(3, 55) = 3;
edge(3, 56) = 3;
edge(3, 57) = 3;
edge(3, 58) = 3;
edge(3, 59) = 3;
edge(3, 60) = 3;
edge(3, 61) = 3;
edge(3, 62) = 3;
edge(3, 63) = 3;
edge(3, 64) = 3;
edge(3, 65) = 3;
edge(3, 66) = 3;
edge(3, 67) = 3;
edge(3, 68) = 3;
edge(3, 69) = 3;
edge(3, 70) = 3;
edge(3, 71) = 3;
edge(3, 72) = 3;
edge(3, 73) = 3;
edge(3, 74) = 3;
edge(3, 75) = 3;
edge(3, 76) = 3;
edge(3, 77) = 3;
edge(3, 78) = 3;
edge(3, 79) = 3;
edge(3, 80) = 3;
edge(3, 81) = 4;
edge(3, 82) = 4;
edge(3, 83) = 4;
edge(3, 84) = 4;
edge(3, 85) = 4;
edge(3, 86) = 4;
edge(3, 87) = 4;
edge(3, 88) = 4;
edge(3, 89) = 4;
edge(3, 90) = 4;
edge(3, 91) = 4;
edge(3, 92) = 4;
edge(3, 93) = 4;
edge(3, 94) = 4;
edge(3, 95) = 4;
edge(3, 96) = 4;
edge(3, 97) = 4;
edge(3, 98) = 4;
edge(3, 99) = 4;
edge(3, 100) = 4;
edge(3, 101) = 5;
edge(3, 102) = 5;
edge(3, 103) = 5;
edge(3, 104) = 5;
edge(3, 105) = 5;
edge(3, 106) = 5;
edge(3, 107) = 5;
edge(3, 108) = 5;
edge(3, 109) = 5;
edge(3, 110) = 5;
edge(3, 111) = 5;
edge(3, 112) = 5;
```

```
edge(3, 113) = 5;
edge(3, 114) = 5;
edge(3, 115) = 5;
edge(3, 116) = 5;
edge(3, 117) = 5;
edge(3, 118) = 5;
edge(3, 119) = 5;
edge(3, 120) = 5;
coor = zeros(2, 2000);
coor(1, 2) = 0.05;
coor(1, 3) = 0.1;
coor(1, 4) = 0.15;
coor(1, 5) = 0.2;
coor(1, 6) = 0.25;
coor(1, 7) = 0.3;
coor(1, 8) = 0.35;
coor(1, 9) = 0.4;
coor(1, 10) = 0.45;
coor(1, 11) = 0.5;
coor(1, 12) = 0.55;
coor(1, 13) = 0.6;
coor(1, 14) = 0.65;
coor(1, 15) = 0.7;
coor(1, 16) = 0.75;
coor(1, 17) = 0.8;
coor(1, 18) = 0.85;
coor(1, 19) = 0.9;
coor(1, 20) = 0.95;
coor(1, 21) = 1;
coor(1, 22) = 0.95;
coor(1, 23) = 0.9;
coor(1, 24) = 0.85;
coor(1, 25) = 0.8;
coor(1, 26) = 0.75;
coor(1, 27) = 0.7;
coor(1, 28) = 0.65;
coor(1, 29) = 0.6;
coor(1, 30) = 0.55;
coor(1, 31) = 0.5;
coor(1, 32) = 0.45;
coor(1, 33) = 0.4;
coor(1, 34) = 0.35;
coor(1, 35) = 0.3;
coor(1, 36) = 0.25;
coor(1, 37) = 0.2;
coor(1, 38) = 0.15;
coor(1, 39) = 0.1;
coor(1, 40) = 0.05;
coor(1, 41) = -3;
coor(1, 42) = -2.923947143554687;
coor(1, 43) = -2.731404876708984;
coor(1, 44) = -2.452036285400391;
coor(1, 45) = -2.099986267089844;
coor(1, 46) = -1.681983947753906;
coor(1, 47) = -1.199986267089844;
coor(1, 48) = -0.6520362854003907;
coor(1, 49) = -0.03140487670898433;
coor(1, 50) = 0.6760528564453125;
coor(1, 51) = 1.5;
coor(1, 52) = 2.323947143554688;
coor(1, 53) = 3.031404876708985;
coor(1, 54) = 3.65203628540039;
coor(1, 55) = 4.199986267089844;
coor(1, 56) = 4.681983947753906;
coor(1, 57) = 5.099986267089844;
coor(1, 58) = 5.452036285400391;
coor(1, 59) = 5.731404876708984;
coor(1, 60) = 5.923947143554687;
coor(1, 61) = 6;
coor(1, 62) = 5.923947143554687;
```

```
coor(1, 63) = 5.731404876708984;
coor(1, 64) = 5.452036285400391;
coor(1, 65) = 5.099986267089844;
coor(1, 66) = 4.681983947753906;
coor(1, 67) = 4.199986267089844;
coor(1, 68) = 3.652036285400391;
coor(1, 69) = 3.031404876708985;
coor(1, 70) = 2.323947143554688;
coor(1, 71) = 1.5;
coor(1, 72) = 0.6760528564453125;
coor(1, 73) = -0.03140487670898433;
coor(1, 74) = -0.6520362854003907;
coor(1, 75) = -1.199986267089844;
coor(1, 76) = -1.681983947753906;
coor(1, 77) = -2.099986267089844;
coor(1, 78) = -2.452036285400391;
coor(1, 79) = -2.731404876708984;
coor(1, 80) = -2.923947143554687;
coor(1, 81) = 0.8;
coor(1, 82) = 0.8500000000000001;
coor(1, 83) = 0.9;
coor(1, 84) = 0.9500000000000001;
coor(1, 85) = 1;
coor(1, 86) = 1.05;
coor(1, 87) = 1.1;
coor(1, 88) = 1.15;
coor(1, 89) = 1.2;
coor(1, 90) = 1.25;
coor(1, 91) = 1.3;
coor(1, 92) = 1.35;
coor(1, 93) = 1.4;
coor(1, 94) = 1.45;
coor(1, 95) = 1.5;
coor(1, 96) = 1.55;
coor(1, 97) = 1.6;
coor(1, 98) = 1.65;
coor(1, 99) = 1.7;
coor(1, 100) = 1.75;
coor(1, 101) = 1.8;
coor(1, 102) = 1.75;
coor(1, 103) = 1.7;
coor(1, 104) = 1.65;
coor(1, 105) = 1.6;
coor(1, 106) = 1.55;
coor(1, 107) = 1.5;
coor(1, 108) = 1.45;
coor(1, 109) = 1.4;
coor(1, 110) = 1.35;
coor(1, 111) = 1.3;
coor(1, 112) = 1.25;
coor(1, 113) = 1.2;
coor(1, 114) = 1.15;
coor(1, 115) = 1.1;
coor(1, 116) = 1.05;
coor(1, 117) = 1;
coor(1, 118) = 0.9500000000000001;
coor(1, 119) = 0.9;
coor(1, 120) = 0.8500000000000001;
coor(1, 121) = 1.238095238095238;
coor(1, 122) = 1.476190476190476;
coor(1, 123) = 1.714285714285714;
coor(1, 124) = 1.952380952380952;
coor(1, 125) = 2.190476190476191;
coor(1, 126) = 2.428571428571428;
coor(1, 127) = 2.666666666666667;
coor(1, 128) = 2.904761904761905;
coor(1, 129) = 3.142857142857143;
coor(1, 130) = 3.380952380952381;
coor(1, 131) = 3.619047619047619;
coor(1, 132) = 3.857142857142857;
```

```
coor(1, 133) = 4.095238095238095;
coor(1, 134) = 4.333333333333333;
coor(1, 135) = 4.571428571428571;
coor(1, 136) = 4.809523809523809;
coor(1, 137) = 5.047619047619047;
coor(1, 138) = 5.285714285714286;
coor(1, 139) = 5.523809523809524;
coor(1, 140) = 5.761904761904762;
coor(1, 141) = 1.99952380952381;
coor(1, 142) = 2.199047619047619;
coor(1, 143) = 2.398571428571429;
coor(1, 144) = 2.598095238095238;
coor(1, 145) = 2.797619047619048;
coor(1, 146) = 2.997142857142857;
coor(1, 147) = 3.196666666666667;
coor(1, 148) = 3.396190476190476;
coor(1, 149) = 3.595714285714286;
coor(1, 150) = 3.795238095238096;
coor(1, 151) = 3.994761904761905;
coor(1, 152) = 4.194285714285715;
coor(1, 153) = 4.393809523809524;
coor(1, 154) = 4.593333333333334;
coor(1, 155) = 4.792857142857144;
coor(1, 156) = 4.992380952380953;
coor(1, 157) = 5.191904761904762;
coor(1, 158) = 5.391428571428571;
coor(1, 159) = 5.590952380952381;
coor(1, 160) = 5.790476190476191;
coor(2, 2) = 0.035369873046875;
coor(2, 3) = 0.04656219482421875;
coor(2, 4) = 0.0531158447265625;
coor(2, 5) = 0.05696868896484375;
coor(2, 6) = 0.0589447021484375;
coor(2, 7) = 0.05948638916015625;
coor(2, 8) = 0.05889129638671875;
coor(2, 9) = 0.05738067626953125;
coor(2, 10) = 0.05510711669921875;
coor(2, 11) = 0.05219268798828125;
coor(2, 12) = 0.04872894287109375;
coor(2, 13) = 0.04479217529296875;
coor(2, 14) = 0.040435791015625;
coor(2, 15) = 0.03571319580078125;
coor(2, 16) = 0.03063201904296875;
coor(2, 17) = 0.02521514892578125;
coor(2, 18) = 0.0194549560546875;
coor(2, 19) = 0.0133514404296875;
coor(2, 20) = 0.0068817138671875;
coor(2, 22) = -0.0068817138671875;
coor(2, 23) = -0.0133514404296875;
coor(2, 24) = -0.0194549560546875;
coor(2, 25) = -0.02521514892578125;
coor(2, 26) = -0.03063201904296875;
coor(2, 27) = -0.03571319580078125;
coor(2, 28) = -0.040435791015625;
coor(2, 29) = -0.04479217529296875;
coor(2, 30) = -0.04872894287109375;
coor(2, 31) = -0.05219268798828125;
coor(2, 32) = -0.05510711669921875;
coor(2, 33) = -0.05738067626953125;
coor(2, 34) = -0.05889129638671875;
coor(2, 35) = -0.05948638916015625;
coor(2, 36) = -0.0589447021484375;
coor(2, 37) = -0.05696868896484375;
coor(2, 38) = -0.0531158447265625;
coor(2, 39) = -0.04656219482421875;
coor(2, 40) = -0.035369873046875;
coor(2, 42) = -0.8239471435546877;
coor(2, 43) = -1.531404876708984;
coor(2, 44) = -2.152036285400391;
coor(2, 45) = -2.699986267089844;
```

```
coor(2, 46) = -3.181983947753906;
coor(2, 47) = -3.599986267089844;
coor(2, 48) = -3.952036285400391;
coor(2, 49) = -4.231404876708984;
coor(2, 50) = -4.423947143554687;
coor(2, 51) = -4.5;
coor(2, 52) = -4.423947143554687;
coor(2, 53) = -4.231404876708984;
coor(2, 54) = -3.952036285400391;
coor(2, 55) = -3.599986267089844;
coor(2, 56) = -3.181983947753906;
coor(2, 57) = -2.699986267089844;
coor(2, 58) = -2.152036285400391;
coor(2, 59) = -1.531404876708984;
coor(2, 60) = -0.8239471435546877;
coor(2, 62) = 0.8239471435546877;
coor(2, 63) = 1.531404876708984;
coor(2, 64) = 2.152036285400391;
coor(2, 65) = 2.699986267089844;
coor(2, 66) = 3.181983947753906;
coor(2, 67) = 3.599986267089844;
coor(2, 68) = 3.952036285400391;
coor(2, 69) = 4.231404876708984;
coor(2, 70) = 4.423947143554687;
coor(2, 71) = 4.5;
coor(2, 72) = 4.423947143554687;
coor(2, 73) = 4.231404876708984;
coor(2, 74) = 3.952036285400391;
coor(2, 75) = 3.599986267089844;
coor(2, 76) = 3.181983947753906;
coor(2, 77) = 2.699986267089844;
coor(2, 78) = 2.152036285400391;
coor(2, 79) = 1.531404876708984;
coor(2, 80) = 0.8239471435546877;
coor(2, 81) = -0.3;
coor(2, 82) = -0.264630126953125;
coor(2, 83) = -0.2534378051757812;
coor(2, 84) = -0.2468841552734375;
coor(2, 85) = -0.2430313110351562;
coor(2, 86) = -0.2410552978515625;
coor(2, 87) = -0.2405136108398437;
coor(2, 88) = -0.2411087036132812;
coor(2, 89) = -0.2426193237304687;
coor(2, 90) = -0.2448928833007812;
coor(2, 91) = -0.2478073120117187;
coor(2, 92) = -0.2512710571289062;
coor(2, 93) = -0.2552078247070312;
coor(2, 94) = -0.259564208984375;
coor(2, 95) = -0.2642868041992187;
coor(2, 96) = -0.2693679809570312;
coor(2, 97) = -0.2747848510742187;
coor(2, 98) = -0.2805450439453125;
coor(2, 99) = -0.2866485595703125;
coor(2, 100) = -0.2931182861328125;
coor(2, 101) = -0.3;
coor(2, 102) = -0.3068817138671875;
coor(2, 103) = -0.3133514404296875;
coor(2, 104) = -0.3194549560546875;
coor(2, 105) = -0.3252151489257813;
coor(2, 106) = -0.3306320190429688;
coor(2, 107) = -0.3357131958007813;
coor(2, 108) = -0.340435791015625;
coor(2, 109) = -0.3447921752929688;
coor(2, 110) = -0.3487289428710938;
coor(2, 111) = -0.3521926879882813;
coor(2, 112) = -0.3551071166992188;
coor(2, 113) = -0.3573806762695313;
coor(2, 114) = -0.3588912963867188;
coor(2, 115) = -0.3594863891601563;
coor(2, 116) = -0.3589447021484375;
```

```
coor(2, 117) = -0.3569686889648438;
coor(2, 118) = -0.3531158447265625;
coor(2, 119) = -0.3465621948242188;
coor(2, 120) = -0.335369873046875;
coor(2, 141) = -0.3;
coor(2, 142) = -0.3;
coor(2, 143) = -0.3;
coor(2, 144) = -0.3;
coor(2, 145) = -0.3;
coor(2, 146) = -0.3;
coor(2, 147) = -0.3;
coor(2, 148) = -0.3;
coor(2, 149) = -0.3;
coor(2, 150) = -0.3;
coor(2, 151) = -0.3;
coor(2, 152) = -0.3;
coor(2, 153) = -0.3;
coor(2, 154) = -0.3;
coor(2, 155) = -0.3;
coor(2, 156) = -0.3;
coor(2, 157) = -0.3;
coor(2, 158) = -0.3;
coor(2, 159) = -0.3;
coor(2, 160) = -0.3;
weight = [0.05;
     0.05;
     0.05;
     0.05;
     0.05;
     0.05;
     0.05;
     0.05;
     0.05;
     0.05;
     0.05;
     0.05;
     0.05;
     0.05;
     0.05;
     0.05;
     0.05;
     0.05;
     0.05;
     0.05;
     0.05;
     0.05;
     0.05;
     0.05;
     0.05;
     0.05;
     0.05;
     0.05;
     0.05;
     0.05;
     0.05;
     0.05;
     0.05;
     0.05];
itrace = int32(0);
[nv, nelt, coorOut, conn, ifail] = d06ac(nvb, edge, coor, weight, itrace)
```

```
nv =
        1892
nelt =
```

```
        3666
coorOut =
     array elided
conn =
     array elided
ifail =
           0
```