NAG Toolbox for MATLAB

d06ba

1 Purpose

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

2 Syntax

```
[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, 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 d06aa (a simple incremental method) or d06ab (Delaunay–Voronoi method) or 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(2,nlines) – double array**

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

2: **lined(4,nlines) – int32 array**

The description of the lines that define the boundary domain. The line i, for $i = 1,\dots,m$, 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 co-ordinates 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 co-ordinates 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 the user-supplied real function **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 co-ordinates are stored in **coorch**$(:, j)$,**crus**$(:, p)$,
crus$(:, p + 1)$,. . .,**crus**$(:, p + r - 3)$, **coorch**$(:, k)$, where
$r =$ **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 the user-supplied function (lines with **lined**$(4, i) > 0$, for $i = 1, \ldots,$ **nlines**) the two end points (**lined**$(2, i)$ and **lined**$(3, i)$) lie on the curve defined by index **lined**$(4, i)$ in the user-supplied real function **fbnd**, i.e.,

$$\mathbf{fbnd}(\mathbf{lined}(4, i), \mathbf{coorch}(1, \mathbf{lined}2i)), \mathbf{coorch}(2, \mathbf{lined}2i)), \mathbf{user}, \mathbf{user}) = 0;$$

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

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

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

or

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

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

3: **fbnd – string containing name of m-file**

fbnd must be supplied by you 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 d06ba and **fbnd** may be the string `'d06bad'`. **d06bad** is included in the NAG Fortran Library.

Its specification is:

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

Input Parameters

1: **ii – int32 scalar**

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

2: **x – double scalar**
3: **y – double scalar**

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

4: **user – Any MATLAB object**

fbnd is called from d06ba with **user** as supplied to d06ba

Output Parameters

1: **result – double scalar**

The result of the function.

2: **user – Any MATLAB object**

fbnd is called from d06ba with **user** as supplied to d06ba

4: **crus(2,sdcrus) – double array**

The co-ordinates 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**$(1, k)$, $k = p, p + 1, \ldots, p +$ **lined**$(1, i) - 3$ must contain the x co-ordinate of the consecutive intermediate points for this line. Similarly **crus**$(2, k)$, $k = p, p + 1, \ldots, p +$ **lined**$(1, i) - 3$ must contain the corresponding y co-ordinate.

5: **rate(nlines) – double array**

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

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

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

6: **nlcomp(ncomp) – int32 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 $k = 1, \ldots, n$.

Constraints:

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

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

7: **lcomp(nlines) – int32 array**

lcomp$(l1:, l2)$, where $l2 = \sum_{i=1}^{k} |\mathbf{nlcomp}(i)|$ and $l1 = l2 + 1 - |\mathbf{nlcomp}(k)|$ is the list of line numbers for the kth components of the boundary, for $k = 1, \ldots,$ **ncomp**.

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

8: **nvmax – int32 scalar**

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

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

9: **nedmx – int32 scalar**

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

Constraint: **nedmx** ≥ 1.

10: **itrace – int32 scalar**

The level of trace information required from d06ba.

itrace ≤ 0

No output is generated.

itrace $= 1$

Output from the boundary mesh generator is printed on the current advisory message unit (see x04ab). This output contains the input information of each line and each connected component of the boundary.

itrace > 1

The output is similar to that produced when **itrace** $= 1$, but the co-ordinates of the generated vertices on the boundary are also output.

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 d06aa, d06ab or d06ac.

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

5.2 Optional Input Parameters

1: **nlines – int32 scalar**

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 – int32 scalar**

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

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

3: **ncomp – int32 scalar**

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

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

Constraint: **ncomp** ≥ 1.

4: **user – Any MATLAB object**

user is not used by 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 Input Parameters Omitted from the MATLAB Interface

rwork, lrwork, iwork, liwork

5.4 Output Parameters

1: **nvb – int32 scalar**

The total number of boundary mesh vertices generated.

2: **coor(2,nvmax) – double array**

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

3: **nedge – int32 scalar**

The total number of boundary edges in the boundary mesh.

4: **edge(3,nedmx) – int32 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 – Any MATLAB object**

user is not used by 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**.

6: **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, **nlines** < 1;
or **nvmax** $<$ **nlines**;
or **nedmx** < 1;
or **ncomp** < 1;
or **lrwork** $< 2 \times ($**nlines** $+$ **sdcrus**$) + 2 \times \max_{i=1,\dots,m}\{$**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 $i = 1, \dots,$ **nlines** with **lined**$(4, i) \geq 0$;
or **lined**$(1, i) < 2$ for some $i = 1, \dots,$ **nlines**;

or **lined**$(2,i) < 1$ or **lined**$(2,i) >$ **nlines** for some $i = 1,\ldots,$ **nlines**;
or **lined**$(3,i) < 1$ or **lined**$(3,i) >$ **nlines** for some $i = 1,\ldots,$ **nlines**;
or **lined**$(2,i) =$ **lined**$(3,i)$ for some $i = 1,\ldots,$ **nlines**;
or **nlcomp**$(k) = 0$, or $|$**nlcomp**$(k)| >$ **nlines** for a $k = 1,\ldots,$ **ncomp**;
or $\sum_{k=1}^{n} |\mathbf{nlcomp}(k)| \neq \mathbf{nlines}$;
or **lcomp** does not represent a valid permutation of the integers in $[1,$ **nlines**$]$;
or one of the end points for a line i described by the user-supplied function (lines with **lined**$(4,i) > 0$, for $i = 1,\ldots,$ **nlines**) does not belong to the corresponding curve in the user-supplied real function **fbnd**;
or the intermediate points for the lines described as polygonal arcs (lines with **lined**$(i) < 0$, for $i = 1,\ldots,$ **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 parameter **lined**) and each connected component of the boundary (the arguments **nlcomp**, and **lcomp**, as well as the co-ordinates of the characteristic points. Setting **itrace** > 0 may provide more details.

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 component 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, user-supplied real function **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

```
d06ba_fbnd.m
```

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

      result = 0.d0;
      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.d0;
      elseif (i == 2)
        % line 24, 27, 33 and 38 are a circle centred in (x0,y0)
        % with radius sqrt(radius2)
         x0 = 20.5d0;
         y0 = 4.d0;
         radius2 = 4.25d0;
         result = (x-x0)^2 + (y-y0)^2 - radius2;
      elseif (i == 3)
         x0 = 17.d0;
         y0 = 8.5d0;
         radius2 = 5.d0;
         result = (x-x0)^2 + (y-y0)^2 - radius2;
      elseif (i == 4)
         x0 = 17.d0;
         y0 = 8.5d0;
         radius2 = 5.d0;
         result = (x-x0)^2 + (y-y0)^2 - radius2;
      elseif (i == 5)
         x0 = 19.5d0;
         y0 = 4.d0;
         radius2 = 1.25d0;
         result = (x-x0)^2 + (y-y0)^2 - radius2;
      end
```

```
coorch = zeros(2, 40);
coorch(1, 1) = 9.5;
coorch(1, 2) = 33;
coorch(1, 3) = 9.5;
coorch(1, 4) = -14;
coorch(1, 5) = -4;
coorch(1, 6) = -2;
coorch(1, 7) = 2;
coorch(1, 8) = 4;
coorch(1, 9) = 2;
coorch(1, 10) = -2;
coorch(1, 11) = -4;
coorch(1, 12) = -2;
coorch(1, 13) = 2;
coorch(1, 14) = 4;
coorch(1, 15) = 7;
coorch(1, 16) = 9;
coorch(1, 17) = 13;
coorch(1, 18) = 16;
coorch(1, 19) = 9;
coorch(1, 20) = 12;
coorch(1, 21) = 7;
coorch(1, 22) = 10;
coorch(1, 23) = 18;
coorch(1, 24) = 21;
coorch(1, 25) = 17;
coorch(1, 26) = 20;
coorch(1, 27) = 16;
coorch(1, 28) = 20;
coorch(1, 29) = 15.5;
coorch(1, 30) = 16;
coorch(1, 31) = 18;
coorch(1, 32) = 21;
coorch(1, 33) = 16;
coorch(1, 34) = 18;
coorch(1, 35) = 18.5811;
coorch(1, 36) = 21;
coorch(1, 37) = 17;
coorch(1, 38) = 20;
coorch(1, 39) = 20.5;
coorch(1, 40) = 23;
coorch(2, 1) = -1;
coorch(2, 2) = 7.5;
coorch(2, 3) = 16;
coorch(2, 4) = 7.5;
coorch(2, 5) = 3;
coorch(2, 6) = 3;
coorch(2, 7) = 3;
coorch(2, 8) = 3;
coorch(2, 9) = 7;
coorch(2, 10) = 8;
coorch(2, 11) = 12;
coorch(2, 12) = 12;
coorch(2, 13) = 12;
coorch(2, 14) = 12;
coorch(2, 15) = 3;
coorch(2, 16) = 3;
coorch(2, 17) = 3;
coorch(2, 18) = 3;
coorch(2, 19) = 5;
coorch(2, 20) = 5;
coorch(2, 21) = 12;
coorch(2, 22) = 12;
coorch(2, 23) = 2;
coorch(2, 24) = 2;
coorch(2, 25) = 3;
coorch(2, 26) = 3;
coorch(2, 27) = 5;
coorch(2, 28) = 5;
coorch(2, 29) = 6;
coorch(2, 30) = 6;
coorch(2, 31) = 6;
coorch(2, 32) = 6;
coorch(2, 33) = 6.5;
coorch(2, 34) = 6.5;
coorch(2, 35) = 10.0811;
coorch(2, 36) = 10.0811;
coorch(2, 37) = 10.7361;
coorch(2, 38) = 10.7361;
coorch(2, 39) = 12;
coorch(2, 40) = 12;
lined = zeros(4, 40, 'int32');
lined(1, 1) = 15;
lined(1, 2) = 15;
lined(1, 3) = 15;
lined(1, 4) = 15;
lined(1, 5) = 4;
lined(1, 6) = 10;
lined(1, 7) = 10;
lined(1, 8) = 4;
lined(1, 9) = 15;
lined(1, 10) = 4;
lined(1, 11) = 10;
lined(1, 12) = 10;
lined(1, 13) = 4;
lined(1, 14) = 15;
lined(1, 15) = 4;
lined(1, 16) = 7;
lined(1, 17) = 4;
lined(1, 18) = 7;
lined(1, 19) = 4;
lined(1, 20) = 13;
lined(1, 21) = 5;
lined(1, 22) = 13;
lined(1, 23) = 4;
lined(1, 24) = 10;
lined(1, 25) = 4;
lined(1, 26) = 4;
lined(1, 27) = 10;
lined(1, 28) = 4;
lined(1, 29) = 4;
lined(1, 30) = 4;
lined(1, 31) = 4;
lined(1, 32) = 4;
lined(1, 33) = 10;
lined(1, 34) = 4;
lined(1, 35) = 4;
lined(1, 36) = 4;
lined(1, 37) = 4;
lined(1, 38) = 10;
lined(1, 39) = 4;
lined(1, 40) = 4;
lined(2, 1) = 1;
lined(2, 2) = 2;
lined(2, 3) = 3;
lined(2, 4) = 4;
lined(2, 5) = 6;
lined(2, 6) = 10;
lined(2, 7) = 7;
lined(2, 8) = 8;
lined(2, 9) = 14;
lined(2, 10) = 13;
lined(2, 11) = 9;
lined(2, 12) = 12;
lined(2, 13) = 11;
lined(2, 14) = 5;
lined(2, 15) = 16;
lined(2, 16) = 19;
lined(2, 17) = 20;
lined(2, 18) = 17;
lined(2, 19) = 18;
lined(2, 20) = 22;
lined(2, 21) = 21;
lined(2, 22) = 15;
lined(2, 23) = 24;
lined(2, 24) = 24;
lined(2, 25) = 31;
lined(2, 26) = 34;
lined(2, 27) = 34;
lined(2, 28) = 36;
lined(2, 29) = 40;
lined(2, 30) = 39;
lined(2, 31) = 38;
lined(2, 32) = 37;
lined(2, 33) = 37;
lined(2, 34) = 30;
lined(2, 35) = 29;
lined(2, 36) = 27;
lined(2, 37) = 28;
lined(2, 38) = 26;
lined(2, 39) = 25;
lined(2, 40) = 23;
lined(3, 1) = 2;
lined(3, 2) = 3;
lined(3, 3) = 4;
lined(3, 4) = 1;
lined(3, 5) = 5;
lined(3, 6) = 6;
lined(3, 7) = 10;
lined(3, 8) = 7;
lined(3, 9) = 8;
lined(3, 10) = 14;
lined(3, 11) = 13;
lined(3, 12) = 9;
lined(3, 13) = 12;
lined(3, 14) = 11;
lined(3, 15) = 15;
lined(3, 16) = 16;
lined(3, 17) = 19;
lined(3, 18) = 20;
lined(3, 19) = 17;
lined(3, 20) = 18;
lined(3, 21) = 22;
lined(3, 22) = 21;
lined(3, 23) = 23;
lined(3, 24) = 32;
lined(3, 25) = 32;
lined(3, 26) = 31;
lined(3, 27) = 35;
lined(3, 28) = 35;
lined(3, 29) = 36;
lined(3, 30) = 40;
lined(3, 31) = 39;
lined(3, 32) = 38;
lined(3, 33) = 33;
lined(3, 34) = 33;
lined(3, 35) = 30;
lined(3, 36) = 29;
lined(3, 37) = 27;
lined(3, 38) = 28;
lined(3, 39) = 26;
lined(3, 40) = 25;
lined(4, 1) = 1;
lined(4, 2) = 1;
lined(4, 3) = 1;
lined(4, 4) = 1;
lined(4, 5) = -1;
lined(4, 8) = -3;
lined(4, 24) = 2;
lined(4, 27) = 3;
lined(4, 33) = 4;
lined(4, 38) = 5;
coorus = zeros(2, 100);
coorus(1, 1) = -2.6667;
coorus(1, 2) = -3.3333;
coorus(1, 3) = 3.3333;
coorus(1, 4) = 2.6667;
coorus(2, 1) = 3;
coorus(2, 2) = 3;
coorus(2, 3) = 3;
coorus(2, 4) = 3;
rate = [0.95;
     1.05;
     0.95;
     1.05;
     1;
     1;
     1;
     1;
     1;
     1;
     1;
     1;
     1;
     1;
     1;
     1;
     1;
     1;
     1;
     1;
     1;
     1;
     1;
     1;
     1;
     1;
     1;
     1;
     1;
     1;
     1;
     1;
     1;
     1;
     1;
     1;
     1;
     1;
     1;
     1];
nlcomp = [int32(4);
     int32(10);
     int32(8);
     int32(18)];
lcomp = [int32(1);
     int32(2);
     int32(3);
     int32(4);
     int32(14);
     int32(13);
     int32(12);
     int32(11);
     int32(10);
     int32(9);
     int32(8);
     int32(7);
     int32(6);
     int32(5);
     int32(22);
     int32(21);
     int32(20);
     int32(19);
     int32(18);
     int32(17);
     int32(16);
     int32(15);
     int32(30);
     int32(29);
     int32(28);
     int32(27);
     int32(26);
     int32(25);
     int32(24);
     int32(23);
     int32(40);
     int32(39);
     int32(38);
     int32(37);
     int32(36);
     int32(35);
     int32(34);
     int32(33);
     int32(32);
     int32(31)];
nvmax = int32(1000);
nedmx = int32(300);
itrace = int32(-1);
user = [23.5; 8.5; 9.5; 7.5];
[nvb, coor, nedge, edge, user, ifail] = ...
       d06ba(coorch, lined, 'd06ba_fbnd', coorus, rate, nlcomp, lcomp,
nvmax, ...
     nedmx, itrace, 'user', user)
```