NAG C Library Function Document

nag_mesh2d_trans (d06dac)

1 Purpose

nag_mesh2d_trans (d06dac) is a utility which performs an affine transformation of a given mesh.

2 Specification

```
#include <nag.h>
#include <nagd06.h>
```

```
void nag_mesh2d_trans (Integer mode, Integer nv, Integer nedge, Integer nelt,
     Integer ntrans, const Integer itype[], const double trans[], double coori[],
     Integer edgei[], Integer conni[], double cooro[], Integer edgeo[],
     Integer conno[], Integer itrace, const char *outfile, NagError *fail)
```

3 Description

nag_mesh2d_trans (d06dac) generates a mesh (co-ordinates, triangle/vertex connectivities and edge/vertex connectivities) resulting from an affine transformation of a given mesh. This transformation is of the form $Y = A \times X + B$, where

Y, X and B are in $\mathbb{R}^2$, and

A is a real 2 by 2 matrix.

Such a transformation includes a translation, a rotation, a scale reduction or increase, a symmetric transformation with respect to a user-supplied line, a user-supplied analytic transformation, or a composition of several transformations.

This function is partly derived from material in the MODULEF package from INRIA (Institut National de Recherche en Informatique et Automatique).

4 References

None.

5 Arguments

1: **mode** – Integer *Input*

On entry: if **mode** $= 1$, the arguments **coori**, **edgei** and **conni** are overwritten on exit by the output values described in **cooro**, **edgeo** and **conno** respectively. In this case **cooro**, **edgeo** and **conno** are not referenced, and you can save storage space.

If **mode** $\neq 1$, no such aliasing is assumed.

2: **nv** – Integer *Input*

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

Constraint: **nv** ≥ 3.

3: **nedge** – Integer *Input*

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

Constraint: **nedge** ≥ 1.

4: **nelt** – Integer *Input*

On entry: the number of triangles in the input mesh.

Constraint: **nelt** $\leq 2 \times$ **nv** $- 1$.

5: **ntrans** – Integer *Input*

On entry: the number of transformations of the input mesh.

Constraint: **ntrans** ≥ 1.

6: **itype**[**ntrans**] – const Integer *Input*

On entry: **itype**$[i-1]$, for $i = 1, \ldots,$ **ntrans**, indicates the type of each transformation as follows:

itype$[i-1] = 0$

Identity transformation.

itype$[i-1] = 1$

Translation.

itype$[i-1] = 2$

Symmetric transformation with respect to a user-supplied line.

itype$[i-1] = 3$

Rotation.

itype$[i-1] = 4$

Scaling.

itype$[i-1] = 10$

User-supplied analytic transformation.

Note that the transformations are applied in the order described in **itype**.

Constraint: **itype**$[i-1] = 0$, 1, 2, 3, 4 or 10, for $i = 1, 2, \ldots,$ **ntrans**.

7: **trans**[$6 \times$ **ntrans**] – const double *Input*

On entry: the arguments for each transformation. For $i = 1, \ldots,$ **ntrans**, **trans**$[6 \times (i-1)]$ to **trans**$[6 \times (i-1) + 5]$ contain the arguments of the ith transformation:

if **itype**$[i-1] = 0$, then elements **trans**$[6 \times (i-1)]$ to **trans**$[6 \times (i-1) + 5]$ are not referenced;

if **itype**$[i-1] = 1$, then the translation vector is $\vec{u} = \begin{pmatrix} a \\ b \end{pmatrix}$, where $a =$ **trans**$[6 \times (i-1)]$ and $b =$ **trans**$[6 \times (i-1) + 1]$, while elements **trans**$[6 \times (i-1) + 2]$ to **trans**$[6 \times (i-1) + 5]$ are not referenced;

if **itype**$[i-1] = 2$, then the user-supplied line is the curve $\{(x, y) \in \mathbb{R}^2;$ such that $ax + by + c = 0\}$, where $a =$ **trans**$[6 \times (i-1)]$, $b =$ **trans**$[6 \times (i-1) + 1]$ and $c =$ **trans**$[6 \times (i-1) + 2]$, while elements **trans**$[6 \times (i-1) + 3]$ to **trans**$[6 \times (i-1) + 5]$ are not referenced;

if **itype**$[i-1] = 3$, then the centre of the rotation is (x_0, y_0) where $x_0 =$ **trans**$[6 \times (i-1)]$ and $y_0 =$ **trans**$[6 \times (i-1) + 1]$, $\theta =$ **trans**$[6 \times (i-1) + 2]$ is its angle in degrees, while elements **trans**$[6 \times (i-1) + 3]$ to **trans**$[6 \times (i-1) + 5]$ are not referenced;

if **itype**$[i-1] = 4$, then $a =$ **trans**$[6 \times (i-1)]$ is the scaling coefficient in the x-direction, $b =$ **trans**$[6 \times (i-1) + 1]$ is the scaling coefficient in the y-direction, and (x_0, y_0) are the scaling centre co-ordinates, with $x_0 =$ **trans**$[6 \times (i-1) + 2]$ and $y_0 =$ **trans**$[6 \times (i-1) + 3]$; while elements **trans**$[6 \times (i-1) + 4]$ to **trans**$[6 \times (i-1) + 5]$ are not referenced;

if **itype**$[i-1] = 10$, then the user-supplied analytic affine transformation $Y = A \times X + B$ is such that $A = (a_{kl})_{1 \leq k,l \leq 2}$ and $B = (b_k)_{1 \leq k \leq 2}$ where

$a_{kl} =$ **trans**$[6 \times (i-1) + 2 \times (k-1) + l - 1]$, and $b_k =$ **trans**$[6 \times (i-1) + 4 + k - 1]$ with $k, l = 1, 2$.

8: **coori**[**2** × **nv**] – double *Input/Output*

On entry: **coori**$[2 \times (i-1)]$ contains the x co-ordinate of the ith vertex of the input mesh, for $i = 1, \ldots,$ **nv**; while **coori**$[2 \times (i-1) + 1]$ contains the corresponding y co-ordinate.

On exit: if **mode** $= 1$, **coori** is assumed to hold the values of **cooro**.

9: **edgei**[**3** × **nedge**] – Integer *Input/Output*

On entry: the specification of the boundary or interface edges. **edgei**$[3 \times (j-1)]$ and **edgei**$[3 \times (j-1) + 1]$ contain the vertex numbers of the two end points of the jth boundary edge. **edgei**$[3 \times (j-1) + 2]$ is a user-supplied tag for the jth boundary edge. Note that the edge vertices are numbered from 1 to **nv**.

On exit: if **mode** $= 1$, **edgei** holds the output values described in **edgeo**.

Constraint: $1 \leq$ **edgei**$[3 \times (j-1) + i - 1] \leq$ **nv** and **edgei**$[3 \times (j-1)] \neq$ **edgei**$[3 \times (j-1) + 1]$, for $i = 1, 2$ and $j = 1, 2, \ldots,$ **nedge**.

10: **conni**[**3** × **nelt**] – Integer *Input/Output*

On entry: the connectivity of the input mesh between triangles and vertices. For each triangle j, **conni**$[3 \times (j-1) + i - 1]$ gives the indices of its three vertices (in anticlockwise order), for $i = 1, 2, 3$ and $j = 1, \ldots,$ **nelt**. Note that the mesh vertices are numbered from 1 to **nv**.

On exit: if **mode** $= 1$, **conni** holds the output values described in **conno**.

Constraints:

$1 \leq$ **conni**$[3 \times (j-1) + i - 1] \leq$ **nv**;
conni$[3 \times (j-1)] \neq$ **conni**$[3 \times (j-1) + 1]$;
conni$[3 \times (j-1)] \neq$ **conni**$[3 \times (j-1) + 2]$ and
conni$[3 \times (j-1) + 1] \neq$ **conni**$[3 \times (j-1) + 2]$, for $i = 1, 2, 3$ and $j = 1, 2, \ldots,$ **nelt**.

11: **cooro**$[dim]$ – double *Output*

Note: the dimension, *dim*, of the array **cooro** must be at least

2 × **nv** when **mode** $\neq 1$;
1 otherwise.

On exit: **cooro**$[2 \times (i-1)]$ will contain the x co-ordinate of the ith vertex of the transformed mesh, for $i = 1, \ldots,$ **nv**; while **cooro**$[2 \times (i-1) + 1]$ will contain the corresponding y co-ordinate. If **mode** $= 1$ the results are instead overwritten in **coori**.

12: **edgeo**$[dim]$ – Integer *Output*

Note: the dimension, *dim*, of the array **edgeo** must be at least

3 × **nedge** when **mode** $\neq 1$;
1 otherwise.

On exit: the specification of the boundary or interface edges of the transformed mesh. If the number of symmetric transformations is even or zero then
edgeo$[3 \times (j-1) + i - 1] =$ **edgei**$[3 \times (j-1) + i - 1]$, for $i = 1, 2, 3$ and $j = 1, \ldots,$ **nedge**; otherwise **edgeo**$[3 \times (j-1)] =$ **edgei**$[3 \times (j-1) + 1]$,
edgeo$[3 \times (j-1) + 1] =$ **edgei**$[3 \times (j-1)]$ and **edgeo**$[3 \times (j-1) + 2] =$ **edgei**$[3 \times (j-1) + 2]$ for $j = 1, \ldots,$ **nedge**. If **mode** $= 1$ the results are overwritten in **edgei**.

13: **conno**[dim] – Integer *Output*

Note: the dimension, *dim*, of the array **conno** must be at least

$3 \times$ **nelt** when **mode** $\neq 1$;
1 otherwise.

On exit: the connectivity of the transformed mesh between triangles and vertices. If the number of symmetric transformations is even or zero then
conno$[3 \times (j-1)+i-1] =$ **conni**$[3 \times (j-1)+i-1]$, for $i = 1, 2, 3$ and $j = 1, \ldots,$ **nelt**; otherwise **conno**$[3 \times (j-1)] =$ **conni**$[3 \times (j-1)]$, **conno**$[3 \times (j-1)+1] =$ **conni**$[3 \times (j-1)+2]$ and **conno**$[3 \times (j-1)+2] =$ **conni**$[3 \times (j-1)+1]$, for $j = 1, \ldots,$ **nelt**. Note that the mesh vertices are numbered from 1 to **nv**. If **mode** $= 1$ the results are instead overwritten in **conni**.

14: **itrace** – Integer *Input*

On entry: the level of trace information required from nag_mesh2d_trans (d06dac).

itrace ≤ 0

No output is generated.

itrace ≥ 1

Details of each transformation, the matrix A and the vector B of the final transformation, which is the composition of all the **ntrans** transformations, are printed.

15: **outfile** – const 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.

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

The NAG error argument (see Section 2.6 of the Essential Introduction).

6 Error Indicators and Warnings

NE_ALLOC_FAIL

Dynamic memory allocation failed.

NE_BAD_PARAM

On entry, argument ⟨*value*⟩ had an illegal value.

NE_INT

On entry, **nedge** $=$ ⟨*value*⟩.
Constraint: **nedge** ≥ 1.

On entry, **ntrans** $=$ ⟨*value*⟩
Constraint: **ntrans** > 0.

On entry, **nv** $=$ ⟨*value*⟩.
Constraint: **nv** ≥ 3.

NE_INT_2

On entry, **itype**$[i-1]$ is not equal to 0, 1, 2, 3, 4, 10, **itype**$[i-1] =$ ⟨*value*⟩, $i =$ ⟨*value*⟩.

On entry, **nelt** $=$ ⟨*value*⟩, **nv** $=$ ⟨*value*⟩.
Constraint: **nelt** $\leq 2 \times$ **nv** $- 1$.

On entry, the endpoints of the edge j have the same index i: $j =$ ⟨*value*⟩, $i =$ ⟨*value*⟩.

On entry, vertices 1 and 2 of the triangle k have the same index i: $k =$ ⟨*value*⟩, $i =$ ⟨*value*⟩.

On entry, vertices 1 and 3 of the triangle k have the same index i: $k = \langle value \rangle$, $i = \langle value \rangle$.

On entry, vertices 2 and 3 of the triangle k have the same index i: $k = \langle value \rangle$, $i = \langle value \rangle$.

NE_INT_4

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

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

NE_INTERNAL_ERROR

A serious error has occurred in an internal call to an auxiliary function. Check the input mesh especially the connectivities and the details of each transformations.

NE_NOT_CLOSE_FILE

Cannot close file $\langle value \rangle$.

NE_NOT_WRITE_FILE

Cannot open file $\langle value \rangle$ for writing.

7 Accuracy

Not applicable.

8 Further Comments

None.

9 Example

For an example of the use of this utility function, see Section 9 of the document for nag_mesh2d_join (d06dbc).