NAG C Library Function Document

d06dac

nag_mesh2d_trans (d06dac)

1 Purpose

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

2 Specification

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 (coordinates, 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 Parameters

1:  \( \text{mode} \) – Integer

\( \text{Input} \)

\( \text{On entry: if mode = 1 the arguments coori, edgei and conni are overwritten on exit by the output values described in cooro, edgeo and conno respectively. In this case cooro, edgeo and conno are not referenced, and the user can save storage space. If mode \neq 1 no such aliasing is assumed.} \)

2:  \( \text{nv} \) – Integer

\( \text{Input} \)

\( \text{On entry: the total number of vertices in the input mesh.} \)

\( \text{Constraint: } \text{nv} \geq 3. \)

3:  \( \text{nedge} \) – Integer

\( \text{Input} \)

\( \text{On entry: the number of the boundary or interface edges in the input mesh.} \)

\( \text{Constraint: } \text{nedge} \geq 1. \)

4:  \( \text{nelt} \) – Integer

\( \text{Input} \)

\( \text{On entry: the number of triangles in the input mesh.} \)

\( \text{Constraint: } \text{nelt} \leq 2 \times \text{nv} - 1. \)
5: \texttt{ntrans} – Integer

\textit{Input}

\textit{On entry:} the number of transformations of the input mesh.

\textit{Constraint:} \texttt{ntrans} \geq 1.

6: \texttt{itype[ntrans]} – const Integer

\textit{Input}

\textit{On entry:} \texttt{itype[i-1]}, for \( i = 1, \ldots, \text{ntrans}, \) indicates the type of each transformation as follows:

\texttt{itype[i-1]} = 0

Identity transformation.

\texttt{itype[i-1]} = 1

Translation.

\texttt{itype[i-1]} = 2

Symmetric transformation with respect to a user-supplied line.

\texttt{itype[i-1]} = 3

Rotation.

\texttt{itype[i-1]} = 4

Scaling.

\texttt{itype[i-1]} = 10

User-supplied analytic transformation.

Note that the transformations are applied in the order described in \texttt{itype}.

\textit{Constraint:} \texttt{itype[i-1]} = 0, 1, 2, 3, 4 or 10 for \( i = 1, 2, \ldots, \text{ntrans}. \)

7: \texttt{trans[6 \times \text{ntrans}]} – const double

\textit{Input}

\textit{Note:} where \texttt{TRANS(i,j)} appears in this document it refers to the array element \texttt{trans[6 \times (j-1) + i-1]}. We recommend using a \#define to make the same definition in your calling program.

\textit{On entry:} the parameters for each transformation. For \( i = 1, \ldots, \text{ntrans}, \) \texttt{TRANS(1,i)} to \texttt{TRANS(6,i)} contain the parameters of the \( i \)th transformation:

- if \texttt{itype[i-1]} = 0, then elements \texttt{TRANS(1,i)} to \texttt{TRANS(6,i)} are not referenced;
- if \texttt{itype[i-1]} = 1, then the translation vector is \( \vec{u} = \left( a \ b \right) \), where \( a = \text{TRANS(1,i)} \) and \( b = \text{TRANS(2,i)} \), while elements \texttt{TRANS(3,i)} to \texttt{TRANS(6,i)} are not referenced;
- if \texttt{itype[i-1]} = 2, then the user-supplied line is the curve \( \{(x, y) \in \mathbb{R}^2; \text{ such that } ax + by + c = 0 \} \), where \( a = \text{TRANS(1,i)} \), \( b = \text{TRANS(2,i)} \) and \( c = \text{TRANS(3,i)} \), while elements \texttt{TRANS(4,i)} to \texttt{TRANS(6,i)} are not referenced;
- if \texttt{itype[i-1]} = 3, then the centre of the rotation is \( (x_0, y_0) \) where \( x_0 = \text{TRANS(1,i)} \) and \( y_0 = \text{TRANS(2,i)} \), \( \theta = \text{TRANS(3,i)} \) is its angle in degrees, while elements \texttt{TRANS(4,i)} to \texttt{TRANS(6,i)} are not referenced;
- if \texttt{itype[i-1]} = 4, then \( a = \text{TRANS(1,i)} \) is the scaling coefficient in the \( x \)-direction, \( b = \text{TRANS(2,i)} \) is the scaling coefficient in the \( y \)-direction, and \( (x_0, y_0) \) are the scaling centre coordinates, with \( x_0 = \text{TRANS(3,i)} \) and \( y_0 = \text{TRANS(4,i)} \); while elements \texttt{TRANS(5,i)} to \texttt{TRANS(6,i)} are not referenced;
- if \texttt{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 \( a_{kl} = \text{TRANS(2 \times (k-1) + l,i)} \), and \( b_k = \text{TRANS(4 + k,i)} \) with \( k, l = 1, 2 \).
8: \texttt{coori[2 \times \text{nv}]} – double \\
Note: where \texttt{COORI(i,j)} appears in this document it refers to the array element \texttt{coori[2 \times (j - 1) + i - 1]}. We recommend using a \#define to make the same definition in your calling program.

On entry: \texttt{COORI(1,i)} contains the \textit{x}-coordinate of the \textit{i}th vertex of the input mesh, for \textit{i} = 1, \ldots, \text{nv}; while \texttt{COORI(2,i)} contains the corresponding \textit{y}-coordinate.

On exit: if \texttt{mode} = 1, \texttt{coori} is assumed to hold the values of \texttt{cooro}.

9: \texttt{edgei[3 \times \text{nedge}]} – Integer \\
Note: where \texttt{EDGEI(i,j)} appears in this document it refers to the array element \texttt{edgei[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 or interface edges. \texttt{EDGEI(1,j)} and \texttt{EDGEI(2,j)} contain the vertex numbers of the two end-points of the \textit{j}th boundary edge. \texttt{EDGEI(3,j)} is a user-supplied tag for the \textit{j}th boundary edge. Note that the edge vertices are numbered from 1 to \text{nv}.

On exit: if \texttt{mode} = 1, \texttt{edgei} holds the output values described in \texttt{edgeo}.

Constraint: \(1 \leq \text{EDGEI}(i,j) \leq \text{nv}\) and \texttt{EDGEI(1,j) \neq EDGEI(2,j)} for \(i = 1, 2\) and \(j = 1, 2, \ldots, \text{nedge}\).

10: \texttt{conni[3 \times \text{nelt}]} – Integer \\
Note: where \texttt{CONNI(i,j)} appears in this document it refers to the array element \texttt{conni[3 \times (j - 1) + i - 1]}. We recommend using a \#define to make the same definition in your calling program.

On entry: the connectivity of the input mesh between triangles and vertices. For each triangle \textit{j}, \texttt{CONNI(i,j)} gives the indices of its three vertices (in anticlockwise order), for \(i = 1, 2, 3\) and \(j = 1, \ldots, \text{nelt}\). Note that the mesh vertices are numbered from 1 to \text{nv}.

On exit: if \texttt{mode} = 1, \texttt{conni} holds the output values described in \texttt{conno}.

Constraints:
\[
1 \leq \text{CONNI}(i,j) \leq \text{nv}; \\
\text{CONNI}(1,j) \neq \text{CONNI}(2,j); \\
\text{CONNI}(1,j) \neq \text{CONNI}(3,j)\quad \text{and} \quad \text{CONNI}(2,j) \neq \text{CONNI}(3,j)\quad \text{for}\quad i = 1, 2, 3\quad \text{and}\quad j = 1, 2, \ldots, \text{nelt}.
\]

11: \texttt{cooro[\text{dim}]} – double \\
Note: where \texttt{COORO(i,j)} appears in this document it refers to the array element \texttt{cooro[2 \times (j - 1) + i - 1]}. We recommend using a \#define to make the same definition in your calling program.

The dimension, \texttt{dim}, of the array \texttt{cooro} must be at least \text{nv} when \texttt{mode} \neq 1 and at least 1 otherwise.

On exit: \texttt{COORO(1,i)} will contain the \textit{x}-coordinate of the \textit{i}th vertex of the transformed mesh, for \textit{i} = 1, \ldots, \text{nv}; while \texttt{COORO(2,i)} will contain the corresponding \textit{y}-coordinate. If \texttt{mode} = 1 the results are instead overwritten in \texttt{coori}.

12: \texttt{edgeo[\text{dim}]} – Integer \\
Note: where \texttt{EDGEO(i,j)} appears in this document it refers to the array element \texttt{edgeo[3 \times (j - 1) + i - 1]}. We recommend using a \#define to make the same definition in your calling program.

The dimension, \texttt{dim}, of the array \texttt{edgeo} must be at least \text{nedge} when \texttt{mode} \neq 1 and at least 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 $\text{EDGEO}(i, j) = \text{EDGEI}(i, j)$ for $i = 1, 2, 3$ and $j = 1, \ldots, \text{nedge}$; otherwise $\text{EDGEO}(1, j) = \text{EDGEI}(2, j)$, $\text{EDGEO}(2, j) = \text{EDGEI}(1, j)$ and $\text{EDGEO}(3, j) = \text{EDGEI}(3, j)$ for $j = 1, \ldots, \text{nedge}$. If mode = 1 the results are overwritten in edgei.

13:  conno[dim] – Integer

Output

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

The dimension, dim, of the array conno must be at least nelt when mode $\neq 1$ and at least 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 $\text{CONNO}(i, j) = \text{CONNI}(i, j)$ for $i = 1, 2, 3$ and $j = 1, \ldots, \text{nelt}$; otherwise $\text{CONNO}(1, j) = \text{CONNI}(1, j)$, $\text{CONNO}(2, j) = \text{CONNI}(3, j)$ and $\text{CONNO}(3, j) = \text{CONNI}(2, j)$, for $j = 1, \ldots, \text{nelt}$. 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) as follows:

if itrace $\leq 0$, no output is generated;

if itrace $\geq 1$, then 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 – 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 parameter (see the Essential Introduction).

6 Error Indicators and Warnings

NE_INT

On entry, ntrans = (value)
Constraint: ntrans $> 0$.

On entry, nv = (value).
Constraint: nv $\geq 3$.

On entry, nedge = (value).
Constraint: nedge $\geq 1$.

NE_INT_2

On entry, nel = (value), nv = (value).
Constraint: nel $\leq 2 \times \text{nvt} - 1$.

On entry, the endpoints of the edge $j$ have the same index $i$: $j = (value)$, $i = (value)$.

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

On entry, vertices 2 and 3 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 $i$: $k = (value)$, $i = (value)$.

On entry, vertices 1 and 2 of the triangle $k$ have the same index $i$: $k = (value)$, $i = (value)$.
On entry, edge\((i,j)\) < 1 or edge\((i,j)\) > nv\(, \) where edge\((i,j)\) denotes edge\([3 \times (j - 1) + i - 1]\): edge\((i,j)\) = \((\text{value}), i = (\text{value}), j = (\text{value}), \) nv = \((\text{value})\).

On entry, conni\((i,j)\) < 1 or conni\((i,j)\) > nv\(, \) where conni\((i,j)\) denotes conni\([3 \times (j - 1) + i - 1]\): conni\((i,j)\) = \((\text{value}), i = (\text{value}), j = (\text{value}), \) nv = \((\text{value})\).

**NE_INTERNAL_ERROR**

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

**NE_ALLOC_FAIL**

Memory allocation failed.

**NE_BAD_PARAM**

On entry, parameter \((\text{value})\) had an illegal value.

**NE_NOT_WRITE_FILE**

Cannot open file \((\text{value})\) for writing.

**NE_NOT_CLOSE_FILE**

Cannot close file \((\text{value})\).

**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**

None.

9  **Example**

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