NAG Toolbox

nag_mesh_2d_transform_affine (d06da)

1 Purpose

nag_mesh_2d_transform_affine (d06da) is a utility which performs an affine transformation of a given mesh.

2 Syntax

```
[coori, edgei, conni, cooro, edgeo, conno, ifail] = nag_mesh_2d_transform_affine
(itype, trans, coori, edgei, conni, itrace, 'nv', nv, 'nedge', nedge, 'nelt',
nelt, 'ntrans', ntrans)
[coori, edgei, conni, cooro, edgeo, conno, ifail] = d06da(itype, trans, coori,
edgei, conni, itrace, 'nv', nv, 'nedge', nedge, 'nelt', nelt, 'ntrans', ntrans)
```

3 Description

nag_mesh_2d_transform_affine (d06da) 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

5.1 Compulsory Input Parameters

1: **itype**(**ntrans**) – INTEGER array

itype(i), for $i = 1, 2, \ldots,$ **ntrans**, indicates the type of each transformation as follows:

itype(i) $= 0$
Identity transformation.

itype(i) $= 1$
Translation.

itype(i) $= 2$
Symmetric transformation with respect to a user-supplied line.

itype(i) $= 3$
Rotation.

itype(i) $= 4$
Scaling.

itype$(i) = 10$
User-supplied analytic transformation.

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

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

2: **trans**(**6**, **ntrans**) – REAL (KIND=nag_wp) array

The arguments for each transformation. For $i = 1, 2, \ldots,$ **ntrans**, **trans**$(1, i)$ to **trans**$(6, i)$ contain the arguments of the ith transformation.

If **itype**$(i) = 0$, elements **trans**$(1, i)$ to **trans**$(6, i)$ are not referenced.

If **itype**$(i) = 1$, the translation vector is $\vec{u} = \begin{pmatrix} a \\ b \end{pmatrix}$, where $a =$ **trans**$(1, i)$ and $b =$ **trans**$(2, i)$, while elements **trans**$(3, i)$ to **trans**$(6, i)$ are not referenced.

If **itype**$(i) = 2$, the user-supplied line is the curve $\{(x, y) \in \mathbb{R}^2;$ such that $ax + by + c = 0\}$, where $a =$ **trans**$(1, i)$, $b =$ **trans**$(2, i)$ and $c =$ **trans**$(3, i)$, while elements **trans**$(4, i)$ to **trans**$(6, i)$ are not referenced.

If **itype**$(i) = 3$, the centre of the rotation is (x_0, y_0) where $x_0 =$ **trans**$(1, i)$ and $y_0 =$ **trans**$(2, i)$, $\theta =$ **trans**$(3, i)$ is its angle in degrees, while elements **trans**$(4, i)$ to **trans**$(6, i)$ are not referenced.

If **itype**$(i) = 4$, $a =$ **trans**$(1, i)$ is the scaling coefficient in the x-direction, $b =$ **trans**$(2, i)$ is the scaling coefficient in the y-direction, and (x_0, y_0) are the scaling centre coordinates, with $x_0 =$ **trans**$(3, i)$ and $y_0 =$ **trans**$(4, i)$; while elements **trans**$(5, i)$ to **trans**$(6, i)$ are not referenced.

If **itype**$(i) = 10$, 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**$(2 \times (k - 1) + l, i)$, and $b_k =$ **trans**$(4 + k, i)$ with $k, l = 1, 2$.

3: **coori**(**2**, **nv**) – REAL (KIND=nag_wp) array

coori$(1, i)$ contains the x coordinate of the ith vertex of the input mesh, for $i = 1, 2, \ldots,$ **nv**; while **coori**$(2, i)$ contains the corresponding y coordinate.

4: **edgei**(**3**, **nedge**) – INTEGER array

The specification of the boundary or interface edges. **edgei**$(1, j)$ and **edgei**$(2, j)$ contain the vertex numbers of the two end points of the jth boundary edge. **edgei**$(3, j)$ is a user-supplied tag for the jth boundary edge.

Constraint: $1 \leq$ **edgei**$(i, j) \leq$ **nv** and **edgei**$(1, j) \neq$ **edgei**$(2, j)$, for $i = 1, 2$ and $j = 1, 2, \ldots,$ **nedge**.

5: **conni**(**3**, **nelt**) – INTEGER array

The connectivity of the input mesh between triangles and vertices. For each triangle j, **conni**(i, j) gives the indices of its three vertices (in anticlockwise order), for $i = 1, 2, 3$ and $j = 1, 2, \ldots,$ **nelt**.

Constraints:

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

6: **itrace** – INTEGER

The level of trace information required from nag_mesh_2d_transform_affine (d06da).

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 on the current advisory message unit (see nag_file_set_unit_advisory (x04ab)).

5.2 Optional Input Parameters

1: **nv** – INTEGER

Default: the dimension of the array **coori**.

The total number of vertices in the input mesh.

Constraint: **nv** ≥ 3.

2: **nedge** – INTEGER

Default: the dimension of the array **edgei**.

The number of the boundary or interface edges in the input mesh.

Constraint: **nedge** ≥ 1.

3: **nelt** – INTEGER

Default: the dimension of the array **conni**.

The number of triangles in the input mesh.

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

4: **ntrans** – INTEGER

Default: the dimension of the arrays **itype**, **trans**. (An error is raised if these dimensions are not equal.)

The number of transformations of the input mesh.

Constraint: **ntrans** ≥ 1.

5.3 Output Parameters

1: **coori**$(\mathbf{2}, \mathbf{nv})$ – REAL (KIND=nag_wp) array

see Section 9.

2: **edgei**$(\mathbf{3}, \mathbf{nedge})$ – INTEGER array

See Section 9.

3: **conni**$(\mathbf{3}, \mathbf{nelt})$ – INTEGER array

See Section 9.

4: **cooro**$(\mathbf{2}, \mathbf{nv})$ – REAL (KIND=nag_wp) array

cooro$(1, i)$ will contain the x coordinate of the ith vertex of the transformed mesh, for $i = 1, 2, \ldots,$ **nv**; while **cooro**$(2, i)$ will contain the corresponding y coordinate.

5: **edgeo**$(\mathbf{3}, \mathbf{nedge})$ – INTEGER array

The specification of the boundary or interface edges of the transformed mesh. If the number of symmetric transformations is even or zero then**edgeo**$(i, j) =$ **edgei**(i, j), for $i = 1, 2, 3$ and $j = 1, 2, \ldots,$ **nedge**; otherwise **edgeo**$(1, j) =$ **edgei**$(2, j)$, **edgeo**$(2, j) =$ **edgei**$(1, j)$ and **edgeo**$(3, j) =$ **edgei**$(3, j)$, for $j = 1, 2, \ldots,$ **nedge**.

6: **conno**(**3**, **nelt**) – INTEGER array

The connectivity of the transformed mesh between triangles and vertices. If the number of symmetric transformations is even or zero then**conno**$(i,j) =$ **conni**(i,j), for $i = 1,2,3$ and $j = 1,2,\ldots,$ **nelt**; otherwise **conno**$(1,j) =$ **conni**$(1,j)$, **conno**$(2,j) =$ **conni**$(3,j)$ and **conno**$(3,j) =$ **conni**$(2,j)$, for $j = 1,2,\ldots,$ **nelt**.

7: **ifail** – INTEGER

ifail $= 0$ unless the function detects an error (see Section 5).

6 Error Indicators and Warnings

Errors or warnings detected by the function:

ifail $= 1$

On entry, **nv** < 3;
or **nelt** $> 2 \times$ **nv** $- 1$;
or **nedge** < 1;
or **edgei**$(i,j) < 1$ or **edgei**$(i,j) >$ **nv** for some $i = 1,2$ and $j = 1,2,\ldots,$ **nedge**;
or **edgei**$(1,j) =$ **edgei**$(2,j)$ for some $j = 1,2,\ldots,$ **nedge**;
or **conni**$(i,j) < 1$ or **conni**$(i,j) >$ **nv** for some $i = 1,2,3$ and $j = 1,2,\ldots,$ **nelt**;
or **conni**$(1,j) =$ **conni**$(2,j)$ or **conni**$(1,j) =$ **conni**$(3,j)$ or
conni$(2,j) =$ **conni**$(3,j)$ for some $j = 1,2,\ldots,$ **nelt**;
or **ntrans** < 1;
or **itype**$(i) \neq 0$, 1, 2, 3, 4 or 10 for some $i = 1,2,\ldots,$ **ntrans**;
or $lrwork < 12 \times$ **ntrans**.

ifail $= 2$

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

ifail $= -99$

An unexpected error has been triggered by this routine. Please contact NAG.

ifail $= -399$

Your licence key may have expired or may not have been installed correctly.

ifail $= -999$

Dynamic memory allocation failed.

7 Accuracy

Not applicable.

8 Further Comments

You may not wish to save the input mesh (**coori**, **edgei** and **conni**) and could call nag_mesh_2d_transform_affine (d06da) using the same arguments for the input and the output (transformed) mesh.

9 Example

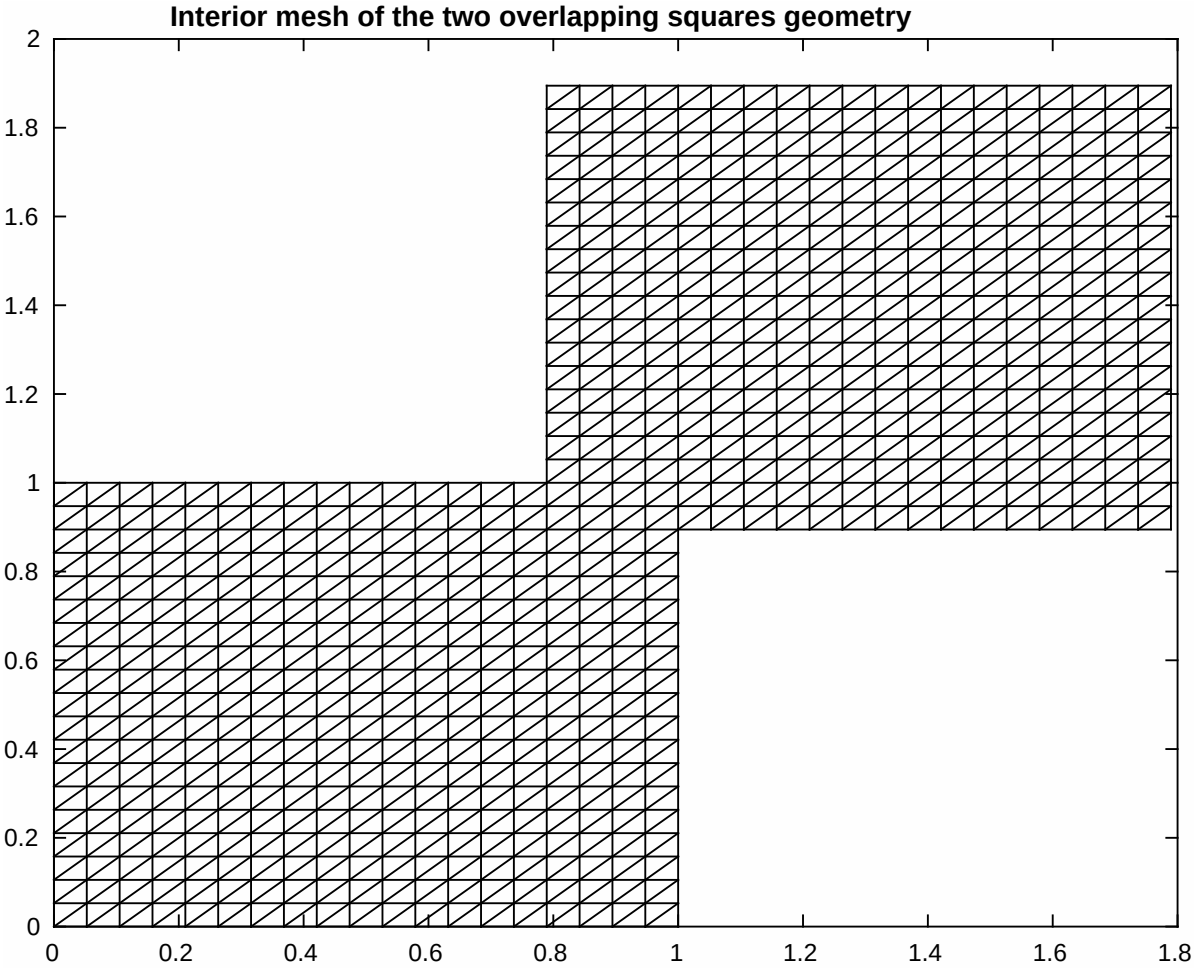
For an example of the use of this utility function, see Section 10 in nag_mesh_2d_join (d06db).

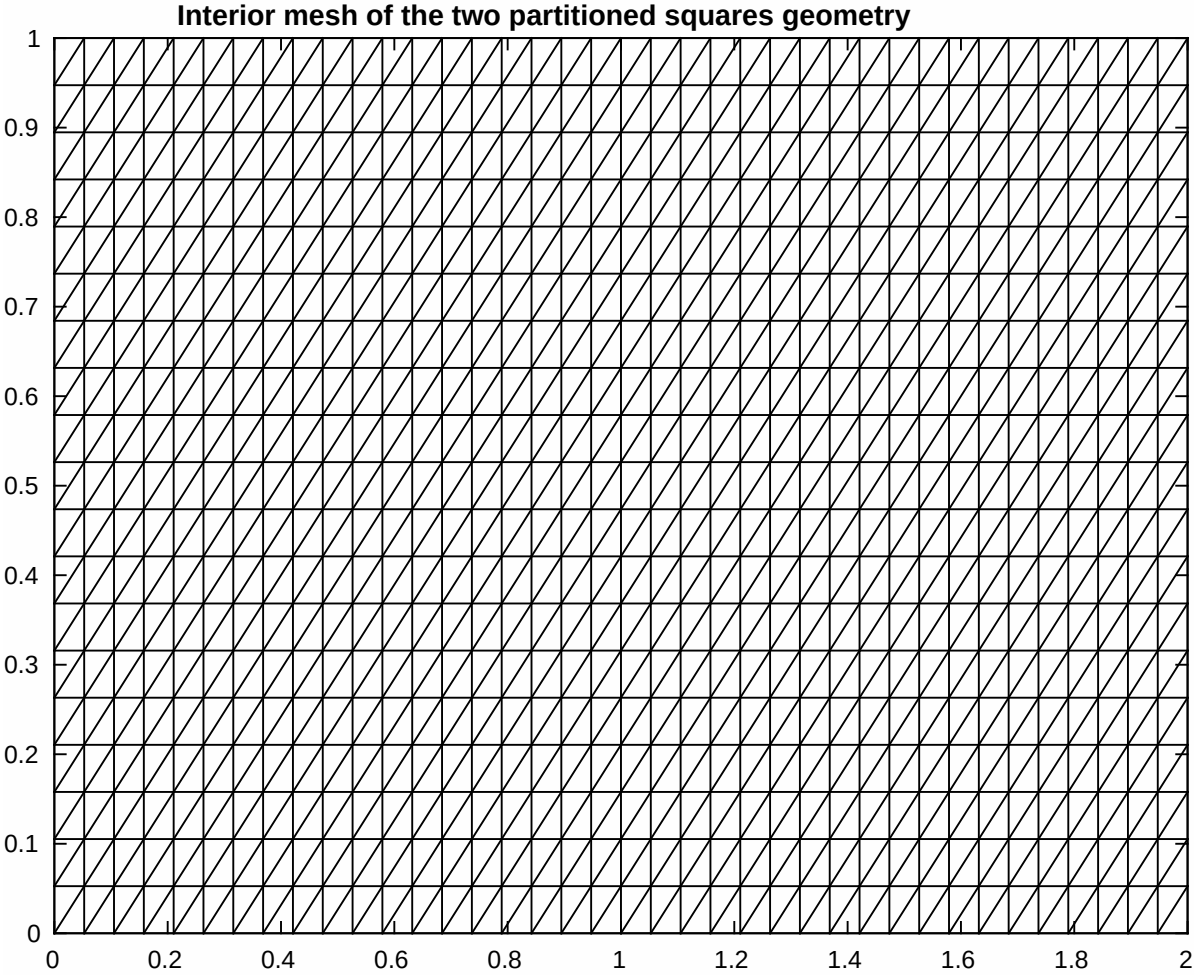
9.1 Program Text

```
     function d06da_example

fprintf('d06da example results\n\n');

coor1 = zeros(2,400);
for j = 1:20
  for i = 1:20
    coor1(1,(j-1)*20+i) = (i-1)/19;
    coor1(2,(j-1)*20+i) = (j-1)/19;
  end
end
edge1 = ones(3, 76, nag_int_name);
edge1(1, 1:76) = nag_int([1:19,20*(1:19),401-(1:19),401-20*(1:19)]);
edge1(2, 1:76) = nag_int([2:19,20*(1:19),401-(1:19),401-20*(1:19),1]);
conn1 = zeros(3, 722, nag_int_name);
ind = -1;
for i=1:379
  if (rem(i, 20) ~= 0)
    ind = ind+2;
    conn1(1, ind)   = nag_int(i);
    conn1(1, ind+1) = nag_int(i);
    conn1(2, ind)   = nag_int(i+1);
    conn1(2, ind+1) = nag_int(i+21);
    conn1(3, ind)   = nag_int(i+21);
    conn1(3, ind+1) = nag_int(i+20);
  end
end
reft1 = ones(722, 1, nag_int_name);
reft2 = reft1;
reft2(:) = nag_int(2);
itype = [nag_int(1)];
itrace = nag_int(1);

% Transform the first domain to obtain an overlapping second domain
trans = [15/19; 17/19; 0; 0; 0; 0];
[coor1, edge1, conn1, coor2, edge2, conn2, ifail] = ...
d06da( ...
       itype, trans, coor1, edge1, conn1, itrace);

% Restitch the meshes
[nv3, nelt3, nedge3, coor3, edge3, conn3, reft3, ifail] = ...
d06db( ...
       coor1, edge1, conn1, reft1, coor2, edge2, conn2, reft2, itrace);

% Plot the result
fig1 = figure;
triplot(transpose(double(conn3(:,1:nelt3))), coor3(1,:), coor3(2,:));
title ('Interior mesh of the two overlapping squares geometry');

% Now consider a partitioned second domain
trans = [1; 0; 0; 0; 0; 0];
[coor1, edge1, conn1, coor2, edge2, conn2, ifail] = ...
d06da( ...
       itype, trans, coor1, edge1, conn1, itrace);

% Restitch the meshes
[nv3, nelt3, nedge3, coor3, edge3, conn3, reft3, ifail] = ...
d06db( ...
       coor1, edge1, conn1, reft1, coor2, edge2, conn2, reft2, itrace);

% Plot the result
fig2 = figure;
triplot(transpose(double(conn3(:,1:nelt3))), coor3(1,:), coor3(2,:));
title ('Interior mesh of the two partitioned squares geometry');
```

9.2 Program Results

```
     d06da example results

 Transformation   1: translation
  translation vector:  0.7895      0.8947

 Final transformation matrix y = A*x + b:
    1.000       0.000         0.7895
    0.000       1.000         0.8947

 Transformation   1: translation
  translation vector:   1.000       0.000

 Final transformation matrix y = A*x + b:
    1.000       0.000          1.000
    0.000       1.000          0.000
```




Interior mesh of the two partitioned squares geometry
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2