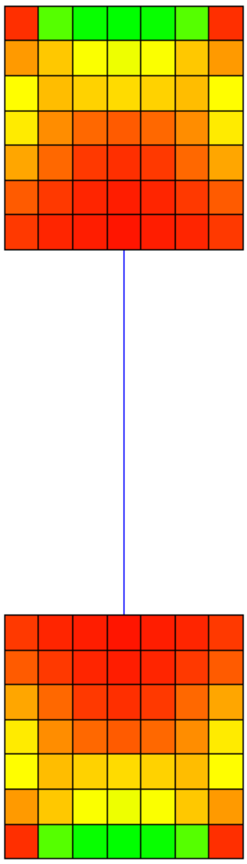
Substituting volume elements with beam elements

in

Code Aster®

A short introduction to using LIAISON_ELEM.

For CAELinux.com, March 2011 – Claus Andersen
Rev. 1.0



1 Table of Contents

1 Introduction and theory

Code Aster® has the ability to connect elements of different dimensions. 0-1-2 and 3D elements can be connected together in all different manners, but in this particular study, we'll connect a 3D element to a 1D element and back to a 3D element again.

In many situations this can drastically reduce the work load on the computer doing the analysis, not to mention post-processing the files after-wards.

In a situation where you're only interested in what happens at the boundaries, there's no need to model and calculate the areas in-between.

The way to go about this, is to describe the area in-between, instead of modeling it.

For this feat, **Code Aster®** provides a powerful command called **LIAISON_ELEM**.

*(Document **[U4.42.01]** explains this command fully)*

To describe an element, **Code Aster®** needs know how many degree of freedom (DX,DY,DZ,DRX,DRY,DRZ) the element has and what the element geometrically looks like.

For this, a model must be assigned to the element and parameters for the geometry must be assigned. Heres an example of how a square pipe is described (Fig. 1.1):

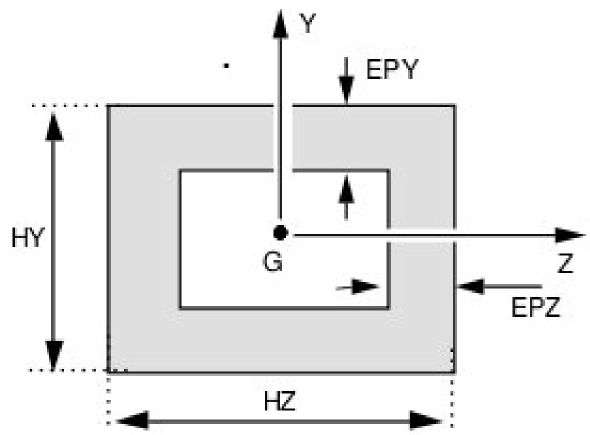


Figure 1.1: parameters describing a cross section with AFFE_CARA_ELEM

In this study we're using a solid rectangle so we only need to assign the parameter **H**, since the sides are of equal length.

This is the result of we'll be working towards in this study (Fig. 1)

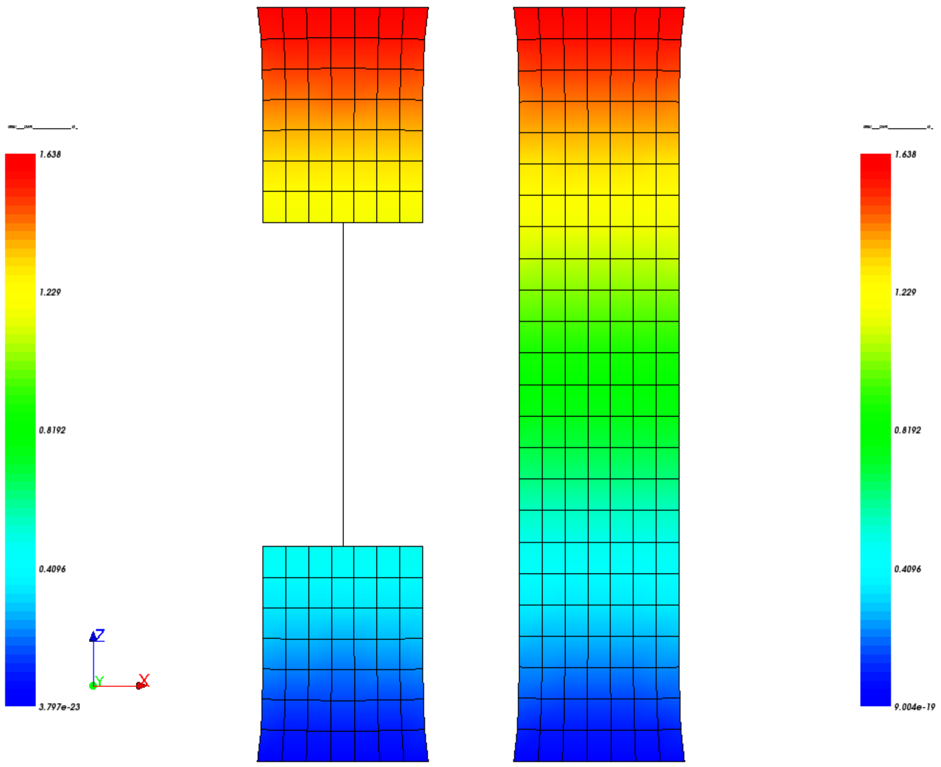


Figure 1:

2 Applying the theory to Code Aster®

2.1 Preparing the object

In this study we'll analyze the boundary conditions of a column, substituting the middle of the column with a 1D element and describing this element to **Code Aster®**, thus saving a lot of computing time. Actually not at all, really. Since the geometry in this situation is so simple, we're talking mere mili-seconds and roughly 1mb of RAM saved - we'll do it anyway just for the heck of it.

From looking at the image of the result, you should have a good idea of how to model the situation.

- Create two boxes of 200x200x200 in the geometry modules of **Salomé®**, enter the mesh module.
- In the mesh module, mesh the two boxes individually with hexahedrons.
- Create a *compound* mesh of the two boxes. Right click the compound mesh and select 'show only'
- Now create two points; one at the center of the top surface of the *bottom* box (if the box is 200x200x200, the coordinates of the first point will be {100,100,200}), and another at the center of the bottom surface of the *top* box.
- Create a node group for each of the two nodes, call the two node groups **N_1** and **N_2** respectively
- Create an *edge* by using the two points you just created. Assign this to an element group and call it **bar**
- The surfaces that are to be connected with the edge just created, are assigned face groups and are called **liaison** and **liaison2**, respectively.
- Bottom face of the *bottom* box is assigned a face group called **hold**, top face of the 'top' box is assigned a face group called **top**

Heres a picture of what it should look like - only the *bottom* box is shown – Fig. 2.2

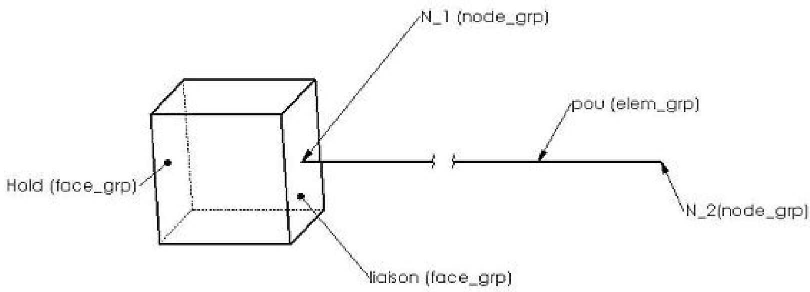


Figure 2.1: Group diagram

You should now have a constellation of groups looking something like this(Fig. 2.2):

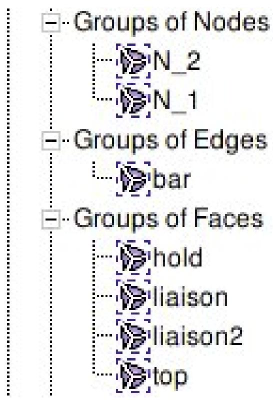


Figure 2.2: Mesh groups

Export the mesh to a *.med* file and we are ready to move on to...

2.2 Applying LIAISON_ELEM in Code Aster®

Walking through the *.comm* file step by step

```
#Define the material, read the mesh
DEBUT();

MA=DEFI_MATERIAU(ELAS=_F(E=210e+03,
                        NU=0.28,),);

MAIL=LIRE_MAILLAGE(FORMAT='MED',
                  INFO_MED=1,);
```

Definition

- Assigning mechanical models to the components; call the models **MODE**
  - Assign a 3D model to everything
  - Assign a beam model with 3D capabilities (rotation and translation) to the element **bar**
    - **POU**tre = beam, **D**roite = straight, **E**uler hypothesis
      - See document **U3.11.01** section **1** for further explanation

```
MODE=AFFE_MODELE(MAILLAGE=MAIL,
                AFFE=(_F(TOUT='OUI',
                         PHENOMENE='MECANIQUE',
                         MODELISATION='3D',),
                      _F(GROUP_MA='bar',
                         PHENOMENE='MECANIQUE',
                         MODELISATION='POU_D_E',),),);
```

Definition

Assign the material **MA** to everything

```
MATE=AFFE_MATERIAU(MAILLAGE=MAIL,
                  AFFE=_F(TOUT='OUI',
                          MATER=MA,),);
```

Definition

- Describing the characteristics of the element **bar**
  - **VERIF (NOUED, MAILLE)**: Verify that both the element and the nodes support the assigned characteristics, otherwise halt the calculation with a fatal error
  - **POUTRE**: Beam
    - Assign to the element **bar**
    - **SECTION**: Cross section is a rectangle of equal length sides
    - **CARA=H**, **VALE=200**: Since the cross section where we connect the beam element to the 3D element is 200x200, **H** must have the same dimensions.

```
cara=AFFE_CARA_ELEM(MODELE=MODE,
                    VERIF=('NOEUD','MAILLE',),
                    POUTRE=_F(GROUP_MA='bar',
                              SECTION='RECTANGLE',
                              CARA='H',
                              VALE=200.0,),);
```

Definition

- Assign loads and boundary conditions, call it **CHAR**
  - **DDL_IMPO:** Impose zero displacements to the face group **hold**, i.e. anchor it down.
  - **LIAISON_ELEM=** Create a relationship between a 3D element and a beam (**3D_POU**tre)
    - Create the relationship between the face group **liaison** and the node group **N_1**
    - Do the same for **liaison2** and **N_2**
  - **LIAISON_UNIF:** Do not let the face group **top** deform AT ALL, i.e. it act as if it was welded to the ceiling
  - **FORCE_FACE**: Self-explanatory

```
CHAR=AFFE_CHAR_MECA(MODELE=MODE,
                   DDL_IMPO=_F(GROUP_MA='hold',
                               DX=0.0,
                               DY=0.0,
                               DZ=0.0,),
                   LIAISON_ELEM=(_F(OPTION='3D_POU',
                                    GROUP_MA_1='liaison',
                                    GROUP_NO_2='N_1',),
                                 _F(OPTION='3D_POU',
                                    GROUP_MA_1='liaison2',
                                    GROUP_NO_2='N_2',),),
                   LIAISON_UNIF=_F(GROUP_MA='top',
                                   DDL=('DX','DY','DZ',),),
                   FORCE_FACE=_F(GROUP_MA='top',
                                 FZ=500.0,),);
```

Definition

- Calculate the solution and call it **RESU**

```
RESU=MECA_STATIQUE(MODELE=MODE,
                  CHAM_MATER=MATE,
                  CARA_ELEM=cara,
                  EXCIT=_F(CHARGE=CHAR,),);
```

Definition

- Calculate the stress at each element and nodes

```
RESU=CALC_ELEM(reuse =RESU,
              MODELE=MODE,
              CHAM_MATER=MATE,
              RESULTAT=RESU,
              OPTION=('SIGM_ELNO_DEPL','EQUI_ELNO_SIGM','EQUI_ELGA
_SIGM',),
              EXCIT=_F(CHARGE=CHAR,),);

RESU=CALC_NO(reuse =RESU,
            RESULTAT=RESU,
            OPTION=('SIGM_NOEU_DEPL','EQUI_NOEU_SIGM',),);
```

Definition

- Write the results to a *.med* file, include displacement, equivalent nodal stress and principal stress at the nodes as a result of the displacement

```
IMPR_RESU(FORMAT='MED',
          UNITE=80,
          RESU=_F(MAILLAGE=MAIL,
                  RESULTAT=RESU,
                  NOM_CHAM=('SIGM_NOEU_DEPL','EQUI_NOEU_SIGM','EQUI
_ELGA_SIGM','DEPL',),),);
FIN();
```

3 Post-processing

To verify that **LIAISON_ELEM** is indeed doing what it is supposed to be doing, a full 3D column is created, meshed and applied identical boundary conditions and load.

3.1 Deformation

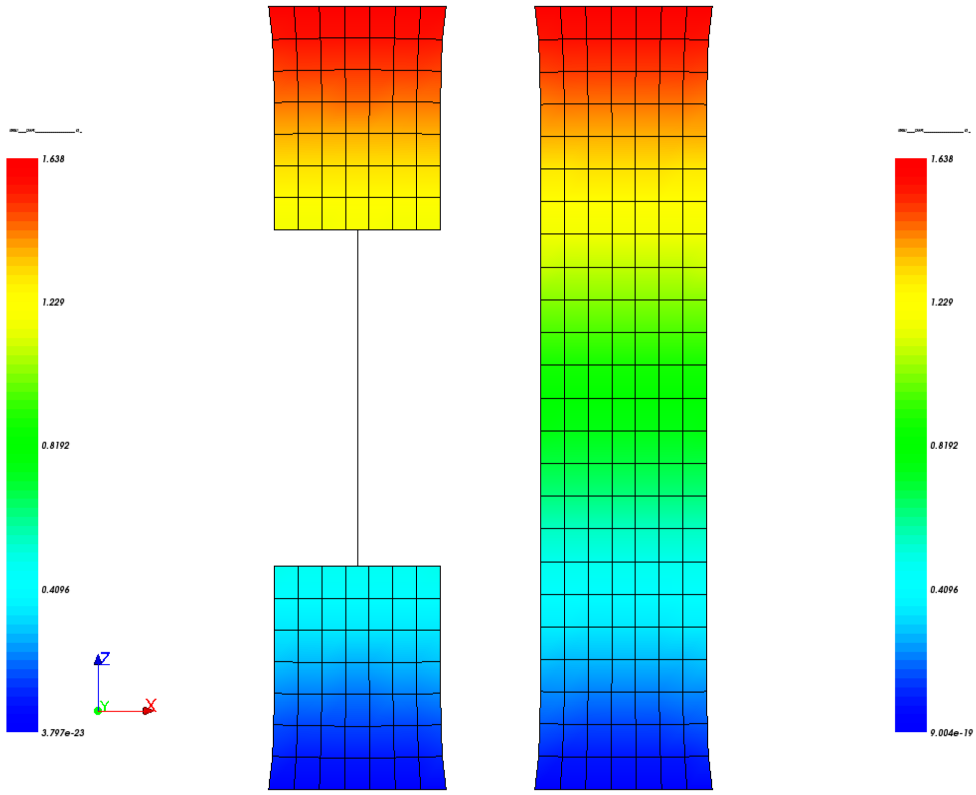


Figure 3.1: Deformation – no difference

3.2 Von Mises stress

The von Mises stress is displayed via the **EQUI_ELGA_SIGM** field in Salomé®, this ensures accurate results – Figure 3.2

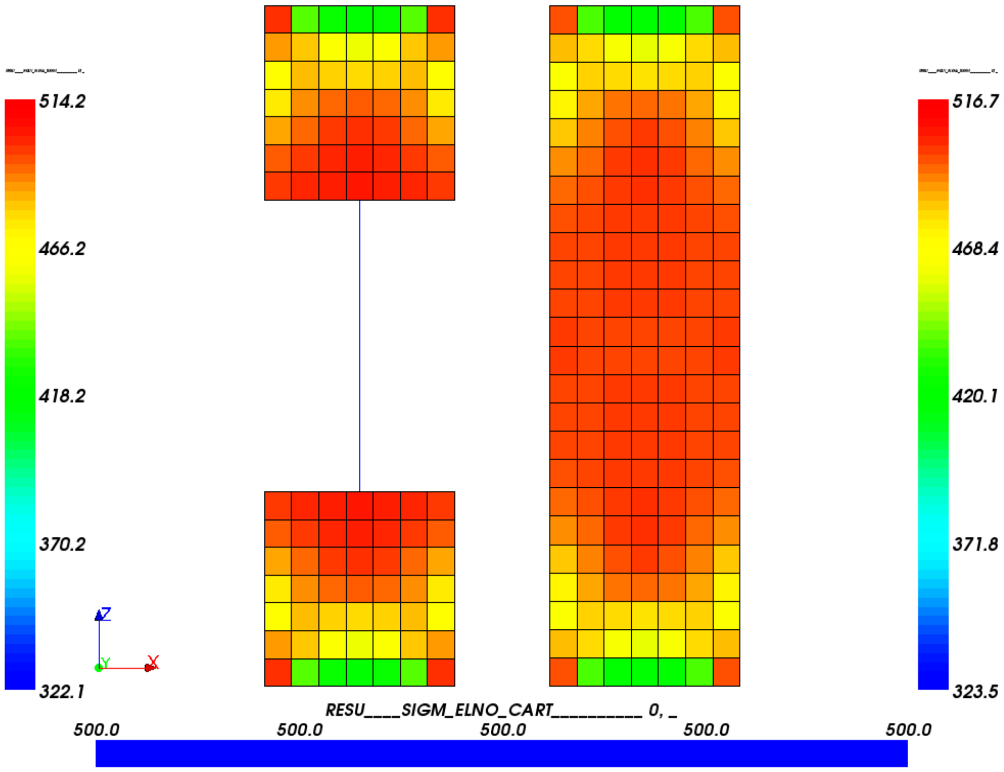


Figure 3.2: Von Mises stress. Left: LIAISON_ELEM version. Bottom: normal stress of beam element. Right: Full 3D version. Difference of 2.5 MPa

To check if the normal stress of the beam element corrosponds with the normal stress of the fully 3D structure, the **SIZZ** component of the **SIGM_DEPL_NOEU** is visualized.

Picking a node close to the center of the structure (with clipping, a node at the absolute center can be picked), a scalar value of 500.15 MPa is shown. This corrosponds with the 500 MPa of the beam element. See figure 3.3

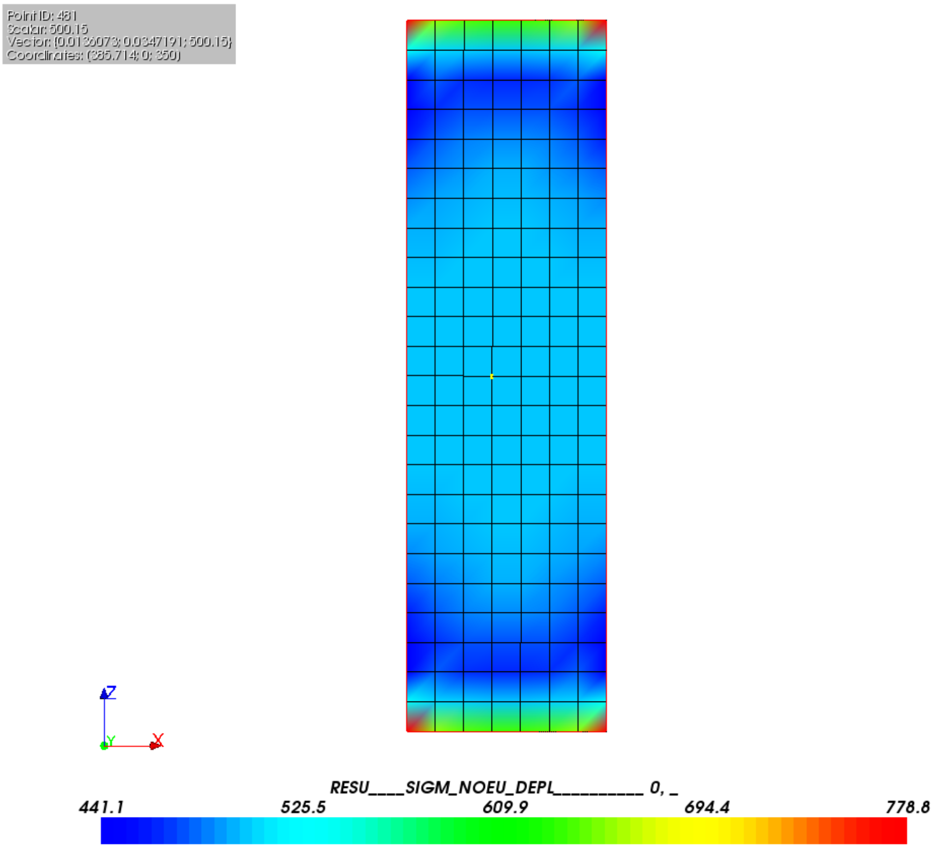


Figure 3.3: SIZZ component of SIGM_DEPL_NOEU

4 Conclusion, remarks and author(s)

That's it for this tutorial. Much more information can be found in the user documents on the Code Aster® website, its forum and on the CAELinux website.

Remark:

Acknowledgment goes out to EDF for releasing Code Aster® as free software and to all those who help out by answering questions in the forum and writing documentation / tutorials.
Contributions and/or corrections to this tutorial are always welcomed.

Author(s):
Claus Andersen – ClausAndersen81_[at]_gmail.com

ENDED OK

5 Links

[CAELinux Website] www.caelinux.com
[Code Aster® Website] www.code-aster.org
[GMSH® Website] www.geuz.org/GMSH
[XMGrace® Website] http://plasma-gate.weizmann.ac.il/Grace/