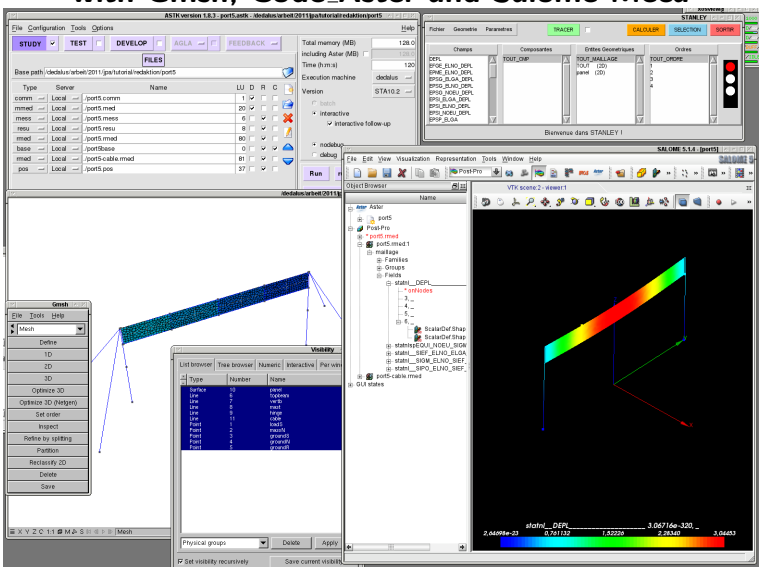
exploring the world of beams, plates, rods and cables structures in a linear and non linear fashion with Gmsh, Code_Aster and Salome-Meca



jean pierre aubry

If everything seems to be going well, you have obviously overlooked something.
11h Murphy's law

version 1.1.01 of 30/01/11, first public release.

to contact the author:
jeanpierre@lamachine.fr
jeanpierreaubry@ouvaton.org
jeanpierreaubry on http://www.code-aster.org/forum2/

This document is made with LaTeX.

Contents

1 FOREWORD

In the present document we will introduce the linear and non linear analysis of structures made of beams and cables.
We will study a very simple A frame beam, which could be the frame supporting a swing for kids in the garden, a frame supporting the signals on a motorway or two poles supporting a cable on a which a cycling clown will cross the nearby river.

We will at first introduce Gmsh as a tool for modeling and meshing the structure, then solve the problem and post process the results in the simplest manner in Salome-Meca.
Then we will go on solving the problem in a subtler way through ASTK, as could be done also in a stand alone version of *Code_Aster*.
We will have a look at the post processing capabilities of STANLEY macro command, and of the post processing module of Gmsh.

We will not use the Geom module of Salome-Meca, neither the Mesh modules of Salome-Meca.

Along this booklet we will increase the complexity of the problem, with more element types, non linear analysis.

We suppose that Gmsh and Salome-Meca are correctly installed on our machine, see the notes about that at the end of this document.
The examples are made with Gmsh version 2.5.1 and Salome-Meca 2010-2[1].

I hope this booklet will save the reader the many hours I spent fiddling around with the documentation and losing my track in the "dedalus" of trial runs.
This however does not dispense the newcomer to run benchmarks to get practice and check his modeling and commanding.

2 MODELING THE FIRST STRUCTURE WITH Gmsh

We will study an A frame, 1 m high, with a 2 m span, with one load under the form of a 10 kg mass at the quarter chord of the span and another load of 100 N, vertical downward at three quarter chord.

Note that the structure is symmetrical, in geometry, not in mass.

The first thing to be done is to create a directory for the problem, anywhere we have read write permissions, we will name this directory "port1".

Now let's launch Gmsh, we should have something looking like figure 1 :
One large window, named "untitled.geo".
One smaller window named "Gmsh", that's the command window.

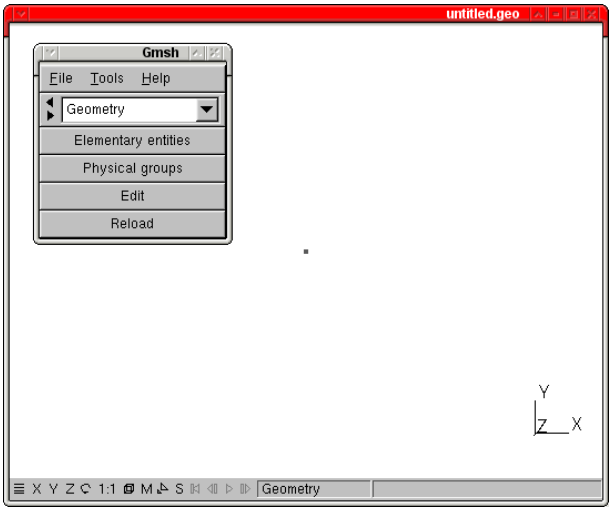


Figure 1: Gmsh and it's 2 windows

From this command window choose the menu File >

[1] and stand alone *Code_Aster*10.2, there seem to be some oddities with version 10.3.

SaveAs and save the file in the directory “port1” recently
created, name it “port1.geo”.
Say “Yes“ to the next dialog box.
.
Notice that the file name has not changed in the Gmsh window title, we must open the ”port1.geo” file thru the File > Open menu..
That’s an important feature of Gmsh when we do “SaveAs“ Gmsh saves a copy but does not switch to this newly created file, it keeps in the working file, as important is the fact that every change made in the GUI is saved on the fly in the .geo file.
.
This behavior may look strange to the beginner used to common Spreadsheet and Text processors, but once understood we wonder how we would do without it.
This is the common behavior of almost all data base processing program and in the field of finite element the mighty and $$$$$ worth Patran is behaving like that.

Now we are in the “Geometry” module of Gmsh (looking at the pull-down list of the command window), push on the button “Elementary entities“, which acts as a pull-down menu then Add > New > Point, another little windows pops up.
Type the coordinates x=0, in ”X coordinate” box, y=-1000, z=0, in the box ”Prescribed mesh element size at point“ enter 100, push ”Add“..

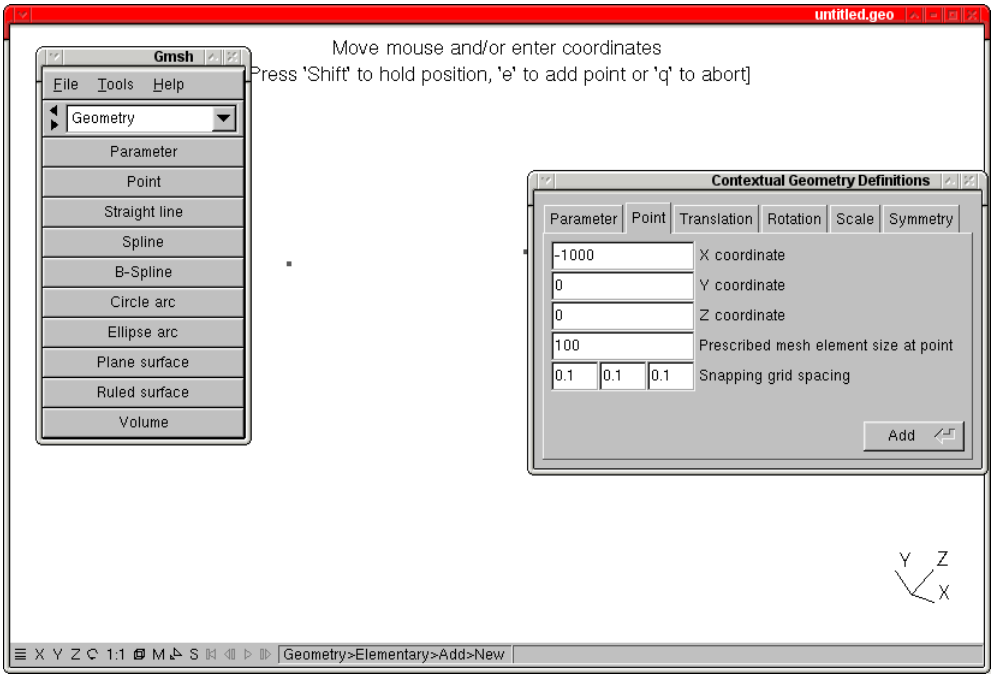


Figure 2: Creating a Point in Gmsh

Notice that at the top center of the main window we have some instructions about what can be done in the context.
Notice also that at the bottom of the Gmsh main window, the command line remainder on the middle left reminds us what is the active command.
.
We can see the newly created point as a little square box on Gmsh main window.

Now let’s open the file ”port1.geo“ with our favorite text editor, as far as I am concerned, I use Kate..
We can see something like this.

```
1 // Gmsh project created on Mon Nov 8 08:32:45 2010
```

```
2 cl1=100;
3 Point(1) = {0, -1000, 0, cl1};
```

Let's enter a new point with x=0, y=-1000, z=1000, "Add"
In the text editor we can see some warning that the source file has been changed, refresh the text window, and we see a line like:

```
1 // Gmsh project created on Mon Nov 8 08:32:45 2010
2 cl1=100;
3 Point(1) = {0, -1000, 0, cl1};
4 Point(2) = {0, -1000, 1000, cl1};
```

Let's type a line number 5 in the text editor like this:

```
5 Point(3) = {0, -500, 1000, cl1};
```

Save the file, in Gmsh use the left arrows at the left of "Geometry" until we can see at the bottom a menu entry named "Reload", push on it:
The new Point will appear in the main window.

Switching from test editor to Gmsh GUI is the real way to do things efficiently!

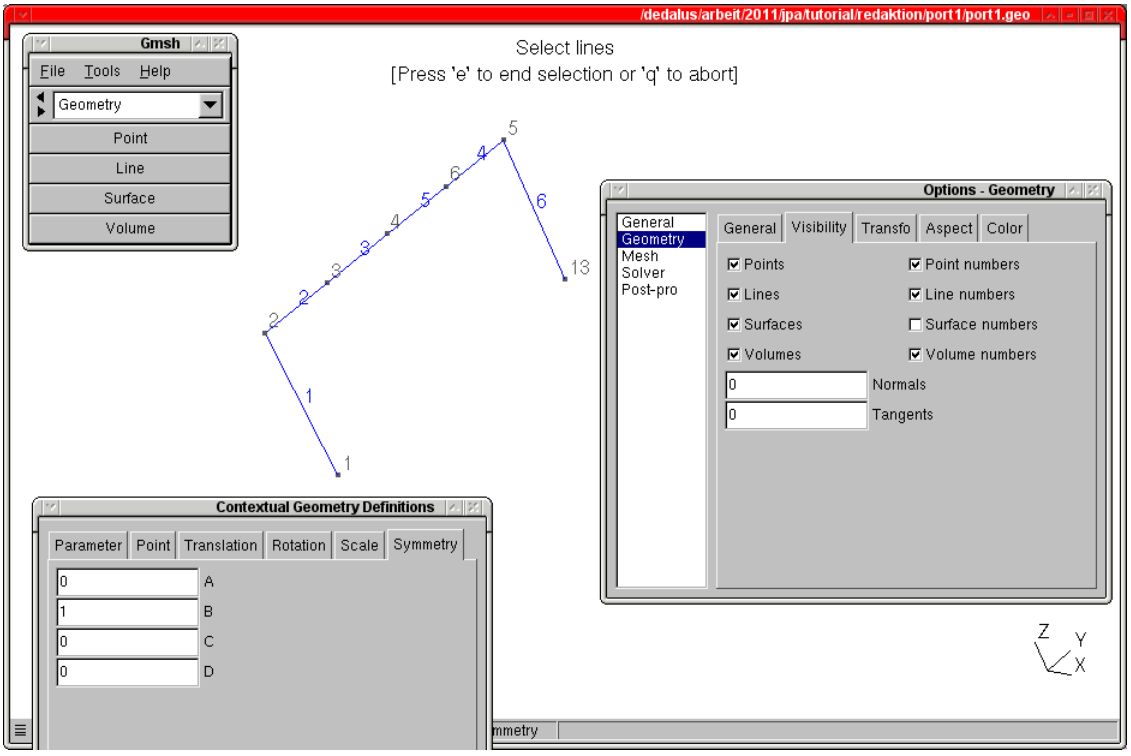


Figure 3: Geometric entities in Gmsh

In the menu "Tools"we choose "Options", then the entry "Geometry" checking or unchecking the "Point numbers" will make visible the numbering of the points.
Complete the geometry with a point:

```
5 Point(4) = {0, 0, 1000, cl1};
```

Now in the "Geometry" module of Gmsh we push on the button "Elementary entities", then Add > New >

Straight line, with the mouse we choose the 2 end points of the line, while looking carefully at the request in the top center of the screen.

Now in "Geometry" Elementary entities > Symmetry > Line, we fill the dialog box with A=0, B=1, C=D=0 [2], then "Return" and pick all the lines with the mouse and type "e".
The structure has been duplicated by symmetry about an xOz plane and looks like figure 3, depending on the visibility toggles.

Note that new points are automatically created, no duplicate point at Point 4 is created.
We can have a look in the text editor to see how this is done.

Now we will put together in "Groups" the geometric entities that share some properties, within "Geometry", Physical groups > Add > Line, pick with the mouse the four line being part of the A frame top bar, type "e", once this is done, like in figure 4.

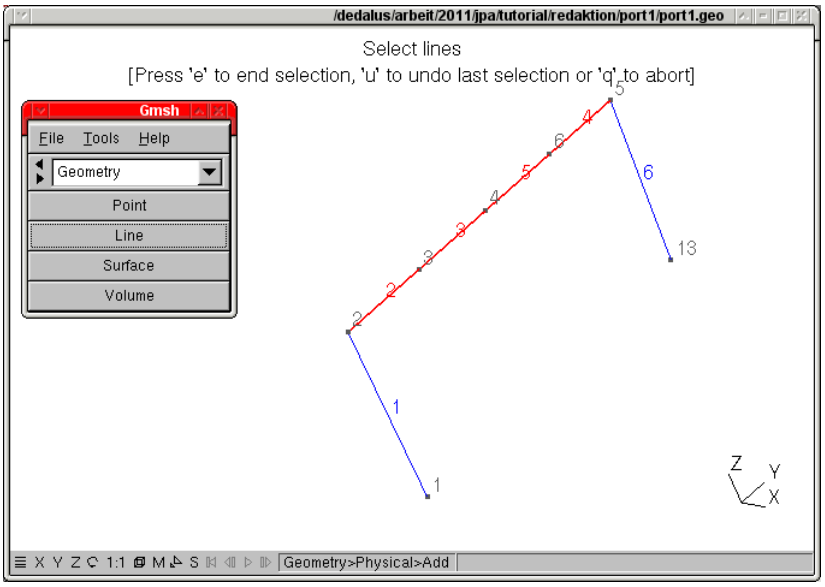


Figure 4: 4 Lines selected to make a Physical

In the text editor we can see a new line created :

```
Physical Line(7) = {2, 3, 5, 4};
```

The actual digit may be different!
Edit it so that it becomes:

```
Physical Line("topbeam") = {2, 3, 5, 4};
```

Here a very important warning: within a mesh file that will be processed later by *Code_Aster* we must not use group number longer than 8 characters!
And this not very much to make self reminding naming in a large problem.

This will give the name "topbeam to the group formed by the four lines 2, 3, 5, 4.
We keep going on either from the GUI or from the text editor until the groups looks like below, notice we have also groups of points.

[2] the symmetry is defined by a vector, A,B,C being the 3 components of the vector in x,y,z and D the distance in space from the vector origin to the origin of the global axis!

The final .geo file looks like this:

```
// Gmsh project created on Mon Nov  8 08:32:45 2010
cl1=100;
Point(1) = {0, -1000, 0, cl1};
Point(2) = {0, -1000, 1000, cl1};
Point(3) = {0, -500, 1000, cl1};
Point(4) = {0, 0, 1000, cl1};
Line(1) = {1, 2};
Line(2) = {2, 3};
Line(3) = {3, 4};
Symmetry {0, 1, 0, 0} {
  Duplicata { Line{2, 3, 1}; }
}
Physical Line("topbeam") = {2, 3, 5, 4};
Physical Line("mast") = {1, 6};
Physical Point("groundS") = {1};
Physical Point("groundN") = {13};
Physical Point("loadS") = {3};
Physical Point("massN") = {6};
```

Some Gmsh hints:
At the lower left corner of the windows there are several buttons:
X, Y, Z, set the view from this axis,
S means snapping, sometimes we have to deactivate it, if it was badly activated (if activated it appears in red).
To make a multiple selection we push "Ctrl" and drag with the mouse

3 MESHING IT WITH Gmsh

Pull down the vertical arrow to the right of "Geometry" until we can see "Mesh" push on the button "1D", the model is meshed like in figure 5.

Toggle the boxes in "Visibility" for "Mesh" and "Geometry" in the "Options" window to see what has been created.
In the text editor we can see an entry at line 2: "cl1=100", this entry "cl1" is repeated as a fourth entry at every node.

That is the elementary mesh size which will be applied around this given point.
Change cl1=500 in this line, save in the text editor, reload in Gmsh and mesh again, we can see that the mesh is now very coarse.

We can just as well push Define > Elements size at points in Mesh, then fill the "Value" with any number let's say 10 and pick up one of the point, then do the meshing.
The mesh is refined around that point.

Now we do File > Save As, in the Format pull down list choose "MED File (*.med)" and save the file as "port1.med".
A small window named "MED Options" pops up, like in figure 6:

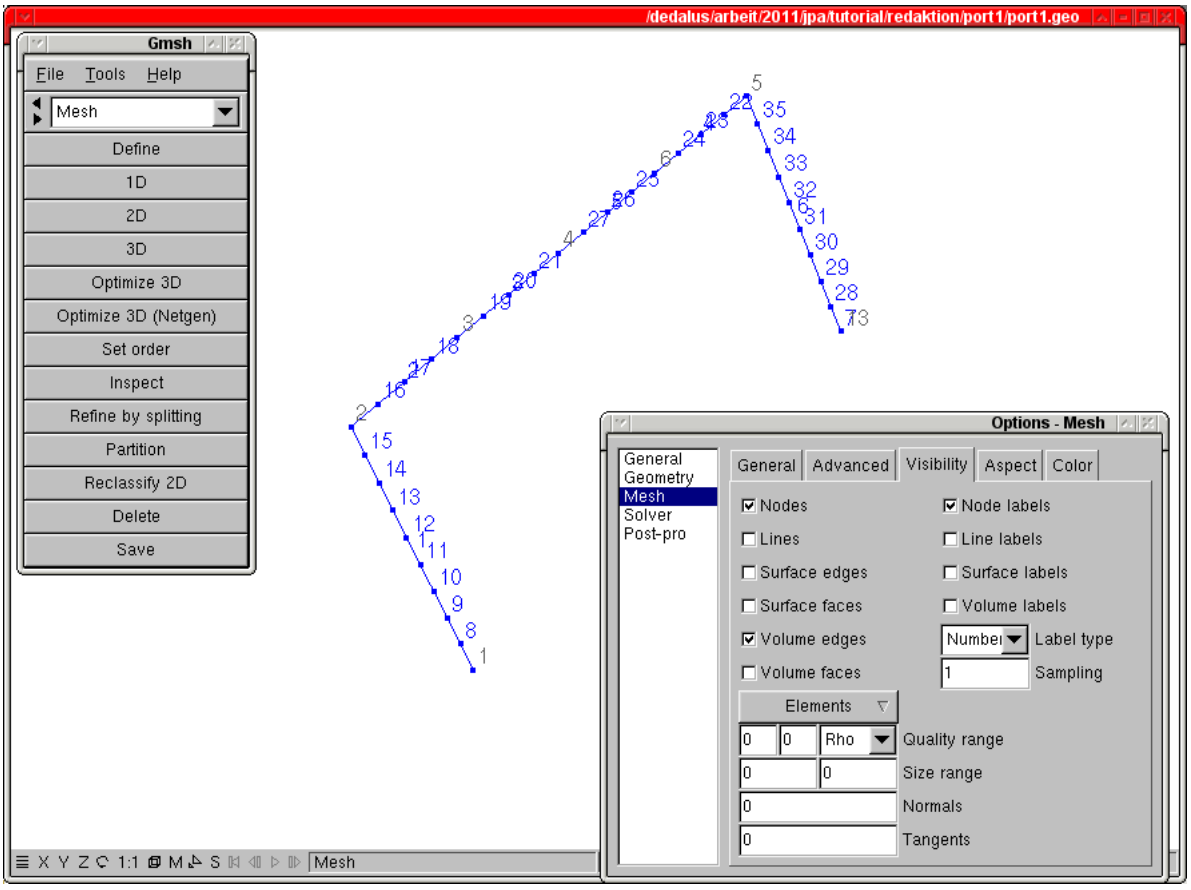


Figure 5: This is the mesh

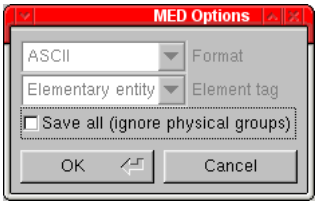


Figure 6: MED Save Options

Leave unchecked the box "Save all (ignore physical groups)" so as to save the created groups in the mesh.

More about that:
If we leave this box unchecked all the elements belonging to groups will be translated to the *.med file, and ONLY these elements, Points elements (or nodes) as well.
This means we have to create groups for every thing we need later.
If we check this box all the elements will be translated, all of them, but WITHOUT any group definition.
This is very important as when we do the meshing Gmsh meshes all the entities it founds without any distinction.

One more hint about using Gmsh, go to the menu Tools > Visibility, in the tab "list browser" at the bottom pull down to "Physicals groups", here we can play with the visibility groups by groups to see if things look like what we want, like in figure 7.

To finish with pushing <Ctrl> + <L> will display the "Message Console" windows which is a log, and will also pops up by itself in case of errors.

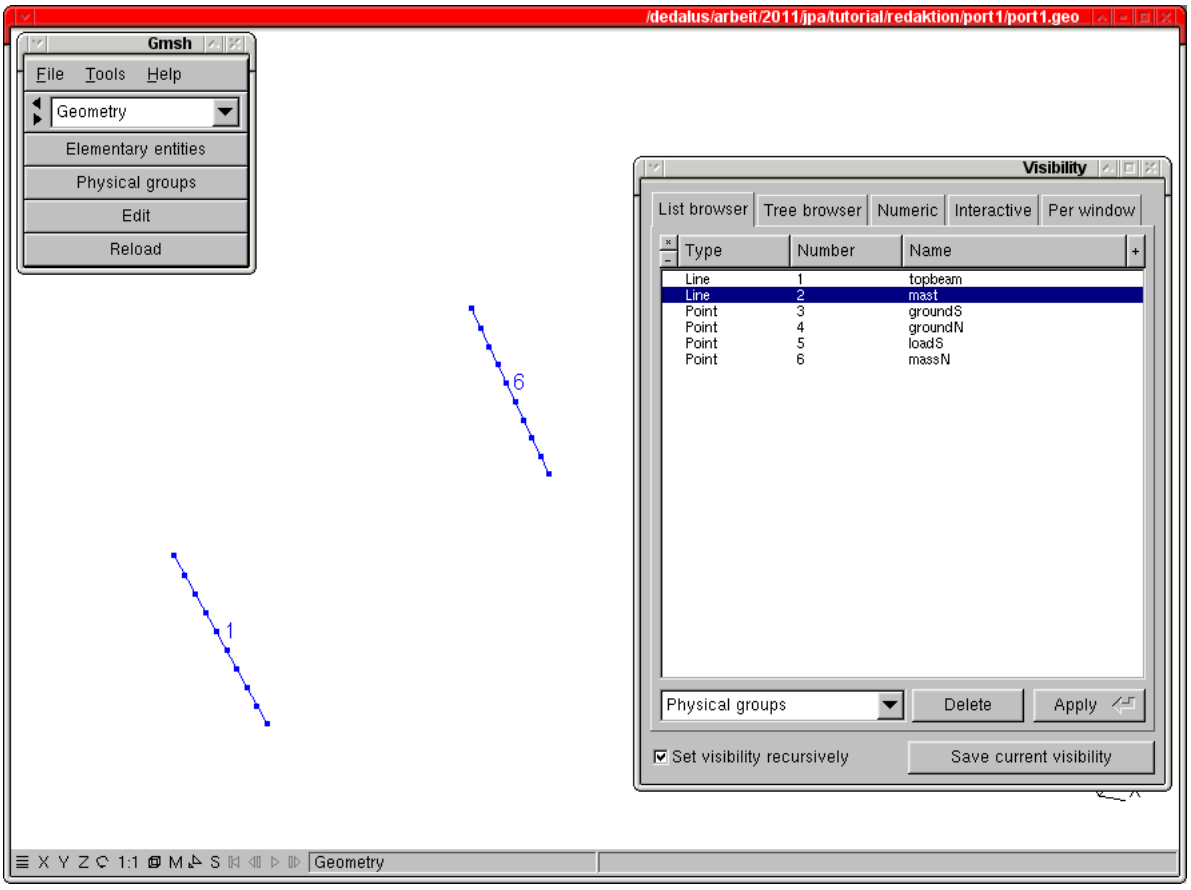


Figure 7: Only Physical "mast" set to visible, with Nodes and Line numbers

4 CALCULATING IT WITH SALOME-MECA

In this example we will solve the problem in Salome-Meca.
Salome-Meca is used here as a front to automate tasks for *Code_Aster*, we will describe only the basics to get a result for this example.
In order to calculate, Salome-Meca needs as input:

-a mesh file, we have just made it,

-a command file *.comm we will do that in the next section

Salome-Meca will then produce a set of files:

-a message file *.mess giving step by step all what has been done along the process,
with, *most important!* the errors and warnings,

-an ASCII results file *.resu giving some results in ASCII format,

-a graphical results file *.rmed readable for post processing by the Post-Pro module.

These last 2 files will only contain the results we instruct them to contain trough the *.comm command file.

5 CREATING THE COMMAND FILE

First but important remark:
In this example we will not use the integrated Aster command editor called Eficas!
I used it once or twice in my beginnings and decided that I would be better without it, so I am not the best qualified to explain how it works.
Moreover as I am still alive with Aster it is possible to do without it!

In this first example we will go straight away at a multiple load case solved at once, problem.

Now let''s have a look bit by bit at the commented command file for the "port1" problem.
The complete file can be found in "WORKED EXAMPLES".

5.1 Beginning with DEBUT()

Note that the lines beginnings with # are comments.

```
#this is the command file (.comm) to be used with the example port1
#according to the editor we use we may give it some syntaxic coloration
#for a more comfortable reading
#the original is written under Kate with tab set to 4 char and line length limited to
#90 char

#a line like:
#U4.21.01
#refers to the relevant document in the Code_Aster doc
#either in the installation directory or on http://www.code-aster.org

DEBUT();
#U4.21.01
```

5.2 Reading and modifying the mesh

```
#here we read the mesh which by default is assigned the fortran unit n° 20
#the INFO_MED=2, line provides a more verbose mode of what is read
#U4.21.01
mesh=LIRE_MAILLAGE( INFO=1,
#                   INFO_MED=2,
                    UNITE=20,FORMAT='MED',);

#here we create some groups within the mesh,
#with CREA_GROUP_MA=_F(NOM='TOUT',TOUT='OUI',),we create a group named 'TOUT' that
#contains all the elements which are in the mesh
#with CREA_GROUP_NO=(_F(TOUT_GROUP_MA='OUI',), we create a group of node for every
#group of element
#each of these groups contains all the nodes belonging to the parent element,
#it bear the same name
#this is very useful with MED file imported from Gmsh because Physical Points (groups)
#are translated as groups of elements (GROUP_MA) in the MED file
#we can thus apply boundary conditions to nodes
#U4.22.01
mesh=DEFI_GROUP(reuse =mesh,MAILLAGE=mesh,
                    CREA_GROUP_MA=_F(NOM='TOUT',TOUT='OUI',),
                    CREA_GROUP_NO=(_F(TOUT_GROUP_MA='OUI',),,)
                    );
```

Many other useful things can be done in this command, for example using 'UNION' of groups to simplify

their manipulation later on.

5.3 Making a finite element model from the mesh

```
#here we transform the mesh in a proper finite element model by affecting some
#propertiesto the mesh elements
#U4.41.01
#for example: we assign (AFFE= to the mesh groups (GROUP_MA=('topbeam','mast',)
#the properties of beams (MODELISATION='POU_D_T'),
#telling as well that we we will deal with a mechanical behavior
#(PHENOMENE='MECANIQUE')
#U3.11.01
#on the next line we assign to the mesh group 'massN', in fact it is a point,
#the properties of a discrete element
#U3.11.02
model=AFFE_MODELE(MAILLAGE=mesh,
  AFFE=(_F(GROUP_MA=('topbeam','mast',),PHENOMENE='MECANIQUE',
  MODELISATION='POU_D_T',),
_F(GROUP_MA=('massN',),PHENOMENE='MECANIQUE',MODELISATION='DIS_T',),
),
);
```

5.4 Defining materials

```
#here we define a material with its properties
#U4.43.01
steel=DEFI_MATERIAU(ELAS=_F(E=210000.,NU=0.3,RHO=8e-9),);
```

5.5 Assigning materials to elements

```
#here we assign to the beam element groups the material properties
#U4.43.03
#note that the dicrete element is not assigned a material
material=AFFE_MATERIAU(MAILLAGE=mesh,
                       AFFE=_F(GROUP_MA=('topbeam','mast',), MATER=steel,),
                       );
```

5.6 Giving properties to elements

With beam this is a rather tricky part, more about that later.

```
#here we define a set named "elemcar" and assigned some physical properties
#to the model elements
#U4.42.01
#this document is most important and should be read carefully!
elemcar=AFFE_CARA_ELEM(MODELE=model,
                       POUTRE=(
#the vertical members are rectangular section (40x20 mm) with a thickness of 1.5 mm
                              _F(GROUP_MA=('mast',),
                                 SECTION='RECTANGLE',CARA=('HY','HZ','EP',),
                                 VALE=(40, 20, 1.5,),),
#same with the horizontal bar
                               _F(GROUP_MA=('topbeam',),
                                  SECTION='RECTANGLE',CARA=('HY','HZ','EP',),
                                  VALE=(40, 20, 1.5,),),
#next line would have produced the same section properties
#                               _F(GROUP_MA=('topbeam',),SECTION='GENERALE',
#                                  CARA=('A','IY','IZ','AY','AZ','EY','EZ','JX','RY',
#                                        'RZ','RT'),
#                                  VALE=(171, 11518, 34908, 1.5, 1.5, 0, 0, 26700, 20,
#                                        10, 12,),),
                              ),
#in the next line we can give some orientation to the top bar along its axis,
#leave it commented at first
#                       ORIENTATION=_F(GROUP_MA=('topbeam',),CARA='ANGL_VRIL', VALE=90.0,),
#in the next line we give to the discrete elemnt the property of a point mass
#(CARA='M_T_D_N'), and give it the value of 0.01 tonnes ie 10 kg
                       DISCRET=(_F(GROUP_MA='massN', CARA='M_T_D_N',VALE=(.01),),
                               ),
                        );
```

5.7 Setting boundary conditions, fixed

```
#now we will set the boundary conditions, in several sets
#U4.44.01
#this document is most important and should be read carefully!

#first set, we fix in all 6 DOFs the bottom of the masts
ground=AFFE_CHAR_MECA(MODELE=model,
                       DDL_IMPO=_F(GROUP_NO=('groundS','groundN',),
                                   DX=0,DY=0,DZ=0,DRX=0,DRY=0,DRZ=0,),
                       );
```

I strongly recommend not to mix fixations and loads in a boundary condition set.
The English translation for DDL is DOF.

5.8 Setting boundary conditions, loads

```
#second we apply the gravity (PESANTEUR) to the beam groups and also to the
#discrete element, gravity is rounded off
selfwght=AFFE_CHAR_MECA(MODELE=model,
                  PESANTEUR =_F(GRAVITE=10000,DIRECTION=(0,0,-1),
                                GROUP_MA=('topbeam','mast','chan',),),
                  );

#third we apply to a node at the first quarter of the top bar a vertical force of 100 N
cc=AFFE_CHAR_MECA(MODELE=model,
                  FORCE_NODALE=_F(GROUP_NO=('loadS',),FZ=-100,),
                  );

#and fourth we apply to the top bar a distributed vertical force of 0.1 N
#per unit length, here mm
cr=AFFE_CHAR_MECA(MODELE=model,
#                 FORCE_POUTRE=_F(GROUP_MA=('topbeam',),FZ=-0.1,),
                  FORCE_NODALE=_F(GROUP_NO=('topbeam',),FZ=-0.1*100,),
                 );
```

In the above lines we have commented the line with "FORCE_POUTRE" as *Code_Aster* cannot yet calculate with more than one distributed load on beam elements!
However it knows how to in a non linear analysis.
This is a very curious limitation.
The work around is to replace it with an equivalent nodal force applied to the "GROUP_NO=('topbeam',)"
which we have created earlier with "DEFI_GROUP" and which contains all the nodes of "GROUP_MA=('topbeam',)"
This equivalent nodal force is the distributed load multiplied by the length of the beam elements.
Of course this is not exact as the beam length will not be exactly the "cl1" we gave in Gmsh (there must be a round number of elements in the line length) and the precision depends on the coarseness of the mesh.

5.9 Stepping for the load case

```
#here we define some step functions which will be applied to the loads
#for exemple the gravity force 'selfwght' will be applied the function selfw_m,
#0 at instant 2, 1 at instant 3 and 1 for all instantt after 3
selfw_m=DEFI_FONCTION(NOM_PARA='INST',VALE=(2,0, 3,1,),PROL_DROITE='CONSTANT',);
cc_m=DEFI_FONCTION(NOM_PARA='INST',VALE=(3,0, 4,1,),
                   PROL_GAUCHE='CONSTANT',PROL_DROITE='CONSTANT',);
cr_m=DEFI_FONCTION(NOM_PARA='INST',VALE=(4,0, 5,1,),
                   PROL_GAUCHE='CONSTANT',PROL_DROITE='CONSTANT',);
```

5.10 Stepping for the solution

```
#here we define a step function which will be applied to the calculation
#we will calculate every single instant from 2 to 5
liste=DEFI_LIST_REEL(DEBUT=2.0,INTERVALLE=_F(JUSQU_A=5,PAS=1.0,),);
```

We must understand what is done just above:
We will perform the calculation at four steps or instants, "INST" in the *Code_Aster* jargon, "from 2 to 5", without any intermediate, "PAS=1.0".
The gravity load "selfwght" will be applied with the instant stepping multiplying function "selfwght_m", that is, 0 and instant 2, 1 and instant 3 and later.
The nodal load "cc" will be applied with the instant stepping multiplying function "cc_m", that is, 0 at instants 2 and 3, 1 and instant 4 and later.
The same logic with the distributed load "cr", so that at instant 5 all the loads are applied together.

5.11 Analyzing it

```
#-------------------------------------------------
#here is the statical analysis
#-------------------------------------------------
#make a linear static calculation: MECA_STATIQUE
#of the previouly defined model: MODELE=model
#with the defined material set: CHAM_MATER=material
#and the physical properties in the set: CARA_ELEM=elemcar,
#U4.51.01
stat=MECA_STATIQUE(MODELE=model,CHAM_MATER=material,CARA_ELEM=elemcar,
#with the load, or boundary condition defined in EXCIT
#with the apllied step function where needed
                   EXCIT=(_F(CHARGE=ground,),
                          _F(CHARGE=selfwght,FONC_MULT=selfw_m,),
                          _F(CHARGE=cc,TYPE_CHARGE='FIXE',FONC_MULT=cc_m,),
                          _F(CHARGE=cr,TYPE_CHARGE='FIXE',FONC_MULT=cr_m,),
                          ),
#the calculation is made along this step function
                   LIST_INST=liste,
#we can give a title to this probleme
#                  TITRE='my_title'
                   );
```

5.12 Calculating results on the elements

```
#here we will enchance the result named stat with some results in the element,
#at this stage it contains only displacments at nodes
#enhance! thats why we use the keyword "reuse" that is curiuously written in lower case
#U4.81.01
stat=CALC_ELEM(reuse =stat,
               MODELE=model,
               CHAM_MATER=material,
```

```
               CARA_ELEM=elemcar,
               RESULTAT=stat,TYPE_OPTION='SIGM_STRUCT',
               OPTION=(
#to get some hints about the meanings of the following keywords have a look
#at this document
#U2.01.05
#                        'SIEF_ELGA_DEPL',
                         'SIEF_ELNO_ELGA','SIGM_ELNO_SIEF',
                         'SIPO_ELNO_DEPL','SIPO_ELNO_SIEF',
                        ),
             );
```

Stricly speaking if the keyword 'RESULTAT=' is present we do no need "MODELE=, CHAM_MATER=, CARA_ELEM=, or EXCIT=" as the are emebedded in the concept 'RESULTAT=', they may be used to restrict the results to a more specific "area" of the problem if used without 'RESULTAT='.
This would be the case if we had used for example several sets of 'CARA_ELEM' and wanted the results on just only one.

5.13 Calculating results at the nodes

```
#here we add to the results the reaction forces at the fixed nodes
#U4.81.02
stat=CALC_NO(reuse =stat,RESULTAT=stat,GROUP_NO_RESU = ('groundS','groundN',),
             OPTION=('REAC_NODA',),);
```

5.14 Calculating and printing the mass of the model

```
#the next lines calculate the structural mass of the given groups and put the results
#in the .resu file in a tabular format
#U4.81.22
#this should always be done to check the consistency of the model and calculation
masse=POST_ELEM(RESULTAT =stat,
                MODELE=model,
                MASS_INER=_F(GROUP_MA=('topbeam','mast','massN',),),
                TITRE= 'masse');
#U4.91.03
IMPR_TABLE (TABLE=masse,)
```

I reckon that checking the calculated mass with the estimates made otherwise is the prime test of the model validity.
Which in constructions made of beams will call for and increased mass density of the materials so as to cope with all the unmodeled bits and pieces which however load the structure by their own weight.

5.15 Printing the reactions, tables or resu

Checking the reactions against what is expected is just as well very important, so:

```
##the next lines calculate the sum of the reactions and put the results in the .resu file
#in a tabular format
#U4.81.21
#same remark as above
sum_reac=POST_RELEVE_T(ACTION=_F(INTITULE='sum reactions',
                GROUP_NO=('groundS','groundN',),RESULTAT=stat,NOM_CHAM='REAC_NODA',
                TOUT_ORDRE='OUI',
                RESULTANTE=('DX','DY','DZ'),OPERATION='EXTRACTION',),);
IMPR_TABLE (TABLE=sum_reac,)
#then in tabular format per node
reac1=POST_RELEVE_T(ACTION=_F(INTITULE='reactions1',
                    GROUP_NO=('groundS',),RESULTAT=stat,NOM_CHAM='REAC_NODA',
                    TOUT_ORDRE='OUI',
                    RESULTANTE=('DX','DY','DZ'),OPERATION='EXTRACTION',),);
IMPR_TABLE (TABLE=reac1,)
reac2=POST_RELEVE_T(ACTION=_F(INTITULE='reactions2',
                    GROUP_NO=('groundN',),RESULTAT=stat,NOM_CHAM='REAC_NODA',
                    TOUT_ORDRE='OUI',
                    RESULTANTE=('DX','DY','DZ'),OPERATION='EXTRACTION',),);
IMPR_TABLE (TABLE=reac2,)
#now we print the individual reactions in the .resu file in RESULTAT format
#U4.91.01
IMPR_RESU(MODELE=model,
          FORMAT='RESULTAT',
          RESU=_F(NOM_CHAM='REAC_NODA',GROUP_NO=('groundS','groundN',),
#                   MAILLAGE=mesh,
                RESULTAT=stat,),
          );
```

5.16 Printing some others results in ASCII file *.resu

```
#then the forces and moment in beams (SIEF_ELNO_ELGA) in the same file
#we could have printed more of the calculated results,but only calculated one
IMPR_RESU(MODELE=model,
          FORMAT='RESULTAT',
          RESU=(
                _F(NOM_CHAM='SIEF_ELNO_ELGA', GROUP_MA=('topbeam',), RESULTAT=stat,),
                _F(NOM_CHAM='SIEF_ELNO_ELGA', GROUP_MA=('mast',), RESULTAT=stat,),
               ),
          );
```

This is a rather restricted set printed to ASCII file we may want more or different according to the problem.

5.17 Printing results for graphical viewing, MED file

```
#here we put in the .med file, in graphical format,
#the results we want to be displayed in the Post-Pro module of Salome-Meca
#U7.05.21
IMPR_RESU(MODELE=model, FORMAT='MED', UNITE=80,
     RESU=(_F(
#next line means print it on the whole mesh
#             MAILLAGE=mesh,
#this one only on the named groups (in our case it is the same!)
              GROUP_MA=('topbeam','mast',),
              RESULTAT=stat,
              NOM_CHAM=('DEPL',
                        'SIEF_ELGA_DEPL',
                        'SIPO_ELNO_DEPL','SIPO_ELNO_SIEF','SIGM_ELNO_SIEF',
                        'REAC_NODA',
                        ),
                    ),
                ),
#         INFO_MAILLAGE='OUI',
           );
```

5.18 Ending the command file with FIN()

```
FIN()
#U4.11.02
```

6 PUTTING IT TOGETHER IN SALOME-MECA

Now we have a working *.comm file saved in the problem's directory.
We can now launch Salome-Meca.

First we create a problem with File > New, then we save it with File > SaveAS under the name "port1.hdf" in the problem directory.
.
Go in the pull down list to "Mesh" module, then menu File > IMPORT > MED file and after the appropriate navigation open "port1.med".

In the browser on the left hand side click on the + sign along "Mesh", then right click on "port1" > Show.
We can now see the mesh previously created, like in figure 8.
We can play a bit with the various possibilities in the menus or by right clicking on entities
.

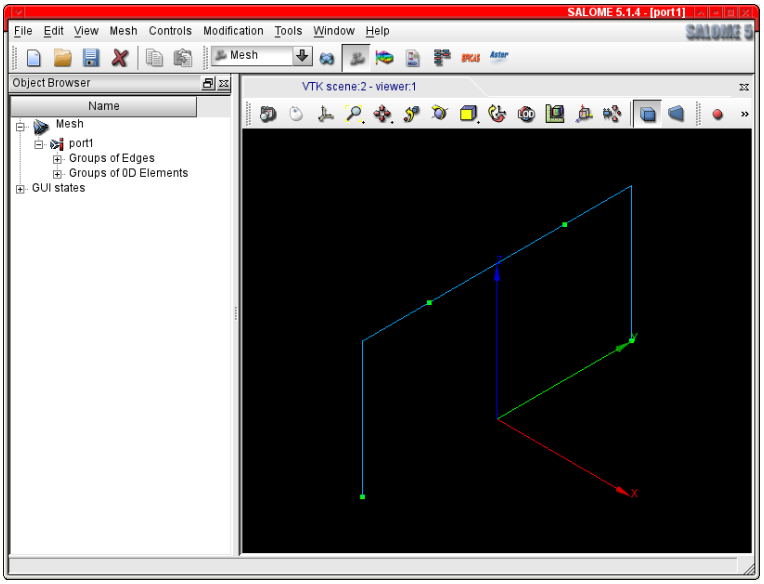


Figure 8: Mesh imported in Salome-Meca

Go in the pull down list to "Aster" module, then menu Aster > Add study case, the following windows appears, figure 9:

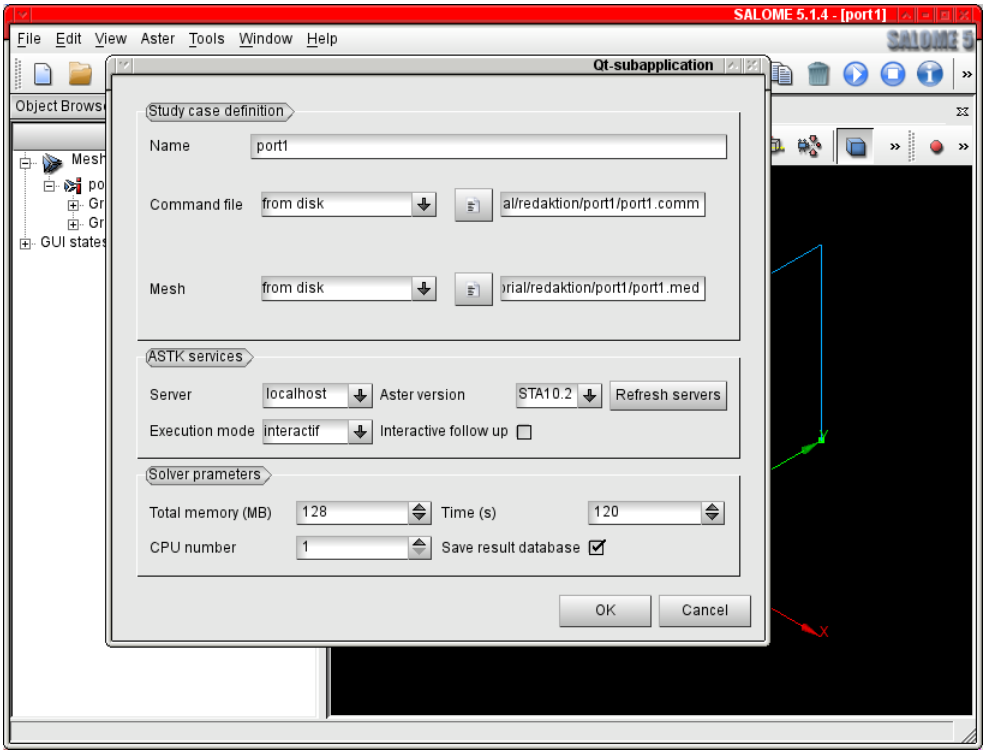


Figure 9: Setting the problem parameters

Name the case as "port1";
Pull the "Command file" to "from disk" then with the icon just on the right choose the "port1.comm" file;
The same with the "MESH" to choose "port1.med"

Uncheck the "Interactive follow up", many installations of Salome-Meca seem unable to run the problem with that box checked.
All the other options can be left as they are by defaults, the meaning of "Total memory" and "Time" is self explanatory, the values can be changed if needed.
The check box "Save result database" could be unchecked at this stage, leaving it does not do any harm, it slows a bit while disk writing after the solution.
Click "OK"
In the browser click on the + sign along "Aster", the right click on "port1" > Run.
Within an few seconds in the browser right click on"port1" > Status. then should appear an "Information" box stating an 'OK', otherwise see "FIXING IT WHEN THINGS GO WRONG".

7 VIEWING THE RESULTS IN SALOME-MECA

A new entry name Post-Pro has been created in the browser,
Click on the + sign along it,
then right click on "port1.med"
then on the + sign left of "mesh"
then on the + sign left of "Fields"
then on the + sign left of "stat_DEPL"
then right click on "4"
then "Activate Post-Pro Module",
then again on "4"
then choose "Deformed Shape and Scalar Map"
and finally click "OK" on the next box
and we will see a nice picture (figure 10) of the the deformed shape of the structure under the load at Instant 4.

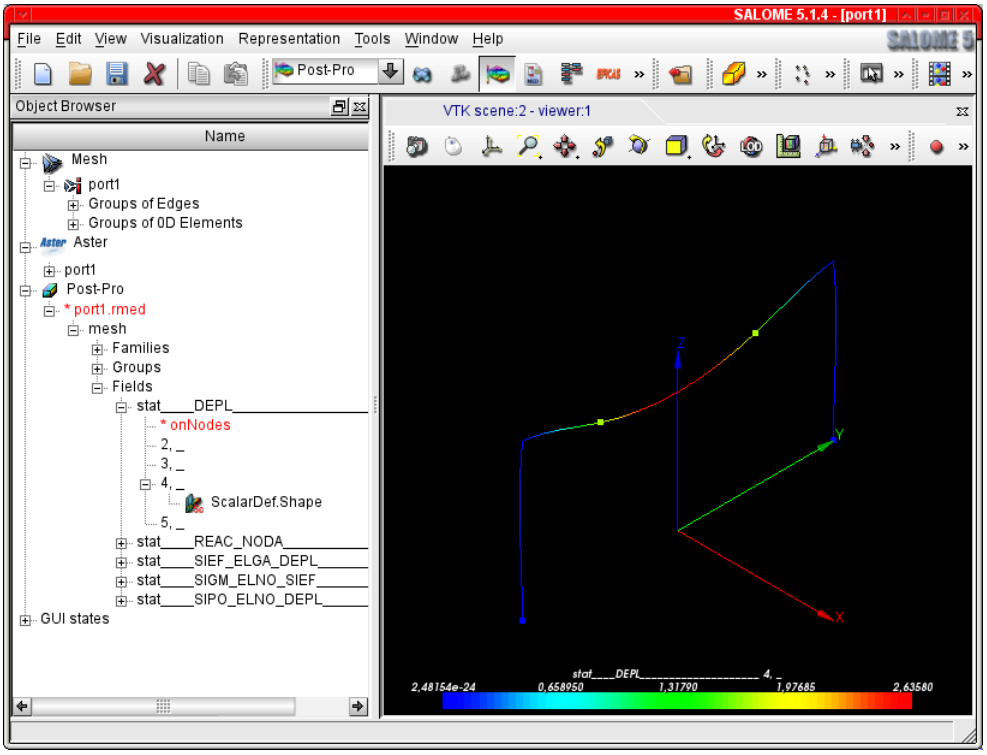


Figure 10: Deformed shape, with values

We can do the same at any other instant.

Now lets try to get the picture of 'SIPO_ELNO_DEPL' component SMFY in a Scalar bar like figure 11.

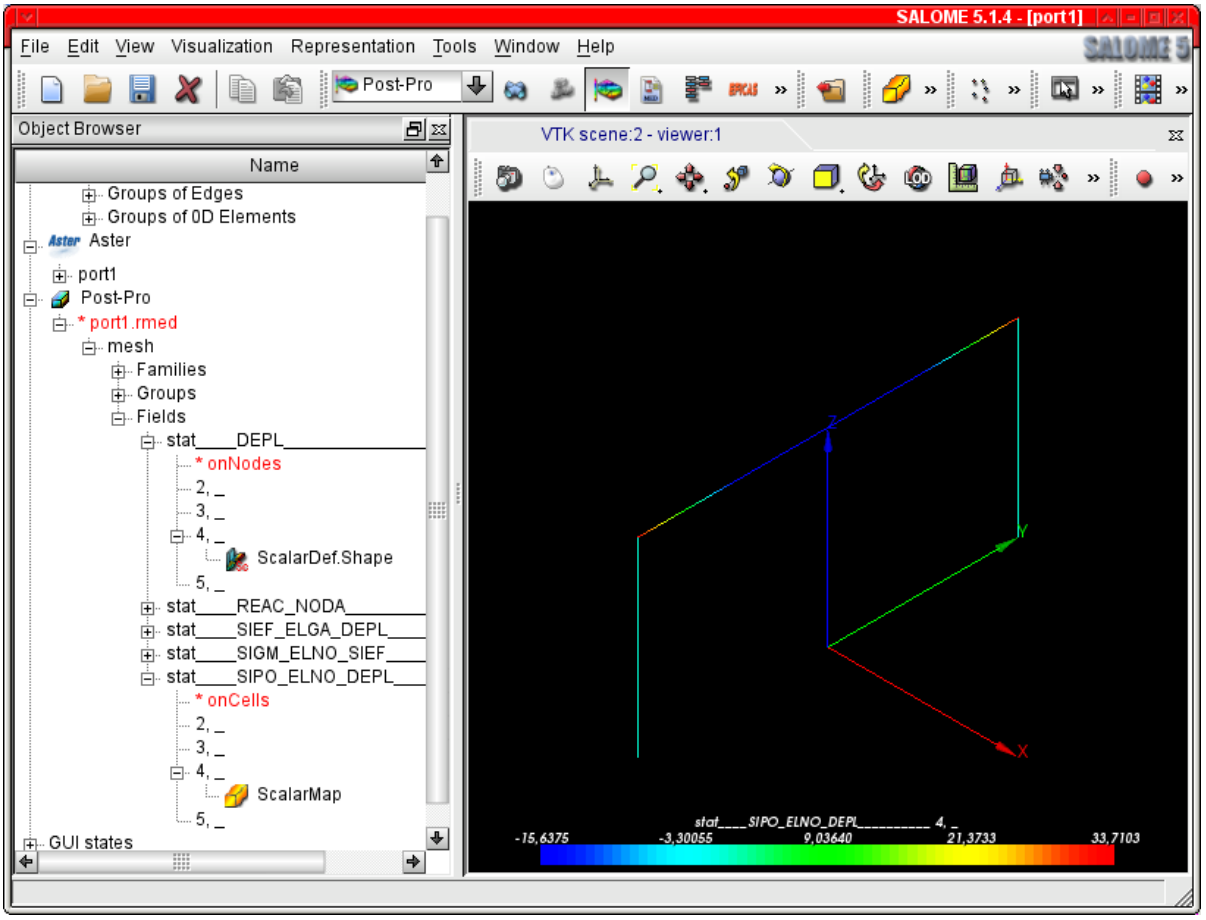


Figure 11: Value of the field 'SIPO_ELNO_DEPL', (its written in the Scalar Bar), component SMFY, but nothing tells us about the component in the bar!

Then with the vector of REAC_NODA on top of it like figure 12:

To access the dialog box that allows us to change the appearance of the displayed picture we can right click in the image title in the browser an choose "Edit...", or right click on its picture in the graphical windows (the object concerned will show within a red bounding box frame) and choose "Edit...".

This dialog box (figure 13 allows us to change almost everything that appears in a graphic Post-pro window:
Scale of deformed shape,
Range of value in the Scalar Bar,
and its Position and Dimensions,
we can also select the Groups that will be displayed.

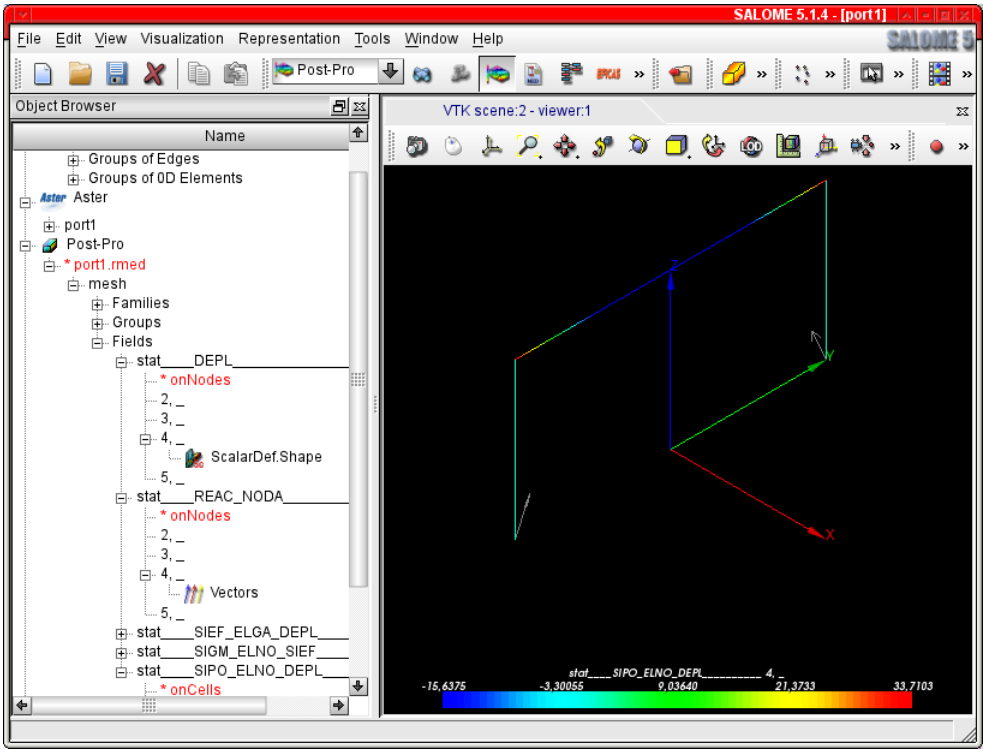


Figure 12: Value of the field 'SIPO_ELNO_DEPL', with the Reaction Vector on top

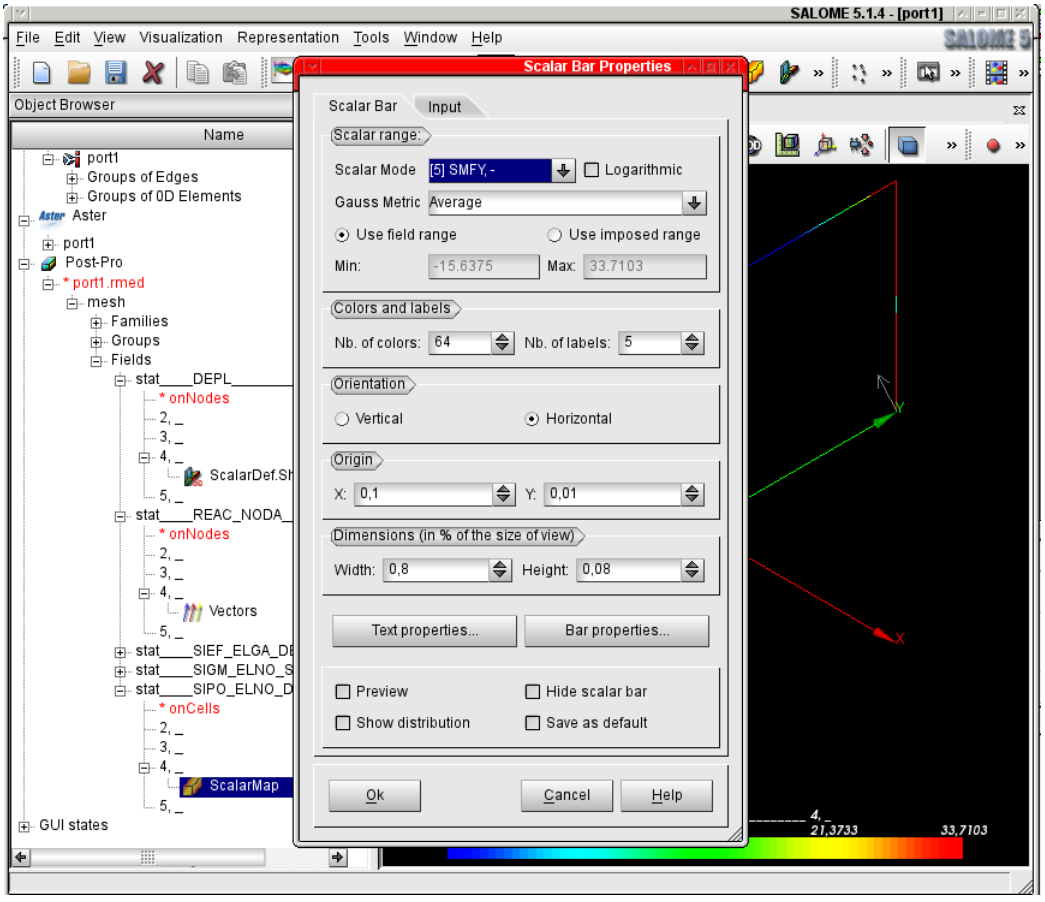


Figure 13: Modifying the appearence

8 SOPHISTICATING THE DISPLAY

We can also "pick" a value from the screen, for this we first need to have one results selceted from the tree, then in the menu select 'View > Windows > and a dialog box, like in figure 14 should appear.

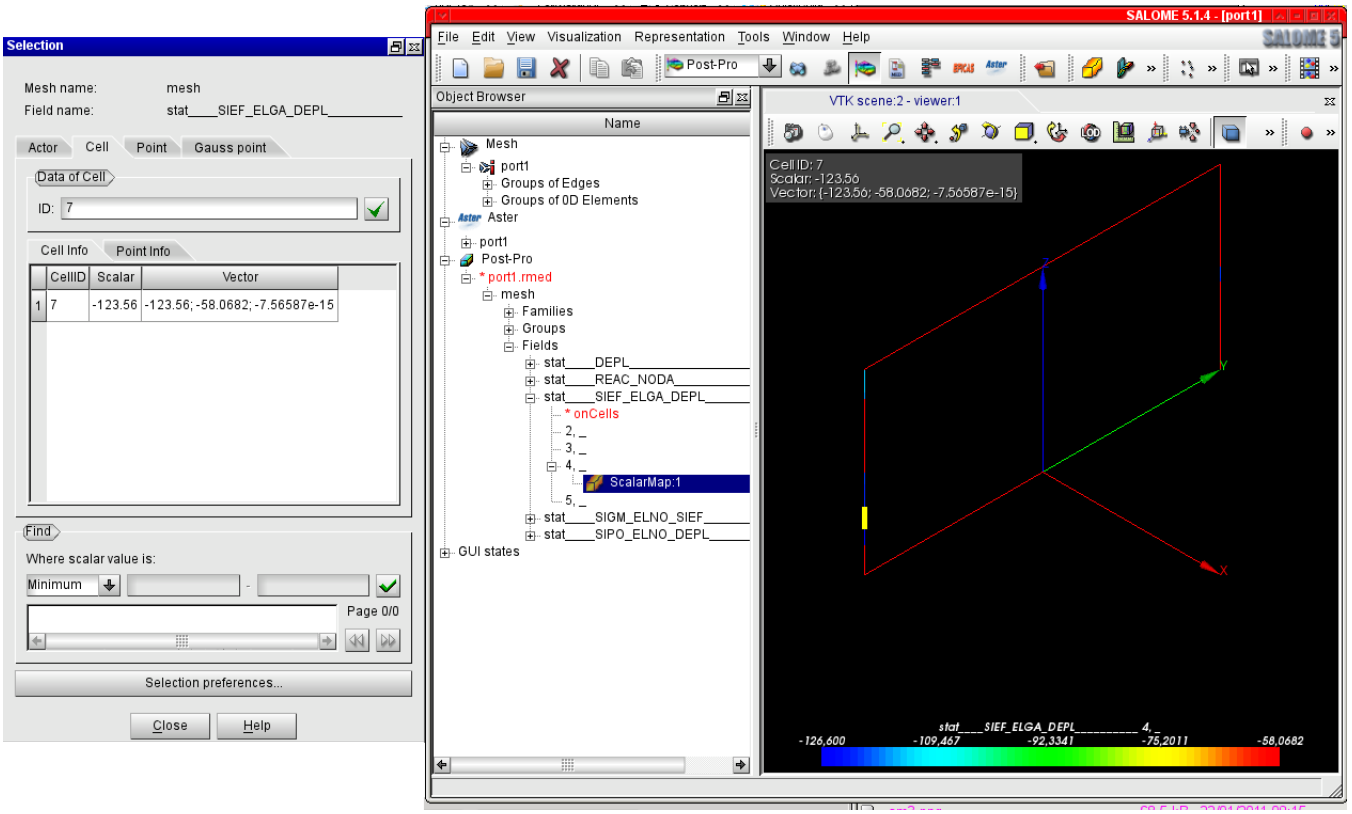


Figure 14: "Picking" a result from the screen

Here we are in the 'SIEF_ELGA_DEPL' with the Scalar Bar showing the "N' component, in the "'Selection" dialog box we choose the "Cell" tab and click on one element, some values appear in the "Selection" window.
Here we can check that the first value of the vector "123.56" is in agreement with the graphical display.
However this nice feature is not of much interest for beam results as only the first 3 component of the tensor (or vector) are shown, which means that it is impossible to access a component like 'MFY', this is rather disappointing!
Last but not least the picking is rather temperamental and quite often nothing wants to appear in the result box!

Maybe more interesting we can plot a tensor value, like 'SMFY' on a deformed shape, for that:
In the tree choose "stat___DEPL", choose the result we want
right click Edit
in "Deformed Shape and Scalar Map",
in the "Scalar Fied" pull down to "stat___SIPO_ELNO_DEPL"
in "Scalar Bar",
in the "Scalar Fied"
pull down to the component "SMFY"
"OK"
And the "ScalarDef.Shape" results display should look like figure 15

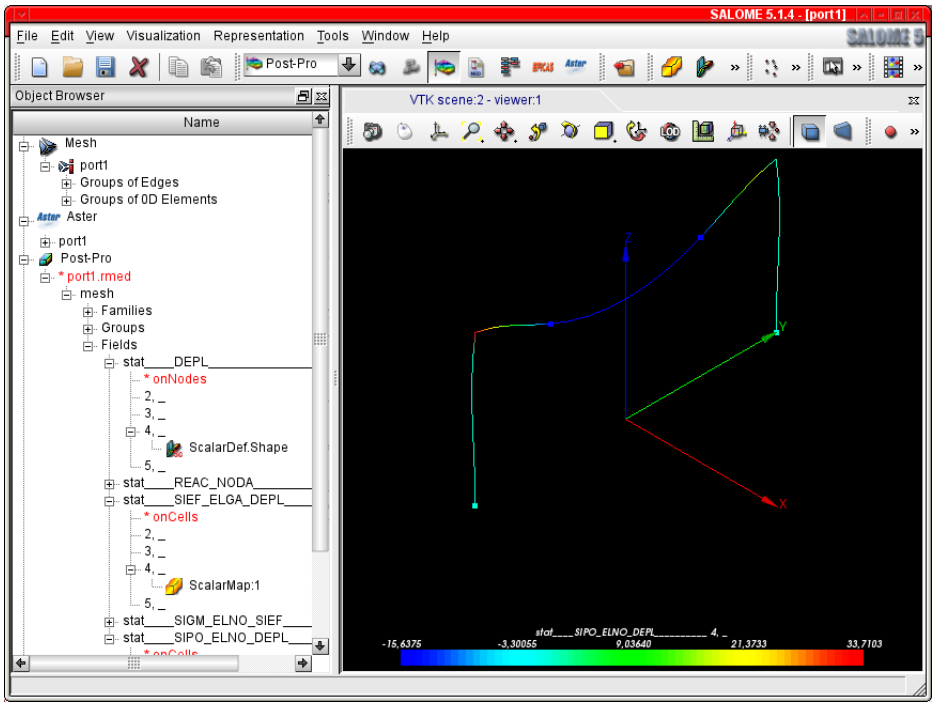


Figure 15: 'SMFY' value on deformed shape

9 LOOKING AT ASCII RESULTS

9.1 Printing 'RESULTAT'

It is easy to read the .resu file with any text editor, we will not reproduce it here.

9.2 Printing 'TABLE'

We can print results with "IMPR_TABLE" or with "IMPR_RESU... FORMAT='RESULTAT"'.
We have to make some remarks here:
The printed results have of course the same value a long as we print the same things.
The appearance may be more pleasing in "RESULTAT" mode, this a matter of taste.
The object "TABLE" allows many manipulations with some Python code which may be very helpful, there is one section near the end of this book with an example of a table manipulated[3] with Python.

[3] That is a very basic manipulation, the options are much wider!

10 DEALING WITH UNITS

The previous model has its lengths in millimeters, its forces in Newton and its time in seconds. in such a system we need to express the mass in tons and the mass density in t/mm3.
This oddities, though common in the world of engineering are necessary to form an homogeneous system of units.
For example the volume of the element will be calculated by *Code_Aster* in cubic mm which multiplied by the mass density will give us mass in tons which again multiplied by an acceleration which is implied in mm per square seconds will give us kilograms multiplied by meter divided by squared seconds also known as Newton.
Then the forces results are in Newton the moments results in Nmm and the stresses in N/mm2.

All this to say that *Code_Aster* is not aware of the units we use, given a set of units as entries it will produce results in a set of homogeneous units.

11 UNDERSTANDING 'SIEF', 'SIGM', 'SIPO'....

The fields 'DEPL' and 'SIEF_ELGA_DEPL' are calculated even if we do not request it in CALC_ELEM.
Calculation or printing of any field can be restricted to one or more of its component with 'NOM_CMP'.

Field 'DEPL' means displacement.

For beams the field 'SIEF_ELGA_DEPL' means Effort, per element at Gauss point (in this case the end nodes) calculated from displacement (that field is restricted to linear analysis).
From a practical point of view it is the normal force, N, the 2 shear forces, VY and VZ, and the 3 moments, MT, MFY and MFZ in the beam, in its LOCAL axis.

For beams the field 'SIPO_ELNO_DEPL' means Stress, per element at end nodes calculated from displacement (again that field is restricted to linear analysis).
This is the stress produced by any of the three forces and moment defined above if they were acting alone on the beam section.
For example SMFY is the stress due to the single bending moment MFY, it is computed from MFY above and the beam characteristics defined in "AFFE_CARA_ELEM...POUTRE".
Have a look at U4.42.01 for more information.
'SIPO_ELNO_SIEF' is almost the same things but calculated from the deformed shape, so it is the one that should be used in non linear analysis, though it can be used as well in linear, and then gives the same results.

'SIGM_ELNO_SIEF' gives the component of the stress in the direction of its last 2 characters, the most important SXX, can be seen as the maximum normal stress, the addition of the normal stresses due to normal force and the two bending moments.

An important note is that the stress due to the torsional moment is only true if the "Saint Venant" conditions are achieved, that is no warping of the section, which is more and more false as the section differs from a round and closed one to an open one, and or if the warping is restrained by boundary conditions.
The actual stress may be the much higher by a factor which be ten or more, the problem then cannot strictly be solved by this type of beam analysis.
That is a classical problem of stress analysis described in many text books.

Another important note, Salome proposes as the first choice of a field the option modulus, which is the modulus of the vector formed by some of the components.
For displacement this is the sum of the first three and is thus a very exact picture.

For some others field like 'SIEF_ELGA_DEPL' or 'SIPO_ELNO_DEPL' I have yet to understand what is this modulus, and I consider it as meaningless and never look at it as a serious information.

Another serious drawback is the fact that the component actually displayed is not reminded in the Scalar Bar Title.

This example is straight forward and all the forces and displacements can be calculated by hand, from text books, we should do so as a check.
Whatever calculation we do with a finite element code we should have at first a good idea of what displacements and forces will be, in some key areas, if not we are prone to waste a lot of time, or produce wrong results.

12 ORIENTING BEAM ELEMENTS

Look for this line in the .comm file

```
#                          ORIENTATION=_F(GROUP_MA=('topbeam',),CARA='ANGL_VRIL', VALE=0.0,),
```

Uncomment it and change VALE=90.0, so it becomes:

```
                           ORIENTATION=_F(GROUP_MA=('topbeam',),CARA='ANGL_VRIL', VALE=90.0,),
```

Run the calculation, and we can see that the maximum displacement becomes 1.23 when it originally was 2.63.
In fact the rectangular section of the top bar (group 'topbeam) was originally lying on its flat side and we have turned it 90 degrees along its own axis so it now lies vertically, just like in the figure 16.
.

The rule for the orientation of the local axis of beams in *Code_Aster* is very simple:
The local x axis lies along the beam.
The local y axis lies by defaults in the xOy plane, and the local z completes the trihedron.
Here by defaults means that the keyword "ORIENTATION" does not exist in the .comm file, for the given group.
If the beam is strictly in the global Z axis, the beam y local axis is then exactly coincident with the global Y axis.

If we have an array of vertical beams along the Z axis around a circle in the XOY plane and want them to have a rectangular section pointing towards the center of the circle, the trick is to offset them slightly from vertical, just enough so that the tangent of the angle is not null.

In the figure 17 we have a few cases depicting beam orientation:

Here the global axis z is supposed vertical.
In "a" an array of three beams, rectangular section, lying "on their flat" at an angle of 30° on the horizontal plane xOy.
In "b" the same array lying exactly vertical, *strictly* parallel to z axis, without any "ORIENTATION" keyword, note the local y axis pointing in Y global axis.
In "c" the same as in "b" but the z local axis is turned a bit so as to be off vertical, note the local axis z pointing towards the centre of the global coordinate system.
"d" is almost the same as "a" but the beam in the top of the figure as been rotated of 90° with the keywords "ORIENTATION=_F(....,CARA='ANGL_VRIL', VALE=90.0,),

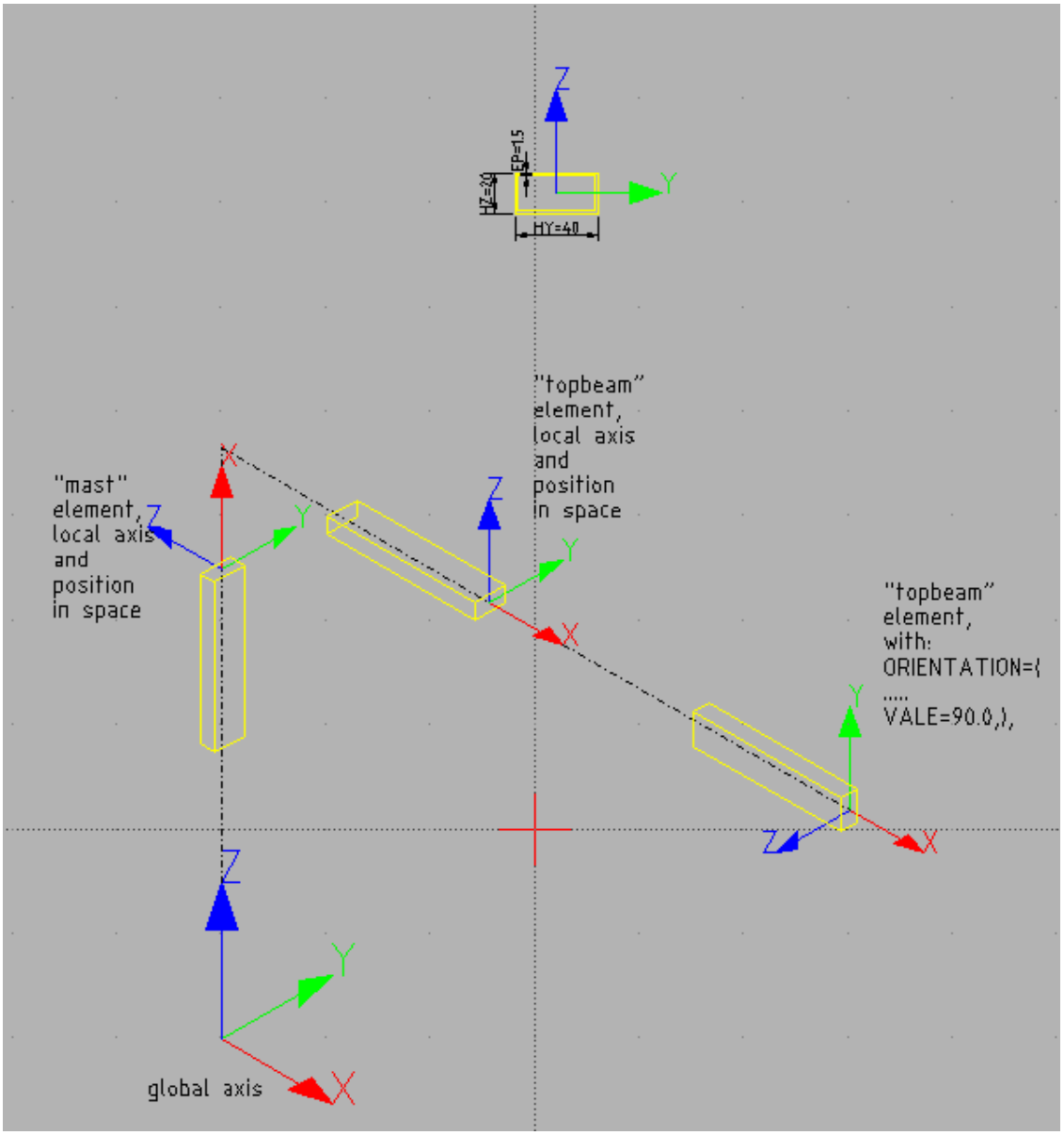


Figure 16: Orienting the topbeam element

In "e" it is the same beam.
I "f" we can see the same beam element again but with the order of his nodes inverted.

Again document U4.42.01 gives all instructions on how to change the orientation of the local axis.

When we perform any change of orientation of one beam, we of course change its local axis, this has to be kept in mind to understand the forces and stresses results.

This applies also to the forces applied along the beam if defined with the keyword "REPERE='LOCAL". This has to be kept in mind and may lead to exactly the reverse of what we expected!

Once we have fully understood the principles of beam orientation in *Code_Aster* and applied them in the mesh and groups of element we may well find it is hardly ever necessary to use anything but the keyword 'ANGL_VRIL' to model any practical model[4].

However when we are in doubt about some orientation it is always a good idea to perform a "dummy" analysis with loads and boundary conditions restricted to the very area that makes doubt and to check the

[4] As far as I concerned I never had to use any other way to define orientation.

results at various orientations with aquick hand calculation.

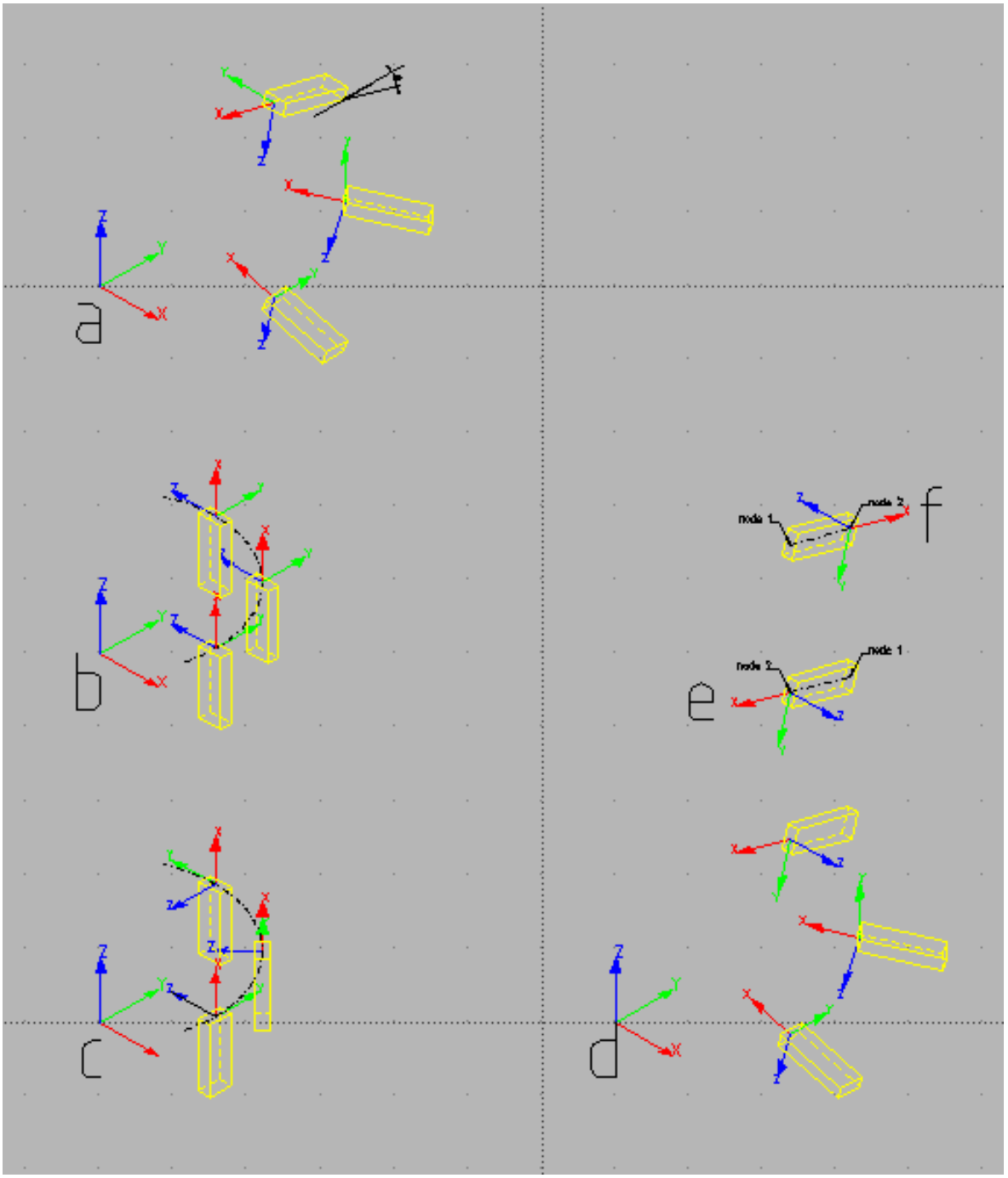


Figure 17: Various cases of beam Orientation

13 FINDING IT WHEN THINGS GO WRONG

Or: one should always read carefully the *.mess file, "almost" *everything is written in it*.

In this case all went well and we got a result at the first try.
Let's say that is an exception we will always have some kinds of errors in the early stages.
The only way around is to look at the *.mess file, the different kinds of errors are quite explicitly described and the *.comm file can be corrected accordingly.
At the beginning there will be many syntax errors, and it can make debugging a rather tedious process.

Even when the calculation gives a result there may be warnings in the *.mess file.
As is usually stated in the warning itself we must understand what it means before taking the results for granted

Here is the end the .mess file in case of success:
——————————-

```
EXECUTION_CODE_ASTER_EXIT_9671=0
```

——————————-
"=O" means no error, no warning. "9671" is the job ID, in case of problem we will find some files with this ID in "$HOME/flasheur" directory.

Here is the end the .mess file with a typical Syntax Error:
——————————-

```
   !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
   ! erreur de syntaxe,  Erreur de nom: name 'DEFI_GROUPE' is not defined ligne 11 !
   !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
```

——————————-
We should have 'DEFI_GROUP' instead of 'DEFI_GROUPE'.

If we have "=1" instead of "=0" at the end of the file, we must look higher in the file until we find something like this:
——————————-

```
   !-----------------------------------------------------------------!
   ! <EXCEPTION> <MODELISA7_75>                                      !
   !                                                                 !
   ! le GROUP_NO  couron  ne fait pas partie du maillage :  maillage !
   !-----------------------------------------------------------------!
```

——————————-
The error description always begins with a <.
Here some command refers to the group 'couron' that is not part of the mesh.

We will not go any further in the description of errors and warnings this is well documented in the .mess file.

And our ingeniousness in raising errors does not seem to have any limit.

14 ADDING END RELEASE TO THE TOP BAR

It is obvious from the results of the preceding example that the top bar is rigidly linked to the top of the mast, maybe we want this to be a rotation free joint.

Code_Aster does not provide any end release option at the end of a beam element.
What we will do is create a short (10 mm) mesh line element at this connection.
And give it the properties of a discrete element 'K_TR_D_L' from U4.42.01, in this type of element:

K stands for stiffness,
TR for Translation and Rotation,
D for diagonal only matrix (only 6 terms),
and L for line, the element is over a SEG2 mesh element.

The ad hoc line in AFFE_CARA_ELEM are so:

```
                        DISCRET=(.........
#here we define the hinges as an element with very low stifness 1e1 in rotation around
#the X axis in GLOBAL coordinates
                                    _F(GROUP_MA=('hinge',),CARA='K_TR_D_L',
                                       VALE=(1e6,1e6,1e6,1e1,1e9,1e9,),REPERE='GLOBAL',),
                                    ),
```

The order of the six value is stiffness in translation along X, Y, Z, stiffness in rotation around X, Y, Z, we have given relatively high value to all of them but for the rotation around X.
X being here in 'GLOBAL' coordinates.
In 'LOCAL' it would have been a free rotation around y with the specified 'ORIENTATION'.
It is risky to put a 0.0 value at a free rotation or translation.
Note that the orientation of this discrete line element follows exactly the same rules as the one described earlier for beams.

We will not go any longer on the subject here, it is fairly straight forward to modify the geometry and mesh in Gmsh and to modify the *.comm file;.
Just check once the calculation is made that the top bar is really articulated, that is maximum displacement is what it should be (from a hand calculation) and no moment are transmitted into to the masts.
Just in case the relevant (port2.*) files are in "WORKED EXAMPLES".

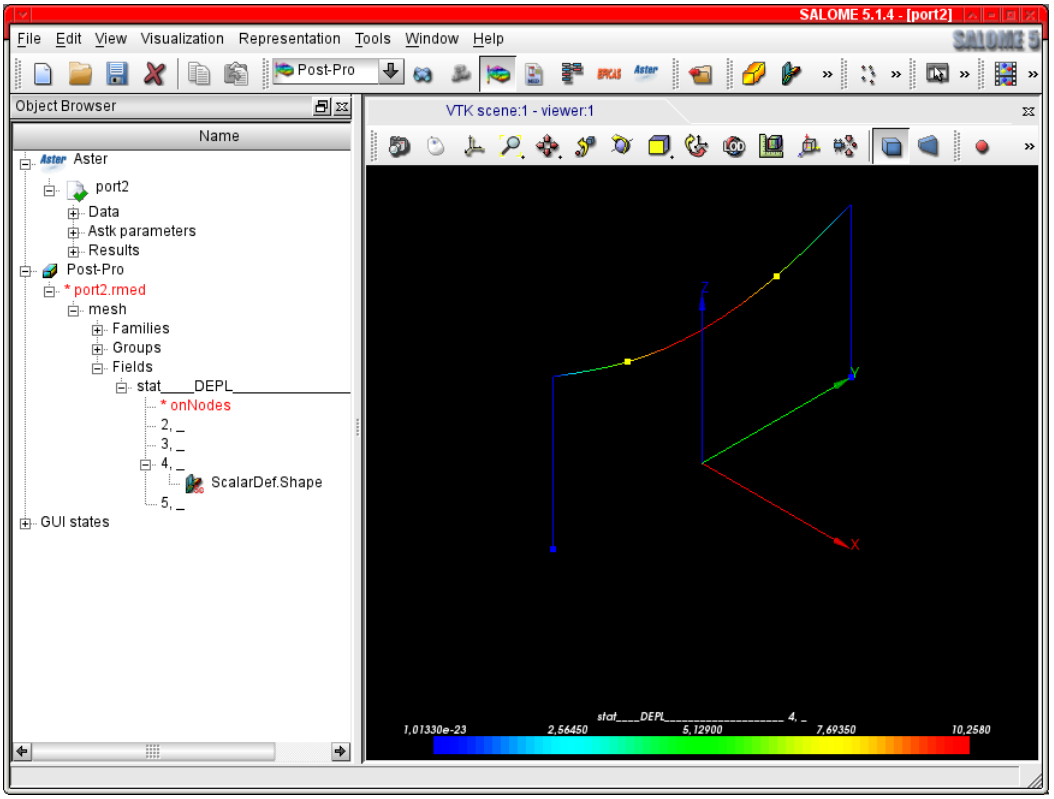


Figure 18: Top beam hinged at the ends

15 ADDING PLATE ELEMENTS, A MOTOR WAY SIGNAL FRAME

For this problem we modify the geometry by adding a second top bar 200 mm below the first one, joining these two bars by 5 vertical members, and filling the gaps with a steel plate..
To create a plane surface in Gmsh proceed like in figure 19:

Geometry>Elementary entities>Add>Plane Surface, pick up the boundary lines with the mouse, when finished type "e as instructed.
Surface appears like this in the .geo file.

```
Line Loop(25) = {17, 20, -5, -19};\newline
Plane Surface(25) = {25};\newline
```

The first line defines the loop, the second creates the surface.

Once the surfaces are created we must check their orientations, if we want to apply pressure for example, they have to have all the same orientation, to do this:
Menu Tools>Options>Geometry then enter a realistic value in the "Normals" field it should look like figure 20:

If one normal is badly oriented change the line like this in the .geo file::
from "Plane Surface(25) = 25;" change the sign of the loop to "Plane Surface(25) = -25;":

We can now mesh the structure with the extra "2D" step in the meshing, with the proper case ticked on the mesh should look like figure 21:

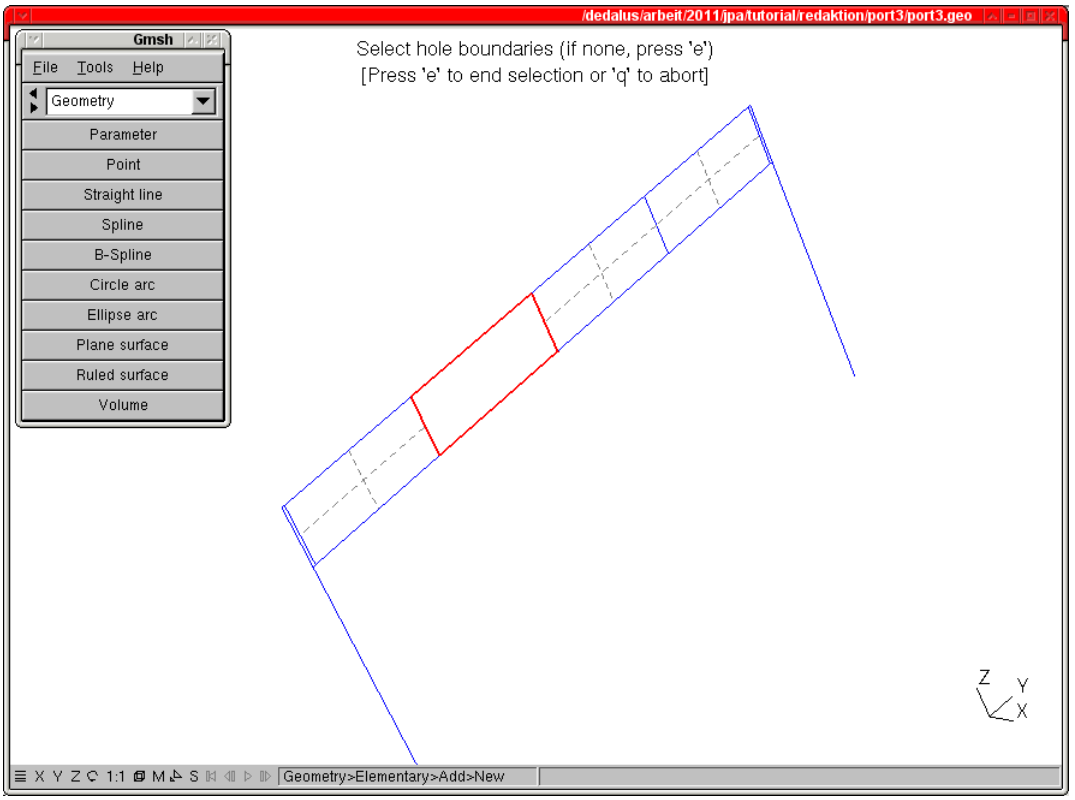


Figure 19: Creating a plane surface in Gmsh

Several notes::
We add a line in the .geo file so that each surface creation looks like this:

```
Line Loop(25) = {17, 20, -5, -19};
Plane Surface(25) = {25};
Recombine Surface {25};
```

This allows to create as many as possible quadrangular elements, instead of triangle that are defaults in Gmsh.

The rendering of the surface is possible only on the mesh not on the geometry.
Creating Line Loop by hand in the text editor must be done with an extreme care as it is easy to create an inverted loop on which Gmsh will crash after an error message.
Finally note that we can create a holed surface by following the Gmsh screen instructions.

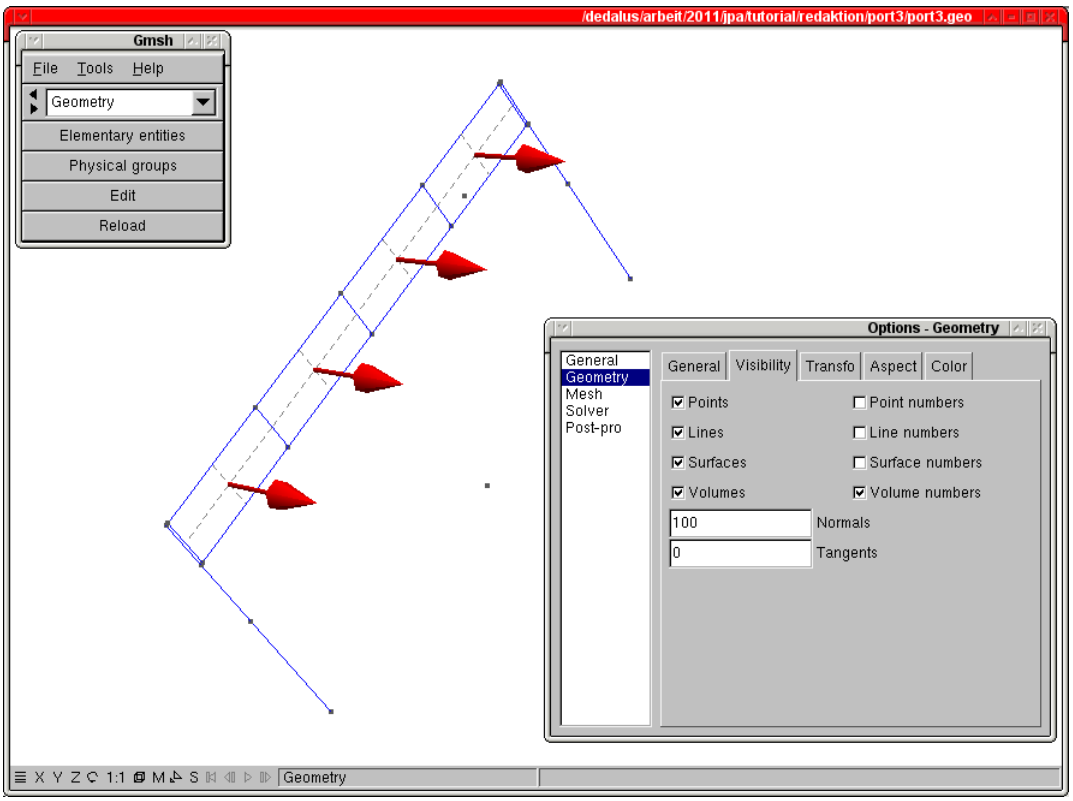


Figure 20: Checking orientation of normals to surfaces

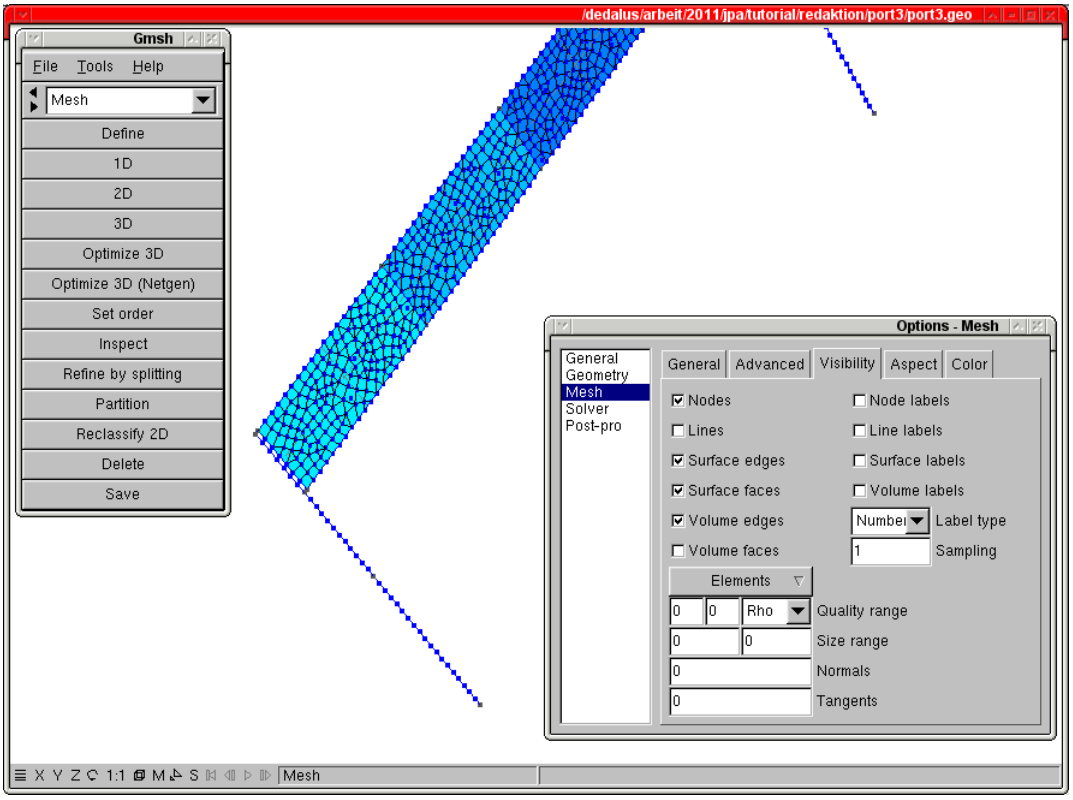


Figure 21: "Motorway signal" meshed

16 COMMANDING FOR PLATE ELEMENTS

Producing a .comm file for plates in quite straight forward for the first part.
In AFFE_MODELE:

```
#here is the modelling of plate element
                          _F(GROUP_MA=('panel',),PHENOMENE='MECANIQUE',
                             MODELISATION='DKT',),
```

In AFFE_CARA_ELEM:

```
#the plate is given a thickness of 3 mm and the orientation of the element is defined
#by VECTEUR see U4.42.01
                       COQUE=_F(GROUP_MA='panel',EPAIS=3,VECTEUR=(0,1,0),),
```

For the pressure load in AFFE_CHAR_MECA:

```
cv=AFFE_CHAR_MECA(MODELE=model,
#here we define a pressure on the plate elements
                  PRES_REP=_F(GROUP_MA=('panel'),PRES=0.001,),
#this line would have meant a distributed fore along x i.e. equivalent
#                  FORCE_COQUE=_F(GROUP_MA=('panel'),FX=0.001,),
                 );
```

We then organize for this horizontal pressure to be the fourth load case, for that look at "port3.comm" in "WORKED EXAMPLES".

The part concerning the calculation is a little bit trickier, as we want to calculate the stresses on the face of the plates, not only at the neutral fiber, as they are subject to a pressure.
First we calculate a new concept "statsup" with CALC_ELEM:

```
statsup=CALC_ELEM(MODELE=model,CHAM_MATER=material,CARA_ELEM=elemcar,
                  GROUP_MA='panel',
                  RESULTAT=stat,
#to be able to calculate 'EQUI_ELNO_SIGM' next line must be commented??
#                  TYPE_OPTION='SIGM_STRUCT',
                  REPE_COQUE=_F(GROUP_MA='panel',NIVE_COUCHE='SUP',ANGL_REP=(0.,1.),),
                  OPTION=(
#                         'SIEF_ELNO_ELGA',
                          'SIGM_ELNO_DEPL',
#'EQUI_ELNO_SIGM' is necessary to later calculate 'EQUI_NOEU_SIGM'
                          'EQUI_ELNO_SIGM',
                          ),
                    );
```

Then we perform the same type of calculation for nodes with the computation of 'EQUI_NOEU_SIGM' which gives various criteria like Von Mises or Tresca:

```
statsup=CALC_NO(reuse =statsup,RESULTAT=statsup,
                GROUP_NO_RESU =('panel',),
                OPTION=('SIGM_NOEU_DEPL',
                        'EQUI_NOEU_SIGM',),);
```

Once this results are calculated they may be printed in the right files for post processing.

And Finally we will explore some new areas:
At the end of the file we add the following lines:

```
#here we include another command file
#U4.13.01
INCLUDE (UNITE=91, INFO=2)
#here we launch STANLEY
STANLEY()
```

INCLUDE will include another .comm file, writing a .pos results file[5].
STANLEY will launch STANLEY post processing macro.
While STANLEY would run correctly from a Salome-Meca "Run" action the INCLUDE commad will only be interpreted in a ASTK run.

17 INTRODUCING ASTK FOR THE ANALYSIS

The ASTK handling is well described in U1.04.00, however here is my way through:
In Salome-Meca, in the browser right click in "Aster, problem name" then choose "Export to ASTK" a window like figure 22 appears:.

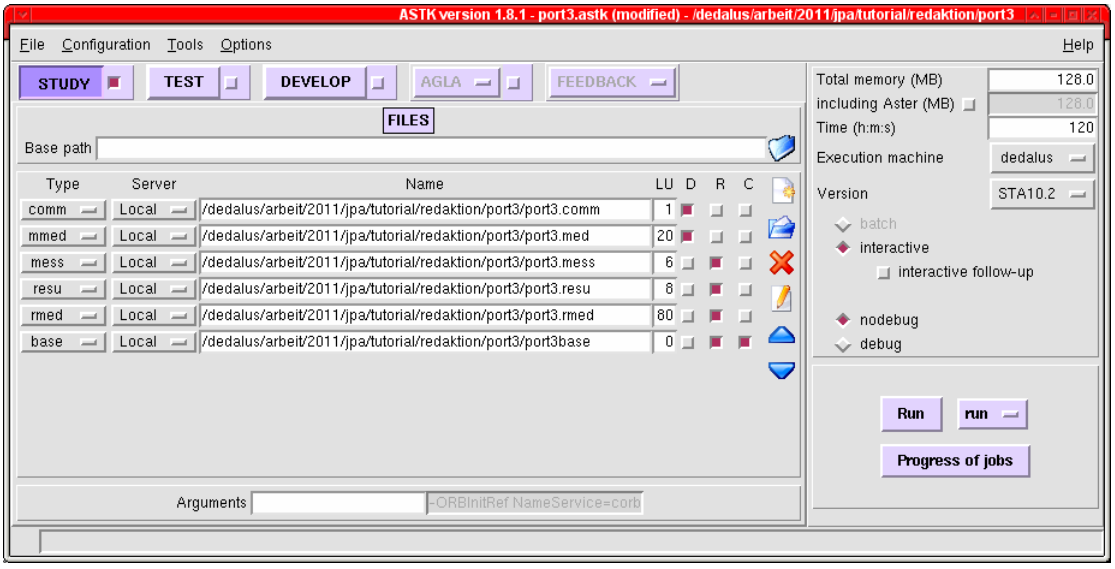


Figure 22: ASTK window

On the top part, right of "Base path" field, click on the icon looking like a directory, then navigate until we reach the problem directory and select it.

[5].pos is Gmsh Post Pro format file.

In the main part of the window, we can see the files of the problem, the one we entered previously in Salome-Meca and the ones created by Salome-Meca.
And a vertical column of icon on the right:
Let's click on the top one to create a new line.
On this line we will create the file for the mesh verification, referenced in the first lines of the "port3.comm".
Pull down the name list on the extreme left and choose "mmed" to specify it as a med file.
Name it "./port3verif.med".
In the column "LU", Logical Unit" change 20 to 71 that we have specified in "port3.comm" file.
Uncheck the column "D" that stands for data and check the column "R" that stands for results as we want to write in this file.

We will also create a new line with reference pos, name "port3.pos", LU=37 with R ticked, to contains the results in Gmsh Post Pro format.

Finally a last line with comm, name "post3.comm2", LU=91 with D ticked, pointing to the command file we want to chain.

Push the button "Run", another window named "ASJOB***" should open like in figure 23

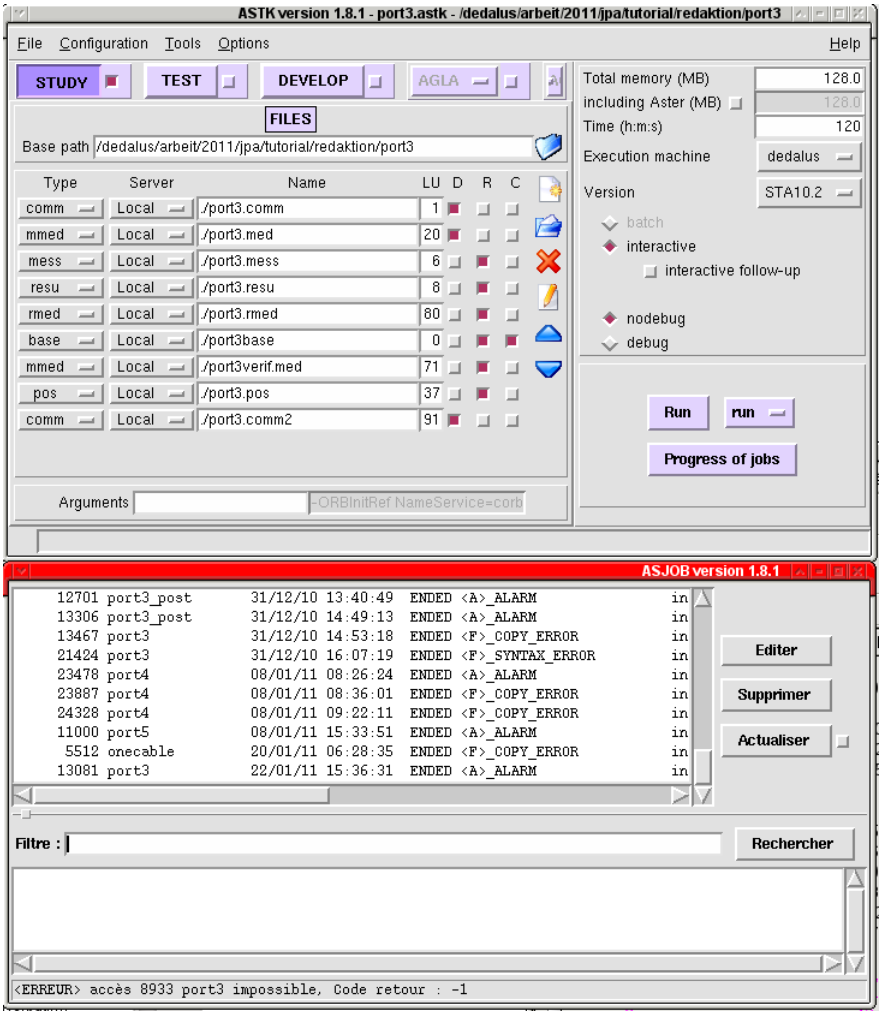


Figure 23: ASTK and ASJOB windows, job 'ENDED' with 'ALARM'

With at the bottom a line with a job ID, our job name, date and time and the mention "PEND" that's stand for pending, I am waiting! Push "Actualiser" that is update.
Wait a bit until we can read on the line something like "ENDED OK" or "ENDED < A >_ALARM" meaning that the calculation is finished.

If not, maybe like some of the lines of the above picture, something went wrong and the procedure of modifying the .comm file is no different from previously.
Looking in a file manager we can see that the file "port3verif.med" has been created, we can open it with Salome-Meca or Gmsh yo check it looks like expected.
A file "port3.pos" has also been created, more about it in a few lines.

ASTK is a powerful tools offering a more sophisticated handling of jobs, like chained .comm entries and multiples output files and much more...
However we have to go back to Salome-Meca to open the .med results file to see how the results look like. And reopen it every time a new analysis is performed as there is no dynamic link, updating the results, this can get confusing sometimes.
Are the results we are looking at really the one of the last analysis? no real way to be strictly sure, but looking at file date and times in a file manager!

A few more notes about ASTK:
On the right side of the ASTK window we can change the "Total memory" and the "Time" allocated to the job.
In the same conditions as stated within Salome-Meca above we may try to check the "interactive follow-up" box, if it works, the "ASJOB***" may not open but instead a terminal showing us what's being done in the job.

There exists in the *Code_Aster* list of tools one that will do the job of post processing on the fly, that is the macro command STANLEY

18 USING STANLEY, A QUICK APPROACH TO POST PROCESSING

The document U4.81.31 explain the use of STANLEY.
Calling STANLEY is simple just add at the end the .comm file, just before "FIN()" the following line:
STANLEY()
Run the problem again, a window like figure 24 will pop up:

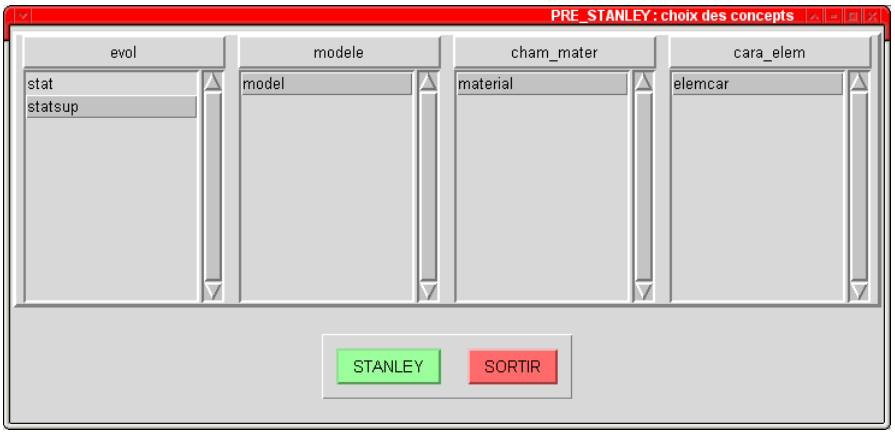


Figure 24: STANLEY window

First we will set some parameters in the menu Parametres > Editer, toggle the "Mode" swith to "Gmsh/Xmgrace" the button "OK", figure 25.
We could have left the Mode on "Salome", but it's more fun to start in Gmsh mode![6].

[6]It is also the default, and only way to proceed if you have a stand alone *Code_Aster* install without Salome

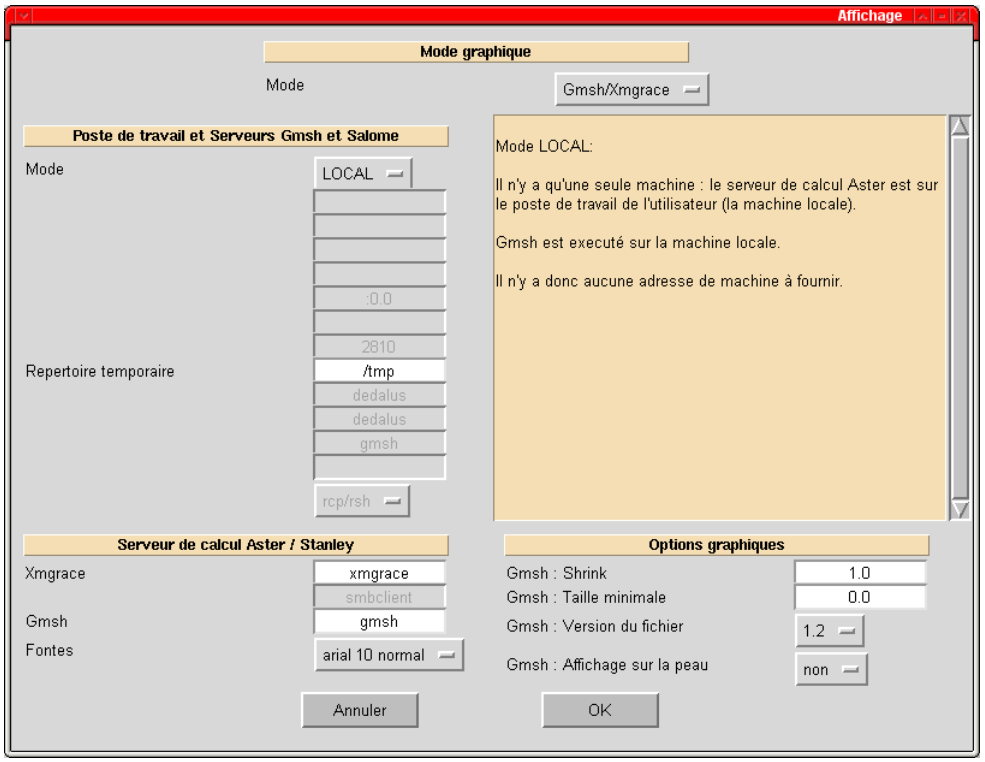


Figure 25: STANLEY parameters set to "Gmsh/Xmgrace" mode

Then select the selected item as in figure 26, push "STANLEY".

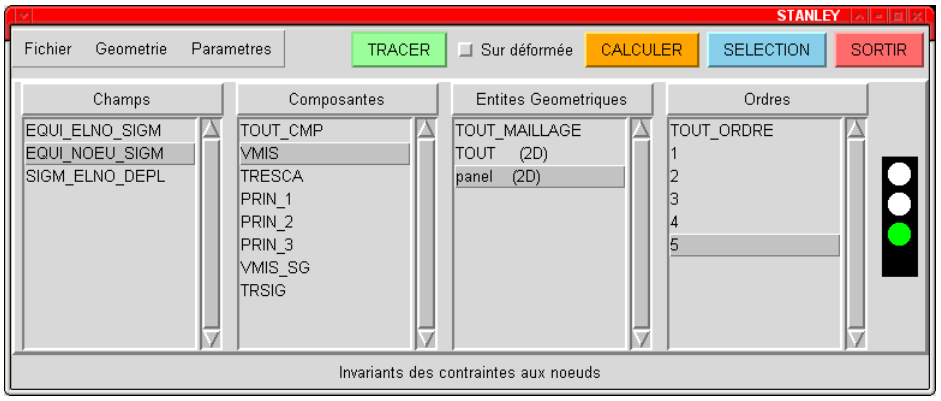


Figure 26: STANLEY window, selection made

On the left column named "Champs" (fields) select "EQUI_NOEU_SIGM",
On the next column named "Composantes" (components) select "VMIS",
On the next column named "Entites Geometriques" (geometric entities) select "panel (2D)",
On the right most column named "Ordres" select "5" (that is the load case at order 5 in Aster)
The window now looks like figure 26.[7].
On the extreme right the traffic light is green, we can push "TRACER"
Had the traffic light been orange we have had to push "CALCULER" so as to calculate the field and turn the light to green.
Had the light been red, then the requirements could not be calculated.
In our our case the light is green.

[7]the label "Sur déformée", On deformed shape, of the check box in between "TRACER" and "CALCULER" seems to have vanished away on my newer versions.

Lets go, push "TRACER"

Humm! nothing happens, then read in the paragraph "CORRECTING INSTALLATION MISHAP", once corrected or if all go well then a window like 27 appears:

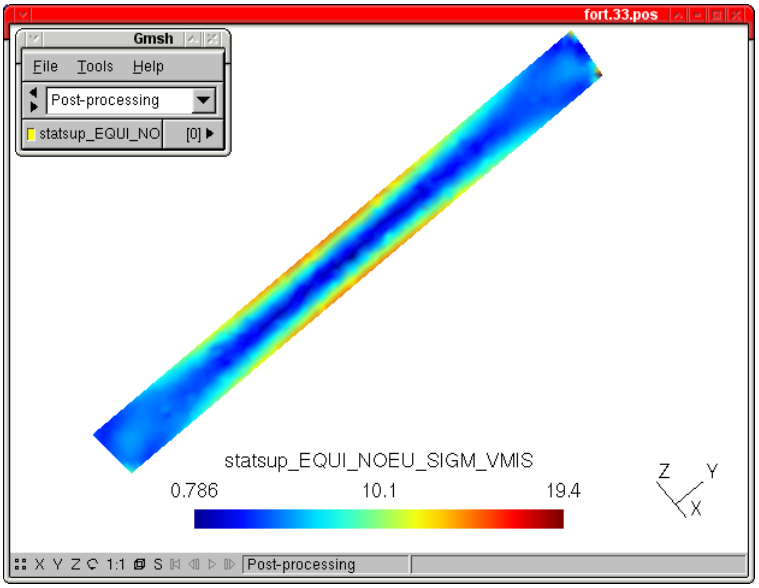


Figure 27: Von Mises criteria in "panel" element in Gmsh post pro View

And we can see a Gmsh Post-processing view.
I am cheating a bit, as at first the selected view is the xOz plane, and our model has a null y dimension, so we have to turn it a bit, to view something.

A useful hint for post pressing view in Gmsh is as follow:

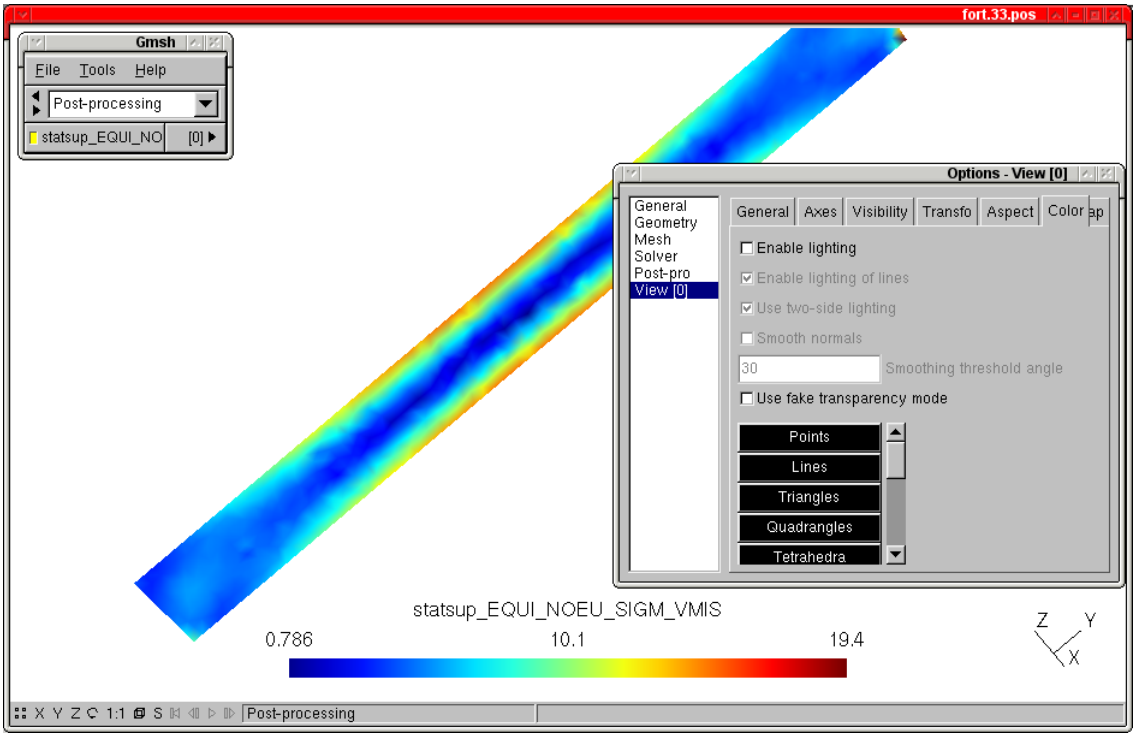


Figure 28: Lighting set to off

In the Gmsh command window, we choose the view we want in the view list (here we have only one, "statsup_EQUI_NO...") push the arrow on the right, then Options, Color tab, then uncheck "Enable lighting", like in figure 28.
With this the color mapping of results is independent of any light source, otherwise it becomes unreadable in areas which are in the shade from the light source.
In the Gmsh command window, menu File > Save Default Options will save this setting for all further Gmsh work.
One annoying draw back of Gmsh post processing with STANLEY is that the groups of 1D elements like beams or rods are not present as groups in the file, while the groups of 2D are.

There is a lot to play with concerning the appearance of the plot in Gmsh, this can set by push the arrow on the right of the View , then Options, a lot of them are available there.

If we play with the "- +" list named "Time step" we will notice that Gmsh has its own time stepping always starting a 0 with steps one by one, whatever the stepping of "INST" stated in the .comm file, when the right number of "INST" is displayed in the scalar bar title, curious isnt it!.

Important note about STANLEY: we have to quit STANLEY by pushing "SORTIR" so as the .mess and .resu and all the results files will be saved on disk.

And even more IMPORTANT Salome-Meca (if ASTK was not run from it) is not aware of the change made in the .med file, so we have to reload it MANUALLY every time in the Post-Pro module!
And the browser in Salome-Meca Pro-Pro will soon show so many results entries that it is really possible to get lost and look at the wrong results!

And a final remark, Stanley can be called just after the calculation, MECA_STATIQUE here, is made, before any CALC_ELEM or CALC_NO, as STANLEY call these functions by itself whenever necessary.
In fact the comm file can be ended here in an early design stage!

Another way to launch STANLEY is: in the ASTK window right click on the ligne with the "base" then "OPEN with.."> "Post-processing using Code_Aster (Stanley)", this will relaunch STANLEY.

19 USING Gmsh FOR POST PROCESSING, WHY NOT?

Once a .pos file is saved it is possible to open with Gmsh, viewing is similar as what we have described before.
With many fields, the first window will be very cluttered at first look!

Gmsh is GPL, it works on any platform (or almost), so it is easy to save a .pos file send it by email it to somebody who has to look at our results.
With a few instructions about how to use same Gmsh, which be can found here for example, this guy will get a much better picture of the results than a few fixed screen copies.
Together with the .geo, .msh, .comm, and .resu file we have a set of file fully describing the problem and readable on any platform.

I have done this quite often with classifications societies.

One of the main draw back of Gmsh .pos file is that the groups are not present in the file.

20 STIFFENING IT WITH RODS

Back to engineer problems.
We may found that that this structure needs stiffening against horizontal loads, so we will add four stiffening rods downward from the top, of the masts.
These rods are what is called "BARRE" in *Code_Aster* terminology.
These elements will transmit axial forces, either tension or compression, but no end moments, of course the real building must be designed and built this way.
To handle correctly these elements *Code_Aster*, like any finite element code need that they is a single element from one end attachment to the other.
Gmsh handles these lines like that:.

```
//next line create the group for rod
Physical Line("rod") = {101, 102, 103, 104};
//next line set the meshing so as to have only one element on each rod
Transfinite Line {101, 102, 103, 104} = 0 Using Progression 1;
```

Once meshed the model looks like figure 29

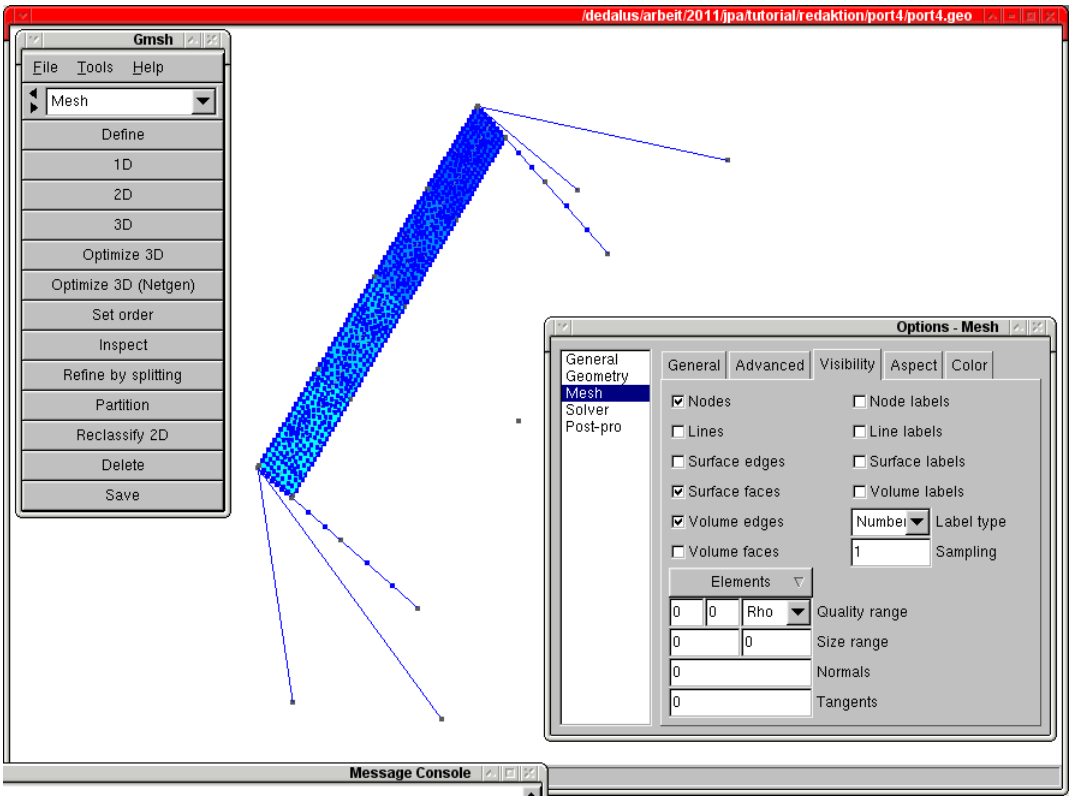


Figure 29: Motor way signal with rod support

Command file will need adding the following lines:

```
model=AFFE_MODELE(....
#here is the modelling of topbeam element
                        _F(GROUP_MA=('rod',),PHENOMENE='MECANIQUE',
                           MODELISATION='BARRE',),
.....
elemcar=AFFE_CARA_ELEM(....
```

```
#the rods are round pipe od 32 mm thickness 1.5 mm
                      BARRE=(_F(GROUP_MA=('rod',),
                                SECTION='CERCLE',CARA=('R','EP',),
                               VALE=(16,1.5,),),
                              ),
....
ground=AFFE_CHAR_MECA(MODELE=model,
                       DDL_IMPO=(_F(GROUP_NO=('groundS','groundN',),
                                  DX=0,DY=0,DZ=0,DRX=0,DRY=0,DRZ=0,),
#this for the ground connection of rod
                                  _F(GROUP_NO=('groundR',),DX=0,DY=0,DZ=0,),
#this line would have raised an error,
#DDL for rod only ends can be fixed in translation only
#                                  _F(GROUP_NO=('groundR',),
#                                     DX=0,DY=0,DZ=0,DRX=0,DRY=0,DRZ=0,),
                                 ),
                       );
....
```

And for the results printing things get complicated due to some restrictions of the MED file format, the following abstract is self explaining:

```
IMPR_RESU(MODELE=model, FORMAT='MED', UNITE=80,
          RESU=(_F(GROUP_MA=('topbeam','topbeamver','mast','panel',),
                   RESULTAT=stat,
                   NOM_CHAM=('DEPL',
                             'SIEF_ELGA_DEPL','SIPO_ELNO_DEPL','SIGM_ELNO_DEPL',),
                   ),
#next lines will raise an error in writing med file, see below
#                _F(GROUP_MA=('rod',),RESULTAT=stat,NOM_CHAM=('DEPL',
#                   'SIEF_ELGA_DEPL'),),
                 _F(GROUP_MA=('panel',),RESULTAT=statsup,NOM_CHAM='SIGM_ELNO_DEPL',),
                 _F(GROUP_NO=('panel',),RESULTAT=statsup,
                    NOM_CHAM=('SIGM_NOEU_DEPL','EQUI_NOEU_SIGM',),
                   ),
                ),
           );

#to get post pro results in a med file
#for BARRE and CABLE we need to write a second med file
#only 'SIEF_ELNO_ELGA' is useful here
#this needs ASTK processing to print the two .med file
#but STANLEY will show results after a simple Salome-Meca run!!
IMPR_RESU(MODELE=model, FORMAT='MED', UNITE=81,
          RESU=_F(GROUP_MA=('rod'),
                  RESULTAT=stat,
                  NOM_CHAM=('SIEF_ELNO_ELGA',),
                 ),
          );
```

Here is a screen view of the 'SIEF_ELNO_ELGA' of the results, figure 30.

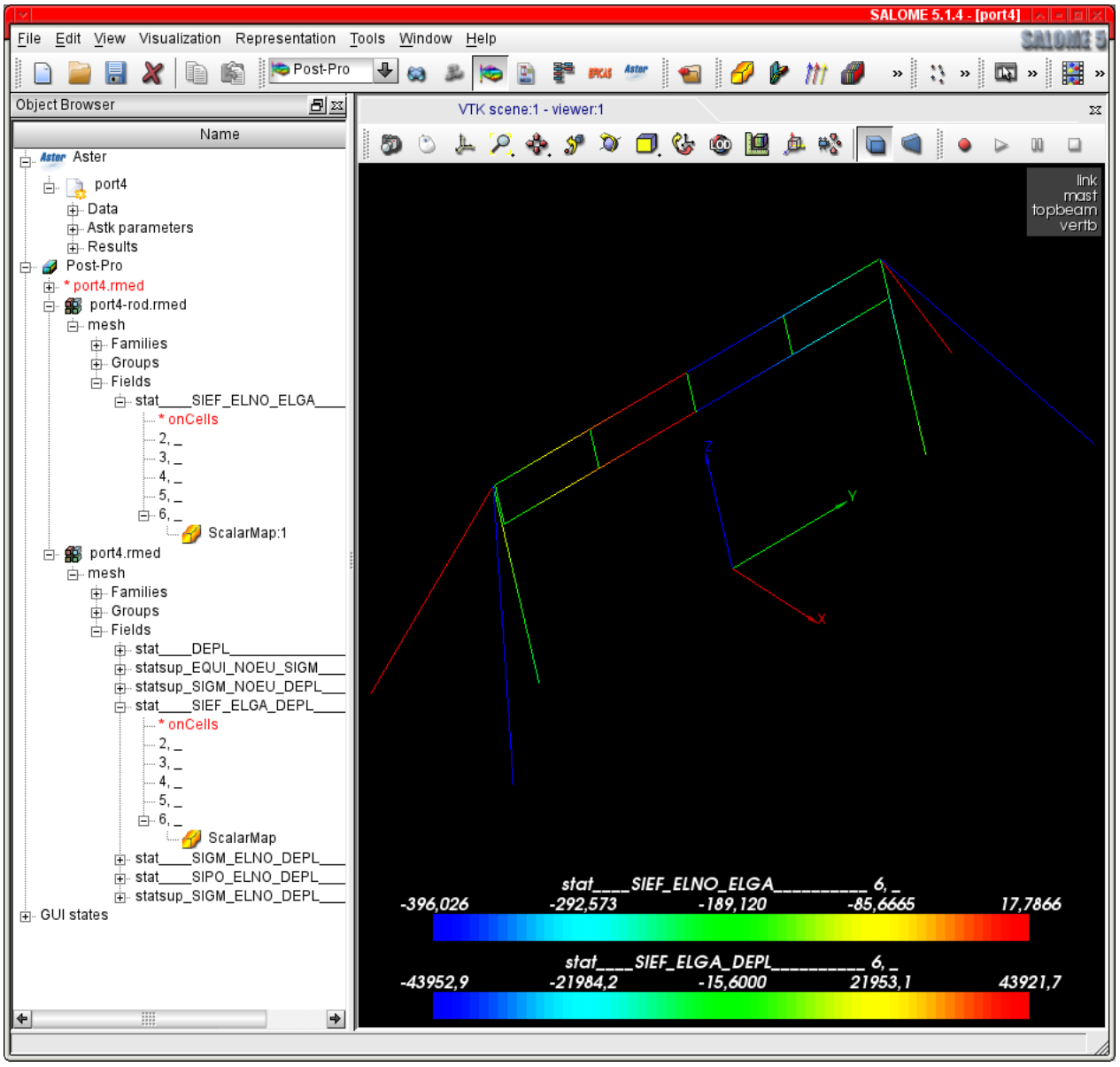


Figure 30: Results with 2 views superimposed

Note the two imported results file “port4-rod.rmed” and “port4.rmed:1” in the browser.
The two views are superimposed in the drawing of the structure, while the two scalar bars are offset.
The top Scalar Bar shows the color for 'N' value in the rods, the bottom one shows the 'MFZ' value in the beams.
But as there is nothing written about 'CMP' in the bars themselves, it is not easy to understand, neither easy to explain to somebody else..
A major drawback of Salome-Meca Post-Pro module!.
Files for “port4“ problem are included in ”WORKED EXAMPLES”.

21 REPLACING ROD BY CABLES,THE NON LINEAR APPROACH

We can see that in the previous model some rods are in tension while the other are in compression.
If we want to replace the rods by cables we need a different approach as the cables will not carry any compression load.
We have to perform a non linear analysis.
This is quite easy.
We will create a "port5.geo" file replacing the group name "rod" by cable, for the shake of clarity.
We will also changee the "Transfinite" command allowing to mesh the cables with 20 elements along their length.
Of course meshing and saving "port5.med".

Roughly speaking a non linear analysis is performed in many calculation steps, computing the stiffness of each element, at every step on the deformed shape.
In the case of cable they be will kind of "eliminated" when they come in compression.
There are many parameters available in non linear analysis, here we will only scratch the surface.
One last remark before starting:
A linear calculation will always give a result, but it may be crazy, with, for example deformations exceeding the model dimensions, that is because the calculation is made only once, on the undeformed shape.
In a non linear the calculation is made in many steps, and the calculation may stop at one step in the middle, we then need to refine some parameters an try it again.
A close look at the .mess file is here of great help.
Non linear analysis may become a rather tedious involvement.

22 COMMANDING FOR NON LINEAR ANALYSIS

Non linear calculation needs some modifications to the command file, we will show them now.

```
model=AFFE_MODELE(MAILLAGE=mesh,
AFFE=(....
#here is the modelling of cable element
                          _F(GROUP_MA=('cable',),PHENOMENE='MECANIQUE',
                             MODELISATION='CABLE',),
....
```

Most of the construction work with cables require some pretension being applied in the cable, here we do it by cooling them:

```
#here we describe a temperature field that will allow us to cool the cable
#thus putting some tension in them
temper1=CREA_CHAMP(TYPE_CHAM='NOEU_TEMP_R',OPTION='SIEF_ELNO_ELGA',
                   OPERATION='AFFE',MODELE=model,
                   AFFE=_F(GROUP_MA=('cable',),NOM_CMP='TEMP',VALE=-100.0,),
                   );
....
#here we describe a material for the cables, it has
#ALPHA  for temperature elongation
#CABLE=_F(EC_SUR_E=1.E-4,) ratio of E compression divided E traction to avoid cables
#transmitting compression forces
```

```
#U4.43.01
msteel=DEFI_MATERIAU(ELAS=_F(E=100000,NU=0.3,RHO=8e-9,ALPHA=12e-6,),
                     CABLE=_F(EC_SUR_E=1.E-4,),);

material=AFFE_MATERIAU(MAILLAGE=mesh,
                       AFFE=(_F(GROUP_MA=('topbeam','topbeamver','mast','panel',),
                                MATER=steel,),
#material to cables
                             _F(GROUP_MA=('cable',),MATER=msteel,),
                             ),
#here we apply the temperature to the sructure, the difference between this temperature
#and the one above makes the pretension
                       AFFE_VARC=_F(TOUT='OUI', NOM_VARC='TEMP',
                                    CHAMP_GD=temper1,VALE_REF=0.0,),
                        );
....
```

Strictly speaking this simple model would have converged without any preload in the cable, and the real construction would not need either a sugnificant pretension, considering the loads level.
Concerning the value the cable has α=12e-6 with ΔT=-100°which gives ΔL/L=-0.0012, and if constrained a force F=ΔL/L*E*SECTION=-1200N or 120 N/mm^2.

```
elemcar=AFFE_CARA_ELEM(MODELE=model,
                       ....
#description of cable properties,only section is required,
#N_INIT=10.0 is a numerical pretension in the cable so the calculation is possible
#this is not seen in the results
                       CABLE=_F(GROUP_MA=('cable',), N_INIT=10.0, SECTION=(10,),),
                       ....
```

The linear analysis in *Code_Aster* allows very many options and parameters here is the command that will solve our problem:

```
#we may need tinkering around with PAS until the problem converges
liste=DEFI_LIST_REEL(DEBUT=2.0,INTERVALLE=_F(JUSQU_A=6,PAS=1.0,),);
#we may also want a more restricted list at which calculate and or print results
listresu=DEFI_LIST_REEL(DEBUT=2.0,INTERVALLE=_F(JUSQU_A=6,PAS=1.0,),);

#here is the non linear analysis see
#U4.51.03
#U4.51.11
statnl=STAT_NON_LINE(MODELE=model,
                     CHAM_MATER=material,
                     CARA_ELEM=elemcar,
                     EXCIT=(_F(CHARGE=ground,),
                            _F(CHARGE=selfwght,FONC_MULT=selfw_m,),
                            _F(CHARGE=cc,TYPE_CHARGE='FIXE_CSTE',FONC_MULT=cc_m,),
                            _F(CHARGE=cr,TYPE_CHARGE='FIXE_CSTE',FONC_MULT=cr_m,),
                            _F(CHARGE=cv,TYPE_CHARGE='FIXE_CSTE',FONC_MULT=cv_m,),
                            ),
#the beam and plates parts are allowed to deform in 'PETIT', small mode
```

```
                    COMP_INCR=_F(RELATION='ELAS',DEFORMATION='PETIT',
                                 GROUP_MA=('topbeam','topbeamver','mast','panel'),),
#the cables parts are allowed to deform in 'GROT_GDEP', large rotation, large displacement
                    COMP_ELAS=_F(RELATION='CABLE',DEFORMATION='GROT_GDEP',
                                 GROUP_MA=('cable',),),
                    INCREMENT=_F(LIST_INST=liste,),
#the resolution method
                    NEWTON=_F(PREDICTION='TANGENTE',MATRICE='TANGENTE',REAC_ITER=1,),
#this trick speeds thinks up
                    RECH_LINEAIRE=_F(),
#how do we consider the calculation is finised at every step
#that is a sort of quality criteria as well
                    CONVERGENCE=_F(RESI_GLOB_RELA=1e-4,ITER_GLOB_MAXI=300,),);
```

With the relevant options for the calculations of results for our problem:

```
statnl=CALC_ELEM(reuse =statnl,MODELE=model,CHAM_MATER=material,CARA_ELEM=elemcar,
             RESULTAT=statnl,
             REPE_COQUE=_F(GROUP_MA='panel',ANGL_REP=(0.,1.),),
             OPTION=(
#not allowed in non linear
#                     'SIPO_ELNO_DEPL','SIGM_ELNO_DEPL',
                      'SIEF_ELNO_ELGA',
                      'SIGM_ELNO_SIEF',
#next line required in non linear
                      'SIGM_ELNO_COQU',
                      'SIPO_ELNO_SIEF',
                      ),
             );

statnlsp=CALC_ELEM(MODELE=model,CHAM_MATER=material,CARA_ELEM=elemcar,
                   GROUP_MA='panel',
                   RESULTAT=statnl,
                   REPE_COQUE=_F(GROUP_MA='panel',NIVE_COUCHE='SUP',ANGL_REP=(0.,1.),),
                   OPTION=(
#next line required in non linear
                           'SIGM_ELNO_COQU',
#'EQUI_ELNO_SIGM' is necessary to later calculate 'EQUI_NOEU_SIGM'
                           'EQUI_ELNO_SIGM',
                            ),
                   );

statnlsp=CALC_NO(reuse =statnlsp,RESULTAT=statnlsp,
                 GROUP_NO_RESU =('panel',),
                 OPTION=('EQUI_NOEU_SIGM',),);

statnl=CALC_NO(reuse =statnl,RESULTAT=statnl,
               GROUP_NO_RESU=('groundS','groundN','approd'),OPTION=('REAC_NODA',),);
```

Printing the results is no different of what we have shown previously keeping in mind that we can print only what's been calculated before.

The results show as follows, figure 31:

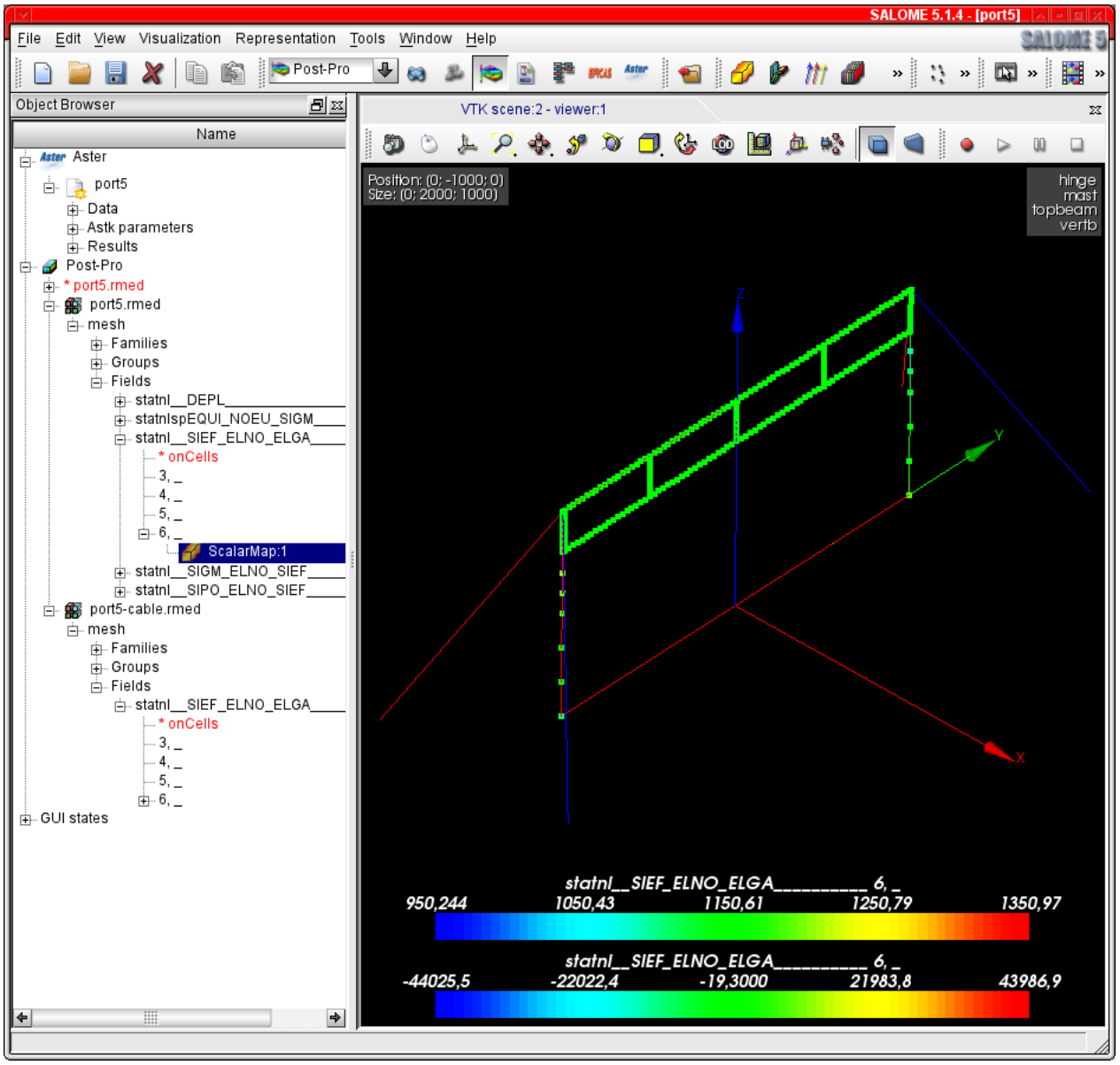


Figure 31: Results with cables

A look at the .mess file shows us that the calculation is done in hardly more than one iteration at every step, which means that the behavior of the structure is almost linear.
That is what we expected any way, here the non linear calculation is made to take into account the cable behavior, and we can actually see that the load carried by the cable is pretty different of the one carried by the rods.
The "leeward" side cable being still in a slight tension, a remaining of the preload.

Note also the curious feature of Salome showing the "Gauss points" in beams for for the 'SIEF_ELNO_ELGA' field which is not making things very understable at this scale!

23 REPLACING THE TOP BAR BY A SUSPENSION CABLE

We will know model a structure with two vertical masts, supported by 4 angled cable shrouds, an horizontal cable extended in between the the two mast stops.
The geometry looks like figure 32:

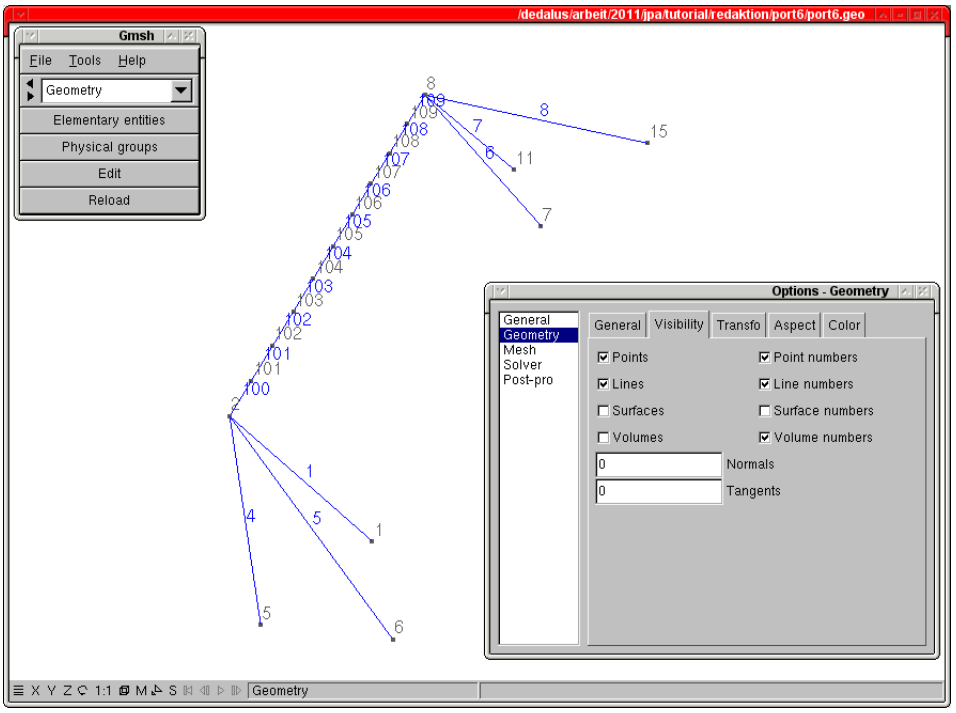


Figure 32: Geometry ready for cycling

With is the corresponding .geo file:

```
cl1=100;
Point(1) = {0, -1000, 0, cl1};
Point(2) = {0, -1000, 1000, cl1};

Point(5) = {-500, -1500, 0, cl1};
Point(6) = {500, -1500, 0, cl1};
Line(1) = {1, 2};
Line(4) = {5, 2};
Line(5) = {6, 2};
Symmetry {0, 1, 0, 0} {
  Duplicata { Line{1, 4, 5}; }
}

Physical Line("mast") = {1, 6};
Physical Line("shroud") = {4, 5, 7, 8};
Physical Point("mastgrd") = {1, 7};
Physical Point("shrgrd") = {5, 6, 11, 15};
//this loop creates the points along the top cable
//notice that Point 100 doubles with Point 2, Point 110 with Point 8
For i In {0:10:1}
   Point(i+100)={0,-1000+200*i,1000,cl1};
```

```
EndFor
//this loop creates the top cable section
For i In {0:9:1}
   Line(i+100)={i+100, i+100+1};
EndFor
//this loops creates individual Pysical Point along the cable
//each one with a "logical" name
For i In {0:10}
Physical Point (Sprintf("cycl%02g",i)) = {i+100};
EndFor
//this loop creates the Physical Line describing the top cable
lg[]={};
For i In {0:9}
  a[]={(i+100)};
  lg[]+=a[];
EndFor
Physical Line ("cblcy") = lg[];
Transfinite Line {lg[]} = 0 Using Progression 1;
//This line removes the double points at geometry level
Coherence;
//This line removes doubles Nodes, once meshed,
//experiment the difference!!
//Coherence Mesh;
```

Notice the loop that creates the points 100 to 109, along the top cable, the lines joining these points. Notice also the loop that creates one named Physical Point for each one of the above points.
And also the "Coherence" command that removes the double points, with its fellow "Coherence Mesh" that does not do exactly the same things, experiment with it.

Once meshed our structure looks like figure 33:

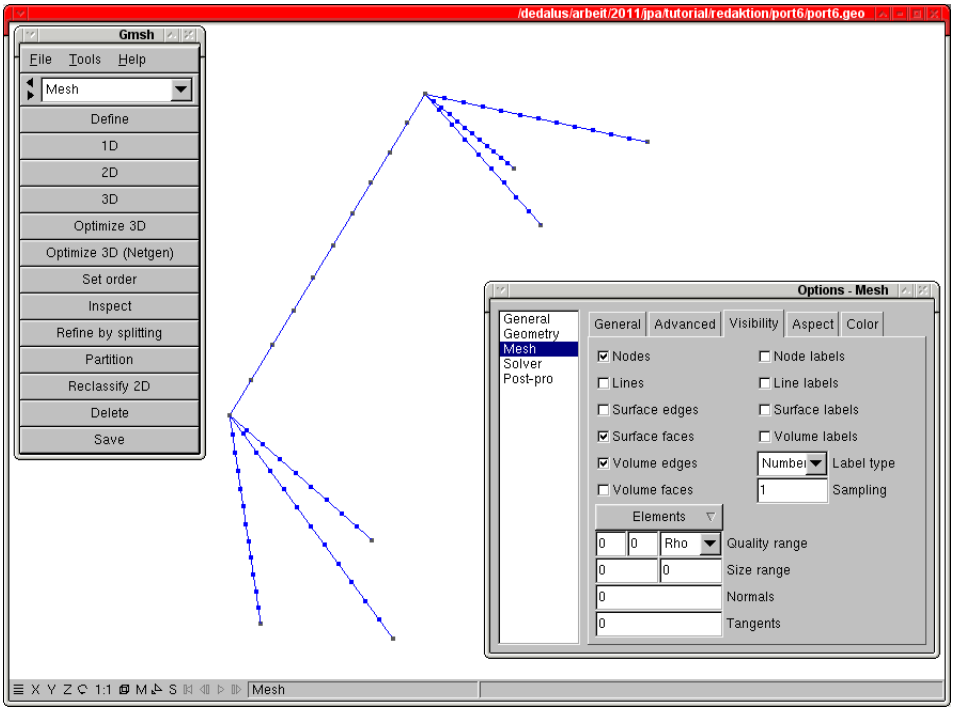


Figure 33: Now meshed

24 CYCLING ON THE CABLE, LIKE A CLOWN!

Now we will look at this model with a clown cycling along the cable.
This is done by applying the load in a saw teeth manner on the node groups previously created, in relation to time.
Note that we apply the clown load on single node, we suppose he is using a one wheel vehicle, easier for us, trickier for him.

———————-

```
DEBUT(
#next line is only useful for more sophiscated Python call within the .comm file
#U4.11.1
#      PAR_LOT='NON',
      );

mesh=LIRE_MAILLAGE( INFO=1,
#                    INFO_MED=2,
                     UNITE=20,FORMAT='MED',);

mesh=DEFI_GROUP(reuse =mesh,MAILLAGE=mesh,
                CREA_GROUP_MA=_F(NOM='TOUT',TOUT='OUI',),
                CREA_GROUP_NO=(_F(TOUT_GROUP_MA='OUI',),),
               );

IMPR_RESU(FORMAT='MED', UNITE=71,  RESU=_F(MAILLAGE=mesh,),);

model=AFFE_MODELE(MAILLAGE=mesh,
                  AFFE=(_F(GROUP_MA=('mast',),PHENOMENE='MECANIQUE',
                           MODELISATION='POU_D_T',),
                        _F(GROUP_MA=('cblcy','shroud',),PHENOMENE='MECANIQUE',
                           MODELISATION='CABLE',),
                       ),
                   );

#putting preload in the vertical cables 'shroud' is enough
temper1=CREA_CHAMP(TYPE_CHAM='NOEU_TEMP_R',OPTION='SIEF_ELNO_ELGA',
                   OPERATION='AFFE',MODELE=model,
                   AFFE=_F(GROUP_MA=('shroud',),NOM_CMP='TEMP',VALE=-100.0,),
                  );

steel=DEFI_MATERIAU(ELAS=_F(E=210000.,NU=0.3,RHO=8e-9),);
cablst=DEFI_MATERIAU(ELAS=_F(E=100000,NU=0.3,RHO=8e-9,ALPHA=12e-6,),
                             CABLE=_F(EC_SUR_E=1.E-4,),);

material=AFFE_MATERIAU(MAILLAGE=mesh,
                       AFFE=(_F(GROUP_MA=('mast',), MATER=steel,),
                             _F(GROUP_MA=('cblcy','shroud',), MATER=cablst,),
                             ),
#putting preload in the vertical cables 'cabnlev' is enough
                       AFFE_VARC=_F(GROUP_MA=('shroud',),
                                    NOM_VARC='TEMP',CHAMP_GD=temper1,VALE_REF=0.0,),
                      );
```

```
elemcar=AFFE_CARA_ELEM(MODELE=model,
                       POUTRE= _F(GROUP_MA=('mast',),
                                  SECTION='RECTANGLE',CARA=('HY','HZ','EP',),
                                  VALE=(100, 50, 5,),),
#mast section is increased compared to other examples, it would not stand the load
#                       ORIENTATION=_F(GROUP_MA=('mast',),CARA='ANGL_VRIL', VALE=90.0,),
                       CABLE=_F(GROUP_MA=('cblcy','shroud',),N_INIT=10.0,SECTION=(10,),),
                      );

ground=AFFE_CHAR_MECA(MODELE=model,
                       DDL_IMPO=(_F(GROUP_NO=('mastgrd',),
                                    DX=0,DY=0,DZ=0,DRX=0,DRY=0,DRZ=0,),
                                 _F(GROUP_NO=('shrgrd',),DX=0,DY=0,DZ=0,),
                                ),
                     );

selfwght=AFFE_CHAR_MECA(MODELE=model,
                        PESANTEUR =_F(GRAVITE=10000,DIRECTION=(0,0,-1),
                                      GROUP_MA=('mast',),),
                                      );

#in the next line we apply a verical load of 100 N on each of the group of
#nodes (1 node in each group here) on the cable
#with a saw teeth style time stepping on the load so as to mimic a rolling load
#this is done a Python loop, note indent in loop
iter=9;
lc=[None]*(iter+1);
lcm=[None]*(iter+1);
for i in range (1,iter+1):
  grpno='cycl%02g' %i;
  lc[i]=AFFE_CHAR_MECA(MODELE=model,FORCE_NODALE=_F(GROUP_NO=(grpno,),FZ=-100,),);
  lcm[i]=DEFI_FONCTION(NOM_PARA='INST',VALE=(i-1,0, i,1, i+1,0,),
                       PROL_GAUCHE='CONSTANT',PROL_DROITE='CONSTANT',);

selfw_m=DEFI_FONCTION(NOM_PARA='INST',VALE=(0,0, 1,1,),PROL_DROITE='CONSTANT',);

liste=DEFI_LIST_REEL(DEBUT=0.0,INTERVALLE=(_F(JUSQU_A=1,PAS=0.2,),
                                           _F(JUSQU_A=11,PAS=0.2,),
                                          ),
                     );
#we may also want a more restricted list at which calculate and or print results
listresu=DEFI_LIST_REEL(DEBUT=1.0,INTERVALLE=_F(JUSQU_A=5,PAS=1.0,),);

#IMPR_RESU(MODELE=model, FORMAT='RESULTAT', RESU=_F(MAILLAGE=mesh,),);

#Python loop to create the argument 'loadr' passed to 'EXCIT'
#loadr is actually a list, or a tuple!
loadr=[];
loadr.append(  _F(CHARGE=ground,),  );
loadr.append(  _F(CHARGE=selfwght,FONC_MULT=selfw_m,),  );
for i in range (1,iter+1):
  loadr.append(  _F(CHARGE=lc[i],TYPE_CHARGE='FIXE_CSTE',FONC_MULT=lcm[i],),  );
```

```
statnl=STAT_NON_LINE(MODELE=model,
                     CHAM_MATER=material,
                     CARA_ELEM=elemcar,
                     EXCIT=loadr,
                     COMP_INCR=_F(RELATION='ELAS',DEFORMATION='PETIT',
                                  GROUP_MA=('mast',),),
                     COMP_ELAS=_F(RELATION='CABLE',DEFORMATION='GROT_GDEP',
                                  GROUP_MA=('cblcy','shroud',),),
                     INCREMENT=_F(LIST_INST=liste,),
                     NEWTON=_F(PREDICTION='TANGENTE',MATRICE='TANGENTE',REAC_ITER=1,),
                     RECH_LINEAIRE=_F(),
                     CONVERGENCE=_F(RESI_GLOB_RELA=1e-4,ITER_GLOB_MAXI=300,),
                     );

statnl=CALC_ELEM(reuse =statnl,MODELE=model,CHAM_MATER=material,CARA_ELEM=elemcar,
                  RESULTAT=statnl,TYPE_OPTION='SIGM_STRUCT',
                  OPTION=('SIEF_ELNO_ELGA','SIPO_ELNO_SIEF',),
                 );

statnl=CALC_NO(reuse =statnl,RESULTAT=statnl,GROUP_NO_RESU = ('mastgrd','shrgrd',),
               OPTION=('REAC_NODA',),);


sum_reac=POST_RELEVE_T(ACTION=_F(INTITULE='sum reactions',
                                 GROUP_NO=('mastgrd','shrgrd',),
                                 RESULTAT=statnl,NOM_CHAM='REAC_NODA',
                                 RESULTANTE=('DX','DY','DZ'),OPERATION='EXTRACTION',
                                 INST=(0, 1, 2, 3, 4, 5, 6, 7,),
#another to restrict the useful printing
#                                LIST_INST=listresu,
#a good way to produce a lot of maybe useless informations
#                                TOUT_ORDRE='OUI',
                                 ),
                          );
IMPR_TABLE (TABLE=sum_reac,)

IMPR_RESU(MODELE=model,
          FORMAT='RESULTAT',
          RESU=_F(NOM_CHAM='REAC_NODA',GROUP_NO=('mastgrd','shrgrd',),
                RESULTAT=statnl,LIST_INST=listresu,),);

IMPR_RESU(MODELE=model,
          FORMAT='RESULTAT',
          RESU=(_F(NOM_CHAM='SIEF_ELNO_ELGA', GROUP_MA=('mast',), RESULTAT=statnl,
                   LIST_INST=listresu,VALE_MAX='OUI',VALE_MIN='OUI',),
                _F(NOM_CHAM='SIEF_ELNO_ELGA', GROUP_MA=('shroud',), RESULTAT=statnl,
                   LIST_INST=listresu,VALE_MAX='OUI',VALE_MIN='OUI',),
                _F(NOM_CHAM='SIEF_ELNO_ELGA', GROUP_MA=('cblcy',), RESULTAT=statnl,
                   LIST_INST=listresu,VALE_MAX='OUI',VALE_MIN='OUI',),
                ),
          );

IMPR_RESU(MODELE=model, FORMAT='MED', UNITE=80,
```

```
          RESU=_F(GROUP_MA=('shroud','cblcy'),RESULTAT=statnl,
                  NOM_CHAM=('SIEF_ELNO_ELGA',),LIST_INST=listresu,
                  ),
          );

IMPR_RESU(MODELE=model, FORMAT='MED', UNITE=81,
          RESU=_F(GROUP_MA=('mast',),RESULTAT=statnl,
                  NOM_CHAM=('DEPL','SIEF_ELNO_ELGA','SIPO_ELNO_SIEF',),
                  LIST_INST=listresu,),
          );

STANLEY()
FIN()
```

———————-

Before speaking of the Python loops, note the following line:
"IMPR_RESU(FORMAT='MED', UNITE=71, RESU=_F(MAILLAGE=mesh,),);"
In this line we save a copy as the mesh modified by 'DEFI_GROUP' or any other command that would have been before it.
If we open this mesh file in Salome we can see all the modification we have made to the mesh before building the model.
This line could also be modified to export the mesh to any other known format.

Well, back to our main purpose:
What has to be noted in this file is the use of one Python loop to create the loads , "lc[i]" of the clown cycling along the cable, in one one line within the loop we create 10 load cases!
Another loop appends all the load case in a single Python list "loadr" that will be used as an argument to "EXCIT" in "STAT_NON_LIN".
Finally we use the list "listresu" to limit the printing to the round numbered "INST".

Figure 34 is a view of the displacement at "INST" 2 and 5, in Salome.

Of course we can animate the view on the screen,so as too see the clown cycling its way along!
This is done from the tree:
Right click on "statnl__DEPL_...", any instant value, for example 2, "ScalarDefShape..." and choose "Sweep".
Or better:
Right click on "statnl__DEPL_...", and choose "Successive Animation", we may even record a .avi file, these files may become very very large with many INST!

At this stage of the course we are grown enough to find our way through without any further instruction!
Same style of animation can be made in Gmsh, that's the tiny arrows at the bottom of the window.

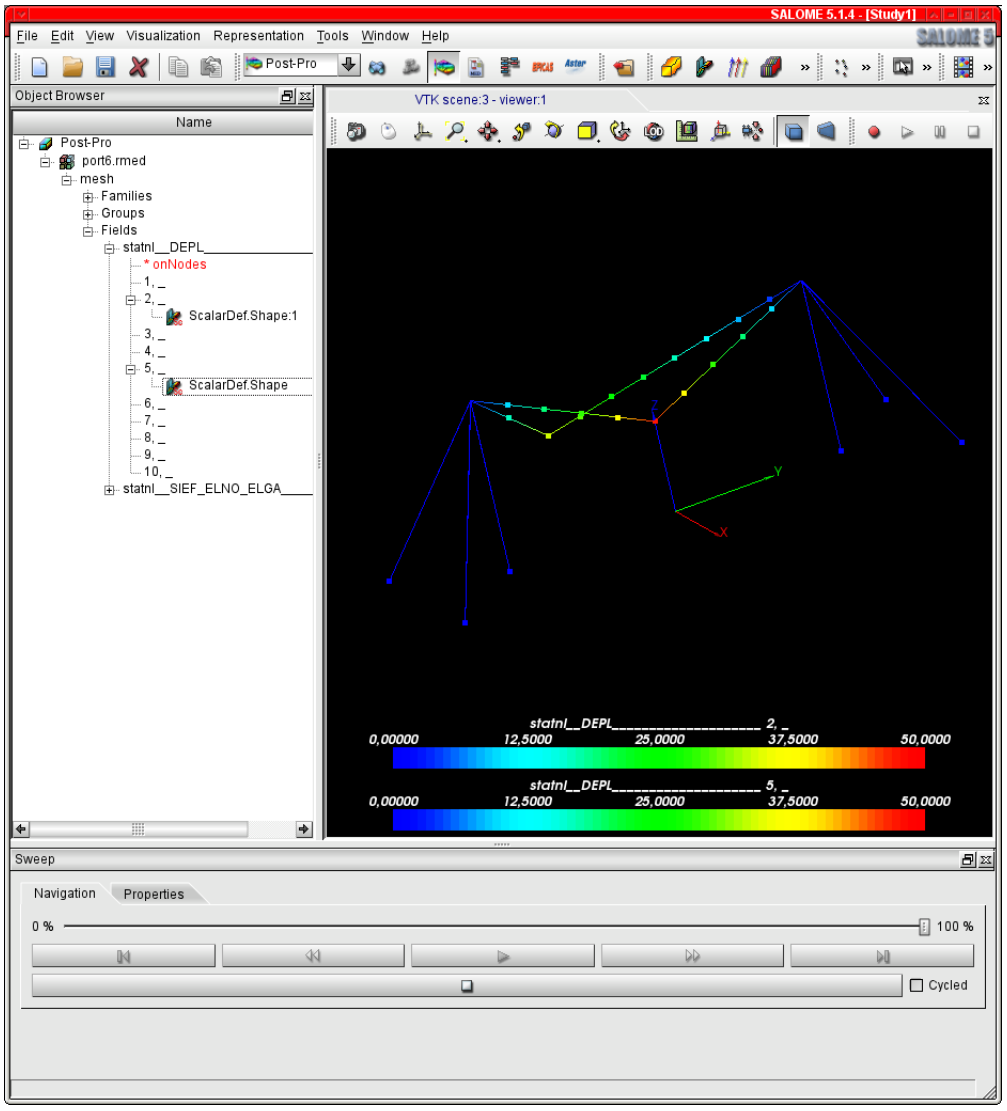


Figure 34: Deformed shape at INST 2 and 5

25 EXTRACTING SOPHISTICATED RESULTS

Code_Aster allows to extract more sophisticated results and to print them in the .resu file, or to manipulate them.

Any engineer used to work with construction codes, "Eurocode" these days in Europe, knows the requirement check for buckling which imposes to make a mixture of N and MF load to check the safety.
We will see here how to find the element, in a given group where N is maximum (or minimum) and print for this element all the sixth component of forces and moment.
This is done with some Python calls.
Here is how to modify the "port1" .comm file to do that:

First of all we have to modify the beginning of the file like this to allow Python calls:

```
DEBUT(PAR_LOT='NON',);
#import numerical method in Python
import numpy
```

Pythons calls need to know a little bit of Python, like indent rule, and to read U1.03.02.
Then in the results section we include the following:

```
#extracting some values:
#here we create a table name "extrma" containing:
#the elements where 'MFY' is maximum or minimum "OPERATION='EXTREMA'"
#at INST=5 in the group 'topbeam'
extrma=POST_RELEVE_T(ACTION=_F(INTITULE='extrma',
 GROUP_MA=('topbeam',),RESULTAT=stat,NOM_CHAM='SIEF_ELNO_ELGA',
 NOM_CMP='MFY',
 INST=5,
 OPERATION='EXTREMA',),);
#here we print it to show what it looks like
IMPR_TABLE (TABLE=extrma,)
#here we put in the variable "mfymax" the name of the element 'MAILLE',
#in the column 'MAILLE' of the table "extrma" for the line number 3, that is
#MAX_ABS
#there will be entry in the .resu file stating this selection
mfymax= extrma['MAILLE',3]
#here we print in the .resu file a line containing the value of all the components
#of 'SIEF_ELNO_ELGA' for this element
IMPR_RESU(MODELE=model,
          FORMAT='RESULTAT',
          RESU=_F(NOM_CHAM='SIEF_ELNO_ELGA', MAILLE=mfymax, RESULTAT=stat,INST=5,),
 );
```

Which produces the following printout in the .resu file:

```
#ASTER 10.02.00 CONCEPT extrma CALCULE LE 15/01/2011 A 15:57:55 DE TYPE
#TABLE_SDASTER
 INTITULE         RESU     NOM_CHAM         NUME_ORDRE   EXTREMA  MAILLE   NOEUD    CMP
 VALE
```

```
 extrma           stat     SIEF_ELNO_ELGA              4 MAX      M14      N2       MFY
6.33321E+04
 extrma           stat     SIEF_ELNO_ELGA              4 MIN      M21      N4       MFY
-3.35079E+04
 extrma           stat     SIEF_ELNO_ELGA              4 MAX_ABS  M14      N2       MFY
6.33321E+04
 extrma           stat     SIEF_ELNO_ELGA              4 MIN_ABS  M17      N18      MFY
2.71104E+03


 --------------------------------------------------------------------------------
 ASTER 10.02.00 CONCEPT stat CALCULE LE 15/01/2011 A 15:57:55 DE TYPE EVOL_ELAS

                    ENTITES TOPOLOGIQUES SELECTIONNEES
 MAILLE   : M14


 ======>

 ------>
 CHAMP PAR ELEMENT AUX NOEUDS DE NOM SYMBOLIQUE  SIEF_ELNO_ELGA
 NUMERO D'ORDRE: 4 INST:  5.00000E+00
 M14       N             VY            VZ            MT            MFY           MFZ
 N2       -9.47405E+01 -5.80118E-15 -1.87825E+02  2.05068E-12  6.33321E+04  2.11557E-12
 N16      -9.47405E+01 -5.80118E-15 -1.87825E+02  2.05068E-12  3.98540E+04  2.84072E-12
```

That is it!
Instead of the "IMRIM_RESU" we could have put the results in a table with some formatting options.

26 CHECKING BUCKLING

The following abstract can be included in a .comm file like "port1.comm" to calculate the so called "eulerian buckling" behavior.
Real understanding of what is done here needs reading U.2.08.04.
———————-

```
#the following allows to make:
#linear buckling analysis
#though not described in details yet it gives results
#it can only be used on the fly with an ASTK run
#buckling according to  U.2.08.04

resc11p1=MECA_STATIQUE(MODELE=model,CHAM_MATER=material,CARA_ELEM=elemcar,
                       EXCIT=(_F(CHARGE=ground,),
                              _F(CHARGE=selfwght,),
                              _F(CHARGE=cc,),
                              _F(CHARGE=cr,),
                             ),
                       OPTION ='SIEF_ELGA_DEPL',);

resc12p1=MECA_STATIQUE(MODELE=model,
                       CHAM_MATER=material,
                       CARA_ELEM=elemcar,
                       EXCIT=(_F(CHARGE=ground,),),
                       OPTION ='SIEF_ELGA_DEPL',);

sigc11p1 = CREA_CHAMP (TYPE_CHAM = 'ELGA_SIEF_R',
                       OPERATION = 'EXTR',
                       RESULTAT =resc11p1,
                       NOM_CHAM ='SIEF_ELGA_DEPL',
                       TYPE_MAXI = 'MINI',
                       TYPE_RESU='VALE');

regc11p1=CALC_MATR_ELEM (OPTION = 'RIGI_GEOM',
                         MODELE=model,
                         CARA_ELEM=elemcar,
                         SIEF_ELGA=sigc11p1,);

sigc12p1 = CREA_CHAMP (TYPE_CHAM = 'ELGA_SIEF_R',
                       OPERATION = 'EXTR',
                       RESULTAT =resc12p1,
                       NOM_CHAM ='SIEF_ELGA_DEPL',
                       TYPE_MAXI = 'MINI',
                       TYPE_RESU='VALE');

regc12p1=CALC_MATR_ELEM (OPTION = 'RIGI_GEOM',
                         MODELE=model,
                         CARA_ELEM=elemcar,
                         SIEF_ELGA=sigc12p1,);

remep1=CALC_MATR_ELEM (OPTION = 'RIGI_MECA',
                       MODELE=model,
                       CHAM_MATER=material,
```

```
                        CARA_ELEM=elemcar,
                        CHARGE = (ground, selfwght, cc, cr,),);

nup1=NUME_DDL(MATR_RIGI=remep1,);

ramc1p1=ASSE_MATRICE (MATR_ELEM=remep1, NUME_DDL=nup1,);

ragep1=ASSE_MATRICE (MATR_ELEM=regc11p1, NUME_DDL=nup1,);

ragc12p1=ASSE_MATRICE (MATR_ELEM=regc12p1, NUME_DDL=nup1,);

ramep1=COMB_MATR_ASSE (COMB_R=(_F(MATR_ASSE=ramc1p1, COEF_R=1.0,),
                               _F(MATR_ASSE=ragc12p1, COEF_R=1.0,),),);
#here we st the mini and maxi value of the modes with STURM
#the calculation will fail if there are no modes in this range
#it is then necessary to adjust
#sturm u45201
#les mini et maxi pour chercher les bons modes dans un premier temps avec STURM
# ensuite pour faire le calcul dans cette plage
mini=100;
maxi=1000;
IMPR_STURM (MATR_A=ramep1,
            MATR_B=ragep1,
            TYPE_RESU='MODE_FLAMB',
            CHAR_CRIT_MIN=mini,
            CHAR_CRIT_MAX=maxi,);

flamb=MODE_ITER_SIMULT(MATR_A=ramep1,
                       MATR_B=ragep1,
                       TYPE_RESU='MODE_FLAMB',
                       CALC_FREQ=_F(OPTION='BANDE',
                                    CHAR_CRIT=(mini,maxi,),
                                    DIM_SOUS_ESPACE=80,
#                                   NMAX_ITER_SOREN=80,
                                    ),);

flamb=NORM_MODE (reuse=flamb,
                 MODE=flamb,
                 NORME='TRAN',);

IMPR_RESU(
#next line not necessary if there is a previous result
#for example from a MECA_STATIQUE in the the .med file
#         MODELE=model,
          FORMAT='MED', UNITE=80,
          RESU=_F(
#same as above
#                  MAILLAGE=mesh,
                   RESULTAT=flamb,
                   NOM_CHAM='DEPL',),
         );

IMPR_RESU(MODELE=model,
```

```
          FORMAT='RESULTAT',
          RESU=_F(
                  RESULTAT=flamb,
#                 INFO_RESU='OUI',
                  TOUT_PARA='OUI',
                  FORM_TABL='OUI',
                 ),
);

_____________________-
```

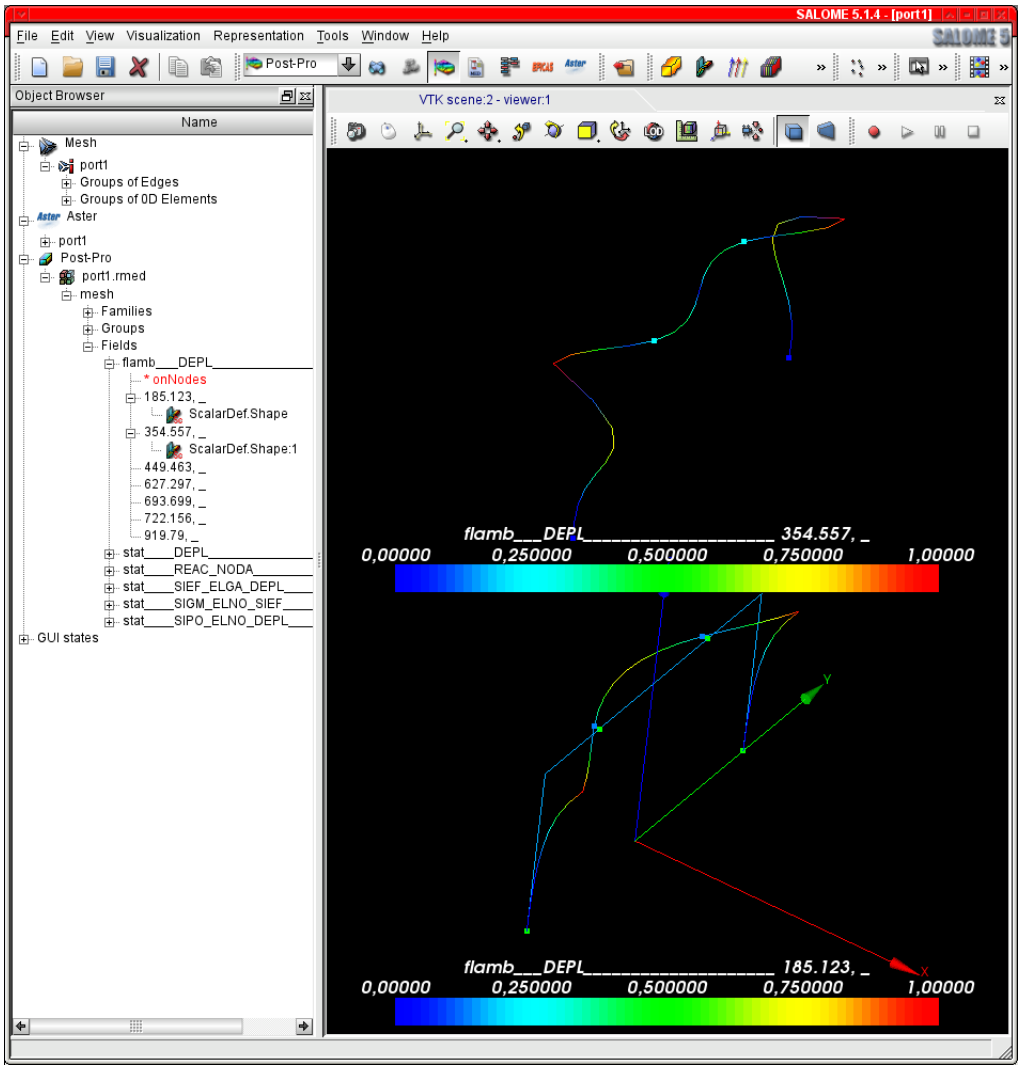


Figure 35: First two buckling modes of example port 1

This buckling analysis can be carried out for only one load case.
For the results in ASCII format the best place to look is the .mess file, after the summary of the flamb=MODE_ITER_SIMULT command we can find a table looking that:

```
-------------------------------------------------------------------------
        CALCUL MODAL:  METHODE D'ITERATION SIMULTANEE
                       METHODE DE SORENSEN

    NUMERO     CHARGE CRITIQUE     NORME D'ERREUR
         1       1.85123E+02         1.92400E-13
```

```
         2      3.54557E+02       1.93393E-14
         3      4.49463E+02       3.39080E-14
         4      6.27297E+02       1.87472E-14
         5      6.93699E+02       9.35241E-15
         6      7.22156E+02       1.28525E-14
         7      9.19790E+02       1.06227E-14
 NORME D'ERREUR MOYENNE:  0.42460E-13
```

The "CHARGE CRITIQUE" is the factor by which the load case has to be multiplied to obtain buckling with the mode that can be viewed in the graphical window.
A value less than one means an unsafe structure, from the point of view of this type of analysis.
The value can well be negative, this means than the mode is obtained with a value of the load case changed of sign.

Figure 35 shows the first two modes in Salome Post-Pro for the example "port1".

Note that for buckling analysis in the example with plates elements, 'DKT' must be replaced by 'COQUE_3D' not to raise an error[8].

27 VIEWING MODE SHAPES

The following abstract can be included in a .comm file to calculate the so called the mode shapes .

```
#analyse modale
MACRO_MATR_ASSE(MODELE=model,
                CHAM_MATER=material,
                CARA_ELEM=elemcar,
                CHARGE=ground,
                NUME_DDL=CO('NUMEDDL'),
                MATR_ASSE=(_F(MATRICE=CO('RIGIDITE'),
                              OPTION='RIGI_MECA',),
                           _F(MATRICE=CO('MASSE'),
                              OPTION='MASS_MECA',),),);

modes=MODE_ITER_SIMULT(MATR_A=RIGIDITE,
                       MATR_B=MASSE,
                       CALC_FREQ=_F(
#we may need altering the next lines to get values in a useful range
                                    OPTION='PLUS_PETITE',
                                    NMAX_FREQ=10,
                                    ),
                        );

IMPR_RESU(MODELE=model,
```

[8] Which is not as simple as that, as the mesh element have to transformed from TRIA6 to TRIA7 for the plate elements but must be kept as SEG2 for the line elements, that is a bit of programing on the .geo file!

```
          FORMAT='MED',
          UNITE=80,
          RESU=_F(
#next line not necessary if there is a previous result
#for example from a MECA_STATIQUE in the the .med file
#                 MAILLAGE=mesh,
                  RESULTAT=modes,
                  NOM_CHAM='DEPL',),);

IMPR_RESU(MODELE=model,
          FORMAT='RESULTAT',
          RESU=_F(
                  RESULTAT=modes,
#this prints all the info relative to modal analysis
                  TOUT_CHAM='NON',
#                 TOUT_PARA='OUI',
                  FORM_TABL='OUI',
                  ),
          );
```

One remark, for the modal analysis the only 'CHARGE' allowed is fixed DOF, the analysis is performed with the mass as assigned with 'RHO' for the materials and the masses values for 'M_T_D_N', all the forces have to be converted to masses if we want them to be taken into account.
But we all know that!

For the numerical results in ASCII format the best place too look is the .mess file, after the summary of the modes=MODE_ITER_SIMULT command we can find a table looking that:

```
-------------------------------------------------------------------------
       CALCUL MODAL:  METHODE D'ITERATION SIMULTANEE
                      METHODE DE SORENSEN

    NUMERO    FREQUENCE (HZ)    NORME D'ERREUR
         1      3.86886E+00       3.28342E-09
         2      1.12121E+01       2.32170E-09
         3      1.64993E+01       2.25551E-10
         4      3.73360E+01       3.97953E-09
         5      4.62410E+01       2.25538E-09
         6      8.17233E+01       3.04475E-09
         7      1.06556E+02       3.98827E-09
         8      1.07386E+02       3.74780E-10
         9      1.34691E+02       6.96490E-09
        10      1.56687E+02       2.65903E-09
 NORME D'ERREUR MOYENNE:  0.29097E-08


-------------------------------------------------------------------------
```

Figure 36 shows the first mode on Gmsh output, with a scale of 100, for the example "port3".
While figure 37 shows the two firsts modes on Salome output, with the same scale.

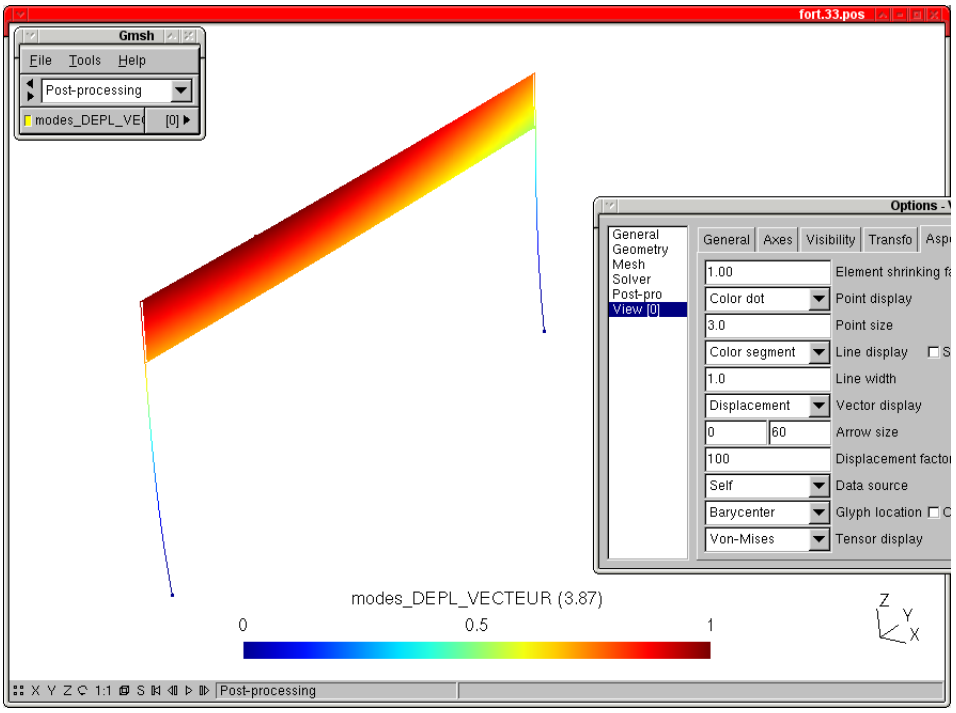


Figure 36: First mode of example port 3

Printing any results in the .resu file is quite useless for this type of calculation.
The results can be viewed and animated in Salome or Gmsh as we explained in the non linear calculation section.

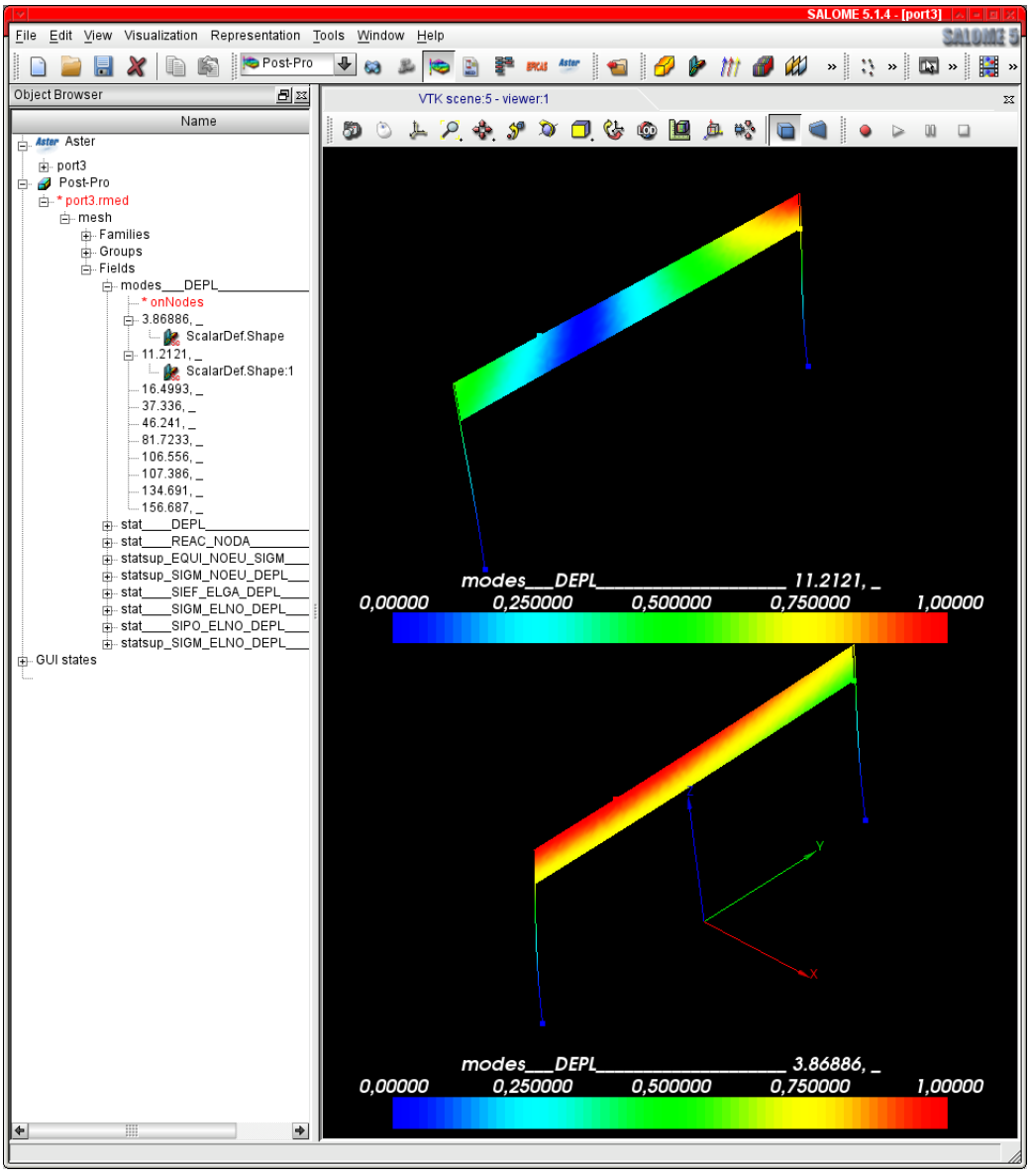


Figure 37: Firsts two modes of example port 3

28 PLAYING WITH Gmsh and Code_Aster VERSIONS

Concerning *Code_Aster*:.
As stated earlier the examples have been tested with *Code_Aster* 10.2.
They should work on version 10.1 and 9.4 with the keyword 'PESANTEUR' altered,.
Concerning version 10.3 there seem to be a serious mishap for printing results in beams when 'AFFE_VARC' has been used (at the time of this writing).
For Gmsh the code is pretty stable for years, and though not tried extensively, i think the examples should be OK with any 2.** versions.

29 IMPORTING EXPORTING IN Gmsh, TIPS

In the previous examples we have created a geometry within Gmsh, meshed it and exported it as *.med file in MED format.
The pull down lists in Gmsh show us that there are many other opportunities.
We can export mesh in:

-.msh format, that is Gmsh native ASCII format
-.inp Abacus Mesh
-.bdf or .dat Nastran mesh

and some others, of course it cannot export more than the entities it knows, that is: node position, element connectivity, and groups.

We can import mesh in:

-.msh format, that is Gmsh native ASCII format
-.bdf or .dat Nastran mesh

and some others, once more it cannot import more than the entities it knows, that is: node position, element connectivity, and groups.

For post processing we can import .med file, alas not completely.

We can import geometry:

-.geo format, that is Gmsh native ASCII format
-.brep, .igs .stp

For these three file type we have to "Save As" *.geo file before making any modification otherwise Gmsh will crash.
.
Finally "Save As" > "Gmsh Unrolled Geometry *.geo" will save a file with Points, Lines, Surfaces, Physicals, ordered, but all the scripting is killed, this may be useful at the end of a very large problem as loading is then much faster, however we advise to keep a copy of the unrolled (scripted file) at hand.

30 CORRECTING INSTALLATION MISHAP

The following is a summary of how I solved these issues with quite a bit of trial and error, there may be other ways, or my advices may not work.
Once we get working with *Code_Aster* graphical tools like STANLEY, the tool may not launch at all with nasty messages in the *.mess file.
Here is the usual workaround:
In the ASTK window, menu Configuration > Preferences > Network:
the "Client machine name" field should be the machine name, which can be obtained by typing the command "hostname" in a terminal,

the "Domain name" can be left empty
"Forced DISPLAY variable" can be set to ":0.0",
and button "rsh" and "rcp" pushed on.
And STANLEY should then launch itself on request.

Some other issues may need to do that:
In the ASTK window, menu Configuration > Servers..., the "Server name" maybe changed from "localhost" to the machine name as above.

If nothing happens when we push "TRACER" in STANLEY, despite a green light, lets try the following:
In the STANLEY window, menu Parametres > Editer, in the pull down list right off "Mode" choose "Gmsh/Xmgrace", push "OK" it should then work in this mode.
We may also choose "Salome" mode, if Stanley is ran from a Salome-Meca study.
To make it run from stand alone *Code_Aster* I suppose we need a stand alone version of Salome and a bit of tinkering around with "Port de Salome", though I have never used STANLEY this way.
.
When the setup is right the "interactive follow-up" box in ASTK may be checked and when pushing "Run" a terminal window will open telling us all what's going, in fact it is almost a copy of what will be written in the .mess file, quite useful in case of longish problems, we know where we are.
.
Some of these advices are worth only for a single machine set up, he same machine acting as client and calculation server.

With most recent versions of STANLEY the label of the check box in between "TRACER" and "CALCULER" may not appear, we should read "Sur déformée", plot on the deformed shape, checking this box does what it claims!

31 GETTING THE TOYS

To get Gmsh go to that link: http://geuz.org/gmsh/.
Download the "Current stables release", that is a .tgz archive.
Unpack it somewhere, I do that in /opt as root.
The executable is then /opt/gmsh*/bin/gmsh.
Run it any way, command line, launching script or window manager menu entry.
There is no need to compile anything from source for an every day use, I have never done it!
On the same link under "Documentation" the "Reference manual" is also very useful.

To get Salome-Meca go to that link: http://www.code-aster.org,
Look for the download area, once founded download the archive that suits.
For the installation *strictly* follow the instructions given on the page.
As for Gmsh I make the installation in /opt.

For a stand alone version of *Code_Aster* go to that link: http://www.code-aster.org,
Look for the download area, once it is found download the archive that suits.
For the installation *strictly* follow the instructions given on the page.
Again I make the installation in /opt.
This installation is somewhat long, about 30 minutes, as it is a true compilation, not installation of binaries.

For a stand alone version of SALOME (without the link to *Code_Aster* in it) go to that link: http://www.salome-platform.org/,
Look for the download area, once founded, download the archive that suits.

I found it is better to download the "Universal binaries", the installation is fast and the other binaries are very dependant on distribution and may no work.
I have never compiled from source.
For the installation *strictly* follow the instructions given on the page.
Again I make the installation in /opt.

For *Code_Aster*, Salome-Meca and Salome I strongly advise *not* to run them for a trial as root after the first install, this will set some ownership to "root" for some configuration files, after that it is a troublesome task to put things the right way up!

There is a very easy way to try out all these programs, it comes under the shape of a live CD containing all of them, plus much more, this is called CAELinux..
It can be downloaded from: http://caelinux.com
I have started this way!
The site contains also a Wiki, a Forum and many useful information.

32 WORKED EXAMPLES

The archive contains sub directories named after the problem names, "port*".
Each directory contains:
The data:
- -a .geo file,
- -a .med file,
- -a .comm file,
- -an .astk file.

As results:
- -a .mess file,
- -a .resu file,

most graphical results file have been dropped out to keep the archive as small as possible.