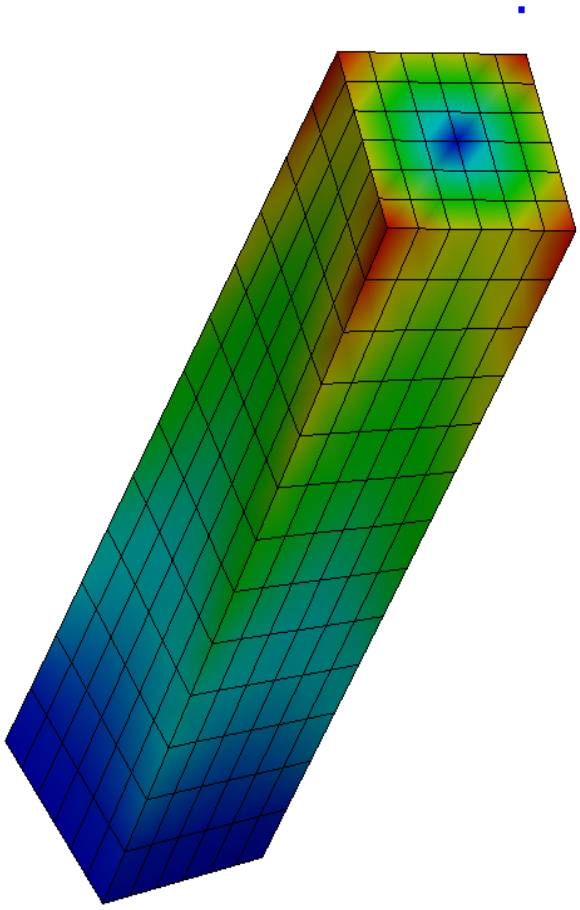
Applying torque
with
Code Aster®

Using LIASON_SOLIDE to simulate a rigid link between nodes

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



1 Table of Contents

1 Introduction and theory

This tutorial will be divided into two major parts: An 'artistic' part where we focus on the use of parameters, set-up of the command file and preparation of the mesh. The realism of the result is not as important as it is to demonstrate application of torque with Code Aster®. The second part will focus on validating the results obtained with an analytical approach.

2 Applying Torque in Code Aster®

Code Aster® does not (yet anyway) have a straight forward way of applying torque to a structure. Only nodes have the option of having moment (**FORCE_NODALE-> MX,MY** and **MZ**) applied to them. What this means is, that you, in order to apply torque to an object, have to connect a single node to the object and and thereby transfer the moment into torque.

There are several ways of achieving this, but I will only describe one of the ways here. In terms of what happens inside Code Aster®, this method places a single node adjacent to the surface you want to apply torque to, creates a rigid link between the single node and the surface and finally applies the moment and thereby torque.

1. Surface you want to apply torque to
2. Create adjacent single node
3. Code Aster® creates rigid links between the node and the surface
4. Apply moment to single node

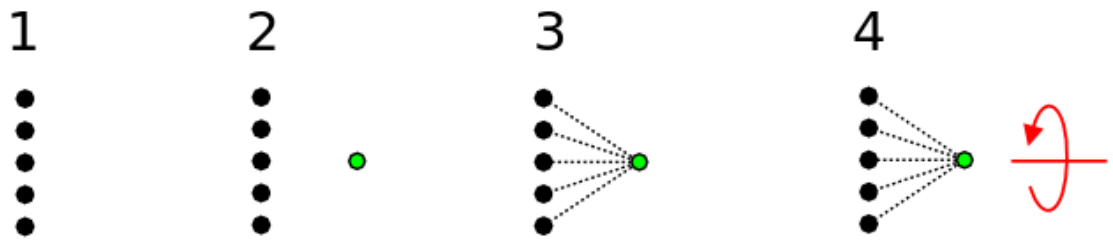


Figure 2.1: Virtual rigid linkage

Note: If applying force (**FORCE_NODALE** – **FX**, **FY** and **FZ**) to the single node, the force is distributed equally to each node, thus in this case each node would receive 1/5 of the applied force.

3 Creating the groups in Salomé®

(You should be familiar with creating a geometry and meshing it already)

Mesh your object and create a single node center and normal to the surface you want to apply torque to. In this case it makes sense to create it with an offset normal to the surface so it's easier to select the single node. Assign a 'node group' to the single node

The surface you want to apply torque to, must be assigned to a 'node group' and not a 'surface group' (important). Include the single node you just created (important).

Finally, assign a surface group (or which ever you prefer) to the surface that will prevent the object from moving. Here the bottom of the object. See figure 3.1

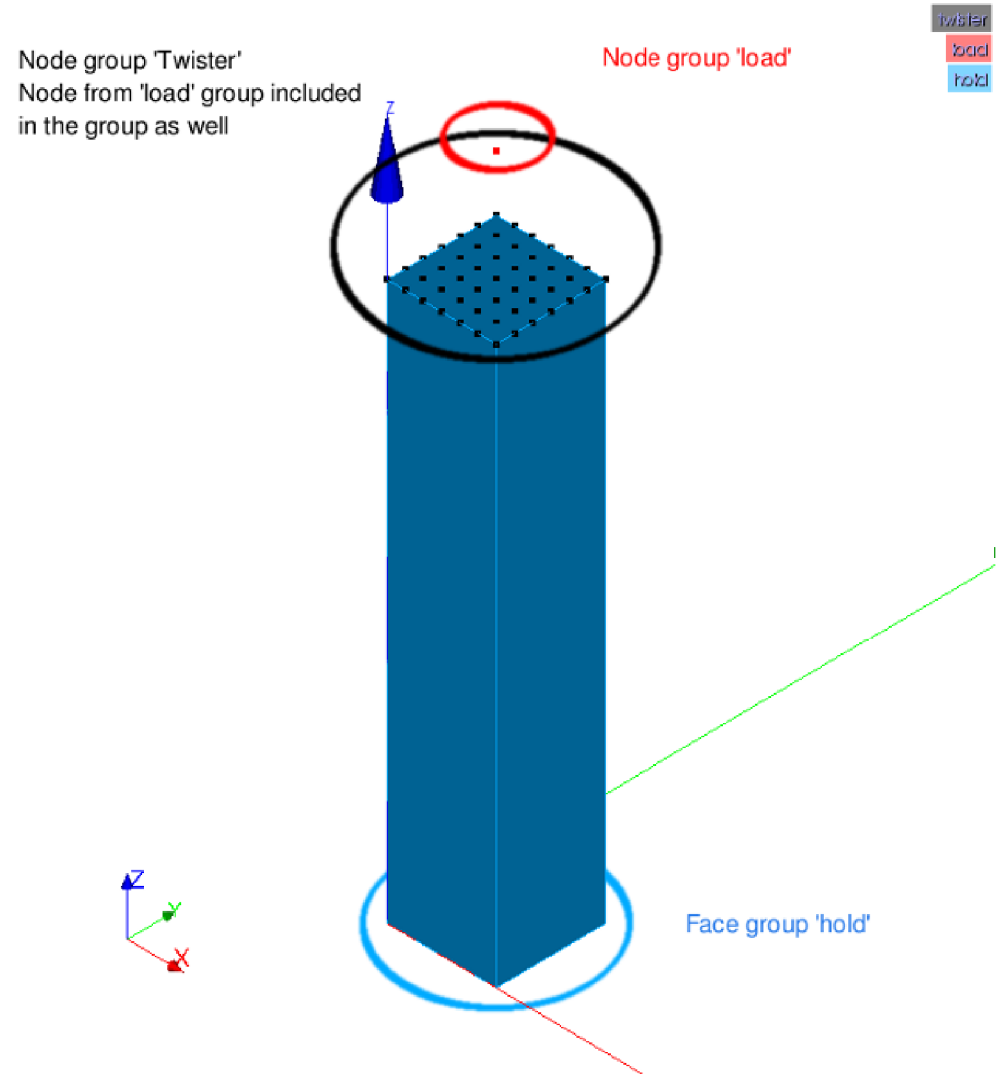


Figure 3.1: Mesh groups

4 Parameters

Using parameters in **Code Aster®** is as useful as in any situation; it lets you change many variables at once and can let you control an entire calculation by changing only one parameter.

In this case one parameter is used to do just that: **T1**

- **T1=10**
- **T2=(2*T1)**
- **T3=(3*T1)**
- **T4=(4*T1)**
- **NBT=T1** (or any of the Tx)
- **T_END**= Any of the Tx

By changing the value of **T1** you change the number of steps (resolution) in the calculation and the length of the calculation.

By using a value of 10 for **T1**, 10 steps is calculated for each interval, and with 4 intervals 40 calculations is done. Using **T1=100** increases the resolution and length by an order of 10.

The parameter **NBT** determines the number of steps for each interval so with **NBT=T1** the number of steps becomes 10 (see above)

The parameter **T_END** determines for how many steps the entire calculation should run for. In other words; it determines how many intervals are calculated, e.g. **T_END=T1** means 10 steps are calculated and outputted (one interval) and with **T_END=T4**, 40 steps are calculated and outputted (four intervals).

I've tried to visualize the influence of the parameters in this diagram (Figure 4.1)

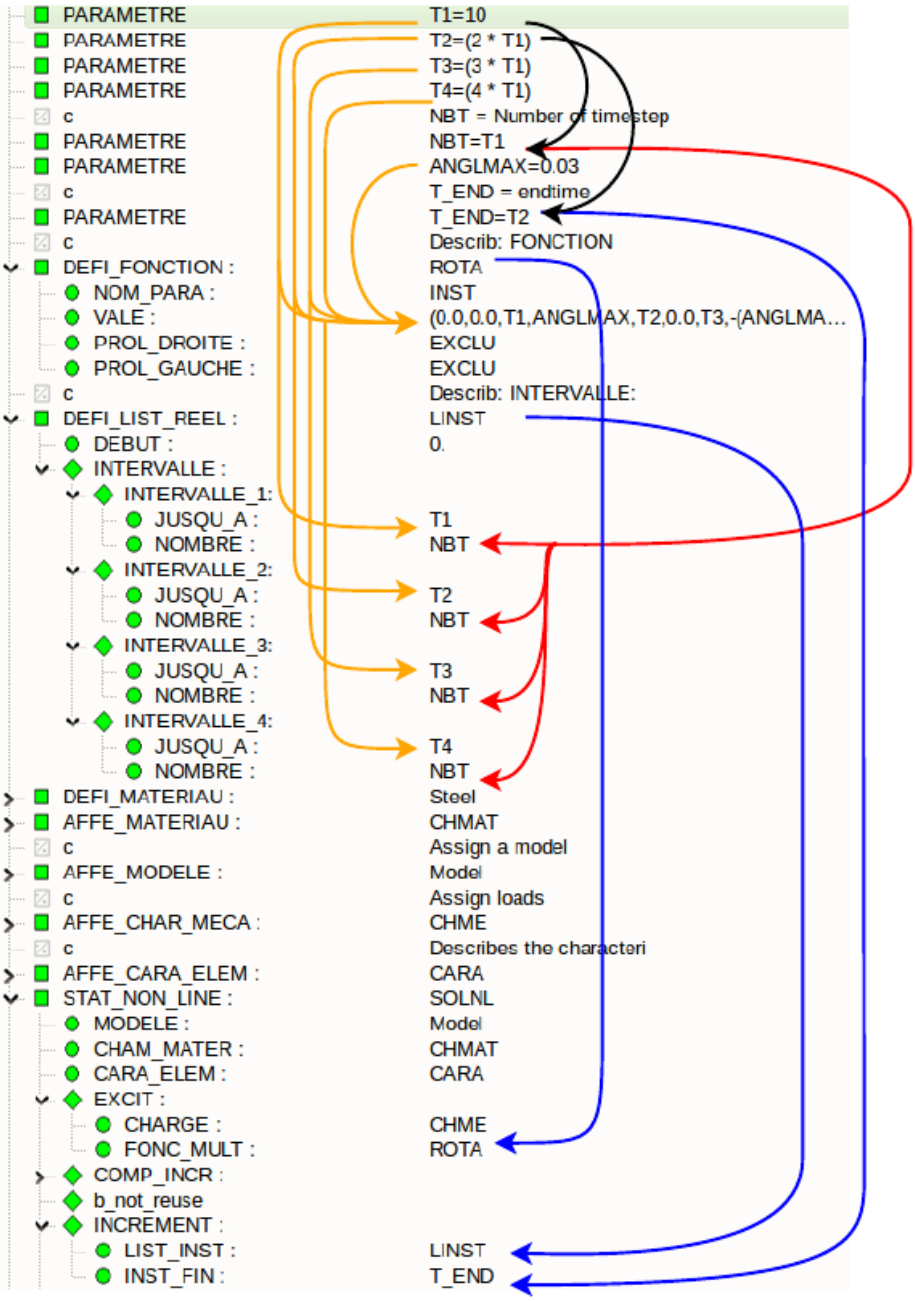


Figure 4.1: Influence of parameters

The load multiplier is also connected to **T1** and thus follows the number of steps you increase or decrease (but not the magnitude of the load). The parameter **ANGLMAX** is used to set the imposed magnitude of the load (angle of twist) in the object.

In the following diagram (figure 4.2), the function for the load and the intervals of the calculation is visualized.

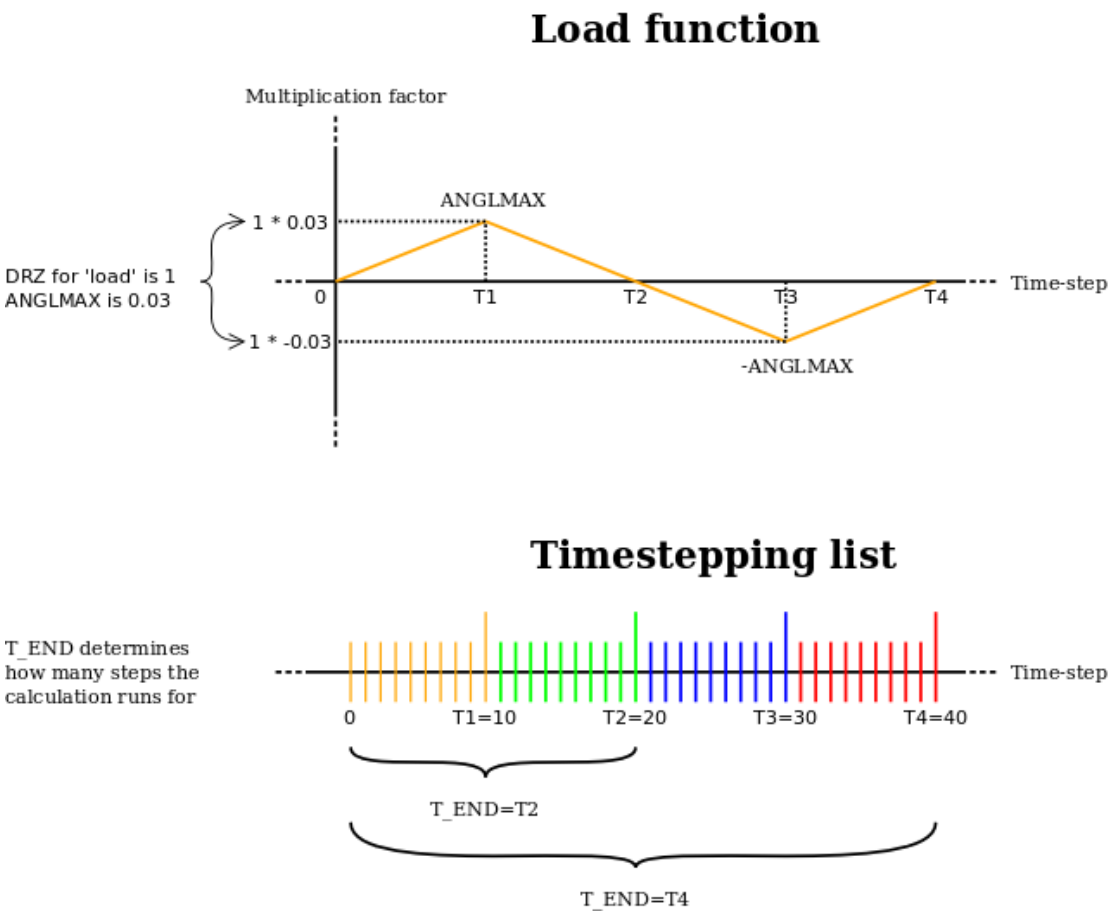


Figure 4.2: Load function and time-stepping list

5 Applying the theory to Code Aster® – the command file

This section will go through the command file describing each relevant step.

5.1 'Creating' the single node inside Code Aster®

In order to apply any kind of load or displacement to anything in **Code Aster®**, a model and material must be assigned to it. This also applies to the single node we're going to use to impose load or displacement in this case. We create a discrete element on the node.

Definition

- **LIRE_MAILLAGE**: Read the original mesh file and assign it the name 'mesh1'
- **CREA_MAILLAGE**: Create a new mesh from the original mesh
  - **CREA_POI1**: Create a mesh containing a single node from the node group 'load' - This is necessary in order to assign a model and material to the node later on.
  - **NOM_GROUP_MA**: Name of the new mesh group 'load' - will now have the same name as the node group 'load' we created in **Salomé**

```
DEBUT();
mesh1=LIRE_MAILLAGE(FORMAT='MED',);
mesh=CREA_MAILLAGE(MAILLAGE=mesh1,
                   CREA_POI1=_F(NOM_GROUP_MA='load',
                                GROUP_NO='load',),);
```

(Document **[U4.23.02]** details the use of **CREA_MAILLAGE**)

5.2 Defining the parameters

```
T1 = 10;

T2 = (2 * T1);

T3 = (3 * T1);

T4 = (4 * T1);

#NBT = Number of time steps
NBT = T1;

ANGLMAX = 0.03;

#T_END = end time
T_END = T4;
```

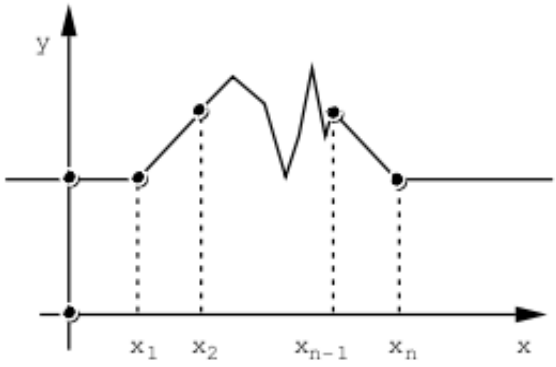
5.3 Defining the load function

Definition

- **DEFI_FONCTION**: Define a function
  - **NOM_PARA=INST**: Define a function based on points and dependent on time (See the image above on the load function)
  - **VALE**: Values (points) in a Cartesian coordinate system - x1,y1 , x2,y2 , x3,y3 to form a domain.
  - **PROL_DROITE** and **PROL_GAUCHE** = extend_right and extend_left. Defines the type of extension to the right (or left) the domain of the variable:
    - **'CONSTANT'** For an extension with the last (or first) value of the function,
    - **'LINEAR'** For an extension along the first segment defined **(PROL_GAUCHE)** or the last segment defined **(PROL_DROITE)**

'EXCLUDED' The domain starts and ends with the values given. In the case a calculation requires a value of the function outside the domain of definition, the code stops with fatal error. See figure 5.1

- PROL_DROITE = 'CONSTANT' , PROL_GAUCHE = 'CONSTANT'



- PROL_DROITE = 'LINEAIRE', PROL_GAUCHE = 'EXCLU'

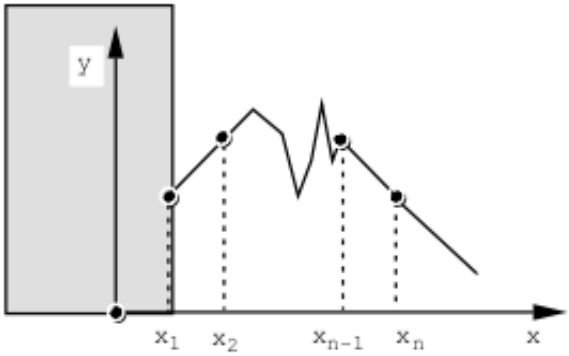


Figure 5.1:

Doc **[U4.31.02]** - **3.8 Operands PROL_DROITE** and **PROL_GAUCHE**

```
ROTA=DEFI_FONCTION(NOM_PARA='INST',VALE=(0.0,0.0,
                           T1,ANGLMAX,
                           T2,0.0,
                           T3,-(ANGLMAX),
                           T4,0.0,
                           ),PROL_DROITE='EXCLU',PROL_GAUCHE='EXCLU',
);
```

5.4 Defining the list of time steps

Definition

4 intervals each containing 10 time steps

- **LINST=DEFI_LIST_REEL:** Create a list called **LINST** consisting of real numbers. Start the list at 0 **(DEBUT=0)**
- **INTERVALLE:** Interval ranging from 0 to 10 (**T1=10**), number of steps in this interval: **NOMBRE=NBT** which is 10. Next interval continues from 10 to **T2=20** and this interval is also divided into 10 steps. And so on.

A single statement with an interval of **JUSQU_A=40** with **NOMBRE=40** would be just the same, but without the option of increasing (or decreasing) the number of time steps within each of the 4 intervals.

```
LINST=DEFI_LIST_REEL(DEBUT=0.,
                     INTERVALLE=(_F(JUSQU_A=T1,
                                    NOMBRE=NBT,),
                                 _F(JUSQU_A=T2,
                                    NOMBRE=NBT,),
                                 _F(JUSQU_A=T3,
                                    NOMBRE=NBT,),
                                 _F(JUSQU_A=T4,
                                    NOMBRE=NBT,),),);
```

5.5 Assigning materials

As said before, everything must have a material assigned to it, even the single node used for applying load/displacement:

Definition

- **DEFI_MATERIAU:** Define material, assign the name 'Steel' to it.
  - **ELAS:** We only deal with a regular elastic material here, with an elasticity module (Young's module) of 210 GPA and a Poisson's ratio of 0.3
- **AFFE_MATERIAU:** Assign the material 'Steel' to everything **(TOUT=OUI, ALL=YES)**

```
Steel=DEFI_MATERIAU(ELAS=_F(E=2.1E5,
                            NU=0.3,),);

CHMAT=AFFE_MATERIAU(MAILLAGE=mesh,
                    AFFE=_F(TOUT='OUI',
                            MATER=Steel,),);
```

5.6 Assigning models

Here we assign models to the components of the object:

Definition

- **AFFE_MODELE:** Assign model, conveniently called 'Model' here
  - **MAILLAGE:** Assign to the mesh called 'mesh'
  - **AFFE_1:** Assign a 3D mechanical model to everything
  - **AFFE_2**: Assign a discrete model to the single node 'load'
    - **MODELISATION=DIS_TR:** A **discrete** model with the capabilities of **T**ranslation and **R**otation is assigned.

```
Model=AFFE_MODELE(MAILLAGE=mesh,
                  AFFE=(_F(TOUT='OUI',
                           PHENOMENE='MECANIQUE',
                           MODELISATION='3D',),
                        _F(GROUP_MA='load',
                           PHENOMENE='MECANIQUE',
                           MODELISATION='DIS_TR',),),);
```

5.7 Assigning loads

Definition

- **AFFE_CHAR_MECA:** Assign a mechanical load. Use the previously assigned model **(MODELE=Model)**
    - **DDL_IMPO:** Impose a boundary condition.
        - **GROUP_MA='hold'**
            - **LIAISON=ENCASTRE:** The group 'hold' is assigned **'ENCASTRE',** which is just an easy way to block movements in all directions instead of assigning **DX=0, DY=0, DZ=0** to the group.
        - **GROUP_MA=load**
            - **DRZ=1.0:** The single node 'load' is assigned a rotation around the Z axis: **DRZ=1**
    - **LIAISON_SOLIDE:** This is where the magic happens
        - **GROUP_NO='twister':** The node group 'twister' is now told to have a completely rigid connection **(LIAISON_SOLIDE)** - if the node 'load' moves, so does the rest of the nodes in the group.

```
CHME=AFFE_CHAR_MECA(MODELE=Model,
                    DDL_IMPO=(_F(GROUP_MA='hold',
                                 LIAISON='ENCASTRE',),
                              _F(GROUP_MA='load',
                                 DRZ=1.0,),),
                    LIAISON_SOLIDE=_F(GROUP_NO='twister',),);
```

5.8 Defining the characteristics of the single node

Again, the single node must have some characteristics in order to work within the calculation, and since it's 0D and not 3D we must describe it. It doesn't make as much sense here, as it does when you describe the characteristics of a 1D beam (POUTRE), but it has to be done.

Definition

- **AFFE_CARA_ELEM:** Assign the characteristics of the element using the **MODELE=Model**
    - **DISCRET=discrete**
        - **CARA=K_TR_D_N:**
            - **K**: Stands for the type of the discrete element: K (stiffness), M (mass) or A (damping)
            - **TR**: It can translate and rotate
            - **D**: It's a Diagonal matrix - the other setting is to omit this letter (you then have to input the whole matrix)
            - **N**: It's a node (**L** for seg2 element)

To make sure the node 'load' is only used to induce loads in the structure and not influencing the transfer of the load, everything is assigned a value of 0.1 (**DX, DY, DZ, DRX, DRY, DRZ**)

See **[U4.42.01]** section **13.3.3** for further information on using **AFFE_CARA_ELEM** and what the values mean

```
CARA=AFFE_CARA_ELEM(MODELE=Model,
                    DISCRET=_F(CARA='K_TR_D_N',
                               GROUP_MA='load',
                               VALE=(0.1,0.1,0.1,0.1,0.1,0.1,),),);
```

5.9 Defining the calculation

Definition: Using the list *LINST*, calculate a static solution for each step of the intervals defined.

- **MECA_STATIQUE:** Create a linear calculation called 'Solution', use the **MODELE='Model'.**
  - **CHAM_MATER=CHMAT:** Use the material *CHMAT* to calculate the material 'field' **(CHAMP=field)**
  - **CARA_ELEM=CARA:** Include the characteristics *CARA* we previously defined for the single node 'load'
  - **EXCIT=** Excitation of the calculation (in lack of a better word)
    - **CHARGE=CHME:** use the loads assigned in *CHME*
    - **FONC_MULT:** Function multiplier; This uses the function '*ROTA*' we defined previously (see image in 'Introduction'). At time step 0 the loads are multiplied by 0 and at time step **T1=10** the loads are multiplied by 1 (**DRZ=1** for the single node and **ANGLMAX=0.03**)
  - **LIST_INST='LINST'** defines progression of the calculation and ending when it reaches **INST_FIN=T_END**

```
Solution=MECA_STATIQUE(MODELE=Model,
                       CHAM_MATER=CHMAT,
                       CARA_ELEM=CARA,
                       EXCIT=_F(CHARGE=CHME,
                                FONC_MULT=ROTA,),
                       LIST_INST=LINST,
                       INST_FIN=T_END,);
```

5.10 Calculating and writing the results

Nothing new here; calculate the elements and nodes, write the displacement and equivalent nodal stress to a .med file.

```
Solution=CALC_ELEM(reuse =Solution,
                   RESULTAT=Solution,
                   OPTION=('EQUI_ELGA_SIGM','EQUI_ELNO_SIGM',),);

Solution=CALC_NO(reuse =Solution,
                 RESULTAT=Solution,
                 OPTION=('FORC_NODA','REAC_NODA','EQUI_NOEU_SIGM',
),);

IMPR_RESU(FORMAT='MED',
          RESU=_F(RESULTAT=Solution,
                  NOM_CHAM=('DEPL','EQUI_NOEU_SIGM','EQUI_ELGA_SIG
M',),),);
```

5.11 Extracting relevant values in text form

Print the extracted values to the *.resu* file. **FORMAT='RESULTAT'** denotes that the values should be written as text in 'Aster' format. Omitting the keyword **UNITE**, defaults the value to '8', which is the default for the *.resu* file. All values extracted at **INST=10**

The **IMPR_RESU** keyword is divided into four parts:

1. Extraction of displacement of the 'load' node from the *DEPL* field – to verify the angle of twist of the node.
2. Extraction of the forces on the 'load' node from the *FORC_NODA* field – to verify that the node isn't influencing the simulation (forces should approximate 0).
3. Extraction of maximum (**VALE_MAX='OUI'**) values from the *SIEF_ELGA_DEPL*, to verify normal stress and shearing stress.
4. Extraction of maximum values from the *EQUI_ELGA_SIGM* field, to verify equivalent stress.

```
IMPR_RESU(MODELE=Model,
          FORMAT='RESULTAT',
          RESU=(_F(RESULTAT=Solution,
                   NOM_CHAM='DEPL',
                   INST=10,
                   GROUP_NO='load',),
                _F(RESULTAT=Solution,
                   NOM_CHAM='FORC_NODA',
                   INST=10,
                   GROUP_NO='load',),
                _F(RESULTAT=Solution,
                   NOM_CHAM='SIEF_ELGA_DEPL',
                   INST=10,
                   NOM_CMP=('SIXX','SIYY','SIZZ','SIXY','SIXZ','SI
YZ',),
                   VALE_MAX='OUI',),
                _F(RESULTAT=Solution,
                   NOM_CHAM='EQUI_ELGA_SIGM',
                   INST=10,
                   VALE_MAX='OUI',),),);
```

5.12 ASTK set-up

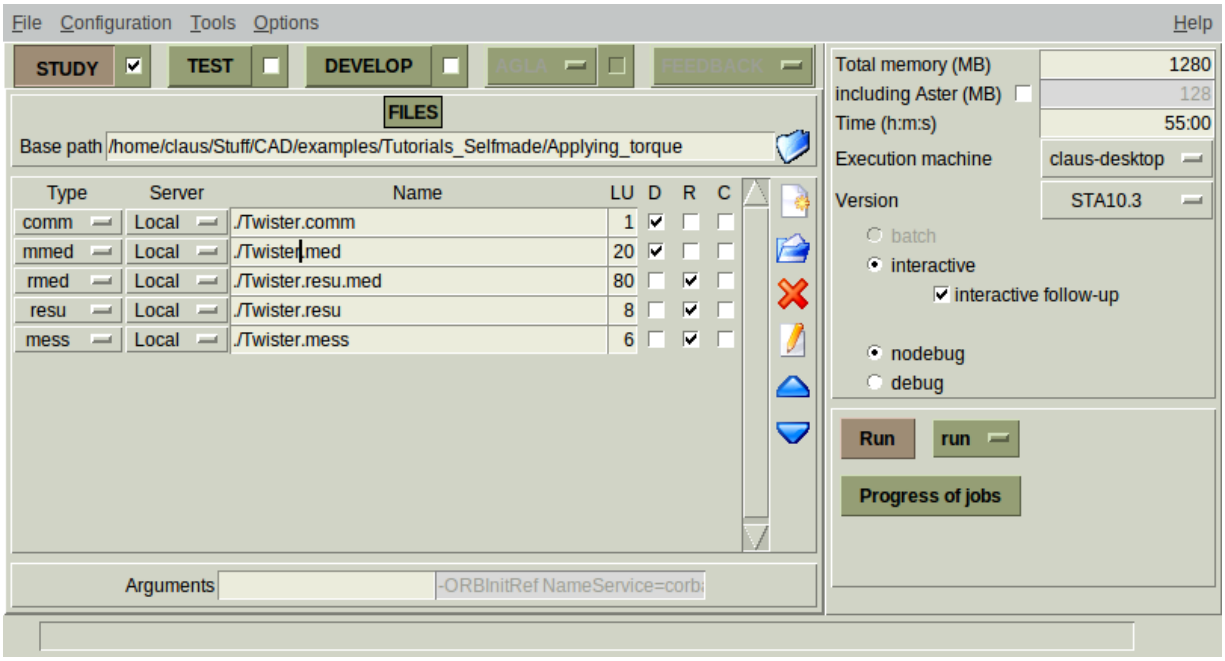


Figure 5.2: ASTK set-up

6 Post-processing

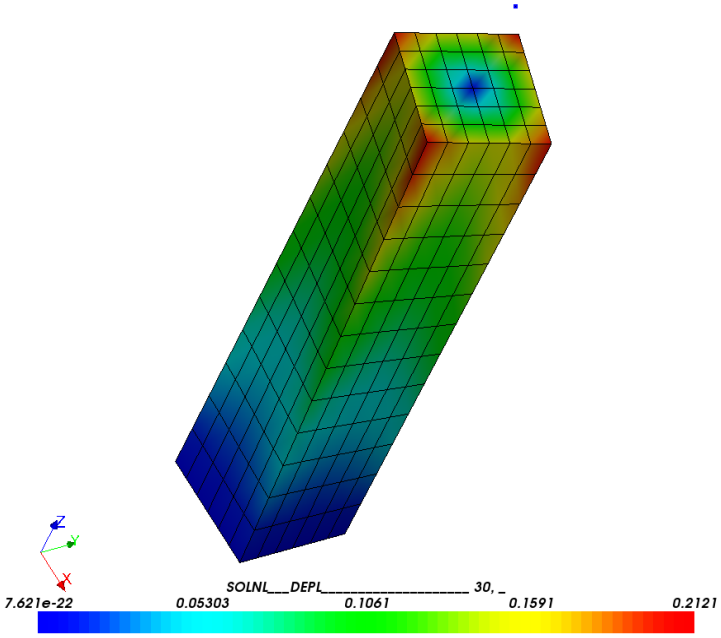


Figure 6.1: Deformation

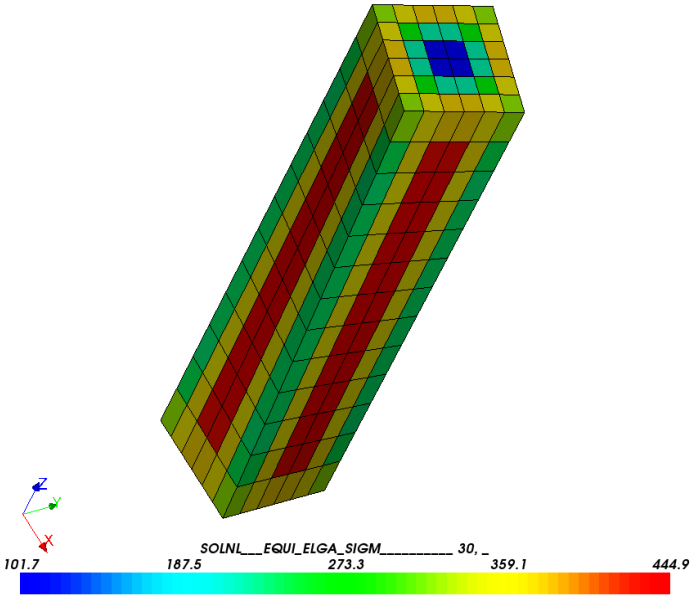


Figure 6.2: Von Mises stress

7 Generating an animated GIF file from the results

7.1 Outputting the images from Salomé®

Use **Salomé's** post-processing modules to open the resulting .med file, right-click the *Solution* field and select 'Successive Animation' (Figure 7.1):

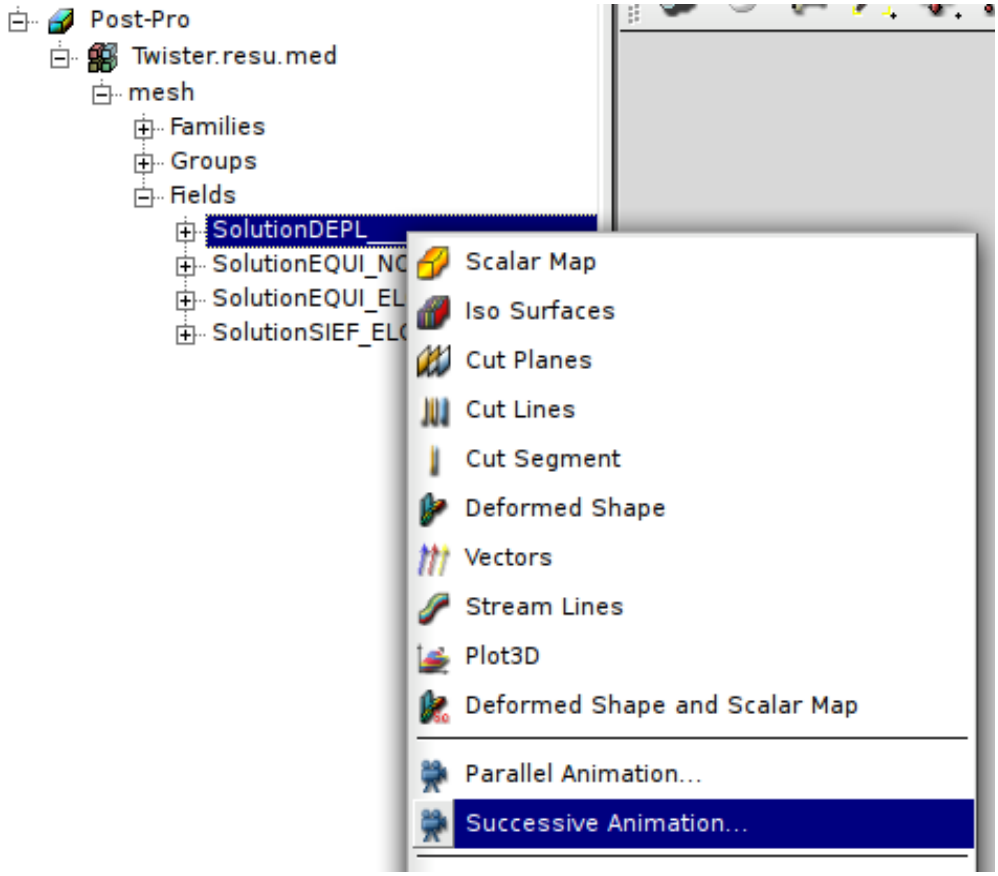


Figure 7.1:

Click 'Setup Animation'.

In the lower right corner select 'Deformed shape and Scalar map'. Click 'Properties' and set scaling to 10 and the field to *SolutionEQUI_ELGA_SIGM*. Under 'Scalar Bar' select **VMIS** from the drop-down menu, click OK, click OK again to return to the animation tab. Figure 7.2

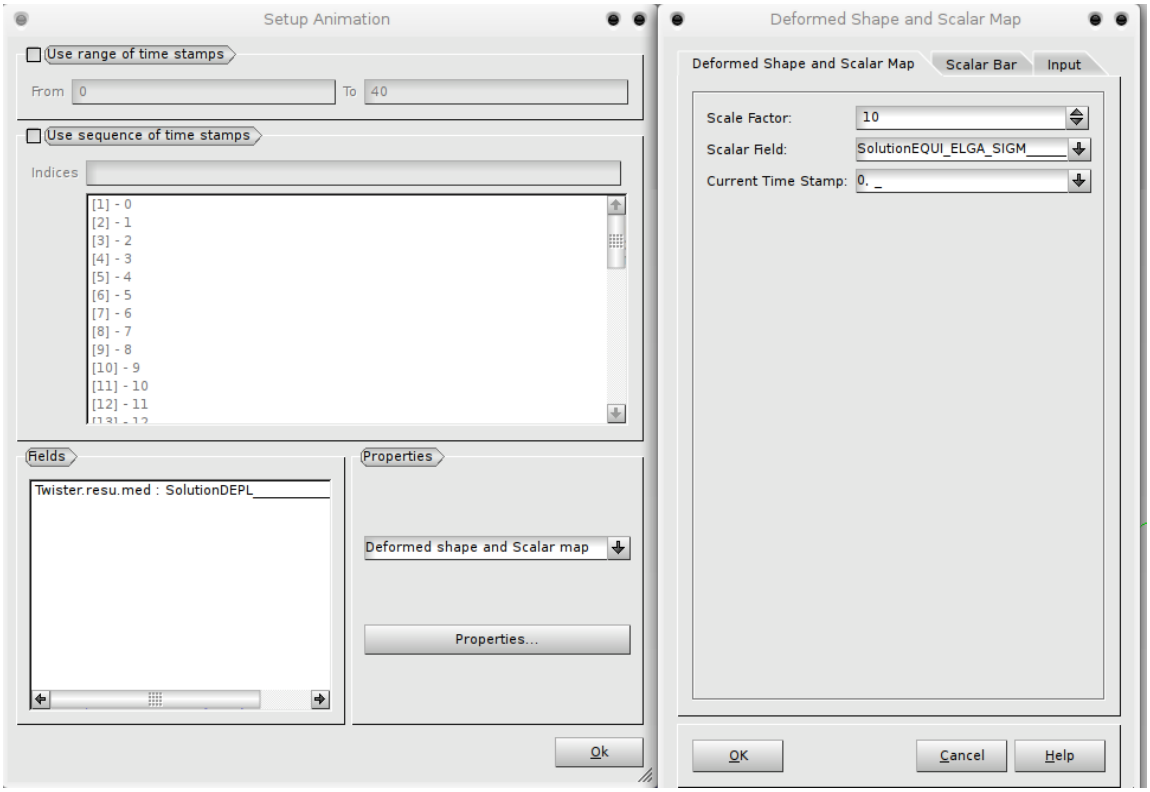


Figure 7.2:

Press 'Generate frames' and tick 'Save pictures to directory'. Select the format of the pictures to be outputted (I prefer PNG) and select which directory the files should be saved to. Now press the Play button and the let animation run once. Figure 7.3

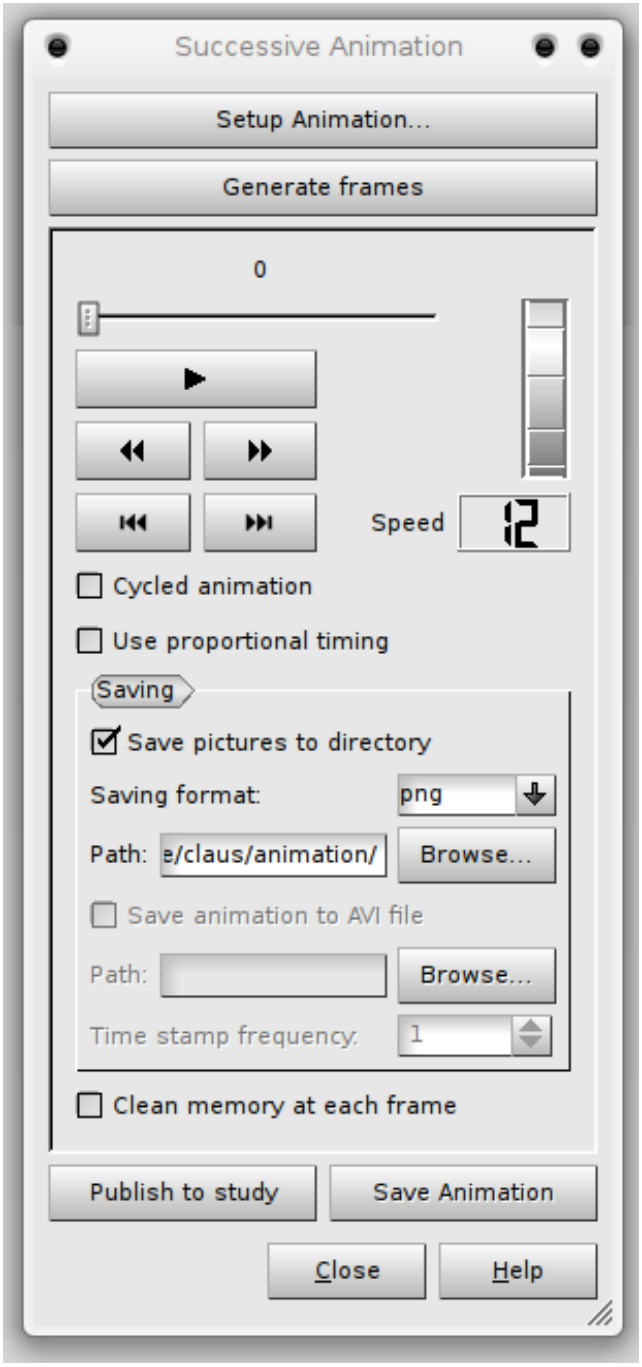

Successive Animation
Setup Animation...
Generate frames
0
Speed
12
Cycled animation
Use proportional timing
Saving
Save pictures to directory
Saving format:
png
Path:
e/claus/animation/
Browse...
Save animation to AVI file
Path:
Browse...
Time stamp frequency:
1
Clean memory at each frame
Publish to study
Save Animation
Close
Help

Figure 7.3:

7.2 Generating the gif from the output

(You must have Imagemagick® installed on your system to follow this step)

Open a terminal and navigate to the directory where you saved the series of images.

Launch the following command:

```
convert -loop 0 -quality 100 -resize %50 -antialias -bordercolor
white -border 5 -blur 0.5x0.5 *.png giffer.gif
```

Definition

'convert' is a program that is a part of Imagemagick and can be used to do a myriad of things to a batch of files. Here we tell it to let the resulting animation loop indefinitely (-loop 0) keep the compression quality at 100, resize the images to 50%, antialias the image to get rid of some noise, give it a white border with the width of 5 pixels and blur the image slightly to remove even more noise. Finally all the PNG files are processed (*.png) and the resulting animated GIF file is called 'giffer.gif'

Change any of the values to suit your need, this is just an example I used to create the animation in the beginning of the page.

8 Tutorial part two

8.1 The mesh

For the verification of the **LIAISON_SOLIDE** rigid link, a circular shaft is build the same way the previous mesh was built (see figure 8.1)

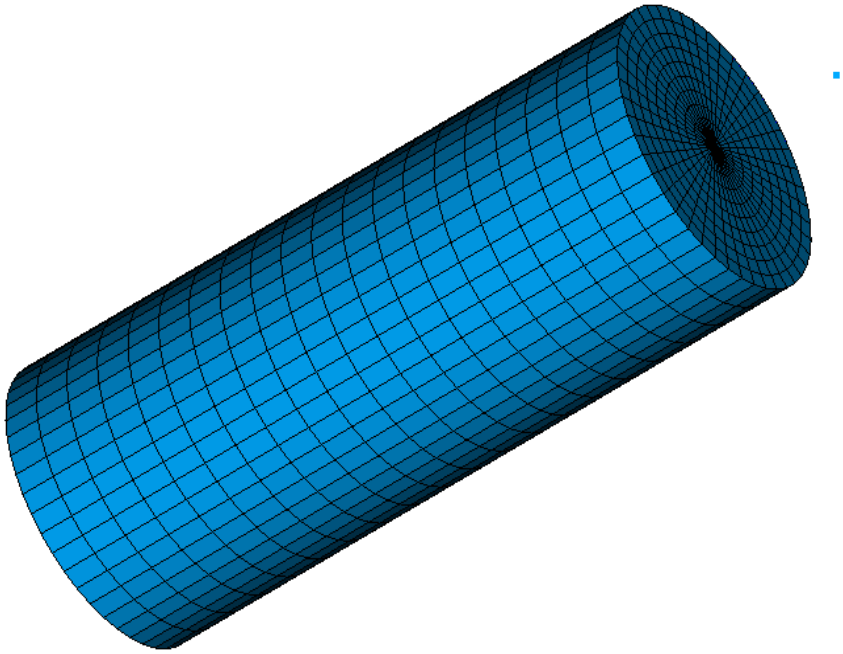


Figure 8.1: Shaft composed mainly of hexahedrons

Groups are created in the same manner, with a load node linked to the end surface of the shaft.

8.2 Analytical approach

To verify our (simple) simulation, we can do some quick analytical calculations and compare the results obtained from Code Aster®.

Contrary to the previous case where an imposed angle of twist was used, here we apply torque to the 'load' node - **MZ=50e6**

```
moment=AFFE_CHAR_MECA(MODELE=Model,
                      DDL_IMPO=_F(GROUP_MA='hold',
                                  LIAISON='ENCASTRE',),
                      LIAISON_SOLIDE=_F(GROUP_NO='twister',),
                      FORCE_NODALE=_F(GROUP_NO='load',
                                      MZ=50e6,),);
```

For the applied torque on the shaft, we'll calculate the theoretical angle of twist, maximum strain, maximum shear stress and then we'll verify our assumption that we're working with structural steel with a shear module of 80GPa.

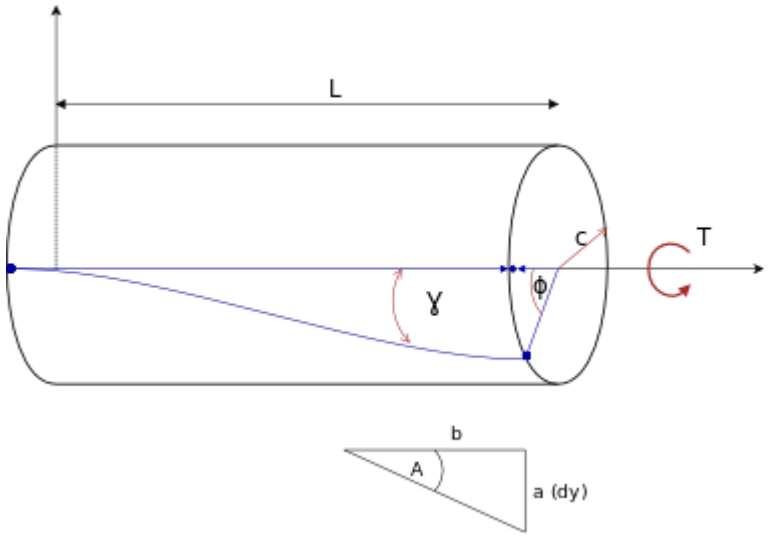


Figure 8.2: Symbols used

8.3 Material properties and parameters

Diameter	D:=200mm
Radius	c:=D/2
Length	L:=500mm
Shear module of structural steel	G:=80·10³ MPa
Polar moment of a circular cross section	J:=1/2 · π · c⁴
Applied torque	T:=50·10⁶ N·mm
For Code Aster®	****
Elasticity module	210·10³ MPa
Poisson's ratio	0.3

8.4 Theoretical values

Max. shearing stress is expressed as:

$$\tau_{max} = \frac{T \cdot c}{J} \rightarrow \frac{50 \cdot 10^6 \, Nmm \cdot 100\text{mm}}{1.57 \cdot 10^8 \, mm^4} \simeq 31.8\text{MPa} \quad (1)$$

Angle of twist is expressed as:

$$\phi = \frac{\tau_{max} \cdot L}{G \cdot c} \rightarrow \frac{31.8 \cdot 500}{80 \cdot 10^3 \cdot 100} \simeq 2 \cdot 10^{-3} \, rad \quad (2)$$

Max. shearing strain is expressed as:

$$\gamma = \frac{c \cdot \phi}{L} \rightarrow \frac{100 \cdot 2 \cdot 10^{-3}}{500} = 4 \cdot 10^{-4} \, rad \quad (3)$$

8.5 Values from Code Aster®

As noted in the command file, we can determine the maximum shearing stress just by looking in the *.resu* file under "*champ par element aux noeuds de nom symbolique SIEF_ELGA_DEPL"*, components *SIXZ* and *SIYZ*.

The values approx. $\tau_{max.Aster} \simeq 32\text{MPa}$

Likewise, in the *.resu* file under "*champ aux noeuds de nom symbolique DEPL",* we can find a value for the rotation around the Z-axis, *DRZ*, for the node in the 'load' group.

The angle of twist is thus $\phi_{Aster} \simeq 2 \cdot 10^{-3} \, rad$

To find the maximum shearing strain from this value, we can plug it into the equation for maximum shearing strain (3), or we can find the strain from the deformation.

A way to do this, is to imagine a right angle triangle on the surface of the shaft, and let the length 'L' and the displacement of a node on the rim in the X- or Y-axis, be the two catheti (see figure 8.2). This way, we can determine the angle 'A' from this equation:

$$\tan(A) = \frac{a}{b} \rightarrow \tan\left(\frac{0.198\text{mm}}{500\text{mm}}\right) \simeq 4 \cdot 10^{-4} \, rad$$

Just to check if our value for shear module is OK, G is expressed as: $G = \frac{\tau_{max}}{\gamma}$, so using the values from Code Aster® we have $G = \frac{\tau_{max.Aster}}{\gamma_{Aster}} \rightarrow \frac{32\text{MPa}}{4 \cdot 10^{-4}} = 80\text{GPa}$

8.6 Comparison

Here is a comparison of the results from Smath® and Code Aster®, reasonably rounded off:

	Analytical	Code Aster®	Pct. Difference
Maximum shearing stress	31.83 MPa	32 MPa	-0.53%
Angle of twist	1.99e-3 rad	1.98e-3 rad	0.50%
Maximum shearing strain	3.98e-4	3.96e-4	0.50%
Shear module	80	80.7	-0.88%

.

5 Apr 2011 01:41:37 - Torsion.sm

$D = 200\,mm$ $L = 500\,mm$

Shear modulus - assumming it is structural steel

$G = 80\,GPa$

Torque

$c = \frac{D}{2}$ $T = 50 \cdot 10^{3}\,Nm$

Polar moment

$J = \frac{1}{2} \cdot \pi \cdot c^{4}$

Angle of twist from Aster

$\Phi_A = 1.9833 \cdot 10^{-3}\,rad$

Max shear

$\tau_{max} = \frac{T \cdot c}{J}$

$\tau_{max} = 31.83\,MPa$

Max Strain based on angle of twist from Aster

$\gamma_{A1} = \frac{c \cdot \Phi_A}{L}$

$\gamma_{A1} = 3.97 \cdot 10^{-4}\,rad$

Angle of twist

$\Phi = \frac{\tau_{max} \cdot L}{G \cdot c}$

$\Phi = 1.99 \cdot 10^{-3}\,rad$

Max strain based on deformation from Aster

$\gamma_{A2} = \tan\left(\frac{0.1982}{500}\right)$

$\gamma_{A2} = 3.96 \cdot 10^{-4}\,rad$

Max strain

$\gamma = \frac{c \cdot \Phi}{L}$

$\gamma = 3.98 \cdot 10^{-4}\,rad$

Max shear based on angle of twist from Aster

$G \cdot \gamma_{A2} = 31.71\,MPa$

Max shear from Aster

$\tau_{maxA} = 31.9898\,MPa$

Shear module based on values from Aster

$G_A = \frac{\tau_{maxA}}{\gamma_{A2}}$

$G_A = 80.7\,GPa$

Figure 8.3: Calculations done in Smath®

8.7 Post-processing

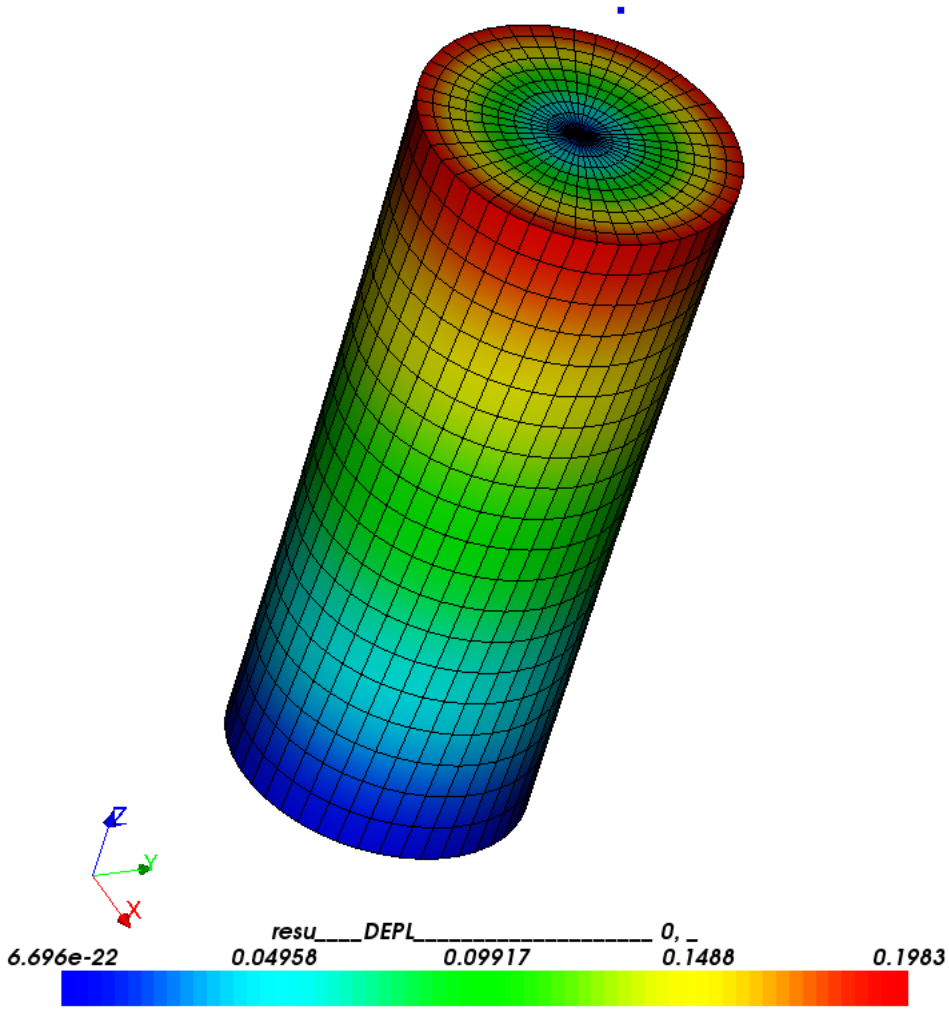


Figure 8.4: Deformation

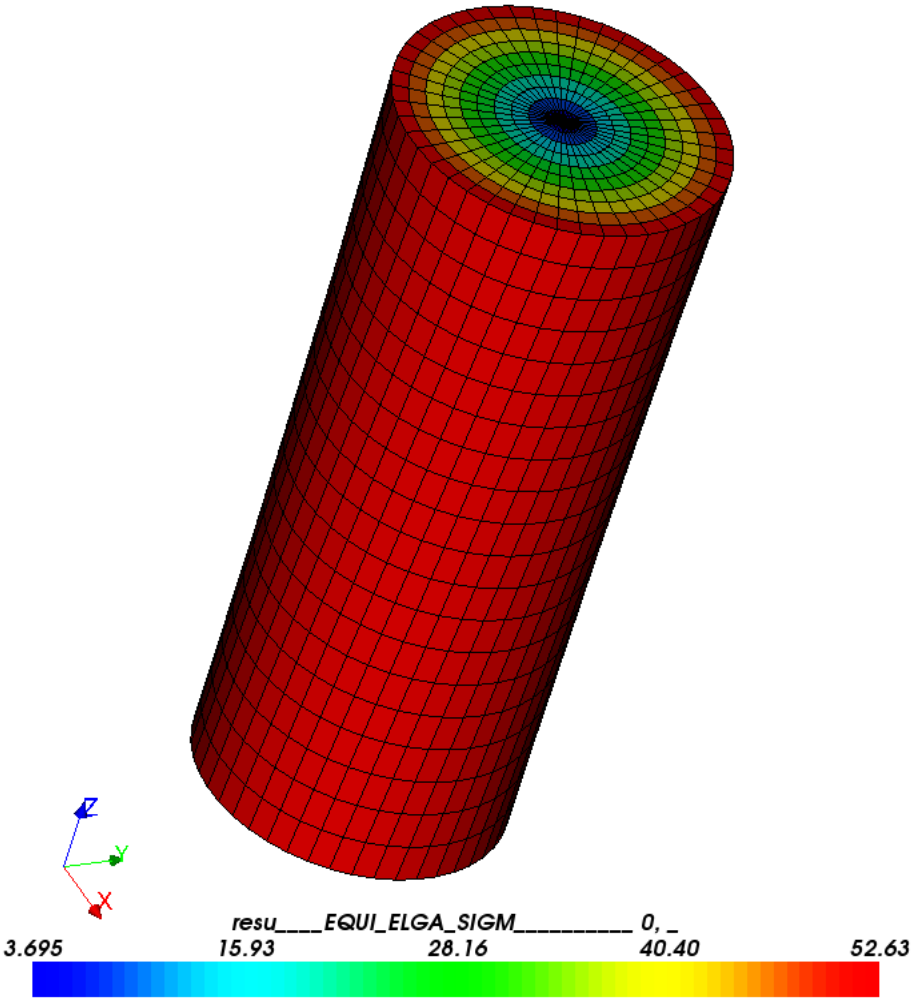


Figure 8.5: Von Mises stress

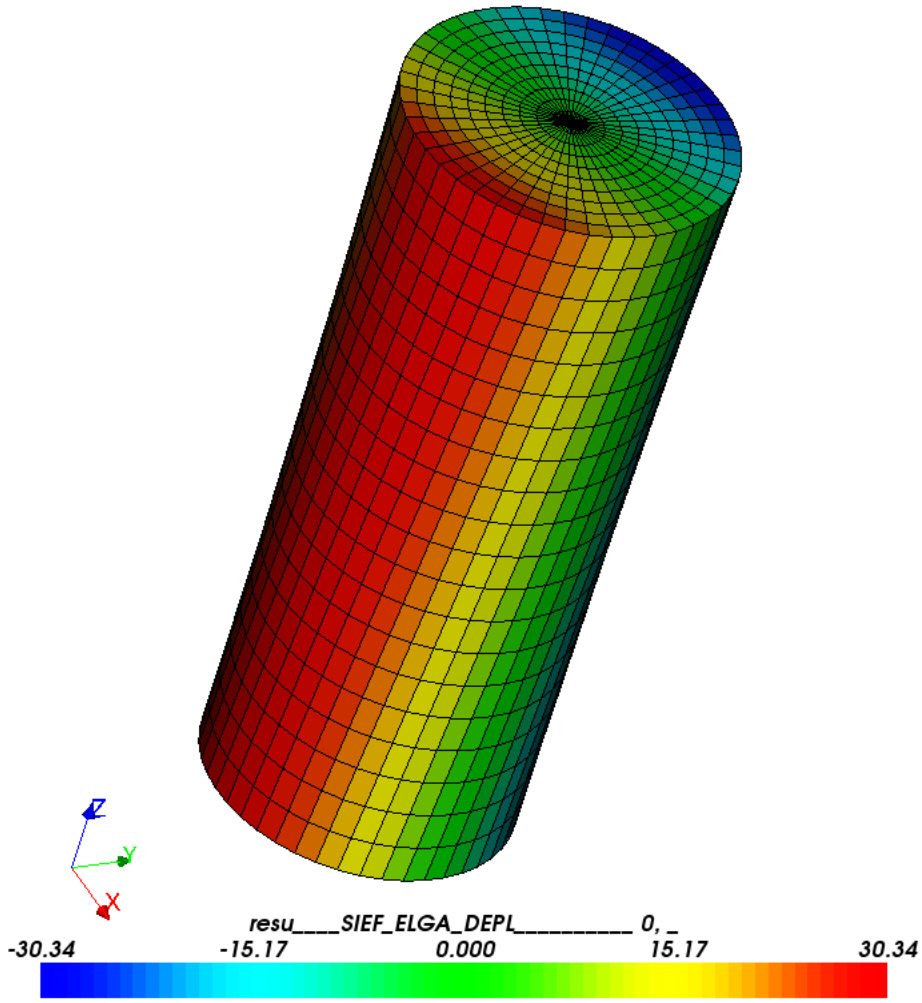


Figure 8.6: Shearing stress SIXZ

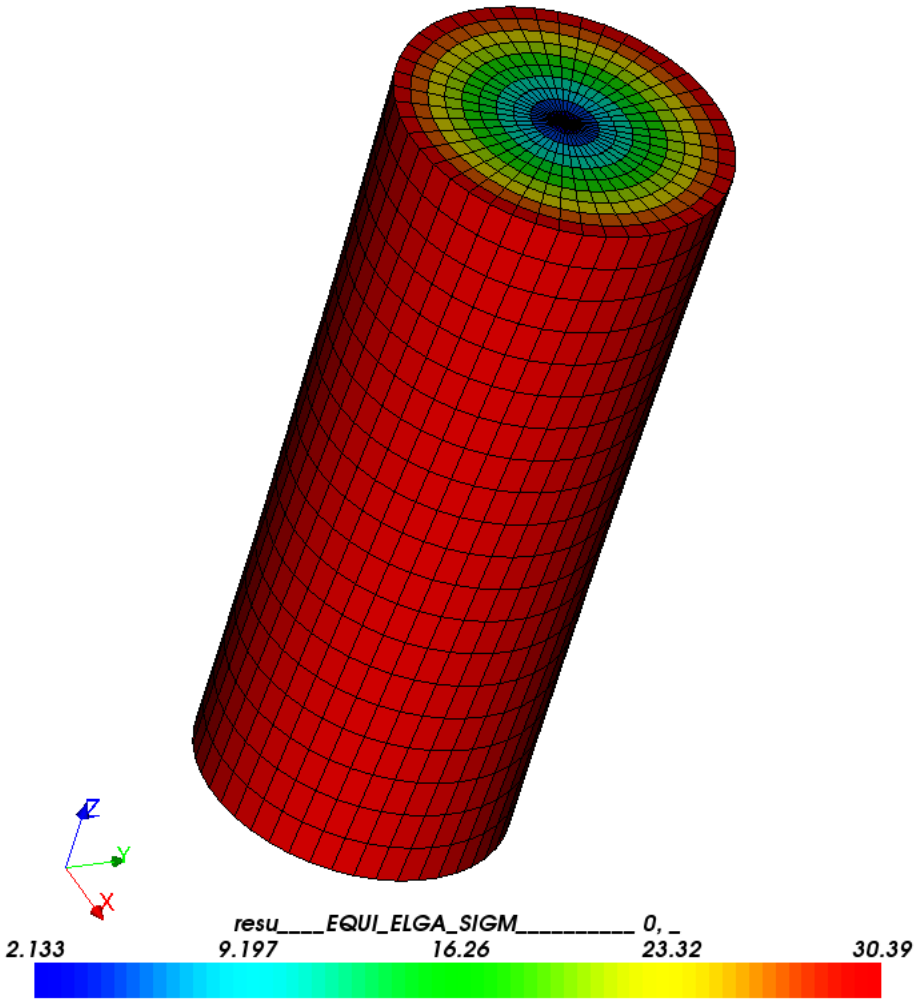


Figure 8.7: Principal stress 3

9 Conclusion, remarks and author(s)

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

Remark:

Any and all information and content in this document is published under the GPL license and can as such be used or reproduced in any way. The author(s) only ask for acknowledgment in such an event.

Acknowledgment goes out to EDF for releasing Code Aster® as free software and to all those who help out by answering questions in the forum and writing documentation / tutorials.

Contributions and/or corrections to this tutorial are always welcomed.

Author(s):

Claus Andersen – ClausAndersen81_[at]_gmail.com

ENDED OK

10 Links

[CAELinux Website] www.caelinux.com

[Code Aster® Website] www.code-aster.org

[GMSH® Website] www.geuz.org/GMSH

[XMGrace® Website] http://plasma-gate.weizmann.ac.il/Grace/