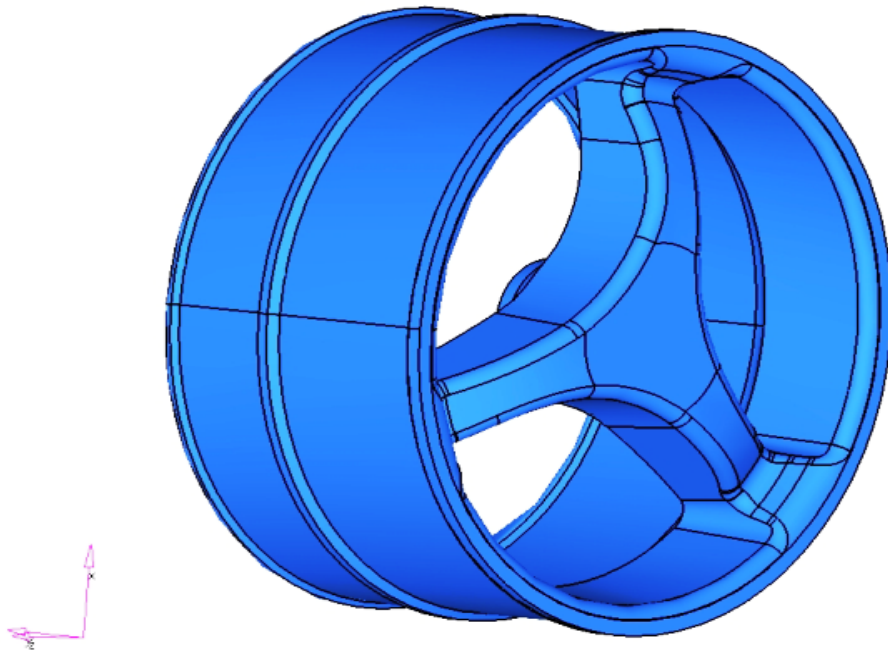
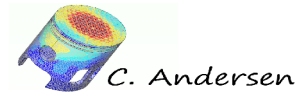


# Plasticity tutorial – rev.1.1

## Code Aster® STA10.3

Plasticity tutorial for CAELinux.com by Paul Carrico  
Translated and reworked by Claus Andersen  
5. April 2011



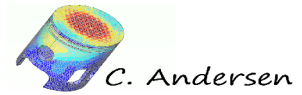


## • Table of Contents

1 Introduction part one.....	4
2 Introduction part two.....	4
3 Preparation of the model.....	5
4 The mesh.....	6
5 Objective of modelization.....	8
6 Procedure of the simulation.....	8
6.1 Stage 1: Mechanical calculation.....	8
6.2 Stage 2: Post-processing of data – MED format.....	8
7 Chronological list of the stages.....	9
8 Note on units in Code Aster.....	9
9 Stage 1 - The command file.....	9
9.1 Loading the mesh and defining the finite element model.....	9
9.2 Definition of the material.....	10
9.3 Converting element groups to node groups.....	11
9.4 Definition of the boundary conditions.....	12
9.5 Load.....	12
9.5.1 Group diagram.....	14
9.5.2 LIAISON_OBLIQUE diagram.....	17
9.6 Definition of lists.....	17
9.7 Nonlinear calculation.....	18
9.8 COMP_INCR.....	18
9.9 DEFORMATION.....	18
9.10 INCREMENT.....	19
9.11 NEWTON.....	19
9.12 CONVERGENCE.....	19
9.13 ARCHIVAGE.....	19
10 Stage 2 – Post processing.....	21
10.1 The command file.....	21
10.1.1 Calculating elements and nodes.....	21



10.1.2 Writing the results to MED.....	22
10.1.3 Writing the results to POS.....	23
10.2 Reviewing the results.....	24
10.2.1 Deformation.....	24
10.2.2 Equivalent von Mises stress.....	26
10.2.3 Equivalent strain.....	28
10.2.4 Cumulative plastic strain.....	30
10.2.5 Indication of plasticity.....	31
10.3 Force vs. displacement on the rim hub.....	32
10.3.1 Extracting nodal forces.....	33
10.3.2 Recover a function for Uy and Fy.....	33
10.3.3 Write / print the curve with XMGrace®.....	34
11 Getting accurate results with Code Aster®.....	36
11.1 Integration points (Gauss points).....	36
11.2 Linear or quadratic elements?.....	36
12 Conclusion, remarks and author(s).....	37
13 System of units.....	38



## 1 Introduction part one

In the interest of helping new users learn about Code Aster® and improve my own knowledge, I've decided to translate and update the now aging plasticity tutorial originally written by Paul Carrico in 2007. He released the document as GPL, and as a derivative this updated version is also released as GPL. I've reused as much of his material as possible, and created my own where needed (results etc.) as well as added a few paragraphs and explanations here and there. In the translation process some points or facts might have been lost or might be downright wrong, in the event of confusion seek out the original document.

Only this document and content within is to be considered GPL; rights remain with the respective owners of the software used throughout the tutorial.

*Claus Andersen – March 2011*

## 2 Introduction part two

In this tutorial we will study the plastic deformation of an aluminum wheel rim. The use of cyclic symmetry will be introduced and the use of *POURSUITE* to break the study into two parts: calculation and post-processing.

Calculations are done with Code Aster® 10.3 Stable release

### 3 Preparation of the model

Figures 3.1 and 3.2 shows the geometry we will be working with and how the model is symmetric at  $120^\circ$

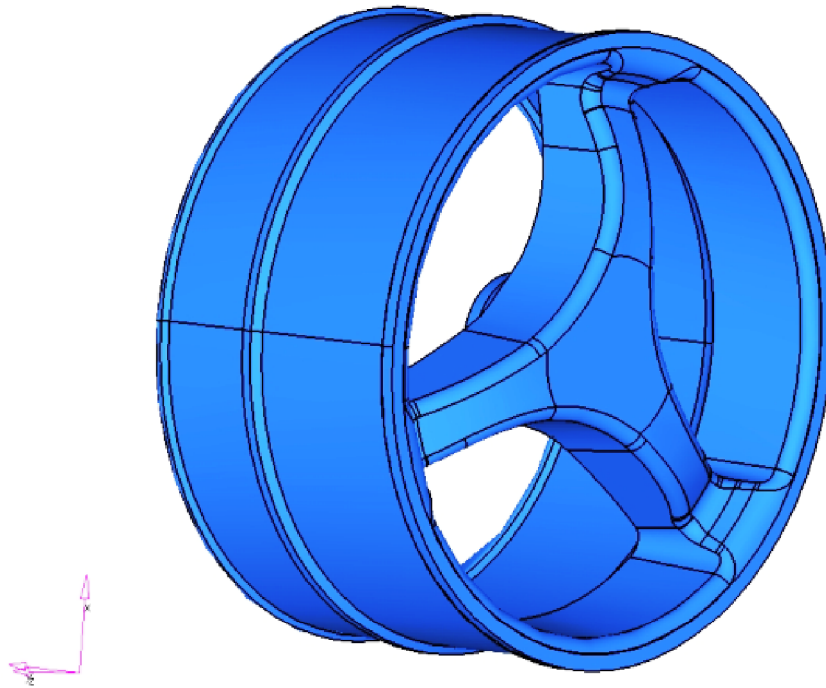


Figure 3.1: Full model

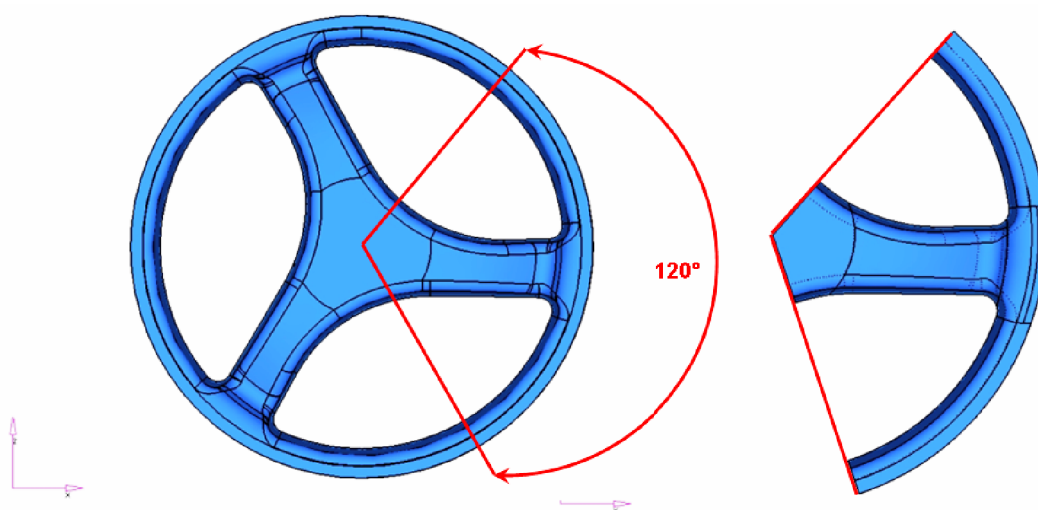


Figure 3.2: 1/3 part

## 4 The mesh

The mesh is mainly comprised of second order hexahedrals. See figures 4.1, 4.2 and 4.3

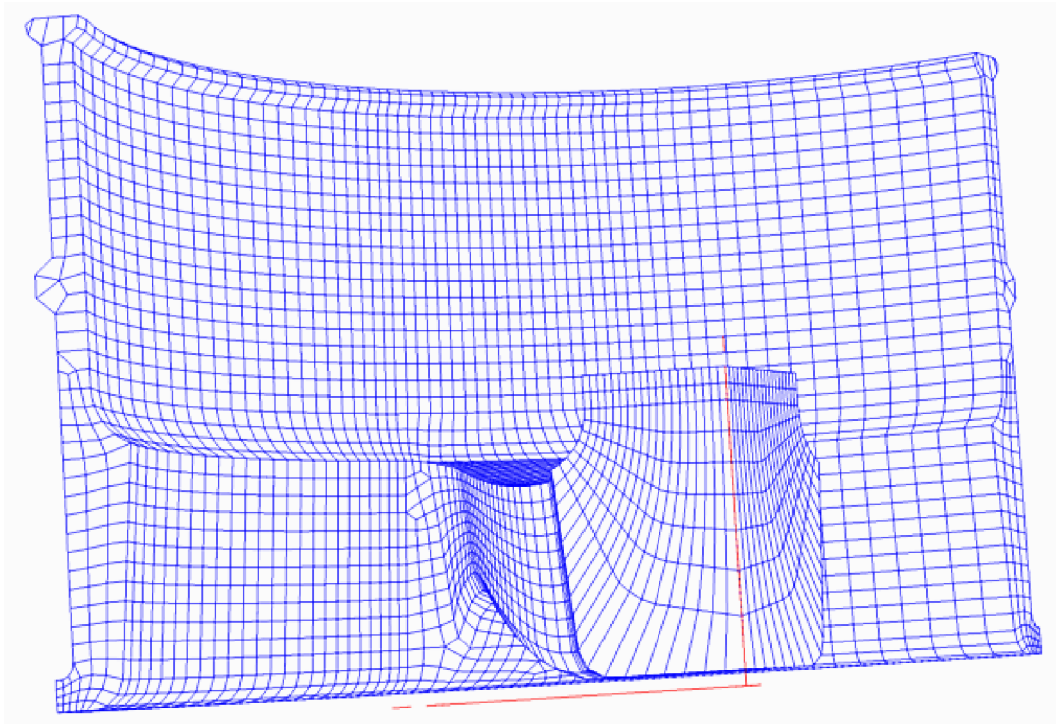


Figure 4.1: Mesh view 1

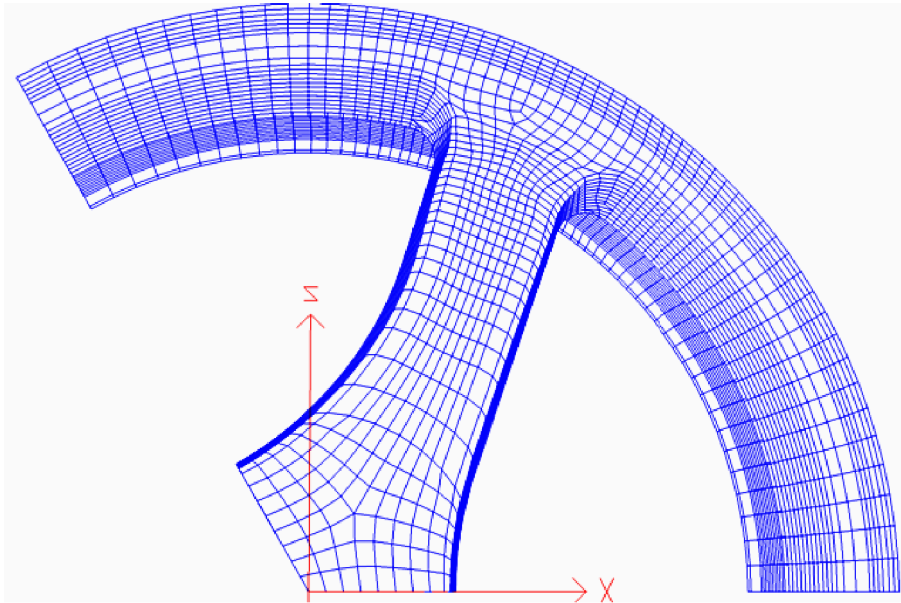


Figure 4.2: Mesh view 2

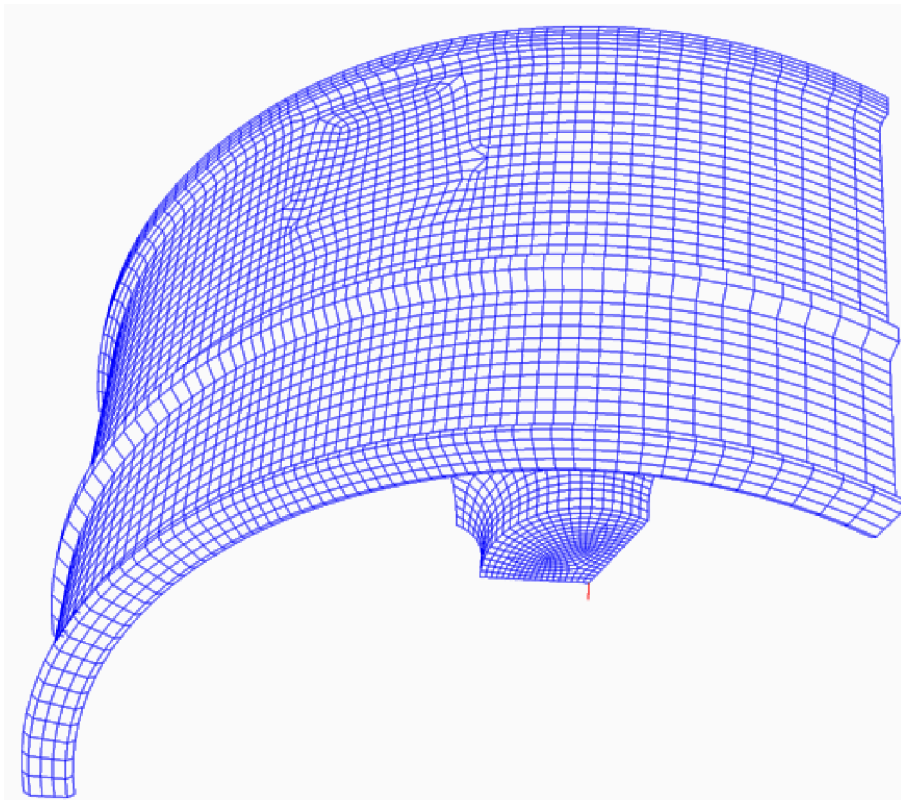
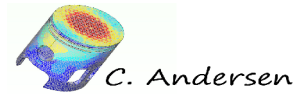


Figure 4.3: Mesh view 3



## 5 Objective of modelization

The imposed displacement at the wheel hub is somewhat a simplification of a real world scenario. If the simulation were to be closer to real world conditions, contact conditions should be introduced at the wheel hub and the support boundary conditions (extremities of the wheel rim) to allow for local rotation as the displacement is enforced.

In this case, it will result in an overestimation of the stresses and strains.

We seek to determine:

1. The elasto-plastic stress field resulting from the displacement imposed
2. Plastic strain resulting from the displacement imposed

## 6 Procedure of the simulation

As always with larger studies, it makes sense to break the simulation into two parts:

1. The mechanical non-linear calculation (**STAT\_NON\_LINE – [U4.51.03]**)
2. Post-processing the data saved from stage 1

This prevents the need to re-run the mechanical calculation just to change a parameter in the post-processing stage. In this particular case the mechanical calculation takes several hours, depending on the system, whereas the post-processing stage takes mere minutes.

### 6.1 Stage 1: Mechanical calculation

F	comm	./rim_solution.comm	D	1
F	mmed	./rim.med	D	20
R	base	./base	R	0
F	mess	./rim_solution.mess	R	6
F	resu	./rim_solution.resu	R	8

### 6.2 Stage 2: Post-processing of data – MED format

F	comm	./rim_postprocess.comm	D	1
F	rmed	./rim_resu.med	R	80
R	base	./base	D	0
F	mess	./rim_postprocess .mess	R	6
F	resu	./rim_postprocess .resu	R	8
F	dat	./rim_curve.dat	R	51

Above demonstrates the two setups in ASTK®; be sure to take special note of the letter denoting whether a file/folder is a result or data. A wrong setting in the post-processing



stage will render the data calculated in the mechanical stage, useless.

## 7 Chronological list of the stages

With the help of the command file editor EFICAS®, the following commands are applied:

- Loading the mesh
- Defining and assigning the material
- Assigning a finite element model (3D)
- Assigning cyclic symmetry
- Defining and assigning boundary conditions and load
- Solution
- Separate post-processing

## 8 Note on units in Code Aster

Code Aster does not have a unit system such as SI or Imperial; it is entirely up to the user to provide a coherent set of units. In this particular case mmNS is used, which means length is in mm, and pressure is in MPa (N/mm<sup>2</sup>)

## 9 Stage 1 - The command file

### 9.1 Loading the mesh and defining the finite element model

```
DEBUT();

#Read the mesh - name the concept 'Mesh'

Mesh=LIRE_MALLAGE (UNITE=20,
                   FORMAT='MED',);

#Definition & assignment of finite element model

Model=AFFE_MODELE (MALLAGE=Mesh,
                  AFFE=_F (GROUP_MA='RIM',
                          PHENOMENE='MECANIQUE',
                          MODELISATION='3D',),);
```

## 9.2 Definition of the material

The material chosen for this study, is an aluminum from the 6000 family with a Young's module of 70000 MPa, a Poisson's ratio of 0.3 and a yield strength (Rp0.2) of 105 MPa. The traction curve for this material is an experimental point by point curve generated from monotone traction on a test piece (See figure 9.1)

Plasticity is implemented in Code Aster in the form of  $\sigma = f(\epsilon_{total})$ . The abscissa (X coordinate) must not be zero: it corresponds to the elastic deformation  $\epsilon = \epsilon^e = \frac{R_{p0.2}}{E}$ .

The entire model is assigned this material.

#Definition of material

```
Traction=DEFI_FONCTION(NOM_PARA='EPSI',NOM_RESU='SIGM',VALE=(0.001
5,105.0,0.002,113.0,0.003,117.0,0.004,120.0,0.005,122.0,0.01,135.0
,0.02,155.0,0.03,172.0,0.04,186.0,0.05,198.0,0.06,208.0,0.07,218.0
,0.08,227.0,0.09,234.0,0.1,241.0,0.11,248.0,0.12,254.0,0.13,259.0,
0.14,264.0,0.15,268.0,0.16,273.0,0.17,276.0,0.18,280.0,0.19,283.0,
0.2,286.0,0.25,299.0,0.3,308.0,0.35,314.0,0.4,319.0,0.45,322.0,0.5
,325.0,0.9,333.0,),INTERPOL='LIN',PROL_DROITE='LINEAIRE',PROL_GAUC
HE='CONSTANT',);
```

```
A6000=DEFI_MATERIAU(ELAS=_F(E=70000,
NU=0.3,),
TRACTION=_F(SIGM=Traction,),);
```

```
Material=AFFE_MATERIAU(MAILLAGE=Mesh,
AFFE=_F(GROUP_MA='RIM',
MATER=A6000,),);
```

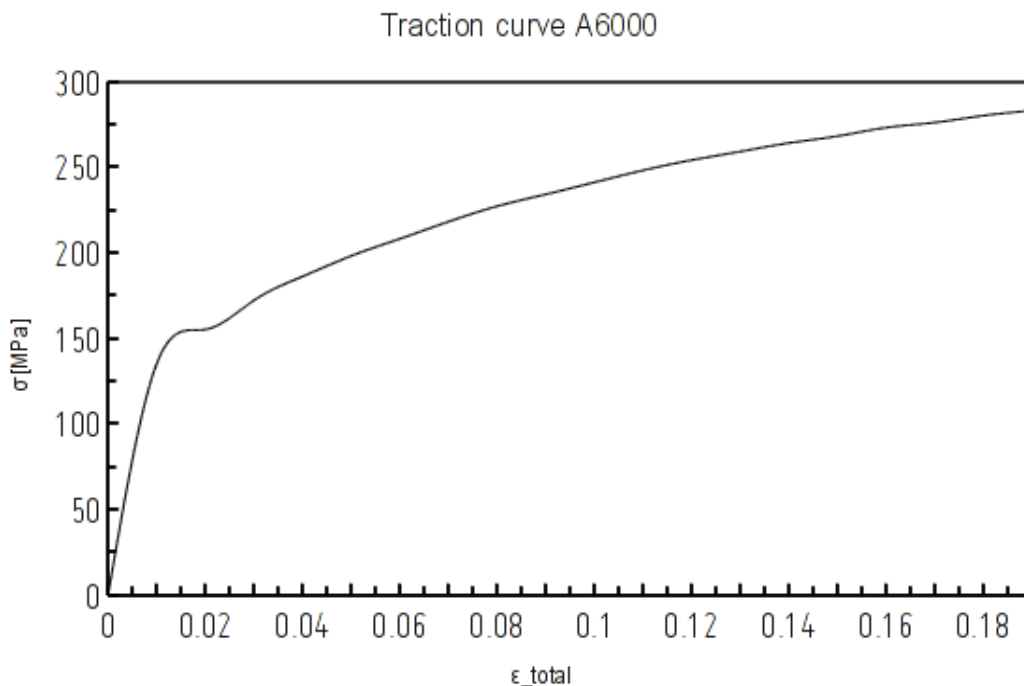


Figure 9.1: Traction curve for A6000

### 9.3 Converting element groups to node groups

Some of the keywords require node groups instead of element groups in order to perform their action – therefore we ask Code Aster® to convert these groups and retain their respective names.

#Definition of node groups based on element groups.

```
Mesh=DEFI_GROUP(reuse =Mesh,
                MAILLAGE=Mesh,
                CREA_GROUP_NO=( _F (GROUP_MA='PLAN_XOY',),
                                _F (GROUP_MA='PLAN_120',),
                                _F (GROUP_MA='suprt1',),
                                _F (GROUP_MA='suprt2',),
                                _F (GROUP_MA='disp1',),
                                _F (GROUP_MA='disp2',),),),);
```



## 9.4 Definition of the boundary conditions

Face- and volume-groups have only 3 translatable degrees of freedom (DDL), DX, DY and DZ – to avoid pivoting (singular matrix) and to have a static situation, the 'support' face-groups need to have these movements blocked. (See figures 9.2, 9.3 and 9.4)

The cyclic symmetry of the model is imposed with the command **LIAISON\_OBLIQUE** – [U4.44.01].

1. The plane of symmetry PLAN\_XOY is blocked from *global* Z-axis movements (DZ=0)
2. The plane of symmetry PLAN\_120 is blocked from Z-axis movements in its *local* coordinate system (as assigned by **LIAISON\_OBLIQUE**) – equivalent to blocking DX and DZ in the *global* coordinate system.
3. The extremity of the rim is blocked from movements in the Y-axis.

#Definition of boundary conditions

```
BC=AFFE_CHAR_MECA (MODELE=Model,
                    DDL_IMPO= (_F (GROUP_NO='PLAN_XOY',
                                   DZ=0.0, ),
                               _F (GROUP_NO='suprt1',
                                   DY=0.0, ), ),
                    LIAISON_OBLIQUE=_F (GROUP_NO='PLAN_120',
                                         ANGL_NAUT=(0, -120, 0),
                                         DZ=0.0, ), );
```

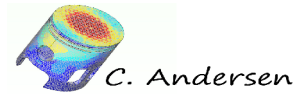
## 9.5 Load

A forced displacement is imposed on the hub of the rim (**disp1**) in the Y-axis. Note that unlike a linear elastic case (**MECA\_STATIQUE**) where only the final result is considered, every intermittent step of a nonlinear case must converge before the next step is initiated.

Assigning an appropriate time stepping list (one step = one time increment) for a nonlinear calculation, is a compromise of having the largest possible steps to save computations time, versus having time steps small enough to let the solution of each step converge.

An approach is to make a step neither too large nor too small and let Code Aster® automatically divide the step in case of non-convergence. A simplified approach has been applied here using the keyword **DEFI\_LIST\_INST** (as described in the next paragraph).

**DEFI\_LIST\_INST** takes an existent list of real numbers (**DEFI\_LIST\_REEL**) for the time stepping scheme, and in the event of non-convergence divides the time step 5 times



---

9.5 Load

---

(**SUBD\_PAS=5**). In the event of non-convergence in the newly created (smaller) time steps, it divides the time step again, up to 4 times (**SUBD\_NIVEAU=4**).

For this study, the hub of the rim is displaced 10mm linearly over the course of the time steps using the function '**LoadRamp**' with **AFFE\_CHAR\_MECA\_F**.

#Definition of load : displacement imposed

```
LoadRamp=DEFI_FONCTION(NOM_PARA='INST',VALE=(0,0,
                                             10,10,
                                             ),INTERPOL='LIN',PROL_DROITE='LINEAIR
E',PROL_GAUCHE='CONSTANT',);
```

```
Load=AFFE_CHAR_MECA_F(MODELE=Model,
                      DDL_IMPO=_F(GROUP_NO='displ',
                                   DY=LoadRamp,),);
```

## 9.5.1 Group diagram

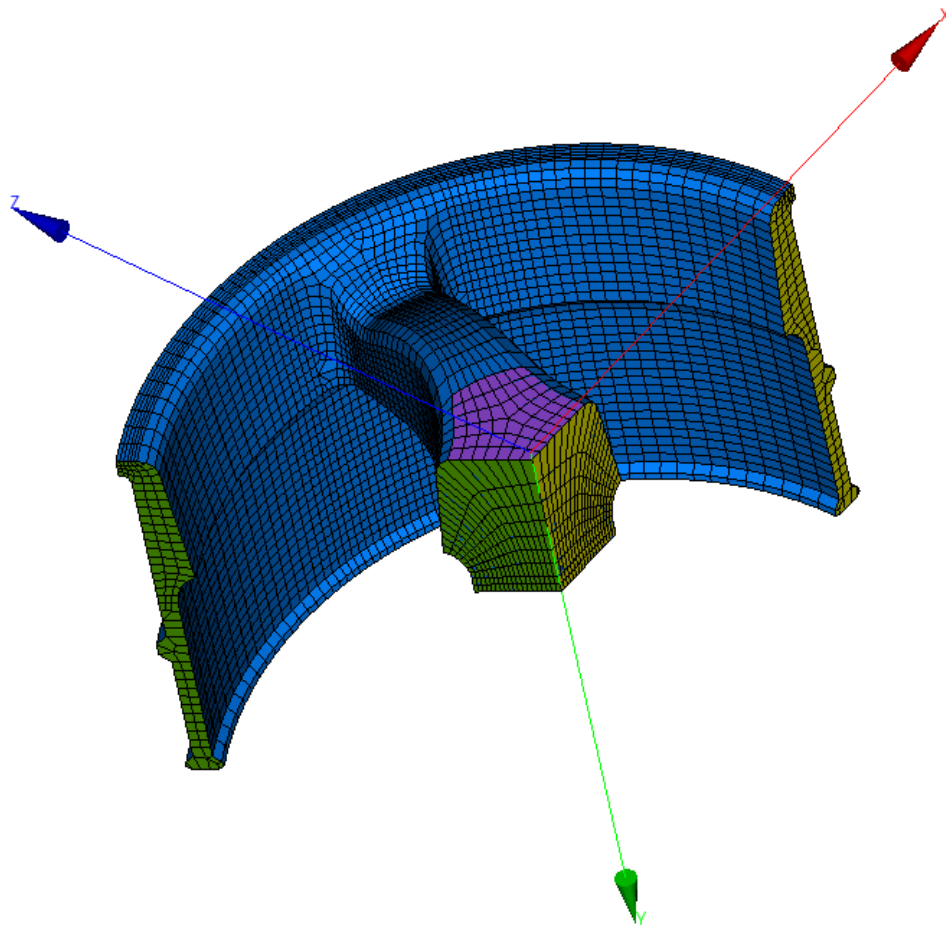


Figure 9.2: Group diagram

### 9.5.1 Group diagram

---

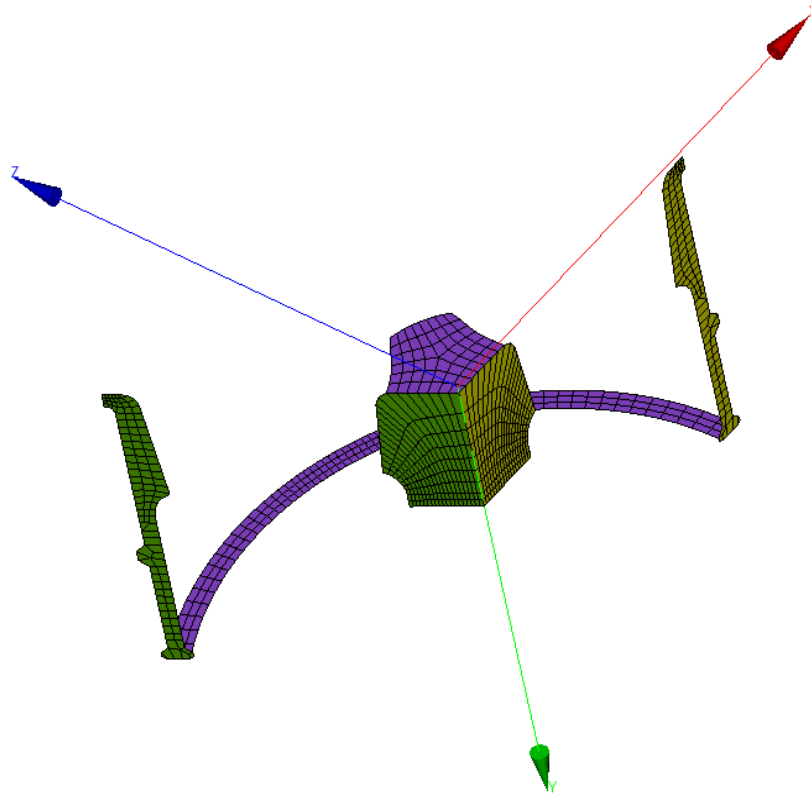


Figure 9.3: Only showing groups

### 9.5.1 Group diagram

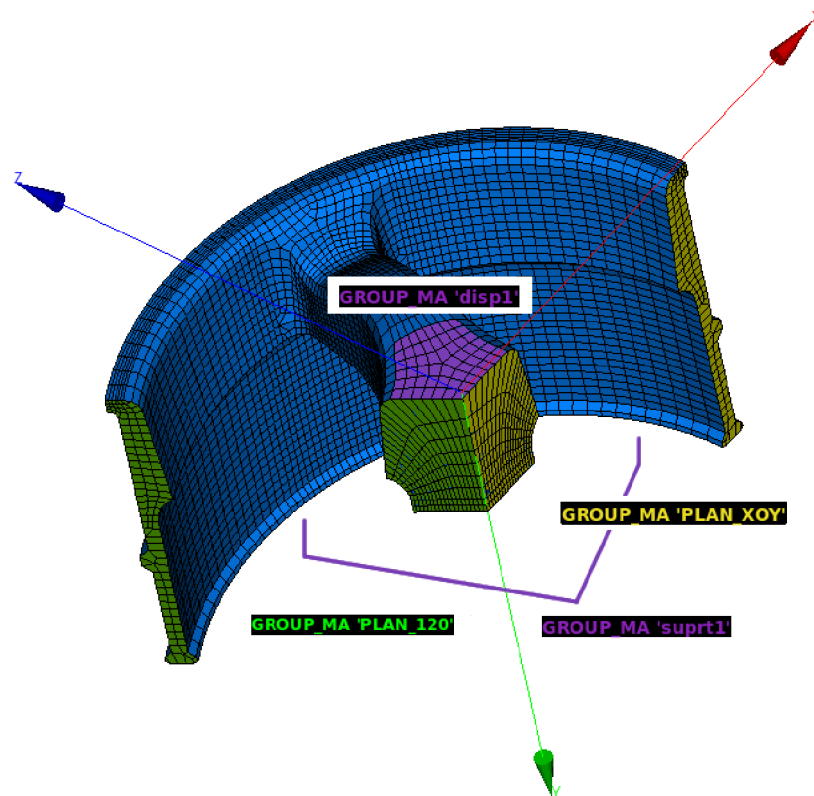


Figure 9.4: Group diagram with names



## 9.5.2 LIAISON\_OBLIQUE diagram

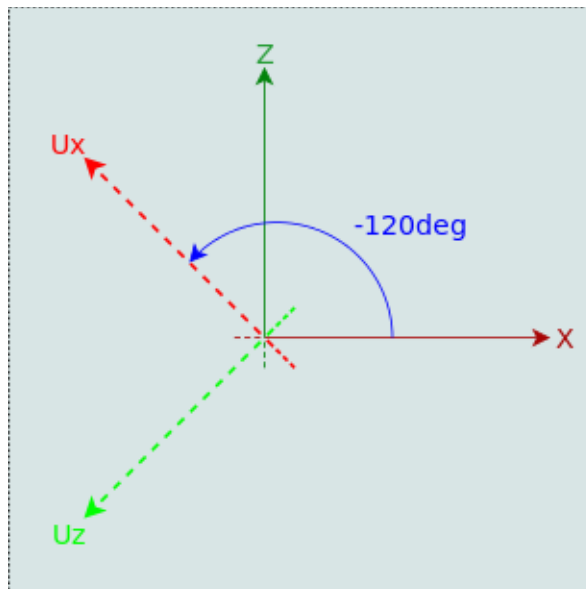


Figure 9.5: Global csys and local csys as assigned by LIAISON\_OBLIQUE

## 9.6 Definition of lists

#Definition of time steps - real numbers

```
ListReel=DEFI_LIST_REEL(DEBUT=0,
                        INTERVALLE=_F(JUSQU_A=10,
                                      PAS=0.1),);
```

#Definition of time stepping list with subdivision of time steps in the event of non-convergence

```
ListInst=DEFI_LIST_INST(DEFI_LIST=_F(METHODE='MANUEL',
                                      LIST_INST=ListReel),
                        ECHEC=_F(EVENEMENT='ERREUR',
                                SUBD_PAS=5,
                                SUBD_NIVEAU=4,
                                SUBD_METHODE='UNIFORME'),);
```

#List of time steps that should be archived (saved) by the solver

```
Archive=DEFI_LIST_REEL(DEBUT=0,
                       INTERVALLE=_F(JUSQU_A=10,
                                      PAS=1),);
```

## 9.7 Nonlinear calculation

Some simple parameters for the nonlinear solver is set up; we'll go through them briefly.

## 9.8 COMP\_INCR

COMP\_INCR is an abbreviation of *Comportement Incremental*, which translates to *Incremental Behavior*. In effect, this means that Code Aster® keeps a history of the material behavior for each time step, and in the case of plasticity, follows the traction curve to determine whether (and how far) the material has entered the plastic domain, based on the previous time step.

The deformation per time step is calculated as  $d\epsilon_{total} = d\epsilon^{elastic} + d\epsilon^{plastic}$

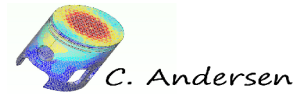
The plasticity is determined as  $d\epsilon^{pl} = d\lambda \cdot \frac{\delta \Phi}{\delta \sigma}$  where the plastic potential  $\Phi$  and type of hardening (isotropic, kinematic or other) is determined by the keyword **RELATION**:

VMIS\_ISOT\_TRAC; which means the material has a plastic potential based on Von Mises (widespread in ferrous metals), has nonlinear isotropic hardening and utilizes a traction curve.

## 9.9 DEFORMATION

Different formulations are available to describe the deformation of the material:

1. **PETIT**: Classic formulation where the strain tensor is linearized for each small change ( $\epsilon_{ij} = \frac{1}{2} [grad U_{ij} + {}^t grad U_{ij}]$ ). This formulation is valid for strain up till 5%.
2. **PETIT\_REAC**: Still remaining with small strains, but with a mesh that is updated on each time step, this formulation can be used with great deformation, provided local rotation stays small and small time steps are used.
3. **GROT\_GDEP**: Allows for great rotation and great displacement, but with small strains.
4. **SIMO\_MIEHE**: Allows for great strains and plastic deformation (and great rotation / displacement) – Only accepts elements of the type 3D, 3D\_INCO, AXIS, AXIS\_INCO, D\_PLAN, PLAN\_INCO and the behaviors ELAS, VMIS\_ISOT\_LINE, VMIS\_ISOT\_TRAC and ROUSSELIER
5. Other types exists as well but not described here: **GDEF\_HYPO\_ELAS**, **GREEN\_REAC**



For more information, read the document [U4.51.11] **Comportements non linéaires** and the reference papers [R5.03.02] and [R5.03.21].

## 9.10 INCREMENT

Uses the previously defined list '**ListInst**' that automatically subdivides time steps in the event of non-convergence.

## 9.11 NEWTON

Two keywords are used here:

1. **PREDICTION**: Using the rate of change of the matrix tangent (See [U4.51.03])
2. **REAC\_ITER**: A value of 0 (zero) means the matrix tangent is updated at the beginning of each time step, whereas a value of 1 means the matrix tangent is updated at each increment of the time step (this can improve convergence, but at the cost of increased computational time).

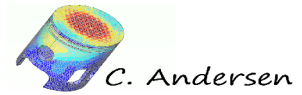
## 9.12 CONVERGENCE

Here we define the greatest residual value we can accept if we are to consider the results valid and continue to the next time step – we also define how many iterations each time step can have before either stopping the calculation or subdividing the time step.

1. **RESI\_GLOB\_MAXI / RESI\_GLOB\_RELA**: A maximum value of  $10^{-4}$  is considered 'okay' for this study, but for greater precision, a value of e.g.  $10^{-6}$  should be used, at the cost of greater computational time of course.
2. **ITER\_GLOB\_MAXI**: A maximum of 10 iterations per time step can be used to reach the above convergence criteria, otherwise stop the the calculation or subdivide the time step.

## 9.13 ARCHIVAGE

Considering that the calculation might subdivide many time steps and perhaps several times, generating a lot of output, an alternative list is used to tell the solver only to save certain time steps for later post processing. The keyword **LIST\_INST** uses the list of real numbers '**Archive**' to tell the solver to save only 10 time steps (calculation ends at **JUSQU\_A=10** and **PAS=1** means that time steps 1,2,3,4,5,6,7,8,9,10 will be saved)



### 9.13 ARCHIVAGE

---

#Nonlinear solution

```
RESOL_NL=STAT_NON_LINE (MODELE=Model,
                        CHAM_MATER=Material,
                        EXCIT=(_F (CHARGE=BC, ),
                                _F (CHARGE=Load, ), ),
                        COMP_INCR=_F (RELATION='VMIS_ISOT_TRAC',
                                        DEFORMATION='SIMO_MIEHE', ),
                        INCREMENT=_F (LIST_INST=ListInst, ),
                        NEWTON=_F (PREDICTION='TANGENTE',
                                    REAC_ITER=0, ),
                        CONVERGENCE=_F (RESI_GLOB_MAXI=1e-4,
                                        RESI_GLOB_RELA=1e-4,
                                        ITER_GLOB_MAXI=10, ),
                        ARCHIVAGE=_F (LIST_INST=Archive,
                                        ARCH_ETAT_INIT='OUI', ), );

FIN();
```

## 10 Stage 2 – Post processing

As noted before, the second stage of this study utilizes the previously obtained data generated from the mechanical calculation, to further process the data.

The second command file begins with the keyword **POURSUITE - [U4.11.03]** which tells Code Aster® to use the data contained in the 'base' folder. As such, when opening a 'poursuite command file' in EFICAS®, EFICAS® will ask which command file it should continue from (i.e. *rim\_solution.comm*) . This is so that it knows what concept names are available/valid for post processing.

### 10.1 The command file

#### 10.1.1 Calculating elements and nodes

For this study, we will use two output file formats: **POS** and **MED** – this ensures that the results can be read by both Salomé® and GMSH® - if one prefers one over the other.

Results are obtained from the integration points (Gauss points - ELGA) for optimal accuracy. More on this subject later.

By using the same concept name as used for the nonlinear solver (**RESOL\_NL**), we tell Code Aster® to 'enrich' the obtained results, with values calculated on the elements. The same is done when values on the nodes are calculated.

For the element calculation, values are only calculated at the last time step (**INST=10**), whereas the values for the forces on the nodes (**FORC\_NODA**) are calculated for the entire simulation (**TOUT\_ORDRE=OUI**) – this is because we need the evolution of the node forces later on. Furthermore we only calculate the node forces on the group '**disp1**'.

Element fields calculated:

- **SIEF\_ELNO\_ELGA**: Principal stresses SXX, SYX, SZZ, SXY... etc.
- **VARI\_ELGA**: Internal variables from the nonlinear calculation – plasticity et al.
- **EQUI\_ELGA\_EPSI**: Equivalent strain calculated at integration points.
- **EQUI\_ELGA\_SIGM**: Equivalent stress calculated at integration points. Von Mises stress, Tresca etc.
- Note: The displacement field **DEPL** and **VARI\_ELGA** does not need to be calculated as it is extracted directly from the mechanical calculation.



### 10.1.1 Calculating elements and nodes

---

```
POURSUIITE();

#Calculate values at integration points (Gauss - ELGA) and on
element nodes (ELNO)

RESOL_NL=CALC_ELEM(reuse =RESOL_NL,
                  RESULTAT=RESOL_NL,
                  INST=10,
                  OPTION=('SIEF_ELNO_ELGA','EQUI_ELGA_EPSI','EQUI
_ELGA_SIGM',),),);

#Calculate values at nodes

RESOL_NL=CALC_NO(reuse =RESOL_NL,
                RESULTAT=RESOL_NL,
                TOUT_ORDRE='OUI',
                OPTION='FORC_NODA',
                GROUP_NO_RESU='disp1',);
```

### 10.1.2 Writing the results to MED

```
#Write/print the results (to a mesh in this case)

IMPR_RESU(FORMAT='MED',
          RESU=_F(MAILLAGE=Mesh,
                RESULTAT=RESOL_NL,
                NOM_CHAM=('DEPL','EQUI_ELGA_SIGM','EQUI_ELGA_EPS
I','VARI_ELGA','EPSI_ELGA_DEPL','FORC_NODA',),
                INST=10,),),);
```

### 10.1.3 Writing the results to POS

GMSH® cannot display the vector components from a field in a **MED** file, but when writing the results to a **POS** file, all the vectors and scalars are presented in the main windows, and this can however, can create somewhat of a mess. Because of this, we divide the **IMPR\_RESU** keyword up into four parts, and for each field we select what components we want to include (**NOM\_CMP**). In the special case of the displacement, we tell Code Aster® that we want the resultant (**TYPE\_CHAM=3D\_VECT**) of the three displacement vectors DX,DY,DZ.

```
IMPR_RESU (FORMAT='GMSH',
           UNITE=80,
           RESU= (_F (MAILLAGE=Mesh,
                     RESULTAT=RESOL_NL,
                     NOM_CHAM='EQUI_ELGA_SIGM',
                     INST=10,
                     NOM_CMP='VMIS',),
                 _F (MAILLAGE=Mesh,
                     RESULTAT=RESOL_NL,
                     NOM_CHAM='DEPL',
                     INST=10,
                     TYPE_CHAM='VECT_3D',
                     NOM_CMP=('DX','DY','DZ',),),),
                 _F (MAILLAGE=Mesh,
                     RESULTAT=RESOL_NL,
                     NOM_CHAM='EQUI_ELGA_EPSI',
                     INST=10,
                     NOM_CMP='INVA_2',),
                 _F (MAILLAGE=Mesh,
                     RESULTAT=RESOL_NL,
                     NOM_CHAM='VARI_ELNO_ELGA',
                     INST=10,
                     NOM_CMP=('V1','V8',),),),),);
```

Note: Normally a **POS** file will have a unit number of 37, but any number can be assigned with the **UNITE** keyword – it must of course have a corresponding number in **ASTK**®.

## 10.2 Reviewing the results

### 10.2.1 Deformation

Figures 10.1, 10.2 and 10.3 shows the deformed shape compared to the original mesh at last instant.

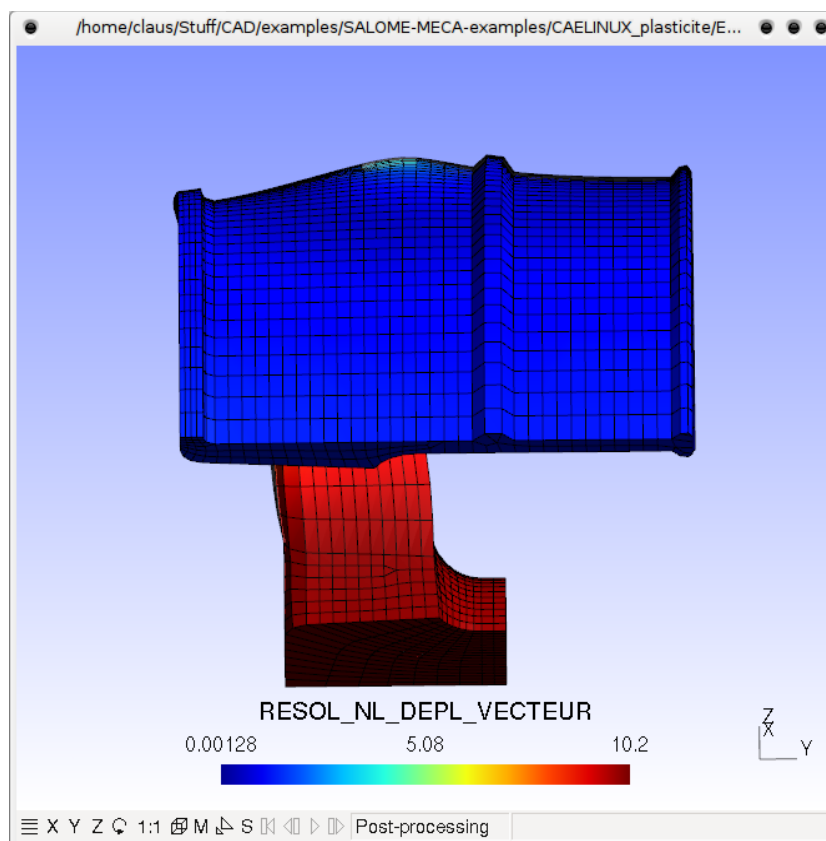


Figure 10.1: Deformation with continuous mapping



### 10.2.1 Deformation

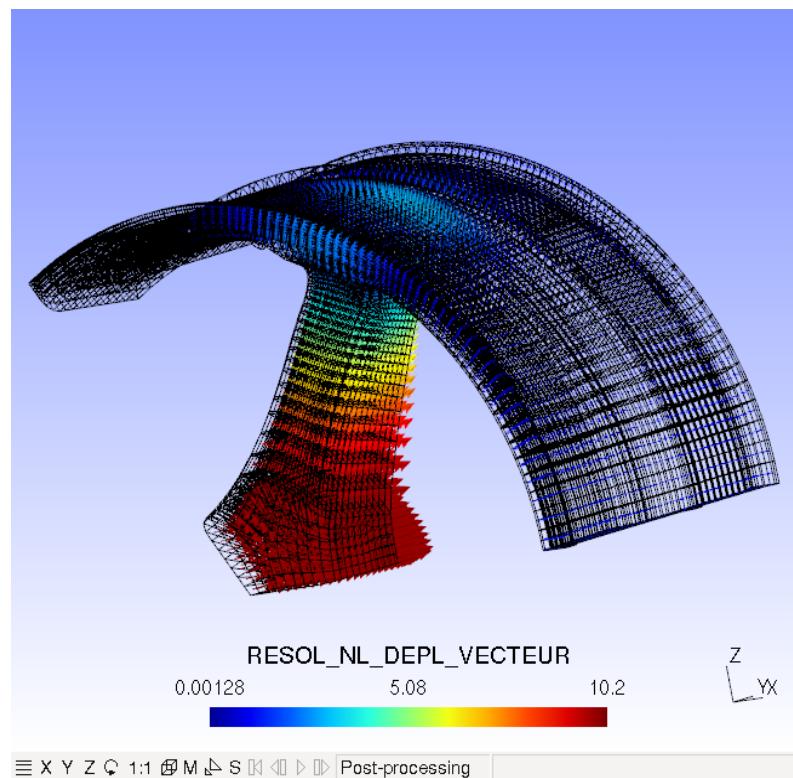


Figure 10.2: Deformation with vector arrows

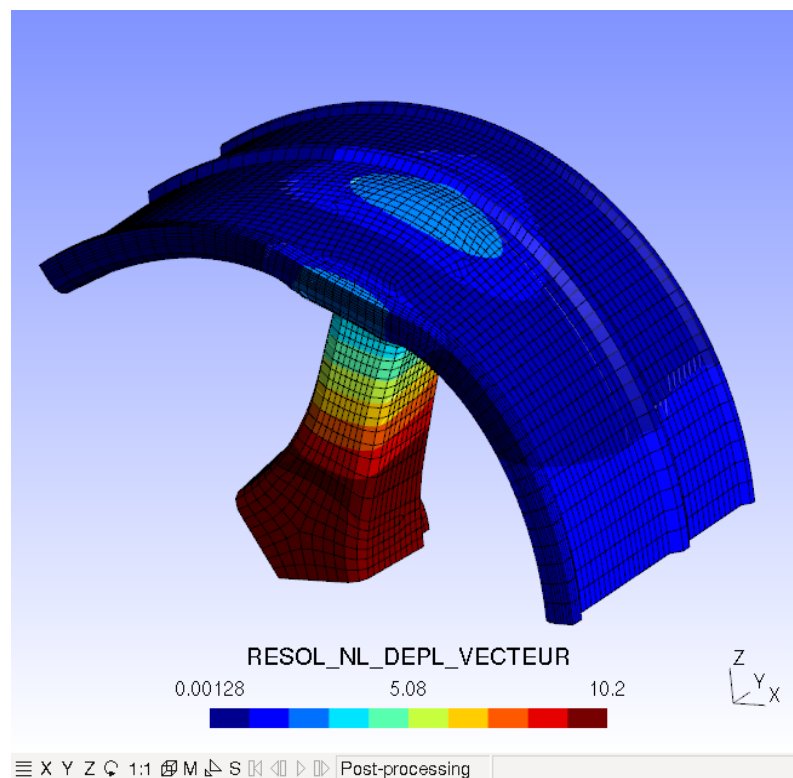


Figure 10.3: Deformation with iso-surfaces

## 10.2.2 Equivalent von Mises stress

Figures 10.4 and 10.5 shows equivalent von Mises stress (**EQUI\_ELGA\_SIGM**, **CMP=VMIS** for **POS** files) at last instant.

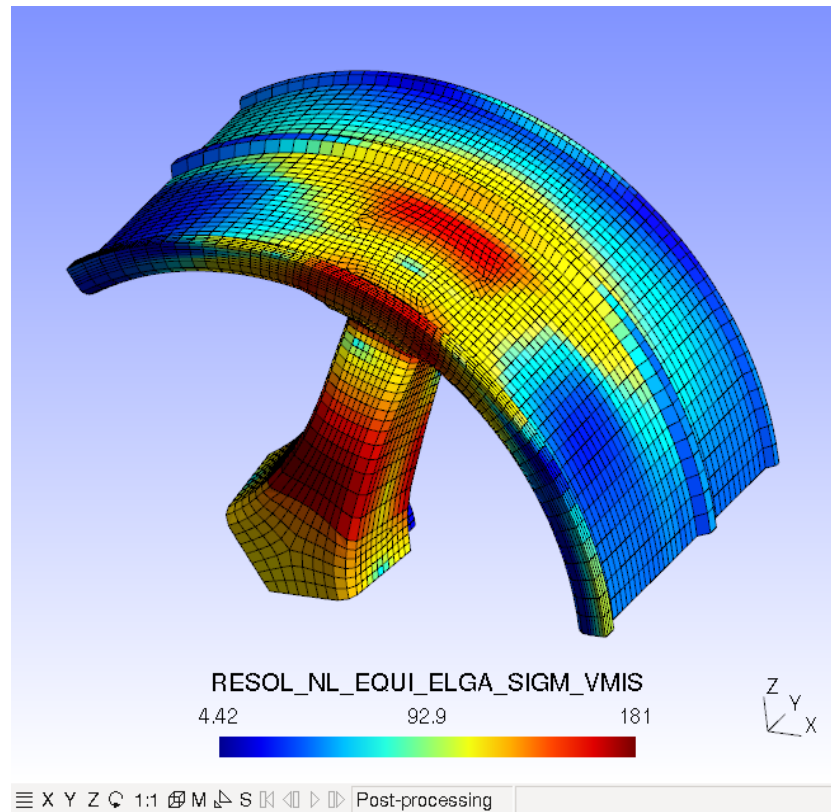


Figure 10.4: Equivalent von Mises stress – continuous mapping

### 10.2.2 Equivalent von Mises stress

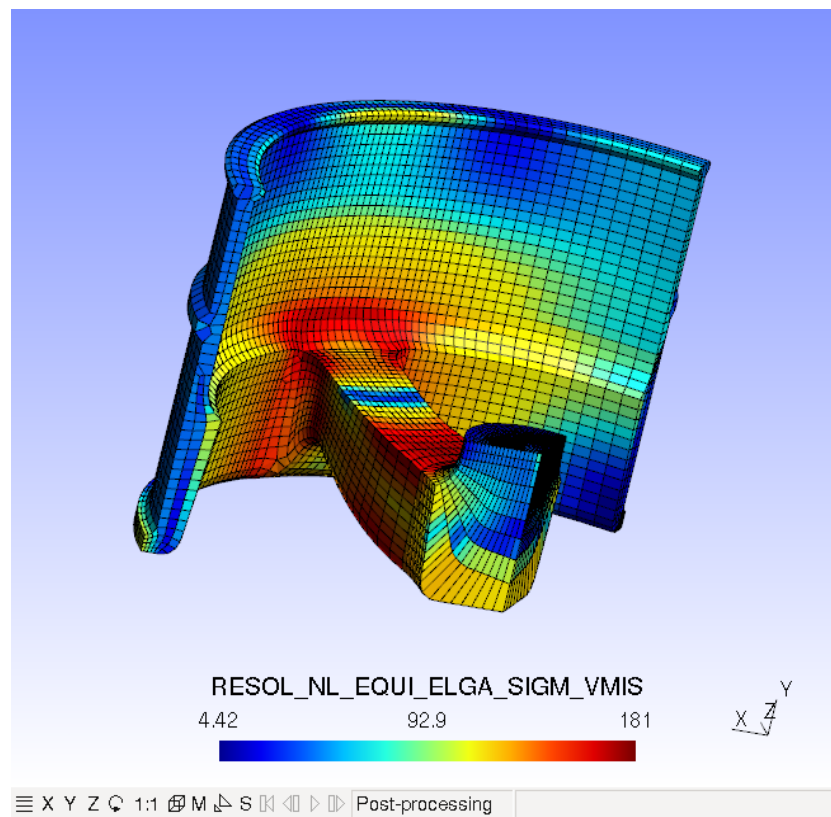


Figure 10.5: Equivalent von Mises stress – continuous mapping

### 10.2.3 Equivalent strain

Figures 10.6 and 10.7 shows equivalent strain (**EQUI\_ELGA\_EPSI, CMP=INVA\_2**) at last instant.

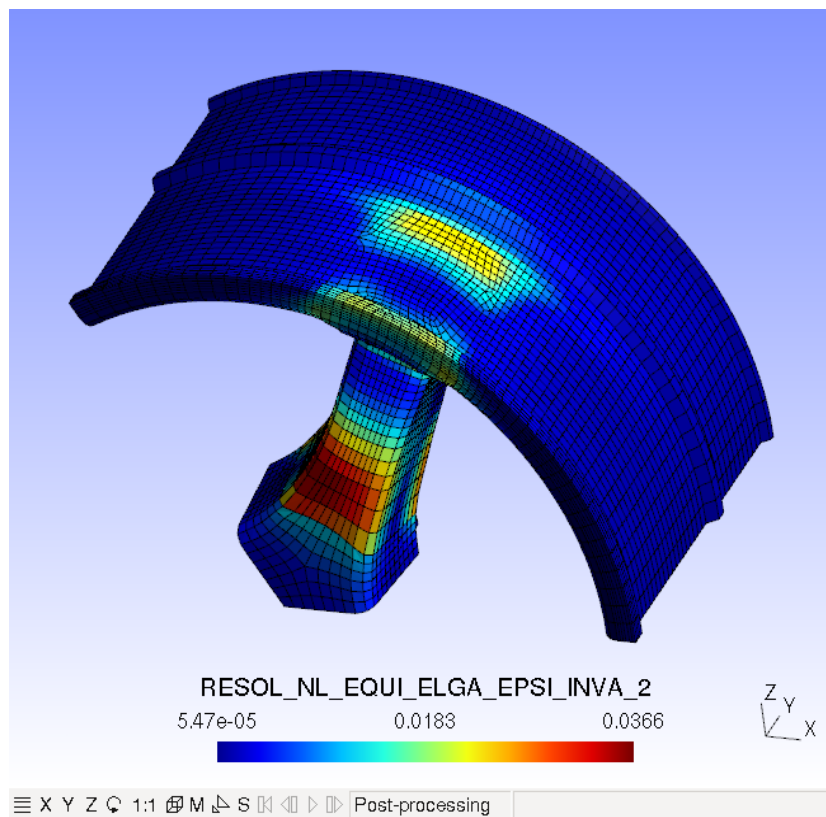


Figure 10.6: Equivalent strain, continuous mapping

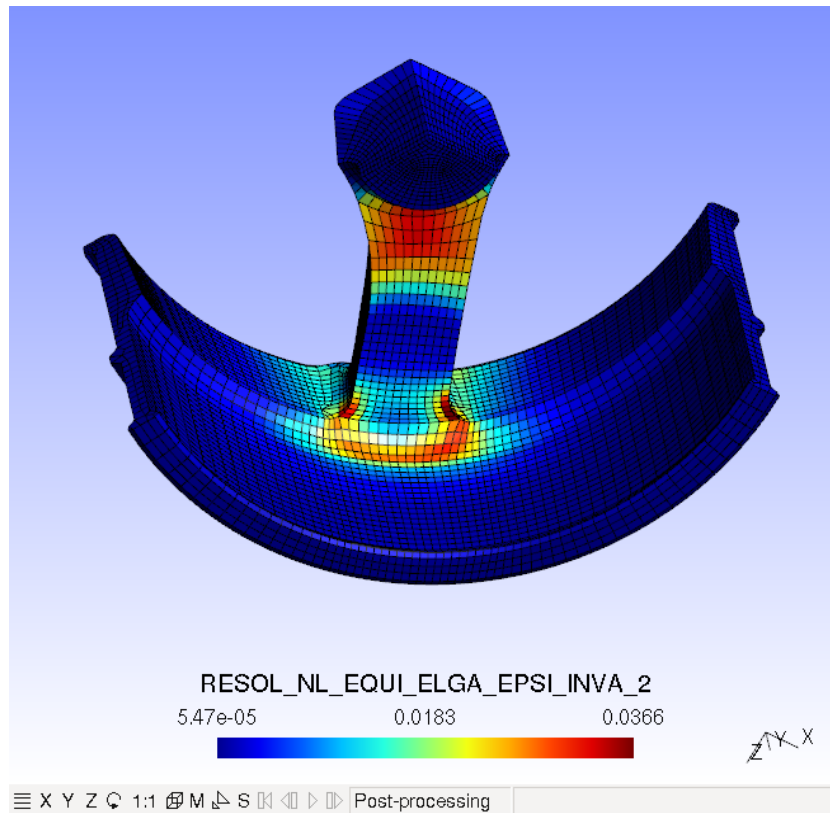


Figure 10.7: Equivalent strain, continuous mapping

## 10.2.4 Cumulative plastic strain

Figure 10.8 shows the cumulative plastic strain (**VARI\_ELGA, CMP=V1**)

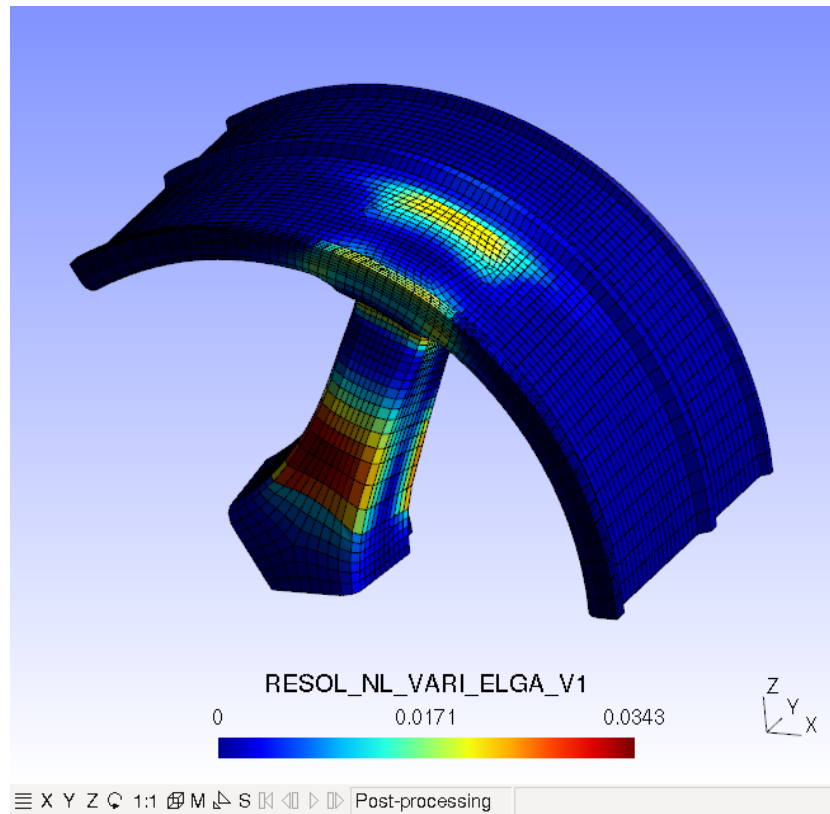


Figure 10.8: Cumulative plastic strain

## 10.2.5 Indication of plasticity

Figures 10.9 and 10.10 indicates where the material has yielded (**VARI\_ELGA, CMP=V8**). A value of 0 indicates no yield whereas a value of 1 indicates material yield.

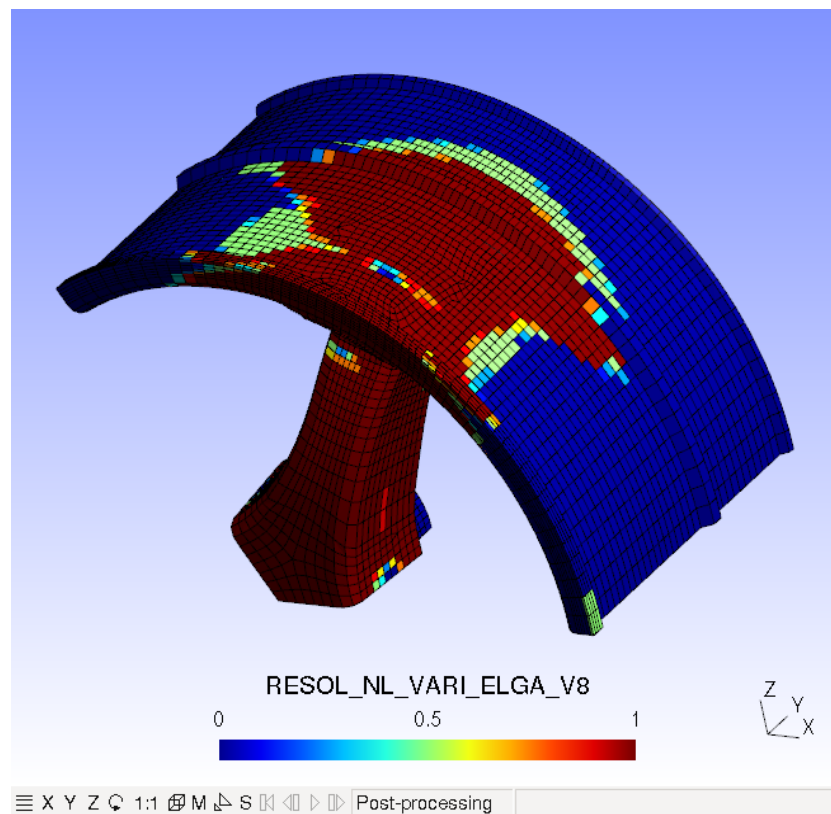


Figure 10.9: Indication of yield

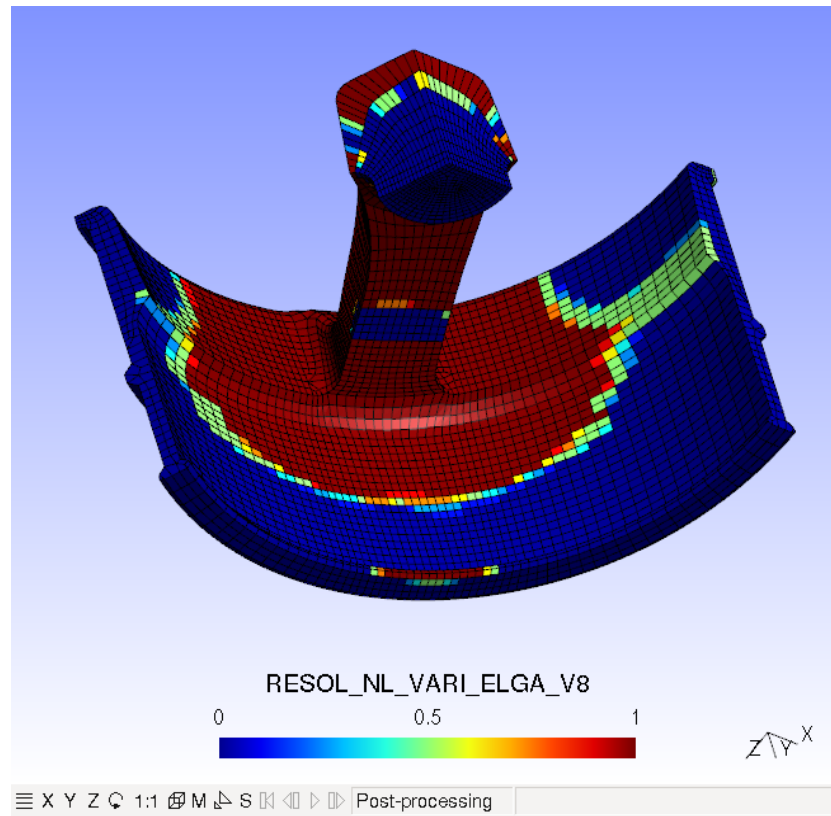


Figure 10.10: Indication of yield

### 10.3 Force vs. displacement on the rim hub

To show how forces evolve on the imposed displacement, the node reactions must be extracted on the data post-processed. This requires a few steps:

- Extract nodal forces and displacements for nodes
- Write extracted results to a table (and a file for review)
- 'Recover' a function for both forces and displacements
- Print (write) the generated curve for force vs. displacement with XMGrace®

Lets take the process step by step.





### 10.3.1 Extracting nodal forces

Using the integrated post processor keyword **POST\_RELEVE\_T** - [U4.81.21], each *ACTION* is assigned an extraction operation and a unique name (*INTITULE*) for later identification.

The previous result concept is reused, and a named field of the result is specified. In this case we want to extract values from displacement and nodal forces – **DEPL** and **FORC\_NODA**.

**TOUT\_ORDRE** denotes that we want values from all instants, we want the values from a single node (**NOEUD=Nxxxx**) and finally which vectorial component to extract (**NOM\_CMP=DY**)

(Only one *action* shown here)

```
POST_R=POST_RELEVE_T (ACTION= (_F (OPERATION='EXTRACTION',
                                   INTITULE='Uy',
                                   RESULTAT=RESOL_NL,
                                   NOM_CHAM='DEPL',
                                   TOUT_ORDRE='OUI',
                                   NOEUD='N7962',
                                   NOM_CMP='DY', ),
```

Write the **POST\_R** table to the *.resu* file – if left blank, unit number 8 is used, which is the default for the *.resu* file – any other unit number / file is also valid.

```
IMPR_TABLE (TABLE=POST_R, );
```

### 10.3.2 Recover a function for Uy and Fy

**RECU\_FONCTION** - [U4.32.03] accepts the previously created table; each instant as *X* parameter, and as *Y* parameter the component **DY**. By using the keyword **FILTRE** and **NOM\_PARA / VALE\_K**, we tell the post processor to extract values from the column *INTITULE* with the rows named *Uy* and *Fy* respectively. Furthermore from the set, extract the values from the columns *INST* and *DY*.

```
Dy=RECU_FONCTION (TABLE=POST_R,
                  PARA_X='INST',
                  PARA_Y='DY',
                  FILTRE=_F (NOM_PARA='INTITULE',
                           VALE_K='Uy', ), ), );
```

```
FY=RECU_FONCTION (TABLE=POST_R,
```



### 10.3.2 Recover a function for Uy and Fy

```

    PARA_X='INST',
    PARA_Y='DY',
    FILTRE=_F(NOM_PARA='INTITULE',
              VALE_K='Resultante',),),);

```

### 10.3.3 Write / print the curve with XMGrace®

To write a curve with XMGrace®, we tell Code Aster® to use XMGrace® as for-matter, and the *file* format Postscript.

The curve is composed of the previously created functions **Uy** and **Dy**.

(The meaning of the rest of the settings can be found in **IMPR\_FONCTION – [U4.33.01]**)

```

IMPR_FONCTION(FORMAT='XMGRACE',
              PILOTE='POSTSCRIPT',
              UNITE=51,
              COURBE=_F(FONC_X=Dy,
                        FONC_Y=Fy,
                        LEGENDE='Fy vs. Uy',
                        STYLE=8,
                        COULEUR=1,
                        MARQUEUR=1,),
              TITRE='Force VS displacement',
              LEGENDE_X='Uy [mm]',
              LEGENDE_Y='Fy [N]',),);
FIN();

```

Alternatively one can extract the values from the group **disp1**, and use the resultant vector instead. This way, the reaction force can be divided over the area of the hub. Remember to multiply the force by 3, since we work with 1/3 of the model. The surface of the imposed displacement at the hub is ~9000mm<sup>2</sup> (Figure 10.11)

10.3.3 Write / print the curve with XMGrace®

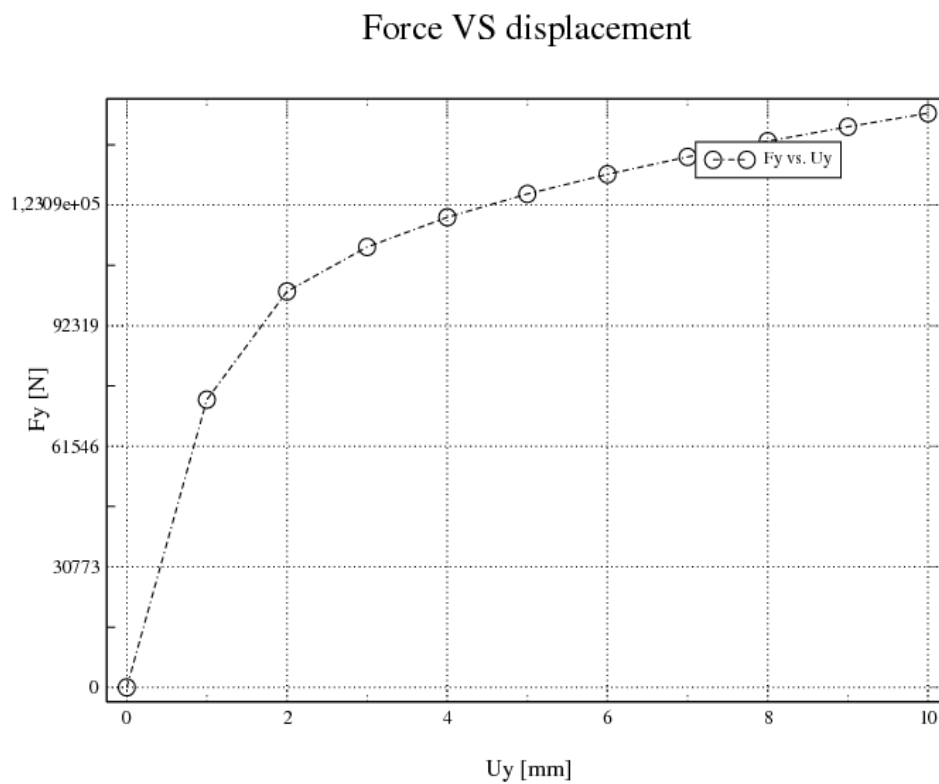


Figure 10.11: Force / displacement curve

## 11 Getting accurate results with Code Aster®

### 11.1 Integration points (Gauss points)

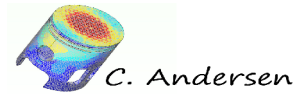
In this tutorial every field we've extracted and printed, we've used the fields generated from the integration points on the elements, the Gauss points (**ELGA** – element Gauss). This is because these are the exact values calculated by Code Aster® - every other type of field such as elements nodes (**ELNO**) and nodes (**NOEU**) are generated by extrapolating and interpolating the values from the integration points. While these fields (**ELNO** and **NOEU**) generates very pretty fields in post processors (such as Salomé®), this interpolation/extrapolation of values leads to inaccurate or ridiculous results, such as negative von Mises stress.

### 11.2 Linear or quadratic elements?

Linear triangles (TRIA3) and tetrahedrons (TETRA4) should never be used for anything other than adjusting time steps, preliminary results and such. For anything else, the element type is too stiff and will not generate reliable results. One must convert them to quadratic (2<sup>nd</sup> order) elements (TRIA6 and TETRA10) using either a preprocessor or within Code Aster®.

Quadrangles and hexahedrons behave much better with finite element programs and one should strive to use them where possible. Quads and hexas however, also benefit from being converted into quadratic elements. Especially in areas where changes in the stress-, strain gradients etc. are steep.

At the cost of increased computational time, quadratic quadrangle/hexahedron elements will always provide a superior and accurate result.



## 12 Conclusion, remarks and author(s)

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)

Original author: Paul Carrico – paul.carrico\_[at]\_free.fr

Translator etc.: Claus Andersen – ClausAndersen81\_[at]\_gmail.com

## 13 System of units

Unit	Metric MKS	Metric MmNS	Anglo-Saxon FPS	Anglo-Saxon IPS
Length	m	mm	ft	in
Time	sec	sec	sec	sec
Mass	Kg	tonne	slug	Lbf-sec <sup>2</sup>
Force	N	N	lbf	lbf
Temperature	°C	°C	°F	°F
Area	m <sup>2</sup>	mm <sup>2</sup>	ft <sup>2</sup>	in <sup>2</sup>
Volume	m <sup>3</sup>	mm <sup>3</sup>	ft <sup>3</sup> (cu-ft)	in <sup>3</sup> (cu-in)
Velocity	m/sec	mm/sec	ft/sec	in/sec
Acceleration	m/sec <sup>2</sup>	mm/sec <sup>2</sup>	ft/sec <sup>2</sup>	in/sec <sup>2</sup>
Angle, rotation	rad	rad	rad	rad
Angular velocity	rad/sec <sup>2</sup>	rad/sec <sup>2</sup>	rad/sec <sup>2</sup>	rad/sec <sup>2</sup>
Volumetric mass	Kg/m <sup>3</sup>	Tonne/mm <sup>3</sup>	slug/ft <sup>3</sup>	lbf-sec <sup>2</sup> /in <sup>4</sup>
Moment, couple	N-m	N-mm	Ft-lbf	In-lbf
Linear force	N/m	N/mm	lbf/ft	lbf/in
Surface force (Stress, pressure, Young's module)	N/m <sup>2</sup> (Pa)	N/mm <sup>2</sup> (Mpa)	lbf/ft <sup>2</sup>	lbf/in <sup>2</sup> (Psi)
Thermal dilation coefficient	/° C (/K)	/° C (/K)	/° F (/K)	/° F (/K)
2nd moment of inertia in a beam I <sub>gz</sub>	m <sup>4</sup>	mm <sup>4</sup>	ft <sup>4</sup>	in <sup>4</sup>
Transverse inertial moment in a beam	Kg-m <sup>2</sup>	Tonne-mm <sup>2</sup>	Slug-ft <sup>2</sup>	Lbf-in-sec <sup>2</sup>
Energy, work, heat	J	mJ	Ft-lbf	In-lbf
Energy, transfer rate	W	mW	ft-lbf/sec	in-lbf/sec
Temperature gradient	° C/m	° C/mm	° F/ft	° F/in
Thermal flux	W/m <sup>2</sup>	mW/mm <sup>2</sup>	lbf/ft-sec	lbf/in-sec
Thermal conductivity	W/m-° C	mW/mm-° C	lbf/sec-° F	lbf/sec-° F
Specific heat C <sub>p</sub>	J/Kg-° C	mJ/tonne-° C	ft-lbf/slug-° F	in <sup>2</sup> /sec <sup>2</sup> -° F