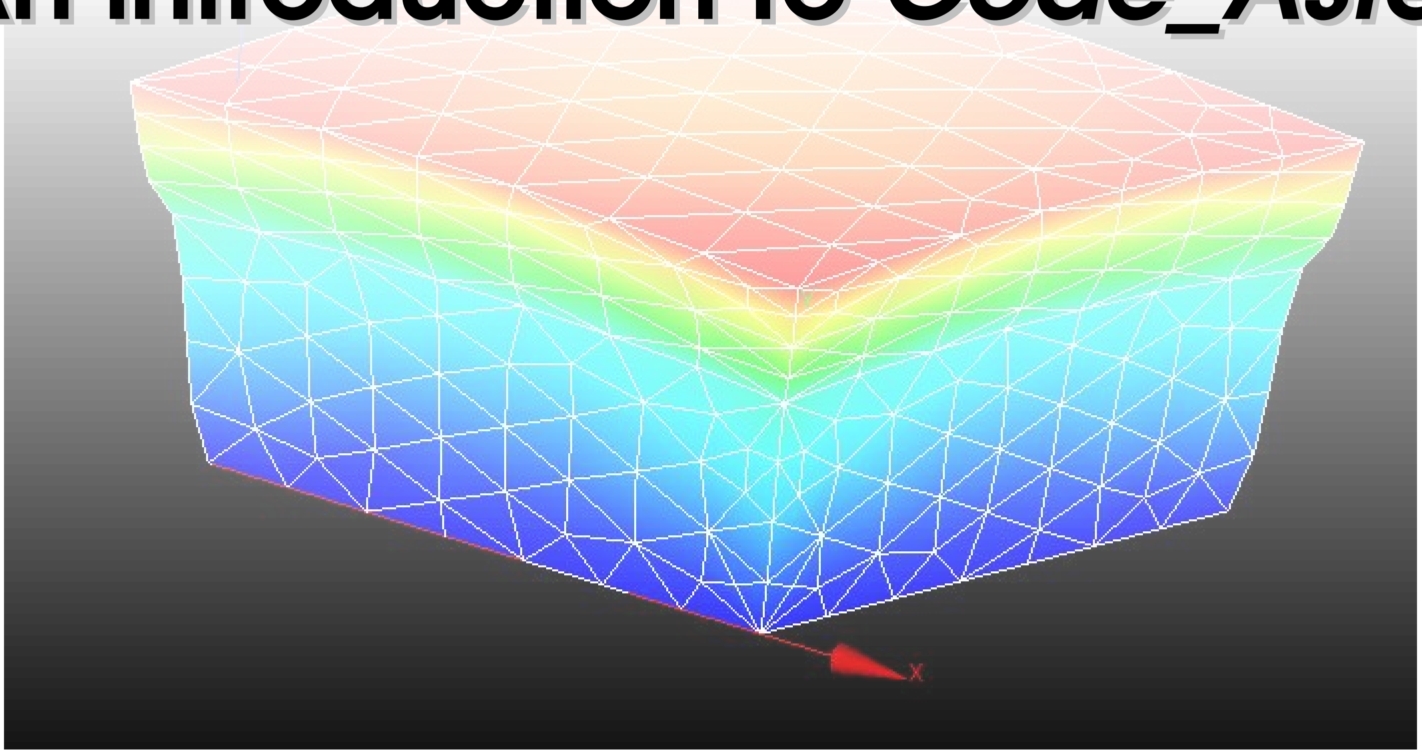
An introduction to *Code_Aster*



Goals and basic knowledge

Goals of this short course:

⇨ give you basic knowledge on how Code_Aster works

⇨ give you an overview on how you can use Code_Aster with the pre/post-processor tool Salome

⇨ give you an overview on how you can use the Salome-Meca environment.

This course is not:

⇨ a course on the finite element method

⇨ a course on how the behaviour of structures can be modelled (statics, dynamics, fatigue...)

⇨ a course on how programming in Code_Aster.

Goals and basic knowledge

Therefore you can get most out of this course if you have:

- ⇨ a theoretical base on the FE method
- ⇨ basic knowledge on how a structural FE software works
- ⇨ some experience on the use of a structural FE software.

The same FE simulation can be successfully carried out in more than one way

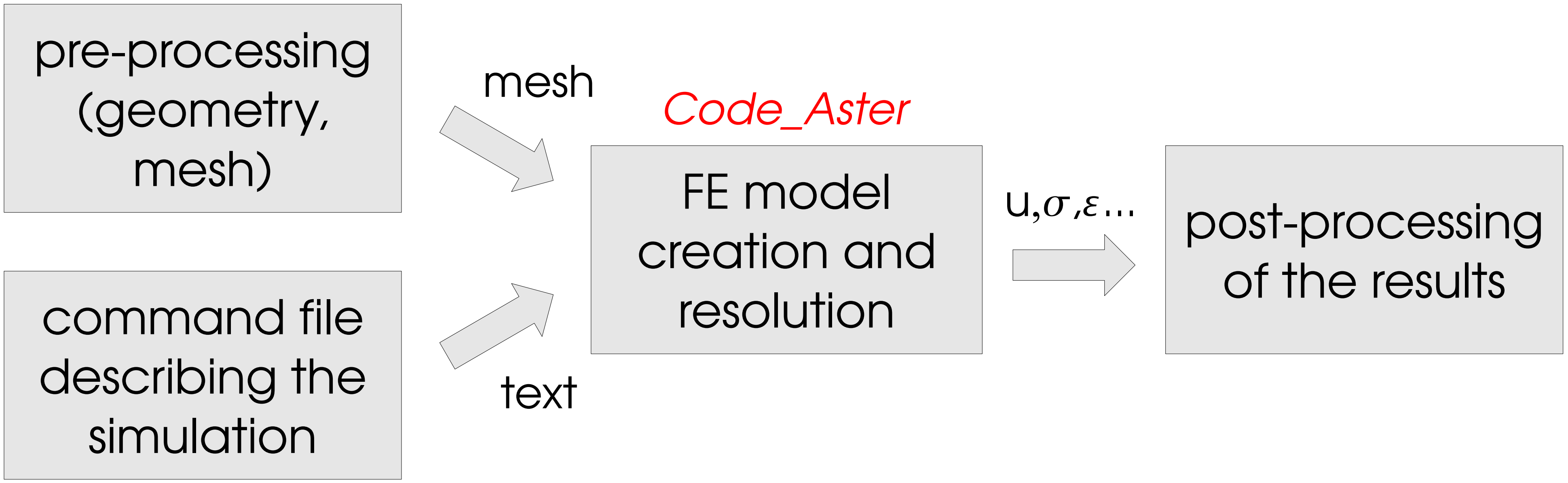
In my opinion there isn't the “best way”

What I will present today is just my personal view and what I think is the simplest way to approach *Code_Aster* in our context.

What is *Code_Aster*?

Code_Aster is a FE solver:

- ⇨ no fancy tools to create a geometry and to mesh it
- ⇨ no colourful post-processing images
- ⇨ no “click 'n' drop” interfaces.


pre-processing (geometry, mesh)
command file describing the simulation
mesh
text
Code_Aster
FE model creation and resolution
u,σ,ε...
post-processing of the results

Some pre/post-processing capabilities are indeed available in Code_Aster but I will not speak about them.

Pre/post-processing

Several tools can be used to create a mesh and to visualize the results, as far as an import/export module exist in *Code_Aster*:

- ⇨ Gmsh
- ⇨ I-DEAS
- ⇨ Gibi
- ⇨ Any tool capable of importing/exporting meshes in MED format

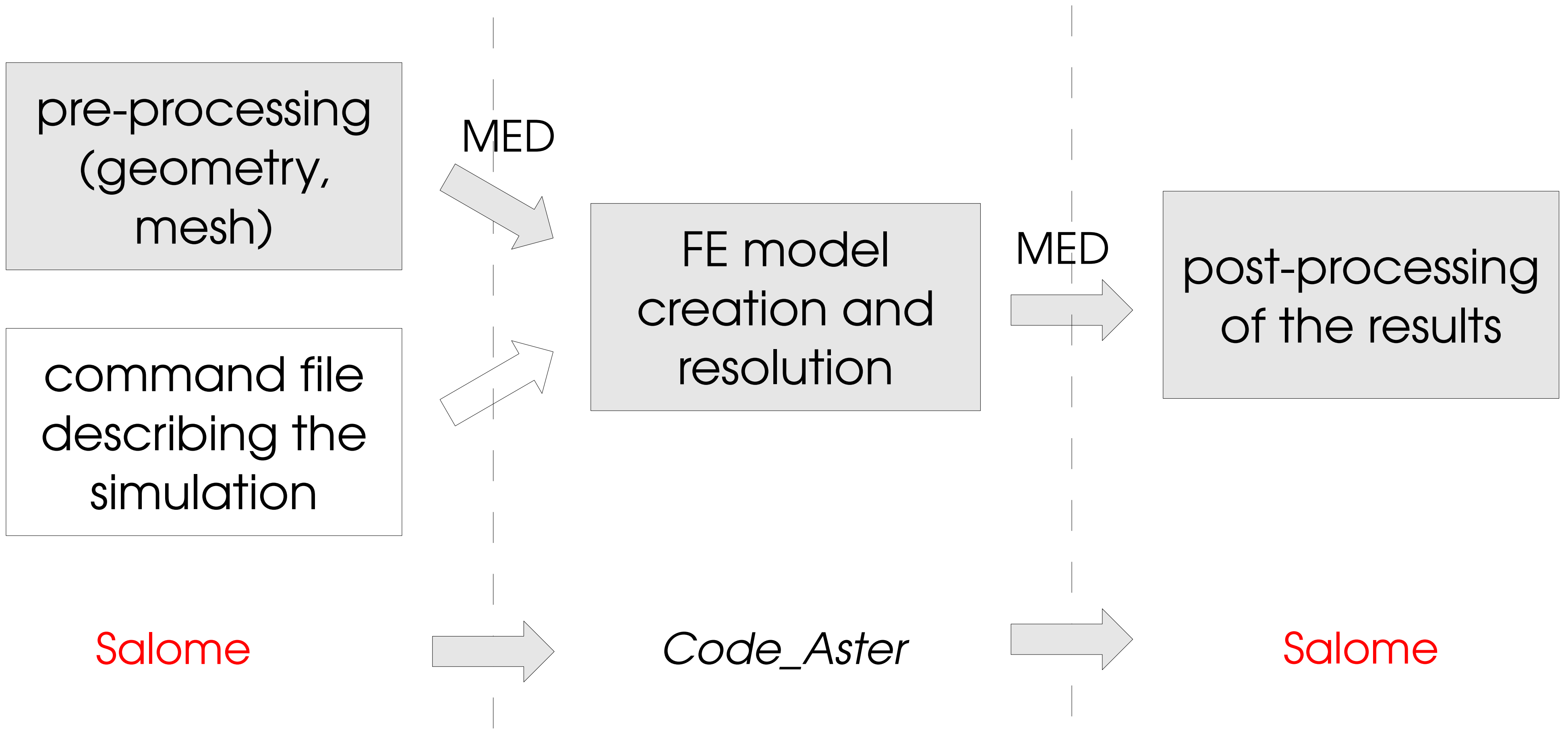
  MED is a platform independent file format for the exchange of mesh data (nodes, elements, group of nodes, group of elements) and fields defined on these data (e.g. stresses, strains, displacements, level sets...)

  A library to manage MED files is freely available.

- ⇨ Your preferred pre-processor software, but you need to write the interface layer (using e.g. MED file): Code_Aster is free software!

Pre/post-processing

Salome is a free-software pre/post-processing tool capable of exporting and importing MED files:



pre-processing (geometry, mesh)
command file describing the simulation
MED
FE model creation and resolution
MED
post-processing of the results
Salome
Code_Aster
Salome

The command file must be however created outside Salome.

MANCHESTER 1824

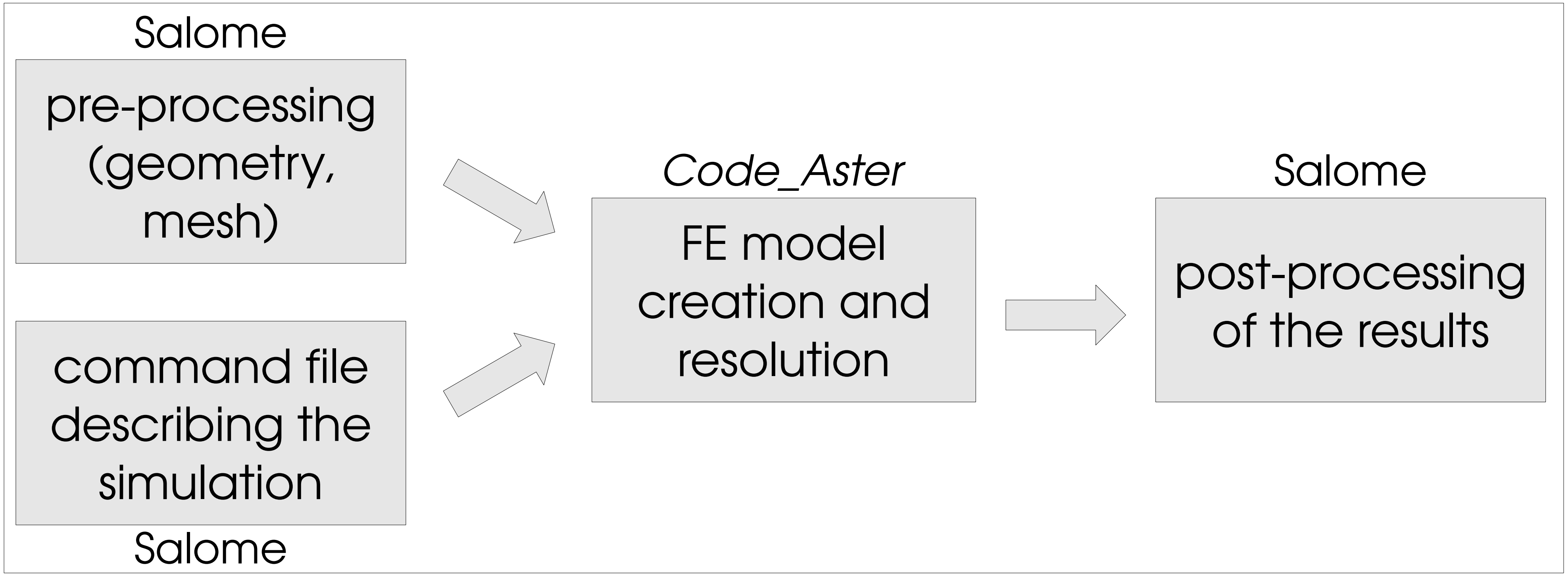
The University of Manchester

Pre/post-processing

An alternative and more comfortable solution is to use the Salome-Meca platform:

⇨ it's a “modified” version of Salome including a module used to control Code_Aster directly from Salome

⇨ the command file is generated automatically.



Salome
pre-processing (geometry, mesh)
command file describing the simulation
Salome
Code_Aster
FE model creation and resolution
Salome
post-processing of the results

Salome-Meca

Salome-Meca

Even if Code_Aster is hidden behind the graphical interface of Salome-Meca, the way in which Salome and Code_Aster interact determine how the FE model is created:

MED file

⇨ only mesh data are passed from the pre-processor to Code_Aster by means of a MED file

nodes, elements, groups of nodes, groups of elements

⇩

use of geometry and mesh capabilities of Salome as any other pre-processor software

text file

⇨ all the other ingredients of the FE model must be written inside the command file

materials, forces, pressures, displacements...

⇩

use of a wizard: you enter all these informations in a GUI

Salome-Meca

How informations entered in the wizard are linked to the mesh data included in the MED file?

⇨ you must create groups of nodes, elements, edges and faces in the mesh

you can define these groups on the geometry of the model or directly on the mesh

using geometrical groups allows the modification of the mesh without the need to redefine the groups.

⇨ in the wizard you can use these groups to apply boundary conditions, forces, pressures, materials and so on.

Salome-Meca

Once Code_Aster has solved the FE model, how are the results imported into Salome-Meca?

⇨ Code_Aster creates at least two text files:

.mess: contains the output of Code_Aster for each command. The error and warning messages are reported in this file.

.resu: contains the output of some Code_Aster results in table form.

⇩

you can read these files directly inside Salome-Meca

⇨ Code_Aster creates one MED file containing all the results

nodal displacements, stresses at nodes, equivalent stresses at nodes...

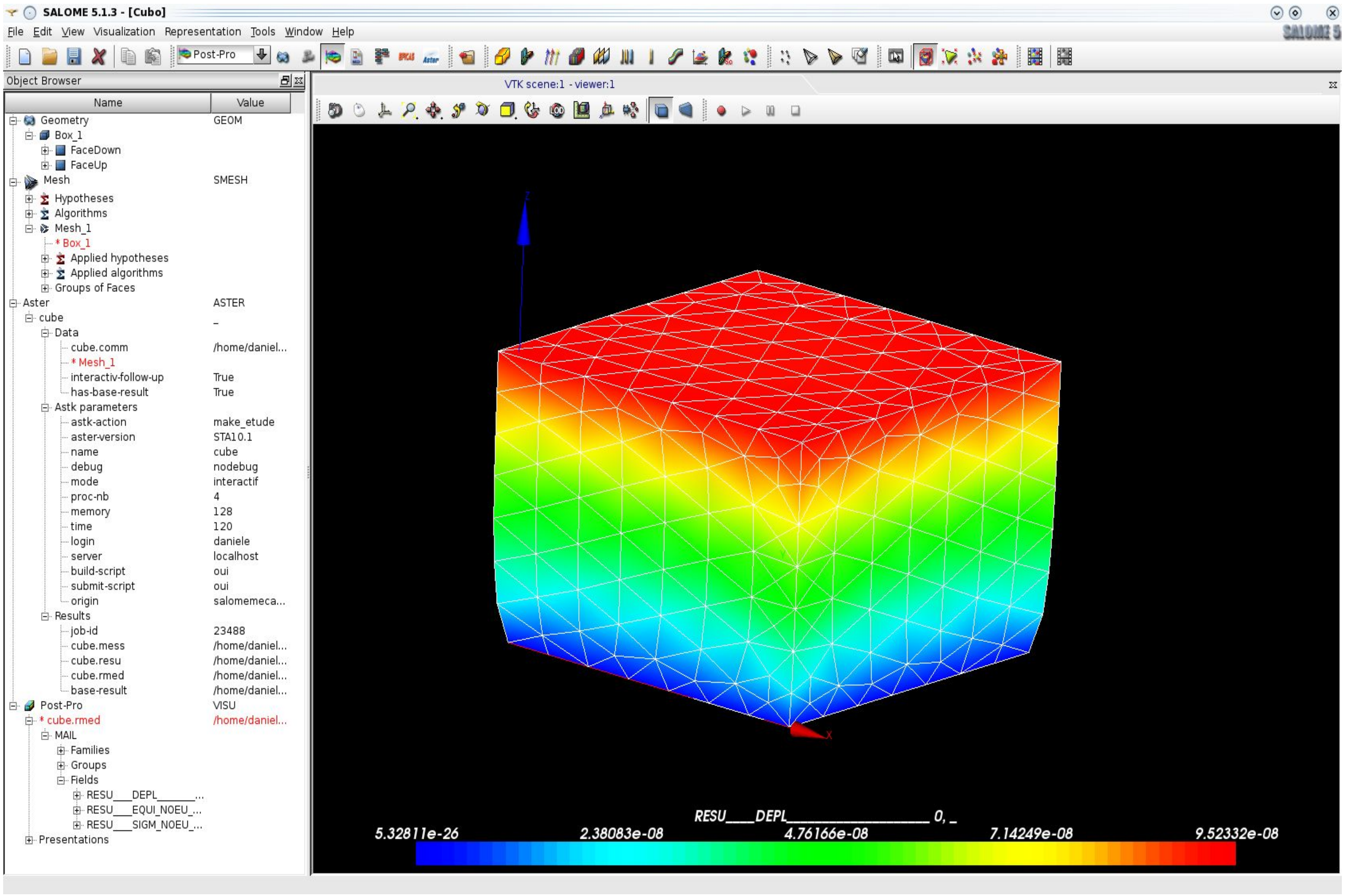
⇩

You can post-process these results using the post-processing capabilities of Salome-Meca

The University of Manchester

MANCHESTER 1824

Salome-Meca demonstration



Name	Value
Geometry	GEOM
Box_1	
FaceDown	
FaceUp	
Mesh	SMESH
Hypotheses	
Algorithms	
Mesh_1	
* Box_1	
Applied hypotheses	
Applied algorithms	
Groups of Faces	
Aster	ASTER
cube	_
Data	
cube.comm	/home/daniel...
* Mesh_1	
interactiv-follow-up	True
has-base-result	True
Astk parameters	
astk-action	make_etude
aster-version	STA10.1
name	cube
debug	nodebug
mode	interactif
proc-nb	4
memory	128
time	120
login	daniele
server	localhost
build-script	oui
submit-script	oui
origin	salomemeca...
Results	
job-id	23488
cube.mess	/home/daniel...
cube.resu	/home/daniel...
cube.rmed	/home/daniel...
base-result	/home/daniel...
Post-Pro	VISU
* cube.rmed	/home/daniel...
MAIL	
Families	
Groups	
Fields	
RESU___DEPL________...	
RESU___EQUI_NOEU_...	
RESU___SIGM_NOEU_...	
Presentations	

Salome-Meca

As you have seen, the wizard has several limitations:

⇨ you can manage only some kinds of simulations

linear elastic analyses

modal analyses

linear thermal analyses

⇨ for each supported analysis, not all the features of Code_Aster are available

e.g.: in a linear elastic analysis you can't define more materials, mechanical contact between parts, cracks...

⇨ the output is always limited to some predefined variables

e.g.: in a linear elastic analysis only nodal displacements, stresses at nodes and equivalent stresses at nodes are produced in the output file. What can you do if you need, for example, reaction forces?

Code_Aster command file

The only way to access all the features available in Code_Aster is to directly manage the command file by hand:

⇨ the command file is a text file: edit it with your preferred text editor!

This is the hardest way!

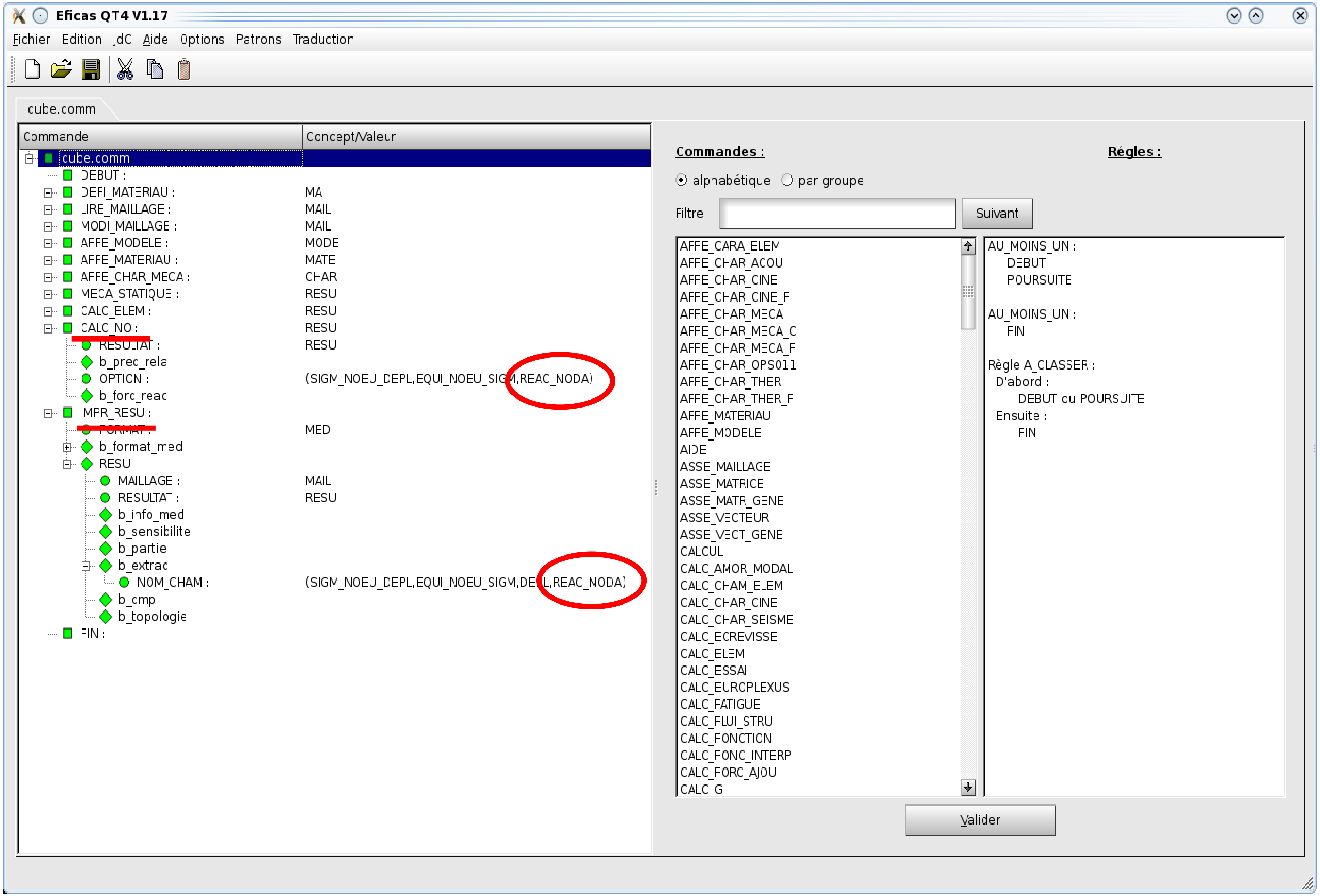
You must know Code_Aster commands, syntax, options and so on.

⇨ a graphical tool called **Eficas** can be used to simplify this task

This is the easiest way: just run Eficas from Salome-Meca.

Let's try to obtain reaction forces from our simulation... ➘

Eficas



```
Commande                    Concept/Valeur
cube.comm
  DEBUT :
  DEFI_MATERIAU :           MA
  LIRE_MAILLAGE :           MAIL
  MODI_MAILLAGE :           MAIL
  AFFE_MODELE :             MODE
  AFFE_MATERIAU :           MATE
  AFFE_CHAR_MECA :          CHAR
  MECA_STATIQUE :           RESU
  CALC_ELEM :               RESU
  CALC_NO :                 RESU
    RESULTAT :              RESU
    b_prec_rela
    OPTION :                (SIGM_NOEU_DEPL,EQUI_NOEU_SIGM,REAC_NODA)
    b_forc_reac
  IMPR_RESU :
    FORMAT :                MED
    b_format_med
    RESU :
      MAILLAGE :            MAIL
      RESULTAT :            RESU
      b_info_med
      b_sensibilite
      b_partie
      b_extrac
        NOM_CHAM :          (SIGM_NOEU_DEPL,EQUI_NOEU_SIGM,DEPL,REAC_NODA)
      b_cmp
      b_topologie
  FIN :
```

Commandes :

(• alphabétique) ○ par groupe

Filtre [ ] Suivant

```
AFFE_CARA_ELEM
AFFE_CHAR_ACOU
AFFE_CHAR_CINE
AFFE_CHAR_CINE_F
AFFE_CHAR_MECA
AFFE_CHAR_MECA_C
AFFE_CHAR_MECA_F
AFFE_CHAR_OPS011
AFFE_CHAR_THER
AFFE_CHAR_THER_F
AFFE_MATERIAU
AFFE_MODELE
AIDE
ASSE_MAILLAGE
ASSE_MATRICE
ASSE_MATR_GENE
ASSE_VECTEUR
ASSE_VECT_GENE
CALCUL
CALC_AMOR_MODAL
CALC_CHAM_ELEM
CALC_CHAR_CINE
CALC_CHAR_SEISME
CALC_ECREVISSE
CALC_ELEM
CALC_ESSAI
CALC_EUROPLEXUS
CALC_FATIGUE
CALC_FLUI_STRU
CALC_FONCTION
CALC_FONC_INTERP
CALC_FORC_AJOU
CALC_G
```

Régles :

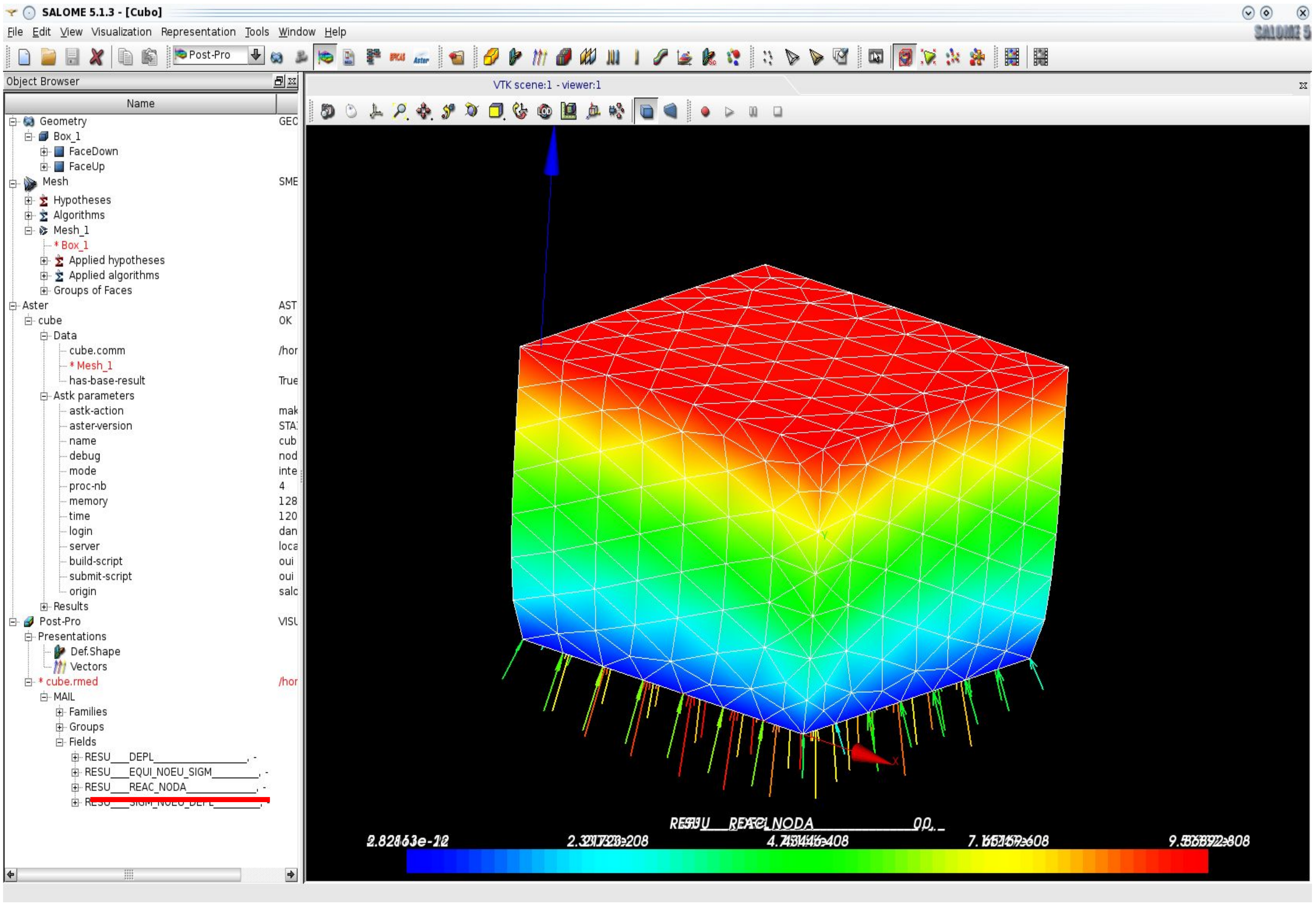
```
AU_MOINS_UN :
    DEBUT
    POURSUITE

AU_MOINS_UN :
    FIN

Règle A_CLASSER :
  D'abord :
      DEBUT ou POURSUITE
  Ensuite :
      FIN
```

Valider

Reaction forces



SALOME 5.1.3 - [Cubo]

File Edit View Visualization Representation Tools Window Help

Object Browser

VTK scene:1 - viewer:1

```
Name
Geometry                                  GEC
  Box_1
    FaceDown
    FaceUp
Mesh                                      SME
  Hypotheses
  Algorithms
  Mesh_1
    * Box_1
    Applied hypotheses
    Applied algorithms
    Groups of Faces
Aster                                     AST
  cube                                    OK
    Data
      cube.comm                           /hor
      * Mesh_1
      has-base-result                     True
    Astk parameters
      astk-action                         mak
      aster-version                       STA
      name                                cub
      debug                               nod
      mode                                inte
      proc-nb                             4
      memory                              128
      time                                120
      login                               dan
      server                              loca
      build-script                        oui
      submit-script                       oui
      origin                              salc
    Results
Post-Pro                                  VISU
  Presentations
    Def.Shape
    Vectors
  * cube.rmed                             /hor
    MAIL
      Families
      Groups
      Fields
        RESU___DEPL________________, -
        RESU___EQUI_NOEU_SIGM______, -
        RESU___REAC_NODA___________, -
        RESU___SIGM_NOEU_DEPL______, -
```

Code_Aster command file: the other way

Using Eficas you can:

⇨ create a command file which is syntactically correct

⇨ create a command file by hand without remembering all the options of each command.

However, you must edit the command file by hand using a text editor if you want to get most out of Code_Aster, e.g.:

⇨ using the latest development features

⇨ using python programming in the command file (loops, conditionals, maths calculations...).

Let's analyse the command file of our example to understand more about Code_Aster.

Command file syntax

```
DEBUT();

MA=DEFI_MATERIAU(ELAS=_F(E=210000000000.0,
                         NU=0.3,),);

MAIL=LIRE_MAILLAGE(FORMAT='MED',);

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

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

CHAR=AFFE_CHAR_MECA(MODELE=MODE,
                    DDL_IMPO=_F(GROUP_MA='Down',
                                DX=0.0,
                                DY=0.0,
                                DZ=0.0,),
                    PRES_REP=_F(GROUP_MA='Up',
                                PRES=100.0,),);

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

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

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

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

FIN();
```

⇨ the command file is a python script

you can simply ignore it!

⇨ it's composed by a sequence of commands

xyz = command(...)

⇨ each command produces a concept (everything is on the left of the "=")

xyz is a python object

⇨ each command has one or more parameters where it's possible to specify input data and options for the command

the objects produced by a previous command can be used as input parameters

MANCHESTER 1824

The University of Manchester

Command file syntax

```
DEBUT();

MA=DEFI_MATERIAU(ELAS=_F(E=210000000000.0,
                         NU=0.3,),);

MAIL=LIRE_MAILLAGE(FORMAT='MED',);

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

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

CHAR=AFFE_CHAR_MECA(MODELE=MODE,
                    DDL_IMPO=_F(GROUP_MA='Down',
                                DX=0.0,
                                DY=0.0,
                                DZ=0.0,),
                    PRES_REP=_F(GROUP_MA='Up',
                                PRES=100.0,),);

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

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

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

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

FIN();
```

⇨ DEBUT = begin

⇨ this function is used to start a new analysis

⇨ if this analysis is the continuation of a previous one, you must use the function POURSUITE() instead.

⇨ POURSUITE = continue

Command file syntax

```
DEBUT();

MA=DEFI_MATERIAU(ELAS=_F(E=210000000000.0,
                         NU=0.3,),);

MAIL=LIRE_MAILLAGE(FORMAT='MED',);

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

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

CHAR=AFFE_CHAR_MECA(MODELE=MODE,
                    DDL_IMPO=_F(GROUP_MA='Down',
                                DX=0.0,
                                DY=0.0,
                                DZ=0.0,),
                    PRES_REP=_F(GROUP_MA='Up',
                                PRES=100.0,),);

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

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

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

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

FIN();
```

```
MA=DEFI_MATERIAU(ELAS=_F(E=210000000000.0,
                         NU=0.3,),);
```

⇨ DEFI_MATERIAU = define material

⇨ You define the mechanical properties of a material

⇨ Name assigned to the material: MA

⇨ You are assigning only the elastic properties of the material by means of the keyword ELAS

Simple and composed keywords

Each command accept one or more parameters identified by a keyword:

```
concept = COMMAND(keyword_1=xxx,
                  keyword_2=yyy,
                  ...
                  keyword_n=zzz)
```

In the case of a simple keyword, its value is given directly after the "=". It can be a simple value (e.g. integer, real or another concept) or a list of simple values:

```
concept_1 = COMMAND_1(simple_keyword_1=3)

concept_2 = COMMAND_2(simple_keyword_1=1.56,
                      simple_keyword_2=(1.0,0.0, 5.6),
                      simple_keyword_3=concept_1)
```

Simple and composed keywords

In the case of a composed keyword, its value is a list of simple keywords:

```
concept_1 = COMMAND_1(simple_keyword_1=3.0)

concept_2 = COMMAND_2(simple_keyword_1=1.56,
                      composed_keyword_1=_F(
                          simple_keyword_2=3.1415,
                          simple_keyword_3=concept_1),)
```

The list of the simple keywords of a composed keyword is given inside **_F(...)**. The letter means "facteur" in French.

Command file syntax

```
DEBUT();

MA=DEFI_MATERIAU(ELAS=_F(E=210000000000.0,
                         NU=0.3,),);

MAIL=LIRE_MAILLAGE(FORMAT='MED',);

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

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

CHAR=AFFE_CHAR_MECA(MODELE=MODE,
                    DDL_IMPO=_F(GROUP_MA='Down',
                                DX=0.0,
                                DY=0.0,
                                DZ=0.0,),
                    PRES_REP=_F(GROUP_MA='Up',
                                PRES=100.0,),);

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

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

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

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

FIN();
```

```
MA=DEFI_MATERIAU(ELAS=_F(E=210000000000.0,
                         NU=0.3,),);
```

⇨ The elastic constants are given by means of the composed keyword ELAS

⇨ The Young's module (E) and Poisson ratio (NU) are given inside the composed keyword by means of two simple keywords

⇨ other behaviours are given by means of other composed keywords

e.g.: ECRO_LINE for a linear hardening material
THERM for the thermal properties
...

Something about the concepts

```
MA=DEFI_MATERIAU(ELAS=_F(E=210000000000.0,
                         NU=0.3,),);
```

The material definition will be contained in a concept named MA:

⇨ The maximum length of the concept name is 8 characters

⇨ If the resulting concept exists before executing the command, an error is issued and the analysis is aborted.

This concept is a python object. This means that it has a **type associated:**

⇨ The resulting concept MA is automatically created by the command, which determines also its type

⇨ The concept MA can be used as the value of a simple keywords only if its type is coincident with the type requested by the keyword.

MANCHESTER 1824

The University of Manchester

Something about the concepts

For example, I want to define two elastic materials (a generic steel and a generic aluminium).

These two materials have different values of the Young's module but the same (more or less) value of the Poisson ratio.

In order to force this equality, I declare them in the following way:

```
steel = DEFI_MATERIAU(ELAS=_F(E=2.06E11,
                              NU=0.3,),)

alu = DEFI_MATERIAU(ELAS=_F(E=0.76E11,
                            NU=steel,),)
```

This is not correct because ***steel*** is a concept of type ***material*** whilst the keyword NU requires a concept of type ***real***!

Something about the concepts

The correct way to do that would be the use of a variable (don't forget we are writing a python script):

```
poisson = 0.3

steel = DEFI_MATERIAU(ELAS=_F(E=2.06E11,
                              NU=poisson,),)

alu = DEFI_MATERIAU(ELAS=_F(E=0.76E11,
                            NU=poisson,),)
```

The University of Manchester

MANCHESTER 1824

Command file syntax

```
DEBUT();

MA=DEFI_MATERIAU(ELAS=_F(E=210000000000.0,
                         NU=0.3,),);

MAIL=LIRE_MAILLAGE(FORMAT='MED',);

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

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

CHAR=AFFE_CHAR_MECA(MODELE=MODE,
                    DDL_IMPO=_F(GROUP_MA='Down',
                                DX=0.0,
                                DY=0.0,
                                DZ=0.0,),
                    PRES_REP=_F(GROUP_MA='Up',
                                PRES=100.0,),);

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

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

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

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

FIN();
```

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

- ⇨ LIRE_MAILLAGE = read mesh
- ⇨ **In Code_Aster the mesh is simply a list of nodes (number and coordinates) and elements (their definition in terms of geometrical shape and defining nodes)**
- ⇨ Everything will be contained in the concept MAIL
- ⇨ The mesh is contained in a MED file (FORMAT='MED')
- ⇨ **Yes, really simple. Just a question: of which file are you speaking about?**

Selecting input and output files

As in the case of LIRE_MAILLAGE, there are other commands working on files (in input or output).

The name of the file read or written by a command is not specified in the command file but outside it:

- ⇨ this abstraction facilitates the creation of scripts, as in the case of parametric studies where the same analysis is run using different meshes
- ⇨ it's really practical in all the cases in which one or more input/output files change (different names or paths, different meshes...): nothing changes in the command file!

A graphical tool, called **ASTK**, can be used to list all the files needed by the simulation.

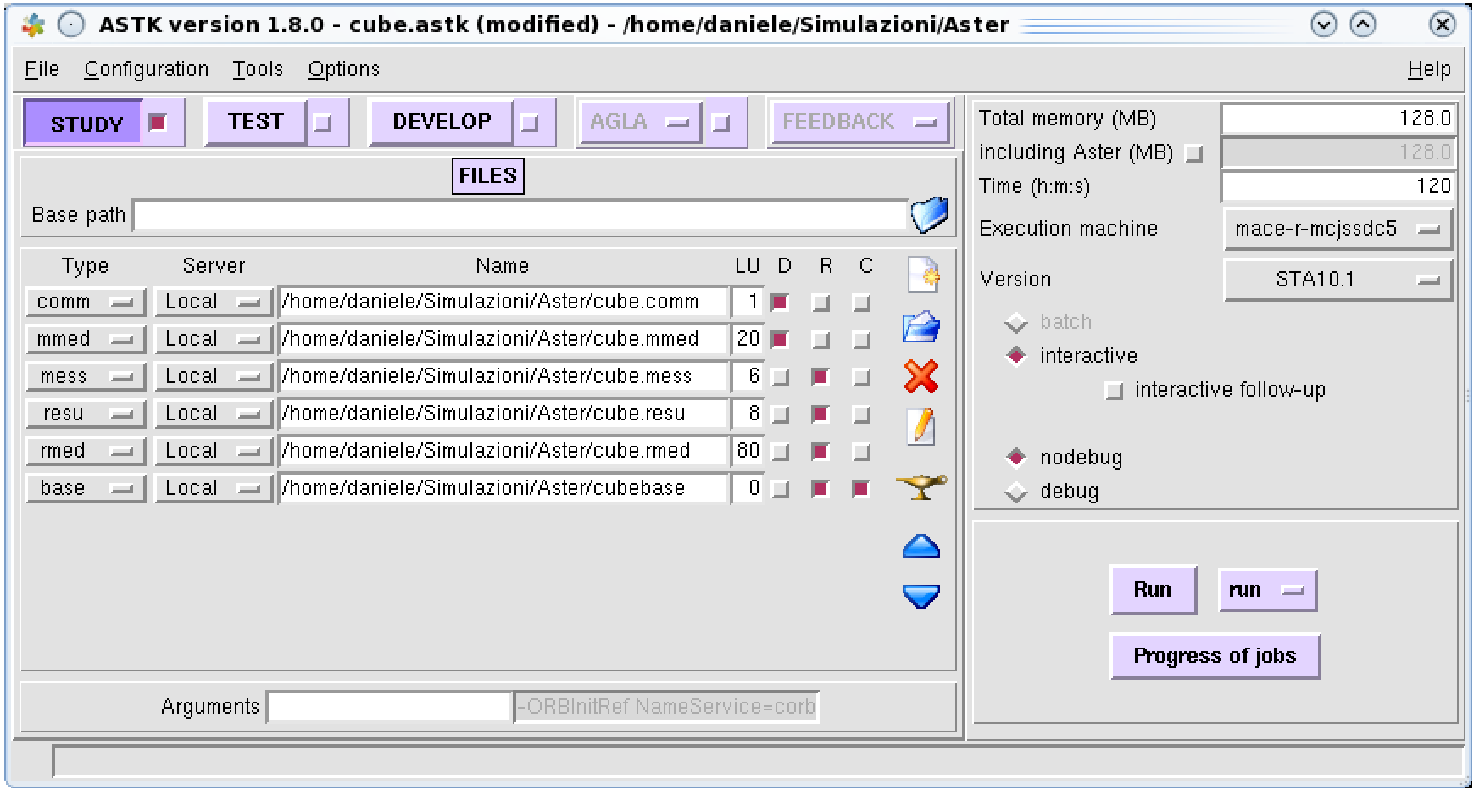
- ⇨ if you are so inclined, you can also use the Unix command line (I'm not going to talk about it today)

Selecting input and output files

If you use Salome-Meca, the input/output files are automatically linked to the command file and you have nothing to do

This is true even in the case in which you use Salome-Meca to create a basic command file and then you use Eficas to modify it

You can run ASTK directly from Salome-Meca. Let's check our example:


ASTK version 1.8.0 - cube.astk (modified) - /home/daniele/Simulazioni/Aster
File Configuration Tools Options Help
STUDY TEST DEVELOP AGLA FEEDBACK
FILES
Run run
Progress of jobs

Base path

Type	Server	Name	LU	D	R	C
comm	Local	/home/daniele/Simulazioni/Aster/cube.comm	1	■	□	□
mmed	Local	/home/daniele/Simulazioni/Aster/cube.mmed	20	■	□	□
mess	Local	/home/daniele/Simulazioni/Aster/cube.mess	6	□	■	□
resu	Local	/home/daniele/Simulazioni/Aster/cube.resu	8	□	■	□
rmed	Local	/home/daniele/Simulazioni/Aster/cube.rmed	80	□	■	□
base	Local	/home/daniele/Simulazioni/Aster/cubebase	0	□	■	■

Arguments -ORBInitRef NameService=corb

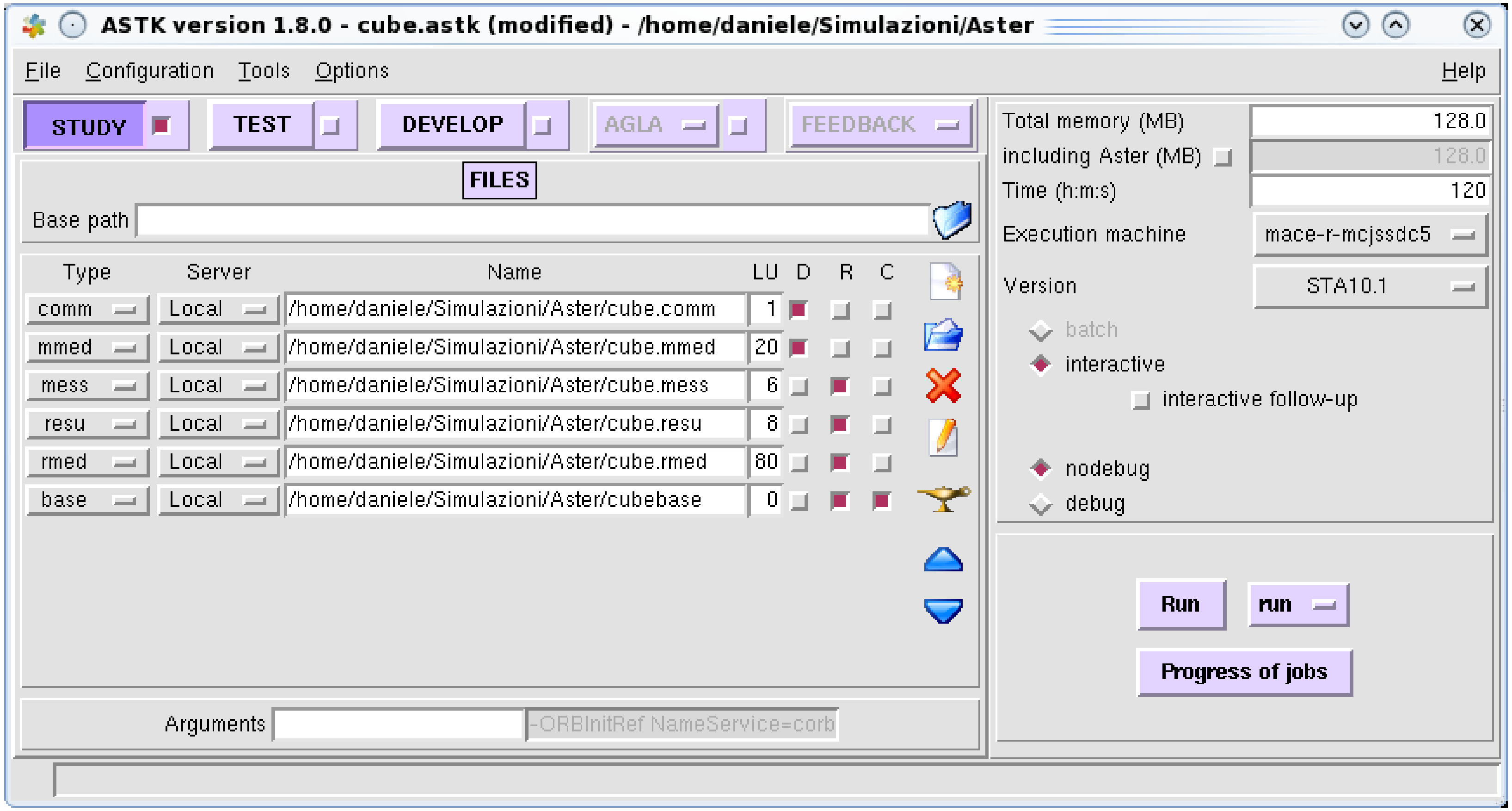
Total memory (MB)	128.0
including Aster (MB) □	128.0
Time (h:m:s)	120
Execution machine	mace-r-mcjssdc5
Version	STA10.1

batch [✓ interactive]
□ interactive follow-up

[✓ nodebug] debug

Selecting input and output files

Each file has a ***type*** associated:


ASTK version 1.8.0 - cube.astk (modified) - /home/daniele/Simulazioni/Aster
File Configuration Tools Options Help
STUDY TEST DEVELOP AGLA FEEDBACK
FILES
Run run
Progress of jobs

Base path:

Type	Server	Name	LU	D	R	C
comm	Local	/home/daniele/Simulazioni/Aster/cube.comm	1	■	□	□
mmed	Local	/home/daniele/Simulazioni/Aster/cube.mmed	20	■	□	□
mess	Local	/home/daniele/Simulazioni/Aster/cube.mess	6	□	■	□
resu	Local	/home/daniele/Simulazioni/Aster/cube.resu	8	□	■	□
rmed	Local	/home/daniele/Simulazioni/Aster/cube.rmed	80	□	■	□
base	Local	/home/daniele/Simulazioni/Aster/cubebase	0	□	■	■

Arguments: -ORBInitRef NameService=corb

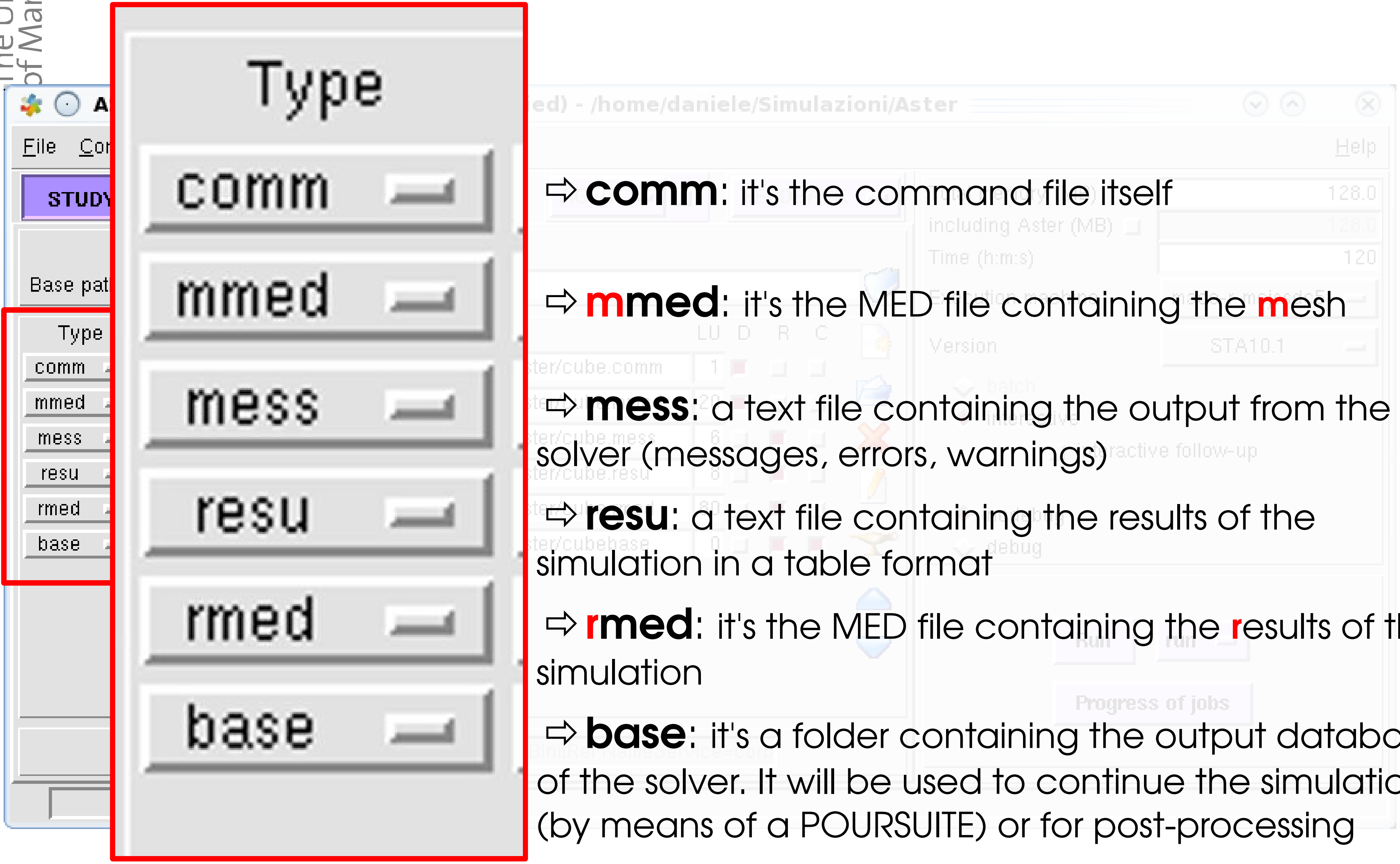
Total memory (MB): 128.0
including Aster (MB): 128.0
Time (h:m:s): 120
Execution machine: mace-r-mcjssdc5
Version: STA10.1
batch [✓ interactive]
interactive follow-up
[✓ nodebug] debug

Each ***type*** has a ***logical unit (LU)*** number associated

Each ***command*** access (read/write) a file by means of ***its logical unit number*** and not its name.

Selecting input and output files

Here are the basic files managed by each simulation:



⇨ **comm**: it's the command file itself

⇨ **mmed**: it's the MED file containing the **m**esh

⇨ **mess**: a text file containing the output from the solver (messages, errors, warnings)

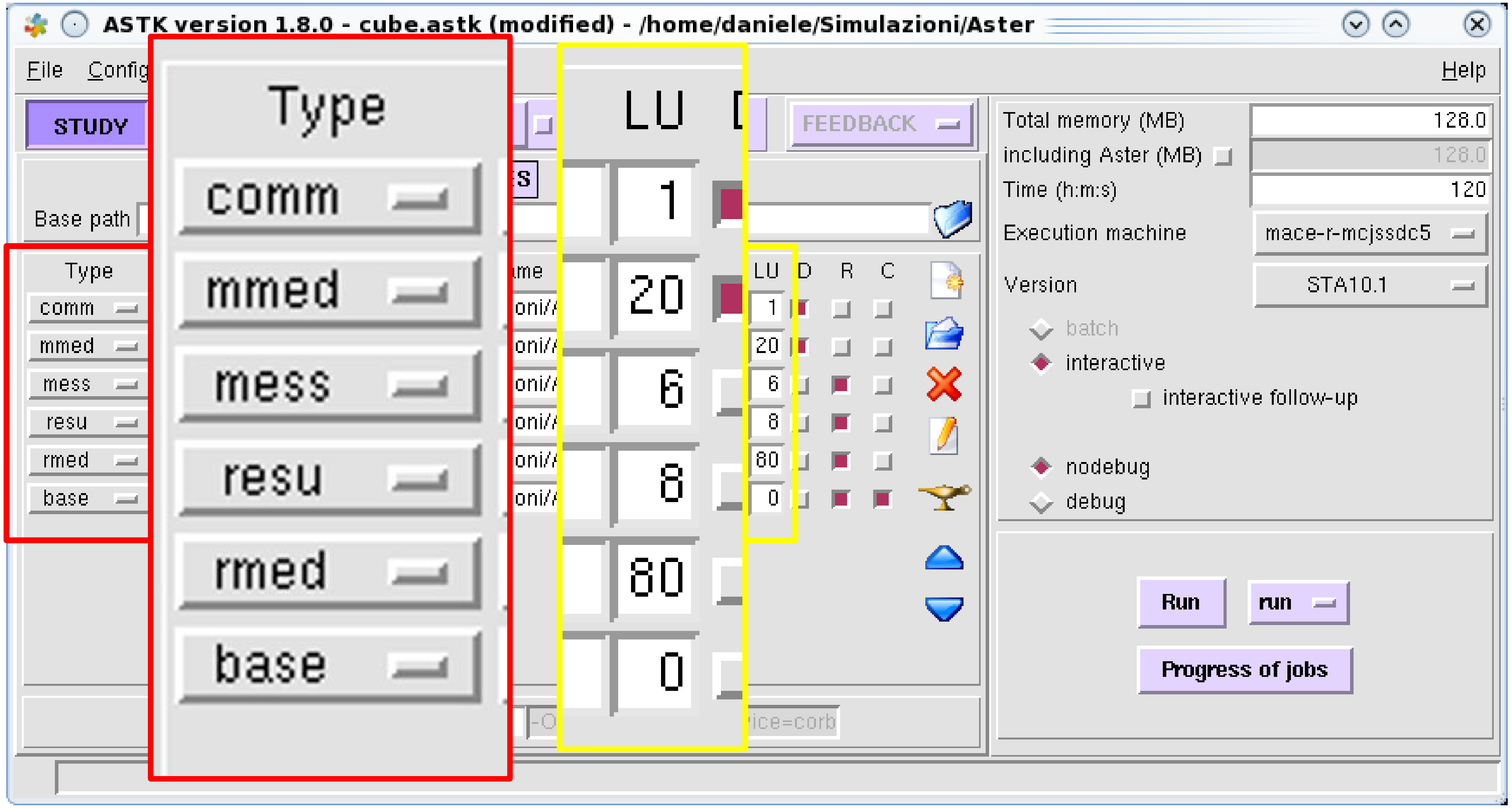
⇨ **resu**: a text file containing the results of the simulation in a table format

⇨ **rmed**: it's the MED file containing the **r**esults of the simulation

⇨ **base**: it's a folder containing the output database of the solver. It will be used to continue the simulation (by means of a POURSUITE) or for post-processing

Selecting input and output files

Each of this file type has its logical unit number assigned by default. The user can change it, although he must pay attention, as we will see soon.



Type	LU
comm	1
mmed	20
mess	6
resu	8
rmed	80
base	0

Total memory (MB)	128.0
including Aster (MB)	128.0
Time (h:m:s)	120
Execution machine	mace-r-mcjssdc5
Version	STA10.1

batch [✓ interactive] interactive follow-up
[✓ nodebug] debug

Selecting input and output files

```
DEBUT();

MA=DEFI_MATERIAU(ELAS=_F(E=210000000000.0,
                         NU=0.3,),);

MAIL=LIRE_MAILLAGE(FORMAT='MED',);

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

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

CHAR=AFFE_CHAR_MECA(MODELE=MODE,
                    DDL_IMPO=_F(GROUP_MA='Down',
                                DX=0.0,
                                DY=0.0,
                                DZ=0.0,),
                    PRES_REP=_F(GROUP_MA='Up',
                                PRES=100.0,),);

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

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

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

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

FIN();
```

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

⇨ if the LU number is not given as a parameter of the command, the default one is used

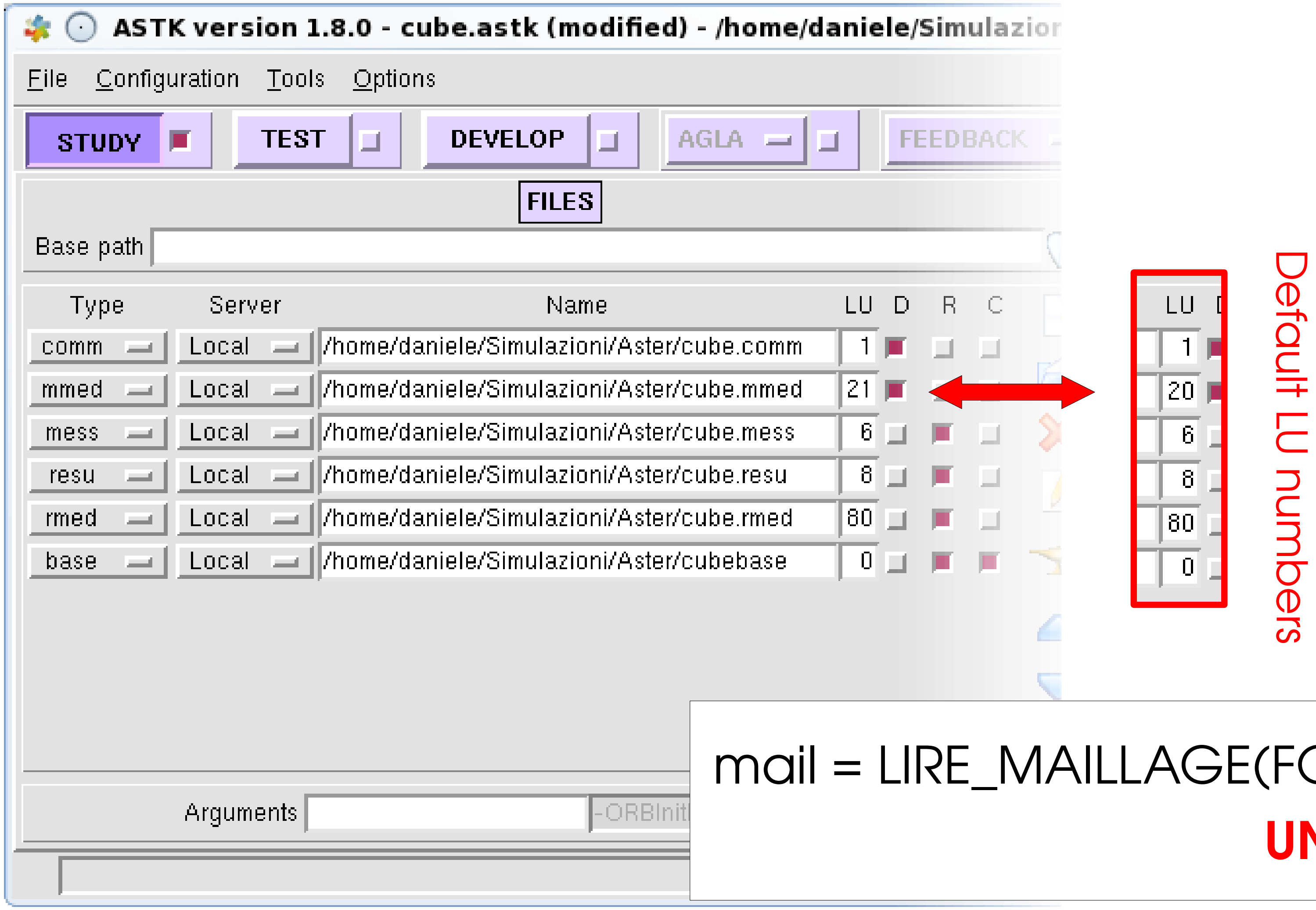
⇨ this is the case for our example: the LIRE_MAILLAGE command will read the file associated to the default LU number for a MED mesh file (***mmed***)

⇨ you don't need to know which is this default LU number if you haven't changed the LU number by hand in ASTK.

The University of Manchester

Selecting input and output files

On the contrary, if you have changed the default LU number in ASTK, you must pass the new one to the command by means of the simple keyword ***UNITE***


ASTK version 1.8.0 - cube.astk (modified) - /home/daniele/Simulazior
File Configuration Tools Options
STUDY
TEST
DEVELOP
AGLA
FEEDBACK
FILES
Base path
Arguments
-ORBInit
Default LU numbers

Type	Server	Name	LU	D	R	C
comm	Local	/home/daniele/Simulazioni/Aster/cube.comm	1	■	□	□
mmed	Local	/home/daniele/Simulazioni/Aster/cube.mmed	21	■		
mess	Local	/home/daniele/Simulazioni/Aster/cube.mess	6	□	■	□
resu	Local	/home/daniele/Simulazioni/Aster/cube.resu	8	□	■	□
rmed	Local	/home/daniele/Simulazioni/Aster/cube.rmed	80	□	■	□
base	Local	/home/daniele/Simulazioni/Aster/cubebase	0	□	■	■

LU
1
20
6
8
80
0

```
mail = LIRE_MAILLAGE(FORMAT='MED',
                     UNITE=21,),)
```

Selecting input and output files

In ASTK you can also specify some important properties of each listed file:

⇨ input or output file?

it will be created during or at the end of the analysis

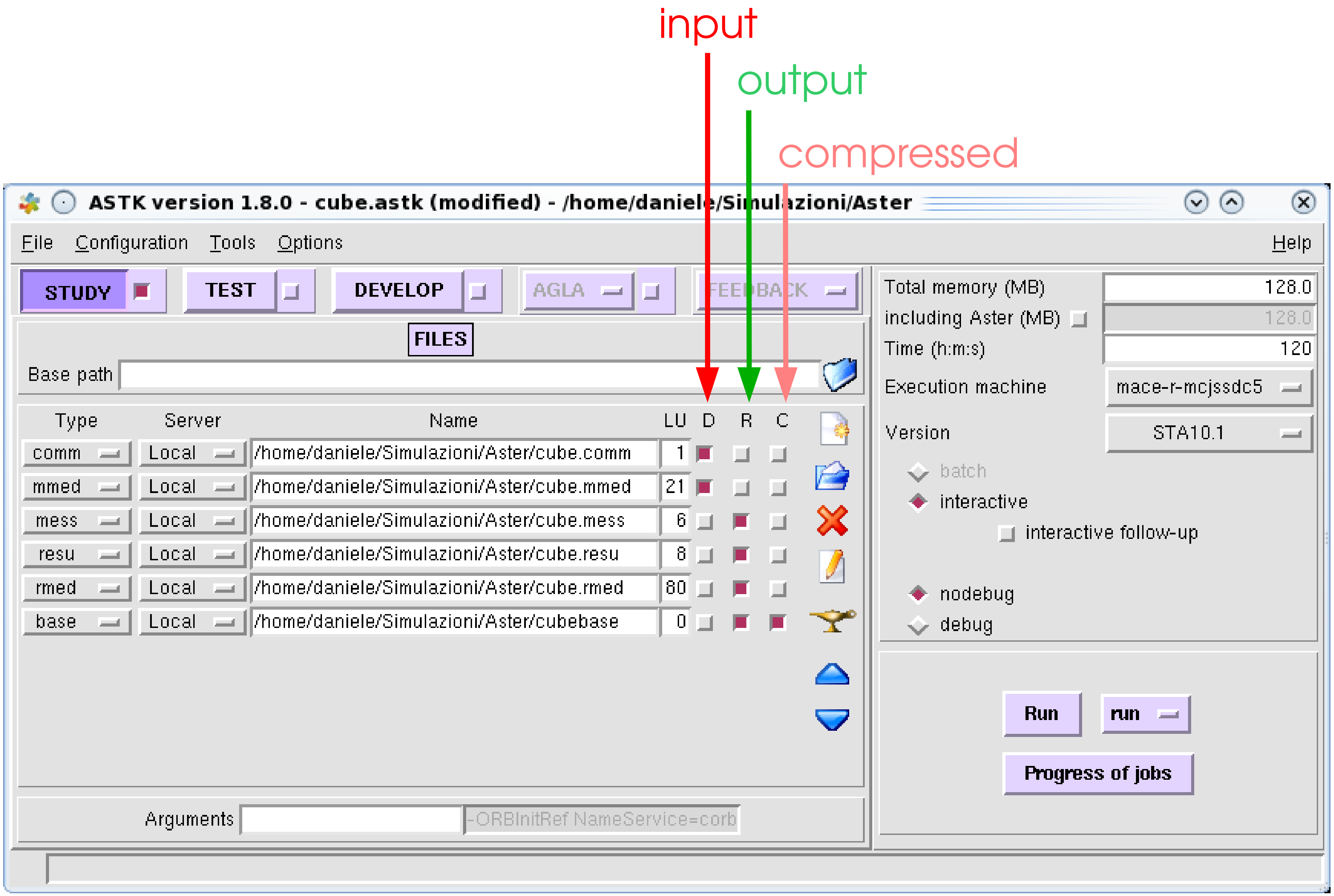
it is provided by the user at the beginning of the analysis

⇨ is the file compressed?

The file can be compressed using gzip in order to save disk space and transfer time (e.g. from the cluster to the local computer)

For the input file, the user must compress the file using gzip. In the case of output files, Code_Aster will compress them automatically at the end of the simulation.

Selecting input and output files


input
output
compressed
ASTK version 1.8.0 - cube.astk (modified) - /home/daniele/Simulazioni/Aster
File Configuration Tools Options Help
STUDY TEST DEVELOP AGLA FEEDBACK
FILES
Base path
Run
run
Progress of jobs

Type	Server	Name	LU	D	R	C
comm	Local	/home/daniele/Simulazioni/Aster/cube.comm	1	■	□	□
mmed	Local	/home/daniele/Simulazioni/Aster/cube.mmed	21	■	□	□
mess	Local	/home/daniele/Simulazioni/Aster/cube.mess	6	□	■	□
resu	Local	/home/daniele/Simulazioni/Aster/cube.resu	8	□	■	□
rmed	Local	/home/daniele/Simulazioni/Aster/cube.rmed	80	□	■	□
base	Local	/home/daniele/Simulazioni/Aster/cubebase	0	□	■	■

Total memory (MB): 128.0
including Aster (MB): 128.0
Time (h:m:s): 120
Execution machine: mace-r-mcjssdc5
Version: STA10.1

batch [✓ interactive]
interactive follow-up

[✓ nodebug] debug

Arguments: -ORBInitRef NameService=corb

Command file syntax

```
DEBUT();

MA=DEFI_MATERIAU(ELAS=_F(E=210000000000.0,
                         NU=0.3,),);

MAIL=LIRE_MAILLAGE(FORMAT='MED',);

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

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

CHAR=AFFE_CHAR_MECA(MODELE=MODE,
                    DDL_IMPO=_F(GROUP_MA='Down',
                                DX=0.0,
                                DY=0.0,
                                DZ=0.0,),
                    PRES_REP=_F(GROUP_MA='Up',
                                PRES=100.0,),);

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

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

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

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

FIN();
```

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

⇨ AFFE_MODELE = assign model

⇨ This command allow to assign a particular **physical model** and **element type** to a mesh. A FE model is created.

mechanical, thermal, acoustic

axial-symmetric, plane stress/strain, shell, beam, plate...

⇨ First the mesh is selected by means of the MAILLAGE(=mesh) keyword

MAIL is an existing concept!

Command file syntax

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

The composed keyword AFFE allows to specify the physical phenomenon and element type to be assigned to a part or the whole mesh:

⇨ **where?** TOUT='OUI' means WHOLE_MESH='YES'

⇨ **which phenomenon?** PHENOMENE='MECANIQUE' specifies that we want to model a mechanical phenomenon (not a thermal or acoustic one)

⇨ **which type of element?** MODELISATION='3D' specifies that we want to use 3D solid elements (and not, for example, 3D beam elements).

Command file syntax

The group of keywords inside _F(...) can be repeated to define all the modelizations required in the model

For example, in the case in which both 3D solid elements and 3D beams are present in the mesh, the following command can be used to assign the properties to the elements:

```
mode = AFFE_MODELE(MAILLAGE=MAIL,
                   AFFE=(_F(GROUP_MA='solid',
                            PHENOMENE='MECANIQUE',
                            MODELISATION='3D',),

                         _F(GROUP_MA='beam',
                            PHENOMENE='MECANIQUE',
                            MODELISATION='POU_D_E',),),)
```

solid and *beam* are two groups of elements (GROUP_MA) defined inside the mesh MAIL (by means of Salome)

Command file syntax

```
DEBUT();

MA=DEFI_MATERIAU(ELAS=_F(E=210000000000.0,
                         NU=0.3,),);

MAIL=LIRE_MAILLAGE(FORMAT='MED',);

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

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

CHAR=AFFE_CHAR_MECA(MODELE=MODE,
                    DDL_IMPO=_F(GROUP_MA='Down',
                                DX=0.0,
                                DY=0.0,
                                DZ=0.0,),
                    PRES_REP=_F(GROUP_MA='Up',
                                PRES=100.0,),);

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

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

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

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

FIN();
```

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

- ⇨ AFFE_MATERIAU = assign material
- ⇨ This command allow to assign a material to the elements of a mesh
- ⇨ The syntax is similar to what we have already discussed
- ⇨ The material must be defined before using the command by means of DEFI_MATERIAU
- ⇨ The concept created by the command (MATE) is a **"material field"**, an unusual entity for FE solvers!

Overloading

How can I assign more then one material to a FE mesh?

We have already defined two materials:

```
poisson = 0.3
steel = DEFI_MATERIAU(ELAS=_F(E=2.06E11,
                              NU=poisson,),)
alu = DEFI_MATERIAU(ELAS=_F(E=0.76E11,
                              NU=poisson,),)
```

Let's suppose that two groups of elements are defined in the mesh, each one containing the element of the same material:

- ⇨ ***ELsteel*** contains the elements for steel
- ⇨ ***ELalu*** contains the elements for aluminium

The union of the two groups covers all the elements of the mesh, that is each element of the mesh belongs to one group.

Overloading

The assignment of the materials can be done in the following way:

```
 mate = AFFE_MATERIAU(MAILLAGE=MAIL,
                                    AFFE=(_F(GROUP_MA='ELsteel',
                                             MATER=steel,
                                          _F(GROUP_MA='ELalu',
                                             MATER=alu,),),)
```

Another solution would be the use of the **overloading** rule:

⇨ if more than one property is assigned to the same entity, only the last assignment is considered

⇨ in other words, if each assignment overwrites the previous one.

Overloading

The overloaded version of the assignment is the following:

```
mate = AFFE_MATERIAU(MAILLAGE=MAIL,
                                         AFFE=(_F(TOUT='OUI',
                                                  MATER=steel,
                                              _F(GROUP_MA='ELalu',
                                                  MATER=alu,),),)
```

Here's what happens:

⇨ first, the material *steel* is assigned to **all the elements** of the mesh MAIL

⇨ then, the material *alu* is assigned **only** to the elements in the group **ELalu**

⇨ the material *steel* assigned previously to these elements (Elalu) is overwritten by the second assignment.

Overloading

The use of the overloading rule can be really effective in many situations.

Even in this really simple example, the advantages would be:

⇨ only one group of elements (ELalu) must be defined and maintained in the mesh

⇨ a material is assigned to all the elements of the mesh. There's no risk that one or more elements have no material assigned because they are outside the two defined groups (ELsteel and ELalu) consequently to an error in the group creation.

```
mate = AFFE_MATERIAU(MAILLAGE=MAIL,
                         AFFE=(_F(GROUP_MA='ELsteel',
                                  MATER=steel,
                               _F(GROUP_MA='ELalu',
                                  MATER=alu,),),)
```

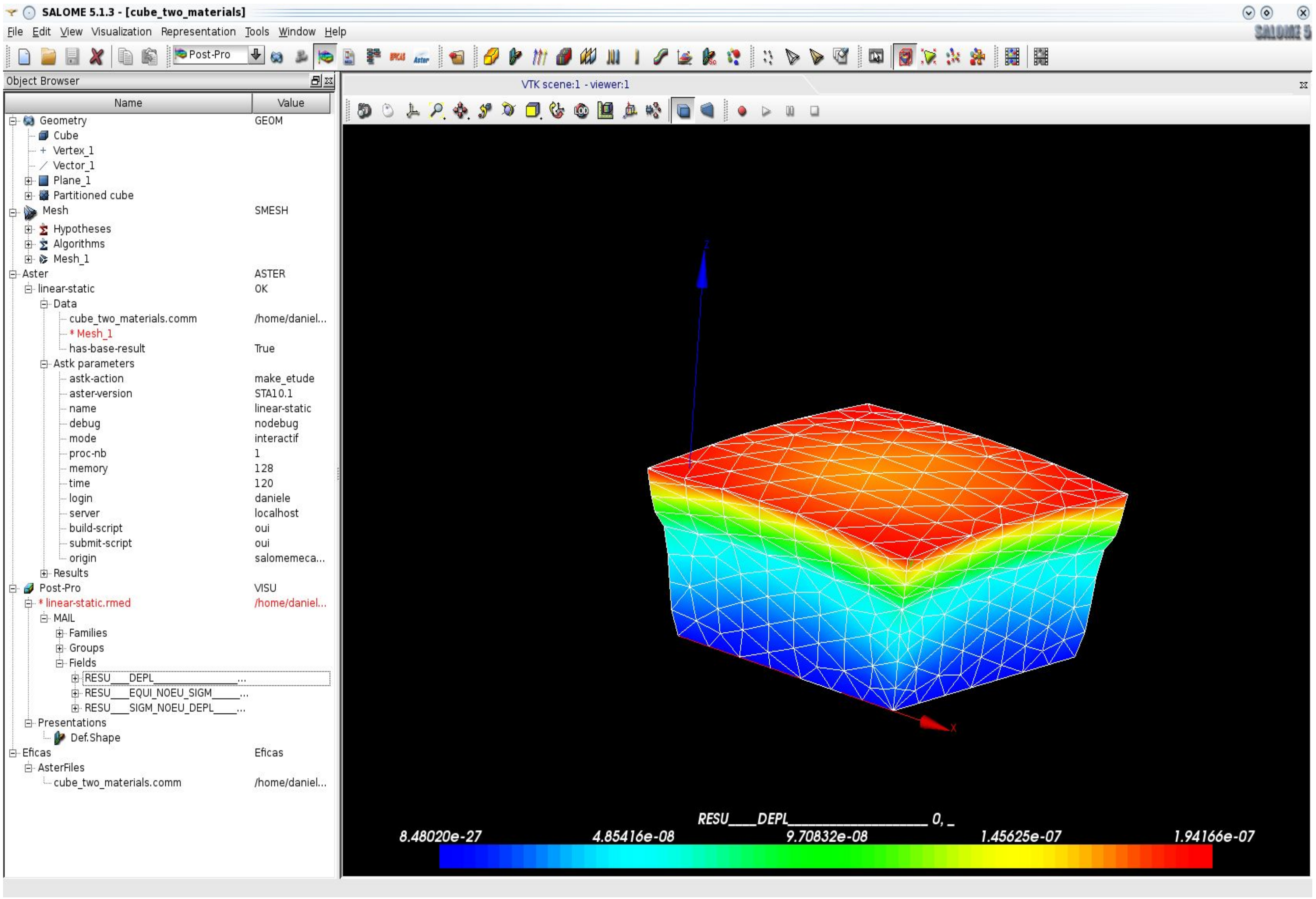
Salome-Meca+Eficas example

Let's try to modify our example in order to define a two materials cube.

We will use Salome-Meca:

- ⇨ creation of a partitioned cube + ELalu group
- ⇨ creation of a one material FE model by means of the wizard
- ⇨ modification of the command file by means of Eficas: definition of the second material and proper material assignments to the mesh.

Salome-Meca+Eficas example


SALOME 5.1.3 - [cube_two_materials]
File Edit View Visualization Representation Tools Window Help
Post-Pro
Object Browser
VTK scene:1 - viewer:1
RESU____DEPL_____________________ 0, _
8.48020e-27
4.85416e-08
9.70832e-08
1.45625e-07
1.94166e-07

Name	Value
Geometry	GEOM
Cube	
Vertex_1	
Vector_1	
Plane_1	
Partitioned cube	
Mesh	SMESH
Hypotheses	
Algorithms	
Mesh_1	
Aster	ASTER
linear-static	OK
Data	
cube_two_materials.comm	/home/daniel...
* Mesh_1	
has-base-result	True
Astk parameters	
astk-action	make_etude
aster-version	STA10.1
name	linear-static
debug	nodebug
mode	interactif
proc-nb	1
memory	128
time	120
login	daniele
server	localhost
build-script	oui
submit-script	oui
origin	salomemeca...
Results	
Post-Pro	VISU
* linear-static.rmed	/home/daniel...
MAIL	
Families	
Groups	
Fields	
RESU____DEPL_____________________...	
RESU____EQUI_NOEU_SIGM_________...	
RESU____SIGM_NOEU_DEPL_________...	
Presentations	
Def.Shape	
Eficas	Eficas
AsterFiles	
cube_two_materials.comm	/home/daniel...

Command file syntax

```
DEBUT();

MA=DEFI_MATERIAU(ELAS=_F(E=210000000000.0,
                         NU=0.3,),);

MAIL=LIRE_MAILLAGE(FORMAT='MED',);

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

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

CHAR=AFFE_CHAR_MECA(MODELE=MODE,
                    DDL_IMPO=_F(GROUP_MA='Down',
                                DX=0.0,
                                DY=0.0,
                                DZ=0.0,),
                    PRES_REP=_F(GROUP_MA='Up',
                                PRES=100.0,),);

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

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

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

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

FIN();
```

```
CHAR=AFFE_CHAR_MECA(MODELE=MODE,
                    DDL_IMPO=_F(GROUP_MA='Down',
                                DX=0.0,
                                DY=0.0,
                                DZ=0.0,),
                    PRES_REP=_F(GROUP_MA='Up',
                                PRES=100.0,),);
```

⇨ AFFE_CHAR_MECA = assign mechanical loads

⇨ This command allow to assign mechanical loads, boundary conditions and cinematic constraints to a **FE model**

⇨ DDL_IMPO = assign a value to one ore more degrees of freedom of a node

⇨ PRES_REP = apply a pressure to a 2D/3D domain

Command file syntax

```
DEBUT();

MA=DEFI_MATERIAU(ELAS=_F(E=210000000000.0,
                         NU=0.3,),);

MAIL=LIRE_MAILLAGE(FORMAT='MED',);

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

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

CHAR=AFFE_CHAR_MECA(MODELE=MODE,
                    DDL_IMPO=_F(GROUP_MA='Down',
                                DX=0.0,
                                DY=0.0,
                                DZ=0.0,),
                    PRES_REP=_F(GROUP_MA='Up',
                                PRES=100.0,),);

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

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

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

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

FIN();
```

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

⇨ MECA_STATIQUE = assembly the FE discrete set of equations and solve it for a **linear static mechanical** model

⇨ In opposition to many other FE solvers, at this point the user must specify the model, the materials and the loads to be applied and used in the assembly.

⇨ This means that not all the defined materials, loads, boundary conditions and **models** (yes, I wrote models) will be used during the assembly of the FE model to be solved!

Non-linearities and dynamic/thermal simulations

The command MECA_STATIQUE can't be used if non-linearities (geometrical and/or in the material behaviour) are present.

In this case another command must be used: STAT_NON_LINE.

Similarly, other commands must be used in the case of a dynamic simulation (DYNA_NON_LINE, DYNA_LINE_TRAN/DYNA_LINE_MODAL, DYNA_LINE_HARM...)

⇨ I will not speak about dynamic simulations because this is outside the goal of this presentation

(it would require also a lot of time and a basic knowledge of structures dynamics from you)

Again, other commands must be used in the case of a thermal simulation (THER_LINEAIRE, THER_NON_LINE, THERM_NON_LINE_MO)

Command file syntax

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

The concept created by the solver (RESU) contains the results of the analysis: the displacement field of the model.

This is a **nodal field**.

What about stresses, strains, reaction forces...?

⇨ They are not available directly

⇨ You must calculate them, not by hand.

Command file syntax

```
DEBUT();

MA=DEFI_MATERIAU(ELAS=_F(E=210000000000.0,
                         NU=0.3,),);

MAIL=LIRE_MAILLAGE(FORMAT='MED',);

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

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

CHAR=AFFE_CHAR_MECA(MODELE=MODE,
                    DDL_IMPO=_F(GROUP_MA='Down',
                                DX=0.0,
                                DY=0.0,
                                DZ=0.0,),
                    PRES_REP=_F(GROUP_MA='Up',
                                PRES=100.0,),);

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

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

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

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

FIN();
```

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

⇨ CALC_ELEM = calculate element

⇨ This command allows to calculate one or more fields associated to an element

stresses, strains...

⇨ The fields to be calculated are specified by means of the OPTION keyword

⇨ The calculation is performed starting from the concept RESU (RESULTAT=RESU) containing the results of our simulation (only nodal displacements)

Enrichment/modification of an existing concept

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

The concept created by the command is RESU. It already exists because it has been created by MECA_STATIQUE.

The goal of the command is the **enrichment** of this concept: some new fields are added (they are specified by means of the OPTION keyword)

⇨ remember that the concept produced by the command must not exist

⇨ we must declare our goal by means of the keyword ***reuse***

⇨ otherwise we must change the name of the concept on the left of "=".

Useless keywords

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

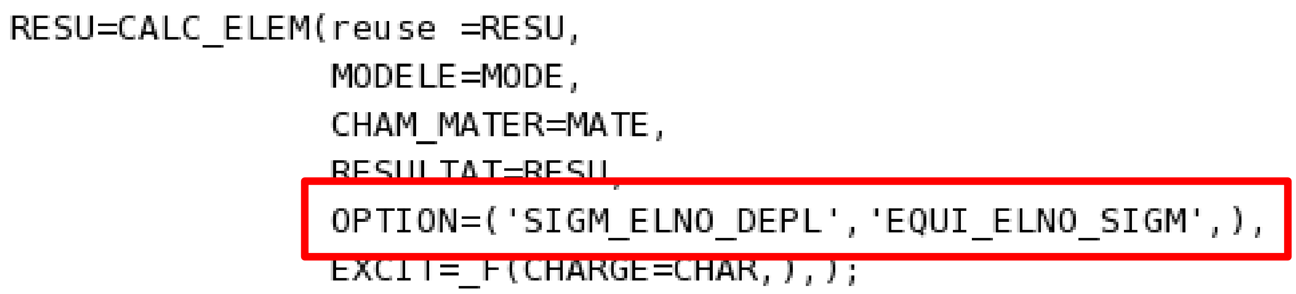
The calculation is performed starting from a **concept of type "result"** (RESULTAT=RESU).

This kind of concept contains all the model informations in addition to the nodal displacement field (which is effectively the result):

⇨ the keywords MODELE, CHAM_MATER and EXCIT are therefore useless in this case and they are ignored

⇨ these keywords must be used only if the informations passed through them are not already available in the concept given by RESULTAT

Field name meaning



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

Which is the meaning of the field names to be calculated?

SIGM_ELNO_DEPL

What you want to calculate
SIGM=stress tensor

Where you want to calculate
ELNO=nodes of the element

From which values the calculation have to be carried out
DEPL=from the displacement field at nodes

MANCHESTER 1824

The University of Manchester

Field name meaning

SIGM_**XXXX**_DEPL

↑

ELGA,ELNO, NOEU

⇨ ELGA (**el**ement **ga**uss): the calculation must be done at the Gauss points of the elements

⇨ ELNO (**el**ement **no**de): the calculation must be done at the nodes of each element, considered separately from the neighbouring elements

more than one value are calculated for each node, one value from each element sharing that node

⇨ NOEU (node): the calculation must be done at each node

a mean value of the values coming from each element sharing the node is calculated'

Field name meaning

The difference between the values calculated at ELNO and the ones calculated at NOEU is due to the FE discretization:

SIGM_ELNO_DEPL: $u \rightarrow \epsilon \rightarrow \sigma$

Only the displacements u are continuous across the edges/faces of two contiguous elements

Strains and stresses are continuous only inside each element.

Command file syntax

```
DEBUT();

MA=DEFI_MATERIAU(ELAS=_F(E=210000000000.0,
                         NU=0.3,),);

MAIL=LIRE_MAILLAGE(FORMAT='MED',);

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

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

CHAR=AFFE_CHAR_MECA(MODELE=MODE,
                    DDL_IMPO=_F(GROUP_MA='Down',
                                DX=0.0,
                                DY=0.0,
                                DZ=0.0,),
                    PRES_REP=_F(GROUP_MA='Up',
                                PRES=100.0,),);

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

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

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

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

FIN();
```

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

➪ CALC_NO = calculate node

➪ This command allows to calculate one or more fields associated to a node and it works like CALC_ELEM

➪ You can calculate nodal fields: reaction forces, nodal forces and all the fields of the type xxxx_NOEU_xxxx

➪ The field xxxx_NOEU_xxxx can be calculated only if the corresponding ELNO field exists in the concept given by RESULTAT

SIGM_NOEU_DEPL can be calculated only if SIGM_ELNO_DEPL exists!

xxxx_ELNO_xxxx → xxxx_NOEU_xxxx

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


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

Trying to calculate SIGM_NOEU_DEPL and EQUI_NOEU_SIGM without first calculating the corresponding xxxx_ELNO_xxxx fields makes Code_Aster issuing a warning and the calculation is not performed:

```
!-----------------------------------------------------------------------------!
! <A> <PREPOST5_4>                                                            !
!                                                                             !
!                                                                             !
! Champ inexistant SIGM_ELNO_DEPL numero d'ordre 1 pour le calcul de l'option SIGM_NOEU_DEPL !
!                                                                             !
!                                                                             !
!                                                                             !
! Ceci est une alarme. Si vous ne comprenez pas le sens de cette              !
! alarme, vous pouvez obtenir des résultats inattendus !                      !
!-----------------------------------------------------------------------------!


!-----------------------------------------------------------------------------!
! <A> <PREPOST5_4>                                                            !
!                                                                             !
!                                                                             !
! Champ inexistant EQUI_ELNO_SIGM numero d'ordre 1 pour le calcul de l'option EQUI_NOEU_SIGM !
!                                                                             !
!                                                                             !
!                                                                             !
! Ceci est une alarme. Si vous ne comprenez pas le sens de cette              !
! alarme, vous pouvez obtenir des résultats inattendus !                      !
!-----------------------------------------------------------------------------!
```

Command file syntax

```
DEBUT();

MA=DEFI_MATERIAU(ELAS=_F(E=210000000000.0,
                         NU=0.3,),);

MAIL=LIRE_MAILLAGE(FORMAT='MED',);

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

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

CHAR=AFFE_CHAR_MECA(MODELE=MODE,
                    DDL_IMPO=_F(GROUP_MA='Down',
                                DX=0.0,
                                DY=0.0,
                                DZ=0.0,),
                    PRES_REP=_F(GROUP_MA='Up',
                                PRES=100.0,),);

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

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

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

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

FIN();
```

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

⇨ IMPR_RESU = print results

⇨ This command allows to print the results in one or more output files

⇨ The format for the output file is chosen by means of the FORMAT keyword

⇨ If you don't specify any field (NOM_CHAM), all the fields contained in the result (RESULTAT=RESU) will be written in the output file.

⇨ Note that no concepts are created by this command

Output formats

Many formats are supported for output: I-DEAS, MED, CASTEM, GMSH, **RESULTAT**

If FORMAT='RESULTAT' is used, the output will be written in a text file by means of one or more tables

For example, for SIGM_NOEU_DEPL field, all the stress tensor components at each node of the mesh are written in table format:

```
CHAMP AUX NOEUDS DE NOM SYMBOLIQUE  SIGM_NOEU_DEPL
NUMERO D'ORDRE: 1 INST:  0.00000E+00
NOEUD        SIXX         SIYY         SIZZ         SIXY         SIXZ         SIYZ
N1       -1.82656E+01 -5.64194E+00 -1.01889E+02  2.75105E-02 -5.06759E+00  7.41975E+00
N2       -3.68609E+00 -4.12042E-01 -9.63465E+01 -6.18302E-02  4.15495E-01 -9.89947E-01
N3        2.48423E-01  1.25553E-01 -9.79220E+01 -1.47314E-01  7.58480E-01  7.53358E-01
N4        4.13188E-02  4.48615E-01 -9.98672E+01  4.36026E-02  1.08470E-01  3.49475E-01
N5        2.70524E-01  3.95353E-01 -9.97949E+01 -5.41052E-02 -2.77949E-01  3.34413E-01
N6       -1.01356E+00 -1.11549E-01 -9.61290E+01 -3.10158E-01 -7.29095E-04 -8.87817E-01
N7        7.09810E-02  6.61161E-02 -9.68025E+01  2.06728E-01 -7.44807E-01  7.81319E-01
N8       -3.50697E+01 -3.13072E+01 -1.06764E+0
N9       -3.54358E+00 -1.28750E+00 -9.97130E+0
N10      -7.92135E+00 -1.42722E+00 -9.74700E+0
N11       4.66344E-01  2.06646E-01 -9.54475E+0
N12      -1.23772E+00 -5.21230E-01 -9.34567E+01  6.17808E-01  1.14656E+00  3.70877E-01
N13      -9.40383E-01 -3.40660E-01 -9.79413E+01 -1.19033E-01  9.71153E-02  6.31982E-01
N14       3.39288E-01  2.76772E-01 -9.47312E+01 -2.89874E-02  4.32728E-01 -5.22789E-01
N15       4.42340E-01  8.41409E-02 -9.96834E+01 -9.78498E-03  4.87942E-01 -1.98468E-01
```

```
IMPR_RESU(FORMAT='RESULTAT',
          RESU=_F(MAILLAGE=MAIL,
                  RESULTAT=RESU,
                  NOM_CHAM=('SIGM_NOEU_DEPL','EQUI_NOEU_SIGM','DEPL',),),);
```

Command file syntax

```
DEBUT();

MA=DEFI_MATERIAU(ELAS=_F(E=210000000000.0,
                         NU=0.3,),);

MAIL=LIRE_MAILLAGE(FORMAT='MED',);

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

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

CHAR=AFFE_CHAR_MECA(MODELE=MODE,
                    DDL_IMPO=_F(GROUP_MA='Down',
                                DX=0.0,
                                DY=0.0,
                                DZ=0.0,),
                    PRES_REP=_F(GROUP_MA='Up',
                                PRES=100.0,),);

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

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

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

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

FIN();
```

⇨ FIN = end of the command file

⇨ Each command file must be terminated by this command.

Documentation

The commands we have analysed are the most common and basic ones.

A huge documentation describing all the commands and options is available on the project web-site:

http://www.code-aster.org

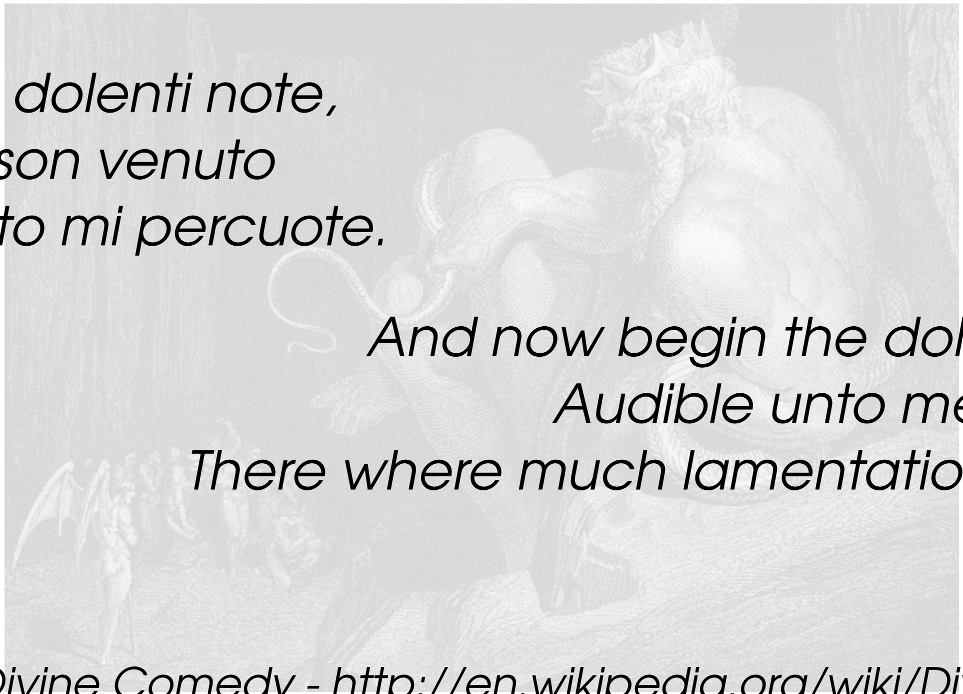
There are three types of documents:

⇨ Doc U: **u**ser documentation for command syntax and options

⇨ Doc R: **r**eference documentation where detailed technical informations about different subjects of the code are explained

⇨ Doc V: documentation of the **v**alidation test cases where the tests used to validate all the commands and options of the code are described in details.

Documentation



Ora incominciar le dolenti note,
a farmisi sentire; or son venuto
là dove molto pianto mi percuote.

And now begin the doleful notes to grow
Audible unto me; now am I come
There where much lamentation strikes upon me.

Dante Alighieri – Divine Comedy - http://en.wikipedia.org/wiki/Divina_commedia

The documentation is in French. An English version will be available soon.

In the meanwhile use a translator software from French to English, even if the result is not good as the translation above.

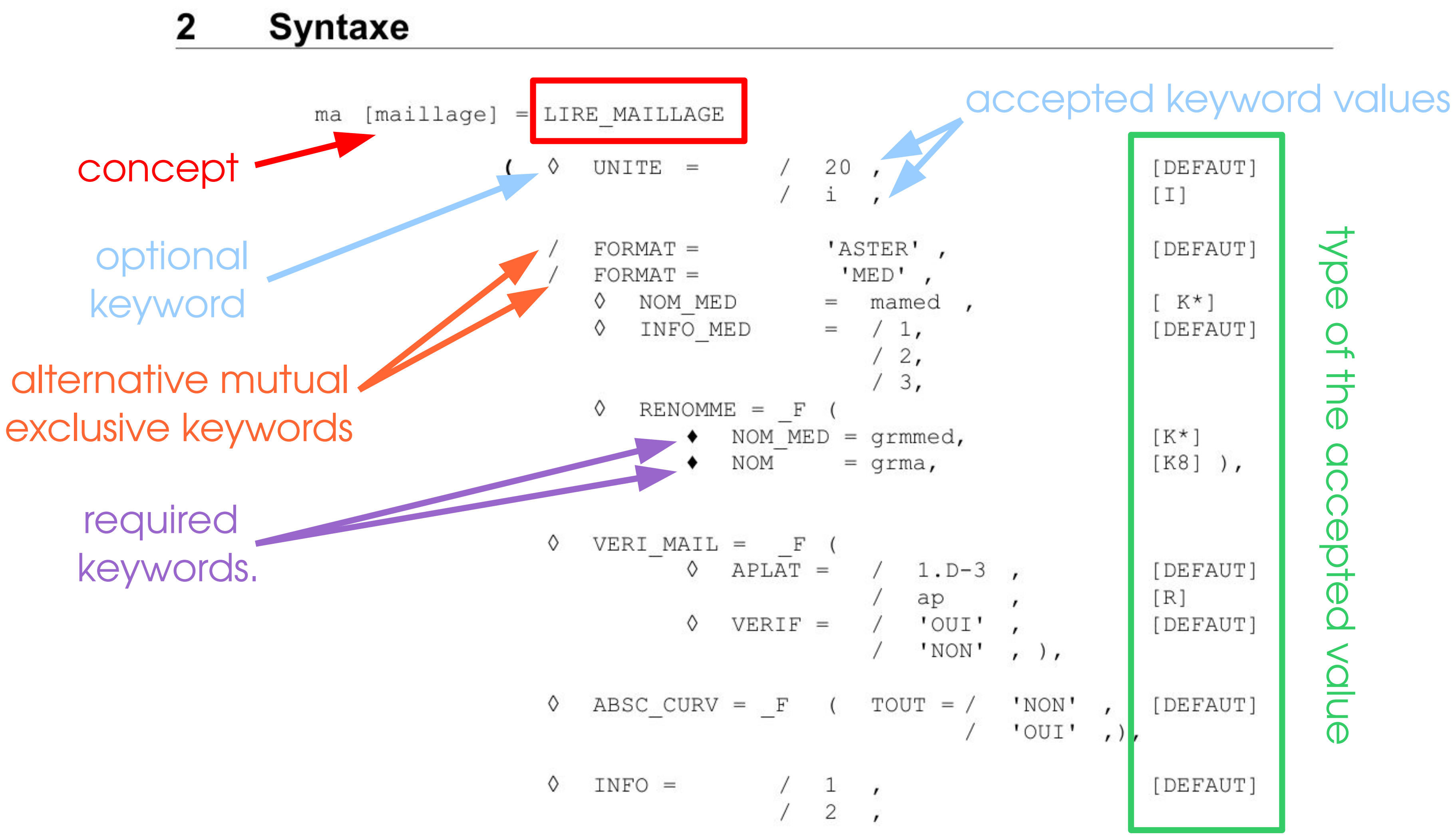
A French to English glossary of Code_Aster abbreviations is available here:

http://www.code-aster.org/wiki/doku.php?id=en:p01_util:p120_terms

Documentation

How the syntax of each command (Doc U) is explained?

For example, for the LIRE_MAILLAGE documentation:



2 Syntaxe

```
ma [maillage] = LIRE_MAILLAGE

            ( ◊  UNITE  =       /  20  ,                          [DEFAUT]
                                /  i   ,                          [I]

              /  FORMAT =          'ASTER'  ,                     [DEFAUT]
              /  FORMAT =           'MED'  ,
                 ◊  NOM_MED     =  mamed   ,                      [ K*]
                 ◊  INFO_MED    =  / 1,                           [DEFAUT]
                                   / 2,
                                   / 3,
                 ◊  RENOMME  =  _F  (
                       ♦  NOM_MED  =  grmmed,                     [K*]
                       ♦  NOM      =  grma,                       [K8]  ),

              ◊  VERI_MAIL  =   _F  (
                       ◊  APLAT  =   /  1.D-3  ,                  [DEFAUT]
                                     /  ap     ,                  [R]
                       ◊  VERIF  =   /  'OUI'  ,                  [DEFAUT]
                                     /  'NON'  ,  ),

              ◊  ABSC_CURV  =  _F   (   TOUT  =  /  'NON'  ,      [DEFAUT]
                                                 /  'OUI'  ,),

              ◊  INFO  =        /  1   ,                          [DEFAUT]
                                /  2   ,
```

concept
optional keyword
alternative mutual exclusive keywords
required keywords.
accepted keyword values
type of the accepted value

As you see, Eficas reproduces this structure and its rules!

Test cases

How can I start using effectively Code_Aster for a certain type of simulation?

Before trying to create your FE model from scratch:

⇨ Get familiar with the basic commands we have seen today

⇨ Search for informations about the kind of simulation you want to do in the R documents

⇨ Go through the U documents for the commands you are going to use (command file templates are usually given in R documents)

⇨ Study the test cases available in Code_Aster covering the commands you want to use

each version of Code_Aster comes with a huge test case base. You can find it in $ASTER/$VERSION/astest

Code_Aster versions

Many versions of Code_Aster are available at the same time.

You always have two versions:

⇨ an **operating** version (at the moment, version 9)

This is the official release to be used for daily calculations

The features are frozen and bug corrections are guaranteed.

⇨ a **development** version (at the moment, version 10)

This version must be considered the unstable release

Features are added and removed and bug corrections are guaranteed

The syntax of the command can undergo heavily modifications at each new release

This version should be used only if you need a new feature not available in the operating version.

Code_Aster versions

For each version (operating and development), you have three branches:

⇨ the **NEW** branch

⇨ the **STA** branch

⇨ the **OLD** branch

For the **development** version, a **NEW** branch is released **weekly** following the new developments of the code

For the **operating** version, a **NEW** branch is released each time a bug correction is released

Each six months, a stabilisation of the code is done. The NEW branch becomes the STA branch and the STA branch becomes the OLD branch

A change in the versions is done similarly each two years.

Code_Aster at the University of Manchester

I've installed and I maintain both the operating and development versions of Code_Aster on the following clusters:

- ⇨ RedQueen
- ⇨ Mace01

Here are the details of the available versions:

⇨ development version:	NEW10	(version 10.2.6)
	STA10	(version 10.1.27)
	OLD10	(version 10.1.0)
⇨ operating version:	NEW9	(version 9.4.11)
	STA9	(version 9.4.0)

Code_Aster at the University of Manchester

All the informations needed to use Code_Aster on the clusters are available on a wiki page hosted on the CFD twiki:

http://cfd.mace.manchester.ac.uk/twiki/bin/view/Aster/WebHome

For the moment you can't use Salome-Meca on the cluster.

A wiki page will be soon available explaining how to use Code_Aster on the clusters directly from your local version of Salome-Meca.

Resources

Salome

Freely available from ***http://www.salome-platform.org/***

Good tutorials (in English!) explaining how to use Salome are available from the *User Section* of the same website.

Salome-Meca

Freely available from ***http://www.code-aster.org/***
(click on "téléchargement" and then select Salome-Meca from the menu on the left)

Code-Aster

Freely available from ***http://www.code-aster.org/***
(click on "téléchargement")

Resources

All these softwares run under Linux

⇨ An unofficial porting of **Code_Aster** to **Windows** is available at the following address (I've never tried it):

http://www.necs.fr/gb/telechargement.php

⇨ An official porting of **Salome** to **Windows** is available for testing at the following address:

http://files.salome-platform.org/cea/adam/salomewindows/download/

MANCHESTER
1824

The University
of Manchester

Thank you for your attention!