Parametric calculations - Distribution of calculations

Summary:

Certain studies result in carrying out a more or less significant number of analyses (several thousands sometimes), corresponding to the variations of the parameters. L'use of standard calculations, consisting to modify or create the command file for each game of parameter is tiresome, difficult to manage and source of error. The method presented in this document makes it possible to facilitate the implementation of such parametric studies by carrying out a minimum of interventions.

From the standard study and of a file containing the set of parameters, calculations are declined and carried out automatically.

Parametric calculations being independent from/to each other, it is possible to use the resources machine available by subjecting in parallel calculations.

Contents

1 General information

1.1 Presentation

One **parametric study** is a standard study (mitre STUDY) in which one wishes to vary one or more parameters, such as for example:

- Parameters materials : Young modulus, limits elastic,…
- Geometrical parameters : thickness of the hull, section of a beam,…
- Parameters loadings : pressure, orientation of a force,…
- …

If one carries out these calculations like a standard calculation, the number of calculations to be realized can become very important, several thousands of calculations and their implementation becomes very tiresome.

The objective of this document is to describe methodology to be implemented in *Code_Aster* to carry out this kind of study with the minimum of intervention.

All in all, a parametric study is the data of a nominal study (command file common to each parameter) and of a set of parameters: *Code_Aster* the nominal study in several studies according to the provided parameters will decline, then will carry out each variation by taking of account the resources machine available.

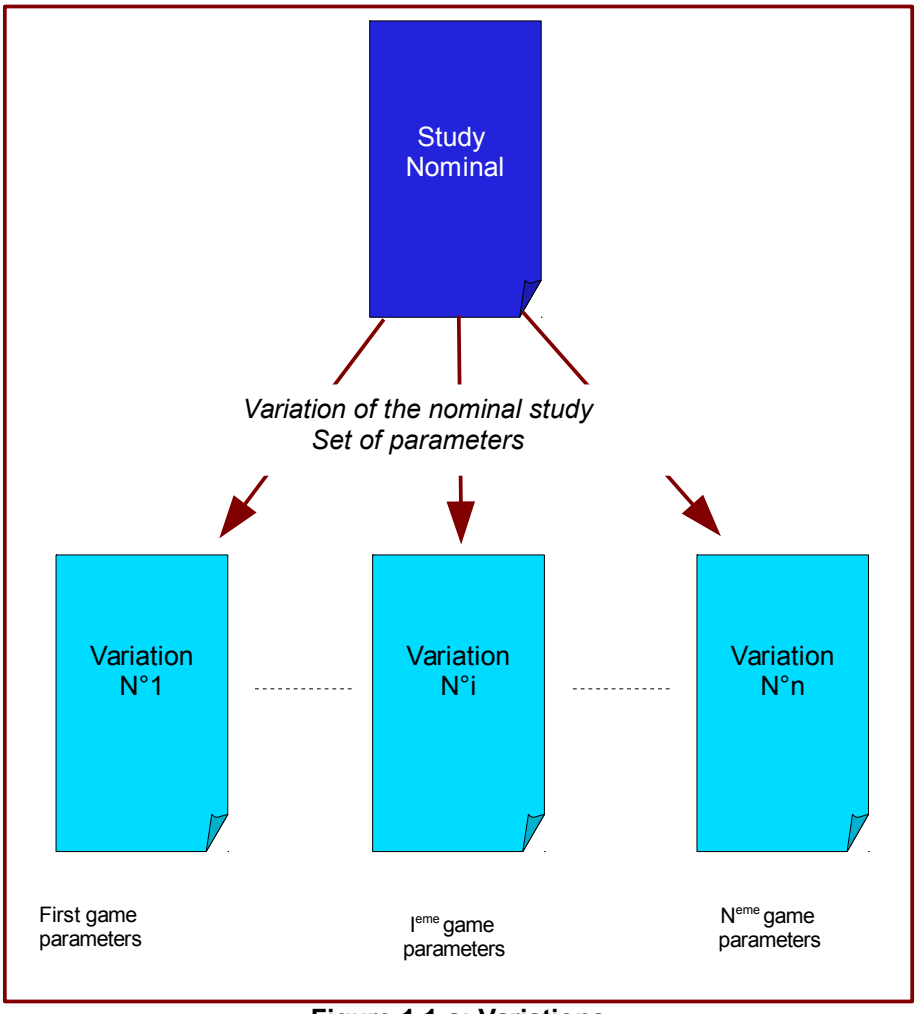


Figure 1.1-a: Variations

1.2 Object of the tutorial

The objective of the tutorial is to provide to the user a maximum of information to enable him to implement parametric studies.

This document answers the following questions:

- How to prepare a parametric study (files,…),
- How to launch a parametric study (repertoires, options,…),
- How to generate and read again a base,
- How to use the resources calculations available (distribution of calculations,…),
- How to recover the results of the parametric study (command file, repertoire of results).

In the last paragraph one presents a detailed example of a parametric study.

In this document, it is supposed that the user is familiarized with standard calculations *Code_Aster*. In the contrary case it is invited to to consult the following documents [U1.04.00],… and starting with carrying out a classical study.

2 Operating process of parametric calculations

2.1 Principle

The implementation of parametric studies in *Code_Aster* is relatively simple. Nevertheless it is important that:

- your standard study turns without problem before undertaking this kind of calculation.
- you defined well as a preliminary the parameters which you wish to vary:
  - their names,
  - their values,
  - scenarios of calculations to be carried out.

In the case of a standard study you lay out at least :

- In data:
  - of a command file (`.comm`)
  - data files necessary (`.mail`, `.med`,...)
  - parameters (time and memory)
- At exit:
  - of a file message (`.mess`)
  - of a file result (`.resu`, `.med`,...)

To carry out a parametric study you need moreover:

- In data:
  - of a file describing the game of parameter (`.distr`)
  - of a repertoire results (`.repe`)
  - of an option Astk `distribution=oui`. According to the waiter, one can need to specify the class batch in which calculations will be subjected.
  - the parameters (time/memory) are the same ones for all calculations, those of the main job can be fixed independently of the study by the mechanism of plugin (cf [U1.04.00]).
- At exit:
  - of a repertoire results (`.repe`)

In the following paragraphs we include all these points in details.

In the following table we summarized the obligatory and optional elements which one finds in the case of a parametric study.

Elements	Studies standards	Parametric studies	Comments	§
Command file	C	C	- The command set defines - Use of the parameters	
File of grid	C	C	Definition of the grid	
distr	X	p	Definition of the parameters and the scenarios of calculation: value of the parameters, either explicitly or in way calculated	2.2.1
.mess	C	X	File message	
.resu	C	X	File result with the format aster	
.rmed,…	C	p	File result with the format med	
repe_out	C	p	Repertoire of the results	
hostfile	X	p	Allows to launch parallel calculations	2.4
astk	C	p	Options of launching **distribution = yes**	2.3
base	C	p	The bases are stored in repe_out	2.4
continuation	C	C		2.4

X : pace of file
C : Fclassical ichier
p : fito shit parametric

Table 2.1-1: Synthesis of the obligatory and optional elements in a parametric study

2.2 Implementation

The actions to undertake to implement a parametric calculation are the following ones:

1) Writing of a set of parameters,
2) Use of the parameters in the command file,
3) Under Astk mitre STUDY : definition of the repertoire results and addition of the set of parameters,
4) Under small Astk TOOLS : definition of the type of calculation

2.2.1 Writing of a set of parameters

Description, in language python, of the game of the parameters is carried out in the file '.distr'. in which one finds

- the list of the parameters,
- the values which these parameters (scenarios of calculations) will take successively.

Example:

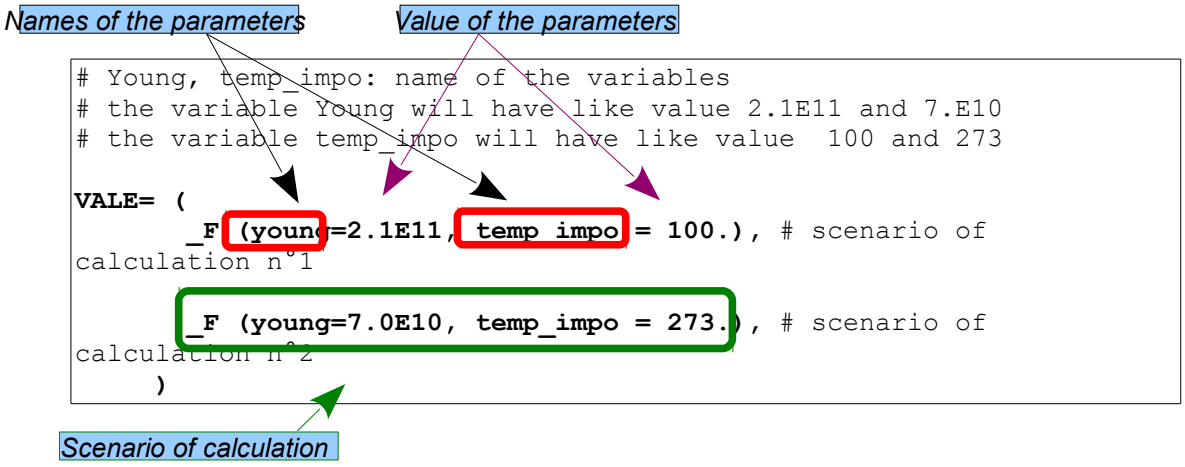


Figure 2.2.1-a: Explicit example of File "distr"

In this example, there are two scenarios of calculation (n°1 and n°2) with the parameters Young and temp_impo who will take successively as value $(2.1E11, 100)$ and $(7.0E10, 273)$.
This stage, although the name of the parameters is sufficiently explicit, nothing makes it possible to say how they will be used. For the parameter Young, that lets suppose that this one will be used at the time of the definition of material, but nothing for the moment makes it possible to affirm it.

From the point of view language python, the file '.distr' a list of dictionaries of name contains VALE, (each dictionary corresponding to a calculation). One uses _F as in the command file to define these dictionaries.

The definition of the parameters of the scenarios of calculation can be given explicitly or calculated.

Notice

In the preceding example, the definition of the parameters and the scenarios of calculations are given explicitly. It is possible to call on the features of the language python to define them in an automatic way, a practical example is presented in the §3.2.2). The name of the parameters is defined by the user, it is advised to use explicit names.

2.2.2 Use of the parameters in the command file

In the paragraph §2.2.1, one defined the set of parameters (name, values and scenarios of calculation), the objective now is to use these parameters in the command file, at the desired place.

The taking into account of these parameters in the command file is carried out in two times:

1) declaration of the variables python: parameters
2) use of these variables in the command file.

- **Declaration of the parameters (variable python)**

  Before any use in an order or an unspecified expression, it is **obligatory** that the parameters are known. The objective is not to initialize them once again, but simply to declare them as variables python.

  Example: in the file `.distr` precedent, one defined the parameters `Young` and `temp_impo`. One will thus find in the command file the following lines:

```
BEGINNING ()
…
Young = 0.
temp_impo = 0.
…
END ()
```

Figure 2.2.2-a: Declaration of the parameters

Note:

The values affected in the command file will not be used, they will be replaced automatically by those defined in the file `'distr'`, for each scenario of calculations.
Avoid choosing a name of parameter identical to a keyword of Code_Aster. Substitution being made by the regular expression `'^ (*) name *= *'`, there could be confusion. Breakage being meaning, to use names in small letters makes it possible to avoid this pitfall.

- **Use of the parameters**

  The use of the parameters in the command file is carried out in a classical way, within the orders, of the mathematical expressions,…

  Example:

```
BEGINNING ()
…
young=0.
temp_impo=0.
…
mat=DEFI_MATERIAU (ELAS=_F (E=Young,…))
…
t0 = AFFE_CHAR_MOVIES= (THER_IMPO=
                          _F (TEMP=temp_impo,
                              GROUP_NO=' CHAUD'.)
…

END ()
```

Figure 2.2.2-б: Command file

2.2.3 Type of the parameters

The type of the parameters can be unspecified. Then, it is necessary to keep in mind that the replacement is textual at the time of the instanciation of the command set for a given game of parameters.

Example:

That is to say set of parameters:

```
VALE= (_F (  Young  =  2.1e11,
           index  =  4,
           nomp   =  "'INST'",
           objf   =  'FON1',),
…)
```

and command set makes dizzy it:

```
Young  =  2.e11
index  =  0
nomp   =  'X'
objf   =  FON0
…
```

The command set declined on the set of parameters will be:

```
Young  =  2.1e11
index  =  4
nomp   =  'INST'
objf   =  FON1
…
```

The difference is visible for the last two parameters where it is seen that one used the text of the character string of `nomp` and `objf` (and not the representation of this one, therefore a set of dimensions/quotation marks disappeared).
That allows paramétriser the use of concepts. Here, the function is used `FON1` like parameter `objf`.

2.2.4 Parameters of type file

Whenever the variable parameter is a file (for example to read a different grid), it is a question of making variable the name of this file (chain of text or number of file) and of using DEFI_FICHIER.
Example which mixes a file name in the form of a character string and the same thing while using a number of grid:

That is to say set of parameters:

```
VALE= (   _F (   fname  =  "'/tmp/maillage_01.mmed'",
           index  =  1,),
         _F (   fname  =  "'/tmp/maillage_02.mmed'",
           index  =  2,),
…)
```

and LE file orders:

```
fname  =  'nom_de_fichier_med'
index  =  0

# by using the file name
DEFI_FICHIER (ACTION=' ASSOCIER', UNITE=20, FICHIER=fNheart)
meshA = LIRE_MAILLAGE (FORMAT=' MED', UNITE=20)

# by using an index of file (one builds the name
# of the file starting from the index on 2 characters
```

```
# with 0 front)
filename = '/tmp/maillage_ {I: 0>2} .mmed' .format (i=indice)
DEFI_FICHIER (ACTION=' ASSOCIER', UNITE=21, FICHIER=filename)
meshB = LIRE_MAILLAGE (FORMAT=' MED', UNITE=21)
```

2.3 Launching of the studies: Astk

The launching of parametric studies is managed directly by Astk. This launching is identical to that of a classical study with the help of the addition of additional information such as:

- addition in the profile of study of a line to take into account the file 'distr'
- addition in the profile of study of a line to define the repertoire results of the type 'repe'
- initialization in small the option, of the parameter distribution with 'YES'.

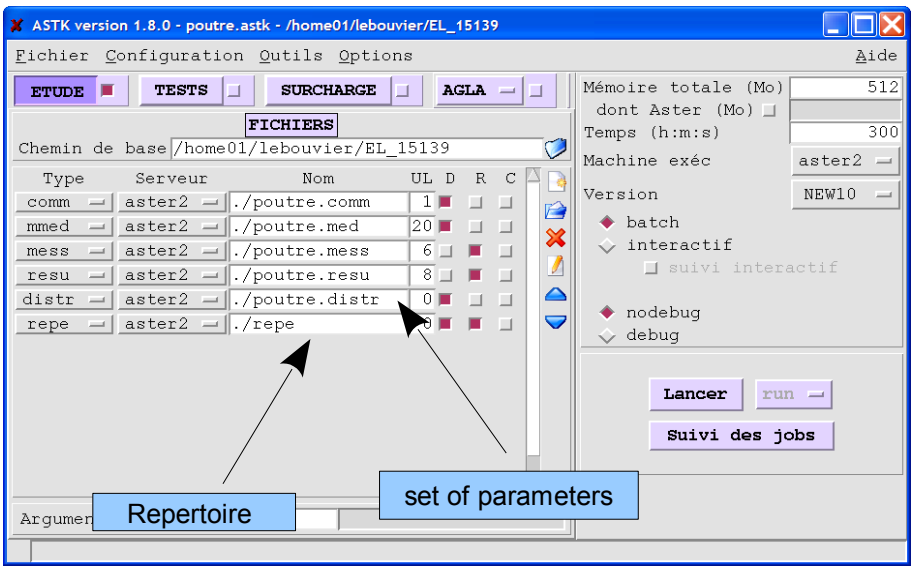


Figure 2.3-a: Definition of the set of parameters, repertoire (astk)

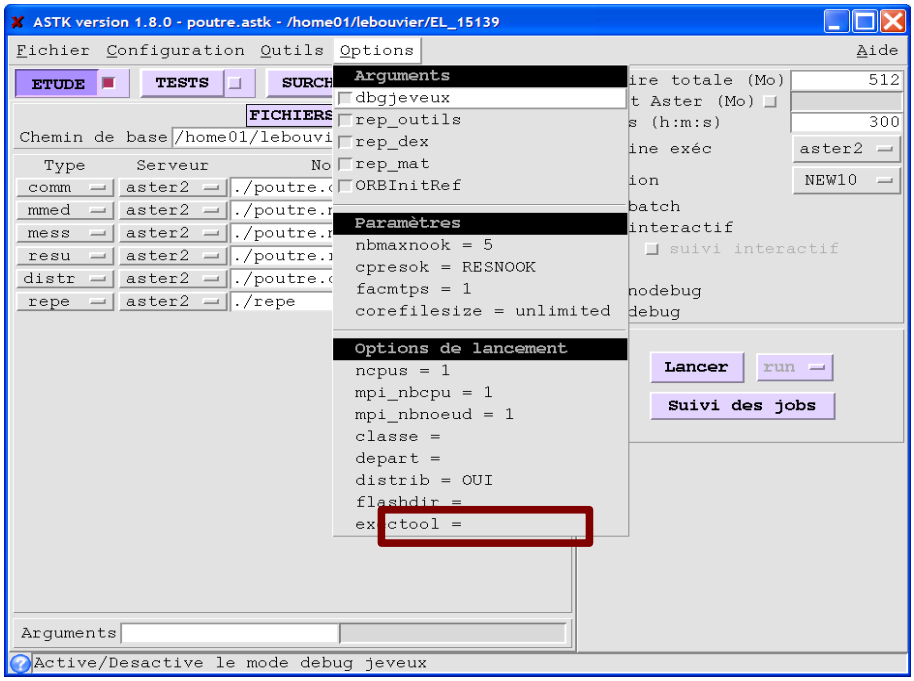


Figure 2.3-б: option distribution (Astk)

Remarks

It is possible, like all Code_Aster calculations, of launching parametric studies in command line mode with `as_run`. For that, it is necessary to have the file as a preliminary `.export`:

```
as_run --serv nom_etude.export
```

2.3.1 Management of calculations and the results

The repertoire of calculations is common to that of the results: it is defined in the profile of study under the type `'repe'`. After the execution of parametric calculations, one as many finds in this repertoire repertoires `calc_i` that launched calculations. A repertoire is also found `flash`. The detail of each one of these repertoires is presented below.

2.3.1.1 Management of calculations

The command files corresponding to each scenario are declined automatically starting from and the nominal set command file of parameters. These files are stored in the repertoires `calc_i`.
The files output (message and error) of each calculation are stored in only one repertoire, the repertoire `flash`. All information concerning the course of each execution is in this repertoire.

2.3.1.2 Management of the results

As for a classical calculation, the user has the possibility of generating output files (tables, graphs,…). He will have to define them in the standard command file. These exits will be common to all the scenarios of calculation.
For example, let us suppose that the user wants to print a table in the file of unit logical 38 and one result with the format med in the logical file of unit 39. It will have to define in the nominal command file, the name of each output file so that they can be stored in the repertoire `REPE_OUT` product automatically by *Code_Aster.* The extract of the command file below illustrates this definition:

```
DEFI_FICHIER (UNITE=38, FICHIER='. /REPE_OUT/TABLE.OUT')
IMPR_TABLE (TABLE=SIYY, UNITE=38, NOM_PARA= ('NODE', 'SIYY'));

DEFI_FICHIER (UNITE=39, FICHIER='. /REPE_OUT/POUTRE.RMED')
IMPR_RESU (FORMAT=' MED', UNITE=39, RESU=_F (RESULTAT=depl))
```

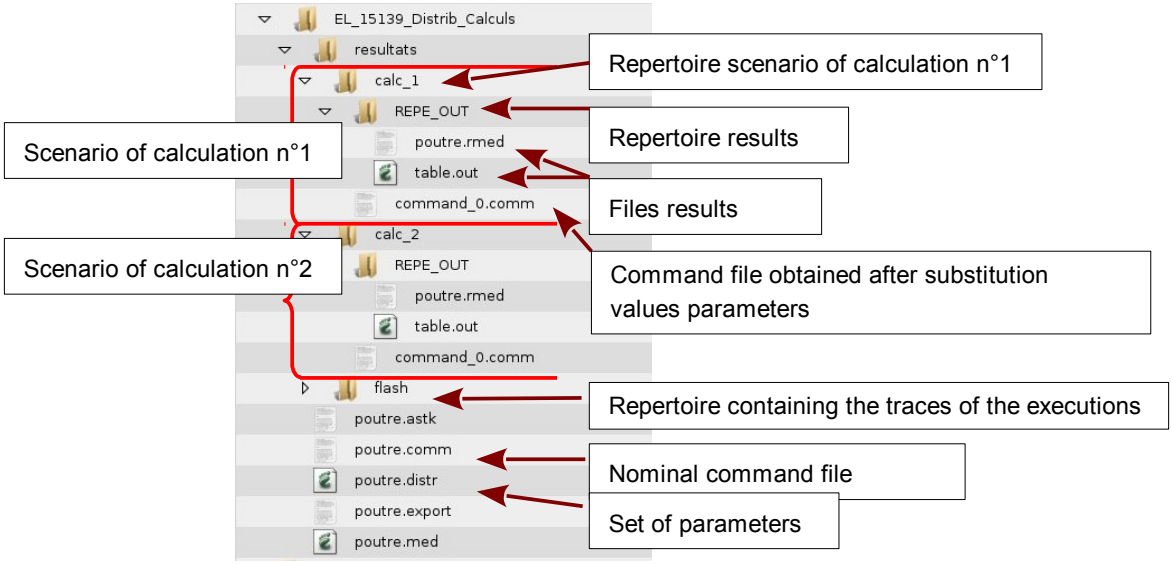
Figure 2.3.1.2-a: file design of exit and their logical units

Note:

In the case of parametric study files `.mess` and `.resu` are not created. However, consultation of the files of the repertoire `flash` allows to collect the information usually transmitted in `.mess`.

2.3.1.3 Tree structure of the repertoires

The figure below presents the localization of the files and repertoires of 2 scenarios of calculation.



2.4 Complementary features

2.4.1 To generate a base

As for a standard study, it is possible to generate a base. For that, it is enough to add in the profile of study an entry of the type 'bases' with for name that of the repertoire repe. One will find for each scenario of calculation the corresponding base stored in the repertoire calc_i/bases.

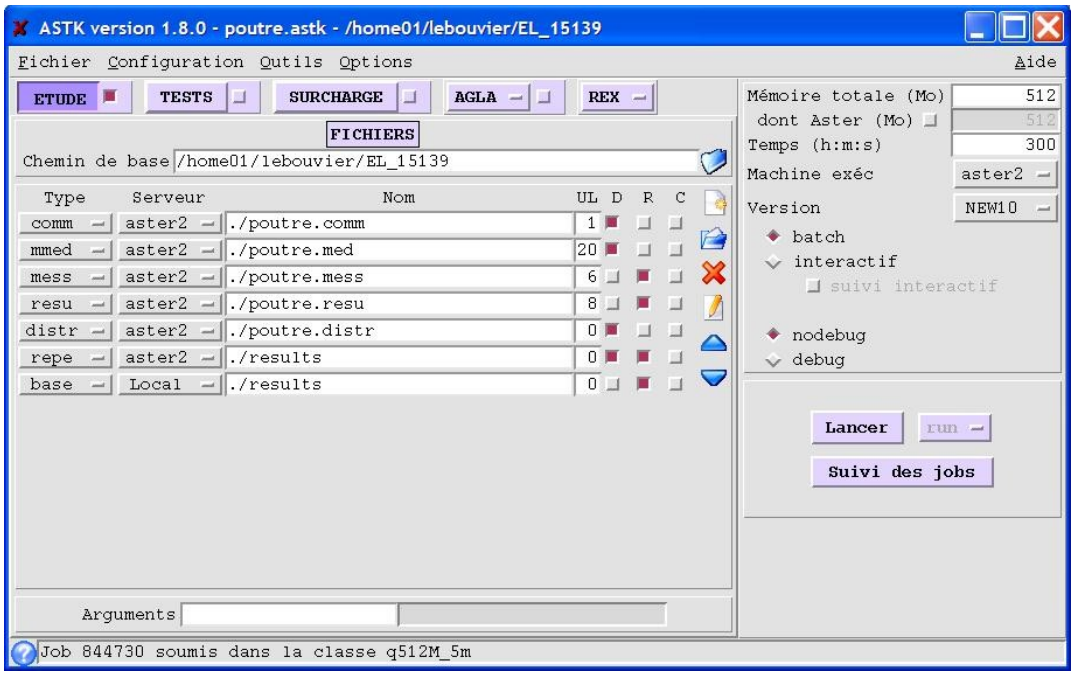


Figure 2.4.1-a: Example of profile of study to generate a base by calculation

2.4.2 To carry out a continuation

As for a standard study, it is possible to exploit a base. For that, it is enough to add in the profile of study an entry of the type 'bases' with for name that of the repertoire repe in which each base with reading is present in the repertoire calc_i/bases.

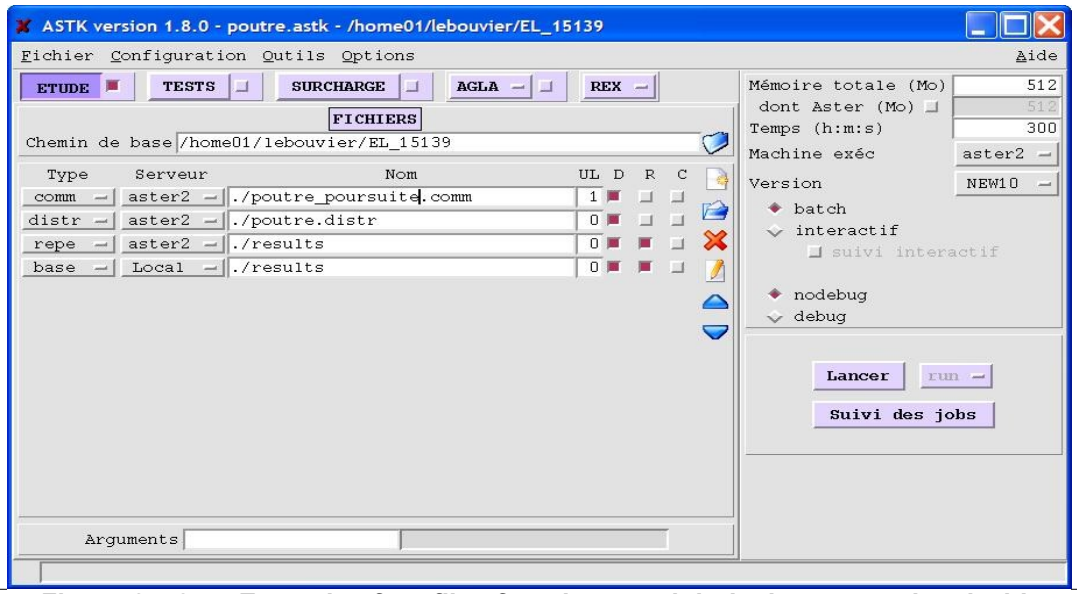
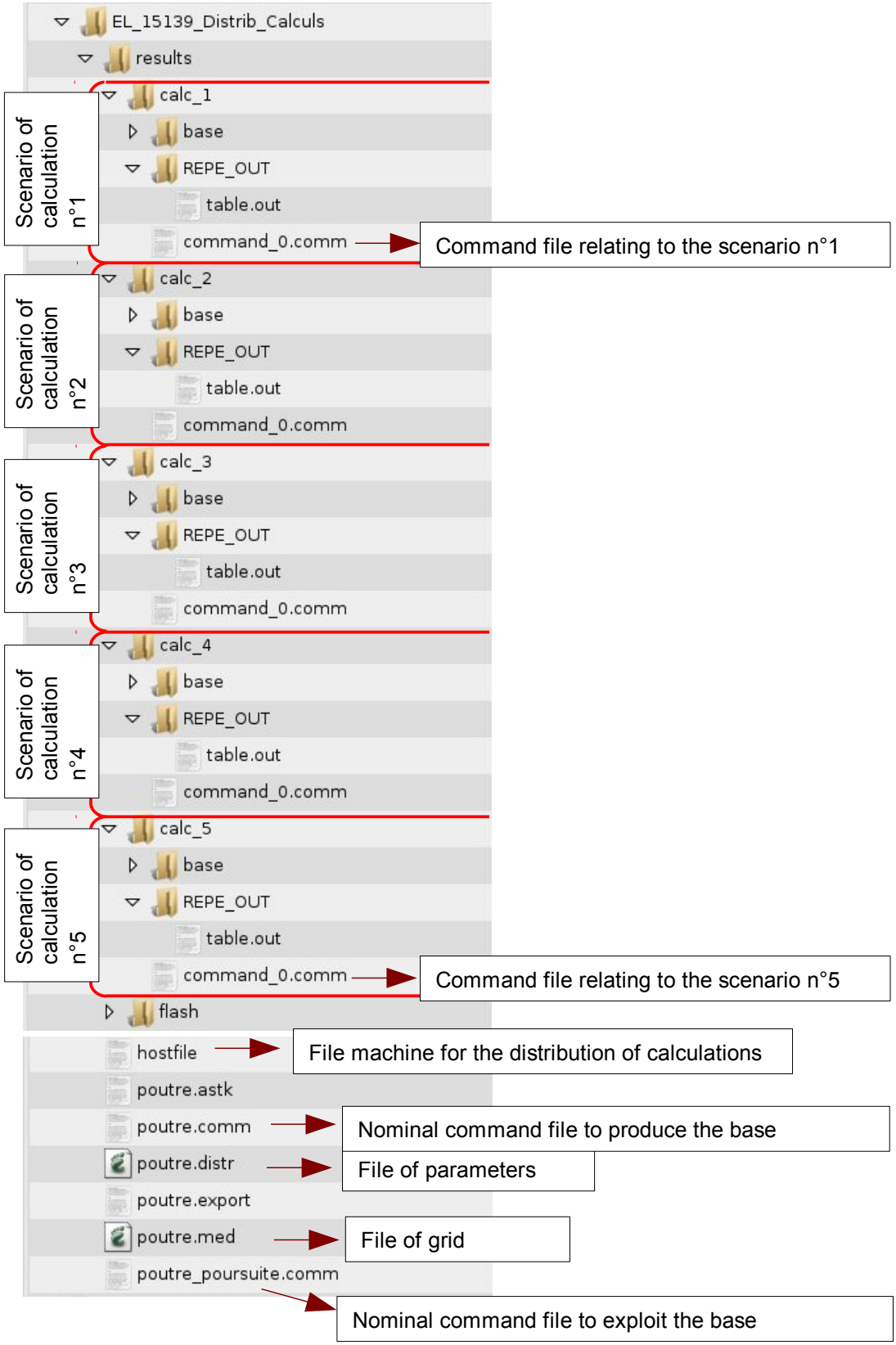


Figure 2.4.2-a: Example of profile of study to exploit the base associated with each calculation

2.4.3 Tree structure of the repertoires

One presents the tree structure of the repertoires in the presence of a basic reading for each of 5 calculations. The presence of a file will be noticed `hostfile` who will be the object of the following paragraph.



2.4.4 Distribution of calculations

In the case of parametric studies, calculations are independent from/to each other. It is thus possible to use the resources machine available at the time of the distribution of calculations. It is thus advisable to create a file of the type 'hostfile' defining the resources machine. One defines in it:

- the name of the node where the jobs will be subjected,
- the maximum number of job to be subjected at the same time,
- total allocated memory.

Note:

On a divided waiter, it is advised to use the file defined by the administrator and thus, not to redefine its own file of resources.
If the file `hostfile` *is not present in the profile of study, it is the file* `batch_distrib_hostfile` *present in the repertoire* `etc/codeaster` *who will be used by default.*

For example on the centralized waiter, Lbe resources are managed by the software of batch. In this case, the file `hostfile` declare simply the number of calculations which will be subjected at the same time. Here one can subject up to 32 calculations on each frontal node (the memory is indicated *infinite*, i.e. that one leaves the software batch manage):

```
[ataster1]
cpu=32
mem=9999999

[ataster2]
cpu=32
mem=9999999
```

Figure 2.4.4-a: Example of file `hostfile` with waiter of batch

So that your file of resource is taken into account, it is enough to add it in your profile of study by specifying the type 'hostfile'.

In the case of use of a set of machines available in interactive, the file `hostfile` could resemble:

```
# name of node
[machine1]
# number of CPU available
cpu=4
# total memory of the machine out of Mo
mem=4000

# name of the node
[machine2]
# number of CPU available
cpu=8
# total memory of the machine out of Mo
mem=4000
```

Figure 2.4.4-b: Example of file `hostfile` in interactive

One will be able to have up to 12 calculations carried out simultaneously according to the memory available distributed on the two machines. If each calculation requires 2 Go of memory, there will be to the maximum two calculations per machine not to exceed the full amount of memory available.

2.4.5 Pre/postprocessings common to all calculations

One lays out of 4 keywords in the file “distr”: PRE_CALCUL, UNITE_PRE_CALCUL, POST_CALCUL, UNITE_POST_CALCUL.
PRE_CALCUL (resp. POST_CALCUL) a text (a set of orders Aster) defines which will be included just afterwards BEGINNING (resp. right front END).
UNITE_PRE_CALCUL (resp. UNITE_POST_CALCUL) propose same operation except that one provides a logical number of unit.

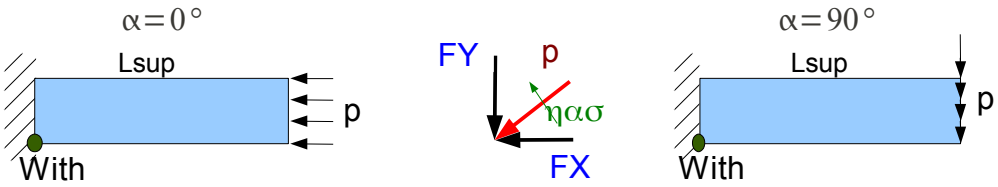
This modification is made for all the files of the type “COM” present in the profile.

This possibility is in particular used by the ordering of retiming MACR_RECAL to add a postprocessing to each calculation slave.

3 Examples of application

3.1 Presentation

The example selected to present the implementation of parametric studies is presented on the figure below. It is about a beam fixed at an end and subjected to a pressure distributed at the other end. The objective of this parametric study is to determine the evolution of the equivalent constraint of Von Mises along the higher line of the beam according to the orientation α.
The data files are those of the CAS-test `distr01a`.



Calculations will be carried out all them $22.5°$, for an angle α varying $0°$ with $90°$. The number of scenarios of calculation to be realized is thus of 5. In the table below we present the values which successively the parameters go FX and FY.

Scenario n°	Angle α	$FX=P.cos(a)$	$FY=P.sin(a)$
1	0.0°	10^6	0.
2	22.5°	9.23879×10^5	-3.82683×10^5
3	45.0°	7.07106×10^5	-7.07106×10^5
4	67.5°	3.82683×10^5	-9.23879×10^5
5	90.0°	0.	-10^6

Table 3.1-1: Values of the parameters

The command file of the nominal study is `distr01a.comm`.
In the continuation of this presentation, we will detail the implementation of this kind of calculation, namely:

- the generation of the file `'distr'`
- the use of the parameters in the command file.

3.2 Definition of the set of parameters and the calculation cases

It is possible to define the parameters in the file `'.distr'` in two ways:

- Explicitly: in this case the user provides all the values which can take the parameters.
- Calculated: in this case the user resorts to programming python
  - to calculate these parameters in conditional form automatically or not,
  - to define these scenarios of calculation automatically, for example by sweeping all the combinations of parameters possible or by choosing the values min/moy/max parameters.

3.2.1 File 'distr' explicit

In this case, one explicitly writes the values of the parameters in the file 'distr'. In the case of the example, it is presented in the following form (file distr01a.50 CAS-test):

```
VALE= (
       _F (F_Norm=1.E6,      F_Tang=0.),          # calculation case
n°1
       _F (F_Norm=9.23879E5, F_Tang=-3.82683E5), # calculation case
n°2
       _F (F_Norm=7.07106E5, F_Tang=-7.07106E5), # calculation case
n°3
       _F (F_Norm=3.82683E5, F_Tang=-9.23879E5), # calculation case
n°4
       _F (F_Norm=0.,        F_Tang=-1.E6),       # calculation case
n°5
     )
```

Figure 3.2.1-a: File 'distr' explicit

3.2.2 File 'distr' calculated

In this case, the writing of the file 'distr' is less simple, one calls on the programming of the language python. For this example, it is presented in the following form (file distr01a.51 CAS-test):

```
from maths importation pi, cos, sin
importation numpy

VALE = []
N = 5
list_theta = numpy.arange (N) * 22.5 * pi/180.
P = 1.e6

for has in list_theta:
   VALE.append (_F (F_Norm = P*cos (a),
                    F_Tang = P*sin (a),))
```

Figure 3.2.2-a: File 'distr' calculated

3.3 Use of the parameters in the command file

It is enough to refer, in the nominal command file, the names of the parameters present in the file 'distr'.

```
BEGINNING ()

# Initialization (here the values of the parameters will appear)
F_Norm=0.
F_Tang=0.

…

CHAR=AFFE_CHAR_MECA (MODELE=MODE,
                     FORCE_CONTOUR=_F (GROUP_MA = 'Near',
                                       FX = F_Norm ,
                                       FY = F_Tang),)
```

3.4 Postprocessings

The file `distr01a.11` give an example of postprocessing with second reading of the whole of the files results, fusion of the results in a single table, impression of a curve with the whole of the evolutions of the constraint…

4 The Councils of use

Here some advices:

- The standard study must is valid before being declined on the game of the parameters. It must turn without error.
- Before being declined on the game of the parameters, it is also important to optimize calculations by decreasing the execution time,…
- To check that the file of parameters 'distr' is correct.
- Attention with the disk space: if the number of calculations and their size are important, the safeguard of the bases can take up much space.
- To check is the definition of the variables in the command file, correct?
- To test the good performance of your parametric study by using a file 'distr' who defines one game of parameter.

5 Questions/Answers

Questions	Answers
Where are the parameters defined?	In the file: • .distr : one defines their names and their values in it • .comm : one defines the variable name in it python
Where are the scenarios of calculations defined?	In the file 'distr': VALE (_F (...) _F (...))
What does one have to make under Astk?	At least: • to define a line for the file 'distr' • to define a line for the repertoire 'repe' • to define the option of launching distribution=oui
Or does one find to you them results?	In the repertoire repe
I do not find my results!	You have in the command file created the following order: DEFI_FICHIER (UNITE=numero_unite_logique, FICHIER='.REPE_OUT/nom_fichier')
I do not find a file .mess and .resu	These files do not exist, on the other hand you can consult in the repertoire repe/flash files describing the execution of each scenario.
Don't my calculations function?	To analyze the messages possibly transmitted with the screen, to consult the files output and error present in the repertoire flash