# Macro-order MACR_RECAL

## 1 Goal

To readjust computation results on experimental results or other computation results.

Let us consider on the one hand one or more test results and on the other hand one or more calculations *Code_Aster* modelling these tests. MACR_RECAL allows to determine the parameters of these calculations (which can be parameters of law of behavior, loading, etc…) describing the tests as well as possible.

For more precise details on the algorithmy put in work, to refer to [R4.03.06].

# Contents

# 2 Syntax

```
Lr = MACR_RECAL                                                    [listr8]
     (  ♦ UNITE_ESCL       =  linked                                  [I]
        ♦ /  RESU_EXP      =  resu_exp                             [assd]
          /  RESU_CALC     =  resu_calc                            [assd]
          /  CURVE  = _F (
             ♦  FONC_EXP     =  fonc_exp                          [sd_fonction]
             ♦  NOM_FONC_CALC=  nom_fonc_calc                     [KN]
             ♦  PARA_X       =  para_X                            [KN]
             ♦  PARA_Y       =  para_Y                            [KN]
             ◊  WEIGHT       =  val_poids                         [R]
             )
          ◊  LIST_POIDS      =  weight                            [assd]
        ♦ /  LIST_PARA       =  will list_para                          [assd]
          /  PARA_OPTI = _F (
             ♦  NOM_PARA     =  will nom_para                        [KN]
             ♦  VALE_INI     =  val_ini                           [R]
             ♦  VALE_MIN     =  val_min                           [R]
             ♦  VALE_MAX     =  val_max                           [R]
             )

          ◊  UNITE_RESU      =  /91                               [defect]
                                /uni_r                            [I]
          ◊  ITER_MAXI       =  /10                               [defect]
                                /it                               [I]
          ◊  ITER_FONC_MAXI  =  /100                              [defect]
                                /it                               [I]

          ◊  RESI_GLOB_RELA  =  /1.E-3                            [defect]
                                / resi                            [R]
          ◊  TOLE_PARA       =  /1.E-8                            [defect]
                                / resi                            [R]
          ◊  RTOLE_FONC      =  /1.E-8                            [defect]
                                / resi                            [R]
          ◊  PARA_DIFF_FINI  =  /1.E-5                            [defect]
                                / coeff                              [R]
          ◊  GRAPH  = _F (
             ◊  UNIT         =  / 90                              [defect]
                                / uni_g                           [I]
             ◊  FORMAT       =  / 'XMGRACE'                       [defect]
                                / 'GNUPLOT'
             # If FORMAT = XMGRACE
             ◊  PILOT        =  / ''                              [defect]
                                / 'POSTSCRIPT'                    [KN]
                                / 'EPS'
                                / 'MIF'
                                / 'SVG'
                                / 'PNM'
                                / 'PNG'
                                / 'JPEG'
                                / 'Pdf'
                                / 'INTERACTIVE'
             )

          ◊  POSTING    =  / 'TOUTE_ITERATION'                    [defect]
                           / 'ITERATION_FINALE'
          ◊  METHOD     =  / 'LEVENBERG'                      [defect]
```

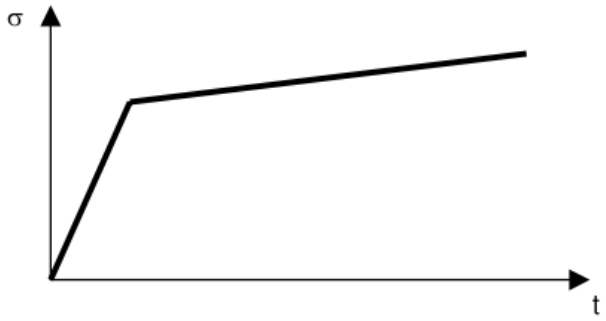
```
                        / 'FMIN'
                        / 'FMINBFGS'
                        / 'FMINNCG'
                        / 'GENETIC'
                        / 'HYBRID'

  ◊  CALCUL_ESCLAVE   = _F (
        ◊  LAUNCHING =/'INCLUSION'                    [defect]
                      / 'DISTRIBUTION'
        # If distribution
        ◊  MODE         =/'INTERACTIVE'                  [defect]
                         / 'BATCH'
        ◊  MEMORY    = memory                         [I]
        ◊  TIME         = time                        [I]
        ◊  CLASS        = class                          [KN]
        ◊  NMAX_SIMULT  = nmax                           [I]

        # If MODE = INTERACTIVE
        ◊  UNITE_SUIVI = unit                            [I]

  # if METHOD = GENETIC, HYBRID
  ◊  NB_PARENTS       =  /10                             [defect]
                         / nb_parents                    [I]
  ◊  NB_FILS          =  /5                              [defect]
                         / nb_fils                       [I]
  ◊  ECART_TYPE       =  /1.                             [defect]
                         / variation                        [R]
  ◊  ITER_ALGO_GENE   =  /10                             [defect]
                         / itergene                      [I]
  ◊  RESI_ALGO_GENE   =  /1.E-3                          [defect]
                         / resige                        [R]
  ◊  SEED             =  seed                            [I]

  ◊  DYNAMICS  = _F (
        ◊  MODE_EXP  = mode_exp                          [assd]
        ◊  MODE_CALC = mode_calc                         [assd]
        ◊  APPARIEMENT_MANUEL  =/'NOT'                   [defect]
                                 / 'YES' )

  ◊  INFORMATION         =  / 1
                        / 2                              [defect]

  # if METHOD = FMINBFGS, FMINNCG
  ◊  GRADIENT     =  / 'NON_CALCULE'                     [defect]
                     / 'NORMAL'
                     / 'ADIMENSIONNE'


  );
```

# 3 General presentation

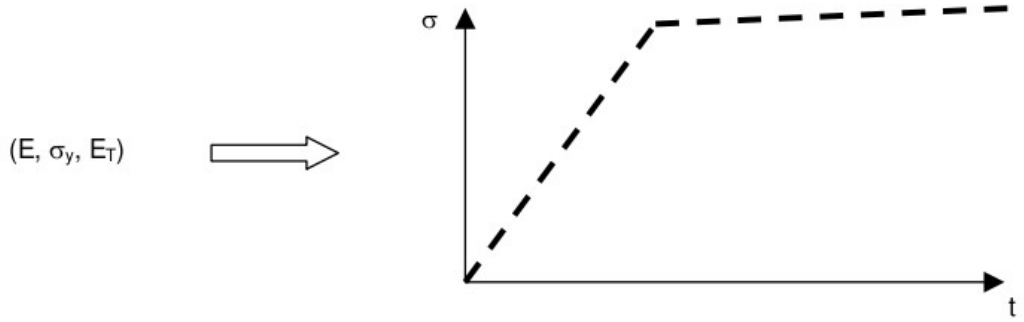
## 3.1 Principle of retiming

Let us consider the model problem of identification of the elastoplastic characteristics $E$, $\sigma_y$, $E_T$ (respectively elastic limit, Young modulus and module of work hardening) of a material on a uniaxial tensile test.

One has of one share the experimental traction diagram giving the evolution of the constraint according to the time and which is a data:




There is in addition one **function** of the 3 parameters which for each value of the triplet $E$, $\sigma_y$, $E_T$ return a calculated traction diagram:




The objective of retiming is then to answer the question:

*Which are the values of $(E, \sigma_y, E_T)$ describing my experiment as well as possible?*

## 3.2 Organization of retiming

To conclude a retiming, it is necessary to have the whole of following information:

- $N$ experimental curves (with each one of these curves can be allotted an arbitrary weight),
- $P$ parameters to readjust like, for each one, an estimate of its initial value, its minimal value and its maximum value,
- the command file modelling them $N$ tests which one wants to readjust,

- names of $N$ sizes to be extracted from the command file above and which will be readjusted on $N$ experimental curves. These sizes must be contained in a table resulting from POST_RELEVE_T.

The setting in data of this information requires the following organization then:

- a command file known as **Master** containing them $N$ experimental curves, them $P$ parameters, names of the sizes to be readjusted as well as other information suitable for retiming, the whole indicated in MACR_RECAL. The various formats used are specified in what follows,
- a command file known as **slave** modelling the experimental tests.

Indeed, retiming is a process **iterative** : the master file carries out the file slave, it recovers them $N$ curves calculated with the current prices of $P$ parameters, it compares the values of the curves calculated with those of the experimental curves, it from of deduced from new values for $P$ parameters and revival the file slave. This process continues until obtaining convergence.

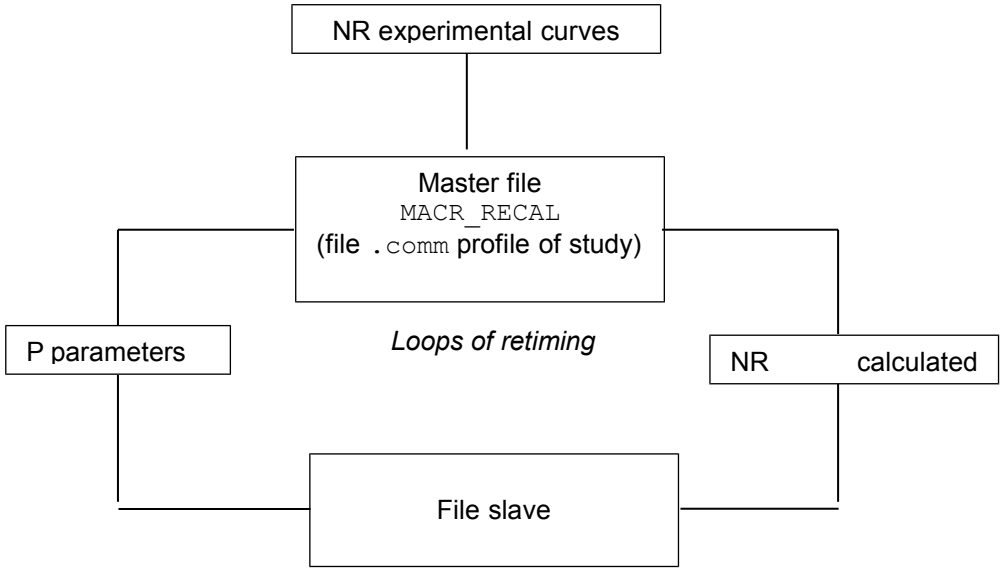



**Figure 3.2-a. Diagram of operation of the procedure of retiming for the classical case**

In the following part the operands are described of MACR_RECAL. One refers there to some notions of the language Python. It is however by no means necessary to know Python to use this macro order. The part “Example of use” is there to light the user.

The structure of data produced is a list of real containing the values of the parameters with convergence in the event of convergence or the last iteration in the contrary case.

## 3.3 Typical case of use: mode EXTERNAL

In this mode of use, the algorithm of optimization is external with *Code_Aster*.

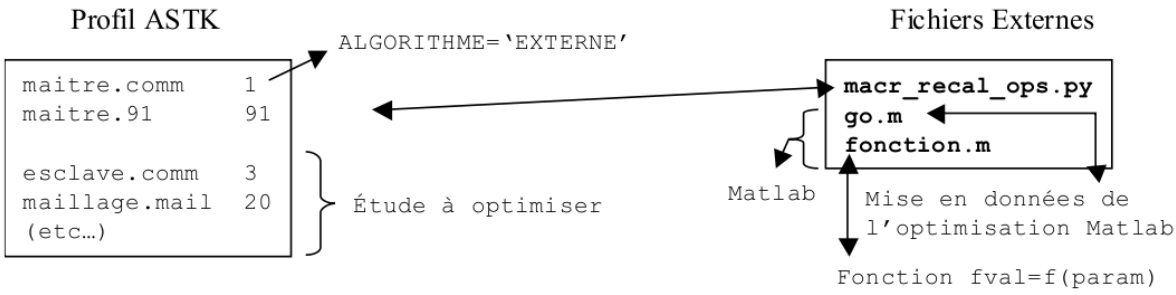



**Figure 3.3-a : MACR_RECAL method EXTERNAL**

Code_Aster is only one black box which takes as starter a textual file containing the list of the values of parameters, carries out the calculation of the functional calculus for this set of parameters, carries out possibly the calculation of the gradients compared to the parameters, and returns to the external algorithm, via a textual file, the value of the functional calculus and possibly of the gradients.

In this operating process, the paragraph 3.1 remain valid. For more details, one returns the user in the paragraph 7 .

## 3.4 Typical case of the retiming of a dynamic model

In the case of the retiming of the parameters of a dynamic model by using experimental data resulting from the dynamic analysis, there exist some characteristics for the implementation. The experimental data are in this case of the Eigen frequencies and the clean vectors (deformed modal) contained in a concept mode_meca resulting from measurement.

The user usually has this concept starting from a software of data acquisition in format " .unv " and it must build a model known as "experimental" in order to exploit it in the Code_Aster environment. This "experimental" model contains inter alia the experimental grid, i.e. the grid of the sensors, coarser than the grid of the digital model, which must be available also for the study of retiming.

Thus the user does not have to provide any more the experimental curves in the master file but it will inform here the names of the concepts which contain them, concepts which will be extracted by a first calculation slave since the external file "unv". The format used to transmit this information to MACR_RECAL is the same one as that for RESU_CALC : a list Python of $N$ lists Python containing the names of the tables and columns containing the experimental answers.

Below into present the diagram of the process of retiming in the case of dynamics:

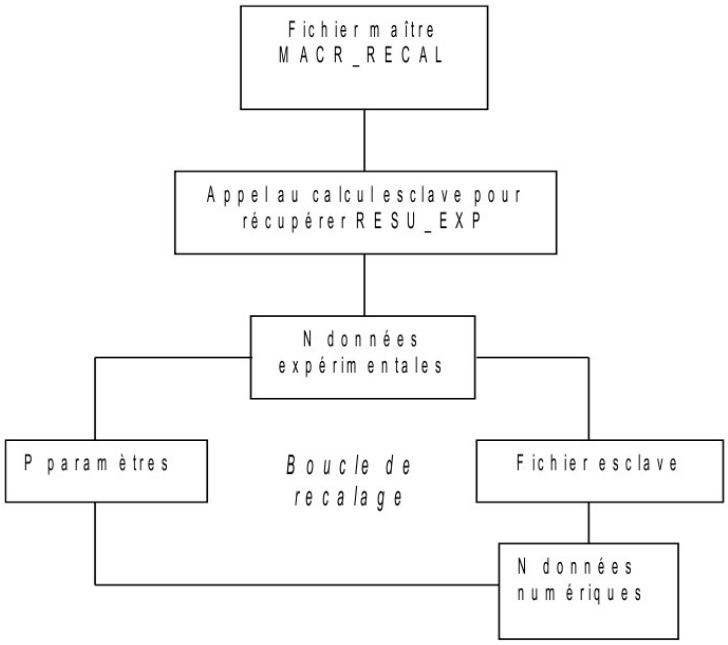



**Figure 3.4-a. Diagram of operation of the procedure of retiming in dynamics**

Two other characteristics relate to the command file of calculation slave:

- it contains also the orders necessary to the construction of the experimental model. The concept mode_meca usable to extract RESU_EXP is provided by LIRE_RESU ;
- The size corresponding to the experimental and digital answers are always contained in tables but more necessarily resulting from POST_RELEVE_T. The answers related to the modal deformations result is of CREA_TABLE (for the criterion of experimental MAC), that is to say of MAC_MODES (for the digital criterion of MAC). It is pointed out that the criterion of MAC [U4.52.15] is used to compare two modal bases, it acts here of that experimental and that digital.

# 4 Operands

## 4.1 Operand UNITE_ESCL

♦ UNITE_ESCL
Logical number of unit of the file slave, allotted in the interface Astk (column UL). The extension of this file can be unspecified.

## 4.2 Operands RESU_EXP, RESU_CALC, FONC_EXP, NOM_FONC_CALC, PARA_X, PARA_Y, WEIGHT

The user has 2 modes of seizure to inform the experimental data and the names of the tables and the columns of the calculated results.
He has the choice between using:

- RESU_EXP and RESU_CALC and possibly LIST_POIDS who require the seizure of lists python,
- the keyword FONC_EXP , NOM_FONC_CALC , PARA_X , PARA_Y , WEIGHT keyword factor CURVE who have the advantage of using concepts Aster.

◊ RESU_EXP

Name of the list Python of $N$ tables numpy containing them $N$ experimental curves. For a static model the list is beforehand defined in the form:

```
resu_exp= [  numpy.array ([   [X0, there0],
                              [X1, there1],
                               …
                              [XN , thereN ]]),
          .....
             numpy.array ([   [U0 , v0 ],
                              [U1 , v1 ],
                             …
                              [UN, vN]    ])
          ]
```

For the retiming of a dynamic model with modal experimental data (clean frequencies and vectors), it will be the name of the list Python $N$ lists Python containing the names of the tables and the columns containing the experimental answers. For example:

```
resu_exp=  [['REPEX1',  'NUM_ORDR',  'FREQ'],  ['REPEX2',  'NUM_ORDR',
'MAC_EXP']]
```

◊ RESU_CALC

Name of the list of NR lists Python containing the names of the tables and the columns containing the digital answers corresponding to experimental measurements on which one will carry out retiming. For example:

```
resu_calc= [['TABLE1', 'INST', 'SIYY'], ['TABLE2', 'INST', 'V1'],…])
```

◊ LIST_POIDS

Name of Numpy table containing them $N$ weight to be affected to $N$ experimental curves. If the keyword is not indicated, then each curve has the same weight. The list is beforehand defined in the form:

```
LIST_POIDS= numpy.array ([p1, p2,…])
```

◊ FONC_EXP

Name of the function defined beforehand by the operator DEFI_FONCTION who corresponds to the experimental data. To establish the link with RESU_EXP , the function provided to this keyword with the iteration $i$ corresponds to $i$ ème digital table of RESU_EXP.

◊ NOM_FONC_CALC

Character string corresponding to the name which one wishes to allot to the function afterwards retiming of the experimental data.

◊ WEIGHT

Value of the weight to be assigned to the experimental curve. So nonwell informed, then the value is 1.

◊ PARA_X

Name of the parameter $X$ calculated function.

◊ PARA_Y

Name of the parameter $Y$ calculated function. For example:

```
COURBE= (
_F (FONC_EXP=fonction1, NOM_FONC_CALC=' TABLE1', PARA_X=' INST', PARA_Y='
SIYY',),
_F (FONC_EXP=fonction2, NOM_FONC_CALC=' TABLE2', PARA_X=' INST', PARA_Y='
V1',),)
```

## 4.3 Operands LIST_PARA, PARA_OPTI, NOM_PARA, VALE_INI, VALE_MIN, VALE_MAX

The user has 2 modes of seizure to inform the names and values of the parameters.
He has the choice between using:

- LISTE_PARA who require the seizure of lists python,
- the keyword NOM_PARA, VALE_INI, VALE_MIN, VALE_MAX keyword factor PARA_OPTI who have the advantage of using concepts Code_Aster (digital values or character strings).

◊ LIST_PARA

Name of the list Python of $P$ lists Python containing the names of the variables, their initial values, their minimal values and their maximum values. This list is beforehand defined in the form:

```
List_para= [ ['PARA1', INI_1, MIN_1, MAX_1],
             ['PARA2', INI_2, MIN_2, MAX_2],
             ….
             ['PARAP', INI_P, MIN_P, MAX_P]]
```

**Caution:**

*It is asked that the names of the variables end in two underlined white (for example: YOUN ).*

**Note:**

*The terminals are not managed by algorithms FMIN, FMINBFGS and FMINNCG.*

◊ NOM_PARA

Name of the parameter. The character string provided to NOM_PARA with the iteration $i$ corresponds to the first element of $i$ ème list python of the list provided to LIST_PARA.

◊ VALE_INI

Initial value of the parameter. The reality provided to NOM_PARA with the iteration $i$ corresponds to the second element of $i$ ème list python of the list provided to LIST_PARA.

◊ VALE_MIN

Minimal value of the parameter. The reality provided to NOM_PARA with the iteration $i$ corresponds to the third element of $i$ ème list python of the list provided to LIST_PARA.

◊ VALE_MAX

Maximum value of the parameter. The reality provided to NOM_PARA with the iteration $i$ corresponds to the fourth element of $i$ ème list python of the list provided to LIST_PARA.

## 4.4 Operand UNITE_RESU

◊ UNITE_RESU
Logical number of unit of the file of result of retiming (evolution of the parameters during iterations, convergence criteria).

## 4.5 Operand ITER_MAXI

◊ ITER_MAXI
Iteration count maximum of retiming.

## 4.6 Operand ITER_FONC_MAXI

◊ ITER_FONC_MAXI
Many maximum evaluations of the functional calculus.

## 4.7 Operand RESI_GLOB_RELA

◊ RESI_GLOB_RELA
Relative total residue of retiming.
This value is disjoined of that well informed for the nonlinear solveurs STAT_NON_LINE and DYNA_NON_LINE.

## 4.8 Operand TOLE_FONC

◊ TOLE_FONC
Criterion of stop of the algorithm of retiming based on the variation of the functional calculus from one iteration to another. This criterion corresponds to the absolute value of the standard of my functional calculus.

## 4.9 Operand TOLE_PARA

◊ TOLE_PARA
Criterion of stop of the algorithm of retiming based on the variation of the parameters from one iteration to another. This criterion corresponds to the standard $L2$ : square root of the sum of the squares of the differences in each parameters.

## 4.10 Operand PARA_DIFF_FINI

◊ PARA_DIFF_FINI
Retiming requires the calculation of the derivative of the answers compared to the parameters.

This calculation is carried out by finished differences. PARA_DIFF_FINIES corresponds to $\alpha$ in the following formula:

$$\frac{\partial f}{\partial x} \approx \frac{f(x+\alpha x)-f(x)}{\alpha x}$$

## 4.11 Operand GRAPH

◊ UNIT
Logical number of unit of the graphs produced during retiming. With each iteration, MACR_RECAL product $N$ display files (of which the format is defined by the keyword PILOT) representing them $N$ experimental and calculated curves.

◊ PILOT
Type of posting of the graphs.
If PILOT = 'INTERACTIVE', xmgrace is open in an interactive way with the graph. If PILOT is worth 'POSTSCRIPT', 'EPS', 'MIF', 'SVG', 'PNM', 'PNG', 'JPEG' or 'Pdf' then xmgrace is used to generate the file of corresponding format, in the logical unit defined by UNIT.

**Caution:**

*Mode INTERACTIVE is not possible that when retiming turns in interactive and not in batch.*

◊ FORMAT
Choice of the software of posting of the curves in interactive mode: xmgrace or gnuplot. The use ofXmgrace is blocking: the window should be closed Xmgrace to continue the execution.

◊ POSTING
Posting of the curves to each iteration or only at the end.

## 4.12 Operand METHOD

◊ METHOD
Method or selected algorithm of optimization:
1) LEVENBERG (defect): algorithm of Levenberg-Marquardt. This algorithm is that recommended in the square problems of standard minimization quadratic least, like problems of retiming of parameters materials.
2) FMIN : Nelder-Mead Simplex algorithm (uses only estimates of the functional calculus).
3) FMINBFGS : method Quasi-Newton (uses the functional calculus and the gradient of the functional calculus).
4) FMINNCG : method Line-search Newton Conjugate Gradient (uses the functional calculus, the gradient of the functional calculus and its hessien).
5) GENETICS: algorithm évolutionnaire based on the mechanism of the selection and the replacement. It is an algorithm which is expensive in time CPU, one advises its use only for one coarse exploration of the space of the parameters within the framework of the hybrid technique of retiming presented in the following point.
6) HYBRID: technique which combines the stochastic one with the determinist - the algorithm évolutionnaire with the algorithm of Levenberg – Marquardt
7) Mode EXTERNAL : this method makes it possible to use an external algorithm of optimization with *Code_Aster*, for example Matlab or Scilab, and to use *Code_Aster* only for the estimate of the functional calculus and possibly of the gradient by finished differences. It is not a keyword of MACR_RECAL because mode EXTERNAL be used directly by the call of the file Macro bibpyt//recal.py

**Concerning the choice of the algorithm of optimization**, it is strongly advised to choose the algorithm by default, **Levenberg-Marquardt**. This one is very often higher than the algorithms FMIN* for problems of minimization of the least type squares, like the retiming of parameters. Indeed, it uses the functional calculus in its vectorial form whereas the other algorithms use a scalar functional calculus, consequently less rich. Moreover, it uses a method of active constraints in order to manage terminals on the parameters, whereas the other algorithms do not manage the terminals.

The other algorithms can nevertheless be useful whenever Levenberg-Marquardt is put in a difficult situation. For example, the algorithm FMIN do not use gradients, whose evaluation can in certain very particular cases generate digital problems (very significant parameters, not enough experimental values, or others). The algorithm FMIN is definitely slower, but will be able to manage to converge (to make a parallel, the problems is similar to that to use the elastic matrix instead of the tangent matrix in STAT_NON_LINE).

For the algorithm of Levenberg-Marquardt, the document [R4.03.06] more precisely described the put mathematical algorithmy concerned.

Algorithms FMIN* were taken again completely of a module Python distributed on Internet (http://pylab.sourceforge.net) under licence LPG by Travis E. Oliphant, in addition principal contributor of the project Python- Scipy and person in charge of the module of optimization of Scipy. The details of algorithmy and implementation can be found on the page http://pylab.sourceforge.net.

Method HYBRID is advised when one has a high degree of uncertainty on the optimal values of the parameters or when the functional calculus presents many local minima. Thus, within the framework of this method, one launches initially a coarse research with the algorithm évolutionnaire, which will

make it possible to avoid the local minima, followed by a refinement of optimization with the algorithm of Levenberg-Marquardt.

## 4.13 Keyword CALCUL_ESCLAVE

### 4.13.1 Operand LAUNCHING

◊ LAUNCHING
Method of launching of the files slaves: inclusion or distribution. The two modes have advantages and disadvantages and the choice of one or other depends mainly on the computing times of the files slaves, as well as theirs compatibility with the Inclusion mode.

### 4.13.2 Operand INCLUSION

◊ INCLUSION
In this mode, the file slave is included. There is thus no waste of time for the generation of a new study, the creation of the temporary repertoire of execution, etc. In counterpart, only a study slave will be able to pass at the same time on the machine. In addition, certain studies slaves (for example those using of the data files included, or a little complex profiles of execution) are not compatible.

**Note:**

> *In the mode INCLUSION, the order Aster INCLUDE donot can be present in the file slave (incompatibility with the supervisor Aster).*
> *The order thus should be replaced:*
> *INCLUDE (UNITE=n)*
> *by the order:*
> *execfile (fort.n)*
> *where N is the logical unit of the file to be included.*

### 4.13.3 Operand DISTRIBUTION, MODE, MEMORY, TIME, CLASS, UNITE_SUIVI

◊ DISTRIBUTION
In this mode, with each iteration of the algorithm of optimization, them $N+1$ calculations slaves (for a retiming of $N$ parameters) are carried out in parallels, batch or interactive, by using the module of calculations distributed ofas_run. Compared to Inclusion mode, each study is slightly longer to be carried out, because a new study should be regenerated *Code_Aster*, to create the temporary files, etc. On the other hand, as the studies are launched in parallels, according to the characteristics of the studies slaves (size and duration), plus the number of parameters is large and more the Distributed mode takes interest on the Inclusion mode.

In distributed mode, additional parameters are available supervise the implementations of calculations slaves:

◊ MODE : INTERACTIVE or BATCH
◊ MEMORY : Mo memory
◊ TIME : time in seconds
◊ CLASS : class of batch, allows to force calculations to use a specific class, for example “distr” on the waiter *Code_Aster*
◊ UNITE_SUIVI : if this keyword is to specify, it definite the logical unit of the file of the profile in which will be stored all the files output of the jobs slaves
◊ NMAX_SIMULT : many calculations slaves launched in parallel in distribution mode (if no value is indicated, the code decide this number automatically)

It is possible to use parallelism MPI for the studies slaves. In this case, it is necessary to launch main calculation with a version MPI and on only one processor (in interactive mode or batch). Characteristics MPI of calculations slaves must be indicated by the keyword following:

- ◊ MPI_NBCPU : many processors MPI for each calculation slaves launched in parallel
- ◊ MPI_NBNOEUD : many nodes for each calculation slave launched in parallel

It should be noted that problems of exploitation can come to disturb the launching of calculations distributed slaves (for example, the classes of batch are badly defined and calculations cannot be carried out). In this case, the error obtained in the .mess of main calculation can be rather sober (an error message of the style “at least one of calculations slaves could not start”). To obtain information additional concerning the module distributed ofas_run, it is necessary to put simultaneously UNITE_SUIVI and INFO=2.

## 4.14 Operands NB_PARENTS and NB_FILS

◊ NB_PARENTS

For the method GENETICS or HYBRID, this operand definite size of the population of parameterized. Initially all the individuals identical and are initialized with the initial values of the parameters provided by the user in LIST_PARA. During optimization the population evolves, the individuals less “adapted” being replaced by others which provided a better value of the functional calculus.

◊ NB_FILS

Represent the rate of replacement of the population. More exactly, the best “relative” (that for which the value of the functional calculus is tiny) has the right to reproduce thus generating NB_FILS “son”. At this time the size of the population is NB_PARENTS+NB_FILS. According to hierarchy of the values of the functional calculus, only the best individuals of all this population are retained and one returns to the initial size: NB_PARENTS.

## 4.15 Operand ECART_TYPE

◊ ECART_TYPE

It is the value of the standard deviation which the user imposes for the quasi-random lotteries of the “sons”. The more one wants to explore the topological space of the parameters, the more it is necessary to increase this value. Corroborated in keeping with the population and at the rate of replacement, this operative makes it possible to control the algorithm évolutionnaire according to the complexity of the model and the degree of uncertainty on the optimal values of the parameters. If one knows few things on the values of the parameters to be readjusted, it are advisable to use a high size of the population, a rate of also high replacement and a large standard deviation. The counterpart will be a very high time CPU.

## 4.16 Operand ITER_ALGO_GENE

◊ ITER_ALGO_GENE

Iteration count maximum for the algorithm évolutionnaire. If the method is used HYBRID, this value (or RESI_ALGO_GENE) will determine the passage to the algorithm of Levenberg-Marquardt.

## 4.17 Operand RESI_ALGO_GENE

◊ RESI_ALGO_GENE

Relative residue of retiming the algorithm évolutionnaire. If the method is used HYBRID, this value (or ITER_ALGO_GENE) will determine the passage to the algorithm of Levenberg-Marquardt.

## 4.18 Operand SEED

◊ SEED

Specified value by the user for seed of the generator of the lotteries in the algorithm évolutionnaire. If one informs a value for this keyword, one force the generator of random numbers present in the algorithm évolutionnaire to always generate the same lotteries, thus there will be a repetition of the solution. Its employment is reserved **only** with the CAS-tests for reasons of follow-up of the not-regression of the code.

## 4.19 Operand DYNAMICS

One informs this operand for the retiming of the parameters of a dynamic model by modal analysis.

◊ MODE_EXP

Name of the concept mode_meca who contains the experimental modal data. This concept is extracted in calculation slave by one LIRE_RESU.

◊ MODE_CALC

Name of the concept mode_meca who contains the digital modal data. This concept is calculated in the file slave by one CALC_MODES.

◊ APPARIEMENT_MANUEL

Choice to display in interactive mode of a chart window which will make it possible to pair the clean modes manually. One thus avoids for example the automatic bad pairing of the clean modes doubles cross.
The capture of screen presented in the Figure 4.19-a illustrate this chart window.

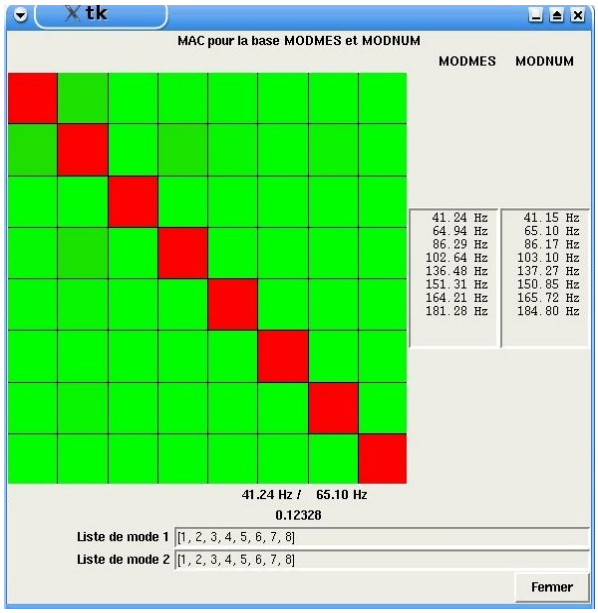



**Figure 4.19-a. Chart window for the manual pairing of MAC**

The user sees thus, with each generation of the algorithm évolutionnaire or with each iteration of the algorithm of Levenberg-Marquardt, the matrix of MAC and it can decide to change pairing into modifying the order of the modes in the lists located in bottom of the window. This window is blocking for the execution of the orders *Code_Aster* thus it should be closed so that the process of retiming continuous. The closing of the window while clicking on the button *To close* allows to recover the new lists of the modes whose order was possibly modified by the user.

## 4.20 Operand GRADIENT

◊ GRADIENT

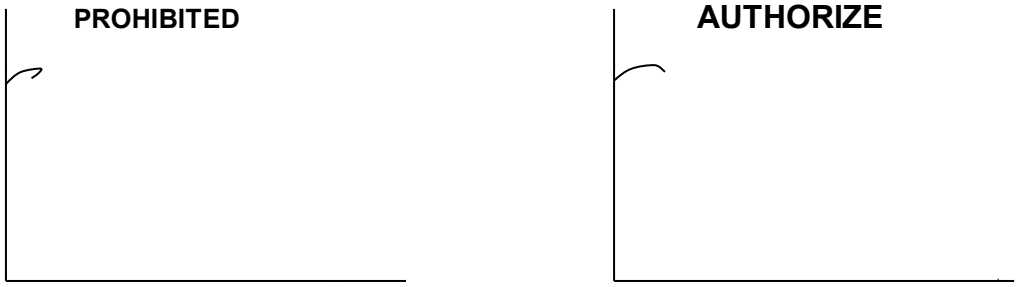
For the methods FMINBFGS, FMINNCG or EXTERNAL, this keyword makes it possible to indicate to *Code_Aster* the way of calculating the gradients (adimensionné or not), or of not calculating them.

Note: for the algorithms which use the gradients, those can be calculated by differences finished automatically by *Code_Aster*, or calculated in the file slave while using, for example calculations of sensitivity (keyword LIST_DERIV).

# 5 Precautions for use

Here a set of advices **essential** with the good use of retiming.

- The experimental curves are defined like tables with two columns: for the X-coordinates and for the ordinates.
- The experimental curves must be functions: to a X-coordinate only one ordinate should correspond. If an experimental curve comprises cycles, (for example forced according to the deformation in load-discharge), it is then necessary to divide this curve parameterized into two curves, expressing on the one hand the X-coordinates, on the other hand the ordinates of the cyclic curve according to the parameter (for example deformation according to time and constraint according to time).




- One must readjust $N$ curves calculated on $N$ experimental curves.
- The first calculated curve will be readjusted on the first experimental curve, the second calculated curve will be readjusted on the second experimental curve, and so on in the order indicated for the operands RESU_EXP and RESU_CALC.
- Sizes calculated well informed under the operand RESU_CALC must be resulting from POST_RELEVE_T (except for dynamics where one can have tables resulting from CREA_TABLE and MAC_MODES)
- The parameters of retiming must be declared in block at the beginning of the command file slave. For example:

```
BEGINNING ();
DSDE = 200. ;
YOUN = 8.E4;
SIGY = 10. ;
......
```

- The initial values of the parameters of retiming are those well informed for the operand LIST_PARA and not those present in the file slave of the user.
- With each iteration of retiming, the calculations defined in the file slave must converge. Within the framework of retiming of nonlinear calculations, it is thus strongly recommended to use the automatic cutting of the step of time.
- Within the framework of retiming of nonlinear calculations with automatic cutting of the step of time, it is **essential** to define a list of filing under the operand LIST_ARCH.
- Retiming is a powerful means to obtain values of parameters starting from tests. It is however not miraculous: curves **experimental** must sufficient contain information to identify the parameters. It is for example impossible to identify elastoplastic parameters with a test remaining in the elastic range. The experimental tests must thus excite the parameters to be identified.
- In same logic, it is desirable that the curves **experimental** contain points of number sufficient for describing the action of the parameters well to be identified.
- Lastly, in the case of the use of several experimental curves, the fact that they have the same number of points balances information that they bring.

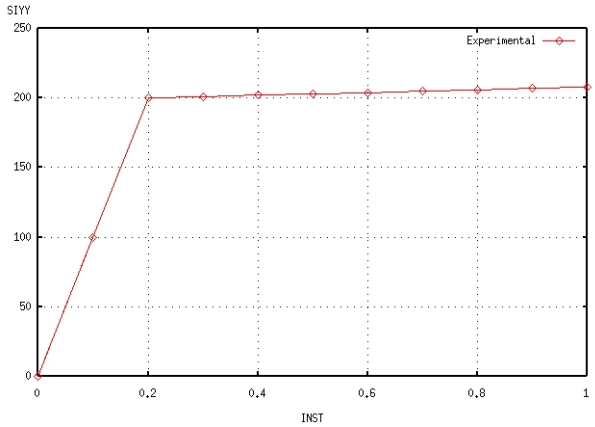
# 6 Examples of use

## 6.1 Identification of the parameters of a law of elastoplastic behaviour on a tensile test

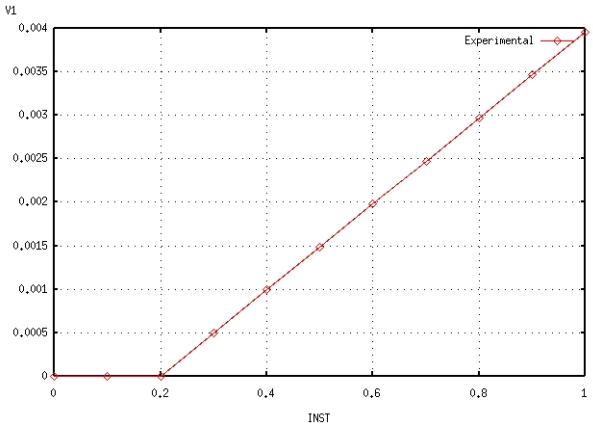
This example is treated by test ZZZZ159A [V1.01.159].

### 6.1.1 Position of the problem

One has the results of a tensile test. It is the evolution of the constraint $\sigma_{yy}$ in the course of time as well as evolution of the cumulated plastic deformation $p$ in the course of time.



Constraint SIYY



Cumulated plastic deformation

One wishes to readjust on these tests the Young modulus, the elastic limit and the slope of work hardening of an elastoplastic law of behaviour to linear isotropic work hardening.

### 6.1.2 Setting in data

#### 6.1.2.1 Experiment

One starts first of all by defining our test results. They consist of two curves which one defines as follows.

```
exper1=DEFI_FONCTION (NOM_PARA=' INST',
                      NOM_RESU=' SIYY',
                      VALE= (0.00000E+00  , 0.00000E+00,
                             5.00000E-02  , 5.00000E+01,
                             …
                             9.50000E-01  , 2.07500E+02,
                             1.00000E+00  , 2.08000E+02),)

exper2=DEFI_FONCTION (NOM_PARA=' INST',
                      NOM_RESU=' V1',
                      VALE= (0.00000E+00  , 0.00000E+00,
                             5.00000E-02  , 0.00000E+00,
                             …
                             9.50000E-01  , 3.71250E-03,
                             1.00000E+00  , 3.96000E-03),)
```

exper1 and exper2 are thus functions of Code_Aster, which represent the constraint respectively $\sigma_{yy}$ and cumulated plastic deformation $p$.

#### 6.1.2.2 Calculation

One writes then the command file *Code_Aster* slave modelling this tensile test where will appear our 3 parameters as well as the two curves to be readjusted.

```
BEGINNING ();
# ASSIGNMENT OF THE VALUES OF THE PARAMETERS TO BE READJUSTED
# THE VALUES INDICATED HERE ARE OF NO IMPORTANCE
# ALONE COUNT THE VALUES INDICATED IN THE MASTER FILE
DSDE = 200. ;

YOUN = 8.E4;

SIGY = 1. ;

ACIER=DEFI_MATERIAU (ECRO_LINE=_F (D_SIGM_EPSI=DSDE,
                                   SY=SIGY,),
                     ELAS=_F (NU=0.3,
                              E=YOUN,),);
.....

U=SIMU_POINT_MAT (COMPORTEMENT=_F (RELATION=' VMIS_ISOT_LINE'),
                  MATER=ACIER,
                  INCREMENT=_F (LIST_INST=INSTANTS,),
                  NEWTON=_F (REAC_ITER=1),
                  EPSI_IMPOSE=_F (EPYY=epyy,),
                  );

# EXTRACTION OF ANSWER SIGMAYY (T)
REPONSE1=CALC_TABLE (TABLE = U,
                ACTION =_F (OPERATION=' EXTR',
                           NOM_PARA= ('INST', 'SIYY')))

# EXTRACTION OF ANSWER EPSP (T)
REPONSE2=CALC_TABLE (TABLE = U,
                ACTION =_F (OPERATION=' EXTR',
                           NOM_PARA= ('INST', 'V1')))

END ();
```

#### 6.1.2.3 MACR_RECAL

It is necessary for us now to define in the master file the initial values and the beaches of variations of our parameters. One wishes:

| | | | | | |
|---|---|---|---|---|---|
| 1.E5 | < | Initial Young modulus = 1.E5 | < | 5.E5 |
| 5. | < | Initial elastic limit = 30. | < | 500 |
| 1.E3 | < | Module of initial work hardening = 1.E3 | < | 1.E4 |

Finally it is also necessary to define in the master file the sizes to be extracted from the command file slave. We wish on the one hand to extract the column INST and the column SIYY table REPONSE1 and in addition the column INST and the column V1 table REPONSE2. We write it:

Here how we inform this information in the body of MACR_RECAL :

```
RESU2=MACR_RECAL(
       UNITE_ESCL = 3,
       PARA_OPTI= ( _F (   NOM_PARA=' YOUN ',  VALE_INI=100000.0,
                           VALE_MIN=50000.0,   VALE_MAX=500000.0),
                    _F (   NOM_PARA=' DSDE ',  VALE_INI=1000.,
                           VALE_MIN=500.,      VALE_MAX=10000.),
                    _F (   NOM_PARA=' SIGY ',  VALE_INI=30.,
                           VALE_MIN=5.,        VALE_MAX=500.),),
       COURBE= ( _F (   FONC_EXP=exper1, NOM_FONC_CALC=' REPONSE1',
                           PARA_X=' INST',  PARA_Y=' SIYY'),
                 _F (   FONC_EXP=exper2, NOM_FONC_CALC=' REPONSE2',
                        PARA_X=' INST',  PARA_Y=' V1'),),
       )
```

#### 6.1.2.4 Alternative setting in data (lists Python)

One can use the setting in data containing list Python and of numpy object. The master file then will be written:

```
experience= [   numpy.array ([[0.00000E+00  , 0.00000E+00],
                              [5.00000E-02  , 5.00000E+01],
                              .........
                              [9.50000E-01  , 2.07500E+02],
                              [1.00000E+00  , 2.08000E+02]]),
                numpy.array ([[0.00000E+00  , 0.00000E+00],
                              [5.00000E-02  , 0.00000E+00],
                              .........
                              [9.50000E-01  , 3.71250E-03],
                              [1.00000E+00  , 3.96000E-03]])]

parameters = [['YOUN', 100000. , 50000. , 500000.], ['DSDE', 1000. , 500. ,
10000.],
              ['SIGY', 30. , 5. , 500.]]

calculation = [['REPONSE1', 'INST', 'SIYY'], ['REPONSE2', 'INST', 'V1']]
```

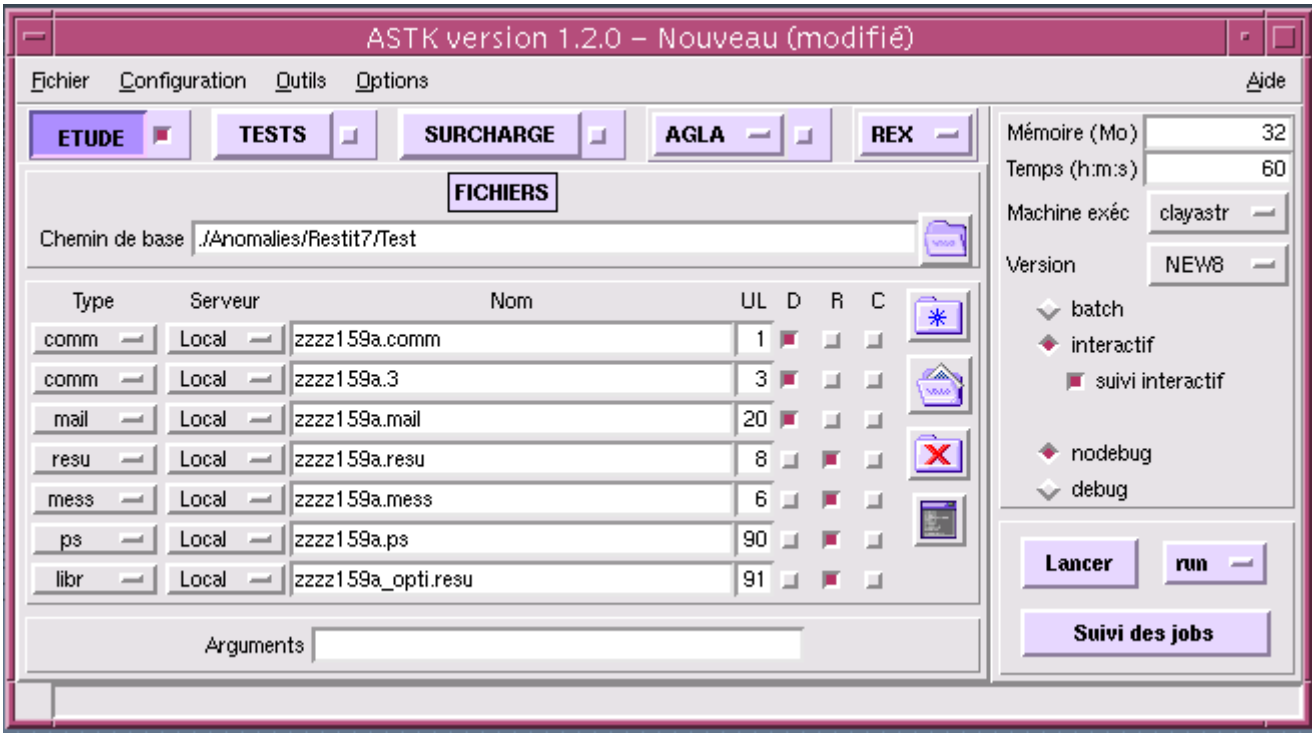
where: `experiment` is the name of a list Python (definite between hooks) of 2 tables numpy,
`parameters` is a list of lists Python containing initial values and beaches of variations of our parameters and `calculation` sizes to be extracted from the command file slave.

The body of the order is written then:

```
RESU=MACR_RECAL (
         UNITE_ESCL       =3,
         RESU_EXP         =experience,
         LIST_PARA        =parametres,
         RESU_CALC        =calcul,);
```

#### 6.1.2.5 Astk

One defines finally the profile of study according to:




### 6.1.3 Results

Once the study carried out, the file of result of ZZZZ159_opti.resu retiming contains following information:

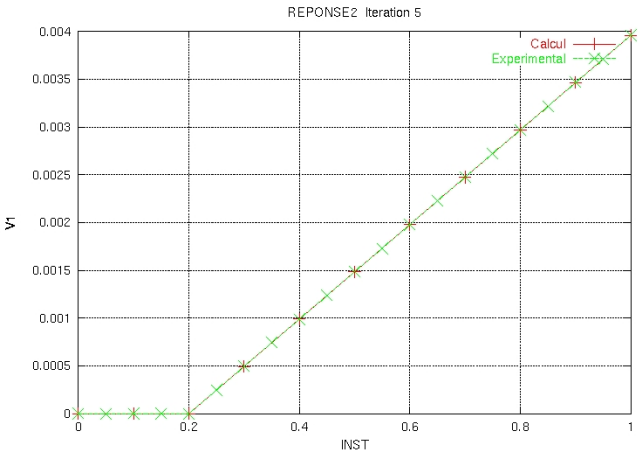
```
Calculation of the sensitivity compared to = YOUN DSDE SIGY
=========================================================
Iteration 0 =

=> Fonctionnelle = 1.0
=> Résidu        = 1.0
=> Paramètres    =
         YOUN = 100000.0
         DSDE = 1000.0
         SIGY = 30.0
=========================================================


Calculation of the sensitivity compared to = YOUN DSDE SIGY
=========================================================
Iteration 1 =

=> Fonctionnelle = 0.259742161795
=> Résidu        = 0.30865397471
=> Paramètres    =
         YOUN = 300857.888503
         DSDE = 9135.12770111
         SIGY = 152.548047532
=========================================================


Calculation of the sensitivity compared to = YOUN DSDE SIGY
=========================================================
```

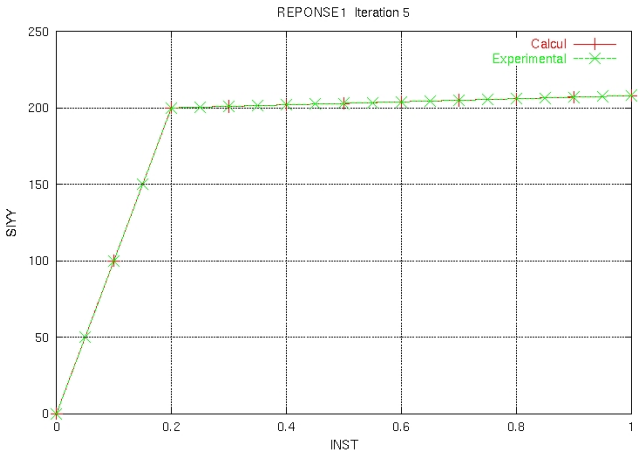
```
Iteration 2 =

=> Fonctionnelle = 0.0757636994765
=> Résidu        = 0.473053125246
=> Paramètres    =
         YOUN = 157723.378846
         DSDE = 2022.7431335
         SIGY = 213.155325073
============================================================


Calculation of the sensitivity compared to = YOUN DSDE SIGY
============================================================
Iteration 3 =

=> Fonctionnelle = 0.00190706595529
=> Résidu        = 0.0520849911718
=> Paramètres    =
         YOUN = 192302.166747
         DSDE = 895.845518907
         SIGY = 203.753909707
============================================================


Calculation of the sensitivity compared to = YOUN DSDE SIGY
============================================================
Iteration 4 =

=> Fonctionnelle = 2.70165453323e-06
=> Résidu        = 0.00172172540305
=> Paramètres    =
         YOUN = 199801.572817
         DSDE = 1928.08902726
         SIGY = 200.274590793
============================================================

Calculation of the sensitivity compared to = YOUN DSDE SIGY
============================================================
Iteration 5 =

=> Fonctionnelle = 2.65431115925e-12
=> Résidu        = 1.83121468206e-06
=> Paramètres    =
         YOUN = 199999.975047
         DSDE = 1999.86955101
         SIGY = 200.000462987
============================================================

============================================================
                  CONVERGENCE REACHED
============================================================

Eigenvalues of Hessien:
[  7.17223479e+00   3.67264061e-01   6.25194340e-04]

Associated clean vectors:
[[0.98093218 -0.00549396 -0.19427266]
 [- 0.19418112 -0.06940835 -0.97850712]
 [0.00810827 -0.9975732   0.0691517]]
```

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

```
One can deduce from it that:

The following combinations of parameters are dominating for your
calculation:

   1) +9.8E-01 * YOUN -1.9E-01 * DSDE
      associated with the eigenvalue 7.2E+00


The following combinations of parameters are insensitive for your
calculation:

   1) -1.9E-01 * YOUN -9.8E-01 * DSDE
      associated with the eigenvalue 6.3E-04
```

And the file POSTSCRIPT ZZZZ159.ps contains:



## 6.2 Identification of the parameters of a dynamic model by exploiting the sensitivity of the clean modes

This example is treated by test SDLS121A [V2.03.121].
One wishes to readjust the thickness of a plate and the value of a discrete mass which is located above by using experimental measurements of clean modes.

### 6.2.1 Setting in data

In the master file one informs initially the names of the tables which will contain at the same time the results calculated thus that the digital results:

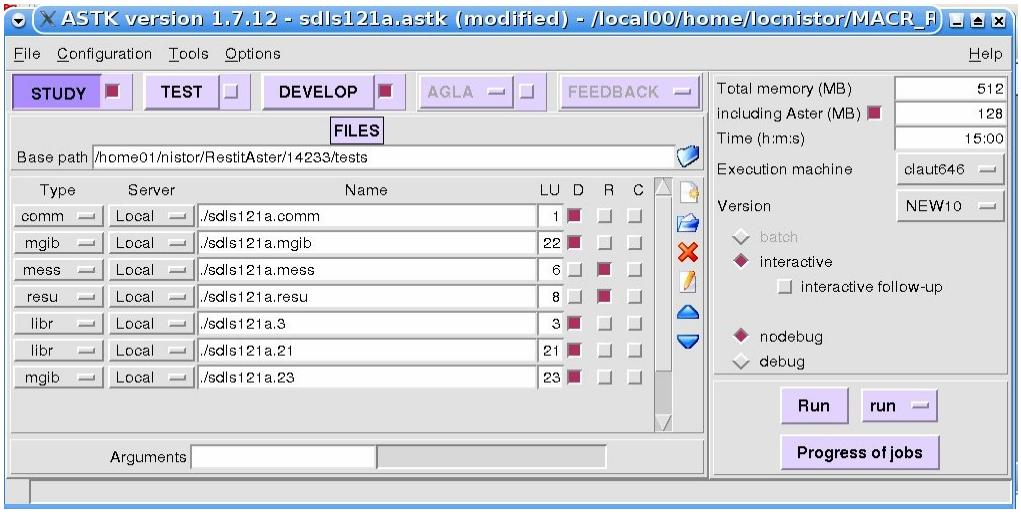
```
calculation = [  ['REPONSE1', 'NUMÉRIQUE_ORDRE',  'FREQ'],
                        ['REPONSE2', 'NUMÉRIQUE_ORDRE',  'MAC']]

experience= [['REPEXP1',  'NUMÉRIQUE_ORDRE',  'FREQ'],
                     ['REPEXP2',  'NUMÉRIQUE_ORDRE',  'MAC_EXP']]
```

It is important to note here that, for the calculated results, REPONSES2 is given by the diagonal of the matrix of MAC resulting from the order MAC_MODES as one will see a little later in this paragraph. The name of the parameter 'MAC'is thus a reserved name, thus chosen in FORTRAN, and thus it cannot be used any more thereafter (for the experimental answer corresponding one chose 'MAC_EXP')

Options retained in the order MACR_RECAL, always in the master file are:

```
RESU=MACR_RECAL(
                    UNITE_ESCL       =3,
                    RESU_EXP         =experience,
                    LIST_PARA        =parametres,
                    RESU_CALC        =calcul,
                    WEIGHT           =poids,
                    METHOD       = ' HYBRIDE',
                    ITER_FONC_MAXI   =500,
                    NB_PARENTS       =10,
                    NB_FILS          =5,
                    ECART_TYPE       =10. ,
                    ITER_ALGO_GENE   =2,
                    DYNAMICS =_F (
                                MODE_EXP=' MODMES',
                                MODE_CALC=' MODNUM',
                                APPARIEMENT_MANUEL=' NON',),);
```

One thus chose to launch retiming by using the HYBRID method with two iterations for the algorithm évolutionnaire. Finally one defines the following profile for the study:




### 6.2.2 Calculation slave

The principal characteristic of calculation slave is the presence in the command file (unit 3) of two models: the experimental model and that digital. Below one presents, for this CAS-test, the command file slave with the explanations necessary.

- reading of the experimental grid (grid of the sensors):

```
PRE_GIBI (UNITE_GIBI=23);

MAILEXP1=LIRE_MAILLAGE ();
```

- creation of a group of nodes which will be used later to define elements discrete of mass:

```
MAILEXP1=DEFI_GROUP (  reuse=MAILEXP1,
                       MAILLAGE=MAILEXP1,
                       CREA_GROUP_NO=_F (
                       NOM=' NO_MA',
                       OPTION=' ENV_SPHERE',
                       POINT= (2.0, 3.0),
                       RAYON=0.1,
                       PRECISION=0.1,
                    ),);
```

- creation of the group of nodes to define the boundary conditions:

```
MAILEXP1=DEFI_GROUP (  reuse =MAILEXP1,
                       MAILLAGE=MAILEXP1,
                       CREA_GROUP_NO=_F (  GROUP_MA=' BORDS',
                                           NOM=' BORDS',),);
```

- creation of elements POI1 to introduce the discrete mass:

```
MAILEXP2= CREA_MAILLAGE (  MAILLAGE=MAILEXP1,
                           CREA_POI1 = (_F (   NOM_GROUP_MA = 'FARMHOUSE',
                                               GROUP_NO = 'NO_MA'),),)
```

- assignment of the experimental model:

```
MODEXP=AFFE_MODELE (    MAILLAGE=MAILEXP2,
                     AFFE= (_F (   GROUP_MA = 'TOUT_ELT',
```

```
                                    PHENOMENE=' MECANIQUE',
                                    MODELISATION=' DST',),
                          _F (   GROUP_MA = 'FARMHOUSE',
                              PHENOMENE=' MECANIQUE',
                              MODELISATION=' DIS_T',),),);
```

- definition of the elementary characteristics, material, etc.

```
CAREXP=AFFE_CARA_ELEM (...);
ACIER=DEFI_MATERIAU (...);
MATEX=AFFE_MATERIAU (...);
```

- elementary calculations and assemblies of the matrices.

```
KELEXP=CALC_MATR_ELEM (OPTION=' RIGI_MECA',
                  …);
MELEXP=CALC_MATR_ELEM (OPTION=' MASS_MECA',
                  …);
NUMEXP=NUMÉRIQUE_DDL (...);

KASSEXP=ASSE_MATRICE (...);
MASSEXP=ASSE_MATRICE (...);
```

- creation of sd_mode_meca with the experimental clean modes:

```
MODMES=LIRE_RESU (TYPE_RESU=' MODE_MECA',
           FORMAT=' IDEAS',
           MODELE=MODEXP,
           UNITE=21,
           NOM_CHAM=' DEPL',
           MATR_RIGI =KASSEXP,
           MATR_MASS =MASSEXP,
           FORMAT_IDEAS=_F (   NOM_CHAM=' DEPL',
                               NUME_DATASET=55,
                               RECORD_6= (1,2,3,8,2,6,),
                               POSI_ORDRE= (7.4,),
                               POSI_NUME_MODE= (7.4),
                               POSI_FREQ= (8.1,),
                               POSI_MASS_GENE= (8.2),
                               POSI_AMOR_GENE= (8.3),
                               NOM_CMP= ('DX', 'DY', 'DZ', 'DRX', 'DRY MARTINI',
'DRZ'),),
           TOUT_ORDRE=' OUI',);
```

And one continues with the digital model, the same stages. Parameterized to readjust are EP (the thickness) and MP (mass):

```
EP__=0.5
MP__=50000.

PRE_GIBI (UNITE_GIBI=22);

…
…
…

#nombre of frequencies
NF=8
```

```
MODES=CALC_MODES (MATR_RIGI=M_AS_RIG,
                  MATR_MASS=M_AS_MAS,
                  OPTION=' PLUS_PETITE',
                         CALC_FREQ=_F (   NMAX_FREQ=NF),
                         SOLVEUR_MODAL=_F (METHODE=' SORENSEN'),
                         VERI_MODE=_F (STOP_ERREUR=' NON'),
                        );
```

The experimental grid is increasingly coarser than the grid of the digital model thus it is necessary to project the digital result on the experimental grid:

```
MODNUM = PROJ_CHAMP (  RESULTAT=MODES,
                       MODELE_1=MODEL,
                       MODELE_2=MODEXP,
                       NUME_DDL=NUMEXP)
```

One recovers the table of the experimental frequencies

```
REPEXP1=RECU_TABLE (   CO=MODMES,
                     NOM_PARA=' FREQ');
```

One builds the table of experimental MAC - makes of them ideal MAC which is 1.0

```
liste_mac= []
for I in arranges (NF):
       liste_mac.append (1.0)

REPEXP2=CRÉA_TABLE (LISTE= (  _F (PARA=' NUMÉRIQUE_ORDRE', LISTE_I=range (1,
NF+1),),
                              _F (PARA=' MAC_EXP', LISTE_R=liste_mac,),),);
```

And finally tables with the calculated answers:

```
REPONSE1=RECU_TABLE (  CO=MODES,
                       NOM_PARA=' FREQ');

REPONSE2=MAC_MODES (   BASE_1=MODNUM,
                     BASE_2=MODMES,);
```

The results (readjusted values of the parameters) are then calculated in a similar way with the classical case of retiming presented in the preceding paragraph.

## 6.3 Parametric identification of conservative system by exploiting the orthogonality of the clean modes

This example is treated by modeling D of the case test SDLS121 [V2.03.121].
One wishes to readjust the thickness of a plate and the value of a discrete mass which is located above by using experimental measurements of clean modes.

From the modal deformations raised at the points of observation, O N carries out an expansion on the digital model support. One tries thereafter to find the parameters of the model such as the matrices of mass and of stiffness generalized relating to in experiments identified modes are diagonal and that the difference between the measured own pulsation and the calculated own pulsation either minimal.

### 6.3.1 Setting in data

The technique, here illustrated, exploits the wide modal deformations identified on the digital model support. This modal deformation can be obtained using an independent calculation by depositing the deformations extended in a file which will be read thereafter by the file slave. For practical reasons, one carries out the expansion in the pilot file. It is easier thus to recover the model support which will be used in the file slave.

In the master file one informs at the same time the target values (experimental values) and the names of the tables which contain the results resulting from calculation digital carried out in the file slave.

The various stages are pointed out hereafter.

- Creation of an experimental model in order to be able to define the localization of the points of measurement. Nodes of the grid can be connected to each other by discrete elements to facilitate the visualization of measured information. The mechanical characteristics that one assigns to this model do not influence on the results, they make it possible the model to accommodate the identified clean modes
- Reading of the identified clean modes
- Creation of the digital model support for the expansion of measurement
- Definition of the base of expansion (here, a modal base)
- Calculation of the coordinates of the modes identified in the base of expansion
- Expansion of the information measured on the model support
- Writing of the extend modes in a file in order to be able to exploit it in calculation slave
- Definition of the target values

The target values are deposited in two lists. The first list contains the extra-diagonal terms which should be worthless and the second list contains the identified Eigen frequencies.

- Definition of the parameters of retiming
- Launching of the procedure of optimization

### 6.3.2 Calculation slave

Calculation slave creates the tables which will contain the computed values starting from the parameters of entry. The various stages of calculation are described hereafter:

- Creation of the model support (even classification of ddl that which was used for the expansion of measurement)

- Reading of the wide modal deformations
- Calculation of the matrices of mass and stiffness generalisees via MAC_MODES
- Creation of the tables which contain the extradiagonnaux terms of the generalized matrices
- Creation of the table which contains the Eigen frequencies estimated using the generalized matrices

## 6.4 Iparametric dentification of dissipative system by exploiting the relations of standard of the modal deformations

This example is treated by modeling E of the case test SDLS121 [V2.03.121].

One wishes to readjust the thickness of a plate, the value of a specific mass on the plate and the discrete depreciation distributed on the plate.

From the complex modal deformations raised at the points of observation, O N carries out an expansion on the digital model support. One tries thereafter to find the parameters of the model so that in experiments identified clean modes check the relations of standard associated with the digital model. One focuses oneself only on the diagonal terms of the matrix of standardisation.

### 6.4.1 Setting in data

The setting in data resembles that presented in the preceding chapter (6.3). The difference is at the level of the expansion of the modal deformation. The modal deformations are complex here. One carries out the expansion by treating the real part and the imaginary part independently. The deformation complexes wide is obtained after assembly of the real deformation and the imaginary deformation wide.

The various stages are described hereafter:

- Creation of the experimental model
- Reading of the identified clean modes (experimental)
- Expansion of measurement
- Definition of the target values (the target values are worthless vectors here)
- Definition of the parameters of retiming
- Launching of the procedure of optimization

### 6.4.2 Calculation slave

Calculation slave creates the tables which will contain the diagonal terms of the matrix of standardisation.

The various stages of calculation are described hereafter:

- Creation of the model support (even classification of ddl that which was used for the expansion of measurement)
- Reading of the wide modal deformations

- Reading of the Eigen frequencies and reduced depreciation
- Calculation of the products stamps * vector by PROD_MATR_CHAM
- creation of the table which contains the terms relating to damping and the mass: $\frac{y_\nu^T B \bar{y}_\nu + 2\Re(s_\nu) y_\nu^T M \bar{y}_\nu}{|n_\nu|} = z_{1\nu}$
- creation of the table which contains the terms relating to rigidity and the mass: $\frac{s_\nu \bar{s}_\nu y_\nu^T M \bar{y}_\nu - y_\nu^T K \bar{y}_\nu}{|n_\nu s_\nu|} = z_{2\nu}$
- creation of the table which contains the terms relating to the mass, rigidity and damping: $\frac{s_\nu^2 y_\nu^T M y_\nu + s_\nu y_\nu^T B y_\nu + y_\nu^T K y_\nu}{n_\nu s_\nu} = z_{3\nu}$

# 7 Use of the EXTERNAL mode

## 7.1 Warning

We draw attention to the fact that this operating process is to be held for a advanced use. Most case should be treated with the algorithms provided in the order MACR_RECAL, and in particular the Levenberg-Marquardt algorithm, which is adapted the most to the problems of retiming of parameters.

This operating process requires an external software with *Code_Aster* to carry out optimization (code Python, Matlab, Scilab, standard software black box, etc). Moreover, it is necessary to be some competent Python.

**Lastly, the use of the EXTERNAL mode leaves perimeter AQ of Aster**, and should not be used for studies IPS.

## 7.2 Methodology

### 7.2.1 Principle

In this mode of use, *Code_Aster* is only used for the evaluation of the functional calculus for a software of optimization which is completely external with *Code_Aster*.

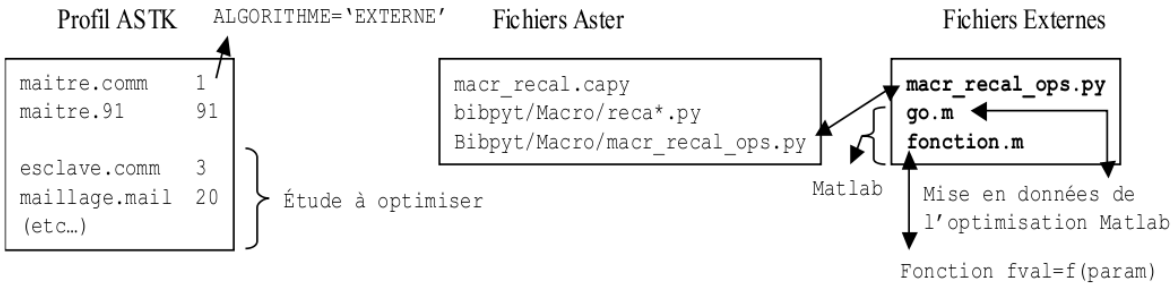



**Figure 7.2.1-a: MACR_RECAL "External algorithm"**

Generally, the software of optimization requires that the user write a procedure for the calculation of the functional calculus: F = F (param).

If the software authorizes the read/write of files and the external code execution (order "system" or others), then it is potentially usable with MACR_RECAL. One will encapsulate the call to the procedure Python **recal.py (in the past MACR_RECAL_ops.py)** as well as the writing of the file of the parameters and the second reading of the file of the value of the functional calculus in the routine of calculation of F.

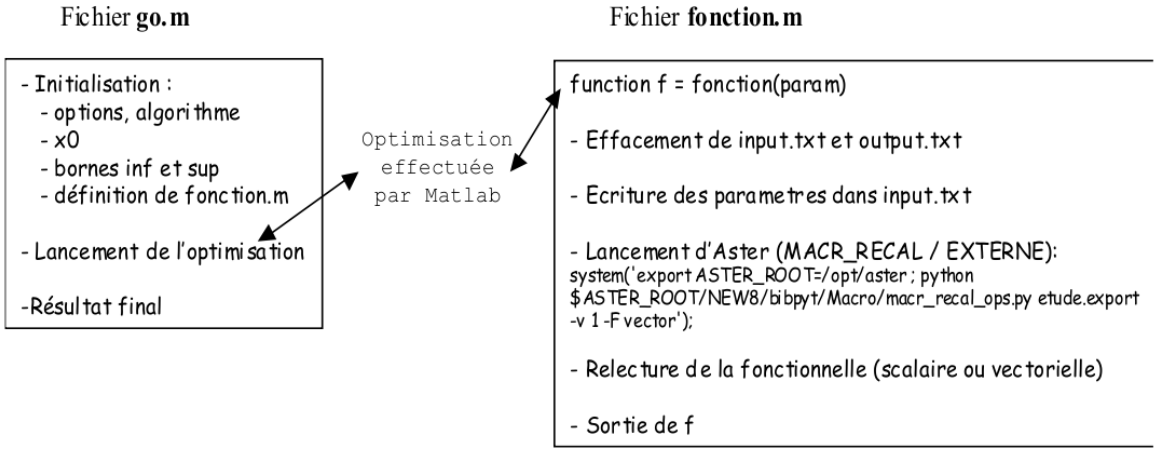



**Figure 7.2.1-b: MACR_RECAL "Principle of the EXTERNAL mode", example Matlab**

The software of optimization launches *Code_Aster* with each evaluation of the functional calculus. The routine of evaluation of the functional calculus passes by textual files for envoyer with *Code_Aster* parameters to be used and to recover the value of the functional calculus. The routine Python recal.py is thus appealable independently of Aster and establishes the link between the software of optimization and *Code_Aster* (recovery of the file of the parameters, fitness of the file slave, launching of Aster, recovery of the functional calculus).

In practice, it is necessary to have a profile Astk classic of the study slave (and not the profile of L one use of MACR_RECAL) . This profile must be completely functional.

Under this mode EXTERNAL, the operating process is the following: the routine recal.py gives the responsibility itself to recover the parameters since the command line or a file input.txt, will replace these parameters in the profile slave, then will launch the execution of the study slave for new the parameters, and finally will return in a textual file output.txt the values of the functional calculus.

To date, this operating process was tested with several software of optimization ( *toolbox* of optimization of Matlab, *toolbox* Tomlab for Matlab, and optimization modulates it of Python- Scipy) but it is possible to use about any software as from the moment when this software authorizes the external execution of a script. One thinks in particular of Zopt (Zebulon), SiDoLo.

## 7.2.2 Use of the external file of launching recal.py

The file recal.py can be carried out in an autonomous way with some optional parameters on the command line:

```
Use: /aster/NEW10/bibpyt/Macro/recal.py fichier_export [options]

Options:
  - H, --help            show this help message and exit
  --input=INPUT         Chain of text containing the parameters
  --input_step=INPUT_STEP
                        Chain of text containing the steps of discretization
                        finished differences
```

```
  --input_file=INPUT_FILE
                        File containing the parameters
  --input_step_file=INPUT_STEP_FILE
                        File containing the steps of discretization of
                        finished differences
  --output=OUTPUT       file containing the functional calculus
  --output_grad=OUTPUT_GRAD
                        file containing the gradient
  --aster_root=ASTER_ROOT
                        Chemin d' installation of Aster
  --as_run=as_run       Way towards as_run
  --resudir=RESUDIR     Way by default of the temporary executions of Aster
  --noclean             Erase temporary Code_Aster execution directory
  --info=INFO           level of message (0, [1], 2)
  --sources_root=SOURCES_ROOT
                        Way by default of the overloads Python
  --objective=OBJECTIVE
                        Functional calculus ([fcalc]/[error])
  --objective_type=OBJECTIVE_TYPE
                        type of the functional calculus (float/[vector])
  --gradient_type=GRADIENT_TYPE
                              calculation of the gradient by Code_Aster ([No]
/normal/adim)
  --mr_parameters=MR_PARAMETERS
                        File of parameters of MACR_RECAL : parameters,
                        calculation, experiment
  --study_parameters=STUDY_PARAMETERS
                        File of parameter of the study: export
  --parameters=PARAMETERS
                        File of parameters
```

This procedure needs:

1. file .export of the study `Astk` defined previously (the standard study of `MACR_RECAL`)
2. of a textual file containing the list of the parameters

From these two files, it launches a calculation *Code_Aster* for the specified parameters (or two calculations *Code_Aster*, to see higher) and generates a textual file containing only the value of the functional calculus (scalar or vectorial).

This procedure can be launched without argument. It then will seek **in the current directory** files by default:

1. if there is one file .export in the current directory, this one will be used (in the contrary case the procedure stops in error)
2. the input file by default will be sought under the name “input.txt”
3. the output file will be generated with the name “output.txt”

Arguments – I (--input) and - O (--output) make it possible to specify the files.

The argument – G (--output_grad): allows to specify the textual file in which the gradient will be written, if this one is required (argument - G).

Other arguments are available:

1. --information: defined the level of message; 0=muet, 1 or 2;
2. --objective: allows to specify if the functional calculus must be turned over in the vectorial or scalar form;
3. --gradient: allows to specify if it is not wanted that Aster calculates and returns the gradient (No), and if the gradients are wanted, to specify if one wants them adimensionné or not;

By default, a file *Code_Aster*`.output` east creates in the repertoire of execution of the procedure, making it possible to have access to the output of the last calculation *Code_Aster*.

Lastly, it is necessary to configure the variable of environment ASTER_ROOT (the basic way of Aster) to be able to launch in Python the file recal.py or aors to use the argument - aster_root.

The good performance of the procedure recal.py can be checked manually, by generating an input file with values of parameters and while launching the procedure manually. Example on the machine *Code_Aster* :

```
Cd repertoire
export ASTER_ROOT=/aster
echo "10. , 20. , 30. " >input.txt
Python $ASTER_ROOT/NEW10/bibpyt/Macro/recal.py etude.export - - info=2
cat ouput.txt
```

Calculation *Code_Aster* should launch out and finish suitably. An output.txt file must be generated in the current directory.

The cas-test zzzz159f illustrate the external use of MACR_RECAL : in this test *Code_Aster*, one defines a function F and one recalls once *Code_Aster* to simulate an iteration of retiming.

## 7.3 Example: module of optimization of Matlab©

If the principle is taken again and that one applies it to an algorithm of optimization which would be written under Matlab, one must write two files go.m and fonction.m who carry out the actions described below:

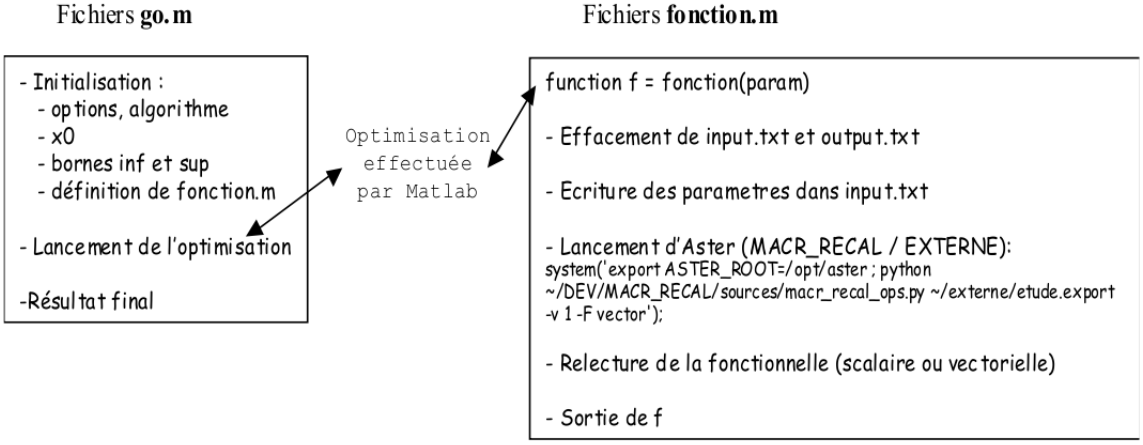



**Figure 7.3-a: MACR_RECAL "Principle of the EXTERNAL mode" applied to Matlab©**

The following example uses the module of optimization of Matlab and carries out calculations *Code_Aster* on the local waiter.

**File go.m** (initialization and launching of optimization)

```
clear all;
long format;

system ('rm - F fort.91');

options = optimset ('Display', 'iter', 'LevenbergMarquardt', 'one',
'TolFun', 1e-8, 'MaxFunEvals', 1000, 'MaxIter', 100);

% Params:  DSDE, SIGY,   YOUN
```

```
x0 = [  1000. ,    30. ,  100000. ];
lb = [   500. ,     5. ,   50000. ];
ub = [10000. ,   500. ,  500000. ];

% vector
[X, resnorm, residual, exitflag, output, lambda, jacobian] = lsqnonlin
(@fonction, x0, lb, ub, options)

% scalar
% [X, fval, exitflag, output] = fmincon (@fonction, x0, [], [], [], [],
lb, ub, [], options)
```

**File fonction.m** (calculation of F (param)) /Linux version

```
function F = function (X)

system ('rm - F input.txt');
system ('rm - F output.txt');

% Writing of the parameters
dlmwrite ('input.txt', X, 'precision', '%.20f');

% Local
iret = system ('exporT ASTER_ROOT=/opt/aster; Python $ASTER_ROOT/NEW10/bibpyt/Macro/recal.py
etude.export --objective_type=vector');

F = dlmread ('output.txt', ',', 0.0);   % reads again a scalar or a vector
```

In `go.m`, two examples of algorithms are given. The algorithm `lsqnonlin` minimize a vectorial functional calculus and one thus needs décommenter the first line “iret=”. This algorithm is based on Levenberg-Marquardt with active constraints, and is thus similar to that established in *Code_Aster*.

**Note:**

*It is important to note the presence of the order `long format` and of the argument `precision` for the order `dlmwrite` . This makes it possible to work with a sufficient number of significant figures and not to stop if the tolerance on the function is not respected any more.*

**Notice 2:**

*When the software of optimization is not able to manage the codes return of the external execution, attention should be paid to properly stop the procedure in the event of abnormal stop of Code_Aster calculation. This is why the files are erased `input.txt` and `output.txt` just after their use, in order not to fall into loops without end.*