Macro-command MACR_RECAL

1 Drank

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 computations *Code_Aster* modelling these tests. MACR_RECAL makes it possible to determine the parameters of these computations (which can be parameters of constitutive law, 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 Lr

```
=MACR_RECAL      [listr8                                          ] (♦
             UNITE_ESCL   =uni          [I]                      ♦
             /  RESU   _EXP=resu_exp           [assd             ] RESU
                    _CALC=resu_calc            [assd             ] /COURBE
                    = _F    (♦
                    FONC   _EXP=fonc_exp       [sd_fonction            ] ♦
                    NOM    _FONC_CALC=nom_fonc_calc      [kN            ] ♦
                    PARA   _X=para_X           [kN                   ] ♦
                    PARA   _Y=para_Y           [kN                   ] ◊
                    POIDS=val_poids               [R]                )
                    ◊
             LISTE  _POIDS=poids            [assd                  ] ♦
             will /LISTE_PARA=liste_para                    [assd
      ] /PARA_OPTI
                    = _F    (♦
                    NOM    will _PARA=nom_para        [kN               ] ♦
                    VALE   _INI=val_ini        [R]                  ♦
                    VALE   _MIN=val_min        [R]                  ♦
                    VALE   _MAX=val_max        [R]                 )
                    ◊

                UNITE_RESU=/91          [default                        ] /
uni_r
                                  [I]                        ◊
                ITER_MAXI=/10           [default                        ] /
it
                                  [I]                           ◊
                ITER_FONC_MAXI=/100     [default                     ] /it
                                  [I                        ] ◊
                RESI_GLOB_RELA=/1.E-3       [default                    ]/
                                  resi [R]                      ◊
                TOLE_PARA=/1.E-8            [default                    ]/
                                  resi [R]                      ◊
                RTOLE_FONC=/1.E-8           [default                    ]/
                                  resi [R]                      ◊
                PARA_DIFF_FINI=/1.E-5       [default                    ]/
                                  coeff [R]                     ◊
             GRAPHIQUE=     _F  (◊
                    UNITE   =     /   90 [default                     ]/
                                      uni_g [I]                   ◊
                    FORMAT=           /  "XMGRACE" [default             ]/
                                      "GNUPLOT" #
                    If FORMAT = XMGRACE ◊
                    PILOTE=           /  '' [default                    ]/
                                      "POSTSCRIPT" [kN          ]/
                                      "EPS"/
                                      "MIF"/
                                      "SVG"/
                                      "PNM"/
                                      "PNG"/
                                      "JPEG"/
                                      "PDF"/
                                      "INTERACTIF")
                    ◊
```

```
                AFFICHAGE=       /  "TOUTE_ITERATION" [default              ]/
                                    "ITERATION_FINALE" ◊
                METHODE=/        "  LEVENBERG" [default                   ]/
                                    "FMIN"/
                                    "FMINBFGS"/
                                    "FMINNCG"/
                                    "GENETIQUE"/
                                    "HYBRIDE" ◊

                   CALCUL_ESCLAVE= _F   (◊
                      LANCEMENT=    /  "INCLUSION" [default
]/
                                       "DISTRIBUTION" #
                      If distribution ◊
                      MODE=  /         "INTERACTIF" [default               ]/
                                         "BATCH" ◊
                      MEMOIRE=   memory       [I]                          ◊
                      TEMPS=  time         [I]                          ◊
                      CLASSE=    classifies          [kN
] ◊
                         NMAX_SIMULT= nmax    [I]                          #

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

                   if METHODE = GENETIQUE, HYBRIDE ◊
                      NB_PARENTS=/10          [default
]/
                                          nb_parents [I]                  ◊
                      NB_FILS=/5           [ default
]/
                                          nb_fils [I]                     ◊
                      ECART_TYPE=/1    .  [default                          ]/
                                          variation [R]                   ◊
                      ITER_ALGORITHME_GENE=/10       [default
          ]/
                                          itergene [I]                 ◊
                      RESI_ALGORITHME_GENE=/1.E-3       [default
          ]/
                                          resige [R]                      ◊
                   GRAINE=graine                  [I]                        ◊

                   DYNAMIQUE=     _F (◊
                             MODE_EXP= mode_exp   [assd                  ] ◊
                             MODE_CALC= mode_calc    [assd                  ] ◊
                             APPARIEMENT_MANUEL=/   "NON" [default
]/
                                                  "YES") ◊

                   INFO=/           1   /
                                       2 [default
] #

                   if METHODE = FMINBFGS, FMINNCG ◊
                   GRADIENT=/          "  NON_CALCULE" [default
]/
                                       "NORMAL"/
                                       "ADIMENSIONNE");
```

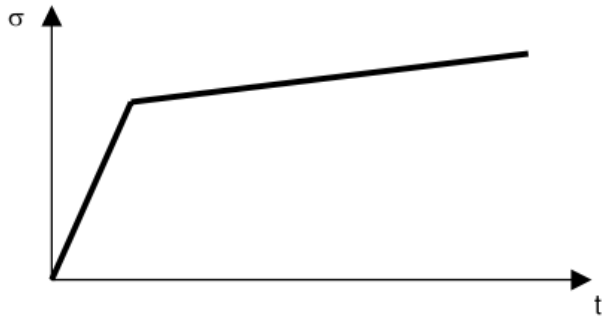
General

3 presentation Principle

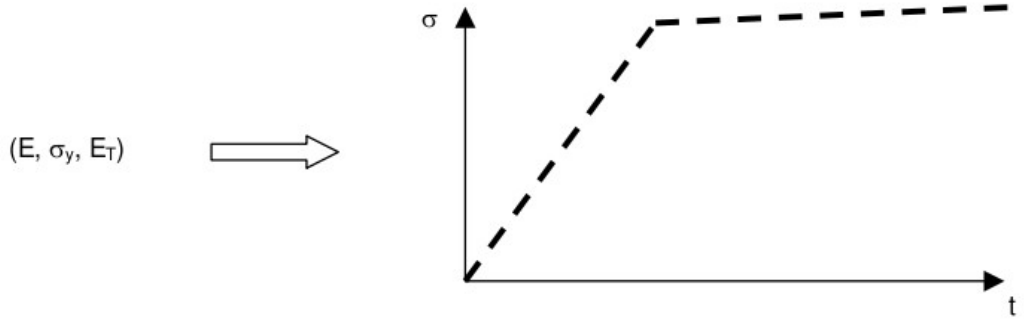
3.1 of retiming Let us consider

the model problem of identification of the elastoplastic characteristics E, σ_y (respectively E_T elastic limit, Young modulus and hardening modulus) of a material on a traction test uniaxial. There

is on the one hand experimental curve of tension giving the evolution of the stress according to the time and which is a data: There



is in addition a function **of** the 3 parameters which for each value of the triplet E, σ_y returns E_T a calculated curve of tension:



The purpose of retiming is then to answer question: Which

are the values of describing (E, σ_y, E_T) *my experiment as well as possible? Organization*

3.2 of retiming

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

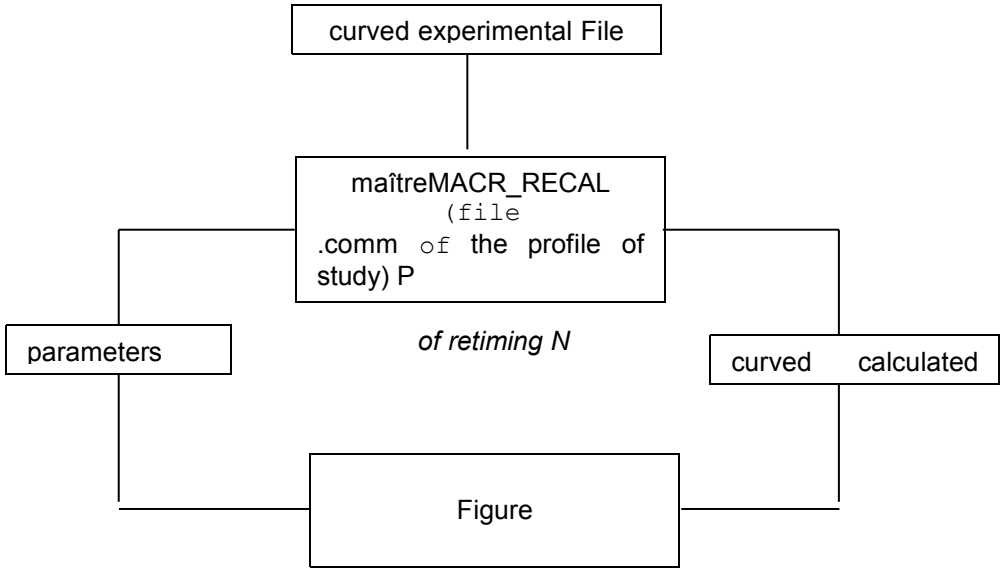
- experimental N curves (with each one of these curves can be allotted an arbitrary weight),
- parameters P to readjust like, for each one, an estimate of its initial value, its minimal value and its maximum value,
- the command file modelling the tests N which one wants to readjust,

- names of the quantities N to be extracted from the command file above and which will be readjusted on the experimental N curves. These quantities must be contained in an array resulting from POST _RELEVE_T.

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

- a command file says main **containing** the experimental N curves, the parameters P , the names of the quantities 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 says slave **modelling** the experimental tests. Indeed

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



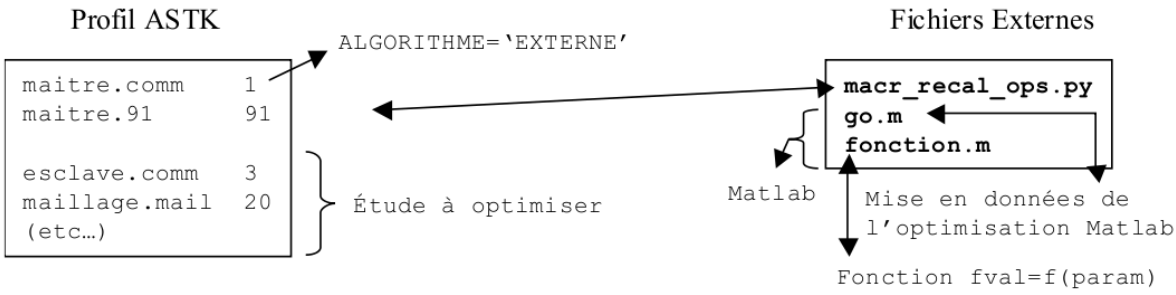
3.2-a. Diagram of operation of the procedure of retiming for the classical case In

the following part are described the operands 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 command. The part “Example of use” is there to light the user.

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

3.3 of use: EXTERNAL mode In

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



3.3 3.3-a EXTERNAL MACR _RECAL method Code_Aster

is not that one limps black which takes in entry a textual file containing the list of the values of parameters, carries out the computation of the functional calculus for this set of parameters, carries out possibly the computation 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, paragraph 3.1 3.1 valid. For more details, one returns the user in paragraph 7 7 Typical case

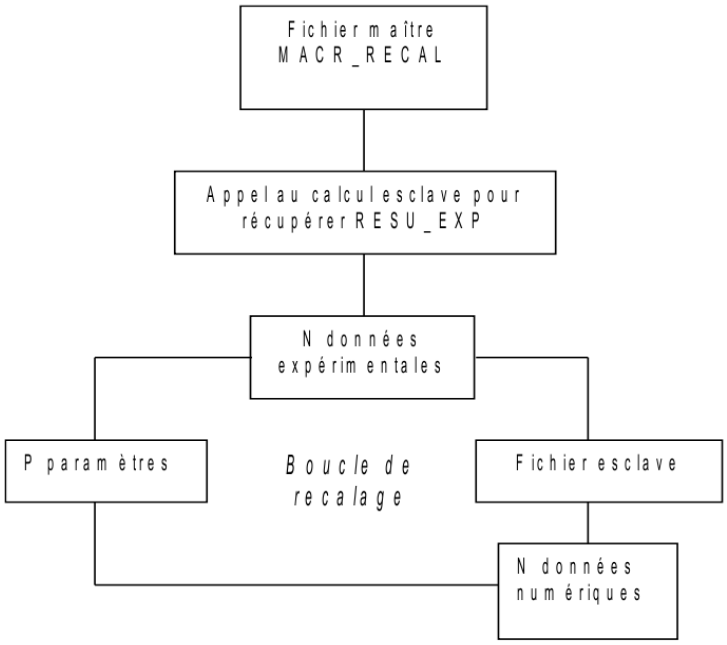
3.4 of the retiming of a dynamic model In the case of

the retiming of the parameters of a dynamic model by means of the experimental data resulting from the dynamic analysis, there exist some characteristics for the implementation. The experimental data are in this case of the eigenfrequencies and the eigenvectors (deformed modal) contained in a concept _meca mode resulting from measurement.

The user usually lays out of this concept 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 environment Code_Aster. This "experimental" model contains inter alia the experimental mesh, i.e. the mesh of the sensors, coarser than the mesh 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 the first computation slave since the external file "unv". The format used to transmit this information to MACR _RECAL is the same one as that for the RESU_CALC : a python list of python lists N containing the names of the arrays and the columns containing the experimental responses. Below

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



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

other characteristics relate to the command file of computation slave: he

- contains also the commands necessary to the construction of the experimental model. The concept _meca `mode usable` to extract RESU_EXP is provided by READING `_RESU;`
- The quantity corresponding to the experimental and numerical responses are always contained in arrays but more necessarily resulting from POST `_RELEVE_T.` The responses related to the modal deformed shapes result either from CREATED `_TABLE (for` the criterion of experimental MAC), or of MAC `_MODES (for` the numerical criterion of MAC). It is pointed out that the criterion of MAC [U4.52.15] is used to compare two modal bases, it being here of that experimental and that numerical. Operands

4 Operand

4.1 UNITE_ESCL ♦

`UNITE_ESCL logical`

Number of unit of slave file, allotted in the interface Astk `(column` UL `).` The extension of this file can be unspecified. Operands

4.2 RESU_EXP , RESU_CALC, FONC_EXP, NOM_FONC_CALC, PARA_X, PARA_Y, POIDS

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

- `and` RESU_CALC `and` possibly LIST_POIDS `which` require the seizure of python lists,
- key word FONC_EXP , NOM_FONC_CALC , PARA_X , PARA_Y , POIDS `of` factor key word the COURBE `which` has the advantage of using Aster concepts. ◊

`RESU_EXP Name`

of the python list of tables N numpy containing the experimental N curves. For a static model the list is beforehand defined in the form: resu

```
_exp= [numpy.array   ([[x0, y0    ], [x1, there
                                  1],… [
                                   xn,
                                  yn] ]), .....
          numpy.array
              ([[u0, v0]    , [ u1, v1
                                  ],… [ one,
                                  vn]]
                                  )] For
```

the retiming of a dynamic model with modal experimental data (frequencies and eigenvectors), it will be the name of the python list of python lists N containing the names of the arrays and the columns containing the experimental responses. For example: resu_exp

```
= [["REPEX1", "NUM_ORDR", "FREQ"], ["REPEX2", "NUM_ORDR", "MAC_EXP"]]
◊RESU_
```

CALC Name of

the list of N python lists containing the names of the arrays and the columns containing the numerical responses corresponding to experimental measurements on which one will carry out retiming. For example: resu_calc

```
= [["TABLE1", "INST", "SIYY"], ["TABLE2", "INST", "V1"],…]) ◊LISTE_
```

POIDS Name of Numpy

table containing the weights N to be assigned to the experimental N curves. If the key word 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

beforehand definite by the operator DEFI_FONCTION which corresponds to the experimental data. To establish the link with RESU_EXP, the function provided to this key word with the iteration formula corresponds i to the formuleème i table of RESU_EXP. ◊ NOM_FONC_CALC

Character string

corresponding to the name which one wishes to allot to the function after retiming of the experimental data. ◊POIDS 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

formulates X function. ◊PARA_Y Name

of the parameter

formulates Y 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',),)
```

Operands

4.3 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

- which require the seizure of python lists, key word
- NOM_PARA, VALE_INI , VALE_MIN, VALE_MAX of factor key word the PARA_OPTI which has the advantage of using concepts Code_Aster (numerical values or character strings). ◊LISTE_PARA

Name of the python list

of python lists P 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]] Attention
```

: It is asked

that the names of the variables end in two underscores (for example: YOUN). Note:

The limits

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 formula corresponds i to the first element of the formuleème i of the list provided to LIST_PARA. ◊VALE_INI

initial Value

of the parameter. The reality provided to NOM_PARA with the iteration formula corresponds i to the second element of the formuleème i of the list provided to LIST_PARA. ◊VALE_MIN

minimal Value

of the parameter. The reality provided to NOM_PARA with the iteration formula corresponds i to the third element of the formuleème i of the list provided to LIST_PARA. ◊VALE_MAX

maximum Value

of the parameter. The reality provided to NOM_PARA with the iteration formula corresponds i to the fourth element of the formuleème i of the list provided to LIST_PARA. Operand

4.4 UNITE_RESU ◊UNITE_RESU

logical Number

of unit of the results file of retiming (evolution of the parameters during iterations, convergence criteria). Operand

4.5 ITER_MAXI maximum ◊

ITER_MAXI
Nombre of iterations of retiming. Operand

4.6 ITER_FONC_MAXI ◊ITER_FONC

_MAXI Many
maximum evaluatings of the functional calculus. Operand

4.7 RESI_GLOB_RELA ◊RESI_GLOB

_RELA relative total
Residue of retiming. This value
is disjoined of that well informed for nonlinear solvers STAT_NON_LINE and DYNA_NON_LINE . Operand

4.8 TOLE_FONC ◊ TOLE_FONC

Stopping criteria
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 norm of my functional calculus. Operand

4.9 TOLE_PARA ◊ TOLE_PARA

Stopping criteria
of the algorithm of retiming based on the variation of the parameters from one iteration to another. This criterion corresponds to the norm: square root $L2$ of the sum of the squares of the differences in each parameters. Operand

4.10 PARA_DIFF_FINI ◊PARA_DIFF

_FINI retiming
requires the computation of derivatives of the responses compared to the parameters. This computation

is carried out by finite differences. PARA_DIFF_FINIES corresponds to in the following α formula: Operand

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

4.11 GRAPHIQUE ◊ UNITE logical

Number
of unit of the graphs produced during retiming. A each iteration, MACR_RECAL produces display files N (whose format is defined by the key word PILOTE) representing the experimental and N calculated curves. Standard ◊PILOTE

of display
of the graphs. If PILOTE
= "INTERACTIF ", xmgrace is open in an interactive way with the graph. If PILOTE is worth "POSTSCRIPT ", "EPS", "MIF", "SVG", "PNM", "PNG", "JPEG" or "PDF" then

xmgrace are used to generate the file of corresponding format, in the logical unit defined by UNITE. Caution:

Mode INTERACTIF

is possible only if when retiming turns in interactive and not in batch. ◊FORMAT Choice

of the software
of display of the curves in interactive mode: xmgrace or gnuplot . The use of Xmgrace is blocking : it is necessary to close the Xmgrace window to continue the execution. ◊AFFICHAGE

Display of
the curves to each iteration or only at the end. Operand

4.12 METHODE ◊ METHODE

Method or
selected algorithm of optimization: LEVENBERG (

1) default) **:** algorithm of Levenberg-Marquardt. This algorithm is that recommended in the problems of standard quadratic minimization least squares, like problems of retiming of materials parameters. FMIN: Nelder
2) - Mead Simplex algorithm (uses only estimates of the functional calculus). FMINBFGS:
3) method Quasi-Newton (uses the functional calculus and the gradient of the functional calculus). FMINNCG: method
4) Line-search Newton Conjugate Gradient (uses the functional calculus, the gradient of the functional calculus and its hessien). GENETIQUE:
5) algorithm évolutionnaire based on the mechanism of the selection and the replacement. It is an algorithm which is expensive in TEMPS CPU, one advises its use only for one coarse exploration of space of the parameters in the frame of the hybrid technique of retiming presented in the following point. HYBRIDE: technique
6) which combines the stochastic one with the determinist - the algorithm évolutionnaire with the algorithm of Levenberg – Marquardt The EXTERNE mode
7) : this method makes it possible to use an external algorithm of optimization to 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 finite differences. It is not a key word of MACR_RECAL because the EXTERNAL mode is used directly by the call of the file bibpyt/Macro /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 algorithms FMIN* for problems of minimization of the least squares type, 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 stresses in order to manage limits on the parameters, whereas the other algorithms do not manage the limits. The other algorithms

can nevertheless be useful whenever Levenberg-Marquardt is put in a difficult situation. For example, algorithm FMIN does not use gradients, whose evaluating can in certain very particular cases generate numerical problems (very significant parameters, not enough experimental values, or others). Algorithm FMIN is definitely slower, but will be able to manage to converge (to make a parallel, the problems are 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 modulus Python distributed on Internet (http://pylab.sourceforge.net) under license LPG by Travis E. Oliphant, in addition principal contributor of the Python-Scipy project and person in charge of the modulus of optimization of Scipy. The

details of algorithmy and implementation can be found on the http://pylab.sourceforge.net page . Method HYBRIDE

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, in the frame of this method, one launches initially a coarse search 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. Key word CALCUL_ESCLAVE

4.13 Operand LANCEMENT

4.13.1 ◊LANCEMENT 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. Operand INCLUSION

4.13.2 ◊INCLUSION In

this mode,
slave file is included. There is thus no waste of time for the generation of a new study, the creation of the temporary directory 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

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

4.13.3 , MODE , MEMOIRE, TEMPS, CLASSE, UNITE_SUIVI ◊DISTRIBUTION In

this mode, with
each iteration of the algorithm of optimization, the slave computations $N+1$ (for a retiming of parameters) N are carried out in parallels, batch or interactive, by means of the modulus of distributed computations of as_run. Compared to Inclusion mode, each study is slightly longer to be carried out, because it is necessary to regenerate a new study 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 period), 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 computations slaves: ◊MODE: INTERACTIF
or BATCH ◊MEMOIRE : Mo
◊TEMPS memory : time
in second ◊CLASSE: classify
batch , allows to force computations to use a specific class, for example “distr” on the server Code_Aster ◊UNITE _*SUIVI:*

if this key word 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 computations slaves launched in parallel in distribution mode (if no value is indicated, the code automatically decides this number) It are possible

to use parallelism MPI for the studies slaves. In this case, it is necessary to launch master computation with a version MPI and on only one processor (in interactive mode or batch). Characteristics MPI of computations slaves must be indicated by the key word following: ◊ MPI_NBCPU: many
processors MPI for each computation slaves launched in parallel ◊ MPI_NBNOEUD:
many nodes for each computation slave launched in parallel It should be noted that problems

of operating can come to disturb the launching of computations distributed slaves (for example, classes of batch are badly defined and the computations cannot be carried out). In this case, the error obtained in the .mess of master computation can be rather sober (an error message of the style “at least one of computations slaves could not start”). To obtain information additional concerning the distributed modulus of as_run, it is necessary to put UNITE_SUIVI and INFO =2 simultaneously. Operands NB_PARENTS

4.14 and NB_FILS ◊NB _PARENTS For

method
GENETIQUE or HYBRIDE , 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 Represents

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 “wires ″. 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. Operand ECART_TYPE

4.15 ◊ECART _TYPE It is

the value of
the standard deviation which the user imposes for quasi-random pullings on the fate of the “wires”. 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 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 is 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 TEMPS CPU. Operand ITER_ALGO_GENE

4.16 ◊ ITER_ALGO_GENE

Number D″ maximum
iterations for the algorithm évolutionnaire. If method HYBRIDE is used, this value (or RESI_ALGO_GENE) will determine the transition with the algorithm of Levenberg-Marquardt. Operand RESI_ALGO_GENE

4.17 ◊ RESI_ALGO_GENE

relative Residue of
retiming the algorithm évolutionnaire. If method HYBRIDE is used, this value (or ITER_ALGO_GENE) will determine the transition with the algorithm of Levenberg-Marquardt. Operand GRANULATES

4.18 ◊ GRAINE Specified value

by
the user for the seed of the generator of pullings on the fate in the algorithm évolutionnaire. If one informs a value for this key word, one forces the generator of random numbers present in the algorithm évolutionnaire to always generate the same lotteries, therefore there will be a repetition of the solution. Its employment is reserved only for the benchmarks **for** reasons of follow-up of the NON-regression of the code. Operand DYNAMIQUE

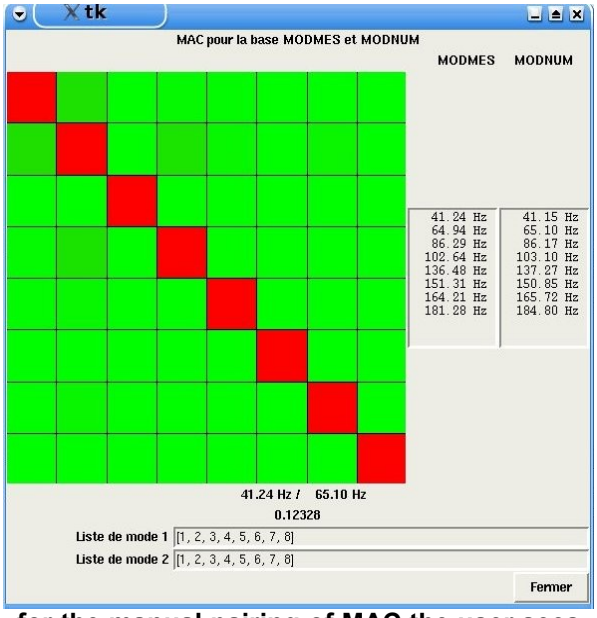
4.19 One informs this

operand for the retiming of the parameters of a dynamic model by modal analysis. ◊MODE_EXP Name of

the concept _meca
mode which contains the experimental modal data. This concept is extracted in computation slave by a LIRE_RESU. ◊MODE _CALC Name

of the concept _meca
mode which contains the numerical modal data. This concept is calculated in slave file by a MODE_ITER_*. ◊APPARIEMENT _MANUEL

Choice to display
in interactive mode of a chart window which will make it possible to pair the eigen modes manually. One thus avoids for example the automatic bad pairing of the eigen modes doubles cross. The capture of screen
presented in the Figure 4.19-a illustrates this chart window . Appear 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 commands 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 makes it possible *to recover* the new lists of the modes whose order was possibly modified by the user. Operand GRADIENT

4.20 ◊GRADIENT `For`

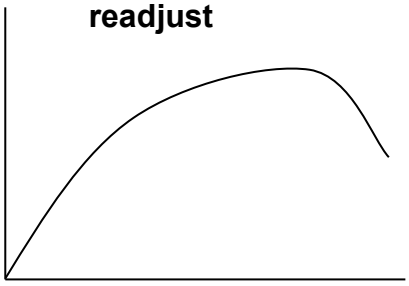
`methods`
FMINBFGS, FMINNCG `or EXTERNE , this key word makes it possible` to indicate to Code_Aster the way *of calculating* the gradients (adimensionné or not), or of not calculating them. Note: for the algorithms

which use the gradients, those can be calculated by finite differences automatically by Code_Aster, or *calculated* in slave file by means of, for example of the sensitivity analyzes (key word LIST_DERIV). Precautions for use

5 Here a set of

advice essential to **the good use** of retiming. The experimental curves

- are defined like tables with two columns: for the X-coordinates and for the Y-coordinates. The experimental curves
- must be functions: to a X-coordinate only one Y-coordinate should correspond. If an experimental curve comprises cycles, (for example forced according to the strain 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 Y-coordinates of the cyclic curve according to the parameter (for example strain according to time and stress according to time). PROHIBITED One



- curved calculated N on experimental curves N . 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 operands RESU_EXP and RESU_CALC . The quantities `calculated`
- well informed under operand RESU_CALC must `result` from POST_RELEVE_T (except `for the dynamics` where one can have arrays 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: DEBUT ()(); DSDE =

```
200. ; YOUN
__= 8.E4; SIGY
__= 10. ; ......
The initial
values
```

- of the parameters of retiming are those well informed for operand LIST_PARA and not `those present` in slave file of the user. A each iteration
- of retiming, the computations defined in slave file must converge. In the frame of retiming of nonlinear computations, it is thus strongly recommended to use the automatic cutting of time step. In the frame of
- retiming of nonlinear computations with automatic cutting of time step, it is essential **to define a list** of archivage under operand LIST_ARCH. Retiming `is`
- a powerful layer to obtain values of parameters from tests. It is however not miraculous: the experimental curves must **sufficient** contain information to identify the parameters. It is for example impossible to identify elastoplastic parameters with a test remaining in the elastic domain. The experimental tests must thus excite the parameters to be identified. In same logic
- , it is desirable that the experimental curves 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. Example of Identification

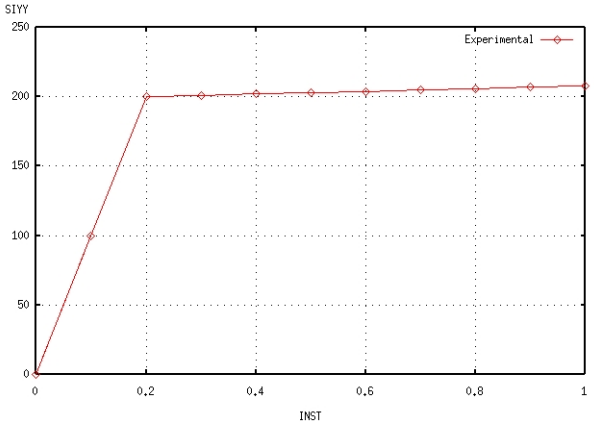
6 use of

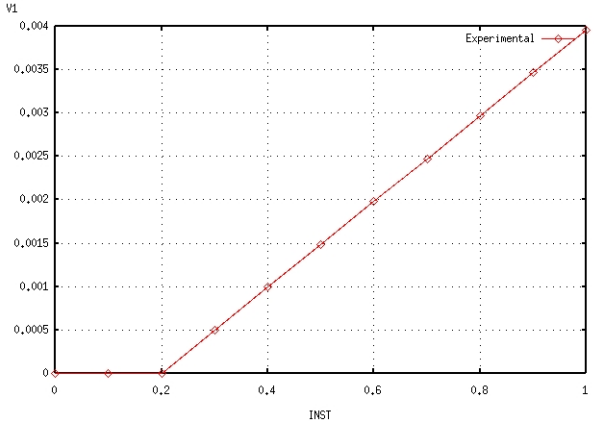
6.1 the parameters of one elastoplastic constitutive law on a traction test This example is treated

by test ZZZZ159A [V1.01.159]. Position of the problem

6.1.1 One has the results

of a traction test. It is the evolution of the stress in the course of time σ_{yy} as well as evolution of the plastic strain cumulated in the course of time p . Stress SIYY cumulated





Plastic strain One wishes to readjust

on these tests the Young's modulus, the elastic limit and the hardening slope of elastoplastic constitutive law with linear isotropic hardening. Put in data

6.1.2 Experiment One starts

6.1.2.1 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, 5.00000  E-02, 5.00000
                          E+01,…  9.50000E-01
                          , 2.07500
                          E+02, 1.00000  E+00, 2.08000
                          E+02),)  exper2=DEFI_FONCTION

(NOM_PARA=' INST', NOM_RESU=' V1', VALE
                    = (0.00000E+00
                    , 0.00000E+00, 5.00000  E-02, 0.00000
                          E+00,…  9.50000E-01
                          , 3.71250
                          E-03, 1.00000  E+00, 3.96000
                          E-03),)  exper1 and exper2
```

is thus functions of Code_Aster, which respectively represent the stress and the cumulated σ_{yy} plastic strain. Computation One writes p

6.1.2.2 then

the command file Code_Aster slave *modelling* this traction test where will appear our 3 parameters as well as the two curves to be readjusted. DEBUT ()(); # ASSIGNMENT

```
OF
THE VALUES OF THE PARAMETERS A TO READJUST # THE VALUES INDICATED
HERE ARE SANS IMPORTANCE # ALONE LE
COUNT THE VALUES INDICATED IN 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 (

COMP_INCR=_F (RELATION=' VMIS_ISOT_LINE'), MATER=ACIER, INCREMENT
                  =_F (LISTE
                  _INST=INSTANTS,), NEWTON=_F (REAC_ITER
                  =1), EPSI_IMPOSE=_F (EPYY
                  =EPYY,),); # EXTRACTION OF


RESPONSE SIGMAYY (T) REPONSE1=CALC_TABLE
(ARRAY = U, ACTION =_F (OPERATION
               = ' EXTR', NOM_PARA= (“INST”,
                           “SIYY”))) # EXTRACTION OF

RESPONSE EPSP (T) REPONSE2=CALC_TABLE
(ARRAY = U, ACTION =_F (OPERATION
               = ' EXTR', NOM_PARA= (“INST”,
                           “V1”))) FIN (); MACR_RECAL
```

It is necessary

6.1.2.3 for us now

and to define in the master file the initial values beaches of variations of our parameters. One wishes: 1.E 5<Module of initial

Young = 1.E5<5.E5 5.<Limite of initial
elasticity = 30. <500 1.E3<Module of initial
hardening = 1.E3<1.E4 Finally it is also necessary

to define in the master file the quantities to be extracted from the command file slave. We wish on the one hand to extract the column INST and column SIYY from the array response 1 and on the other hand the column INST and the column V1 of the array response 2. We write it: Here how we

inform this information in the body of MACR_RECAL: RESU 2=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=exper
                 2,   NOM_FONC_CALC='  REPONSE2', PARA_X=' INST',
                         PARA_Y=' V1'),),)   Put in data
```

6.1.2.4 alternative (python lists) One can use

the setting in data containing python list and of numpy object. The master file then will be written: experience= [numpy.array

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

100000. , 50000. , 500000.], ["DSDE", 1000. , 500. , 10000.], ["SIGY", 30. ,
5. , 500.]]
             computation = [["REPONSE1", "

INST", "SIYY"], ["REPONSE2", "INST", "V1"]] where: experiment is
```

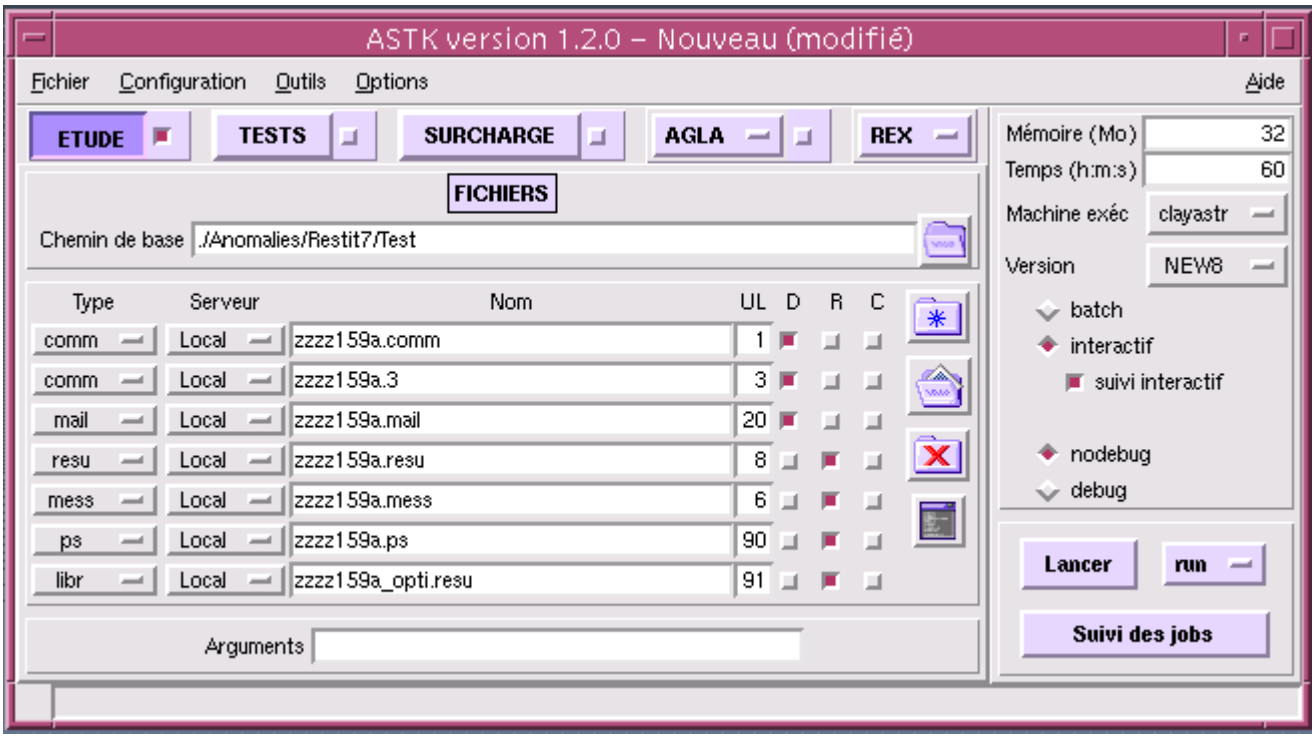
the name D "a python list (definite between hooks) of 2 tables numpy, parameters is a list of python lists containing the initial values and the beaches of variations of our parameters and computation the quantities to be extracted from the command file slave. The body of the command

is written then: RESU=MACR_RECAL (UNITE_ESCL

```
=3, RESU_EXP
         =EXPERIENCE       ,
         LIST_PARA         =PARAMETRES,
         RESU_CALC        =CALCUL,); Astk
         One defines        finally
```

6.1.2.5 the profile

D" study according to: Results Once



6.1.3 the study carried out

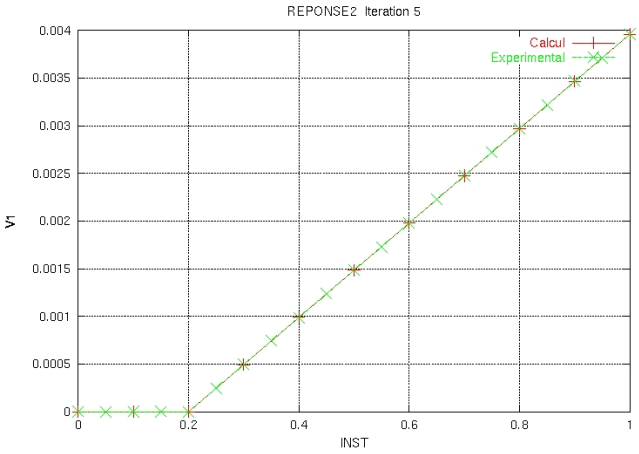
, the results file of ZZZZ159_opti.resu retiming contains following information: Computation of the sensitivity

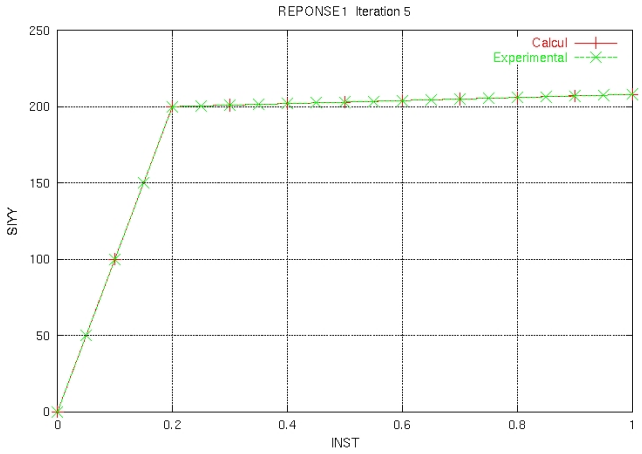
```
compared to = YOUN DSDE SIGY ==========================
============================== Iteration 0 = => Fonctionnelle
= 1.0 =>

Résidu = 1.0 => Paramètres
= YOUN          __
= 100000.0 DSDE_     _=
          1000.0 SIGY = 30.0
          ============
          ==============
============================== Computation 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 ===
          ======================
============================== Computation 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 ===
          ======================
============================== Computation sensitivity


compared to = YOUN DSDE SIGY ==========================
============================== Iteration 3 = => Fonctionnelle
= 0.00190706595529

=> Résidu = 0.0520849911718
=> Paramètres        = YOUN
= 192302.166747    DSDE
          __= 895.845518907 SIGY
          __= 203.753909707 ===
          ======================
============================== Computation of the sensitivity


compared to = YOUN DSDE SIGY ==========================
============================== Iteration 4 = => Fonctionnelle
= 2.70165453323

e-06 => Résidu = 0.00172172540305
=> Paramètres        = YOUN
= 199801.572817    DSDE
          __= 1928.08902726 SIGY
          __= 200.274590793 ===
          ======================
============================== Computation of the sensitivity

compared to = YOUN DSDE SIGY ==========================
============================== Iteration 5 = => Fonctionnelle
= 2.65431115925

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

============================== CONVERGENCE REACHED ====
                    ======================
============================== Eigenvalues of Hessien

: [7.17223479e+00 3.67264061
 e-01 6.25194340   e-04] associated   Eigenvectors

: [[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 computation: 1) +9.8E-01 * YOUN -1.9E

       - 01 * DSDE associated with the eigenvalue
         7.2E+00 the following combinations


    with parameters are insensitive for your computation: 1) -1.9E-01 * YOUN
    -9.8E

       - 01 * DSDE associated with the eigenvalue
         6.3E-04 And file POSTSCRIPT
```

ZZZZ159.ps contain : `Identification` of the parameters





6.2 of a dynamic model by means of the experimental data resulting from the modal analysis 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 means of experimental measurements of eigen modes. Put in data In the master file

6.2.1 one

informs initially the names of the arrays which will contain at the same time the results calculated thus that the numerical results: computation = [[“REPONSE1”, “NUME_ORDRE

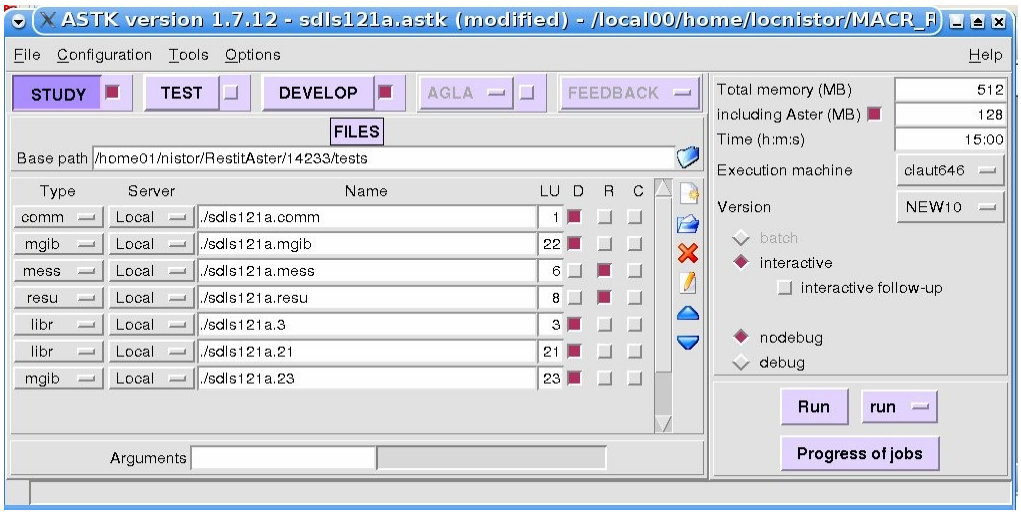
```
”, “FREQ  ”], [“REPONSE2   ”, “NUME_ORDRE   ”, “MAC
                ”]] experience=   [[“REPEXP1   ”, “NUME_ORDRE

”, “FREQ  ”], [“REPEXP2”, “NUME_ORDRE   ”, “MAC_EXP
                ”]] It  is important to note
```

here that, for the calculated results, the REPONSES2 is given by the diagonal of the matrix of MAC resulting of the command 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 response corresponding one chose “MAC_EXP”) the options retained in

command MACR_RECAL, always in the master file are: RESU=MACR_RECAL (UNITE_ESCL

```
=3, RESU_EXP=experience
                    , LISTE_
                    PARA=parametres         , RESU
                    _CALC=calcul        , POIDS=poids
                    ,           METHODE=' HYBRIDE
                    “,              ITER_FONC
                    _MAXI=500         , NB_PARENTS
                    =10, NB_FILS=5   ,
                    ECART_TYPE=10    .,
                    ITER_ALGO_
                    GENE=2, DYNAMIQUE
                    =_F (  MODE_EXP=' MODMES
                    ”, MODE_CALC
                                = ' MODNUM', APPARIEMENT
                                _MANUEL=' NON',)
                                ,); One thus chose to launch
```

retiming by means of method HYBRIDE with two iterations for the algorithm évolutionnaire. Finally one defines the following profile for the study: Computation slave the principal



6.2.2 characteristic

of computation slave is the presence in the command file (unit 3) of two models: the model experimental and that numerical. Below one presents, for this benchmark, the command file slave with the explanations necessary. - reading of the experimental mesh

(mesh of the sensors): PRE_GIBI (UNITE_GIBI=23); MAILEXP

```
1=LIRE_MAILLAGE ();
```

```
- creation of a nodes group
```

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 nodes group
```

to define the boundary conditions: MAILEXP1=DEFI_GROUP (reuse

```
=MAILEXP1, MAILLAGE=MAILEXP1   , CREA_GROUP
                        _NO=_F (GROUP_MA=
                        "EDGES", NOM=' EDGES    ",),); - 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 computations and
```

assemblies of the matrixes. 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 the sd_mode
```

_meca with the experimental eigen 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", "DRZ"),), TOUT_ORDRE=' OUI',); And one
           continues with the model
```

numerical, 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=MODE_ITER_SIMULT
(MATR
```

```
_RIGI=M_AS_RIG, MATR_MASS=M_AS_MAS, METHODE
                        = ' SORENSEN', CALC_FREQ
                        =_F (OPTION=' PLUS_
                        PETITE', NMAX_FREQ=NF  ), VERI_MODE=_
                                    F (STOP_ERREUR='
                        NON'),); The mesh experimental
```

is increasingly coarser than the mesh of the digital model thus it is necessary to project the result numerical one on the experimental mesh: MODNUM = PROJ_CHAMP (RESULTAT

```
=MODES, MODELE_1=MODEL , MODELE_2=MODEXP
                        , NUME_DDL=
                        NUMEXP) One recovers
                        the table of
```

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 responses: 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. Use of the EXTERNAL mode Warning

7 We attract L

7.1 “attention on

the fact that this operating mode is to be held for a advanced use. Most case should be treated with the algorithms provided in command MACR_RECAL, and in particular L” Levenberg-Marquardt algorithm, which is adapted the most to the problems of retiming of parameters. This operating process requires

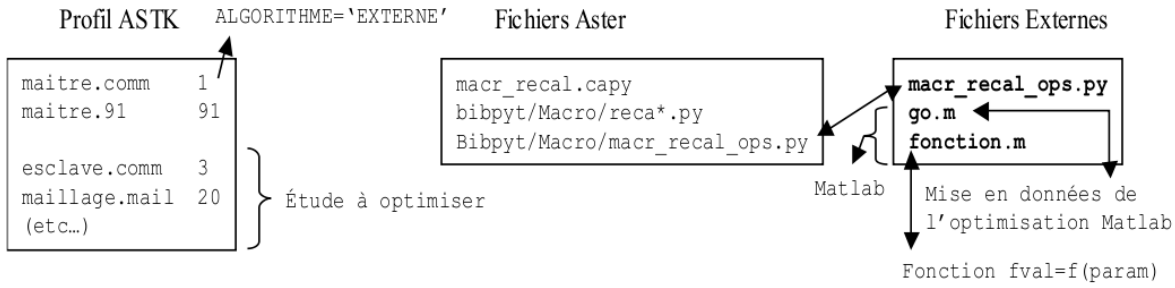
an external software with Code_Aster to carry out *optimization* (code Python, Matlab, Scilab, standard software black box, etc). Moreover, it is necessary to be some competent Python. Lastly, the use of the EXTERNAL

mode leaves perimeter QA of Aster, and should not be used for studies IPS. Methodology Principle In this

7.2 mode of use

7.2.1 ,

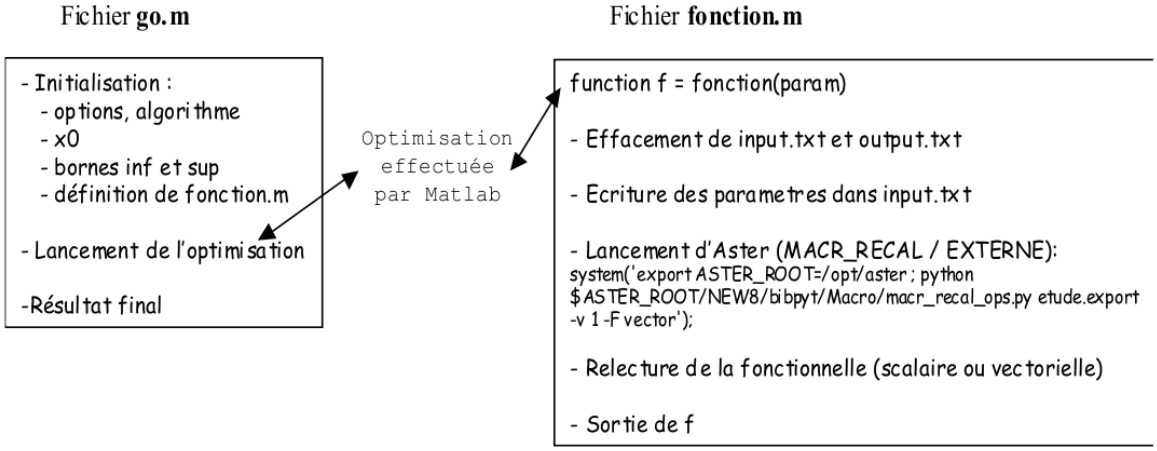
Code_Aster is only used *for* the evaluating of the functional calculus for a software of optimization which is completely external with Code_Aster. Appear 7.2.1-a: *MACR_RECAL*



"Algorithm 7.2.1-a " Generally , the software

of optimization require that the user write a procedure for the computation of the functional calculus: F = F (param). If the software authorizes the read/write

of files and the external code execution (command "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 relecture of the file of the value of the functional calculus in the routine of computation of F. Figure 7.2.1-b: MACR_RECAL "Principle



of 7.2.1-b ", Matlab example the software of optimization launches

Code_Aster to each evaluating of *the functional calculus* . The routine of evaluating of the functional calculus passes by textual files to send to Code_Aster the 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 slave file, 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 profile of L one use of MACR_RECAL). This profile must be completely functional . Under this EXTERNAL mode, the operating mode

is the following: the routine recal.py is given the responsibility to recover the parameters since the command line or since an input.txt file, 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 (the toolbox of optimization of Matlab, *the toolbox* Tomlab for Matlab, and modulate it ***optimization*** 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. Use of the external file of launching

7.2.2 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 file

```
_export [options] Options: - H, --help show this help message and

exit --input
  =INPUT            Chains text containing the parameters
  --input          _step=INPUT_STEP Character string of text containing
  the steps of discretization
                        of the finite differences --input_file=INPUT_FILE
                        File containing the parameters
  --input_step_file
                        =INPUT_STEP_FILE File containing
  the steps of discretization of
                        the finite differences --output=OUTPUT file containing
                        the functional calculus
  --output_grad       =OUTPUT_GRAD file containing
  the gradient --aster_root=ASTER
                        _ROOT Path of installation of
  Aster --as_run=as_run Path
                        towards as_run --resudir=RESUDIR
  Path 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 Path by default of
  the overloads Python --objectify
                        Functional =OBJECTIVE ([fcalc ]/[error
  ]) --standard
                        objective_type =OBJECTIVE_TYPE of the functional calculus
  (float/[vector]) --gradient
                        _type=GRADIENT_TYPE computation of the gradient
  by Code_Aster ([No] /normal/
                        adim) --mr_parameters=MR _PARAMETERS File of parameters
  of MACR_RECAL: parameters
                                    , computation, experiment --study _parameters
=STUDY_PARAMETERS
                        File of
  parameter of the study: export --parameters
                        =PARAMETERS File of parameters This
  procedure needs
                        : file .export
```

of the Astk study defined

1. previously (the standard study of MACR_RECAL) of a textual file containing the list of the parameters
2. From these two files, it launches a computation

Code_Aster for the specified parameters (or *two computations* 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 of the files by default **: if there is** one file .export in
1. the current directory, this one will be used (in the contrary case the procedure stops in error) the input file by default will be sought
2. under the name “input.txt” the output file will be generated with the name
3. “output.txt” the 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: -- information

: defined the level of message; 0=muet
1. , 1 or 2; --objective: allows to specify if the functional calculus
2. must be turned over in the vectorial or scalar form; --gradient: allows to specify if it is not wanted
3. 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 directory of *execution* of the procedure, making it possible to have access to the output of the last computation Code_Aster. Lastly, environment variable ASTER_ROOT (

the basic path of Aster) should be configured 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 directory export ASTER_ROOT=/aster echo *“10. , 20* . ,

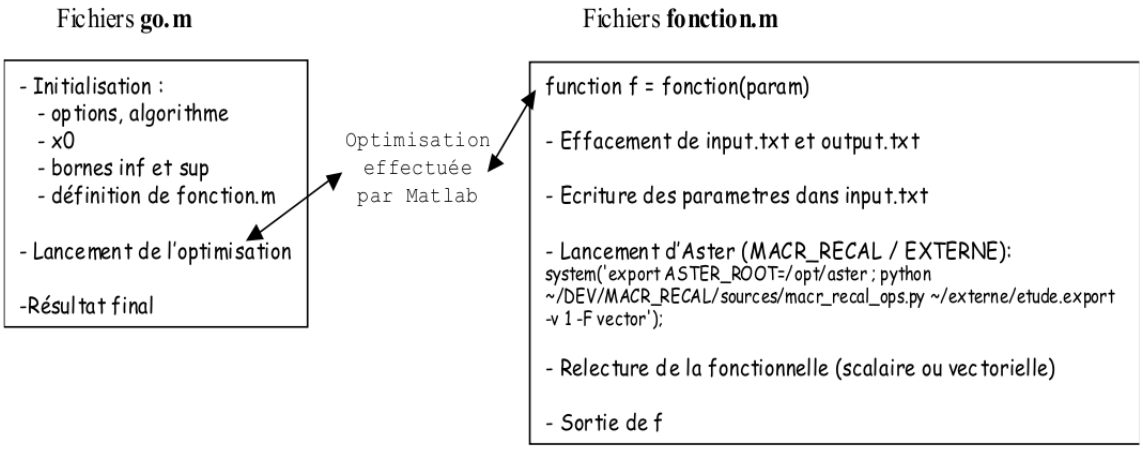
```
30. ” >input.txt
Python $ASTER_ROOT/NEW
10/bibpyt/Macro/recal.py etude.export – - information
=2 cat ouput.txt The computation Code_Aster should launch out and finish
suitably
```

. An output.txt *file* must be generated in the current directory. The zzzz159f benchmark illustrates the external use of MACR_RECAL

: in this test Code_Aster , one defines a function F and one once points out Code_Aster to simulate an iteration of retiming. Example: modulate *optimization* of Matlab© If the principle

7.3 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 which carries out the actions described below: Figure 7.3 – has: MACR _RECAL “Principle of the EXTERNAL mode” applied to



Matlab 7.3-a the following example uses the modulus of optimization of Matlab and carries out

computations Code_Aster on the local server. 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 (computation of F (param)) /version Linux function F =
function
```

```
(X) system ("rm - F input.txt"); system ("rm - F output.txt"

); % Writing of the parameters

dlmwrite ("input.txt", X, "
accuracy", "%.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, one gives

two examples of algorithms. The algorithm   lsqnonlin minimizes a vectorial
```

functional calculus and one thus needs décommenter the first line "iret =". This algorithm is based on Levenberg-Marquardt with active stresses, and is thus similar to that established in Code_Aster. Note: It is important to note the presence of the command long format and *of the argument*

accuracy

for the command 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 return codes

of the external

execution, attention should be paid to properly stop the procedure in the event of abnormal stop of computation Code_Aster. This is why one erases the files input.txt and output.txt right after their use, in order not to fall into loops without end.