Macro-commande MACR_RECAL

1 But

Recaler des résultats de calculs sur des résultats expérimentaux ou sur d'autres résultats de calculs.

Considérons d'une part un ou plusieurs résultats d'essais et d'autre part un ou plusieurs calculs Aster modélisant ces essais. MACR_RECAL permet de déterminer les paramètres de ces calculs (qui peuvent être des paramètres de loi de comportement, de chargement, etc ...) décrivant au mieux les essais. Pour plus de précisions sur l'algorithmie mise en œuvre, se reporter à [R4.03.06].

Table des Matières

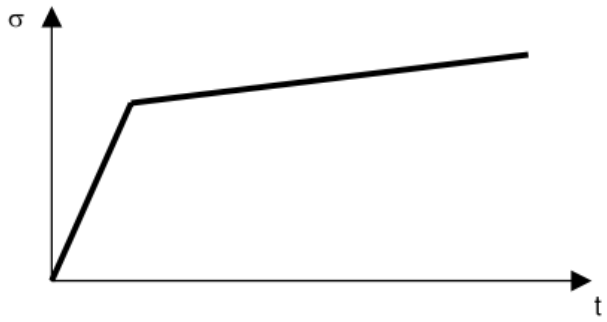
2 Syntaxe

```
lr =  MACR_RECAL                                                    [listr8]
          (  ♦  UNITE_ESCL       =  uni                             [I]
             ♦  RESU_EXP         =  resu_exp                        [assd]
             ◊  POIDS            =  poids                           [assd]
             ♦  RESU_CALC        =  resu_calc                       [assd]
             ♦  LIST_PARA        =  list_para                       [assd]
             ◊  LIST_DERIV       =  list_deriv                      [assd]

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

             ◊  RESI_GLOB_RELA   =  /1.E-3                          [défaut]
                                    / resi                          [R]
             ◊  PARA_DIFF_FINI   =  /1.E-3                          [défaut]
                                    / coef                          [R]
             ◊  GRAPHIQUE = _F(
                   ◊  UNITE      =  / 90                          [défaut]
                                    / uni_g                         [I]
                   ◊  PILOTE     =  / ''                          [défaut]
                                    / 'POSTSCRIPT'                  [Kn]
                                    / 'EPS'
                                    / 'MIF'
                                    / 'SVG'
                                    / 'PNM'
                                    / 'PNG'
                                    / 'JPEG'
                                    / 'PDF'
                                    / 'INTERACTIF'
                   ◊  FORMAT     =  / 'XMGRACE'                   [défaut]
                                    / 'GNUPLOT'
                   )

             ◊  AFFICHAGE =   / 'TOUTE_ITERATION'                   [défaut]
                              / 'ITERATION_FINALE'
             ◊  SUIVI_ESCLAVE=  / 'NON'                             [défaut]
                                / 'OUI'
             ◊  METHODE       =  / 'LEVENBERG'                      [défaut]
                                 / 'FMIN'
                                 / 'FMINBFGS'
                                 / 'FMINNCG'
                                 / 'EXTERNE'

             ◊  INFO          =  / 1                                [défaut]
                                 / 2

             # si METHODE = FMINBFGS, FMINNCG, EXTERNE
             ◊  GRADIENT      =  / 'NON_CALCULE'                    [défaut]
                                 / 'NORMAL'
                                 / 'ADIMENSIONNE'

             # si METHODE = EXTERNE
             ◊  TYPE_FONCTIONNELLE= / 'SCALAIRE'                    [défaut]
                                    / 'VECTORIELLE'
             );
```

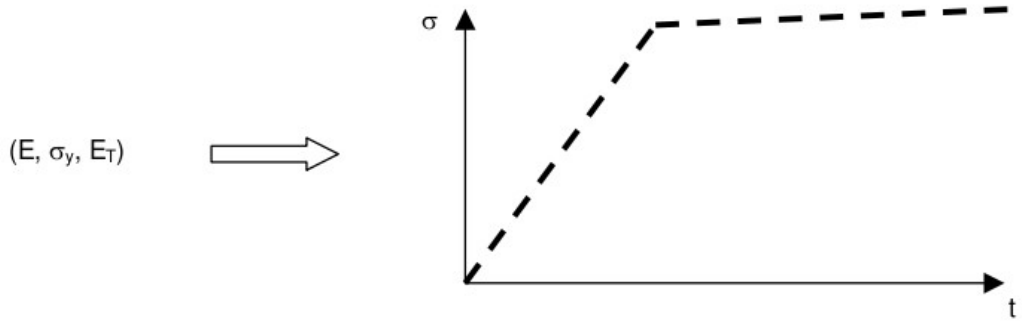
3 Présentation générale

3.1 Principe du recalage

Considérons le problème modèle d'identification des caractéristiques élastoplastiques E, σ_y , E_T (respectivement module d'Young, limite d'élasticité et module d'écrouissage) d'un matériau sur un essai de traction uniaxiale.
On a d'une part la courbe de traction expérimentale donnant l'évolution de la contrainte en fonction du temps et qui est une donnée :



On a d'autre part une **fonction** des 3 paramètres qui pour chaque valeur du triplet E, σ_y , E_T renvoie une courbe de traction calculée :



L'objectif du recalage est alors de répondre à la question :

Quelles sont les valeurs de (E, σ_y, E_T) *décrivant au mieux mon expérience ?*

3.2 Organisation du recalage

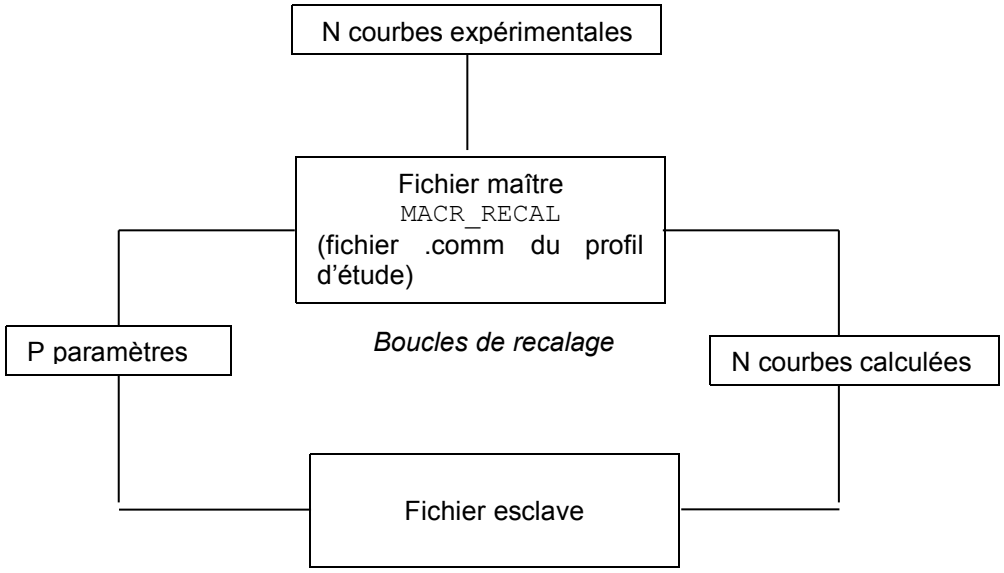
Pour mener à bien un recalage, il est nécessaire de disposer de l'ensemble des informations suivantes :

- les N courbes expérimentales (à chacune de ces courbes peut être attribué un poids arbitraire),
- les P paramètres à recaler ainsi que, pour chacun, une estimation de sa valeur initiale, sa valeur minimale et sa valeur maximale,
- le fichier de commandes modélisant les N essais que l'on veut recaler,
- les noms des N grandeurs à extraire du fichier de commandes ci-dessus et qui seront recalées sur les N courbes expérimentales. Ces grandeurs doivent être contenues dans une table issue de `POST_RELEVE_T`.

La mise en données de ces informations nécessite alors l'organisation suivante :

- un fichier de commandes dit **maître** contenant les N courbes expérimentales, les P paramètres, les noms des grandeurs à recaler ainsi que d'autres informations propres au recalage, le tout renseigné dans MACR_RECAL. Les différents formats utilisés sont précisés dans ce qui suit,
- un fichier de commandes dit **esclave** modélisant les essais expérimentaux.

En effet, le recalage est un processus **itératif** : le fichier maître exécute le fichier esclave, il récupère les N courbes calculées avec les valeurs courantes des P paramètres, il compare les valeurs des courbes calculées à celles des courbes expérimentales, il en déduit de nouvelles valeurs pour les P paramètres et relance le fichier esclave. Ce processus continue jusqu'à obtention de la convergence.



Dans la partie suivante sont décrites les opérandes de MACR_RECAL. On y fait référence à quelques notions du langage Python. Il n'est cependant nullement nécessaire de connaître Python pour utiliser cette macro commande. La partie « Exemple d'utilisation » est là pour éclairer l'utilisateur.

La structure de données produite est une liste de réels contenant les valeurs des paramètres à convergence en cas de convergence ou à la dernière itération dans le cas contraire.

3.3 Cas particulier d'utilisation : mode EXTERNE

Dans ce mode d'utilisation, l'algorithme d'optimisation est externe à Aster.

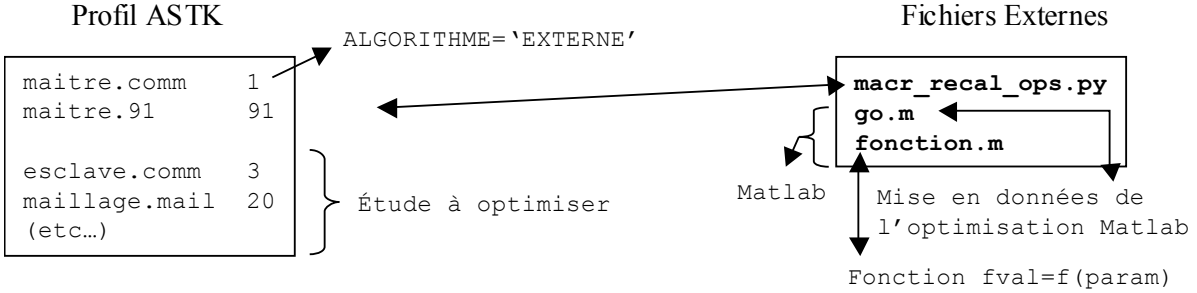


Figure 3.3-a : MACR_RECAL méthode EXTERNE

Aster n'est qu'une boite noire qui prend en entrée un fichier texte contenant la liste des valeurs de paramètres, effectue le calcul de la fonctionnelle pour ce jeu de paramètres, réalise éventuellement le calcul des gradients par rapport aux paramètres, et renvoie à l'algorithme externe, par l'intermédiaire d'un fichier texte, la valeur de la fonctionnelle et éventuellement des gradients.

Dans ce mode de fonctionnement, le paragraphe 3.1 reste valable. Pour plus de détails, on renvoie l'utilisateur au paragraphe 7 .

4 Opérandes

4.1 Opérande UNITE_ESCL

♦ UNITE_ESCL

Numéro d'unité logique du fichier esclave, attribué dans l'interface ASTK (colonne UL). L'extension de ce fichier peut être quelconque.

4.2 Opérande RESU_EXP

♦ RESU_EXP

Nom de la liste Python de N tableaux Numeric Python contenant les N courbes expérimentales. La liste est définie préalablement sous la forme :

```
resu_exp=[    Numeric.array([  [x0,y0],
                               [x1,y1],
                                ...
                               [x n ,y n ]    ]),
          .....
              Numeric.array([  [u 0 ,v 0 ],
                               [u 1 ,v 1 ],
                               ...
                               [un,vn] ])
          ]
```

4.3 Opérande POIDS

◊ POIDS

Nom du tableau Numeric Python contenant les N poids à affecter aux N courbes expérimentales. Si non renseigné, alors la table est composée de 1. La liste est définie préalablement sous la forme :

```
POIDS= Numeric.array([p 0 ,p 1 ,...])
```

4.4 Opérande RESU_CALC

♦ RESU_CALC

Nom de la liste Python de N listes Python contenant les noms des tables et des colonnes contenant les réponses numériques correspondant aux mesures expérimentales sur lesquelles on va effectuer le recalage. Par exemple :

```
resu_calc=[['TABLE1','ABSC1','ORDO1'], ['TABLE2','ABSC2','ORDO2'],,...])
```

4.5 Opérande LIST_PARA

♦ LIST_PARA

Nom de la liste Python de P listes Python contenant les noms des variables, leurs valeurs initiales, leurs valeurs minimales et leurs valeurs maximales. Cette liste est définie préalablement sous la forme :

```
List_para=[  ['PARA1__',INI_1,MIN_1,MAX_1],
             ['PARA2__',INI_2,MIN_2,MAX_2],
             ....
             ['PARAP__',INI_P,MIN_P,MAX_P]]
```

Attention :

On demande que les noms des variables se terminent par deux blancs soulignés (par exemple : YOUN__).

Remarque :

Les bornes ne sont pas gérées par les algorithmes FMIN, FMINBFGS et FMINNCG.

4.6 Opérande LIST_DERIV

◊ LIST_DERIV

Nom du dictionnaire Python contenant les informations relatives au calcul des dérivées des paramètres dans le fichier esclave. Ce dictionnaire est défini préalablement sous la forme :

```
List_deriv= {
  'PARA1__': ['DR1_PARA1','ABS1','ORDO1'],['DR2_PARA1','ABS2','ORDO2']],
  'PARA2__': ['DR1_PARA2','ABS1','ORDO1'],['DR2_PARA2','ABS2','ORDO2']],
   ....
  'PARAP__': ['DR1_PARAP','ABS1','ORDO1'],['DR2_PARAP','ABS2','ORDO2']],
            }
```

Les DRi_PARAj sont des concepts Aster de type TABLE, extraits dans le fichier de commande esclave. Pour chaque (i,j), DRi_PARAj correspond à la table dérivée de la réponse i du paramètre j.

4.7 Opérande UNITE_RESU

◊ UNITE_RESU

Numéro d'unité logique du fichier de résultat du recalage (évolution des paramètres au cours des itérations, critères de convergence).

4.8 Opérande ITER_MAXI

◊ ITER_MAXI

Nombre d'itérations maximales de recalage.

4.9 Opérande ITER_FONC_MAXI

◊ ITER_FONC_MAXI

Nombre d'évaluations maximales de la fonctionnelle.

4.10 Opérande RESI_GLOB_RELA

◊ RESI_GLOB_RELA

Résidu global relatif du recalage.
Cette valeur est disjointe de celle renseignée pour les solveurs non linéaires STAT_NON_LINE et DYNA_NON_LINE.

4.11 Opérande PARA_DIFF_FINI

◊ PARA_DIFF_FINI

Le recalage nécessite le calcul des dérivées des réponses par rapport aux paramètres.

Dans le cas où LIST_DERIV est absent, ce calcul est réalisé par différences finies. PARA_DIFF_FINIES correspond à α dans la formule suivante :

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

4.12 Opérande GRAPHIQUE

◊ UNITE

Numéro d'unité logique des graphiques produits au cours du recalage. A chaque itération, MACR_RECAL produit N fichiers graphiques (dont le format est défini par le mot-clé PILOTE) représentant les N courbes expérimentales et calculées.

◊ PILOTE

Type d'affichage des graphiques.
Si PILOTE = 'INTERACTIF', xmgrace est ouvert de façon interactive avec le graphique. Si PILOTE vaut 'POSTSCRIPT','EPS', 'MIF', 'SVG', 'PNM', 'PNG', 'JPEG', ou 'PDF' alors xmgrace est utilisé pour générer le fichier de format correspondant, dans l'unité logique définie par UNITE.

Attention :

> *Le mode INTERACTIF n'est possible que lorsque le recalage tourne en interactif et non en batch.*

◊ FORMAT

Choix du logiciel d'affichage des courbes en mode interactif : xmgrace ou gnuplot. L'utilisation d'Xmgrace est bloquante : il faut fermer la fenêtre Xmgrace pour continuer l'exécution.

◊ AFFICHAGE

Affichage des courbes à chaque itération ou uniquement à la fin.

4.13 Opérande SUIVI_ESCLAVE

◊ SUIVI_ESCLAVE

Permet d'avoir l'écho de l'exécution du calcul esclave dans le fichier output (et dans la fenêtre « Terminal » si le calcul est lancé en interactif avec suivi.

4.14 Opérande METHODE

◊ METHODE

Méthode ou algorithme d'optimisation choisi :

1) LEVENBERG (défaut) : algorithme de Levenberg-Marquardt. Cet algorithme est celui préconisé dans les problèmes de minimisation quadratique type moindres carrés, comme des problèmes de recalage de paramètres matériaux.
2) FMIN : Nelder-Mead Simplex algorithm (n'utilise que des estimations de la fonctionnelle).
3) FMINBFGS : méthode Quasi-Newton (utilise la fonctionnelle et le gradient de la fonctionnelle).
4) FMINNCG : méthode Line-search Newton Conjugate Gradient (utilise la fonctionnelle, le gradient de la fonctionnelle et son hessien).
5) EXTERNE : cette méthode permet d'utiliser un algorithme d'optimisation externe à Aster, par exemple Matlab ou Scilab, et d'utiliser Aster uniquement pour l'estimation de la fonctionnelle et éventuellement du gradient par différences finies.

Concernant le choix de l'algorithme d'optimisation, il est fortement conseillé d'opter pour l'algorithme par défaut, **Levenberg-Marquardt**. Celui-ci est très souvent supérieur aux algorithmes FMIN* pour des problèmes de minimisation de type moindres carrés, comme le recalage de paramètres. En effet, il utilise la fonctionnelle sous sa forme vectorielle alors que les autres algorithmes utilisent une fonctionnelle scalaire, par conséquent moins riche. De plus, il utilise une méthode de contraintes actives afin de gérer des bornes sur les paramètres, alors que les autres algorithmes ne gèrent pas les bornes.

Les autres algorithmes peuvent néanmoins être utiles dans les cas où Levenberg-Marquardt est mis en difficulté. Par exemple, l'algorithme FMIN n'utilise pas de gradients, dont l'évaluation peut dans certains cas très particuliers générer des problèmes numériques (paramètres très sensibles, trop peu de valeurs expérimentales, ou autres). L'algorithme FMIN est nettement plus lent, mais pourra arriver à converger (pour faire un parallèle, la problématique est similaire à celle d'utiliser la matrice élastique à la place de la matrice tangente dans STAT_NON_LINE).

Pour l'algorithme de Levenberg-Marquardt, le document [R4.03.06] décrit plus précisément l'algorithmie mathématique mise en jeu.

Les algorithmes FMIN* ont été repris intégralement d'un module Python distribué sur Internet (http://pylab.sourceforge.net) sous licence GPL par Travis E. Oliphant, par ailleurs contributeur principal du projet Python-Scipy et responsable du module d'optimisation de Scipy. Les détails d'algorithmie et d'implémentation peuvent être trouvés sur la page http://pylab.sourceforge.net.

4.15 Opérande GRADIENT

◊ GRADIENT

Pour les méthodes FMINBFGS, FMINNCG ou EXTERNE, ce mot-clé permet d'indiquer à Aster la façon de calculer les gradients (adimensionné ou non), ou bien de ne pas les calculer.

Note : pour les algorithmes qui utilisent les gradients, ceux-ci peuvent être calculés par différences finies automatiquement par Aster, ou bien calculés dans le fichier esclave en utilisant, par exemple des calculs de sensibilité (mot-clé LIST_DERIV).

4.16 Opérande TYPE_FONCTIONNELLE

◊ TYPE_FONCTIONNELLE

Pour le mode Externe, permet d'indiquer à Aster de renvoyer la fonctionnelle sous la forme vectorielle ou sous une forme scalaire. En interne, Aster calcule une fonctionnelle vectorielle, la forme scalaire n'est que la norme 2 de la fonctionnelle vectorielle.

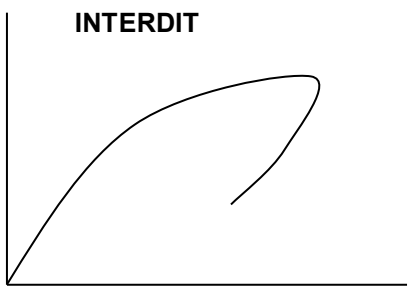
Remarque :

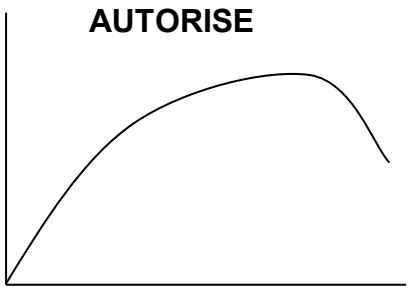
Dans les problèmes de minimisation de type moindre carrés, il est préférable d'utiliser des algorithmes capables de minimiser une fonctionnelle vectorielle, car l'information est plus riche qu'avec une fonctionnelle scalaire.

5 Précautions d'emploi

Voici un ensemble de conseils **indispensables** à la bonne utilisation du recalage.

- Les courbes expérimentales sont définies comme des tableaux à deux colonnes : une pour les abscisses et une pour les ordonnées.
- Les courbes expérimentales doivent être des fonctions : à une abscisse ne doit correspondre qu'une ordonnée. Si un courbe expérimentale comporte des cycles, (par exemple contrainte en fonction de la déformation en charge-décharge), il faut alors scinder cette courbe paramétrée en deux courbes, exprimant d'une part les abscisses, d'autre part les ordonnées de la courbe cyclique en fonction du paramètre (par exemple déformation en fonction du temps et contrainte en fonction du temps).





- On doit recaler N courbes calculées sur N courbes expérimentales.
- La première courbe calculée sera recalée sur la première courbe expérimentale, la deuxième courbe calculée sera recalée sur la deuxième courbe expérimentale, et ainsi de suite dans l'ordre renseigné pour les opérandes RESU_EXP et RESU_CALC.
- Les grandeurs calculées renseignées sous l'opérande RESU_CALC doivent être issues de POST_RELEVE_T.
- Les paramètres du recalage doivent être déclarés en bloc au début du fichier de commandes esclave. Par exemple :

```
DEBUT();
DSDE__ = 200.;
YOUN__ = 8.E4;
SIGY__ = 10.;
......
```

- Les valeurs initiales des paramètres du recalage sont celles renseignées pour l'opérande LIST_PARA et non celles présentes dans le fichier esclave de l'utilisateur.
- A chaque itération de recalage, les calculs définis dans le fichier esclave doivent converger. Dans le cadre de recalage de calculs non linéaires, il est donc fortement recommandé d'utiliser la découpe automatique du pas de temps.
- Dans le cadre de recalage de calculs non linéaires avec découpe automatique du pas de temps, il est **indispensable** de définir une liste d'archivage sous l'opérande LIST_ARCH.
- Le recalage est un moyen puissant d'obtenir des valeurs de paramètres à partir d'essais. Il n'est cependant pas miraculeux : les courbes **expérimentales** doivent contenir suffisamment d'informations pour identifier les paramètres. Il est par exemple impossible d'identifier des paramètres élastoplastiques avec un essai restant dans le domaine élastique. Les essais expérimentaux doivent donc exciter les paramètres à identifier.
- Dans le même logique, il est souhaitable que les courbes **expérimentales** contiennent des points en nombre suffisant pour bien décrire l'action des paramètres à identifier.
- Enfin, dans le cas de l'utilisation de plusieurs courbes expérimentales, le fait qu'elles aient le même nombre de points équilibre l'information qu'elles apportent.

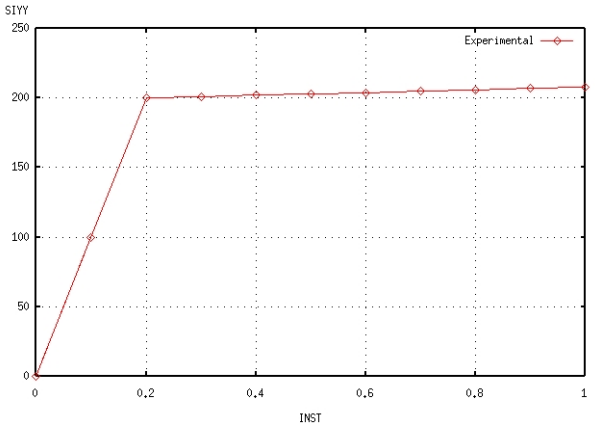
6 Exemple d'utilisation

6.1 Identification des paramètres d'une loi de comportement élastoplastique sur un essai de traction

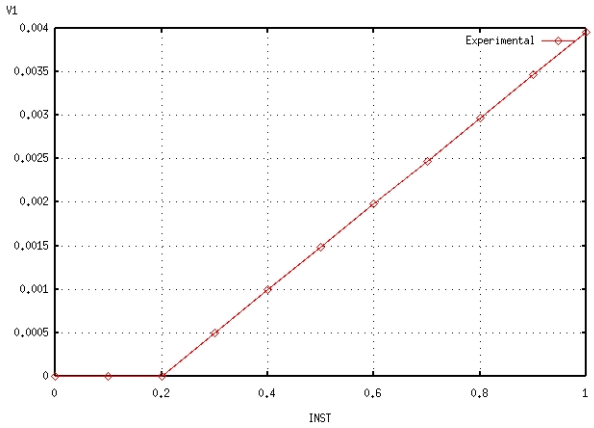
Cet exemple est traité par le test ZZZZ159A [V1.01.159].

6.1.1 Position du problème

On dispose des résultats d'un essai de traction. Il s'agit de l'évolution de la contrainte SIYY au cours du temps ainsi que de l'évolution de la déformation plastique cumulée au cours du temps.



Contrainte SIYY



Déformation plastique cumulée

On désire recaler sur ces essais le module de Young, la limite d'élasticité et la pente d'écrouissage d'une loi de comportement élastoplastique à écrouissage isotrope linéaire.

6.1.2 Mise en données

6.1.2.1 Expérience

On commence tout d'abord par définir nos résultats d'essais. Ils consistent en deux courbes que l'on définit comme suit.

```
experience=[   Numeric.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 ]]),
               Numeric.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 ]]) ]
```

`experience` est donc le nom d'une liste Python (définie entre crochets) de 2 tableaux Numeric Python.

6.1.2.2 Calcul

On écrit ensuite le fichier de commandes Aster esclave modélisant cet essai de traction où vont apparaître nos 3 paramètres ainsi que les deux courbes à recaler.

```
DEBUT();
# AFFECTATION DES VALEURS DES PARAMETRES A RECALER
# LES VALEURS RENSEIGNEES ICI SONT SANS IMPORTANCE
# SEULES COMPTENT LES VALEURS RENSEIGNEES DANS LE FICHIER MAITRE
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__,),);
........

EVOL=STAT_NON_LINE(CHAM_MATER=CHMAT,
                   MODELE=MO,
                   ARCHIVAGE=_F(LIST_INST=INSTANTS,
                                ARCH_ETAT_INIT='OUI',),
                   CONVERGENCE=_F(ITER_GLOB_MAXI=10,
                                  RESI_GLOB_RELA=1.E-05,),
                   COMP_INCR=_F(RELATION='VMIS_ISOT_LINE',),
                   INCREMENT=_F(LIST_INST=INSTANTS,
                                SUBD_PAS=4,
                                SUBD_METHODE=’EXTRAPOLE’,
                                SUBD_PAS_MINI=1.E-05,),
                   NEWTON=_F(REAC_ITER=1,
                             REAC_INCR=1,),
                   EXCIT=_F(CHARGE=TRACTION,
                            FONC_MULT=RAMPE,),);
.......

# EXTRACTION DE LA REPONSE SIGMAYY(T)
REPONSE1=POST_RELEVE_T(ACTION=_F(OPERATION='EXTRACTION',
                         INTITULE='SIGYY',
                         RESULTAT =EVOL,
                         NOM_CHAM ='SIEF_ELNO_ELGA',
                         NOM_CMP  = 'SIYY',
                         GROUP_NO = 'A',),);

# EXTRACTION DE LA REPONSE EPSP(T)
REPONSE2=POST_RELEVE_T(ACTION=_F(OPERATION='EXTRACTION',
                         INTITULE='V1',
                         RESULTAT =EVOL,
                         NOM_CHAM ='VARI_ELNO_ELGA',
                         NOM_CMP  = 'V1',
                         GROUP_NO = 'A',),);


FIN();
```

Il nous faut maintenant définir dans le fichier maître les valeurs initiales et les plages de variations de nos paramètres. On désire :

1.E5	<	Module d’Young initial = 1.E5	<	5.E5
5.	<	Limite d’élasticité initiale = 30.	<	500.
1.E3	<	Module d’écrouissage initial = 1.E3	<	1.E4

Ce que l’on écrit sous la forme=

```
Parametres =[['YOUN__',100000.,50000.,500000.],['DSDE__',1000.,500.,10000.],
```

```
                ['SIGY__',30.,5.,500.]]
```

Enfin il nous reste à définir dans le fichier maître les grandeurs à extraire du fichier de commandes esclave ci-dessus. Nous désirons d'une part extraire la colonne INST et la colonne SIYY de la table REPONSE1 et d'autre part la colonne INST et la colonne V1 de la table REPONSE2. Nous l'écrivons :

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

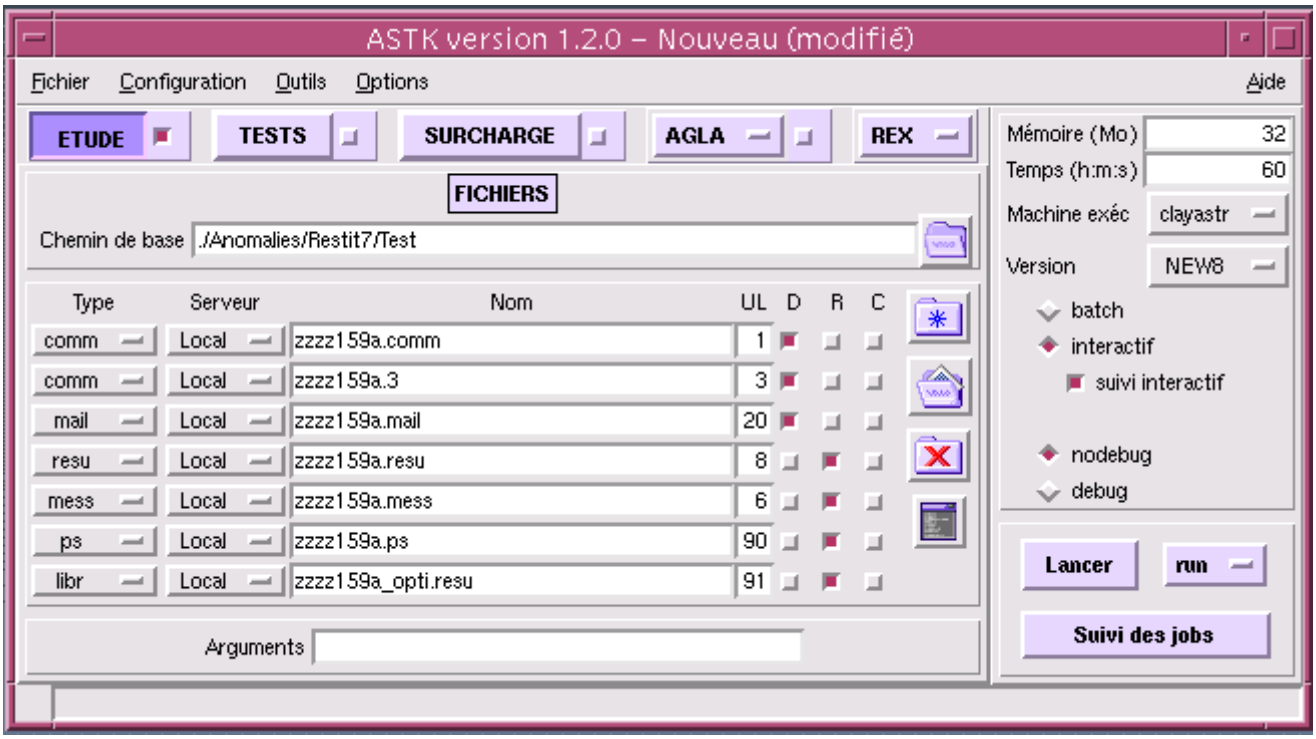
6.1.2.3 MACR_RECAL

Nous renseignons maintenant ces informations dans le corps de MACR_RECAL :

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

6.1.2.4 ASTK

On définit enfin le profil d'étude suivant :



6.1.3 Résultats

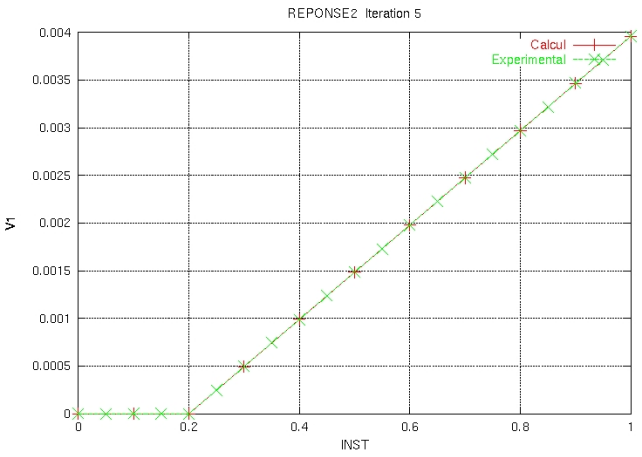
Une fois l'étude réalisée, le fichier de résultat du recalage ZZZZ159_opti.resu contient les informations suivantes :

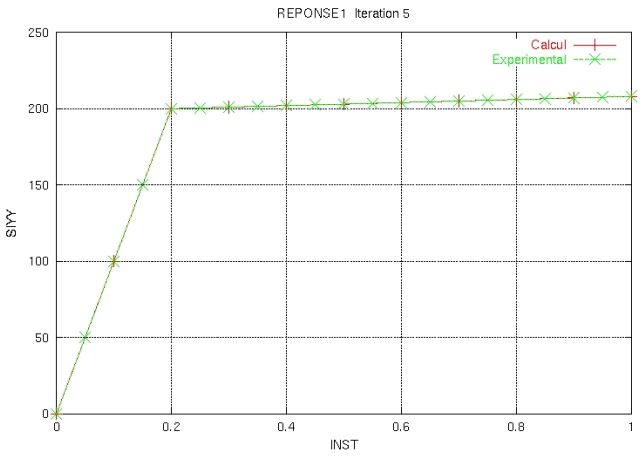
```
Calcul de la sensibilité par rapport à = YOUN__ DSDE__ SIGY__
=========================================================
Iteration 0 =

=> Fonctionnelle = 1.0
=> Résidu        = 1.0
=> Paramètres    =
```

```
         YOUN__ = 100000.0
         DSDE__ = 1000.0
         SIGY__ = 30.0
 ============================================================


 Calcul de la sensibilité par rapport à = YOUN__ DSDE__ SIGY__
 ============================================================
 Iteration 1 =

 => Fonctionnelle = 0.259742161795
 => Résidu        = 0.30865397471
 => Paramètres    =
         YOUN__ = 300857.888503
         DSDE__ = 9135.12770111
         SIGY__ = 152.548047532
 ============================================================


 Calcul de la sensibilité par rapport à = YOUN__ DSDE__ SIGY__
 ============================================================
 Iteration 2 =

 => Fonctionnelle = 0.0757636994765
 => Résidu        = 0.473053125246
 => Paramètres    =
         YOUN__ = 157723.378846
         DSDE__ = 2022.7431335
         SIGY__ = 213.155325073
 ============================================================


 Calcul de la sensibilité par rapport à = YOUN__ DSDE__ SIGY__
 ============================================================
 Iteration 3 =

 => Fonctionnelle = 0.00190706595529
 => Résidu        = 0.0520849911718
 => Paramètres    =
         YOUN__ = 192302.166747
         DSDE__ = 895.845518907
         SIGY__ = 203.753909707
 ============================================================


 Calcul de la sensibilité par rapport à = YOUN__ DSDE__ SIGY__
 ============================================================
 Iteration 4 =

 => Fonctionnelle = 2.70165453323e-06
 => Résidu        = 0.00172172540305
 => Paramètres    =
         YOUN__ = 199801.572817
         DSDE__ = 1928.08902726
         SIGY__ = 200.274590793
 ============================================================

 Calcul de la sensibilité par rapport à = YOUN__ DSDE__ SIGY__
 ============================================================
 Iteration 5 =

 => Fonctionnelle = 2.65431115925e-12
```

```
=> Résidu        = 1.83121468206e-06
=> Paramètres    =
         YOUN__ = 199999.975047
         DSDE__ = 1999.86955101
         SIGY__ = 200.000462987
============================================================

============================================================
                    CONVERGENCE ATTEINTE
============================================================

Valeurs propres du Hessien:
[  7.17223479e+00   3.67264061e-01   6.25194340e-04]

Vecteurs propres associés:
[[ 0.98093218 -0.00549396 -0.19427266]
 [-0.19418112 -0.06940835 -0.97850712]
 [ 0.00810827 -0.9975732   0.0691517 ]]

              --------

On peut en déduire que :

Les combinaisons suivantes de paramètres sont prépondérantes pour votre
calcul :

   1) +9.8E-01 * YOUN__ -1.9E-01 * DSDE__
      associée à la valeur propre 7.2E+00


Les combinaisons suivantes de paramètres sont insensibles pour votre
calcul :

   1) -1.9E-01 * YOUN__ -9.8E-01 * DSDE__
      associée à la valeur propre 6.3E-04
```

Et le fichier POSTSCRIPT ZZZZ159.ps contient :





7 Utilisation du mode EXTERNE

7.1 Avertissement

Nous attirons l'attention sur le fait que ce mode de fonctionnement est à réserver à un usage avancé. La plupart des cas devraient être traités avec les algorithmes fournis dans la commande MACR_RECAL, et notamment l'algorithme Levenberg-Marquardt, qui est le plus adapté aux problèmes de recalage de paramètres.

Ce mode de fonctionnement nécessite un logiciel externe à Aster pour effectuer l'optimisation (code Python, Matlab, Scilab, logiciel type boite noire, etc.). De plus, il est nécessaire d'avoir quelques compétences Python.

Enfin, l'utilisation du mode EXTERNE sort du périmètre AQ d'Aster, et ne devrait pas être utilisé pour des études IPS.

7.2 Méthodologie

7.2.1 Principe

Dans ce mode d'utilisation, Aster est uniquement utilisé pour l'évaluation de la fonctionnelle pour un logiciel d'optimisation qui est complètement externe à Aster.

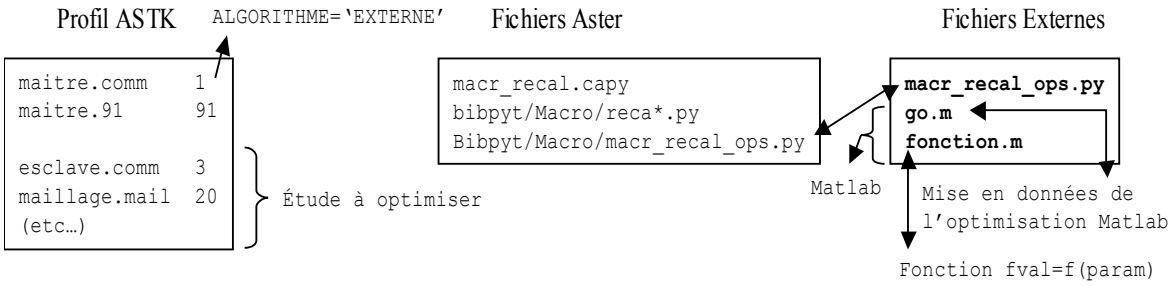


Figure 7.2.1-a: MACR_RECAL « Algorithme externe »

Généralement, les logiciels d'optimisation demandent que l'utilisateur écrive une procédure pour le calcul de la fonctionnelle : f = F(param).

Si le logiciel autorise la lecture/écriture de fichiers et l'exécution de code externe (commande « system » ou autres), alors il est potentiellement utilisable avec MACR_RECAL. On va encapsuler l'appel à la procédure Python macr_recal_ops.py ainsi que l'écriture du fichier des paramètres et la relecture du fichier de la valeur de la fonctionnelle dans la routine de calcul de F.

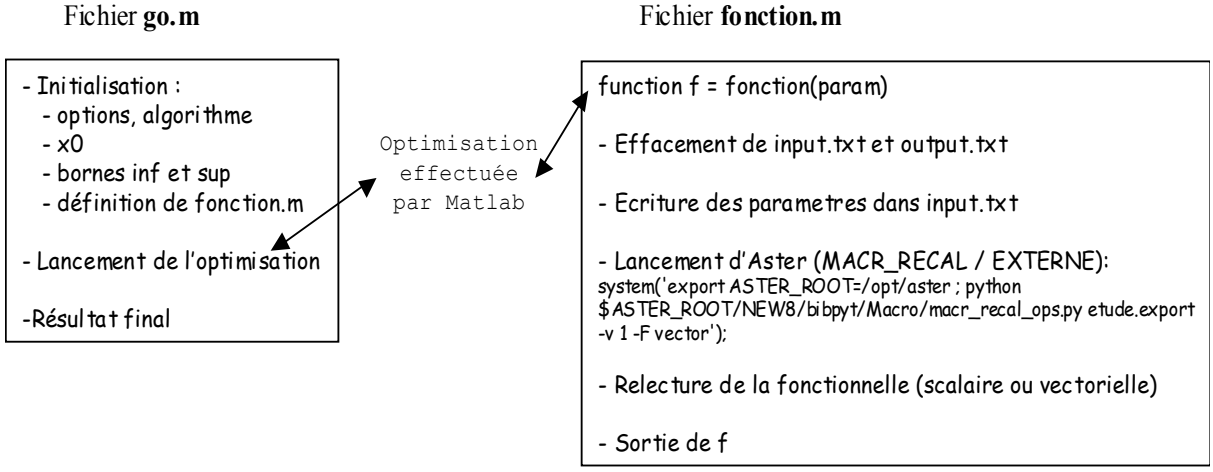


Figure 7.2.1-b: MACR_RECAL « Principe du mode EXTERNE », exemple MATLAB

Le logiciel d'optimisation lance Aster à chaque évaluation de la fonctionnelle. La routine d'évaluation de la fonctionnelle passe par des fichiers textes pour envoyer à Aster les paramètres à utiliser et pour récupérer la valeur de la fonctionnelle. La routine Python `macr_recal_ops.py` est donc appelable indépendamment d'Aster et fait le lien entre le logiciel d'optimisation et Aster (récupération du fichier des paramètres, remise en forme du fichier esclave, lancement d'Aster, récupération de la fonctionnelle).

En pratique, il faut disposer d'un profil ASTK typique d'une utilisation de `MACR_RECAL` (fichier de commande maître contenant les directives de `MACR_RECAL`, fichier de commande esclave et maillage, etc…). Ce profil doit être complètement fonctionnel. A partir de ce profil fonctionnel, on change uniquement le mot-clé `ALGORITHME = 'EXTERNE'` pour indiquer à `MACR_RECAL` de ne faire qu'une évaluation de la fonctionnelle (c'est à dire de lancer une fois l'étude esclave).

Sous ce mode `EXTERNE`, il y a en fait deux fonctionnements possibles.
La routine `macr_recal_ops.py` se charge de récupérer les paramètres depuis un fichier `input.txt`, et tente d'évaluer le fichier maître en Python pur, jusqu'à la commande `MACR_RECAL`.

1. Si le fichier maître est un fichier maître « classique » de `MACR_RECAL`, c'est à dire qu'il ne contient que du Python et les commandes `DEBUT`, `MACR_RECAL` et `FIN`, l'évaluation en Python fonctionnera et la routine `macr_recal_ops.py` se chargera de lancer le calcul esclave sous la forme d'une exécution d'Aster (en gros, exécution de `as_run export_esclave.export`).
2. Si l'évaluation en Python échoue, alors la routine `macr_recal_ops.py` lance une première exécution d'Aster standard avec le fichier maître en fichier .comm, après avoir modifié les paramètres en entête du fichier maître. Durant cette exécution d'Aster, la commande `MACR_RECAL` est rencontrée et une deuxième exécution d'Aster est lancée pour le calcul esclave.

Afin d'éviter une baisse de performance due à une exécution d'Aster supplémentaire, on a donc intérêt à construire un fichier maître ne contenant aucune commande Aster à part `DEBUT`, `MACR_RECAL` et `FIN`. En revanche, ce fichier esclave peut contenir du Python (lecture des courbes expérimentales, etc..).

A ce jour, ce mode de fonctionnement a été testé avec plusieurs logiciels d'optimisation (la *toolbox* d'optimisation de Matlab, la *toolbox* Tomlab pour Matlab, et le module d'optimisation de Python-Scipy) mais il est possible d'utiliser à peu près n'importe quel logiciel à partir du moment où ce logiciel autorise l'exécution externe d'un script. On pense notamment à Zopt (Zebulon), SiDoLo.

7.2.2 Définition du fichier maître de MACR_RECAL

Le fichier maître, c'est-à-dire le fichier `.comm` contenant la commande `MACR_RECAL`, doit être légèrement modifié. Il faut mettre dans les arguments de la commande `MACR_RECAL` :

le mot-clé `ALGORITHME = 'EXTERNE'` dans `MACR_RECAL`

Le fichier maître sera évalué en Python (c'est-à-dire en dehors d'Aster). Si ce fichier ne contient que du Python et les commandes Aster `DEBUT`, `MACR_RECAL` et `FIN`, alors l'exécution Aster de l'étude esclave se lancera.

Si l'évaluation en Python de fichier maître échoue, alors Aster est lancé « normalement » avec le profil fourni. Celui-ci doit être un profil ASTK d'utilisation standard de `MACR_RECAL` (fichier maître, fichiers pour l'étude esclave avec maillage, etc..), et il devra respecter les points suivants :

1. les paramètres initiaux doivent être donnés au début du fichier maître `.comm` (celui contenant l'appel à `MACR_RECAL`) sous la forme d'une variable `_PARAM_` :

```
_PARAM_ = [10., 20., 30.]
```

2. puis utilisé, par exemple :

```
parametres=[['P1__',_PARAM_[0],5.,50.],['P2__',_PARAM_[1],5.,100.],
            ['P3__',_PARAM_[2],5.,50.]]
```

7.2.3 Utilisation du fichier de lancement externe macr_recal_ops.py

Le fichier `macr_recal_ops.py` est le fichier « `ops` » de la macro-commande Aster `MACR_RECAL`. Il peut de plus s'exécuter de manière autonome avec quelques paramètres optionnels sur la ligne de commande :

```
usage: /opt/aster/NEW9/bibpyt/Macro/macr_recal_ops.py fichier_export [options]

options:
  -h, --help            show this help message and exit
  -i INPUT, --input=INPUT
                        fichier contenant les parametres
  -o OUTPUT, --output=OUTPUT
                        fichier contenant la fonctionnelle
  -g OUTPUT_GRAD, --output_grad=OUTPUT_GRAD
                        fichier contenant le gradient
  -p PREFIX_GRAPH, --prefix_graph=PREFIX_GRAPH
                        prefixe des fichiers contenant les courbes
  -v INFO, --info=INFO  niveau de message (-1, 0, 1, 2)
  -f FOLLOW_OUTPUT, --follow=FOLLOW_OUTPUT
                        affiche ou non l'output du fichier Aster (True/False)
  -F OBJECTIVE, --objective=OBJECTIVE
                        type de la fonctionnelle (float/vector)
  -G GRADIENT, --gradient=GRADIENT
                        calcul du gradient par Aster (no/normal/adim)
  -d DISPLAY, --display=DISPLAY
                        renvoi du DISPLAY (pour que la creation des courbes
                        soit moins genante)
```

Cette procédure a besoin :
1. du fichier .export de l'étude ASTK définie précédemment (l'étude standard de MACR_RECAL)
2. d'un fichier texte contenant la liste des paramètres

A partir de ces deux fichiers, elle lance un calcul Aster pour les paramètres spécifiés (ou deux calculs Aster, voir plus haut) et génère un fichier texte contenant uniquement la valeur de la fonctionnelle (scalaire ou vectorielle).

Cette procédure peut être lancée sans argument. Elle va alors chercher **dans le répertoire courant** des fichiers par défaut :
1. s'il n'y a qu'un seul fichier .export dans le répertoire courant, celui-ci sera utilisé (dans le cas contraire la procédure s'arrête en erreur)
2. le fichier d'entrée par défaut sera cherché sous le nom « input.txt »
3. le fichier de sortie sera généré avec le nom « output.txt »

Les arguments –i (--input) et -o (--output) permettent de préciser les fichiers.

L'argument –g (--output_grad) : permet de préciser le fichier texte dans lequel sera écrit le gradient, si celui-ci est demandé (argument -G).

D'autres arguments sont disponibles :
1. -v (--info) : défini le niveau de message ; 0=muet, 1 ou 2 ;
2. -f (--follow) : signifie *follow_output*, permet de suivre de façon interactive l'exécution Aster et la sous-exécution du calcul Aster fils. Note : Si le niveau de message est 0 (muet) alors le suivi est automatiquement désactivé ;
3. -F (--objective) : permet de préciser si la fonctionnelle doit être retournée sous la forme vectorielle ou scalaire ;

4. -G (--gradient) : permet de préciser si on ne veut pas qu'Aster calcule et renvoie les gradient (no), et si on veut les gradients, de préciser si on les veut adimensionné ou non ;
5. -d (--display) : permet d'exporter le DISPLAY vers un écran défini, si on demande la visualisation des courbes dans MACR_RECAL ;
6. -p (--prefix_graph) : défini le préfixe des fichiers graphiques ;

Remarque :

Note pour follow_output : pour le moment, il doit également être activé avec le mot-clé SUIVI_ESCLAVE='OUI' dans la commande MACR_RECAL, sinon on n'obtient que le suivi du calcul maître. De plus, il faut mettre le mot-clé IMPR_MACRO='OUI' dans DEBUT.

Par défaut, un fichier `aster.output` est crée dans le répertoire d'exécution de la procédure, permettant d'avoir accès à l'output du dernier calcul Aster.

Enfin, il faut configurer la variable d'environnement ASTER_ROOT (le chemin de base d'Aster) pour pouvoir lancer en python le fichier `macr_recal_ops.py`.

Le bon fonctionnement de la procédure `macr_recal_ops.py` peut être vérifié manuellement, en générant un fichier d'entrée avec des valeurs de paramètres et en lançant manuellement la procédure. Exemple sur la machine Aster :

```
cd repertoire
export ASTER_ROOT=/aster
echo "10., 20., 30. " >input.txt
python $ASTER_ROOT/STA8/bibpyt/Macro/macr_recal_ops.py etude.export -v 2
cat ouput.txt
```

Le calcul Aster devrait se lancer et terminer convenablement. Un fichier output.txt doit être généré dans le répertoire courant.

Le cas-test zzzz159f illustre l'utilisation externe de `MACR_RECAL` : dans ce test Aster, on définit une fonction f et on rappelle une fois Aster pour simuler une itération de recalage.

7.3 Exemple 1 : module d'optimisation de MATLAB©

Si on reprend le principe et qu'on l'applique à un algorithme d'optimisation qui serait écrit sous MATLAB, on doit écrire deux fichiers `go.m` et `fonction.m` qui effectuent les actions décrites ci-dessous :

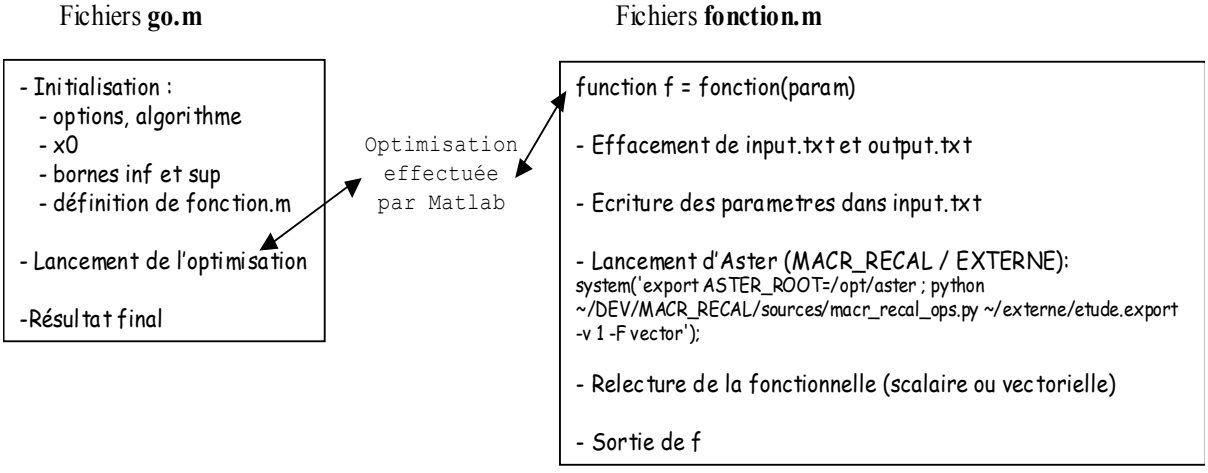


Figure 7.3-a: MACR_RECAL « Principe du mode EXTERNE » appliqué à MATLAB©

7.3.1 Module d'optimisation de MATLAB et calcul Aster en local

L'exemple suivant utilise le module d'optimisation de Matlab et effectue les calculs Aster sur le serveur local.

Fichier go.m (initialisation et lancement de l'optimisation)

```
clear all;
format long;

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

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

% Params : DSDE__, SIGY__, YOUN__
x0 = [ 1000., 30., 100000. ];
lb = [ 500., 5., 50000. ];
ub = [ 10000., 500., 500000. ];

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

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

Fichier fonction.m (calcul de F(param)) / version Linux

```
function f = fonction(x)

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

% Ecriture des parametres
dlmwrite('input.txt', x, 'precision', '%.20f');

% Local
iret = system('export ASTER_ROOT=/opt/aster ; python
$ASTER_ROOT/bibpyt/Macro/macr_recal_ops.py etude.export -v 1 -F vector');

%iret = system('export ASTER_ROOT=/opt/aster ; python
$ASTER_ROOT/bibpyt/Macro/macr_recal_ops.py etude.export -v 1 -F float');

f = dlmread('output.txt', ',', 0, 0);   % relit un scalaire ou un vecteur
```

Dans `go.m`, on donne deux exemples d'algorithmes. L'algorithme `lsqnonlin` minimise une fonctionnelle vectorielle et il faut donc décommenter la première ligne « iret= ». Cet algorithme est basé sur Levenberg-Marquardt à contraintes actives, et est donc similaire à celui implanté dans Aster.

Remarque :

Il est important de noter la présence de la commande `format long` et de l'argument `precision` pour la commande `dlmwrite` . Ceci permet de travailler avec un nombre suffisant de chiffres significatifs et de ne pas s'arrêter si la tolérance sur la fonction n'est plus respectée.

Remarque 2 :

Lorsque le logiciel d'optimisation n'est pas capable de gérer les codes retour de l'exécution externe, il faut faire attention à arrêter proprement la procédure en cas d'arrêt anormal du calcul Aster. C'est la raison pour laquelle on efface les fichiers `input.txt` *et* `output.txt` *juste après leur utilisation, afin de ne pas tomber dans des boucles sans fin.*

7.3.2 Module d'optimisation de MATLAB sous Windows et calcul Aster sur un serveur Linux distant

Cet exemple est le plus compliqué qu'on peut imaginer : le logiciel d'optimisation ne tourne pas sur la même machine que les calculs Aster, et les deux utilisent des OS différents.

Fichier fonction.m (calcul de F(param)) / version Windows

```
function f = fonction(x)

% Windows
dos('del input.txt');
dos('del output.txt');

% Ecriture des parametres
dlmwrite('input.txt', x, 'precision', '%.20f');

% Distant Windows -> Linux
system('pscp -i C:\cle.ppk input.txt assire@cli75ca:~/externe/ >nul');

system('plink -i C:\cle.ppk assire@cli75ca 'export ASTER_ROOT=/opt/aster ; cd ~/externe/ ;
python /opt/aster/NEW8/bibpyt/Macro/macr_recal_ops.py etude.export --info=1');

system('pscp -i C:\cle.ppk assire@cli75ca:~/externe/output.txt >nul');

f = dlmread('output.txt', ',', 0, 0);   % permet de relire un scalaire ou un vecteur
```

Le fichier go.m est un fichier standard de définition de lancement d'un problème d'optimisation sous Matlab (identique au paragraphe précédent).

Le fichier fonction.m prend en argument le vecteur des paramètres x et renvoie une valeur scalaire f.

Ce fichier peut s'utiliser avec un serveur Aster local ou distant. Dans ce dernier cas, les 3 commandes exécutées par `system` gèrent respectivement la recopie du fichier `input.txt` sur le serveur distant, le lancement distant d'Aster et la recopie du fichier résultat en local. Dans cet exemple, on utilise les deux exécutables `pscp.exe` et `plink.exe` qui font partie de la distribution libre Putty (http://www.chiark.greenend.org.uk/~sgtatham/putty/). Pour l'identification sur le compte distant, on utilise un système de clés SSH (fichier .ppk) mais on peut également spécifier le mot de passe du compte distant avec l'argument –pw password (à utiliser à la place de -i clé).

7.4 Exemple 2 : module d'optimisation de Scipy

Le fichier suivant permet d'utiliser le module d'optimisation de Scipy et l'évaluation de la fonctionnelle par Aster.

Fichier optim.py

```
import numpy
from numpy import Inf
from scipy import optimize
import os, sys

nom_pro = 'OPTIMISATION'

# -------------------- PARAMETRES UTILISATEURS --------------------
```

```
fichier_in  = 'input.txt'
fichier_out = 'output.txt'
commande          =          'export          ASTER_ROOT=/opt/aster          ;          python
$ASTER_ROOT/NEW8/bibpyt/Macro/macr_recal_ops.py'

# -------------------- FIN PARAMETRES UTILISATEURS --------------------


def UTMESS(code,sprg,texte):
   fmt='\n <%s> <%s> %s\n\n'
   print fmt % (code,sprg,texte)
   if code=='F': sys.exit()

# ------------------------------------------------------------
def fonction(val):
   """
      Calcul de F
   """

   # Menage
   for fic in ['input.txt', 'output.txt']:
      try:    os.remove(fic)
      except: pass

   # Ecriture du fichier des parametres
   txt = ' '.join( [ str(x) for x in val ] )
   try:
      f=open(fichier_in, 'w')
      f.write(txt)
      f.close()
   except:
         UTMESS('F', nom_pro, "Probleme : impossible d'ecrire le fichier des parametres :
\n"+fichier_in)

   # Execution de Code_Aster
   cmd = 'python ' + commande
   iret = os.system(cmd)
   if iret!=0: UTMESS('F', nom_pro, "Probleme lors de l'execution de la commande : \n" + cmd)

   # Lecture du fichier de F
   try:
      f=open(fichier_out, 'r')
      txt = f.read()
      f.close()
      fval = float(txt)
   except:
      UTMESS('F', nom_pro, "Probleme : impossible de lire le fichier de sortie)

   return fval


# ------------------------------------------------------------

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


grad        = ()

gtol        = 1e-5
disp        = 1
maxiter     = 100
maxfun      = 1000
full_output = 1
retall      = 0
callback    = None

avextol     = 1e-5
xtol        = 1e-4
ftol        = 1e-4

_epsilon    = numpy.sqrt(numpy.finfo(float).eps)

# Choisir un des algo suivants

#retval  =  optimize.fmin  (fonction,  x0,  args=(),  xtol=xtol,  ftol=xtol,  maxiter=maxiter,
maxfun=maxfun, full_output=full_output, disp=disp, retall=retall, callback=callback)
```

```
#retval = optimize.fmin_bfgs (fonction, x0, fprime=None, args=(), gtol=gtol, norm=Inf,
maxiter=maxiter, full_output=full_output, disp=disp, retall=retall, callback=callback)

#retval = optimize.fmin_cg (fonction, x0, fprime=None, args=(), gtol=gtol, norm=Inf,
epsilon=_epsilon, maxiter=maxiter, full_output=full_output, disp=disp, retall=retall,
callback=callback)

#retval = optimize.fmin_ncg (fonction, x0, fprime, fhess_p=None, fhess=None, args=(),
avextol=avextol, maxiter=maxiter, full_output=full_output, disp=disp, retall=retall,
callback=callback)

#retval = optimize.fmin_powell (fonction, x0, args=(), xtol=xtol, ftol=xtol, maxiter=maxiter,
maxfun=maxfun, full_output=full_output, disp=disp, retall=retall, callback=callback)

print retval

#(params, fopt, func_calls, grad_calls, warnflag) = retval
```

7.5 Exemple 3 : plate-forme d'optimisation Tomlab

Tomlab (http://www.tomlab.biz) est une plate-forme commerciale regroupant un nombre important d'algorithmes d'optimisation. Tomlab est utilisable sous Matlab et LabView. Le fondateur de cette société, Kenneth Holmström, est très actif dans le domaine de la recherche en optimisation, et a encore des charges académiques (Professor in Optimization, Mälardalen University, Department of Mathematics and Physics, P.O. Box 883, SE-721 23 Västeras, Sweden).

Un des intérêt de Tomlab est de regrouper pratiquement tous les algorithmes connus pour tous les types de problèmes d'optimisations. Toutes les librairies open source sont intégrées (MINOS, etc…), des librairies commerciales (KNITRO, CPLEX, etc..) et tous les algorithmes dignes d'intérêt qui ont fait l'objet d'une publication ont été recodés dans Tomlab.

L'autre intérêt est de proposer un format d'entrée unique pour tous les algorithmes. Ceci rend le prototypage et le changement d'algorithme très simple.

Son défaut principal est d'être commercial, en plus de nécessiter un logiciel commercial (Matlab ou LabView).

Fichier optim.m (initialisation et lancement de l'optimisation)

```
clear all
close all hidden
tic

%pack                % Libere plus de memoire temporaire
%warning off         % Enleve les warning

format long;

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

Name = 'probleme1';
fLowBnd = -10000000.;

% Params :  DSDE__, SIGY__,   YOUN__
x_0     = [  1000.,    30.,  100000. ];
x_L     = [   500.,     5.,   50000. ];
x_U     = [ 10000.,   500.,  500000. ];
```

```
calc_f  = 'fonction';
calc_g  = [];
calc_H  = [];

Prob = conAssign(calc_f, calc_g, calc_H, [], x_L, x_U, Name, x_0,...
[], fLowBnd, [], [], [], [], [], [], [], [], []);
Prob.Warning   = 0; % Turning off warnings.
Prob.PriLevOpt = 3;

% Result = tomRun('ucSolve', Prob, 2);     % Ignores constraints.
% Result = tomRun('conopt', Prob, 2);      % vecteur
% Result = tomRun('snopt', Prob, 2);       % scalaire
% Result = tomRun('minos', Prob, 2);       % scalaire
% Result = tomRun('clsSolve', Prob, 2);
% Result = tomRun('nlssol', Prob, 2);
Result = tomRun('lqsnonlin', Prob, 2);   % vecteur


% Prob.KNITRO.options.MAXIT = 200; % Setting maximum number of iterations
% Prob.KNITRO.options.FEASIBLE = 1; % Select feasible KNITRO
% Prob.optParam.MaxIter = 200;
% Result = tomRun('knitro',Prob,2)

PrintResult(Result);
```

Le fichier fonction.m est identique à celui utilisé pour la *toolbox* optimisation de Matlab.

Fichier fonction.m (calcul de F(param)) / version Linux

```
function f = fonction(x)

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

% Ecriture des parametres
dlmwrite('input.txt', x, 'precision', '%.20f');

% Local
iret      =      system('export      ASTER_ROOT=/opt/aster      ;      python
$ASTER_ROOT/bibpyt/Macro/macr_recal_ops.py etude.export -v 1 -F vector');

%iret      =      system('export      ASTER_ROOT=/opt/aster      ;      python
$ASTER_ROOT/bibpyt/Macro/macr_recal_ops.py etude.export -v 1 -F float');

f = dlmread('output.txt', ',', 0, 0);   % relit un scalaire ou un vecteur
```