L’optimisation dans Cast3M

1 Introduction

Au fur et à mesure des progrès de la simulation numérique, le problème de l’optimisation prend de plus en plus le devant de la scène. En mécanique des structures, non seulement on veut vérifier la bonne tenue de la structure sous différents chargements mais en plus on aimerait qu’elle soit la plus légère possible ou la plus endurante…
De même, les lois de comportement des matériaux sont de plus en plus sophistiquées, et l’identification de tous les paramètres d’une loi relève des techniques d’optimisation.
Dans un premier temps, le problème de l’identification sera examiné et les solutions proposées par Cast3M seront discutées. Ensuite nous regarderons le problème, plus général, de minimisation sous contraintes et montrerons une méthode qui peut être mise en œuvre dans Cast3M.

2 Identification

Soit les k solutions (G1,….Gn) du problème pour un nombre n de jeux des k variables $X^i= (X^i_1,…X^i_k)$. On cherche les p paramètres (ou coefficients) d’une fonction F qui permettent de minimiser l’écart entre les solutions connues (G) et les solutions fournies par la fonction F. Il est pratique de prendre pour mesure globale de l’écart la somme des carrées des différences de chaque solution connue avec la solution calculée correspondante.
Notons $A=(A_1,….A_p)$ les paramètres de la fonction F, on cherche alors à minimiser la fonction S tel que :

$$S = \sum_{i=1,n} \left(Gi - F(A, X^i) \right)^2$$

Le minimum de cette fonction est atteint lorsque les p dérivées partielles de S par rapport aux p inconnues A_j sont nulles. Ceci amène un système de p équations à p inconnues à résoudre.

2.1 Opérateur MOCA

Dans le cas particulier où F est linéaire par rapport aux inconnues A_j la dérivée partielle de G par rapport à une inconnue A_j quelconque est aussi une fonction linéaire. Il faut alors résoudre un système de p équations linéaires à p inconnues.

La jème équation s'écrit :

$$0 = \sum_{i=1,n} (G_i - F(A^0, X^i)) * \dot{F}(A^0, X^i)_j$$

Si F est linéaire par rapport à A_j, $\dot{F}$ est une constante et l'équation est bien linéaire par rapport aux inconnues A.

L'opérateur MOCA (pour MOindre CArrés) résout le système d'équations linéaires et fournit la solution A^s qui minimise la fonction S.

Si les solutions Gi sont plus ou moins fiables on peut leur attribuer un poids différent pour privilégier les solutions qui semblent les plus sures. La fonction S à minimiser devient alors :

$$S = \sum_{i=1,n} \left(Gi - F(A, X^i)\right)^2 * Poids_i$$

Les données de l'opérateur sont :

- A^0 liste des valeurs initiales des p inconnues
- G liste des valeurs des n solutions connues
- $F(A^0,X)$ liste des n valeurs de la fonctions F pour les valeurs A^0 des inconnues et pour les n jeux de variables X
- $\dot{F}(A^0, X^i)_j$ P listes de valeurs donnant chacune les coefficients de la dérivée partielle pour chacun des n jeux de variables X
- POIDS suivi de Poids liste des valeurs des n poids à affecter aux n solutions connues

Remarque : si toutes les dérivées partielles par rapport à une inconnue sont nulles alors cette inconnue n'intervient pas. Cet opérateur sera généralement appelé par une procédure de minimisation qui pourra utiliser son propre algorithme.

2.2 Opérateur LEVM

La méthode de Levenberg-Marquardt [0] [0] est plus générale, elle ne présuppose pas que la fonction F soit linéaire par rapport aux inconnues A. De plus des contraintes sur le domaine

de définition des inconnues A pourront être ajoutées. Cette méthode, très classique, se trouve dans tous les manuels de minimisation. C'est une méthode itérative et la détermination du paramètre qui contrôle le choix de la direction de descente et du rayon de confiance est faite simplement en cherchant la plus grande valeur, parmi les puissances de 10, qui permet d'avoir une solution meilleure que le point de départ. D'une itération sur l'autre la valeur de ce paramètre peut grandir d'un cran ou diminuer si besoin est.
La qualité essentielle de cette méthode est sa robustesse, même si elle n'est pas toujours très rapide. Elle est écrite dans Cast3M dans le cas où il n'y a qu'une variable X qui est appelée l'abscisse. Les valeurs de la fonction G pour ces abscisses s'appellent les ordonnées.

Les données de l'opérateur sont :

- Liste des abscisses
- Liste des ordonnées
- Liste des valeurs des n poids à affecter aux solutions connues
- A^0 liste des valeurs initiales des p inconnues
- PRO1 nom d'une procédure gibiane qui calcule la fonction F pour les n abscisses et les dérivées partielles par rapport aux p paramètres aux n abscisses

Remarque : Il aurait été possible d'écrire cet opérateur à l'aide d'une procédure qui aurait appelé l'opérateur MOCA.

2.3 Procédure AJUSTE

Cette procédure cherche à tirer partie de la connaissance de la dépendance linéaire ou non de la fonction F par rapport aux inconnues A. Pour les inconnues entraînant une dépendance linéaire une résolution de type moindres carrés est faite et pour les autres inconnues un algorithme fondé sur la direction de plus grande descente est mis en œuvre. Cette procédure admet de limiter le domaine de validité des inconnues, ainsi que de donner un poids aux valeurs G que l'on veut ajuster.

Il est nécessaire d'écrire en gibiane deux procédures, l'une FCT qui calcule la valeur de la fonction F pour toutes les abscisses et l'autre DERIV qui calcule les dérivées partielles.

Les données de cette procédure sont :

- Les n listes de valeurs des variables X (TAB1.X.i)
- Liste de n valeurs de G (TAB1.F)
- Le nombre d'inconnues « linéaires »

- Le nombre d'inconnues « nonlinéaires »
- La procédure FCT
- La procédure DERIV
- La liste des valeurs de bornes minimum des inconnues (TAB1.PMIN)
- La liste des valeurs des bornes maximum des inconnues (TAB1.PMAX)
- La liste des n valeurs des poids à affecter à chaque valeur de G (TAB1.POIDS)

En cas de fortes nonlinéarités, cette procédure peut ne pas converger surtout si les bornes min et max des inconnues sont éloignées. En effet le point de départ de l'algorithme est pris au milieu de l'intervalle et il sera peut être trop éloigné de la solution.

2.4 Conclusion sur les identifications

LEVM et AJUSTE nécessitent de fournir les procédures de calcul de la fonction F ainsi que celle du calcul des dérivées partielles puis ces deux méthodes sont censées donner la solution du problème d'identification.
Pour utiliser MOCA dans le cadre d'une fonction F nonlinéaire, il faut construire, en gibiane, un algorithme de recherche de la solution. Celui-ci fera vraisemblablement lui aussi appel au calcul de la fonction F et de ses dérivées partielles. Dans les cas où LEVM et AJUSTE ne converge pas il sera alors possible de tenter une programmation spécifique au type de fonction à identifier.

On trouvera en annexe A le manuel d'utilisation de MOCA,LEVM et AJUSTE. En annexe B on trouvera un exemple d'application de ces trois opérateurs.

3 Minimisation sous contraintes

Il s'agit de trouver le minimum d'une fonction $F(x_1, x_2, x_3, ...)$ tel que les conditions $C_j(x_1, x_2, x_3, ..) \leq \hat{C}_j$ soient respectées et sachant que les n variables $x_1, x_2, x_3 \cdots$ ont des bornes inférieures et supérieures.
Ce problème est très complexe, l'opérateur EXCE, programmé dans Cast3M, résout un problème similaire en particularisant la fonction F et les m fonctions C_j. Pour résoudre le « vrai » problème avec la vraie fonction F on peut tenter de résoudre une succession de problèmes approchés.

3.1 L'opérateur EXCE

Cet opérateur a, dans un premier temps, cherché le minimum d'une fonction Y que l'on écrira, au voisinage du point X^0 , sous la forme :

$$Y = F^0 + \sum_{F>0} \dot{F}_i^0 \quad * \left(x_i - x_i^0 \right) + \sum_{F<0} \dot{F}_i^0 \quad * \left(x_i - x_i^0 \right) \frac{x_i^0}{x_i}$$

Ceci est une approximation convexe de F autour du point X^0, c'est la fonction que Fleury et Braibant [0] [0] ont prise pour leur minimisation en se plaçant dans l'espace dual des contraintes C_j . Les contraintes C_j sont elles aussi développées suivant la même méthode. La méthode à été reprise par Svanberg et l'approximation utilisée est :

$$Y = F^0 - \sum_{F>0} \frac{(U_i^0 - x_i^0)^2 \dot{F}_i^0}{(U_i^0 - x_i^0)} + \sum_{F<0} \frac{(x_i^0 - L_i^0)^2 \dot{F}_i^0}{(x_i^0 - L_i^0)} + \sum_{F>0} \frac{(U_i^0 - x_i^0)^2 \dot{F}_i^0}{(U_i^0 - x_i)} - \sum_{F<0} \frac{(x_i^0 - L_i^0)^2 \dot{F}_i^0}{(x_i - L_i^0)}$$

U_i^0 et L_i^0 définissent des asymptotes que l'on peut faire varier au cours des itérations. Si U_i^0 vaut l'infini quel que soit i et L_i^0 vaut 0 quel que soit i alors les deux approximations sont identiques.
Les contraintes C_j sont développées suivant la même méthode et pour résoudre ce problème on se place aussi dans l'espace dual des contraintes.

Les données de cet opérateur sont :

- Les valeurs de X0
- Les valeurs de F(X0) et des dérivées partielles
- Pour chaque contrainte les valeurs de C_j^0 et des dérivées
- Les bornes des x_i
- Les valeurs des $\hat{C}_j$
- Le choix de la méthode et certains paramètres.

Un algorithme construit autour de cet opérateur ne converge pas toujours et il peut être nécessaire de jouer sur les paramètres proposés de façon un peu aléatoire ! Par exemple, on peut, en s'appuyant sur la solution trouvée par EXCE, chercher un point de départ, sur la

droite passant par la « solution nouvelle » et la « solution ancienne », tel que ce point minimise F tout en respectant les conditions. Ceci améliore souvent la convergence.

3.2 Bilan sur la minimisation sous contraintes

Nous ne disposons pas de méthode sûre, le seul opérateur utilisable est EXCE. Malgré tout il est toujours possible de s'essayer en programmant en gibiane son propre algorithme.

4 Conclusion

Nous avons présenté différents opérateurs de Cast3M qui permettent d'aborder les problèmes d'optimisation. Il existe par ailleurs des opérateurs de calcul de fiabilité qui dans une autre optique réalise aussi une optimisation spécifique.
Il est évident que ces opérateurs ne sont qu'une faible partie de la panoplie envisageable des outils nécessaire au traitement des problèmes d'optimisation. De nouveaux développements se feront au fur et à mesure des besoins.

Bibliographie

[0] K.Levenberg : “ A method for the solution of certain problems in least squares“. Quart. Appl. Math. 1944 : vol 2, p164-168.

[0] D. Marquardt : “ An algorithm for least squares estimation of non linear parameters”. SIAM J. Appl. Math. 1963 : vol 11, p431-441.

[0] C.Fleury V.Braibant : “ Structural optimization : A new dual method using mixed variables”. International Journal for Numerical methods in Engineering. 1986: vol 23, p409-428

[0] K. Svanberg : “The method of moving asymptotes – a new model for structural optimization“. InternationalJournal for Numerical Methods in Engineering. 1987 : vol 24, p359-373

Annexe A

MOCA

Opérateur MOCA — Voir aussi : AJUSTE EXCE, LEVM

```
RESU1 = MOCA LISTPARA LISTMESU LISTFONC LISTDERi (('POIDS' LISTPOI);
```

Objet :

Soit une fonction G connue en n points. On cherche à déterminer les paramètres (a,b,c...,p) d'une fonction F de manière à approcher au mieux la fonction G.
L'opérateur MOCA permet de déterminer ce jeu de paramètres. LISTMESU (listreel) qui contient les n valeurs de G.

Il faut donner les valeurs LISTFONC (listreel de n valeurs) obtenues pour F pour un jeu de parametres LISTPARA (listreel) et qui seront à comparer à LISTMESU. Enfin il faut fournir LISTDERi (listreels de n valeurs)(i=1,p) qui contiennent les dérivées partielles de F par rapport aux paramètres.
Cet opérateur fournit le meilleur jeu de paramètres si F varie linéairement en fonction des paramètres.Il choisit de minimiser un critère égal à :
Somme sur j=1,n(poidj*poidj*(listmesu(j)-listF(j))*(listmesu(j)-listF(j))

Commentaire :

LISTPARA : LISTREEL de P valeurs donnant les paramètres initiaux

LISTMESU : LISTREEL de n valeurs donnant l'objectif pour la fonction G.

LISTFONC : LISTREEL de n valeurs donnant les valeurs de F pour le jeu de paramètres LISTPARA aux n points.

LISTDERi : p LISTREEL donnant chacun la derivée partielle de F (pour chacun des n points) par rapport au ième paramètre.

POIDS : mot introduisant LISTPOI qui contient les n poids à prendre en compte pour le cacul du critère à minimiser. En l'absence de cette donnée tous les poids valent 1.

RESU1 : LISTREEL contenant les valeurs pour les P paramètres.

Remarque : On trouvera un exemple d'utilisation de cet opérateur dans un des jeux de données de Cast3m (identifi.dgibi).

Cet exemple utilise moca,dans un système itératif, pour approcher une fonction nonlinéaire.

LEVM

Opérateur LEVM — Voir aussi : EXCE MOCA AJUSTE

```
LREE5 CHI2  = LEVM  'ABSC' LREE1 'ORDO' LREE2 'SIGM' LREE3
                 'PARA' LREE4 'PROC' PRO1 ;
```

Objet :

L'opérateur LEVM établit la meilleure proposition d'un jeu de paramètres d'une fonction visant à approcher une suite de points (abscisse, ordonnée) spécifiée. Le critère est une moyenne quadratique pondérée des écarts des ordonnées. L'algorithme reprend la méthode dite de Levenberg-Marquardt. L'opérateur n'est pas réinitialisée en cas d'interruption par l'utilisateur.

Commentaire :

LREE5 : type LISTREEL, liste des paramètres proposés

CHI2 : type FLOTTANT, valeur finale du critère

LREE1 : type LISTREEL, liste des abscisses

LREE2 : type LISTREEL, liste des ordonnées

LREE3 : type LISTREEL, liste des poids de chacun des points

LREE4 : type LISTREEL, liste des paramètres d'initialisation (il convient de donner des réels non nuls dans l'ordre de grandeur des valeurs attendues)

PRO1 : procédure gibiane de calcul des ordonnées et des dérivées partielles en chacun des points de LREE1. Il convient de s'assurer d'une précision cohérente pour le calcul des dérivées partielles. Exemple de données :

```
DEBPROC PRO2 LREEX*LISTREEL LREEA*LISTREEL ;

* calcul de la fonction parametree
* compute the parameters dependant function
* LREEX : liste des abscisses / abscissas
* LREEA : liste des parametres / parameters
* LREEY : liste des ordonnees / ordinates

FINPROC LREEY ;

DEBPROC PRO1 LREEX*LISTREEL LREEA*LISTREEL ;

* fonction argument
* input procedure for LEVM
* LREEX : liste des abscisses / abscissas
* LREEA : liste des parametres / parameters
* LREEY : liste des ordonnees / ordinates
```

```
* calcul de LREEY
LREEY = PRO2 LREEX LREEA ;

* derivees partielles / partial derivatives
TLRE = TABLE ;
REPETER BPAR (DIME LREEA) ;
  ai = EXTR LREEA &BPAR ;
  LREEB = COPIE LREEA ;
  REMP LREEB &BPAR (ai * (1. + 1.e-2)) ;
  TLRE . &BPAR = PRO2 LREEX LREEB ;
FIN BPAR ;

* LREDY : derivees partielles / partial derivatives
LREDY = PROG ;
REPETER BX (DIME LREEX) ;
  REPETER BPAR (DIME LREEA) ;
    dyi = ((EXTR TLRE . &BPAR &BX) - (EXTR LREEY &BX) )
              / 1.e-2 / (EXTR LREEA &BPAR) ;
  FIN BPAR ;
FIN BX ;

FINPROC LREEY LREDY ;
```

AJUSTE

```
procédure AJUSTE
----------------

Q P = AJUSTE  TAB1  TAB2;

                        TAB1.'X'     TAB2.'PMIN'
                        TAB1.'F'     TAB2.'PMAX'
                        TAB1.'K'     TAB2.'PRECISION' (facultatif)
                        TAB1.'L'     TAB2.'MXTER'     (facultatif)
                                     TAB2.'POIDS'     (facultatif)
                        TAB1.'IMPRESSION' (facultatif)



 Objet :
--------

  Soit une fonction F(x,y,z...,p1,..,pk) mise sous la forme:

   F(x,p)=q1*f1(x,y,z,.,p1,..,pk) + q2*f2(x,y,z,.,p1,..,pk) +
            q3*f3(x,y,z,.,p1,..,pk) +..
              ..+ ql*fl(x,y,z,.,p1,..,pk) + g(x,y,z,.,p1,..,pk)

   qi (i=1,l) sont les paramètres linéaires.
   pj (j=1,k) sont les paramètres non linéaires.

   La procédure ajuste ces différents paramètres afin que la
   fonction  passe au mieux dans une série de n couples
   ( (x,y,z,...);Fdi(x,y,z,..) ) fournie par l' utilisateur.
   En fait, on cherche à minimiser la fonction :
     G =[ ((poids(i)*(F(x,p)-Fdi(x,y,z..)))**2 ]
         somme sur i=1,n


 Données :
----------

   TAB1.'X':TABLE indicée par des entiers i=1,n qui contient
            le(s) LISTREEL(s) x,y,z,.....
   TAB1.'F':LISTREEL des ordonnés (F(x,y,z...).
   TAB1.'K':ENTIER, nombre de paramètres non linéaires.
   TAB1.'L':ENTIER, nombre de paramètres linéaires.
   TAB1.'IMPRESSION' : ENTIER donne la frequence des impressions
   TAB2.'PMIN' :LISTREEL des valeurs minimum de p.
   TAB2.'PMAX' :LISTREEL des valeurs maximum de p.

   UN ENTIER :TAB1.'IMPRESSION' donne la frequence des impressions,
   la valeur par defaut est toutes les 20 itération.
   Un LISTREEL tab2.'precision'facultatif peut être crée,
   c'est le  critère de précision de convergence pour les k
   paramètres non linéaires.(Par défaut la précision est 1.e-7)

   Un ENTIER tab2.'MXTER' facultatif peut être crée, c'est le
   nombre maximum d'itérations (par défaut 100).

   TAB2.'POIDS':LISTREEL des valeurs de poids à affecter à
                chacun des points de mesure fournis. Par défaut
                chaque points a le meme poids.
 Sortie :
```

```
--------
  Q et P sont des LISTREELs contenant les paramètres qi et pi.


Utilisation :
--------------

  Deux procédures sont à créer par l'utilisateur.

 -Procédure FCT:

  Son but est de calculer la fonction à ajuster connaissant les
  valeurs des abscisses x,y,. et ceci pour un jeu de paramètres
  p donné. En fait, on demande de calculer les fonctions
  f1,f2,...,fl et la fonction g.
  Pour chaque fonction fi, la procédure calcule autant de
  valeurs qu'il y a de valeurs dans x,y,z .. Le résultat doit
  etre mis sous la forme d'un objet TABLE.

  tbfonc=FCT xtab p;

  En argument FCT recevra la table TAB2.'X', ainsi que le
  LISTREEL p qui contient les valeurs courantes de P.

  La table doit se mettre sous la forme suivante:

                tbfonc.'F'.i  =  listréel des valeurs de fi

                tbfonc.'G'    =  listréel des valeurs de g

  Les paramètres linéaires qi n'ont pas à être écrit.

 -Procédure DERI:

  Elle construit une table de listréels contenant les valeurs des
  fonctions f1,f2,...,fl et g derivées par rapport aux paramètres
  non linéaires pj pour chaque valeur de x,y,z.. et de p.

            tbderi=DERI xtab p;

  En argument DERI recevra la table TAB2.'X', ainsi que le
  LISTREEL p.

  La table doit être crée de la façon suivante:

              tbderi.'F'. j . i = listréel des valeurs de dfi/dpj

              tbderi.'G'. j     = listréel des valeurs de dg/dpj


Remarques :
-----------


 - Les abscisses étant dans une listréel , il faut que les
   constantes soient exprimées sous forme de listréel.

   ex: f1(x)=x+1 donne f1(x)=x+(prog n*1);
       n étant la dimension de x.

 - Si la fonction g n'existe pas il est quand même nécessaire
```

```
        d'introduire dans les procédures
        FCT et DERI des valeurs nulles.
        ex: g(x)=prog n*0;
            n dimension de x.

      - Attention, les fonctions sinusoidales appellent les
        les opérateurs de castem 2000 qui utilisent es degrés.

*  EXEMPLE :
*     On cherche a interpoler un nuage de points avec la fonction
*      suivante:
*     the given sets of points ( x , f(x)) must be adjusted with
*     a function as follows :
*
*       f(x) =( A * (x**B ))+ (C * sin(D*x + B)) + (E * x)
*
*    les deux parametres p1 et p2 non linéaires sont B et D.
*    The two non linear parameters p1 and p2 are B and D.
*
*    les autres A,C,E  sont linéaires.On peut donc mettre la
*    fonction sous la forme :
*    A,C,E are linear parameters. It is possible to write the function
*    in the following way :
*     f(x)= A*f1(x) +C*f2(x)+E*f3(x)
*
*     PROCEDURES:  FCT
*
     debproc fct  xtab*table  p*listreel;
            tab1=table; tab2=table; x = xtab . 1;
            n=dime x;
* fonction f1
             tab1 . 1=x**(extr p 1);
* fonction f2
             aa=(((extr p 2)*x) + (prog n*(extr p 1)))*180. / PI;
             tab1 . 2=sin(aa );
* fonction f3
             tab1 . 3 = x;
*remplissage de la table résultat
* filling of the result  table
             tab2.'F'=tab1;
* fonction g
             tab2.'G'=prog n*0;
            finproc tab2;

*     PROCEDURES:  DERI
      debproc deri  xtab*table  p*listreel;

            tab1=table;tab2=table; tabg=table;tabf=table;tab=table;
            x = xtab . 1; n=dime x;
* fonction  df1/dp1
             tab1 . 1=(log x)*(x**(extr p 1));
* fonction df2/dp1
             aa = (((extr p 2)*x) + (prog n*(extr p 1)))*180. / PI;
             tab1 . 2=cos ( aa);
* fonction df3/dp1
             tab1 . 3=prog n*0;

* fonction df1/dp2
             tab2 . 1=prog n*0;
* fonction df2/dp2
             tab2 . 2 = x * ( cos (aa));
```

```
* fonction df3/dp2
            tab2 . 3=prog n*0;

* fonction dg/dp1
            tabg . 1=prog n*0;
* fonction dg/dp2
            tabg . 2=prog n*0;

   tabf . 1=tab1; tabf . 2=tab2; tab.'F'=tabf;tab.'G'=tabg;
   finproc tab;
*     ***** Exemple éxécution
* definition de la fonction pour le calcul des couples x,F(x)
           x=prog  0.01 pas 0.01 1.;
*  posons A = 1 ; B=0.5 ; C=1; D=1; E=3.14159...(pi)
   af1 = x ** 0.5;
   aa = (X + ( prog 100*0.5))* 180.  / PI;cf2 = sin aa;
   ef3 = pi * x;y = af1 + cf2 + ef3;

            k=2; l=3; xtab = table; xtab . 1 = x;
            tab1=table; tab2=table;
            tab1.'X'=xtab;     tab1.'F'=y;
            tab1.'K'=k;        tab1.'L'=l;
            tab2.'PMIN'=prog k*0;
            tab2.'PMAX'=prog k*10;
            tab2.'POIDS' = prog 0.75 pas 0.01 1.74;
            q p=ajuste tab1 tab2;

* recalcul de F(x) pour comparaison avec courbe initiale
* computation of f(x) for comparison with initial curve
  n=dime x;B = extraire p 1;D = extraire p 2;
```

EXCE

```
    Opérateur EXCELLENCE
    --------------------

    TAB1 = EXCE  TAB1 ;
                                    TAB1.'VX0'   .'VF'    .'VXMIN'
                                        .'VXMAX' .'MC'    .'VCMAX'
                                        .'METHODE'        .'DELTA0'
                                        .'MAXITERATION'   .'XSMAX'
                                        .'VDIS'   .'T0'   .'S0'


    Objet :
    -------

 L'opérateur EXCELL cherche le minimum d'une fonction F(Xi) avec
i=1,N et sachant que :

   - il existe des relations   Cj(Xi) < Cjmax       j > 0   j=1,M

   - Il existe des relations sur chaque inconnue  Ximin <  Xi < Ximax

 La donnée des fonctions F et Cj se fait a l'aide des valeurs des
fonctions et de leurs dérivées au point de départ X0.




    Données :
    ---------
 TAB1.'VX0'        : table (sous-type VECTEUR) contenant les valeurs
                     initiales des variables X0i.
                     La table est indicée par les ENTIERs i. (i=1,N)

 TAB1.'VF'         : table (sous-type VECTEUR) contenant :
                   - dans TAB1.'VF'.0 : la valeur de F(X0i)
                   - dans TAB1.'VF'.I : la valeur de la dérivée de F
                     par rapport a Xi en X0 (i=1,N).

 TAB1.'MC'         : table indicée par des ENTIERs j (j=1,M) et
                     contenant autant de tables que de relations Cj.

      TAB1.'MC'.J    est une table représentant la fonction Cj
                    - dans TAB1.'MC'.J.0 : la valeur initiale de
                      Cj(X0) (j=1,M)
                    - dans TAB1.'MC'.J.I : la valeur de la dérivée de
                      Cj par rapport a Xi en X0 (i=1,N).

 TAB1.'VXMIN'      : table indicée par des ENTIERs (i=1,N) et
                     contenant :
                    - dans TAB1.'VXMIN'.I : la valeur de Ximin

 TAB1.'VXMAX'      : table indicée par des ENTIERs (i=1,N) et
                     contenant :
                    - dans TAB1.'VXMAX'.I : la valeur de Ximax
```

TAB1.'VCMAX' : table indicée par des ENTIERs (i=1,M) et contenant :
- dans TAB1.'VCMAX'.I : la valeur de Cjmax

TAB1.'METHODE' : (facultatif) est un MOT précisant la méthode de linéarisation à utiliser.

- 'STA' pour l'emploi de la méthode standard.
- 'MOV' si les fonctions sont très fortement non-linéaires.
- 'LIN' si les fonctions sont peu non-linéaires et qu'il y a des variables a variations non continues.

TAB1.'T0' : (facultatif) change la valeur du réel compris entre 0. et 1. qui gouverne la convexité des fonctions. Plus t0 est grand plus les fonctions sont convexes. Par défaut, pour la méthode standard, t0 est pris égal à 0.3333.

TAB1.'S0' : (facultatif) change la valeur du réel compris entre 0. et 1. qui gouverne la convexité des fonctions. Plus s0 est grand plus les fonctions sont convexes. Par défaut, pour la méthode MOV, s0 est pris égal à 0.7.

TAB1.'MAXITERATION' : (facultatif) change la valeur maximum autorisée pour le nombre d'itérations. (Par défaut 100)

TAB1.'VDIS' : table indicée par des ENTIERs k (k=1,KK) et contenant autant de tables que de variables n'ayant que des valeurs discrètes autorisées. Cette option n'est pas encore disponible.

Remarque :

- Au départ les variables X0i doivent satisfaire aux conditions Ximin < X0i < Ximax

- Le point de départ ne satisfait pas forcément les relations Cj < Cjmax
  Dans ce cas une variables supplémentaire de relaxation est introduite et la solution trouvée par EXCELL ne satisfera peut-être pas non plus les relations. L'influence de cette variable de relaxation peut être modifiée par deux réels TAB1.'DELTA0' et TAB1.'XSMAX'. Par défaut DELTA0=50. et XSMAX=500. (il faut DELTA0 >1. et XSMAX > DELTA0)

Exemple :

La fonction que l'on désire minimiser n'est pas celle qui est minimisée par l'opérateur EXCE. La démarche à suivre est de résoudre une succession de problème. Partant d'un état connu des variables on demande à EXCE de calculer le minimum d'un problème approché, la fonction F à minimiser est remplacée par la fonction linéarisée décrite ci-dessus ainsi que les fonctions C. Puis on repart de la solution trouvée par EXCE. L'algorithme se présente ainsi :

- création des objets TABLES

```
- initialisations de TAB1.'VX0' ( valeurs de Xi0)
- initialisation de TAB1.'VF'.
- initialisation de TAB1.'MC'.  J .
- initialisation de TAB1.'VXMIN' et TAB1.'VXMAX'
- initialisation de TAB1.'VCMAX'
- répéter 5 fois la suite :
   - calcul de DF et mise dans TAB1.'VF' , ainsi que F(XI0)
   - calcul des Cj  et mise dans TAB1.'MC'.j.
     ainsi que Cj(Xi0)
   - appel à EXCE avec la table TAB1 en entrée.
   - imprimer TAB1.'VX0'
```

- fin de boucle

Annexe B

```
*
* soit le polynome du 3eme degre y=a*x*x*x + b*x*x + c*x + d
*
* on suppose connue la valeur d'une fonction G pour les
* abscisses x= 1, 2, 3, 4 et 5 .On recherche les coefficients a b c d
* de la fonction Y (polynome du troisieme degré) qui ajuste au mieux la
* fonction G sur les points connus.


* poly est une procédure calculant la fonction Y pour des valeurs de
* a b c d données en argument et pour les valeurs expérimentales de x

debp poly a*flottant b*flottant c*flottant  d*flottant;
x=1 ;
*list x; list a; list b; list c ; list d;
f1=(a*x*x*x) + (x*x*b) + ( c * x ) + d  ;
x=2;
f2=(a*x*x*x) + (x*x*b) + ( c * x ) + d  ;
x=3;
f3=(a*x*x*x) + (x*x*b) + ( c * x ) + d  ;
x=4;
f4=(a*x*x*x) + (x*x*b) + ( c * x ) + d  ;
x=5;
f5=(a*x*x*x) + (x*x*b) + ( c * x ) + d  ;
ll=prog f1 f2 f3 f4 f5;
finpro ll;

* Nous choisissons la base connue de G correspondant à un polynome du
* 3eme degré avec a=2 b=3 c=4 d=5 . Nous devons donc retrouver ces
* valeurs pour a b c d.
 fG = poly 2. 3. 4.  5.;



* on cherche par l'opérateur MOCA les valeurs de a b c d
* on suppose un point de départ a=1 b=-1. c=10; d=8.

a=1.; b=-1.;c=10.; d = 8.;
fdep = poly a b c d ;
* calcul de la fonction y pour ces paramètres et pour les abscisses
depa = prog a b c d ;
* calcul des "dérivées partielles" par rapport à : a , b , c , d par
* différence finie
debproc deriv a*flottant b*flottant c*flottant d*flottant fdep*listreel;
deri_a= ((poly (a*1.01) b c  d) - fdep)/( a*0.01);
deri_b= ((poly a (b*1.01) c  d) - fdep)/ (b*0.01);
deri_c =((poly a b (c*1.01)  d) - fdep)/(C*0.01);
deri_d =((poly a b c  (d*1.01)) - fdep)/(D*0.01);
finproc deri_a deri_b deri_c deri_d ;
deri_a deri_b deri_c deri_d = deriv a b c d fdep;
ss = moca  depa fG fdep deri_a deri_b deri_c deri_d ;

list ss;
sol = prog 2. 3. 4. 5. ;
er = abs ( ss - sol);
```

```
ermax1= maxi er;

* remarque : l'opérateur MOCA cherche à minimiser l'écart en moyenne
* quadratique entre la base expérimentale et la fonction F(a,b,c,d),
* supposée linéaire en a b c d. Le polynome Y étant bien linéaire
* en a b c d, il trouve la "bonne" solution.


*
* envisageons maintenant que la fonction Y soit y= a + bx + cx**d
*
* on peut utiliser la procedure ajuste

* parametres lineaires a b c
* parametre non lineaire d
* on peut ecrire la fonction sous la forme:

* y = a * f1 + b* f2 + c* f3;
* avec f1= 1    f2 = X     f3 = x**d

debp fct xtab*table p*listreel;

    tab1=table; tab2=table; x = xtab . 1;
            n=dime x;
* fonction f1
    tab1.1 = prog n*1.;
    tab1.2 = x;
    tab1.3 = x**( extr p 1);
    tab2.'F'= tab1;
    tab2.'G' = prog n*0;
finproc tab2;
debproc deri  xtab*table  p*listreel;
       tab1=table;tab2=table; tabg=table;tabf=table;tab=table;
       x = xtab . 1; n=dime x;

*df1/dp1
   tab1 . 1=prog n*0.;
*df2/dp1
   tab1. 2 = prog n*0.;
*df3/dp1
   tab1. 3= ( log x) * (x** ( extr p 1))
* dg/dp1
   tabg.1 = prog n*0.;tabf . 1 = tab1;
   tab. 'F' = tabf;   tab.'G' = tabg;
finproc tab;
* fabriquons un cas "experimental" en
* posant a =1 b=-6 c=4 d=3 e et en prenant pour base des abscisses x
sol2= prog 1. -6.  4.  3.;
x = prog   0.1 0.3 0.5 1 2.5 3.6 6. 7.8 9 11; nexp=10;
* on va calculer nos "resultats experimentaux" : fobj
ct= prog nexp*1.;
bx = -6. * X;
cxpd= 4*( X ** 3);
fobj= ct + bx + cxpd;

* preparation des données pour appel ajuste
k=1;L=3; xtab=table; xtab. 1= x;
tab1=table; tab2= table;
tab1.'X' = xtab;  tab1. 'F' = fobj;
tab1. 'K'= k; tab1.'L'= L;
tab2.'PMIN'=prog 0.  -12   -20  -2   ;
```

```
tab2.'PMAX' = prog 10.  26.  12.  4.     ;
tab2.'POIDS'= prog nexp*1. ;

temps;
 p  q = ajuste tab1 tab2;
temps;
mess ' sortie de la procedur ajuste valeur trouvée :';
mess ' a = ' ( extr p 1)  ' valeur attendue   1.';
mess ' b = ' ( extr p 2)  ' valeur attendue  -6.';
mess ' c = ' ( extr p 3)  ' valeur attendue   4.';
mess ' d = ' ( extr q 1)  ' valeur attendue   3.';
ltrou= p et q;
era= abs ( ltrou - sol2);
ermax2= maxi era;
*
* on peut aussi tenter de resoudre le probleme à l'aide de MOCA
* Comme la fonction n'est pas linéaire en fonction du parametre d
* la solution fournie ne sera pas la bonne. on utilise le résultat
* trouvé pour nous donner une direction de descente le long de laquelle
* on cherche un minimun de la "distance" entre Fcalculée et Fobjectif
* une approche par itération est necessaire
*

para = prog  2. 2. 2. 2.;


* procedure pour calculer la fonction
debproc fcal para*listreel x*listreel;
n = dime x;
fa=  prog n*( extr para 1) ;
fb= ( extr para 2)*x ;
fc= ( extr para 3) * ( x ** ( extr para 4));
ff= fa + fb + fc;
finproc ff;
* procedur pour calculer les derivees partielles
debproc fderiv para*listreel x*listreel ;
n=dime x;
fdera= prog n*1.;
fderb= x * 1;
fderc= x**(extr para 4);
fderd = (extr para 3)*( log x) * (x** ( extr para 4));
finproc fdera fderb fderc fderd;
* procedure pour calculer le critère de distance
debproc criter fcalc*listreel fobjec*listreel ;
cri= 0.;
repe aa ( dime fcalc);
cri = cri + (  ((extr fcalc &aa) - (extr fobjec &aa) )**2);
fin aa;
finproc cri;
* schema iteratif
temps;
repeter iter 500;


fdep = fcal para x; crii = criter fdep fobj;
mess ' debut iteration ' &iter ' critère ' crii;

*
* on teste la convergence ( coutnouv - coutanc ) / coutanc < 1.e-4
*
si ( &iter. ega 1) ;
```

```
  crianc = crii;
sinon;
  cri_conv= ( crianc - crii ) / crianc;
  mess ' critère de convergence ' cri_conv;
  crianc= crii;
  si ( cri_conv < 1.e-4) ;
    mess  ' convergence à l itération ' &iter ' cout ' crii;
    quitter iter;
  finsi;
finsi;

fdera fderb fderc fderd = fderiv para x ;
*list fobj;list fdep; list fdera ; list fderb; list fderc; list fderd;
nouvpara= moca para fobj fdep fdera fderb fderc fderd;
*list nouvpara;
*
* recherche le long de la direction donnée par nouvpara - para
*
desc= (nouvpara - para ) * 0.333333333;

imu=1;
repe ide 30;
nouvpara= para + (desc * imu);
fnou= fcal nouvpara x;
cria= criter fnou fobj;
*mess ' ide imu  crianc    crinouv' &ide imu crii cria;
si ( cria > crii );
 si ( imu ega 1) ;
   si (&ide < 6) ; desc = desc / 10.;iterer ide;
   sinon;
     mess ' pas de longueur trouvée le long de la descente';
     quitter iter;
   finsi;
 finsi;
* recherche d'un min par approximation parabolique
    aa = cri2 + cria - (2.*crii) / 2.;
    bb= cria - cri2 / 2.;
    xx = bb / -2. / aa ;
    iies=   xx - 1. ;
    para= nouvpara + ( iies * desc);
    quitter ide;
sinon;
  si ((imu ega &ide ) et  (imu ega 9)) ;
       para= nouvpara;quitter ide;
  finsi;
    imu=imu+1 ;
    cri2=crii;
    crii=cria;
finsi;
fin ide;
fin iter;
temps;
mess ' résutats par utilisation de moca itératif';
mess ' a = ' ( extr para 1)  ' valeur attendue   1.';
mess ' b = ' ( extr para 2)  ' valeur attendue  -6.';
mess ' c = ' ( extr para 3)  ' valeur attendue   4.';
mess ' d = ' ( extr para 4)  ' valeur attendue   3.';
erb= abs ( para - sol2);
ermax3= maxi erb;
*
*
```

```
* utilisation de l'opérateur levmar
*
* il faut définir la procédure feval qui evalue la fonction
* à minimiser et qui calcule les derivées partielles
*
debproc feval x*listreel   para*listreel;
dy=prog;
n=dime x;m = dime para;
a1= extr para 1 ; b1= extr para 2;
c1= extr para 3; d1= extr para 4;
aa1 = prog n*a1;

y= aa1 + ( b1 * x ) + ( c1 * ( x ** d1)) ;

fdera= prog n*1.;
fderb= x * 1;
fderc= x**(extr para 4);
fderd = (extr para 3)*( log x) * (x** ( extr para 4));
l_dy=prog;ia=0;
repe baa n;
l_dy= l_dy et (prog ( extr fdera &baa));
l_dy= l_dy et (prog ( extr fderb &baa));
l_dy= l_dy et (prog ( extr fderc &baa));
l_dy= l_dy et (prog ( extr fderd &baa));
fin baa;
finp y l_dy;

aa = prog 2. 2. 2. 2.;sis = prog nexp*1. ;
temps;
a0 chi2 = levm ABSC x ORDO fobj 'PARA' aa SIGM sis PROC feval  ;
temps;
mess ' résultats par utilisation de levm ' ;
mess ' a = ' ( extr a0 1)  ' valeur attendue   1.';
mess ' b = ' ( extr a0 2)  ' valeur attendue  -6.';
mess ' c = ' ( extr a0 3)  ' valeur attendue   4.';
mess ' d = ' ( extr a0 4)  ' valeur attendue   3.';
erc = abs ( a0 - sol2);
ermax4= maxi erc;

message ' erreur pour moca ' ermax1;
message ' erreur pour ajuste ' ermax2;
message ' erreur pour moca '  ermax3;
message ' erreur pour levm ' ermax4;
si ( (ermax1 + ermax2 + ermax3 + ermax4 ) > 1.e-4);
   erreur  5;
finsi;

fin;
```

Résumé

L’optimisation, qui est la recherche du » meilleur » possible, se traduit mathématiquement par la recherche du minimum d’une fonction de plusieurs variables. Une étude des outils, dont dispose le logiciel Cast3M, qui permettent à l’utilisateur de traiter un problème d’optimisation est présentée.
Le problème d’identification des paramètres d’une fonction est particulièrement étudié et la minimisation sous contraintes est elle aussi évoquée.
Enfin, quelques exemples d’applications illustre les solutions proposées par Cast3M.