Les modules
M2QN1 et MQHESS
(Octobre 1983)

Claude LEMARÉCHAL[†] et Éliane PANIER

1 Objet

C'est un optimiseur de classe 2, c'est-à-dire qu'il résout un problème d'optimisation avec contraintes de bornes :

$$\begin{cases} \min f(x) \\ b^{\inf} \leq x \leq b^{\sup}, \end{cases}$$

où $f : \mathbb{R}^n \to \mathbb{R} : x \mapsto f(x)$ est le critère à minimiser, supposé différentiable, et les bornes sont données au moyen de deux vecteurs $b^{\inf} \in \mathbb{R}^n$ et $b^{\sup} \in \mathbb{R}^n$. On demande donc à ce que la solution x vérifie $b_i^{\inf} \leq x_i \leq b_i^{\sup}$, pour $i = 1, \ldots, n$. On dit qu'un point x est *admissible* s'il vérifie les bornes.

M2QN1 utilise une méthode de quasi-Newton, dite de BFGS (en fait, c'est le module M1QN1 à peine modifié – rappelons ici que M1QN1 est lui-même adapté de VA13A de la librairie Harwell), et une stratégie de contraintes actives dite à la Rosen-Goldfarb. De plus, une facilité est accordée pour faire une analyse post-optimale.

Le module d'optimisation M2QN1 se compose du programme chapeau `m2qn1` et de l'optimiseur proprement dit `m2qn1a`, lequel fait appel à la sous-routine `mlis0` (de Modulopt), aux esclaves `emc11a`, `emc11b`, `emc11e` (communs avec M1QN1) ainsi que `emc11z`, `eajc1`, `eretc1`, `emani1`, `ecomp1`, `emlag1`. L'analyse post-optimale est faite par `mqhess`, qui appelle `e1qhes`, `e2qhes`, ainsi que `eajc1`.

2 Description sommaire de la méthode

Soit x l'itéré courant, satisfaisant les contraintes de bornes, I l'ensemble des indices des bornes jugées actives en x ($i \in I$ si $x_i \simeq b_i^{\inf}$ ou $x_i \simeq b_i^{\sup}$, voir la section 5.3 pour la définition précise de I), et M l'approximation courante de la matrice hessienne. Une itération de l'algorithme est formée des étapes suivantes.

1. On calcule la direction de déplacement d en x en résolvant le problème

$$\begin{cases} \min \nabla f(x)^\top d + \dfrac{1}{2} d^\top M d \\ d_i = 0, \quad \text{pour } i \in I, \end{cases}$$

   où $\nabla f(x) = (\frac{\partial f}{\partial x_i}(x))_{i=1}^n$ et le *gradient* de f en x (relatif au produit scalaire euclidien).

2. On effectue (dans `mlis0`) une recherche linéaire sur le segment réalisable porté par la direction d ainsi calculée.

3. On modifie M par la formule de BFGS.

4. Avant de boucler en 1, on révise éventuellement I de la façon suivante.

[†]INRIA, 655 avenue de l'Europe, 38330 Montbonnot, France. Tel: 33/04 76 61 52 02. E-mail: `Claude.Lemarechal@inria.fr`.

(a) Si la recherche linéaire a buté sur une nouvelle borne, on insère l'indice de cette borne dans I.

(b) Si aucune borne nouvelle n'a été rencontrée, soit on garde le même I, soit on lui retranche un indice de borne si cela en vaut la peine.

Voir la section 5 pour plus de détails.

[1] Si $r \in \mathbb{R}$, on note $r^+ = \max(0, r)$ et $r^- = \max(0, -r)$. On appelle *gradient projeté* en un point x admissible, le vecteur de $\mathbb{R}^n$ dont la composante i vaut

$$\begin{cases} (\nabla f(x))_i & \text{si } b_i^{\inf} < x_i < b_i^{\sup} \\ (\nabla f(x))_i^- & \text{si } x_i = b_i^{\inf} \\ (\nabla f(x))_i^+ & \text{si } x_i = b_i^{\sup}. \end{cases}$$

3 Usage

3.1 Description des paramètres

L'instruction d'appel est la suivante

```
subroutine m2qn1 (simul, n, x, f, g, dxmin, df1, epsabs, imp, io,
                  mode, iter, nsim, binf, bsup, iz, rz, izs, rzs,
                  dzs)
```

Dans la description qui suit, (E) désigne un paramètre devant être initialisé avant l'appel à `m2qn1`, (S) un paramètre n'ayant de signification qu'à la sortie et (ES) un paramètre devant être initialisé, mais qui est aussi modifié par `m2qn1` (ce ne peut donc être une constante).

`simul` : point d'entrée dans la sous-routine de simulation (voir Normes Modulopt I[2]). Il est supposé que cette sous-routine est de la forme suivante : [3]

`subroutine simul (indic, n, x, f, g, izs, rzs, dzs)`.

Le nom de cette sous-routine doit être déclaré `external` dans la sous-routine appelant `m2qn1`. Les arguments `n`, `x`, `f`, `g`, `izs`, `rzs` et `dzs` ont la même signification que ci-dessous. Une particularité de M2QN1 est de n'appeler le simulateur `simul` qu'avec `indic` = 1 ou 4 (jamais avec `indic` = 2 ou 3). Lorsque `indic`=1, le simulateur fera ce que bon lui semble, mais ne modifiera pas les arguments de `simul`. Lorsque `indic`=4, le simulateur calculera à la fois f et g au point courant x.

Au retour de la simulation, la valeur `indic` = 0 est interprétée par l'optimiseur comme une demande d'arrêt de l'optimisation, ce qui sera suivi d'effet. C'est donc une possibilité offerte au simulateur d'arrêter l'optimisation parce qu'un événement indésirable s'est produit (point ou valeur de fonctions en dehors de bornes raisonnables, par exemple).

Au retour de la simulation, la valeur `indic` < 0 est interprétée par l'optimiseur comme le fait que l'évaluation du critère a été demandée en un point en dehors d'une zone définie implicitement par le simulateur (par exemple parce que celle-ci n'est pas définissable explicitement par des bornes). L'optimiseur essaiera de

[1] Ce paragraphe a été ajouté. D'accord?

[2] Mieux vaut décrire ce qu'il faut faire ici plutôt que de faire référence aux Normes qui n'existent pas en Latex.

[3] Vérifier les arguments.

sauver cette situation en prenant un pas plus petit. En aucun cas cette technique ne peut être utilisée pour introduire des contraintes générales.

`n` (E) : scalaire `integer`. Nombre de variables dont dépend f, dimension n de x.

`x` (ES) : vecteur `real` de dimension `n`.

En entrée : c'est le point initial. Si x n'y est pas, M2QN1 s'arrête dès le départ.[4]

En sortie : le point final calculé par M2QN1.

`f` (ES) : scalaire `real`.

En entrée : valeur de f au point x d'entrée.

En sortie : valeur de f au point x de sortie.

`g` (ES) : vecteur `real` de dimension `n`.

En entrée : valeur du gradient g de f au point x d'entrée.

En sortie : valeur du gradient g de f au point x de sortie.

Il faut donc appeler le simulateur `simul` avant `m2qn1` pour remplir f et g.

`dxmin` (E) : vecteur `real` de dimension `n`. Son rôle est quadruple.

1. Précision requise sur chaque coordonnée de x. M2QN1 s'arrête lorsqu'il trouve un x tel que, pour obtenir y avec $f(y) < f(x)$, il doit prendre

$$|y_i - x_i| \leq \texttt{dxmin}_i \quad i = 1, \ldots, n.$$

   Ceci n'entraîne pas qu'il y a alors un minimum local dans la boite centrée en x (sortie) et de côté `dxmin` car M2QN1 n'a pas examiné tous les y possible !

2. Précision sur les bornes. Une borne i est considérée comme active dès que x_i lui est égal à $\texttt{dxmin}_i$ près.

3. Changement de variables implicite $\tilde{x}_i = x_i/\texttt{dxmin}_i$, ce qui a une inférence sur le test d'arrêt (voir la section 5.4).

4. Si `mode` (entrée) = 1, `dxmin` sert à calculer un préconditionnement en initialisant M à la matrice diagonale

$$M_{ii} = K/\ \texttt{dxmin}_i^2, \quad i = 1, \ldots, n,$$

   où $K > 0$ est calculé de façon à ce que le modèle quadratique décroisse de `df1` (autre paramètre de `m2qn1`) à la première itération.

   Il convient donc, si `mode` = 1, de déterminer `dxmin` avec quelque soin.

   Exemple. Supposons deux variables : `x(1)` taux d'intérêt (de l'ordre de quelques %) et `x(2)` investissement (de l'ordre de quelques millions). Lorsque `mode` est initialisé à 1, M2QN1 se comportera mieux si `dxmin(2)` est quelque cent millions de fois plus grand que `dxmin(1)`.

`df1` (ES) : scalaire `real`.

En entrée. Cette valeur n'est utilisée que si `mode` = 1 en entrée. Dans ce cas, ce doit être un nombre strictement positif estimant grossièrement la décroissance du critère prévue à la première itération. Celle-ci est normalement de l'ordre de la différence ente la valeur de f en x (entrée) et le minimum.

Si `df1` est beaucoup trop petit, `m2qn1` peut se bloquer dès la première itération. Une valeur trop grande de `df1` n'est en principe pas grave.

Si `mode` > 1 en entrée, `df1` n'a pas à être initialisé.

[4] Pourquoi ne pas le projeter sur l'ensemble admissible?

En sortie : `df1` n'a pas une valeur très utile; c'est la diminution de f obtenue à la dernière itération.

`epsabs` (ES) : scalaire `real`.

En entrée : détermine le test d'arrêt sur les conditions de Kuhn et Tucker. M2QN1 considère que la convergence a été obtenue lorsque les conditions d'optimalité sont à peu près satisfaites. Plus précisément, M2QN1 s'arrête lorsque la valeur moyenne des composantes du gradient projeté devient inférieure à `epsabs`. `Epsabs` doit donc être positif.

Pour fixer les idées, sur une machine travaillant avec 8 chiffres significatifs, `epsabs` $= 10^{-4}$ fois la valeur moyenne des dérivées initiales correspond déjà à une bonne précision.

En sortie : `epsabs` contient la valeur moyenne des composantes du gradient projeté.

Voir la section 5.3 pour plus de détails.

`imp` (E) : scalaire `integer`. Contrôle les messages d'impression.

< 0: Le simulateur `simul` est appelé toutes les `-imp` itérations avec `indic` $= 1$. [5]

$= 0$: Rien n'est imprimé.

$= 1$: Impressions initiales et finales. En particulier, imprime à la fin `epsabs` (sortie) décrit à la section 5.4.

$= 2$: Une impression chaque fois que l'on ajoute ou retranche une borne active (nombre d'itérations, nombre de simulations, valeur courante de f, numéro de la borne).

$= 3$: Une impression par itération (nombre d'itérations, nombre de simulations, valeur courante de f).

$= 4$: Informations supplémentaires des recherches linéaires (très utile pour détecter les erreurs dans le gradient, voir la notice de `mlis0`, section 6).

`io` (E) : scalaire `integer`. C'est le numéro du canal de sortie, i.e., les impressions commandées par `imp` sont faites par

```
write (io,...)  ...
```

`mode` (ES) : scalaire `integer`.

En entrée : précise l'initialisation.

$= 1$: "démarrage à froid" : M2QN1 se charge en particulier d'initialiser M et de déterminer les bornes actives et le pas initial.

$= 2$: L'approximation initiale du hessien est donnée dans `rz` sous forme symétrique compressée, c'est-à-dire que ses coordonnées se suivent dans l'ordre $(1,1)$, $(2,1)$, ..., $(n,1)$, $(2,2)$, $(3,2)$, ..., $(n,2)$, ..., $(n-1,n-1)$, $(n,n-1)$, (n,n).

$= 3$: Le hessien initial est également donné dans `rz`, mais factorisé par `emc11b`.

$= 4$: Le présent appel s'enchaîne à un appel précédent. La fonction f à minimiser peut avoir légèrement changé mais la matrice de quasi-Newton est néanmoins réinitialisée sur celle obtenue lors du précédent appel. Entre les deux appels de M2QN1, il faut recalculer `epsabs`, `mode`, `iter`, `nsim`, il *faut* laisser tels quels `n`, `x`, `binf`, `bsup`, `iz` et `rz` ; si la fonction f a changé, il faut recalculer f et g.

En sortie : précise les circonstances de l'arrêt.

< 0: Le simulateur `simul` a répondu `indic` < 0 abusivement et l'optimisation ne peut se poursuivre (alors `m2qn1` sort avec `mode` $=$ `indic`). Ordinairement,

[5] Imprime t'il q.ch.?

cela signifie qu'il faut expliciter une contrainte qui était implicitement prise en compte par `indic` (voir les Normes Chap. II section 1.2[6]).

$= 0$: Le simulateur `simul` a répondu `indic` $= 0$, ce qui implique l'arrêt de M2QN1. Ceci constitue donc un moyen d'arrêter M2QN1 à partir du simulateur.

$= 1$: Fin normale, le test d'arrêt est satisfait (voir la section 5.4).

$= 2$: L'un des paramètres d'entrée a été mal initialisé : `n`, `dxmin`, `df1`, `epsabs`, `iter` et `nsim` n'est pas positif, ou bien `mode` n'est pas entre 1 et 4, ou bien x n'est pas dans les bornes.

$= 3$: La matrice de quasi-Newton est devenue non définie positive. Cette sortie ne devrait pas se produire avant d'avoir presqu'atteint la convergence car elle contredit la théorie ; si besoin est, relancer M2QN1 avec `mode` $= 1$ (de façon à réinitialiser le hessien sur l'identité).

$= 4$: Nombre maximum d'itérations atteint.

$= 5$: Nombre maximum de simulations atteint.

$= 6$: M2QN1 a atteint la précision `dxmin` (voir ci-dessus le rôle de `dxmin`, clause 1).

$= 7$: Échec dans la factorisation de la matrice hessienne (cette sortie est en fait incompréhensible et ne devrait pas se produire ; sauf erreur grossière dans la liste d'appel, elle signifie en tous cas que le problème est dégénéré).

`iter` (ES) : scalaire `integer`.

En entrée : nombre maximum autorisé d'itérations.

En sortie : nombre d'itérations réellement faites.

Une itération est l'enchaînement des étapes 1 à 4 de la section 2.

`nsim` (ES) : scalaire `integer`.

En entrée : nombre maximum autorisé d'appels effectifs au simulateur `simul` (c'est-à-dire avec `indic` $= 4$).

En sortie : nombre de tels appels réellement faits.

`binf` (E) : vecteur `real` de dimension `n`. Les bornes inférieures sur x.

`bsup` (E) : vecteur `real` de dimension `n`. Les bornes supérieures sur x.

`iz`, `rz` (S) : vecteurs `integer` et `real`, respectivement. Zones de travail.

$$\begin{aligned} \text{Dimension de } \texttt{iz} &= 2n+1. \\ \text{Dimension de } \texttt{rz} &= \frac{1}{2}n(n+9). \end{aligned}$$

Lorsque `mode` (entrée) $= 2$ ou 3, les $n(n+1)/2$ coordonnées de `rz` doivent être initialisées

- soit par l'utilisateur (`mode` $= 2$),
- soir par un appel préalable à `emc11b` (`mode` $=3$).

Lorsque `mode` (entrée) $= 4$, `iz` et `rz` doivent être tout entiers tels que restitués par M2QN1 au cours de l'appel précédent.

`izs`, `rzs`, `dzs` : scalaires ou vecteurs de type `integer`, `real` et `double precision`, respectivement. Mémoires réservées au simulateur `simul`. M2QN1 n'y touche pas et se contente de transmettre ces variables au simulateur `simul` qui en fait ce qu'il veut. Ces variables peuvent donc être utilisées pour transmettre de l'information au simulateur `simul` autrement que par des blocs `common`.

[6] Assez obscure; à préciser en l'absence de Normes en Latex.

3.2 Séquence d'appel

[7] Nécessairement, les instructions précédant l'appel à `m2qn1` incluent les éléments suivants :

1. la déclaration en `external` du nom du simulateur (appelé `simul` par M2QN1) dans le programme ou la sous-routine appelant `m2qn1`;
2. la détermination du point de départ x (entrée); **rappelons que ce x doit vérifier les bornes spécifiées par `binf` et `bsup`**;
3. **l'appel au simulateur pour calculer f et g en x (entrée)**;
4. le calcul de `df1` et l'initialisation de `dxmin`, `epsabs`, `imp`, `io`, `mode`, `iter`, `nsim`, `binf`, `bsup` et éventuellement celle de `iz` et `rz`;
5. l'appel à `m2qn1`.

4 Analyse post-optimale

Après exécution de M2QN1, on peut obtenir une estimation de la matrice des dérivées secondes de f en x (sortie). Appelons quasi-hessien cette estimation, puisqu'elle a été calculée par des formules de quasi-Newton. Pour cela, il faut appeler `mqhess` par l'instruction

```
call mqhess (n, imp, io, iz, rz)
```

où la signification des paramètres est claire. Les paramètres `n`, `iz` et `rz` ne doivent pas avoir été modifiés depuis le retour de M2QN1. Si `imp` > 0, le quasi-hessien est imprimé ; sinon rien n'est imprimé.

Après exécution de `mqhess`, le quasi-hessien se trouve dans les $n(n+1)/2$ premières positions de `rz` sous forme symétrique compressée (i.e., comme décrit à la section 3.1, paramètre `mode`).

N'accorder qu'une confiance limitée à ce quasi-hessien. Il a été calculé par une succession de mises à jour de quasi-Newton (au cours de l'exécution de M2QN1) et aucun théorème n'assure que dans ces conditions, on obtient même une approximation du vrai hessien. En tout cas, on peut remarquer les choses suivantes :

- le quasi-hessien est probablement assez faux si M2QN1 a fait peu d'itérations (l'information du second ordre n'a alors pas eu le temps de s'accumuler) ;
- la partie de `rz` correspondant aux variables bloquées sur une borne a toutes les chances d'être mauvaise (l'information correspondante n'a pas pu être calculée puisque ces variables n'ont pas bougé en fin d'itérations) ;
- même dans des conditions favorables, la qualité du quasi-hessien peut être tout à fait médiocre.

Pour illustrer ce dernier point, considérons la fonction[8]

$$f(x) = \sum_{i=1}^{n} \left(i x_i^2 + \frac{1}{i} x_i \right)^2 + \sum_{i=1}^{n} \left(2x_i + \frac{1}{i^2} \right)^4 ,$$

dont la solution est $x_i = \frac{1}{2} i^2$. Lorsque $n = 7$, M2QN1 donne la sortie suivante

[7] Cette section a été ajoutée.
[8] Vérifier cette fonction, sa solution, son hessien en la solution. Ça ne me parait pas très clair.

```
entree dans m2qn1. dimension du probleme  7

mode= 1  epsabs=  0.10e-06  niter = 50 nsim = 1500  imp = 3

dimensions minimales iz(15) rz(56)

m2qn1 bornes initiales  (0 inactive, -1 binf active, +1 bsup active)
m2qn1                     1  0    2  0    3  0    4  0    5  0
m2qn1                     6  0    7  0

m2qn1    1 iters    1 simuls   f= 0.1004077e+01
m2qn1    2 iters    4 simuls   f= 0.2549078e+00
m2qn1    3 iters    7 simuls   f= 0.2766499e+00
m2qn1    4 iters    8 simuls   f= 0.2813445e+00
m2qn1    5 iters   11 simuls   f= 0.2924372e+00
m2qn1    6 iters   13 simuls   f= 0.2975580e+00
m2qn1    7 iters   15 simuls   f= 0.2976179e+00
m2qn1    8 iters   18 simuls   f= 0.2982445e+00
m2qn1    9 iters   20 simuls   f= 0.2982486e+00
m2qn1   10 iters   22 simuls   f= 0.2982539e+00
m2qn1   11 iters   25 simuls   f= 0.2982971e+00
m2qn1   12 iters   26 simuls   f= 0.2982984e+00
m2qn1   13 iters   29 simuls   f= 0.2982994e+00
m2qn1   14 iters   31 simuls   f= 0.2983015e+00
m2qn1   15 iters   32 simuls   f= 0.2983016e+00
m2qn1   16 iters   36 simuls   f= 0.2983016e+00
m2qn1   17 iters   38 simuls   f= 0.2983018e+00
m2qn1   18 iters   40 simuls   f= 0.2983018e+00
m2qn1   19 iters   40 simuls   f= 0.2983018e+00

mlis0   fin sur dxmin       pas           fonctiions       derivees
mlis0                 0.00000000E+00   -0.29830178E+00   -0.647E-09
mlis0                 0.58819054E-01   -0.29830178E+00   -0.644E-09

sortie de m2qn1. norme du gradient projete 0.104e-03
mode = 6  niter = 19  nsim = 91
```

et la solution est obtenue avec 3 chiffres significatifs corrects. Pourtant `mqhess` donne

```
mqhess   hessienne au point final
mqhess   1   0.2004e+01
mqhess   2   0.1192e-01    0.4129e+01
mqhess   3  -0.4056e-03   -0.3234e+00    0.8132e+01
mqhess   4   0.3348e-01    0.1081e+01   -0.5767e+01    0.2415e+02
mqhess   5  -0.3765e-01   -0.1742e+01    0.9941e+01   -0.2751e+02    0.5706e+02
mqhess   6   0.4310e-01    0.1584e+01   -0.8702e+01    0.2425e+02   -0.4137e+02
mqhess       0.4842e+02
mqhess   7  -0.1109e-01   -0.5471e+00    0.3128e+01   -0.8651e+01    0.1480e+02
mqhess      -0.1301e+02    0.1865e+02
```

alors que la véritable hessienne est $\mathrm{diag}(2, 4, 6, 8, 10, 12, 14)$.

5 Détails algorithmiques

5.1 Relâchements

On sait que relâcher les bornes trop souvent conduit à des zigzags pouvant entraîner la convergence de l'algorithme vers un point qui n'est pas solution (voir [4]). Le fait d'interdire un relâchement si une borne vient d'être ajoutée (voir la section 2, étape 4(a)) constitue un procédé anti-zigzag (voir [1] ; signalons toutefois que l'on ne connaît pas de théorème de convergence).

A l'étape 4(b), la connaissance de g, M et I permet de calculer de combien le modèle

quadratique décroîtrait si on relâchait un indice donné. On relâche alors l'indice donnant la décroissance maximale, pourvu que[9]

- cette décroissance soit suffisante (2 fois ce que l'on obtiendrait sans relâchement),
- le multiplicateur associé (dans le problème quadratique de l'étape 1) soit du mauvais signe,
- la composante du gradient soit également du mauvais signe.

De plus, on relâche une borne (à moins que l'on ne déclare avoir convergé) si le gradient projeté est petit (cet événement est appelé "convergence dans un sous espace" ; le mot "petit" est compris par référence à `epsabs`).

Cette stratégie de contraintes actives est assez similaire dans son esprit à [2]).

[10]

5.2 Calculs algébriques

La décomposition $LDL^\top$ permet d'effectuer tous les calculs algébriques nécessaires.

Lorsque `mode` (entrée) = 2, vérification est faite que le hessien est défini positif. Si ce n'est pas le cas, on repart sur `mode` = 1.[11]

La matrice M est formée de deux blocs : l'un pour les variables libres (stockage sous forme décomposée par `emc11b`), l'autre pour les variables bloquées (stockage sous forme symétrique compressée).

A chaque itération, l'approximation du hessien est mise à jour par `emc11z`. Lorsqu'une borne est ajoutée à (retranchée de) l'ensemble des bornes actives, la décomposition du hessien est mise à jour par `eajc1` (`eretc1`). La décision de relâcher une borne est prise par `ecomp1`, qui appelle `emlag1` (calcul des multiplicateurs du problème quadratique).

5.3 Rôle de epsabs

[12] On note

$$I^- := \{i : |x_i - b_i^{\inf}| \leq \mathtt{dxmin}_i\} \quad \text{et} \quad I^+ := \{i : |x_i - b_i^{\sup}| \leq \mathtt{dxmin}_i\}.$$

L'ensemble des indices des bornes jugées actives en x est alors $I := I^- \cup I^+$.

A l'itération courante, notons J le complémentaire de I dans $\{1, \ldots, n\}$, i.e., l'ensemble des indices des variables libres, et g le gradient. Posons alors

$$E^2 = \frac{1}{|J|} \sum_{i \in J} g_i^2 \ \mathtt{dxmin}_i^2.$$

E est donc la valeur moyenne des composantes du gradient de f correspondant aux variables libres, par rapport aux variables "adimensionnées" $\tilde{x}_i = x_i/\mathtt{dxmin}_i$. Soit d'autre part la tolérance[13]

$$\epsilon^2 = \frac{\mathtt{epsabs}^2}{n} \sum_{i=1}^{n} \mathtt{dxmin}_i^2.$$

[9] Ça me parait un peu vague ...

[10] Ça vaudrait la peine de faire un pas de gradient avec projection pour relâcher un grand nombre de bornes à la fois et entraîner du même coup la convergence (voir Moré ...).

[11] Avec quel `df1`?

[12] Il faudrait d'abord définir le I approché. D'accord avec ce qui suit?

[13] Il est difficile de savoir à partir de ceci s'il faut calculer `epsabs` en variables adimensionnelles ou pas. On ne comprend pas bien non plus pourquoi certaines quantités sont calculées à partir des variables adimensionnelles et d'autres à partir des variables réelles. Peux-tu en dire davantage?

Si $E \leq \epsilon$, alors on considère que l'on a convergé dans le sous espace défini par les bornes actives. Dans ce cas, on teste les multiplicateurs pour vérifier l'optimalité ou éventuellement relâcher une contrainte.

Regardons d'abord si l'itéré courant x est optimal (on suppose toujours que $E \leq \epsilon$). Si pour tout $i \in I$ (les bornes actives), on a

$$\begin{array}{ll} g_i \,.\, \texttt{dxmin}_i \leq \epsilon & \text{lorsque } x_i \geq b_i^{\sup} - \texttt{dxmin}_i \\ g_i \,.\, \texttt{dxmin}_i \geq -\epsilon & \text{lorsque } x_i \leq b_i^{\inf} + \texttt{dxmin}_i \end{array}$$

alors on considère que tous les multiplicateurs ont le bon signe et M2QN1 s'arrête normalement (`mode` $= 1$).

En sortie (normale ou non), `epsabs` contient la racine carrée de

$$\frac{1}{|J|} \sum_{i \in J} g_i^2,$$

c'est-à-dire la valeur moyenne du gradient de f correspondant aux variables libres, par rapport aux "vraies" variables x (sans tenir compte de `dxmin`).

5.4 Circonstances de l'arrêt

Une sortie en `mode` < 6 est satisfaisante ou peut en général être corrigée par une action directe de l'exécutant.

Une sortie en `mode` > 6 correspond à un échec sans recours. Sauf erreur grossière dans l'appel de M2QN1, cela signifie en tous cas que le problème est terriblement mal conditionné car les sous-programmes matriciels sont tout à fait fiables (Harwell).

La sortie `mode` $= 6$ est douteuse.

- Si `dxmin` est grand, cette sortie est normale puisque l'on n'exige pas une très grande précision.
- Si `dxmin` est petit, alors la sortie `mode` $= 6$ signifie que le comportement du gradient est incohérent vu le comportement de la fonction. Cette incohérence ne peut être imputée aux erreurs d'arrondi que si `epsabs` aussi est petit.
- Par conséquent, si `dxmin` est petit et `epsabs` grand, alors on peut parier que le calcul des dérivées dans le simulateur `simul` est erroné. La vérification de ce calcul est grandement facilitée par `imp` $= 4$ en calculant des différences divisées, comme il est expliqué dans la section 6 de la notice de `mlis0`.

6 Double précision

Le programme N2QN1 effectue le même travail que M2QN1, sauf que tous les identificateurs autres que `rzs` sont déclarés par

```
implicit double precision (a-h, o-z)
```

Il se compose de `n2qn1`, `n2qn1a`, `nlis0`, `fmc11a`, `fmc11b`, `fmc11e`, `fmc12a`, `fajout`, `fretir`, `fchang`, `fsqrt`, `fmulti`.

L'analyse post-optimale est effectuée dans `nqhess`, `f1qhess`, `f2qhess`.

14

[14] J'ai enlevé la section "Résumé" qui m'a paru une répétition moins utile dans un texte écrit en Latex. Avis?

References

[1] R.H. Byrd, G.A. Schultz (1982). A practical class of globally convergent active set strategies for linearly constrained optimization. University of Colorado. (À paraître dans *Mathematical Programming*)

[2] P.E. Gill, W. Murray, M.H. Wright (1981). *Practical optimization*. Academic Press.

[3] B.A. Murtagh, R.W.H. Sargent (1969). A constrained minimization method with quadratic convergence. In R. Fletcher (Ed.), *Optimization*, Academic Press.

[4] P. Wolfe (1972). On the convergence of gradient methods under constraints. *IBM Journal of Research and Development*, 16, 407-411.