EPFL
ENAC INTER TRANSP-OR
Prof. M. Bierlaire

Recherche opérationnelle

Automne 2013/2014

Serie 3

Problème 1

Afin de pouvoir implémenter a l'aide du logiciel Matlab les différentes méthodes proposées au course pour résoudre un problème non-linéaire sans contrainte, un programme d'optimisation vous est donné a l'adresse suivante :

http ://transp-or.epfl.ch/courses/rofall2013/exercices.php

Il comprend six fichiers. Le fichier principal, qui se nomme **optSerie3.m**, fournit l'pssature d'un algorithme classique de descente. Il utilise les cinq fonctions suivantes :

direction.m : donne une direction,
taillepas.m : calcule une taille de pas,
normGradient.m : fournit un critere d'arret,
visual3d.m : permet la visualisation des résultats,
f1.m : évalue la fonction objectif, son gradient et son hessien.

La fonction calculée par **f1.m** est :
$f(x, y) = x^2 + 3y^2$

Methodes de descente

1. Télécharger les différents fichiers Matlab sur votre machine. Familiarises-vous avec ces fichiers en faisant varier la valeur des differents paramètres du programme.

2. Implémenter une fonction **direction_descente.m** qui calcule *la direction de la plus forte pente.*

3. Implémenter une fonction **direction_newton.m** qui donne la *la direction de Newton locale.*

4. Implémenter une fonction **taillepas_optquad.m** qui donne le pas suivant :
   $\alpha_k = (\nabla\ f(x_k)^T\ \nabla f(x_k))/\ (\nabla f(x_k)^T \nabla^2 f(x_k) \nabla f(x_k))$
   Comment est-il choisi ce pas α_k ?

5. Construire une fonction **taillepas_wolfe.m** implémentant *la première condition de Wolfe* (décroissance suffisante) pour le calcul de la longueur de pas :
   $f(x_k + \alpha_k d_k) \leq f(x_k) + \alpha_k \beta \nabla f(x_k)^T d_k$ avec $\beta \in]0,1[$.

   Il est en effet possible de se passer de la deuxième condition (pas suffisamment grand) en utilisant pour une stratégie dite de "'backtracking"' pour le choix du pas α_k :

Définir $\gamma \in (0,1)$,

Le pas choisi sera le premier élément de la suite 1, γ, γ^2, γ^3 ... qui satisfait la condition ci-dessus.

6. Tester l'efficacite des différentes combinasions entre calcul des directions d_k et politique de calcul du pas α_k décrits ci-dessus sur la fonction **f1.m**, sur la fonction donnée par :
   $f(x, y) = 1/2\ (y - x^2)^2 + 1/2\ (1 - x)^2$.

   et sur la fonction de Rosenbrock (implémentée lors de la serie 1) :
   $f(x, y) = 100\ (y - x^2)^2 + (1 - x)^2$.

   Comparer les résultats obtenus en termes de robustesse (atteint-on le minimum ?) et de vitesse de convergence (en combien d'itérations ?). Identifiez les combinasions les plus efficaces et expliques pourquoi on obtient des différences.

   Remarque
   Pour utiliser la rechereche linéaire inexacte basée sur la condiiton de Wolfe, il est nécessaire que d_k soit une direction de descente. Si l'on souhaite combiner cette recherche linéaire avec la direction de Newton, celle-ci doit donc etre modifiée lorsqu'elle ne constitue pas une direction de descente !

Problème 2

Le but de cette série est de résoudre le problème des moindres carrés à l'aide des algorithmes de Gauss-Newton et du filtre de Kalman.

La structure du filtre de Kalmann est donnée par les fichiers `moindrecar.m`, qui est le fichier principal, et `kalman.m` qui contient la partie itérative (à implémenter !).

Évaluez, à l'aide de MATLAB, x tel que :

$$x = \arg\min_{x \in \mathbb{R}^n} \|Ax - b\|^2$$

où A est une matrice $m \times n$ et b un vecteur de dimension m définis par :

a) $A = \begin{pmatrix} 2 & 3 & 4 \\ 5 & 1 & 1 \\ 5 & 3 & 2 \\ 4 & 1 & 2 \end{pmatrix}$ et $b = \begin{pmatrix} 21 \\ 19 \\ 24 \\ 19 \end{pmatrix}$.

b) A et b sont définis par blocs de la manière suivante :

$$A = \begin{bmatrix} C_1 \\ C_2 \\ \vdots \\ C_J \end{bmatrix} \text{ et } b = \begin{bmatrix} z_1 \\ z_2 \\ \vdots \\ z_J \end{bmatrix}$$

avec $A(10^6, 10)$, $b(10^6, 1)$ et $J = 100$. Les blocs de données C_j et z_j, pour $j = 1, \ldots, J$, sont fournis dans les fichiers correspondants `Cj.mat` et `zj.mat` qui se trouvent dans le fichier `matrix.zip` sur le site du cours.

A faire :

1. Résolvez les deux problèmes à l'aide des deux méthodes et comparez les résultats. Afin de comparer les temps d'esecution, vous pouvez utiliser la fonction *cputime* de Matlab.

2. Variez la valeur du paramètre λ du filtre de Kalman à temps réel. Quelles différences voit-on ? Expliquez pourquoi !

Aide MATLAB

i) La fonction de MATLAB qui résout les problèmes de moindres carrés est donnée par \ et s'utilise de la manière suivante :

```
> xopt=A\B
```

Aide de Matlab : If A is a square matrix, A \ B is roughly the same as inv(A)*B, except it is computed in a different way. If A is an n-by-n matrix and B is a column vector with n components, or a matrix with several such columns, then X = A\B is the solution to the equation AX = B computed by Gaussian elimination (see "Algorithm" for details). A warning message prints if A is badly scaled or nearly singular. If A is an m-by-n matrix with m ~= n and B is a column vector with m components, or a matrix with several such columns, then X = A \ B is the solution in the least squares sense to the under- or overdetermined system of equations AX = B. The effective rank, k, of A, is determined from the QR decomposition with pivoting (see "Algorithm" for details). A solution X is computed which has at most k nonzero components per column. If k<n, this is usually not the same solution as pinv(A)*B, which is the least squares solution with the smallest norm $\|X\|$.

ii) Afin d'importer les données C_j et z_j, pour $j = 1, \ldots, 100$, il suffit de se placer dans un répertoire contenant tous les fichiers `Cj.mat` et `zj.mat` puis d'utiliser la routine suivante pour un j donné :

```
> Cj=strcat('C',int2str(j));
> zj=strcat('z',int2str(j));
> load(zj) ;
> load(Cj) ;
```

De cette manière, le bloc C_j (resp. z_j) contenu dans le fichier `Cj.mat` (resp. `zj.mat`) sera importé dans votre fenêtre de commande sous la variable nommée C (resp. z).

iii) *Aide de Matlab* : CPUTIME returns the CPU time in seconds that has been used by the MATLAB process since MATLAB started.
For example :
`t=cputime; your_operation; cputime-t`
returns the cpu time used to run your_operation. The return value may overflow the internal representation and wrap around.