PW1 – Applied optimization – M2 E3A SAM

Authors : L. NOUVELIERE, M. CHADLI

OPTIMISATION LOCALE SANS CONTRAINTE

L'objectif de ce TP est d'utiliser le logiciel Matlab (MathWorks) afin de comparer différents algorithmes de recherche linéaire pour la minimisation locale d'une fonction définie sur R^n.

PARTIE I :

On cherche à programmer tout d'abord la méthode du gradient avec une stratégie de type backtracking avec condition d'Armijo.

La condition d'Armijo pour un point de départ X_0 et une direction de descente d s'écrit:

$$J(X_0 + \alpha d) \leq J(X_0) + \beta\alpha < d, \nabla J(X_0) >$$

1) Ecrire une fonction MATLAB ayant pour arguments J, ∇J , β, α_{init}, τ (paramètres du procédé de backtracking), X_0 (point initial) et N, fonction renvoyant la valeur de X_N. Pour cela :
   - On choisit $d = -\nabla J(X_0)$ parce que l'on sait que c'est une direction de descente.
   - $\alpha=\alpha_{init}$ pour garder l'information initiale sur α_{init}.
   - Une boucle *while* s'arrête quand on atteint « un bon alpha » (c'est-à-dire $J(X_0 + \alpha d) \leq J(X_0) + \beta\alpha < d, \nabla J(X_0)$. Sinon, on diminue α.
   - On stocke $X_0 + \alpha_d$ dans X_0. On stocke tous les X_0 dans X_N.

2) On cherche maintenant à minimiser la fonction suivante en 2D :
   $J(x,y) = 100(y - x^2)^2 + (x-1)^2$
   On prend pour cela :

   $\beta=0.1$

   $\alpha_{init}=1$

   $\tau=0.3$

   $X_0=(0,1)$.

   Représenter graphiquement sur deux figures séparées :

   (i) les lignes de niveau de la fonction J sur [-1,2] et les 100 premières itérations de l'algorithme précédent

   (ii) la fonction N → J(X_N) pour $N \in 0, \dots, 100$

Pour le point (i), on obtient un polynôme de degré 2 sur *y* qui dépend de *x*, donc deux solutions pour *y* : niveau+ et niveau- . Pour (ii), on utilisera une fonction MATLAB de contour.

PARTIE II :

On souhaite remplacer l'algorithme de backtracking par un nouveau procédé de dichotomie où le second critère (en plus du critère d'Armijo) est donné par la condition :

$$J(X_0 + \alpha d) > J(X_0) + \beta_2 \alpha < d, \nabla J(X_0) >$$

avec $0<\beta_2<\beta$. Ce second critère (condition de Goldstein) permet de sélectionner un pas suffisamment grand. Partant d'un intervalle $[0,\alpha_{max}]$ où α_{max} ne satisfait pas la condition d'Armijo, proposer et implémenter un algorithme permettant de trouver un pas convenable et comparer avec la solution précédente. On pourra prendre $\beta_2=0.001$.

Cette méthode évite d'évaluer ∇J beaucoup de fois, et nous permet de prendre un α suffisamment grand parce que l'on délimite la zone admissible. Donc on cherche un α qui vérifie :

$$\beta < \frac{J(X_0 + \alpha d) - J(X_0)}{\alpha < d, \nabla J(X_0) >} < \beta_2$$

qui est équivalent à :

$$\beta \alpha < d, \nabla J(X_0) >< J(X_0 + \alpha d) - J(X_0) < \beta_2 \alpha < d, \nabla J(X_0) >$$

PARTIE III :

En suivant la même démarche que dans la Partie I, écrire une nouvelle fonction MATLAB, utilisant la méthode de Newton pour minimiser une fonction *J* et la tester sur le même exemple que précédemment. La recherche linéaire est alors supprimée (prendre $\alpha=1$) et un nouvel argument apparaît, en l'occurrence le Hessien de J.

On appelle *g(x)*= ∇J(x) (donc *g'(x)=H(x)*). On obtient l'équation de la tangente

$$Y = g(X_k) + g'(X_k).(x - X_k)$$

Pour la méthode de Newton, on veut chercher le point où *y=0*, c'est-à-dire :

$$0 = g(X_k) + g'(X_k) * (X_{k+1} - X_k) \Rightarrow X_{k+1} = X_k - g'^{-1}(X_k) g(X_k)$$

PARTIE IV :

On refait la partie III, en appliquant la méthode BFGS (à noter que l'argument H disparaît).

Avec la méthode BFGS, on évite de calculer l'inverse de H avec une approximation de celle-çi. La structure du programme est identique à celle de la Partie III, mais avec la construction de la matrice avant :

$$G_k = G_{k-1} + \frac{y'_{k-1} y_{k-1}}{y'_{k-1} d_{k-1}} - \frac{G_{k-1} d_{k-1} d'_{k-1} G_{k-1}}{d'_{k-1} G_{k-1} d_{k-1}}$$

$$\begin{cases} y_{k-1} = g(X_k) - g(X_{k-1}) \\ d_{k-1} = X_k - X_{k-1} \end{cases}$$

Pour la première itération, on choisit : $H=I_d$.

PARTIE V :

Conclure sur l'efficacité comparée des méthodes de gradient, Newton et BFGS.