UFR
DE MATHÉMATIQUES
ET INFORMATIQUE

UNIVERSITÉ
PARIS
DESCARTES

MASTER 1 INFO 2018–2019

Optimisation Algorithmique

Polycopié et notes autorisés. Durée 1h15.

Fichiers disponibles :

1. Cours, fiches de TD/TP et corrigé des TP dans le répertoire COURS_TP ;
2. dans le répertoire SCILAB le polycopié Scilab ;
3. les fichiers CC_Ex1.sci, CC_Ex1.sce, CC_Ex2.sci et CC_Ex2.sce contiennent déjà une partie du code à utiliser, souvent il suffit d'enlever les mise en commentaire adéquats pour la question posée.

À remettre :

1. les fichier `CC_Ex1.sci`, `CC_Ex1.sce`, `CC_Ex2.sci` et `CC_Ex2.sce` des fonctions respectivement des commandes Scilab avec votre nom en commentaire ;
2. la copie double sur laquelle vous pouvez expliquer ce que vous avez fait : calculs, formules, graphiques, problèmes rencontrés,...
3. le sujet avec
   NOM PRÉNOM / NUMÉRO ÉTUDIANT :

. . . . . .

Exercice 1

On considère la fonction définie sur $\mathbb{R}$ par $f(x) = \ln(e^x + e^{-x})$.

1. Calculer $f'(x)$ et $f''(x)$.
2. Déterminer $x^* = \arg\min_{x\in\mathbb{R}} f(x)$
3. On veut illustrer le comportement de la méthode de descente de gradient et de la méthode de Newton pour différentes valeurs de $x^{(0)}$.
   Utiliser comme valeurs initiales $x^{(0)} = 0,5$, $x^{(0)} = 1$ et $x^{(0)} = 1,1$.
   Comparer les vitesses de convergence et commentez les résultats affichés.

Exercice 2

On s'intéresse à la minimisation de la fonctionnelle coût $f(x) = \sum_{k=1}^{p} e^{<a_k,x>+b_k}$,

où $x = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \in \mathbb{R}^2$, $a_k = \begin{pmatrix} (a_k)_1 \\ (a_k)_2 \end{pmatrix} \in \mathbb{R}^2$ et $b_k \in \mathbb{R}$ pour $1 \leq k \leq p$.

On note $b = (b_1 \cdots b_p)^t \in \mathcal{M}(p,1)$ et A la matrice formée à partir des vecteurs $a_k \in \mathbb{R}^2$:

$$A = \begin{pmatrix} (a_1)_1 & (a_1)_2 \\ (a_2)_1 & (a_2)_2 \\ \vdots & \vdots \\ (a_p)_1 & (a_p)_2 \end{pmatrix} \in \mathcal{M}(p,2)$$

1. Expliquer brièvement le code Scilab `v = sum(exp(A*x+b))` utilisé.
2. Expliquer ce que le code Scilab `A*x_min+b` calcule.
3. On considère la fonction

$$f(x) = e^{x_1+3x_2-0.1} + e^{x_1-3x_2-0.1} + e^{-x_1-0.1}\,.$$

   Calculer $\nabla f(x)$.
   Utiliser la matrice A fourni dans le fichier CC_Ex2.sce pour illustrer ce qui suit.
4. Pour comparer les deux méthodes, on doit atteindre la même précision, pour cela chaque algorithme devra par exemple s'arrêter dès que la valeur courante de la fonction coût, `v`, est proche à 10^{-10} près, de la valeur minimale obtenue par un calcul distinct et appelée `v_min`.

   Indiquer comment et où est calculé `v_min` dans le code fourni.

   Adapter le code des fonctions `[x_h,v_h] = methode_gradient(x0,c,rho)` qui utilise la méthode de descente du gradient en utilisant un "backtracking" et `[x_h,v_h]=methode_newton(x0)` afin d'avoir ce d'est d'arrêt.
5. Comparer les performances et le comportement des deux méthodes en affichant des lignes de niveau et le trajet de la suite des points minimisants pour différentes valeurs initiales. Commentez les résultats.
6. Coder la fonction

$$f(x) = e^{x_1+3x_2-0.1} + e^{x_1-3x_2-0.1} + e^{-x_1-0.1}\,.$$

   dans le fichier CC_Ex2.sce, *i.e.* donner A et b, et recommencer les tests.