TP 2

On rappelle qu'on veut résoudre le problème

$$\begin{array}{ll} \min & F(x) = x^T A x \\ \text{s.c.} & \left\{ \begin{array}{l} e^T x = 1 \\ r^T x = \rho \\ x \geq 0, \end{array} \right. \end{array}$$

pour cela on a introduit la fonction objectif pénalisée

$$F_\mu(x) = F(x) - \mu \sum_i \log(x_i) - \mu \log(r^T x - \rho),$$

où $\mu \in \mathbb{R}_+^*$. On est donc ramené au problème

$$\begin{array}{ll} \min & F_\mu(x) \\ \text{s.c.} & \left\{ \begin{array}{l} e^T x = 1, \end{array} \right. \end{array}$$

qu'on va résoudre par la méthode du gradient projeté à pas fixe. Les fichiers à télécharger sont disponibles sur la page web : `http://www.ceremade.dauphine.fr/~levitt/optinum/`

1. Coder les fonctions `oracle_tp2` et `proj_tp2` de l'exercice 5, et les tester.
2. Télécharger le fichier `gradfix.m`, et l'utiliser comme modèle pour une fonction `gradproj` qui accepte en paramètre une fonction `proj` chargée de réaliser la projection. Pourquoi ne peut-on pas utiliser la norme du gradient comme critère d'arrêt ?
3. Comment choisir le pas de la méthode de gradient ? Le point initial ?
4. Tester votre algorithme avec

$$A = \begin{pmatrix} 4 & 0 & 0 \\ 0 & 64 & 0 \\ 0 & 0 & 100 \end{pmatrix}, \; r = \begin{pmatrix} 5 \\ 10 \\ 15 \end{pmatrix}.$$

   Pour ceci, télécharger le fichier de données `data.m` (à placer dans votre répertoire courant), où sont définis 3 jeux de données, la matrice A et le vecteur r ci-dessus correspondent au premier. Voici un exemple d'utilisation :

```
...
global A r e rho mu
data                        % initialisation de la structure stocks
A = stocks(1).covar         % on utilise le premier jeu de données...
r = stocks(1).returns
...
```

   Étudier l'impact de ρ et μ. Idéalement, comment devrait-on choisir μ, le paramètre de pénalisation ?
5. Modifier la procédure `gradproj` pour qu'elle retourne la liste des itérés x_n et tracer la courbe de convergence $\|x_{n+1} - x_n\|$ en fonction de n.

6. L'analyse théorique de l'algorithme de gradient mène à un résultat de convergence linéaire du type

$$\|x_n - x^*\| \leq K\nu^n$$

où $K > 0$, et $\nu < 1$ est la constante de convergence.
Combien d'itérations faut-il pour obtenir une précision de 10^{-16} sur x ?
Prouver que

$$\|x_{n+1} - x_n\| \leq K'\nu^n$$

Montrer qu'un tracé en échelle semi-logarithmique permet de vérifier visuellement cette inégalité, et en utilisant la fonction `polyfit` de Matlab, obtenir numériquement une majoration de la constante ν.

7. Comment la constante de convergence ν varie-t-elle en fonction du pas t ?
8. Recommencer l'analyse de la question de 4 avec les 2 autres jeux de données, et tracer les courbes de convergence. Qu'est-ce qui change ?
9. On souhaite maintenant utiliser une autre méthode que la descente à pas fixe. Pour cela, on va utiliser un algorithme de recherche linéaire qui est une variante de la méthode de dichotomie : "golden section search"[1].
   Télécharger le fichier `goldensearch.m` qui définit une procédure prenant en paramètre une fonction $f : \mathbb{R} \to \mathbb{R}$ et une tolérance δ, et qui rend la valeur approchée du minimum de cette fonction dans l'intervalle $[0, \infty)$. Survoler le code pour comprendre le principe de la méthode.
   Écrire une fonction `gradproj_gold` (basée sur la fonction `gradproj`) qui utilise `goldensearch`.
   Indication : pour la recherche linéaire, on peut définir une fonction anonyme en Matlab en utilisant la syntaxe suivante :

   ```
   linesearch = @(t) oracle_tp2(x-t*g)
   ```

   ce qui définit la fonction `linesearch` qui prend en paramètre `t` et renvoie `oracle_tp2(x-t*g)`.
10. Tracer les nouvelles courbes de convergence.
11. Modifier les fonctions `goldensearch` et `gardproj_gold` pour que cette dernière retourne le nombre total d'appels à l'oracle, et le comparer au nombre d'appels pour le gradient à pas fixe.

1. Introduite en 1953 par Keifer : `http://en.wikipedia.org/wiki/Golden_section_search`