Université de Strasbourg
M1 - Optimisation

TP 2 : Optimisation avec contrainte

Exercice 1 (mise en oeuvre d'une méthode de gradient projeté et de pénalisation)**.** On considère la fonction J définie sur $\mathbb{R}^2$ par : $J(x, y) = 2x^2 + 3xy + 2y^2$. On appelle Q le cadran défini par :

$$Q = \{x \leqslant -\frac{1}{2}, y \leqslant -\frac{1}{2}\}.$$

1. Analyse théorique du problème.
   a) Quelle est la solution du problème de minimisation de J sans contrainte ?
   b) On appelle $\bar{X}$, le minimum de J sous la contrainte $\bar{X} \in Q$. Démontrer que, nécessairement, $\nabla J(\bar{X}) = 0$ ou $\bar{X} \in \partial Q$. En déduire la solution du problème sans utiliser le théorème de Kuhn-Tucker.
2. Méthode du gradient projeté. Mettre en oeuvre la méthode du gradient projeté pour résoudre le problème de minimisation
   $$\inf_{(x,y)\in Q} J(x, y)$$
   Représenter les itérés obtenus à l'aide de cette méhode.
3. Méthode de pénalisation. Nous allons reprendre le même problème que précédemment, et évaluer une méthode de pénalisation. On propose les étapes suivantes :
   a) Mettre en place une fonction de pénalisation $x \mapsto \phi(x)$, en réfléchissant à l'expression qu'elle aura sur le cadran Q (attention la fonction pénalisée doit être différentiable). Déterminer alors le gradient de la fonction pénalisée.
   b) Tracer les courbes de niveau de la nouvelle fonction coût et son gradient, pour plusieurs valeurs de ε. Que constatez-vous ?
   c) Pour une valeur petite de ε, par exemple $\varepsilon = 10^{-4}$ (pénalisation forte), et un point de départ $x_0 = (-0.3, 0.5)^T$, tester la méthode de pénalisation pour la méthode de gradient à pas fixe. En visualisant les itérés, commenter la vitesse de convergence. Répéter le test pour $\varepsilon = 0.5$ (pénalisation faible). Que peut-on dire de la convergence ? Quid de la solution trouvée ?
   d) Pour aller plus loin (optionnel). Implémenter la méthode du gradient à pas optimal pour le problème pénalisé.

Exercice 2 (méthodes de type gradient pour des fonctions quadratiques)**.** On considère le problème de l'optimisation d'un portefeuille. Supposons que l'on possède n actions, que l'on représente par des variable aléatoires $R_1, \ldots, R_n$. Chaque action rapporte en moyenne à l'actionnaire $e_i = \mathbb{E}(R_i)$ (espérance de R_i) au bout d'un an. On suppose que l'on investit une somme S donnée et on note $x_i \in \mathbb{R}$, la proportion de la somme investie dans l'action i. Ainsi, on a $\sum_{i=1}^n x_i = 1$ que l'on réécrira $\langle x, u\rangle = 1$ avec $u = [1, \ldots, 1]^T$. Le portefeuille total est représenté par la variable aléatoire $R = \sum_{i=1}^n x_i R_i$ et rapporte donc en moyenne

$$\mathbb{E}(R) = \sum_{i=1}^{n} x_i e_i = \langle x, e\rangle.$$

On désire imposer un rendement donné $r_0 > 0$, ce qui se traduit par $r_0 = \langle x, e\rangle$. On modélise le risque du portefeuille par $\sigma^2(x) = \mathbb{E}[(R - \mathbb{E}(R))^2]$. On note $A = (a_{ij})_{1\leqslant i,j\leqslant n}$ la matrice de covariance définie par la relation

$$\forall (i, j) \in [\![1, n]\!]^2, \quad a_{ij} = E[(R_i - \mathbb{E}(R_i))(R_j - \mathbb{E}(R_j))].$$

On peut alors écrire que $\sigma^2(x) = \langle x, Ax\rangle$. On appelle J, la fonction définie sur R^n par

$$J(x) = \frac{1}{2}\langle x, Ax\rangle.$$

On désigne par K, l'ensemble des contraintes, soit

$$K = \{x \in \mathbb{R}^n \mid \langle h, u\rangle = 1 \text{ et } \langle x, e\rangle = r_0\}.$$

L'objectif de cet exercice est de résoudre numériquement le problème

$$\inf_{x\in K} J(x).$$

1. Mettre l'ensemble des contraintes sous la forme $K = \{x \in \mathbb{R}^n : Cx = f\}$, où C et f désignent respectivement une matrice et un vecteur à préciser. Rappeler comment se traduisent les conditions d'optimalité de ce problème et formuler l'équation en x à résoudre à chaque itération.

2. Supposons que $r_0 = 2.5$ et $e_i = i$ pour tout $i \in [\![1, n]\!]$. Pour les tests numériques, on se placera par exemple dans le cas où $n = 5$.Voici un programme permettant de générer une matrice A :

   ```
   A = np.diag(e)
   R = np.random.rand(n,n)
   A = A + 0.05 * (np.transpose(R) @ R)
   ```

   Expliquer la dernière ligne du programme.

3. Pour différentes matrices A, programmer l'algorithme d'Uzawa. On n'oubliera pas d'imposer un nombre maximal d'itérations. Quel inconvénient majeur constatez-vous ici ?

4. Adapter l'algorithme pour palier ce problème.