Exercices du cours d'analyse convexe et d'optimisation non différentiable

Chaque étudiant rendra un rapport décrivant les calculs effectués, les résultats obtenus, les scripts ayant permis de les obtenir et les sorties de matlab/scilab correspondantes (commande `diary`). Le rapport sera au format PDF et inclura les graphiques. L'ensemble sera archivé sans sous répertoires au format zip dans un fichier nommé *nom_prenom.zip*. Il sera envoyé par email à `nicolas.omont_tp_ensta à m4x.org` pour le Mardi 1er Mars au plus tard.

Première partie

On définit à minimiser la fonction f définie par :

$$f(x) = \max(0.005\,x^2 - x, 0.05x^2 - 2x, 0.5x^2 - 3x, 0.1x^2 - 4x, 0.03\,(x-1)^4 - 3x)$$

1. Montrer que cette fonction est bien convexe.
2. Dans un premier temps, on écrira un programme calculant la valeur de cette fonction en un point donné, sous Scilab ou Matlab. Le programme, qui pourrait s'appeler `simul`, prend comme entrées le point x et rend la valeur de la fonction f et un sous-gradient g en ce point :

   ```
   function [f,g]=simul(x)
   ```

   Notes : - il n'est pas nécessaire de « vectoriser » la fonction. On pourra faire l'hypothèse que x est scalaire et non un vecteur.

   - ne pas calculer une approximation d'un sous-gradient par différence finie, mais en dérivant symboliquement la fonction (à la main par exemple) puis en implémentant la dérivée obtenue.
3. On essaiera ensuite de résoudre ce problème par différentes méthodes :
   a. D'abord, on visualisera f entre -6 et 8 ainsi qu'un sous-gradient
   b. Ensuite, on appliquera la méthode de sous-gradient :
      i. On implémentera la fonction `sous_gradient` dont la signature est donnée ci-dessous. Les arguments sont les suivants :

         `fun` : pointeur vers la fonction à minimiser (Par la suite, lors de l'appel de la fonction `sous_gradient`, pour passer la fonction dont le nom est `simul`, on écrira comme argument `@simul`).
         `pt_initial` : point initial de l'algorithme
         `nb_iter` : nombre d'itérations à effectuer
         `epsilon` : pas de l'algorithme
         `mode_it` : méthode :
            1 : pas constants de longueur $epsilon$
            2 : pas de longueur $epsilon/k$ à la k^{ème} itération
            3 : pas de longueur $epsilon/\sqrt{k}$ à la k^{ème} itération
         `mode_graphe` :
            0 : pas d'affichage des itérations

1 : affichage des itérations en superposition sur le graphe de la fonction

`sx` : séquence des points x

`sy` : séquence des points y=f(x)

`f` : dernière valeur y obtenue

```
function[sx,sy,f]=sous_gradient(fun,pt_initial,nb_iter,epsilon,mode_it,mode_graphe)
```

ii. On donnera le tableau des valeurs finales obtenues pour les cas suivants :

pt_initial	epsilon	nb_iter	mode_it
0 :2 :8	0.1	100	1 :3
-6 :2 :2	0.1	1000	1 :3
3	1	1000	1

iii. Tracer les itérations successives pour quelques exemples dans le mode à pas constant. Que se passe-t-il si le point de départ est supérieur à 6 dans le mode à pas constant avec $epsilon$ assez grand ? Pourquoi ?

Enfin, on appliquera la méthode des plans sécants en ajoutant au problème les contraintes de bornes suivantes : $-6 \leq x \leq 8$

iv. Ecrire la fonction ci-dessous dont les arguments sont :

`fun` : nom de la fonction à minimiser.

`xmin` : borne inférieure de x

`xmax` : borne supérieure de x

`tolf` : arrêt de l'algorithme quand l'écart entre la « vraie » valeur de la fonction au point x et la valeur approchée par les plans sécants (avec x minimisant la fonction approchée par les plans sécants) est plus petite que `tolf`.

`maxit` : nombre maximum d'itération

`mode_graphique` :

0 : pas d'affichage des plans sécants

1 : affichage des plans sécants (ici des droites) en superposition sur le graphe de la fonction

`xf` : séquence des points x où les plans sécants sont calculés

`f` : dernière valeur y obtenue

On pourra utiliser la fonction `linpro` (présente dans scilab 4 mais absente dans scilab 5, à défaut on peut utiliser `quapro`) ou `linprog` (matlab) pour résoudre le problème linéaire associée à chaque itération de la méthode des plans sécants. On n'oubliera pas les contraintes de bornes.

```
function [xf,f]= plans_secants(fun,xmin,xmax,tolf,maxit,mode_graphique)
```

v. Exécuter la méthode des plans sécants et afficher le graphique correspondant. Comparer la méthode du sous-gradient et la méthode des plans sécants (complexité d'une itération, critère d'arrêt…)

Seconde partie

En seconde partie du TD, nous allons résoudre le problème :

$$\text{minimiser : } 10\,p_1 + p_1^2 + 2p_2 + 2p_2^2$$
$$\text{sous les contraintes:}$$
$$p_1 + p_2 = 20$$
$$0 \leq p_1 \leq 12$$
$$0 \leq p_2 \leq 12$$

1. Résolution directe
    a. On utilisera pour cela la fonction `quapro` (Scilab < 5.0) ou `quadprog` (Matlab).
    b. On vérifiera graphiquement le résultat en remplaçant p_2.par son expression en fonction de p_1 dans l'objectif.

2. Résolution par décomposition
    a. Ecrire le problème d'optimisation de la fonction duale associée au problème d'optimisation. Cette fonction dépendra uniquement de la variable duale associée à la contrainte égalité. Les contraintes de bornes ne seront pas dualisées mais seront conservées telles quelles dans le problème dualisé. Montrer que la fonction duale se décompose en deux sous-parties.
    b. En utilisant les conditions de Karush, Kuhn et Tucker (KKT), résoudre analytiquement les 2 problèmes de minimisation présent dans la fonction duale afin d'obtenir une forme explicite de celle-ci (dépendant uniquement de la variable duale de la contrainte égalité). On implémentera alors la fonction suivante qui revoit la valeur de la fonction duale `f` et un sous-gradient (`grad`) au point `lambda` où `lambda` est la variable duale associée à la seule contrainte égalité du problème.

```
function [f,grad] = simul2(lambda)
```

    c. Maximiser par la méthode du sous-gradient et par la méthode des plans sécants la fonction duale. Pour cheque résultat, on calculera la solution primale associée. Avec la méthode du sous-gradient, on fera les tests suivants :

pt_initial	epsilon	nb_iter	mode_it
-50 :10 :0	1	10	1 :3
-50 :10 :0	1	1000	1 :3
-10	10	1000	1