Projet fin d'étude - optimisation algorithmique

Question 1

1. Écrire une fonction

   `descente_gradient_pas_fixe(f,gradf,x0,eta,eps)`

   qui, étant donnée une fonction $f : \mathbb{R}^n \to \mathbb{R}$ et son gradient ∇f, implémente l'algorithme de descente de gradient avec point initial $x^{(0)} \in \mathbb{R}^n$, un pas fixe $\eta > 0$, et une condition d'arrêt $\|\nabla f(x)\| < \varepsilon$. En sortie, la fonction devra retourner X, fX de tailles (n, K) et $(1, K)$ respectivement, K étant le nombre d'itérations final, et donnant les séquences des valeurs $x^{(k)}$, $f(x^{(k)}$ obtenues lors des itérations successives.

2. On considère les fonctions

   $$f_1(x) = x_1^2 + 2x_2^2 + x_1x_2 + x_1 - x_2 + 30.$$

   $$f_2(y) = x_1^2 + 10x_2^2.$$

   Montrer qu'elles satisfont les conditions du théorème de convergence de l'algorithme de descente de gradient à pas fixe pour des valeurs du pas $\eta \in]0, \eta_c[$ où η_c est à déterminer dans chaque cas.

3. Tester la descente de gradient avec les fonctions f_1 et f_2, des valeurs de η compatibles, $\varepsilon = 10^{-5}$ et $x^{(0)} = (3, 3)$. Dans chaque cas, afficher les points $x^{(k)}$ sur la fenêtre représentant les lignes de niveaux de la fonction (obtenu à l'aide de la fonction `plot_fun` fournie), ainsi que le point x^* correspondant au minimiseur exact (à déterminer par le calcul).

4. Pour chacune des deux fonctions, exécuter à présent les commandes :

   ```
   import matplotlib.pyplot as plt
   plt.semilogy(fX-f(xstar))
   ```

   (où xstar est un vecteur de taille 2 donnant la valeur exacte du minimiseur). Ceci affiche les valeurs de la suite $f(x^{(k)}) - f(x^*)$ avec une échelle logarithmique sur l'axe des ordonnées.

5. Écrire à présent une fonction

   `descente_gradient_rebr(f,gradf,x0,alpha,beta,eps)`

   qui implémente la descente de gradient avec méthode de rebroussement pour la recherche du pas. Tester cette méthode avec la fonction f_2 précédente, en prenant $\alpha = 0.1$ et $\beta = 0.7$, et afficher à nouveau les points $x^{(k)}$. Combien d'itérations ont été nécessaires pour atteindre le critère d'arrêt ? Comparer avec la méthode à pas fixe.

6. Écrire une fonction `hessf2(x)` qui calcule la matrice hessienne $Hf_2(x)$ pour un point x donné.

7. Écrire une fonction

   `descente_Newton(f,gradf,hessf,x0,alpha,beta,eps)`

qui implémente la méthode de Newton avec méthode de rebroussement pour la recherche du pas. Tester cette méthode avec la fonction f_2 précédente, en prenant toujours $\alpha = 0.1$ et $\beta = 0.7$. Combien d'itérations ont été nécessaires pour converger ? Pourquoi ? On testera la méthode de Newton dans une situation plus pertinente à l'exercice suivant.

8. Écrire une fonction

`gradient_conjugue(A,b,c,x0,eps)`

qui implémente l'algorithme du gradient conjugué pour une fonctionnelle quadratique :

$$x \mapsto \frac{1}{2}\langle Ax\,,\,x\rangle - \langle b\,,\,x\rangle + c.$$

Tester cette méthode avec la fonction f_2 précédente, et comparer avec les algorithmes précédents.

Question 2

On considère la fonction suivante de $\mathbb{R}^2$ dans $\mathbb{R}$:

$$f_3(x) = e^{x_1+3x_2-0.1} + e^{x_1-3x_2-0.1} + e^{-x_1-0.1}.$$

1. Montrer que f_3 peut s'écrire $f_3 = u \circ v$ où $v : \mathbb{R}^2 \to \mathbb{R}^3$ est du type $v(x) = Ax + b$, avec A matrice et b vecteur à préciser, et $u : \mathbb{R}^3 \to \mathbb{R}$ est une autre fonction à préciser.
2. En déduire les expressions du gradient et de la matrice hessienne de f_3.
3. Montrer que f_3 est convexe.
4. Écrire la fonction `f3(x)` puis afficher le graphe et les lignes de niveaux de f_3 à l'aide de `plot_fun` sur le domaine $[-2, 1] \times [-1, 1]$.
5. Écrire les fonctions `gradf3(x)` et `hessf3(x)` puis tester la méthode de descente de gradient (avec méthode de rebroussement) ainsi que la méthode de Newton pour cette fonction, et les comparer (nombre d'itérations et temps de calcul).

Question 3

Ouvrir le fichier `script_poisson.py`. Ce script définit une fonctionnelle quadratique sur $\mathbb{R}^{100}$ $f(x) = \frac{1}{2}\langle Ax\,,\,x\rangle - \langle b\,,\,x\rangle$ où A et b sont choisis de telle manière que le minimiseur

$x^* = A^{-1}b$ corresponde à la solution de la discrétisation par différences finies du problème de Poisson :

$$\text{Trouver } u \in \mathcal{C}^2(]0,1[) \text{ telle que}$$
$$\Delta u(t) = t(1-t)e^{-t} \text{ et } u(0) = u(1) = 0 \text{ (conditions de Dirichlet).}$$

Ainsi, chaque point $x^{(k)}$ correspond aux valeurs d'une fonction u discrétisée sur l'ensemble $1/(n+1), 2/(n+1), \ldots, n/(n+1)$. On peut donc afficher les courbes correspondant à ces vecteurs.

Tester la méthode de descente de gradient (avec méthode de rebroussement) ainsi que la méthode du gradient conjugué pour cette fonction, et les comparer. Afficher dans un même graphique les courbes correspondant à la solution $x^{(k)}$ obtenue par l'algorithme (pour k la dernière itération), la solution x^* obtenue par résolution du système linéaire, ainsi que b.