TD et TP 2 : Méthodes de descente de gradient — Nom(s) :

On commence toujours par charger les librairies. Les questions avec une étoile sont facultatives (et difficiles). Les parties 1 et 2 sont indépendantes.

```
In [1]: %pylab inline
```

```
Populating the interactive namespace from numpy and matplotlib
```

1. Retour en dimension un

(a-TD) Montrer que les méthodes de Newton et de la sécante pour résoudre l'équation $f'(x) = 0$ peuvent être vues dans certains cas comme des méthodes de descente de gradient. Quel est alors la valeur du pas α_k, et quelles sont les conditions à satisfaire pour rentrer dans le cadre des méthodes de descente ?

(b-TD) On considère la fonction $f : x \mapsto x^2 - \frac{1}{4}x^4$. Écrire la relation de récurrence vérifiée par les points x_k correspondant à une descente de gradient à pas fixe α pour la fonction f. On suppose que aucun des x_k n'est nul (sinon la suite est stationnaire). Pour quelles valeurs de α est-on sûr que la suite des $f(x_k)$ est décroissante dès que x_0 est suffisamment proche de 0 ? Montrer que dans tous ces cas, la suite des x_k converge vers 0. Dans quels cas la convergence est-elle linéaire ou superlinéaire (donner alors le taux de convergence linéaire, ou l'ordre de convergence s'il existe) ?

(c-TP) Programmer numériquement la suite x_k pour des valeurs de α et de x_0 où la convergence est linéaire ou superlinéaire. Illustrer les différents types de convergence à l'aide de graphiques.

(d*-TP) Programmer la suite des x_k dans le cas où elle est convergente mais ne converge pas linéairement. Conjecturez un équivalent de $|x_n|$ lorsque $n \to \infty$. Indication : on pourra tracer $|x_n|$ en fonction de n avec des échelles logarithmiques en abscisses et en ordonnées (pour des valeurs de n assez grandes).

(e*-TD) Démontrer la conjecture de la question précédente. Indication : calculer la limite lorsque $n \to \infty$ de $\frac{1}{x_{n+1}^2} - \frac{1}{x_n^2}$.

(f-TD) On considère la fonction $f : x \mapsto \frac{x^2}{1+x^2}$. Écrire la relation de récurrence vérifiée par les points x_k pour la méthode de descente de gradient à pas fixe α. On suppose encore qu'aucun des x_k n'est nul.

Pour quelles valeurs de α la suite converge-t-elle lorsque x_0 est suffisamment proche de 0 ?

(g-TP) Programmer la suite x_k lorsque $x_0 = 3$ et $\alpha = \frac{1}{3}$. La convergence est-elle linéaire ? Si oui, quel est le taux de convergence ?

Discuter de l'intérêt ou non de ce genre de méthode par rapport par exemple à la méthode de la section dorée.

Indication : programmer au moins une centaine de points, et tracer également par exemple $\varphi^{-k}|x_0|$, où $\varphi = \frac{1}{2}(1 + \sqrt{5})$ est le nombre d'or...

2. Méthode de gradient à pas fixe, cas test en dimension 2.

On va tester la méthode de descente de gradient sur la fonction $f_a : (x_0, x_1) \mapsto 1 - \frac{1}{1+ax_0^2+x_1^2}$, où $a > 0$ est un paramètre qu'on pourra changer pour voir comment se comporte la méthode en fonction de a.

Attention au changement de notation : ici x_i est la $i$ème coordonnée d'un vecteur $X \in \mathbb{R}^n$. On commence par $i = 0$ pour être cohérent avec la numérotation en python. On notera les itérées X_k. On utilisera deux indices si on veut préciser les coordonnées, par exemple en dimension 2 on écrit $X_k = (x_{0,k}, x_{1,k})$.

(a-TP) Définir a comme une variable globale, et définir f_a. Elle ne doit prendre qu'un argument, la variable correspondant au vecteur X.

Tracer les lignes de niveau de la fonction f_a sur $[-5, 5] \times [-5, 5]$ pour différentes valeurs de a (faire plusieurs graphiques). On utilisera la fonction `contour`, qui peut s'appeler par exemple comme ceci : `contour(X0,X1,Z,12)`, où `X0` est de type `array` à une dimension (qui liste les abscisses x_0 des points du maillage), de même pour `X1` (pour les ordonnées), et où `Z` est un tableau de type `array` à deux dimensions qui contient les valeurs des $f_a(X)$, avec $X = (x_0, x_1)$ où x_0 parcourt la liste des abscisses et x_1 parcourt celle des ordonnées. Cela trace 12 lignes de niveau.

On pourra utiliser la commande `axis("scaled")` pour que l'échelle des ordonnées et des abscisses soit la même.

(b-TD) On veut approximer le gradient par différence finies. On peut choisir une des approximations suivantes (e_i étant le vecteur de la base).

— Différences finies : $\partial_i f(X) \approx \dfrac{f(X+\varepsilon e_i) - f(X)}{\varepsilon}$ pour $0 \leqslant i < n$.

— Différences finies centrées : $\partial_i f(X) \approx \dfrac{f(X+\varepsilon e_i) - f(X-\varepsilon e_i)}{2\varepsilon}$ pour $0 \leqslant i < n$.

Discuter de l'intérêt de l'une ou l'autre des méthodes. Quel serait le bon choix d'ordre de grandeur de ε dans chaque cas ?

(c-TP) Définir une variable globale ε qui correspondra au paramètre d'approximation (on n'aura pas besoin de le changer souvent, autant ne pas le mettre comme argument dans les méthodes). Définir une fonction qui prend en argument une fonction f dont on veut approximer le gradient, un point X (sous la forme d'un vecteur) et qui renvoie l'approximation du gradient par différences finies à droite.

Vérifier que cela fonctionne quand on l'applique à f_a en un point donné, que cela renvoie bien un vecteur de type `array` qui est proche du gradient au point X.

(d-TD) Calculer la Hessienne de f_a en 0. Quel pas prendre pour appliquer la méthode de descente de gradient à pas fixe ?

(e-TP) Définir une fonction qui prend en argument une fonction f à minimiser, un point initial X_0, un pas α, et une tolérance, et qui calcule la suite X_k correspondant à la méthode de descente de gradient à pas fixe pour f. On renverra la liste des X_k. On prendra comme critère d'arrêt le fait que la norme du gradient est plus petite que la tolérance (utiliser la fonction `norm` pour calculer la norme). Et on se mettra une limite au nombre de passages dans la boucle, au cas où on ne convergerait pas.

La tester sur la fonction f_a. On pourra tracer la suite des points sur le graphique de f_a à l'aide de `plot` et de marqueurs. Vérifier graphiquement que le gradient est bien orthogonal aux lignes de niveau.

Illustrer le taux de convergence de X_k vers 0 par un graphique.

3. Application à la recherche de trajectoires fermées sur un billard.

On se donne un convexe de $\mathbb{R}^2$, dont le bord est noté Γ. On cherche à placer n points $M_0, \dots, M_{n-1}$ sur Γ qui correspondent à une trajectoire de billard parfaite : l'angle entre la normale au bord et la trajectoire avant rebond doit être le même que celui entre la normale et la trajectoire après rebond.

Si on se donne M_0 et un angle (donc deux paramètres), alors les points $M_1, \dots M_{n-1}$ sont uniquement déterminés. On aimerait que la trajectoire après rebond en M_{n-1} passe par M_0 avec le même angle. On a donc deux conditions à satisfaire, avec deux paramètres, donc on espère qu'il puisse y avoir une solution !

(a-TD) On modélise d'abord notre problème. On se donne $t \mapsto \gamma(t) \in \mathbb{R}^2$ une paramétrisation 2π-périodique du bord Γ. Par exemple si le convexe est une ellipse, on prendrait $\gamma(t) = (a\cos t, b\sin t)$. On suppose que γ est de classe C^1 et que $\gamma'(t) \neq 0$ pour tout $t \in \mathbb{R}$.

On pose $M_i = \gamma(t_i)$ pour $0 \leqslant i < n$, et on pose $T_i = \frac{\gamma'(t_i)}{\|\gamma'(t_i)\|}$ le vecteur unitaire tangent à la courbe en M_i. Pour des raisons de notation, on pose $M_n = M_0$ et $T_n = T_0$. Lorsque les points consécutifs sont distincts, on pose $U_i = \frac{M_{i+1}-M_i}{\|M_{i+1}-M_i\|}$ le vecteur unitaire dirigé de M_i à M_{i+1} (et de même on pose $U_n = U_0$).

Montrer que la condition d'être une trajectoire de billard parfaite équivaut à

$$\forall i \in [\![1, n]\!],\ T_i \Delta (U_i - U_{i-1}) = 0,$$

où on a posé $U_n = U_0$ pour simplifier les notations. Faire un dessin pour le comprendre !

(b-TD) On veut transformer le problème en un problème d'optimisation. On va montrer que les solutions d'un problème d'optimisation bien choisi conduisent à des solutions du problème original.

On pose L la fonction de $\mathbb{R}^n$ dans $\mathbb{R}$ qui donne la longueur totale de la trajectoire passant successivement pas les points (on ne se préoccupe pas de savoir si la trajectoire est une trajectoire de billard).

$$L(t_0, \dots, t_{n-1}) = \|\gamma(t_0) - \gamma(t_{n-1})\| + \sum_{i=1}^{n-1} \|\gamma(t_i) - \gamma(t_{i-1})\|.$$

Montrer que la fonction L admet un maximum global sur $\mathbb{R}^n$ (on pourra utiliser la périodicité).

On considère maintenant $(t_0, \dots, t_{n-1})$ un point de maximum local de la fonction L. Montrer qu'alors on a pour tout $i \in [\![1, n]\!]$, $M_i \neq M_{i-1}$ (on pourra supposer que le billard est strictement convexe pour simplifier la démonstration, mais en fait cela fonctionne tout le temps). En déduire que la fonction L est différentiable en $(t_0, \dots, t_{n-1})$ et qu'alors la trajectoire est une trajectoire de billard parfaite.

On a donc obtenu que tout point de maximum local (et il en existe) correspond à une trajectoire de billard parfaite. Il existe donc au moins une trajectoire parfaite, et on va essayer d'en approximer numériquement.

(c-TP) Définir γ comme une fonction globale (on peut par exemple commencer par une ellipse). Définir ensuite L comme une fonction prenant comme argument un seul vecteur, le vecteur des t_i.

Appliquer la méthode de gradient à pas fixe pour obtenir des trajectoires de billards parfaites, en 3, 4, 5 bandes ou plus... On pourra éventuellement pour simplifier prendre une version modifiée de la méthode qui ne renvoie que le point final.

Quelle méthodologie appliquer pour choisir un pas fixe α qui convienne ?

Afficher le billard et les trajectoires obtenues.

(d-TP) Observer ce qui se passe lorsque l'on prend des points initiaux différents. Par exemple pour $n = 4$ ou $n = 5$, peut-on obtenir des trajectoires décroisées, des trajectoires croisées ?

(e*-TP) Lire et comprendre la fonction de génération de courbes convexes ci-dessous. L'utiliser pour obtenir des trajectoires de billard parfaites sur un convexe généré de la sorte (on obtient des convexes moins symétriques).

```
In [ ]: def genereGamma(checkConvexe=True,Npoints=100):
            a=randn(5);b=randn(5);c=randn(5);d=randn(5)
            a[1]+=5;d[1]+=5

            def courbe(t):
                x=sum((a[i]*cos(i*t) + b[i]*sin(i*t))/i**2 for i in range(1,5))
                y=sum((c[i]*cos(i*t) + d[i]*sin(i*t))/i**2 for i in range(1,5))
                return array([x,y])

            if checkConvexe:
                t=linspace(0,2*pi,Npoints)
                xp=sum((-a[i]*sin(i*t) + b[i]*cos(i*t))/i for i in range(1,5))
                xs=sum((-a[i]*cos(i*t) - b[i]*sin(i*t)) for i in range(1,5))
                yp=sum((-c[i]*sin(i*t) + d[i]*cos(i*t))/i for i in range(1,5))
                ys=sum((-c[i]*cos(i*t) - d[i]*sin(i*t)) for i in range(1,5))
                if any(xp*ys-yp*xs<0):
                    return genereGamma(True,Npoints)
            return courbe
```

4. Descente de gradient à pas optimal

(a-TP) Programmer une fonction de recherche de pas optimal à l'aide de la méthode de la section dorée. Pour éviter d'évaluer les fonctions là où elles sont déjà calculées, on donnera comme arguments la fonction f, le point x_k, la valeur de $f(x_k)$, la direction de descente d et un pas par défaut α (ainsi qu'une tolérance, ou un nombre d'itérations maximal).

On renverra le point x_{k+1}, éventuellement le pas α_k calculé, ainsi que la valeur de $f(x_{k+1})$ si elle a déjà été calculée.

(b-TP) Programmer une fonction qui effectue la méthode de descente de gradient à pas optimal à l'aide de la fonction de recherche précédente.

L'appliquer sur la fonction f_a de la partie 1. Le taux de convergence linéaire est-il amélioré ? Même question si on considère cette fois-ci le taux de convergence effectif (en effet, la méthode de recherche du pas optimal consomme des évaluations de fonctions). On pourra redéfinir f_a en rajoutant une variable globale de compteur dans sa définition.

Observer comment se comporte le taux de convergence effectif si on change la tolérance dans la fonction de recherche de pas optimal (par exemple, si on augmente la tolérance, la recherche du pas optimal sera moins bonne, mais on fera moins d'évaluations de fonctions).

(c-TP) Appliquer la méthode de descente de gradient à pas optimal pour obtenir des trajectoires de billards parfaites comme dans la partie 3. Observer également si cette méthode est efficace.

(d-TD) On considère la fonction auxiliaire $h(t) = f(x_k + td)$. On suppose qu'on connait déjà $h(0)$, et $h'(0) < 0$. Plutôt que de rechercher un pas optimal, on souhaiterait trouver un pas qui approximerait bien le pas optimal lorsque h est bien approximée par une fonction quadratique, en effectuant le moins d'évaluations de fonctions possible.

On se donne un premier pas $\alpha > 0$ et on évalue $h(\alpha)$. On interpole h par le polynôme p de degré 2 tel que $p(0) = h(0)$, $p'(0) = h'(0)$ et $p(\alpha) = h(\alpha)$.

Montrer que le point α_* tel que $p'(\alpha_*) = 0$ est donné par

$$\alpha_* = \frac{1}{2} \frac{\alpha}{1 - \frac{h(\alpha) - h(0)}{\alpha h'(0)}}.$$

(e-TP) Programmer une méthode de descente qui utilise la formule précédente pour calculer le pas. Attention à gérer les cas pathologiques dans le cas où $\alpha_* < 0$ ou α_* trop grand. On pourra par exemple augmenter le pas initial α et recalculer un nouvel α_*, ou tout simplement rester sur un pas fixe lorsque cette méthode ne fonctionne pas.

Appliquer cette méthode de descente sur un des problèmes de ce TP. Le taux de convergence effectif est-il amélioré ?