TP1
Algorithme du Gradient

A. Algorithme du Gradient pour une fonction à 1 variable

On s'intéresse aux fonctions suivantes :

$$f_1(x) = x^2 - 4x + 4, \quad \text{et} \quad f_2(x) = x^3 - 6x.$$

Le but de cette partie est d'utiliser la méthode du gradient pour approcher les minima de ces fonctions.

1. **Sur papier.** Calculer les dérivées de f_1 et f_2.
   En Python. Créer les fonctions Python `f1prime(x)` et `f2prime(x)` correspondant à ces dérivées.

2. Ecrire une fonction Python `Gradient1D(x0,alpha,n,fp)`, qui renvoie l'itération x_n de l'algorithme du gradient, et qui prend en paramètres :
   - l'initialisation `x0` de l'algorithme,
   - le pas `alpha` de l'algorithme,
   - le nombre `n` d'itérations souhaitées,
   - la fonction Python `fp` qui contiendra la dérivée de la fonction à laquelle on souhaite appliquer l'algorithme.

   Remarque : oui, on peut passer une fonction en paramètre en Python ! On peut ensuite l'utiliser à l'intérieur de la fonction Gradient1D, en faisant par exemple `fp(x0)`.

3. Tester votre fonction `Gradient1D` avec la fonction `f1prime` et plusieurs choix d'initialisations et de pas. Vérifier que vous obtenez bien les mêmes résultats que dans l'Exemple fait en cours page 14 du Chapitre 1.

4. Tester votre fonction `Gradient1D` avec la fonction `f2prime` et plusieurs choix d'initialisations et de pas.
   Quel semble être le minimum de f_2 ?

B. Algorithme du gradient pour une fonction à 2 variables

On s'intéresse aux fonctions suivantes :

$$f_1(x, y) = 2x^2 + 2x + 5y^2 + xy, \quad \text{et} \quad f_2(x, y) = x^2 + y^2 - 1.9x + 1.$$

Le but de cette partie est d'utiliser la méthode du gradient pour approcher les minima de ces 2 fonctions.

5. **Sur papier.** Calculer les dérivées partielles de f_1 et de f_2 par rapport à x et par rapport à y.
   En Python. Définir ces dérivées partielles comme des fonctions Python `df1x(x,y)`, `df1y(x,y)`, `df2x(x,y)` et `df2y(x,y)`.
6. Ecrire une fonction Python `Gradient2D(fx,fy,x0,y0,alpha,n)`, qui prend en paramètres :
   - les fonctions Python `fx` et `fy` correspondant aux 2 dérivées partielles de la fonction à laquelle on veut appliquer l'algorithme du Gradient,
   - l'initialisation `x0,y0` de l'algorithme,
   - le pas `alpha` de l'algorithme,
   - le nombre `n` d'itérations souhaitées,

   et renvoie l'itération (x_n, y_n) de l'algorithme du gradient.
7. Tester votre fonction `Gradient2D` avec `df1x` et `df1y` et plusieurs choix d'initialisations, de pas et de nombres d'itérations. Vérifier que vous obtenez bien les mêmes résultats que dans l'Exemple fait en cours pages 17 et 18 du Chapitre 1.
8. Tester votre fonction `Gradient2D` avec `df2x` et `df2y` et plusieurs choix d'initialisations, de pas et de nombres d'itérations.
   Quel semble être le minimum de $f2$?

C. Algorithme du gradient pour la régression linéaire

Dans cette partie, on reprend l'Exemple des âges et tailles des enfants de 2 à 12 ans qui est énoncé dans le cours page 20 du Chapitre 1.

9. Commencer par rentrer les dérivées partielles trouvées dans le cours, dans des fonctions Python `dFa(a,b)` et `dFb(a,b)`.

10. Utiliser votre fonction `AlgoGradient2D` pour calculer les itérations de l'algorithme du gradient avec les conditions initiales et le pas choisis dans le cours. Pour les 2 première itérations, trouvez-vous les mêmes résultats qu'en cours ?
11. En continuant d'itérer, se rapproche-t-on des solutions exactes (16.8796, trouvées avec les formules de statistiques ? Tester en changeant le paramètre α, le nombre d'itérations, les conditions initiales...

D. Calcul matriciel de la fonction F de la régression

Dans l'exemple du cours sur les âges et les tailles des enfants, on avait l'expression de la fonction $F(a,b)$ à minimiser :

$$\begin{aligned} F(a,b) \ =&(2-(0.84a+b))^2+(4-(0.98a+b))^2+(8-(1.14a+b))^2 \\ &+(10-(1.32a+b))^2+(12-(1.44a+b))^2 \end{aligned}$$

La valeur de $F(a,b)$ vous avait été ensuite donnée pour vous épargner les calculs (et de même dans les exercice et exemple suivants du chapitre) :

$$F(a,b) = 6.7816\ a^2 + 5\ b^2 + 11.44\ ab - 90.4\ a - 72\ b + 328$$

Heureusement, il est en fait possible de calculer $F(a,b)$ et ses dérivées partielles grâce à une écriture matricielle. C'est l'objectif de cette partie.

12. Rentrer les données de taille des enfants dans un `np.array` X **en colonne**. De même pour les âges des enfants, dans une colonne Y.
13. A l'aide de X et Y, écrire une fonction `Erreurs(a,b)` qui renvoie un `np.array` en colonne contenant les erreurs commises avec la droite $y = ax + b$ pour chacun des enfants : âge réel $-$ âge estimé par la droite.
    Tester votre fonction sur des valeurs de a et b simples pour pouvoir vérifier (par exemple, $a = 0$ ou $a = 1$, et $b = 0$ ou $b = 1$.
14. **Sur papier.** Soit C une matrice colonne. On rappelle que la transposée de C, notée C^T est alors la matrice ligne contenant les mêmes coefficients :

$$C = \begin{pmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{pmatrix}, \qquad C^T = \begin{pmatrix} c_1 & c_2 & \dots & c_n \end{pmatrix}$$

Pour une telle matrice C, que vaut alors le produit matriciel $C^T C$?

15. **Sur papier.** La fonction `F(a,b)` va pouvoir s'obtenir à l'aide du tableau `E` renvoyé par la fonction `Erreurs` et de la transposée de ce tableau `E`. En s'aidant de la question précédente, trouver l'expression de $F(a,b)$ en fonction de E et E^T.
16. Coder la fonction $F(a,b)$ en Python à partir de la fonction `Erreurs` et de la formule trouvée dans la question précédente.
On pourra utiliser les fonctions `np.transpose` *et* `np.dot`
17. Tester $F(a,b)$ sur quelques valeurs pour vérifier que c'est bien la même fonction que la F donnée ci-dessus : essayer par exemple avec $a = 0, b = 0$, puis avec $a = 1, b = 0$ ou encore $a = 0, b = 1$.
18. On aura surtout besoin des dérivées partielles de F pour coder l'algorithme du gradient. Elles s'obtiennent avec les produits matriciels suivants :
$$\frac{\partial F}{\partial a}(a,b) = -2X^T E,$$
$$\frac{\partial F}{\partial b}(a,b) = -2U^T E,$$
où la matrice U est une matrice colonne de la même taille que X et Y, ne contenant que des 1.
Coder les 2 dérivées partielles `dFa1` et `dFb1` à l'aide de ces formules.
On pourra utiliser la fonction `np.ones`.

E. Quelques représentations graphiques

Dans les prochaines questions on utilisera les librairies Python suivantes :

```
import numpy as np
import matplotlib.pyplot as plt
```

19. Tester la commande `np.linspace(0,5,21)` puis, changer les paramètres pour comprendre le fonctionnement de cette fonction. Faire ensuite de même avec

`X,Y=np.meshgrid(np.linspace(0,5,21),np.linspace(10,15,21))`

Utiliser ces fonctions pour créer les coordonnées `X,Y` d'un maillage en 2 dimensions avec x dans l'intervalle $[-2,3]$ et y dans $[-2,2]$.

20. Définir en Python la fonction `f2(x,y)` de la partie B, puis le tableau

```
Z=f(X,Y)
```

21. Pour créer une représentation en 3D dans Python, la commande est la suivante :

```
ax = plt.axes(projection ='3d')
```

Puis pour tracer le graphe de la fonction f dans cet espace en 3D :

```
ax.plot_surface(X,Y,Z)
```

22. Créer une nouvelle figure avec `fig1=plt.figure()`, puis faire le tracé suivant :

```
plt.contour(X,Y,Z,20)
```

A quoi correspond ce tracé ?