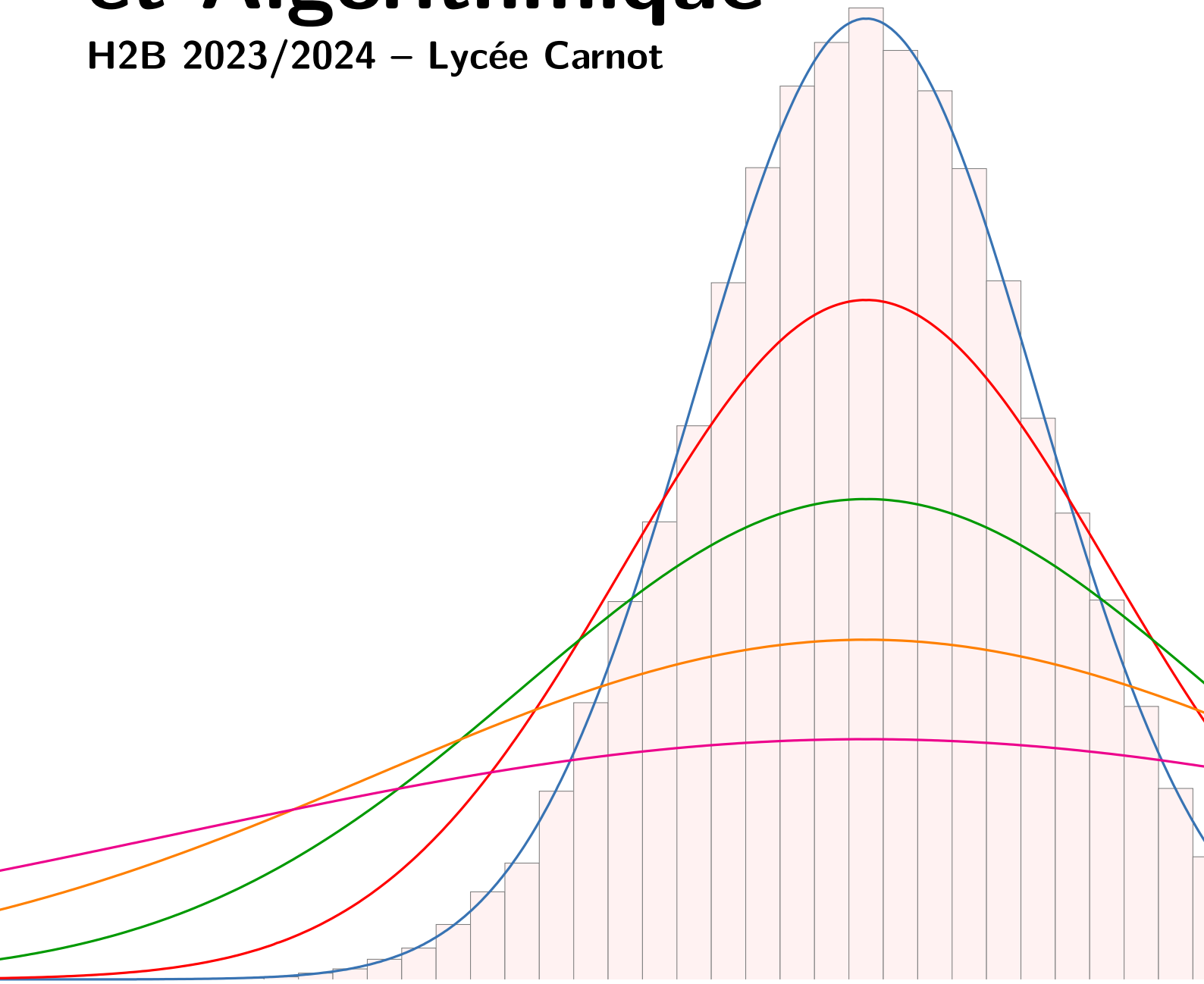
TP d'Informatique et Algorithmique

H2B 2023/2024 – Lycée Carnot



Matthias Gorny

www.matthiasgorny.fr

Table des matières

Introduction

I Ouvrir votre compte utilisateur

Lors de votre inscription au lycée Carnot, un compte informatique à votre nom a été créé. Votre login est composé de la première lettre de votre prénom, suivi de votre nom (en minuscule et sans espace). Votre mot de passe par défaut est votre date de naissance sous la forme : *jjmmaaaa*.

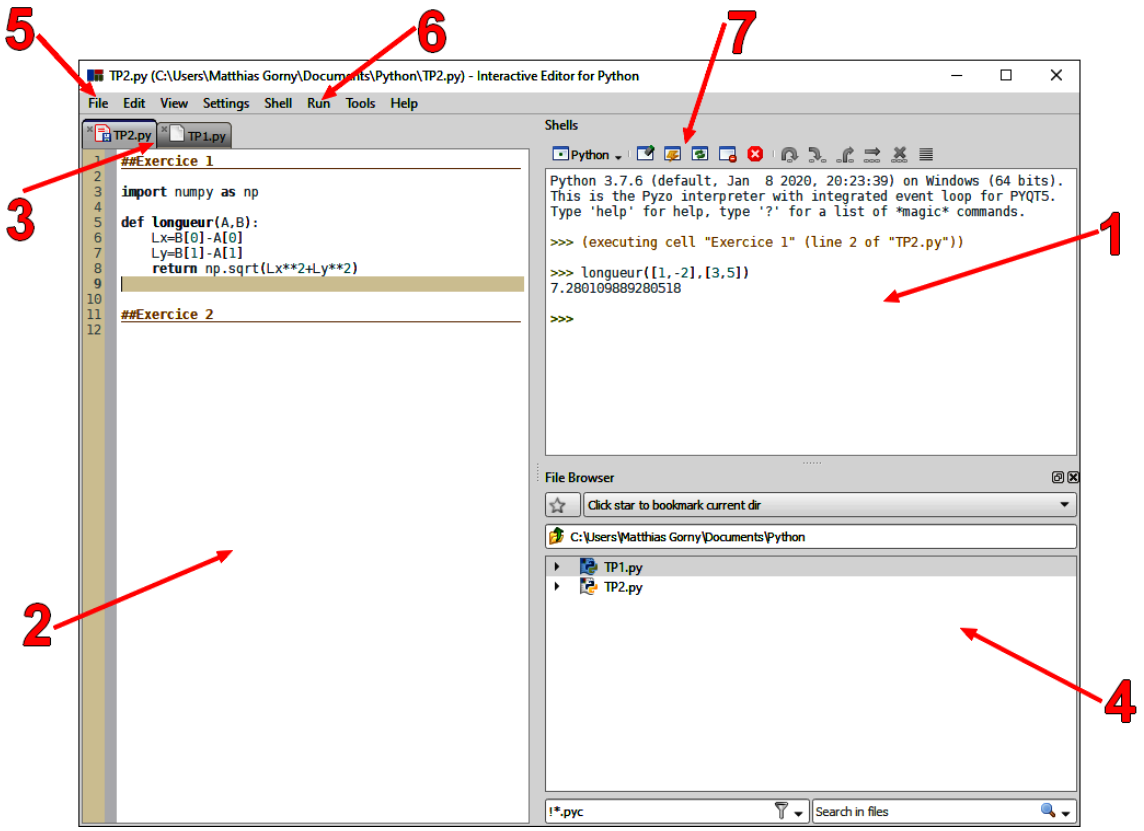
Une fois connecté sur votre compte, créez un dossier intitulé `H2B_TPInfo`. A chaque TP, on créera un nouveau fichier au nom adéquat.

Démarrez le logiciel **Pyzo** ou **Spyder**.

II Environnement de travail

Le langage de programmation officiel du programme de Mathématiques Approfondies en ECG est **Python**. Une fois Python installé, on peut utiliser un environnement de travail (IDE pour Integrated Development Environment) plus « convivial » que celui installé automatiquement. Il en existe plusieurs et, cette année, nous utiliserons **Pyzo**[1] ou **Spyder**. Le fonctionnement et l'interface de ces deux environnements sont essentiellement les mêmes.

Lorsqu'on lance Pyzo ou Spyder, une fenêtre apparaît composée de plusieurs sous-fenêtres dont la *console* et l'*éditeur*.



1. Car c'est celui que j'utilise, « moi personnellement » (mais rien ne vous empêche d'en utiliser un autre chez vous). Pour installer Pyzo sur votre ordinateur personnel, rendez-vous à cette page : `https://pyzo.org/start.html`
Sur mon site, vous trouverez un guide d'installation.

1. La **console** (ou *shell*) est la zone où l'on tape les commandes. Le symbole >>> signifie que l'ordinateur attend une instruction. On peut alors saisir une commande et appuyer sur la touche [Entrée] pour que cette commande soit interprétée.
   - On peut écrire plusieurs instructions sur une même ligne en les séparant d'un point virgule.
   - L'usage des touches fléchées [↑] et [↓] fait apparaitre les commandes précédemment tapées.
2. L'**éditeur** est la zone dans laquelle nous allons principalement travailler. En effet, la console est vite limitée lorsqu'on veut écrire des programmes faisant plus d'une ligne ou lorsqu'on doit en utiliser plusieurs en même temps. On crée alors des fichiers (au format .py) dans lesquels on écrit les différentes commandes.
   - On peut séparer un fichier en différentes parties appelées *cellules*, en utilisant la commande `#%%`. On peut écrire un titre et un trait est tracé automatiquement. C'est particulièrement utile pour séparer les exercices.
   - Il est important de penser à enregistrer régulièrement ses fichiers, en utilisant le raccourci [Ctrl] + [S] .
3. On peut ouvrir plusieurs fichiers .py en même temps qui s'organisent en onglets. Pour accéder à un fichier déjà ouvert, il suffit de cliquer sur son onglet.
4. Le **navigateur de fichier** est une zone où l'on peut accéder aux différents fichiers Python enregistrés dans l'ordinateur. On peut notamment les ouvrir en double-cliquant dessus. En cliquant sur `Tools` en haut de l'écran, on peut ouvrir d'autres fenêtres comme l'historique des commandes utilisées par exemple.
5. Le menu **File** regroupe des fonctions utiles : création d'un nouveau fichier (*New*), ouverture d'un fichier (*Open...*), sauvegarde (*Save*), etc.
6. Une fois un programme terminé, il faut l'exécuter dans la console. Il y a différentes façon d'exécuter qui sont regroupées dans le menu **Run** :
   - Cliquer sur *Execute selection* exécute les lignes de codes sélectionnées.
   - Cliquer sur *Execute cell* exécute tout le contenu de la cellule sur laquelle on vient de cliquer. Un raccourci est [Ctrl] + [Entrée] .
   - Cliquer sur *Execute file* exécute tout le contenu du fichier. Un raccourci est [Ctrl] + [E] .
7. Les icônes en haut de la console permettent d'interrompre un code en cours d'exécution (pratique pour les boucles `while` qui s'avèrent infinies) ou de redémarrer la console en cas de bug.

Feuille de TP n° 0

Révisions de première année

I Commandes et structures de bases

Exercice 1 – Renversement d'une liste. (★) Écrire une fonction qui prend en argument une liste et qui renvoie la liste dans laquelle les éléments sont listés dans l'autre sens (du dernier de la liste d'entrée au premier de la liste d'entrée).

Exercice 2 – Somme, produit, sommes cumulées, moyenne, variance, écart type. (★)

1) Écrire une fonction qui prend en argument une liste de réels et renvoie la somme des éléments (sans utiliser `np.sum`).
2) Écrire une fonction qui prend en argument une liste de réels et renvoie le produit des éléments (sans utiliser `np.prod`).
3) Écrire une fonction qui prend en argument une liste de réels et renvoie la liste des sommes cumulées des éléments (sans utiliser `np.cumsum`).
4) Écrire une fonction qui prend en argument une liste de réels et renvoie la moyenne des éléments (sans utiliser `np.mean`).
5) Écrire une fonction qui prend en argument une liste de réels et renvoie la variance des éléments (sans utiliser `np.var`).
6) Écrire une fonction qui prend en argument une liste de réels et renvoie l'écart type des éléments (sans utiliser `np.std`).

Exercice 3 – Recherche du nombre d'occurrences. (★) Écrire une fonction qui prend en argument une liste et une variable et qui renvoie le nombre d'occurrences de la variable dans la liste.

Exercice 4 – Recherche du max et du min. (★★)

1) Écrire une fonction qui prend en argument une liste de réels et qui renvoie le maximum de la liste (sans utiliser `np.max`), ainsi que le plus petit indice où se trouve le maximum.

   Indications : le premier élément de la liste est le maximum provisoire. Ensuite on parcourt la liste de gauche à droite et, dès qu'on trouve un élément strictement plus grand que le maximum provisoire, celui-ci devient le nouveau maximum provisoire.
2) Écrire une fonction qui prend en argument une liste de réels et qui renvoie le minimum de la liste (sans utiliser `np.min`), ainsi que le plus petit indice où se trouve le minimum.
3) Tester ces fonctions avec la liste `L=[rd.binomial(15,0.3) for k in range(20)]`.
4) Modifier ces fonctions pour qu'elles renvoient la liste de tous les indices où se trouvent le maximum et le minimum respectivement. Les tester avec la liste de la question précédente.

II Suites, sommes, produits

Exercice 5 – Classique : factorielle. (★) Écrire une fonction qui prend en argument un entier naturel et renvoie sa factorielle.

Exercice 6 – Classique : coefficients binomiaux. (★) Écrire une fonction qui prend en argument deux entiers naturels n et p et qui renvoie $\binom{n}{p}$ (avec la convention que $\binom{n}{p} = 0$ si $p > n$).
Cette fonction ne devra pas utiliser de fonction factorielle.

Exercice 7. (★) On considère la suite $(u_n)_{n\in\mathbb{N}}$ définie par $u_0 = 1$ et

$$\forall n \in \mathbb{N}, \qquad u_{n+1} = \frac{(n-2)u_n + 2}{u_n + n}.$$

1) Construire une fonction en Python qui prend en argument un entier naturel N et qui renvoie u_N.
2) Construire une fonction en Python qui prend en argument un entier naturel N et qui représente graphiquement $u_0, u_1, \ldots, u_N$. La tester pour $N = 50$. Que peut-on conjecturer ? Que peut-on superposer à la courbe pour améliorer la conjecture ?

Exercice 8. (★★) Soient a et b deux réels. On considère la suite $(u_n)_{n\in\mathbb{N}}$ définie par $u_0 = a$, $u_1 = b$ et

$$\forall n \in \mathbb{N}, \qquad u_{n+2} = \frac{u_{n+1}}{2} + e^{-u_n}.$$

1) Recopier et compléter la fonction Python ci-dessous pour qu'elle prenne en argument les réels a et n et un entier naturel N et renvoie u_N.

```
1 def Suite(a,b,N):
2     u=.............
3     v=.............
4     for n in range(N):
5         aux=.............
6         u=v
7         v=..............
8     return  .............
```

2) Comment peut-on remplacer les lignes 5,6,7 par une seule ligne ?
3) Construire une fonction en Python qui prend en argument les réels a et n et un entier naturel N et qui représente graphiquement $u_0, u_1, \ldots, u_N$. La tester pour $N = 50$ et pour plusieurs valeurs de a et b. Que peut-on conjecturer ?

Exercice 9 – Nombres de Bell. On pose $B_0 = 1$ et

$$\forall n \in \mathbb{N}, \qquad B_{n+1} = \sum_{k=0}^{n} \binom{n}{k} B_k.$$

1) (★★) Recopier et compléter le programme ci-dessous pour qu'il prenne en entrée un entier naturel N non nul et pour qu'il renvoie la liste des entiers $B_0, \ldots, B_N$.

```
1  import ........................
2  def Bell(N):
3      B=np.zeros(N+1)
4      B[0]=1
5      Facto=[]
6      for n in range(N):
7          #Utilisation de la formule de Pascal pour former la
8          #liste des (p parmi n) pour tout p de 0 à n
9          Temp=Facto+[1]
10         for p in range(n-1):
11             Temp[p+1]=.............................
12         Facto=Temp
13         #Calcul de B_(n+1)
14          ..............................
15          ..............................
16          ..............................
17         B[n+1]=S
18     return B
```

2) (★★★) On appelle partition d'un ensemble E, toute famille de parties non vides deux à deux disjoints dont la réunion est E tout entier. Montrer que, pour tout $n \in \mathbb{N}^*$, B_n est le nombre de partitions de $[\![1\,;n]\!]$.
On raisonnera par récurrence. Pour tout $n \in \mathbb{N}$ et $k \in [\![0\,;n]\!]$, on pourra introduire E_k l'ensemble des partitions de $[\![1\,;n+1]\!]$ pour lesquelles la partie contenant $n+1$ est de cardinal $k+1$.

III Analyse numérique

1) Dichotomie

Exercice 10 – D'après EDHEC 2016. (★★) Soit $f : x \in \mathbb{R}_+^* \longmapsto \dfrac{e^{-x}}{x}$.

1) Étudier les variations de la fonction $g : x \longmapsto e^{-x} - x^2$ sur $\mathbb{R}_+$.
2) Montrer que l'équation $f(x) = x$, d'inconnue $x \in \mathbb{R}_+^*$, admet une unique solution que l'on notera α.
3) Montrer que $e^{-1} < \alpha < 1$.
4) Compléter le script Python suivant qui affiche une valeur approchée de α à 10^{-3} près :

```
import ..................
def g(x):
    return ..............
a=...........
b=...........
while ..................
    c=(a+b)/2
    if g(c)>0:
        ...........
    else:
        ...........
print('Une valeur approchée de alpha est '+str(......))
```

Exercice 11 – D'après HEC 2020. (★★) Soit $f : t \in]0\,;1[\longmapsto \dfrac{t}{1-t}e^{-1/t}$.

1) Montrer que f est prolongeable en une fonction continue sur $[0\,;1[$.
2) Montrer que f, ainsi prolongée, est bijective de $[0\,;1[$ sur un intervalle I à préciser.
3) On cherche à représenter graphiquement f^{-1} sur I à l'aide de Python.
   a) Première méthode : avec un algorithme de dichotomie. Recopier et compléter le script Python suivant pour qu'il affiche une représentation graphique de f^{-1} sur $[0\,;f(0,95)]$.

```
import ..................
import ..................
#Implémentation de f
def f(t):
    if .........
        return .........
    else:
        return .........

#Implémentation de la réciproque de f
def Inv_f(x):
    a=0
    b=0.95
    while ..............
        c=(a+b)/2
        if f(c)>x:
            ..............
        else:
            ..............
    return ..............

#Tracé de la fonction
X=..........................
Y=..........................
plt.plot(X,Y)
plt.show()
```

   b) Deuxième méthode : en se rappelant que la courbe de f^{-1} s'obtient facilement via une transformation géométrique de la courbe de f.
4) Conjecturer le domaine sur lequel l'application f^{-1} est dérivable. Prouver cette conjecture.

2) Méthode des rectangles

Soient $n \in \mathbb{N}^*$, $a \in \mathbb{R}$ et $b \in \mathbb{R}$. On rappelle que :

- Approximation dans le cas C^1 (non exigible mais on l'a démontré l'an passé) : si f est une fonction de classe C^1 sur un segment $[a\,;b]$ et si $S_n(f)$ désigne une somme de Riemann à n pas constant associée à f sur $[a\,;b]$, alors
$$\left|S_n(f) - \int_a^b f(t)\,dt\right| \leqslant \frac{(b-a)^2}{2n} \max_{[a\,;b]} |f'|.$$
- Approximation dans le cas monotone (non exigible mais on le retrouve facilement par comparaison série/intégrale, cf. cours de Python de première année) : si f est une fonction monotone sur $[a\,;b]$, alors
$$\left|\frac{b-a}{n}\sum_{k=1}^{n} f\left(a + k\frac{b-a}{n}\right) - \int_a^b f(t)\,dt\right| \leqslant \frac{\big|(b-a)\big(f(b)-f(a)\big)\big|}{n}.$$
La somme ci-dessus est la somme de Riemann à droite associée à f sur $[a\,;b]$. C'est encore vrai si on la remplace par la somme de Riemann à gauche.

Exercice 12 – « Vraie » vitesse de convergence. Soit $f : x \longmapsto \dfrac{4}{1+x^2}$. Pour tout $n \in \mathbb{N}^*$, notons $S_n(f)$ la somme de Riemann à droite associée à f sur $[a\,;b]$.

1) Calculer $I = \displaystyle\int_0^1 f(x)\,dx$.
2) Écrire une fonction qui prend en argument un entier naturel non nul n et qui renvoie $S_n(f)$.
3) Soit $\varepsilon > 0$. Déterminer $n \in \mathbb{N}$ tel que $S_n(f)$ soit une approximation de I à ε près.
On utilisera le fait que f est C^1 puis le fait que f est monotone. Laquelle donne le plus petit n ?
4) Écrire une fonction qui prend en argument ε et qui renvoie le plus petit n tel que $S_n(f)$ soit une approximation de I à ε près (pourquoi le programme s'arrête-t-il au fait ?). Comparer avec les valeurs trouvées à la question précédente.
On utilisera une approximation de I fournie par l'ordinateur (on sait alors que, si n est une des valeurs obtenues à la question précédente, $S_n(f)$ est une approximation de I à ε près et le but de cette question est de déterminer si l'approximation était déjà bonne pour un plus petit n).
5) a) Justifier que
$$u_n(f) = 2S_{2n}(f) - S_n(f) \xrightarrow[n\to+\infty]{} I \qquad \text{et} \qquad v_n(f) = \frac{8S_{4n}(f) - 6S_{2n}(f) + S_n(f)}{3} \xrightarrow[n\to+\infty]{} I.$$
b) Écrire un programme qui prend en argument ε et qui renvoie le plus petit n tel que $u_n(f)$ soit une approximation de I à ε près. Comparer avec les valeurs trouvées précédemment.
c) Écrire un programme qui prend en argument ε et qui renvoie le plus petit n tel que $v_n(f)$ soit une approximation de I à ε près. Comparer avec les valeurs trouvées. précédemment.

Exercice 13. (★★★) Le but de cet exercice est de démontrer que, en gardant les notations de l'exercice précédent, les convergences de $(u_n(f))_{n\geqslant 1}$ et $(v_n(f))_{n\geqslant 1}$ vers I sont en effet bien plus rapides que celle de $(S_n(f))_{n\geqslant 1}$. On fait pour cela l'hypothèse que f est une fonction quelconque de classe C^4 sur un segment $[a\,;b]$ avec $a < b$. Pour tout $k \in [\![1\,;4]\!]$, on note $M_k = \sup\limits_{[a\,;b]} \left|f^{(k)}\right|$. On se donne $n \in \mathbb{N}^*$ et, pour tout $i \in [\![0\,;n]\!]$, on pose $x_i = a + i\dfrac{b-a}{n}$.

1) a) Pour tout $i \in [\![0\,;n]\!]$, justifier que $|f(x_{i+1}) - f(x_i)| \leqslant M_1 \dfrac{b-a}{n}$.
b) En déduire que $|I - S_n(f)| \leqslant \dfrac{(b-a)^2}{2n} M_1$.
2) Soit $[\alpha\,;\beta]$ un segment inclus dans $[a\,;b]$. On introduit :
$$p : x \longmapsto \int_\alpha^x f(t)\,dt - (x-\alpha)f(\alpha) \qquad \text{et} \qquad q : x \longmapsto p(x) - \frac{(x-\alpha)(f(x)-f(\alpha)}{2}.$$

a) Calculer les deux premières dérivées de p et q.

b) A l'aide d'inégalité de Taylor-Lagrange, montrer que

$$\left|\int_\alpha^\beta f(t)\,dt - (\beta-\alpha)f(\alpha) - \frac{(\beta-\alpha)(f(\beta)-f(\alpha)}{2}\right| \leqslant \frac{(\beta-\alpha)^3}{4}\,M_2.$$

c) En déduire que

$$\left|I - S_n(f) - \frac{(b-a)(f(b)-f(a))}{2n}\right| \leqslant \frac{(b-a)^3}{4n^2}M_2.$$

3) Soit $[\alpha\,;\beta]$ un segment inclus dans $[a\,;b]$. On introduit de plus :

$$r : x \longmapsto q(x) + \frac{(x-\alpha)^2(f'(x)-f'(\alpha))}{12}.$$

En procédant de la même manière qu'à la question précédente, établir que

$$\left|I - S_n(f) - \frac{(b-a)(f(b)-f(a))}{2n} + \frac{(b-a)^2(f'(b)-f'(a))}{12n^2}\right| \leqslant \frac{(b-a)^5}{72n^4}M_4.$$

4) A l'aide des résultats précédents, déterminer le développement asymptotique en $\dfrac{1}{n}$ à l'ordre 3 de $S_n(f)$ lorsque n tend vers $+\infty$. Cela consiste à trouver des réels A, B, C et D tels que

$$S_n(f) \underset{+\infty}{=} A + \frac{B}{n} + \frac{C}{n^2} + \frac{D}{n^3} + \mathrm{o}\left(\frac{1}{n^3}\right).$$

5) En déduire les développements asymptotiques en $\dfrac{1}{n}$ à l'ordre 3 de $u_n(f)$ et $v_n(f)$ lorsque n tend vers $+\infty$. Conclure.

Exercice 14 – Implémentation de Φ. (★★) Notons $\varphi : t \longmapsto \dfrac{e^{-t^2/2}}{\sqrt{2\pi}}$ la densité d'une variable aléatoire de loi $\mathcal{N}(0,1)$ et Φ sa fonction de répartition.

1) Justifier que φ est de classe C^1 sur $\mathbb{R}$ et que φ' est bornée par $\dfrac{1}{\sqrt{2\pi e}}$ sur $\mathbb{R}$.

2) Montrer que, pour tout $x \in \mathbb{R}$, $\Phi(x) = \dfrac{1}{2} + \displaystyle\int_0^x \varphi(t)\,dt$.

3) Écrire une fonction, appelée `Phi`, qui prend en argument un réel x et qui renvoie une valeur approchée de $\Phi(x)$ à 10^{-3} près à l'aide de la méthode des rectangles.

IV Matrices

Exercice 15. (★) Écrire une fonction en Python qui prend en argument une matrice A, un vecteur colonne B ayant autant de ligne que A a de colonnes, deux entiers naturels i, j (indices possibles pour les lignes de A) et deux réels α et β et qui fait l'opération élémentaire $L_i \leftarrow \alpha L_i + \beta L_j$ sur les matrices A et B.
Cette fonction ne renverra rien si $\alpha \neq 0$ mais renverra le message « le pivot est nul » si α est nul.

Exercice 16 – Bon courage sans Python. (★★) Résoudre le système suivant, d'inconnues x, y, z réelles, à l'aide de la méthode du pivot de Gauss implémentée avec Python.

$$\begin{cases} -45x - 11y + 22z = -1 \\ -32x - 16y + 20z = -4 \\ -13x + 24y - 10z = 5 \end{cases}$$

On utilisera l'exercice précédent et on vérifiera ensuite avec la commande `al.solve`.

Exercice 17 – Codage matriciel d'un endomorphisme défini à l'aide de polynômes. (★★★)

1) Soit $n \in \mathbb{N}$. En Python, on représentera un polynôme P de degré inférieur ou égal à n par le vecteur ligne (de type `numpy` et non une liste) égal aux coordonnées de P dans la base canonique de $\mathbb{R}_n[X]$.

   Par exemple :

   — *Si* $n = 3$, `P=np.array([0,1,2,-1])` *implémente le polynôme* $P = X + 2X^2 - X^3$.

   — *Si* $n = 4$, `P=np.array([1,0,0,-1,0])` *implémente le polynôme* $P = 1 - X^3$.

   — *Si* $n = 5$, *alors on implémente* $P = 7 - X + \sqrt{2}X^3 - 9X^4$ *par* `P=np.array([7,-1,0,2**(1/2),-9,0])`.

   Ainsi si P et Q sont deux polynômes de $\mathbb{R}_n[X]$ implémentés par `P` et `Q` respectivement et si a est un réel implémenté par `a`, alors `a*P+Q` implémente $aP + Q$. C'est moins immédiat pour les autres opérations (ne serait-ce que par défaut de stabilité) et c'est que nous allons voir.

   a) Recopier et compléter la fonction ci-dessous afin qu'elle prenne en argument un polynôme implémenté en liste ou vecteur ligne et renvoie sa dérivée sous forme de vecteur ligne.

```
import ......................
def Deriv(P):
    n=..........
    Q=np.zeros(n)
    for i in range(n-1):
        Q[i]=.............
    return Q
```

   b) Recopier et compléter la fonction ci-dessous afin qu'elle prenne en argument un entier naturel non nul k et un polynôme P implémenté en vecteur ligne et renvoie X^kP implémenté en Python (par un vecteur ayant k coordonnée de plus, n'est-ce pas ?).

```
def MultX(P,k):
    n=.............
    Q=np.zeros(n+k)#On travaille dans R_{n+k}[X]
    for i in range(....,....):
        Q[i]=.............
    return Q
```

   c) Écrire une fonction en Python qui prend en argument deux entiers naturels n et k avec $k \leqslant n$ et qui implémente le polynôme X^k de $\mathbb{R}_n[X]$ en vecteur ligne.

2) Soit $n \in \mathbb{N}^*$. On considère l'application $f_n : P \in \mathbb{R}_n[X] \longmapsto nXP + (1 - X^2)P'$.

   a) Montrer que f_n est un endomorphisme de $\mathbb{R}_n[X]$.

   b) A l'aide des fonctions Python précédentes, construire une fonction en Python qui prend en argument un entier naturel non nul n et un polynôme P de $\mathbb{R}_n[X]$ implémenté en vecteur ligne et renvoie $f_n(P)$. *Attention avant de sommer les trois polynômes de* $f_n(P)$ *car les fonctions définies précédemment vont fournir des polynômes de* $\mathbb{R}_{n+1}[X]$, $\mathbb{R}_n[X]$ *et* $\mathbb{R}_{n+2}[X]$ *respectivement (il faudra donc enlever les coordonnées superflues).*

   c) Recopier et compléter la fonction ci-dessous afin qu'elle prenne en argument un entier naturel non nul n et qu'elle renvoie la matrice de f_n dans la base canonique de $\mathbb{R}_n[X]$.

```
def Mat_f(n):
    A=.....................#Matrice carrée nulle d'ordre n+1
    #On va construire la transposée de la matrice
    for k in range(n+1):
        A[k,:]=.............
    return ................#La transposée de A
```

   d) Pour $n \in [\![1\,;10]\!]$, déterminer les valeurs propres et vecteurs propres de f_n à l'aide de Python et de la commande `al.eig`. Que peut-on conjecturer quant à la diagonalisabilité de f ?

   e) Montrer que, pour tous $n \in \mathbb{N}$ et $k \in [\![0\,;n]\!]$, $(X-1)^k(X+1)^{n-k}$ est un vecteur propre de f_n. Prouver alors la conjecture de la question précédente.

V Probabilités discrètes

Exercice 18. Une urne contient 1 boule jaune, 2 boules rouges et 4 boules bleues.

1) On effectue le tirage d'une boule dans l'urne. Écrire un programme en Python qui simule cette expérience et qui affiche à l'écran la couleur de la boule tirée.
   Pour simplifier la simulation, on pourra supposer que la boule jaune est numérotée 1, que les boules rouges sont numérotées 2 et 3 et que les boules bleues sont numérotées 4,5,6 et 7.

2) Soit n un entier naturel non nul. On effectue n tirages d'une boule dans l'urne, avec remise de la boule tirée avant chaque tirage. On note X la variable aléatoire égale au nombre de boules rouges obtenues au cours de cette expérience.

   a) Écrire une fonction qui prend en entrée n, qui simule l'expérience aléatoire décrite et qu'elle renvoie une réalisation de X.
   Dans un premier temps, on pourra utiliser une boucle `for`. *Puis on pourra simplifier en utilisant la commande* `np.sum(A<x)` *, où* `A` *est un vecteur et* `x` *un réel. En effet, lorsqu'on effectue des opérations algébriques sur* `True` *et* `False`, *Python les assimile respectivement à 1 et 0.*

   b) Déterminer la loi de X. Préciser son espérance et sa variance.

3) On considère désormais l'expérience suivante : on tire une boule dans l'urne avec remise jusqu'à obtenir une boule jaune. On note N la variable aléatoire égale au nombre de tirages effectuée et Y la variable aléatoire égale au nombre de boules bleues obtenues au cours de cette expérience.

   a) Compléter la fonction Python suivante afin qu'elle simule cette expérience et qu'elle renvoie des réalisations de N et Y.

```
import .....................
def simul():
    T=rd.random()
    N=1; Y=0
    if T>3/7:
        .....................
    while ..................
          T=rd.random()
           ..................
           if T>3/7:
              ..............
    return N,Y
```

   b) Déterminer la loi de N. Préciser son espérance et sa variance.

   c) Déterminer la loi du couple (N, Y).

   d) Calculer la probabilité $\mathbb{P}(Y = 0)$.

4) Expliquer ce que fait le script Python ci-dessous. Plus précisément que contient la variable `S` ?

```
import numpy as np
import matplotlib.pyplot as plt
Nexp=10000
L=[]
for k in range(Nexp):
    [N,Y]=simul()
    L.append(Y)
M=np.cumsum(L)/np.array(range(1,Nexp+1))
plt.plot(range(1,Nexp+1),M)
plt.show()
```

   Éxécuter-le plusieurs fois. Que peut-on conjecturer ? Sur quel résultat théorique s'appuie-t-on pour faire cette conjecture ?

5) Montrer la conjecture à l'aide de la formule de transfert.

Exercice 19 – D'après ECRICOME 2019.

Une urne contient initialement une boule blanche et une boule noire. On effectue une succession de tirages d'une boule dans cette urne. Après chaque tirage, on remet la boule tirée dans l'urne, et on rajoute dans l'urne une boule de couleur opposée à celle qui vient d'être tirée. On suppose que cette expérience est modélisée par un espace probabilisé $(\Omega, \mathcal{A}, \mathbb{P})$. Pour tout $k \in \mathbb{N}$, on note X_k le nombre de boules blanches présentes dans l'urne juste avant le $(k+1)^{\text{ième}}$ tirage. En particulier, on a $X_0 = 1$. On admet que pour tout entier k, X_k est une variable aléatoire de $(\Omega, \mathcal{A}, \mathbb{P})$.

1) Soit $k \in \mathbb{N}$.
   a) Déterminer $X_k(\Omega)$.
   b) Pour tous $i \in X_{k+1}(\Omega)$ et $j \in X_k(\Omega)$, déterminer $\mathbb{P}_{[X_k=j]}(X_{k+1} = i)$.
   On distinguera différents cas selon les valeurs relatives de i et j.
   c) En déduire que
   $$\forall i \in [\![2\,;k+1]\!], \qquad \mathbb{P}(X_{k+1} = i) = \frac{i}{k+2}\mathbb{P}(X_k = i) + \frac{3+k-i}{k+2}\mathbb{P}(X_k = i-1).$$
   d) Montrer que $\mathbb{P}(X_k = 1) = \dfrac{1}{(k+1)!}$.
   e) Déterminer la valeur de $\mathbb{P}(X_k = k+1)$.
2) Que renvoie la fonction Python suivante pour un entier k non nul implémenté en Python par `k` ?

```
import numpy.random as rd
def mystere(k):
    n=1
    b=1
    for i in range(k):
        r=rd.randint(1,n+b+1)
        if r>n:
            n=n+1
        else:
            b=b+1
    return b
```

3) Écrire une fonction Python d'en-tête appelée `loi_exp` qui prend en argument deux entiers strictement positifs k et N, qui effectue N simulations de k tirages successifs dans l'urne et qui retourne un vecteur `LE` qui contient une estimation de la loi de X_k (c'est-à-dire que pour chaque $i \in [\![1\,;k+1]\!]$, `LE[i-1]` contient la fréquence d'apparition de l'événement $[X_k = i]$ au cours des N simulations). On justifiera cette approche en citant un théorème du cours.
   On pourra utiliser la fonction `mystere`.
4) Recopier et compléter la fonction suivante afin qu'elle prenne en entrée un entier strictement positif n et qu'elle retourne un vecteur qui contient la loi théorème de X_n.

```
import numpy as np
def loi_theo(n):
    M=np.zeros([n+1,n+1])
    M[0,0]=1
    M[1,0]=1/2
    M[1,1]=1/2
    for k in range(1,n):
        M[k+1,0]=.....
        for i in range(2,k+2):
            M[k+1,i-1]=.....
        M[k+1,k+1]=.....
    return .....
```

5) Un étudiant nous propose comme loi de X_5 le résultat suivant :

k	1	2	3	4	5	6
$\mathbb{P}(X_5 = k)$	0.001368	0.079365	0.419434	0.418999	0.079454	0.00138

A-t-il utilisé `loi_exp` ou bien `loi_theo` ?

6) Pour différentes valeurs de n, superposer le diagramme à barres de la loi théorique et le digramme à barres de $N = 10000$ réalisations de l'expérience.
Si `X` *et* `Y` *sont des vecteurs de même taille, la commande* `plt.bar(X,Y)` *construit un diagramme à barre dont les abscisses des barres sont les coordonnées de* `X` *et la hauteur des barres les coordonnées de* `Y`. *On peut ajouter une option de couleur comme pour* `plt.plot(X,Y)` *et une option pour changer la largeur des barres (essayez* `width=0.4` *pour l'un des deux diagramme).*

Exercice 20 – D'après les oraux ESCP 1999 et 2001. (★★) Une urne contient n boules numérotées de 1 à n. On prélève au hasard ces n boules une par une et sans remise. A la suite de l'expérience, on note, pour tout $i \in [\![1\,;n]\!]$, u_i le numéro de la boule obtenue au cours du $i^{\text{ième}}$ tirage. Pour tout $i \in [\![2\,;n]\!]$, on dit qu'il y a un record en i si l'on a $u_i > \max\{u_1, \dots, u_{i-1}\}$. D'autre part, on convient qu'il y a systématiquement un record à l'instant 1.

1) Calculer, pour tout $i \in [\![1\,;n]\!]$, la probabilité r_i qu'il y ait un record à l'instant i.
On pourra commencer par calculer, pour tout $k \in [\![i\,;n]\!]$, la probabilité qu'il y ait un record au $i^{\text{ième}}$ tirage en tirant la boule numérotée k lors de ce tirage. On pourra utiliser, en la redémontrant, la formule de Pascal généralisée :
$$\forall n \in \mathbb{N}^*, \qquad \forall p \in [\![1\,;n]\!], \qquad \sum_{j=p}^{n} \binom{j-1}{p-1} = \binom{n}{p}.$$

2) Calculer les probabilités que, durant la totalité des tirages, on assiste exactement à :
   a) un seul record.
   b) n records.
   c) deux records.

3) On considère la fonction Python ci-dessous :

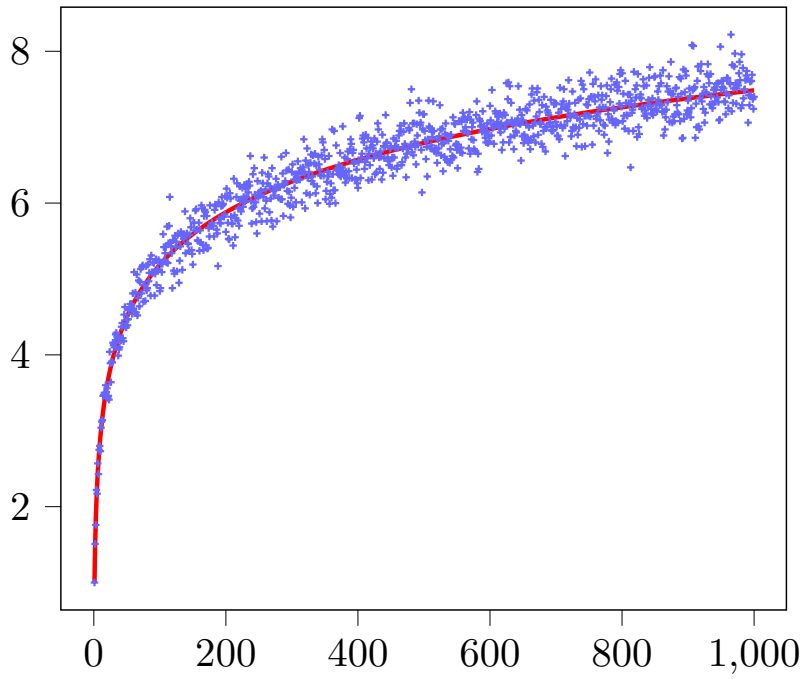
```
import numpy.random as rd
def Tirages(n):
    L=list(range(n))
    T=[]
    for k in range(n):
        i=rd.randint(n-k)
        x=L.pop(i)#Renvoie et enlève L[i]
        T.append(x)
    S=1
    m=T[0]
    for k in range(1,n):
        if T[k]>m:
            m=T[k]
            S=S+1
    return T,S
```

Expliquer ce que représentent les variables `T,m,S` ? Après exécution de cette fonction pour un n quelconque, qu'est-il affiché à l'écran ?

4) Après exécution de la fonction suivante,

```
import matplotlib.pyplot as plt
import numpy as np
N=100
n=1000
E=[1]
for k in range(2,n+1):
    S=0
    for i in range(N):
        S=S+Tirages(k)[1]
    E.append(S/N)
plt.plot(range(1,n+1),E,'+')
H=np.cumsum([1/i for i in range(1,n+1)])
plt.plot(range(1,n+1),H)
plt.show()
```

on obtient le graphique suivant :



Interpréter le graphique puis justifier vos observations.

On pourra introduire, pour tout $i \in [\![2\,;n]\!]$, la variable aléatoire X_i qui prend la valeur 1 s'il y a un record à l'instant i et 0 sinon.

Feuille de TP n° 1

Variables aléatoires à densité avec Python

I Représentation graphique des lois à densité usuelles

Exercice 1 – Représentation de lois uniformes. (★)

1) Écrire une fonction qui prend en argument deux réels a et b tels que $a < b$ et qui trace la densité d'une variable aléatoire de loi $\mathcal{U}([a\,;b])$ sur l'intervalle $[a-1\,;b+1]$.
2) Écrire une fonction qui prend en argument deux réels a et b tels que $a < b$ et qui trace la fonction de répartition d'une variable aléatoire de loi $\mathcal{U}([a\,;b])$ sur l'intervalle $[a-1\,;b+1]$.
3) Tester les fonctions précédentes pour quelques valeurs de a et b.

Exercice 2 – Représentation de lois exponentielles. (★)

1) Soient $a > 0$ et X une variable aléatoire de loi $\mathcal{E}(a)$. Déterminer $t_a \in \mathbb{R}$ (le plus petit possible) tel que $\mathbb{P}(X > t_a) \leqslant 10^{-3}$.
2) Écrire une fonction qui prend en argument $a > 0$ et qui trace la densité d'une variable aléatoire de loi $\mathcal{E}(a)$ sur l'intervalle $[-1\,;t_a]$.
3) Écrire une fonction qui prend en argument $a > 0$ et qui trace la fonction de répartition d'une variable aléatoire de loi $\mathcal{E}(a)$ sur l'intervalle $[-1\,;t_a]$.
4) Tester les fonctions précédentes pour quelques valeurs de a.

Exercice 3 – Représentation de lois normales. (★)

1) Écrire une fonction qui prend en argument $m \in \mathbb{R}$ et $\sigma^2 > 0$ et qui trace la densité d'une variable aléatoire de loi $\mathcal{N}(m,\sigma^2)$ sur l'intervalle $[m-4\sigma\,;m+4\sigma]$.
2) a) Exprimer la fonction de répartition d'une variable aléatoire de loi $\mathcal{N}(m,\sigma^2)$ en fonction de Φ.
   b) Écrire une fonction qui prend en argument $m \in \mathbb{R}$ et $\sigma^2 > 0$ et qui trace la fonction de répartition d'une variable aléatoire de loi $\mathcal{N}(m,\sigma^2)$ sur l'intervalle $[m-4\sigma\,;m+4\sigma]$.
   On utilisera la fonction `Phi` *implémentée dans le TP précédent ou bien la fonction* `ndtr` *de la bibliothèque* `scipy.special` *(importée sous le nom* `sp` *par exemple).*
3) Tester les fonctions précédentes pour quelques valeurs de m et σ^2.

Exercice 4 – Représentation de lois gamma. (★★)

1) Écrire une fonction qui prend en argument un **entier naturel** n non nul et qui trace la densité d'une variable aléatoire de loi $\gamma(n)$ sur l'intervalle $[-1\,;5n]$.
2) Écrire une fonction qui prend en argument un **entier naturel** n supérieure ou égale à 2 (pour $n = 1$, voir cf. exercice 2) et qui trace la fonction de répartition d'une variable aléatoire de loi $\gamma(n)$ sur l'intervalle $[-1\,;5n]$.
   On utilisera la méthode des rectangles.
3) Tester les fonctions précédentes pour quelques valeurs de n.

II Simulation de variables aléatoires à densité

1) Utilisation de `numpy.random`

Soient $a, b, \lambda, m, \sigma^2, \nu$ des réels tels que $a < b$, $\lambda > 0$, $\sigma^2 > 0$ et $\mu > 0$. On les implémente en Python dans les variables `a,b,lam,m,s2,mu` respectivement. La commande :

- (b-a)*rd.random()+a simule une variable aléatoire de loi $\mathcal{U}([a\,;b])$.
- np.exponential(1/lam) simule une variable aléatoire de loi $\mathcal{E}(\lambda)$.
- np.normal(m,s2**(1/2)) simule une variable aléatoire de loi $\mathcal{N}(m, \sigma^2)$.
- np.gamma(n,1) simule une variable aléatoire de loi $\gamma(\nu)$.

Exercice 5 – « Vérification » des simulations. (★★)

1) Écrire une fonction qui prend en argument des réels a et b (avec $a < b$), qui simule un échantillon de 10000 variables aléatoires de loi $\mathcal{U}([a\,;b])$ en utilisant `rd.random`, qui trace l'histogramme empirique renormalisé de cet échantillon et qui lui superpose le graphe d'une densité d'une variable de loi $\mathcal{U}([a\,;b])$. Tester avec plusieurs valeurs de a et b.
2) Écrire une fonction qui prend en argument un réel strictement positif a, qui simule un échantillon de 10000 variables aléatoires de loi $\mathcal{E}(a)$ en utilisant `rd.exponential`, qui trace l'histogramme empirique renormalisé de cet échantillon et qui lui superpose le graphe d'une densité d'une variable de loi $\mathcal{E}(a)$. Tester avec plusieurs valeurs de a.
3) Écrire une fonction qui prend en argument deux réels m et σ^2 avec $\sigma^2 > 0$, qui simule un échantillon de 10000 variables aléatoires de loi $\mathcal{N}(m, \sigma^2)$ en utilisant `rd.normal`, qui trace l'histogramme empirique renormalisé de cet échantillon et qui lui superpose le graphe d'une densité d'une variable de loi $\mathcal{N}(m, \sigma^2)$. Tester avec plusieurs valeurs de m et σ^2.
4) Écrire une fonction qui prend en argument un entier naturel strictement positif n, qui simule un échantillon de 10000 variables aléatoires de loi $\gamma(n)$ en utilisant `rd.gamma`, qui trace l'histogramme empirique renormalisé de cet échantillon et qui lui superpose le graphe d'une densité d'une variable de loi $\gamma(n)$. Tester avec plusieurs valeurs de n.

Exercice 6 – Somme de loi exponentielles. (★★)

1) Écrire une fonction en Python qui prend en argument un entier naturel n non nul et qui renvoie la somme de n variables aléatoire de lois $\mathcal{E}(1)$.
2) Pour différentes valeurs de n, exécuter 10000 fois cette fonction, construire l'histogramme empirique renormalisé de ces réalisations et lui superposer le graphe d'une densité d'une variable aléatoire de loi $\gamma(n)$.

Exercice 7. (★★) On dispose d'un bâton d'un mètre. On le casse en deux morceaux en choisissant le point de cassure au hasard (uniformément). On note X la longueur du plus petit morceau et Y la longueur du plus grand morceau.

1) a) Écrire une fonction qui prend en argument $N \in \mathbb{N}^*$ et qui renvoie un vecteur (une matrice ligne et non pas une liste) contenant N réalisations indépendantes de X.
   b) Tracer l'histogramme empirique renormalisé de 10000 réalisations de X. Que peut-on conjecturer ?
   c) Recopier et exécuter le script suivant :

```
1  import numpy as np
2  import matplotlib.pyplot as plt
3  N=10000
4  X=SimulX(N)
5  T=np.linspace(0,1,1000)
6  F=[]
7  for t in T:
8      F.append(np.sum(X<=t)/N)
9  plt.plot(T,F)
10 plt.show()
```

   Que fait ce script (en particulier que font les lignes 7 et 8) ? Que peut-on conjecturer de nouveau ?

d) Prouver la conjecture.

2) a) Tracer l'histogramme empirique renormalisé de 10000 réalisations de $Z = X/Y$.
   b) Déterminer la fonction de répartition de Z puis une densité de Z.
   c) Superposer le graphe de cette densité à l'histogramme de la question a.
   d) Superposer le graphe fonction de répartition empirique (qu'est ce qu'on peut entendre par là ?) de cet échantillon et le graphe de la fonction de répartition de Z ? Qu'observe-t-on ?

Exercice 8 – Fonction de répartition empirique. (★★) Soit $(X_n)_{n\geqslant 1}$ une suite de variables aléatoires indépendantes de même loi définies sur un espace probabilisé $(\Omega, \mathcal{A}, \mathbb{P})$. Pour tout $n \in \mathbb{N}^*$, on appelle fonction de répartition empirique de l'échantillon $(X_1, \dots, X_n)$ la fonction

$$F_n : (\omega, t) \in \Omega \times \mathbb{R} \longmapsto \frac{\text{card}(\{i \in [\![1\,;n]\!] \mid X_i(\omega) \leqslant t\})}{n} = \frac{1}{n}\sum_{i=1}^{n} \mathbb{1}_{X_i(\omega)\leqslant t}.$$

Autrement dit, pour tous $\omega \in \Omega$ et $t \in \mathbb{R}$, $F_n(\omega, t)$ est la proportion de variables aléatoires de l'échantillon qui sont à valeurs inférieures ou égales à t.

1) À l'aide des exemples de l'exercice précédent, construire une fonction qui prend en argument un vecteur (implémentant un échantillon) et qui trace la courbe de la fonction de répartition empirique de cet échantillon.
2) Pour plusieurs choix de paramètre, tester cette fonction avec un vecteurs de 10000 réalisations de variables aléatoires indépendantes de loi exponentielle et lui superposer la courbe de la fonction de répartition théorique.
3) Pour plusieurs choix de paramètres, tester cette fonction avec un vecteurs de 10000 réalisations de variables aléatoires indépendantes de loi binomiale et lui superposer la courbe de la fonction de répartition théorique.
4) Comment expliquer que ces courbes se superposent ?

2) Utilisation de la méthode d'inversion

Exercice 9 – Méthode d'inversion de la fonction de répartition. (★★★) Soit X une variable aléatoire réelle admettant une densité. On suppose qu'il existe $a \in \mathbb{R} \cup \{-\infty\}$ et $b \in \mathbb{R} \cup \{+\infty\}$ tels que $a < b$ et :

- la fonction de répartition F de X est strictement croissante sur $]a\,;b[$,
- $\mathbb{P}(X \in]a\,;b[) = 1$.

On se donne U une variable aléatoire de loi $\mathcal{U}([0\,;1])$.

1) a) Calculer les limites de F en a et b dans le cas où ce sont des réels.
   b) Justifier que F est une bijection de $]a\,;b[$ sur $]0\,;1[$.
2) a) On pose $Y = F^{-1}(U)$. Déterminer la fonction de répartition de Y.
   b) En déduire une méthode pour simuler la variable aléatoire X.
3) Écrire une fonction qui prend en argument $a > 0$ et qui simule une variable aléatoire de loi $\mathcal{E}(a)$ avec la méthode d'inversion.
4) On dit qu'une variable aléatoire suit une loi de Laplace si elle admet pour densité la fonction $x \longmapsto \frac{1}{2}e^{-|x|}$.
   a) Montrer que la fonction de répartition d'une variable de loi de Laplace satisfait aux hypothèses de l'énoncé avec $a = -\infty$ et $b = +\infty$ et que sa bijection réciproque est :
   $$x \longmapsto \begin{cases} \ln(2x) & \text{si} \quad x \in]0\,;1/2] \\ -\ln(2(1-x)) & \text{si} \quad x \in]1/2\,;1[\end{cases}$$
   b) Écrire une fonction sans argument qui simule une variable aléatoire de loi de Laplace avec la méthode d'inversion.
5) a) Écrire une fonction sans argument et qui simule une variable aléatoire de loi de Cauchy, c'est-à-dire de densité $x \longmapsto \frac{1}{\pi(1+x^2)}$.
   b) Soit $(X_i)_{i\geqslant 1}$ est une suite de variables aléatoires indépendantes et de loi de Cauchy. Recopier, compléter et exécuter le script suivant afin qu'il représente les 10000 premières valeurs de la suite $\left(\frac{1}{n}\sum_{k=1}^{n} X_i\right)_{n\geqslant 1}$.

```
.....
.....
.....
L=[]
S=0
for n in range(1,10001):
    X=.....#Simule une Cauchy
    S=.....
    L.append(.....)
.....#Trace
plt.show()
```

Que remarque-t-on ? Comment l'expliquer ?

6) On se donne maintenant X une variable aléatoire à densité mais on ne suppose plus rien sur sa fonction de répartition. Pour tout $x \in]0\,;1[$, on introduit l'ensemble $I_x = \{t \in \mathbb{R} \,|\, F(t) \geqslant x\}$.
   a) Soit $x \in]0\,;1[$. Montrer que I_x admet une borne inférieure. On la note[1] $G(x)$.
   b) Montrer que $G(x) \in I_x$ puis que $I_x = [G(x)\,;+\infty[$.
   On pourra utiliser, après l'avoir démontré, le fait que $G(x)$ est limite d'une suite d'éléments de I_x.
   c) Montrer que, pour tous $x \in]0\,;1[$ et $t \in \mathbb{R}$, $G(x) \leqslant t \iff F(t) \geqslant x$.
   d) En déduire que, si $U \hookrightarrow \mathcal{U}(]0\,;1[)$, alors les variables aléatoires X et $G(U)$ ont la même loi.

3) Retour sur la simulation de lois discrètes

Exercice 10 – Lois géométriques à partir de lois exponentielles. (★★)

1) Soit $X_a \hookrightarrow \mathcal{E}(a)$ avec $a > 0$. Déterminer la loi de $Y_a = \lfloor X_a \rfloor + 1$.
2) En déduire une méthode de simulation d'une variable aléatoire de loi géométrique de paramètre $p \in]0\,;1[$ en utilisant une simulation d'une variable aléatoire de loi exponentielle (à l'aide de la méthode d'inversion par exemple).
3) Écrire une fonction qui prend en argument $p \in]0\,;1[$, qui trace le diagramme à barres empirique d'un échantillon de 10000 simulations de lois géométriques obtenues via cette méthode, et lui superpose le diagramme à barre théorique d'une loi $\mathcal{G}(p)$. La tester pour différentes valeurs de p.

Exercice 11 – Simulation de lois de Poisson. (★★) Soit $\lambda \in \mathbb{R}_+^*$. On se donne $(U_i)_{i\geqslant 1}$ une suite de variables aléatoires indépendantes et de même loi $\mathcal{U}(]0\,;1])$. On considère la variable aléatoire

$$Y = \min\{n \in \mathbb{N} \,|\, U_1 U_2 \dots U_{n+1} \leqslant e^{-\lambda}\}.$$

1) Pour tout $i \in \mathbb{N}^*$, on pose $F_i = -\ln(U_i)$. Quelle est la loi de F_i ?
2) En déduire la loi de $S_n = F_1 + \dots + F_n$ pour tout $n \in \mathbb{N}^*$.
3) Calculer $\mathbb{P}(Y > n)$ pour tout $n \in \mathbb{N}^*$.
4) En déduire que Y suit une loi de Poisson de paramètre λ. Commenter en réécrivant Y à l'aide des F_i, $i \geqslant 1$.
5) Compléter la fonction suivante afin qu'elle prenne en argument λ et renvoie la valeur d'une réalisation d'une variable aléatoire de loi $\mathcal{P}(\lambda)$.

```
.....
def Poisson(lam):
    n=0
    P=rd.random()
    while .....
        .....
    return n
```

1. Si $t < G(x)$, alors $t \notin I_x$ donc $F(t) < x$. En faisant tendre t vers $G(x)$, et par continuité de F en $G(x)$ (c'est la fonction de répartition d'une variable à densité) on obtient $F(G(x)) \leqslant x$. Comme $G(x) \in I_x$, on en déduit que $F(G(x)) = x$. Ainsi $G(x)$ est le plus petit antécédent de x. Si F est bijective alors il s'agit de $F^{-1}(x)$. Lorsque F n'est pas bijective, comme elle est croissante, elle possède des paliers et on prend alors la borne inférieure du palier.

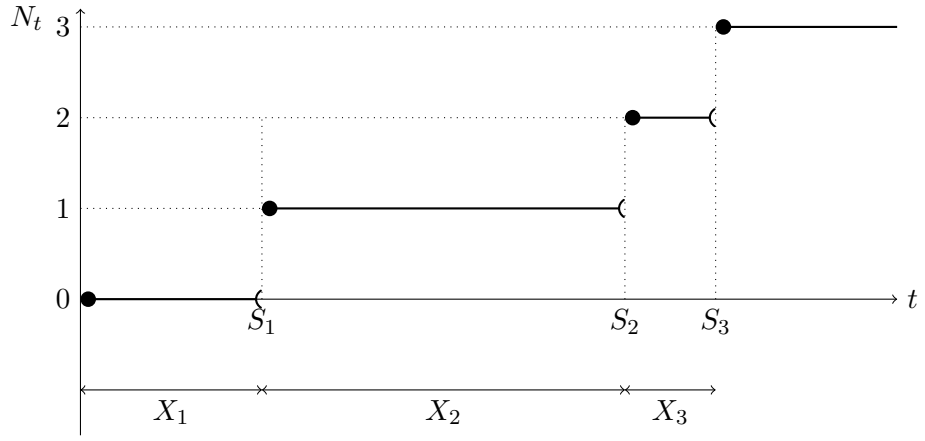
Exercice 12 – D'après concours. (★★★) On considère une suite de variables aléatoires $(X_n)_{n\geqslant 1}$, définies sur un espace probabilisé $(\Omega, \mathcal{A}, \mathbb{P})$ et qui sont mutuellement indépendantes et identiquement distribuées selon une loi exponentielle de paramètre $\lambda \in \mathbb{R}_+^*$. Pour tout $n \in \mathbb{N}$, on note $S_n = \sum_{k=1}^{n} X_k$, avec la convention $S_0 = 0$.

Enfin, pour tout $t \in \mathbb{R}_+^*$, on note N_t la variable aléatoire égale à la plus grande valeur de n pour laquelle S_n est inférieure ou égale à t, c'est-à-dire :

$$\forall \omega \in \Omega, \qquad N_t(\omega) = \sup\{n \in \mathbb{N} \,|\, S_n(\omega) \leqslant t\}.$$

Par convention, si l'ensemble écrit ci-dessus n'est pas fini, N_t prend la valeur -1.

Voici un exemple de réalisation de N_t en fonction de t :



1) Pour tout $t \in \mathbb{R}_+^*$, montrer que $\mathbb{P}(N_t = 0) = e^{-\lambda t}$.
2) Pour tout $n \in \mathbb{N}^*$, déterminer une densité de la variable aléatoire λS_n.
3) Soit $t \in \mathbb{R}_+^*$.
   a) Comparer les événements $[N_t = -1]$ et $\bigcap_{n=1}^{+\infty} [S_n \leqslant t]$.
   b) Montrer que, pour tout $x \in \mathbb{R}_+^*$,
   $$\lim_{n\to+\infty} \int_0^x \frac{u^{n-1}}{(n-1)!} e^{-u}\, du = 0.$$
   c) En déduire que $\mathbb{P}(N_t = -1) = 0$.
   d) Établir que, pour tout $n \in \mathbb{N}$, $\mathbb{P}(N_t \geqslant n) = \mathbb{P}(S_n \leqslant t)$.
4) Soient $t \in \mathbb{R}_+^*$ et $n \in \mathbb{N}$.
   a) Montrer que $\mathbb{P}([N_t = n]) = \mathbb{P}([S_n \leqslant t] \cap [S_{n+1} > t])$.
   b) En déduire que
   $$\mathbb{P}(N_t = n) = \int_0^{\lambda t} \frac{u^{n-1}}{(n-1)!} e^{-u}\, du - \int_0^{\lambda t} \frac{u^n}{n!} e^{-u}\, du.$$
   c) En intégrant par parties une des intégrales ci-dessus, montrer que
   $$\forall n \in \mathbb{N}^*, \qquad \mathbb{P}(N_t = n) = \frac{(\lambda t)^n}{n!} e^{-\lambda t}.$$
   Quelle est la loi de N_t?
5) a) Écrire une fonction Python d'en-tête `def simulation_S(n,lam):` renvoyant une réalisation de S_n.
   b) Écrire une fonction Python d'en-tête `def simulation_N(t,lam):` renvoyant une réalisation de N_t.
   c) Compléter la fonction en Python `evolution_S` afin qu'elle renvoie toutes les valeurs $S_1, S_2, \ldots, S_n$ tant que $S_n \leqslant t$.

```
.....
def evolution_S(t,lam):
    L=[]
    S=.....
    while .....
        L.append(S)
        S=S+.....
    return .....
```

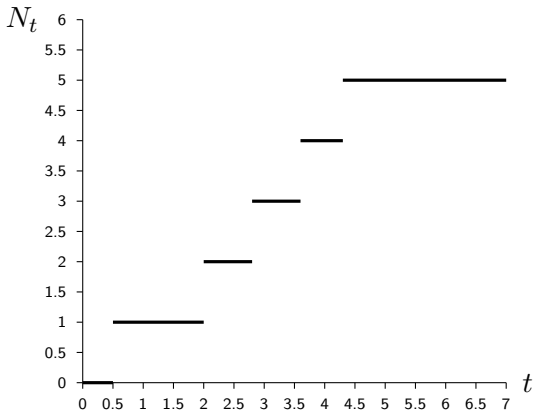
d) On a commencé à écrire une fonction Python ci-dessous. Dans ce script, on implémente par `S` la liste constituée de $S_1, \dots, S_n$ (où n est le plus grand entier tel que $S_n \leqslant T$) et on souhaite tracer l'évolution de N_t du temps $t = 0$ au temps $t = T$ de la même manière que sur la figure de la page précédente.

```
import matplotlib.pyplot as plt
def trace_N(T,lam):
    S=evolution_S(T,lam)
    n=len(S)
    plt.plot([0,S[0]],[0,0])
    for i in range(1,n):
        .....
    plt.show()
```

Par laquelle des instructions suivantes faut-il compléter la ligne manquante ?

(i) `plt.plot([i-1,i],[S[i-1],S[i-1]])` (ii) `plt.plot([i,i+1],[S[i],S[i+1]])`
(iii) `plt.plot([S[i],S[i+1]],[i,i])` (iv) `plt.plot([S[i-1],S[i]],[i,i])`

e) Un étudiant exécute le script précédent pour $T = 7$ et $\lambda = 1$ et obtient la figure suivante :



Que valent dans ce cas $N_{3,2}$ et $N_{5,5}$? Donner une valeur approximative de S_2 et de X_4.

Exercice 13 – Simulation de lois finies quelconques. (★★) Soit X une variable aléatoire finie définie sur un espace probabilisé $(\Omega, \mathcal{A}, \mathbb{P})$. On suppose que $X(\Omega) = \{x_1, \dots, x_n\}$ et, pour tout $i \in [\![1\,;n]\!]$, on note $p_i = \mathbb{P}(X = x_i)$. Pour tout $k \in [\![1\,;n]\!]$, notons $f_k = \sum_{i=1}^{k} p_i$. On pose $f_0 = 0$.

1) Soit U une variable aléatoire de loi $\mathcal{U}(]0\,;1[)$. Montrer que, pour tout $k \in [\![1\,;n]\!]$, $\mathbb{P}(f_{k-1} < U \leqslant f_k) = p_k$.
2) En déduire une méthode de simulation de X à l'aide de U.
3) Soit `P` une liste implémentée en Python. Que font les commandes suivantes ?

```
F=np.cumsum(P)
np.sum(F<rd.random())
```

4) En déduire une fonction qui prend en entrée une liste implémentant $x_1, \dots, x_n$ et une liste implémentant $p_1, \dots, p_n$ et qui simule X.

Feuille de TP n° 2

Statistiques bivariées

I Introduction

Dans une population donnée, on peut souhaiter étudier simultanément deux caractères X et Y. Par exemple :

- la taille X et le poids Y d'un individu en France.
- la note X en maths et la note Y en HGG d'un élève au concours HEC.
- la température X et le taux d'humidité Y un jour donné à Paris.
- la densité X de population et le taux de criminalité Y dans une ville.

Nous supposerons (comme dans les exemples ci-dessus) que les caractères étudiés sont quantitatifs. On peut alors s'intéresser aux propriétés de chacun des deux caractères pris séparément (on parle de statistiques univariées), mais aussi au lien entre ces 2 caractères (on parle statistiques bivariées). En particulier, on peut penser que l'une des variables, Y par exemple, est une cause de l'autre, par exemple X. On dit alors que X est la **variable explicative** et Y est la **variable à expliquer**. Dans le cas où X est explicative et Y à expliquer, on tentera d'exprimer Y en fonction de X en commençant par tracer le nuage des points de Y en fonction de X pour deviner la relation entre ces données.

Pour cela, on commence par faire un relevé statistique : on prélève sur la population initiale un échantillon de taille $n \in \mathbb{N}^*$ (ou sur la population entière lorsque cela est possible) et on note les valeurs des variables X et Y observées sur ces n individus. On dispose alors d'une **série statistique double**, noté $\big((x_1,y_1),\dots,(x_n,y_n)\big)$ où, pour tout $k \in [\![1\,;n]\!]$, le caractère X (resp. Y) de l'individu n° k vaut x_k (resp. y_k). On note $x=(x_1,\dots,x_n)$ et $y=(y_1,\dots,y_n)$.

On suppose que les données $x_1,\dots,x_n$ sont implémentées en Python dans une liste ou un vecteur `X` et que $y_1,\dots,y_n$ sont implémentées en Python dans une liste ou un vecteur `Y`.

On munit l'espace euclidien $\mathbb{R}^n$ de son produit scalaire canonique noté $\langle\cdot\,,\cdot\rangle$ et de sa norme euclidienne associée notée $\|\cdot\|$.

II Modèle de régression

1) Nuage de points

Définition. *Le nuage de points associé à la série statistique* $\big((x_1,y_1),\dots,(x_n,y_n)\big)$ *est l'ensemble des points du plan* $\mathbb{R}^2$ *de coordonnées* (x_1,y_1), (x_2,y_2), $\cdots$, (x_n,y_n).

La commande Python est la suivante :

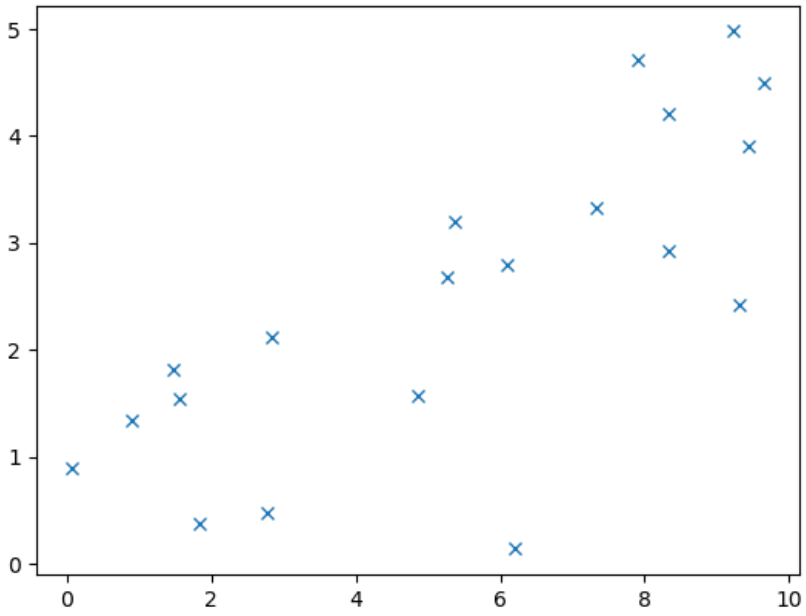
```
import matplotlib.pyplot as plt
plt.plot(X,Y,'x')
plt.show()
```

Exemple : *Si les données sont implémentées dans*

```
X=[0.06,0.9,1.47,1.56,1.84,2.77,2.83,4.86,5.26,5.38,6.1,6.2,
                                        7.34,7.9,8.32,8.32,9.22,9.32,9.44,9.66]
```

```
Y=[0.89,1.34,1.82,1.55,0.38,0.48,2.12,1.57,2.68,3.2,2.8,0.14,
                                        3.33,4.71,2.92,4.21,4.98,2.42,3.9,4.5],
```

alors le nuage de points correspondant est :



L'examen du nuage de points permet de faire des constatations qualitatives :

- Est-il concentré ou dispersé ?
- Relève-t-on une tendance (variations dans le même sens, courbe particulière, etc.) ?
- Y a-t-il des valeurs aberrantes ?

2) Point moyen

Définition. *Soit* $((x_1, y_1), \dots, (x_n, y_n))$ *un série statistique double.*

- *On appelle point moyen de la série statistique le point du plan* $(\overline{x}, \overline{y})$ *où*

$$\overline{x} = \frac{1}{n}\sum_{k=1}^{n} x_i \qquad \text{et} \qquad \overline{y} = \frac{1}{n}\sum_{k=1}^{n} y_i.$$

- *On définit la variance de la série statistique* $(x_1, \dots, x_n)$ *par* $\sigma_x^2 = \dfrac{1}{n}\displaystyle\sum_{k=1}^{n}(x_i - \overline{x})^2$.
- *On définit la variance de la série statistique* $(y_1, \dots, y_n)$ *par* $\sigma_y^2 = \dfrac{1}{n}\displaystyle\sum_{k=1}^{n}(y_i - \overline{y})^2$.

Remarques :

- On remarque que $\overline{x}$ est la moyenne des données $x_1, \dots, x_n$ et $\overline{y}$ est la moyenne des données $y_1, \dots, y_n$.
- On note $\overline{x^2} = \dfrac{1}{n}\displaystyle\sum_{k=1}^{n} x_i^2$ et $\overline{y^2} = \dfrac{1}{n}\displaystyle\sum_{k=1}^{n} y_i^2$. On a la formule de Koenig-Huygens :

$$\sigma_x^2 = \frac{1}{n}\sum_{k=1}^{n} x_i^2 - \overline{x}^2 = \overline{x^2} - \overline{x}^2 \qquad \text{et} \qquad \sigma_y^2 = \frac{1}{n}\sum_{k=1}^{n} y_i^2 - \overline{y}^2 = \overline{y^2} - \overline{y}^2.$$

- Si on note $u = (1, \dots, 1) \in \mathbb{R}^n$, on a

$$\overline{x} = \frac{1}{n}\langle x, u\rangle, \quad \overline{y} = \frac{1}{n}\langle y, u\rangle, \quad \sigma_x^2 = \frac{1}{n}\|x - \overline{x}u\|^2, \quad \sigma_y^2 = \frac{1}{n}\|y - \overline{y}u\|^2.$$

- On a $\sigma_x^2 = 0$ si et seulement si, pour tout $i \in [\![1\,;n]\!]$, $x_i = \overline{x}$ si et seulement tous les x_i, $1 \leqslant i \leqslant n$ sont tous égaux. Il est raisonnable de supposer que ce n'est pas le cas.

Avec Python, on calcule $\overline{x}, \overline{y}, \sigma_x^2, \sigma_y^2$ avec les commandes `np.mean(X)`, `np.mean(Y)`, `np.var(X)` et `np.var(Y)` respectivement.

Exemple : *Si on reprend l'exemple précédent, on obtient* $\overline{x} = 5.438$, $\overline{y} = 2.497$, $\sigma_x^2 = 9.903$ *et* $\sigma_y^2 = 2.093$ *approximativement.*

3) Covariance et coefficient de corrélation linéaire

Définition. *On définit la covariance de la série statistique double* $((x_1, y_1), \ldots, (x_n, y_n))$ *par*

$$\mathrm{Cov}(x, y) = \frac{1}{n}\sum_{k=1}^{n}(x_i - \overline{x})(y_i - \overline{y}).$$

Remarques :

- On note $\overline{xy} = \dfrac{1}{n}\displaystyle\sum_{k=1}^{n} x_i y_i$. On a la formule de Koenig-Huygens :

$$\mathrm{Cov}(x, y) = \frac{1}{n}\sum_{k=1}^{n} x_i y_i - \overline{x}\,\overline{y} = \overline{xy} - \overline{x}\,\overline{y}.$$

- On a $\mathrm{Cov}(x, y) = \dfrac{1}{n}\langle x - \overline{x}u, y - \overline{y}u\rangle$.
- L'inégalité de Cauchy-Schwarz entraîne que

$$|\mathrm{Cov}(x, y)| = \frac{1}{n}|\langle x - \overline{x}u, y - \overline{y}u\rangle| \leqslant \frac{1}{n}\|x - \overline{x}u\|\,\|y - \overline{y}u\| = \sqrt{\sigma_x^2}\sqrt{\sigma_y^2}$$

Avec Python, on calcule $\mathrm{Cov}(x, y)$ avec la commande `np.cov(x,y)[0,1]` (la commande `np.cov(x,y)` renvoie en fait une matrice symétrique d'ordre 2 dont les coefficients diagonaux sont les variances de x et y et les coefficients non diagonaux sont la covariance). Cependant cette commande n'est pas au programme donc il faut savoir la recalculer :

```
S=0
n=len(x)
for i in range(n):
    S=S+x[i]*y[i]
cov=S/n-np.mean(x)*np.mean(y)
```

Définition. *On définit le coefficient de corrélation linéaire de la série statistique double* $((x_1, y_1), \ldots, (x_n, y_n))$ *par*

$$\rho(x, y) = \frac{\mathrm{Cov}(x, y)}{\sqrt{\sigma_x^2}\sqrt{\sigma_y^2}},$$

lorsque $\sigma_x^2 \neq 0$ *et* $\sigma_y^2 \neq 0$.

Exemple : *Si on reprend l'exemple précédent, on obtient* $\mathrm{Cov}(x, y) = 3,655$ *et* $\rho(x, y) = 0.803$ *approximativement.*

Remarques :

- La dernière remarque entraîne que $\rho(x, y) \in [-1\,;1]$.
- La covariance (ou le coefficient de corrélation) est nul si et seulement si $x - \overline{x}u$ et $y - \overline{y}u$ sont orthogonaux.
- Le cas d'égalité de l'inégalité de Cauchy-Schwarz et la dernière remarque assure que $|\rho(x, y)| = 1$ si et seulement si $x - \overline{x}u$ et $y - \overline{y}$ sont colinéaires si et seulement si il existe $\lambda \in \mathbb{R}$ tel que $y - \overline{y} = \lambda(x - \overline{x})$ (ce qui se réécrit $y = \lambda(x - \overline{x}) + \overline{y}$) si et seulement si les points du nuage sont alignés. On en déduit :

Interprétations du coefficient de corrélation linéaire :

- Lorsque le coefficient de corrélation linéaire (ou la covariance) d'une série statistique est positive (resp. négative), le nuage de points a une forme « montante » (resp. « descendante »), c'est-à-dire les caractères X et Y varient dans le même sens (resp. le sens contraire).
- Lorsque $|\rho(x,y)|$ est proche de 1, il se peut que la variable à expliquer Y dépende de manière « presque » affine de X. On observera dans ce cas le nuage de points de la série statistique, et si celui-ci a bien l'allure d'une droite, on cherchera l'équation de la droite de régression linéaire (cf. paragraphe suivant).
- Une covariance (ou un coefficient de corrélation linéaire) proche de 0 traduit l'absence de relation affine entre les caractères X et Y.

⚠ Corrélation ne veut pas dire causalité (cf. exercice 3).

III Droite de régression linéaire

Supposons que X est la variable explicative et Y la variable à expliquer.

Chercher un modèle de régression consiste à savoir si Y est une fonction de X à un bruit près. Plus formellement, il s'agit de déterminer une fonction f telle que $Y = f(X) + \varepsilon$ où f est appelée fonction de régression et ε est une variable aléatoire réelle appelée erreur d'ajustement (ou bruit, ou résidu).

Dans la suite de ce paragraphe, nous verrons le cas où f est une fonction affine : il existe $(a,b) \in \mathbb{R}^2$ tel que $f(X) = aX + b$. On parle alors de régression linéaire. Il existe cependant bien d'autres choix pour la fonction f (par exemple $f(X) = e^{aX} + b$) et on essaye souvent de se ramener au cas linéaire (cf. exercices 1 et 2).

Lorsque le nuage de points associé à la série statistique a tendance à « s'étirer linéairement », il se peut que la variable à expliquer Y dépende de manière « presque » affine de X. On cherche dans ce cas l'équation de la droite s'approchant au mieux du nuage de points au sens des moindres carrés.

Définition. *La droite de régression linéaire de Y (variable à expliquer) en X (variable explicative) de la série statistique double $\big((x_1,y_1),\dots,(x_n,y_n)\big)$ est la droite d'équation $y = ax + b$ telle que la quantité*

$$F(a,b) = \sum_{k=1}^{n} (y_k - (ax_k + b))^2$$

est minimale.

Remarques :

- ⚠ Cette droite existe dès que les x_i, $1 \leqslant i \leqslant n$, ne sont pas tous égaux (ce qui est une hypothèse raisonnable). Par ailleurs elle est unique, comme on le verra dans la démonstration de la proposition suivante.
- Graphiquement, $F(a,b)$ représente la somme des carrés des distances entre les points de la droite d'abscisse x_k et les points du nuage de même abscisse x_k pour $k \in [\![1\,;n]\!]$.

Proposition. *Soit $\big((x_1,y_1),\dots,(x_n,y_n)\big)$ une série statistique double associée à X et Y. On suppose que les x_i, $1 \leqslant i \leqslant n$, ne sont pas tous égaux. On a alors $\sigma_x^2 > 0$ et la droite de régression linéaire de Y en X a pour équation $y = \widehat{a}x + \widehat{b}$ avec*

$$\widehat{a} = \frac{\mathrm{Cov}(x,y)}{\sigma_x^2} \quad \text{et} \quad \widehat{b} = \overline{y} - \frac{\mathrm{Cov}(x,y)}{\sigma_x^2}\overline{x},$$

c'est-à-dire c'est la droite d'équation

$$y = \frac{\mathrm{Cov}(x,y)}{\sigma_x^2}(x - \overline{x}) + \overline{y}.$$

Exemple : *Si on reprend l'exemple précédent, on calcule que $\widehat{a} = 0,369$ (avec la commande `a=np.cov(X,Y)[0,1]/np.var(X)`) et $\widehat{b} = 0,490$ (avec la commande `np.mean(Y)-a*np.mean(X)`).*

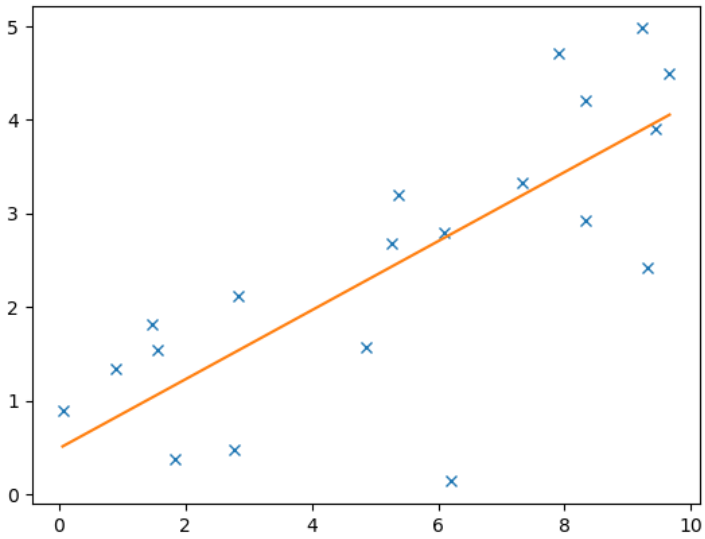
Démonstration.

□

Voici un programme en Python qui représente la droite de régression sur l'intervalle $[m, M]$, où m et M sont des réels tels que $m < M$ implémentés en Python par `m` et `M` (on peut prendre pour m et M la plus petite et la plus grand valeur respectivement de la série $(x_1, \ldots, x_n)$) :

```
import numpy as np
import matplotlib.pyplot as plt
a=np.cov(X,Y)[0,1]/np.var(X)
b=np.mean(Y)-a*np.mean(X)
plt.plot([m,M],[a*m+b,a*M+b])
plt.show()
```

Exemple : *Voici ce que l'on obtient avec l'exemple précédent, superposé au nuage de points :*

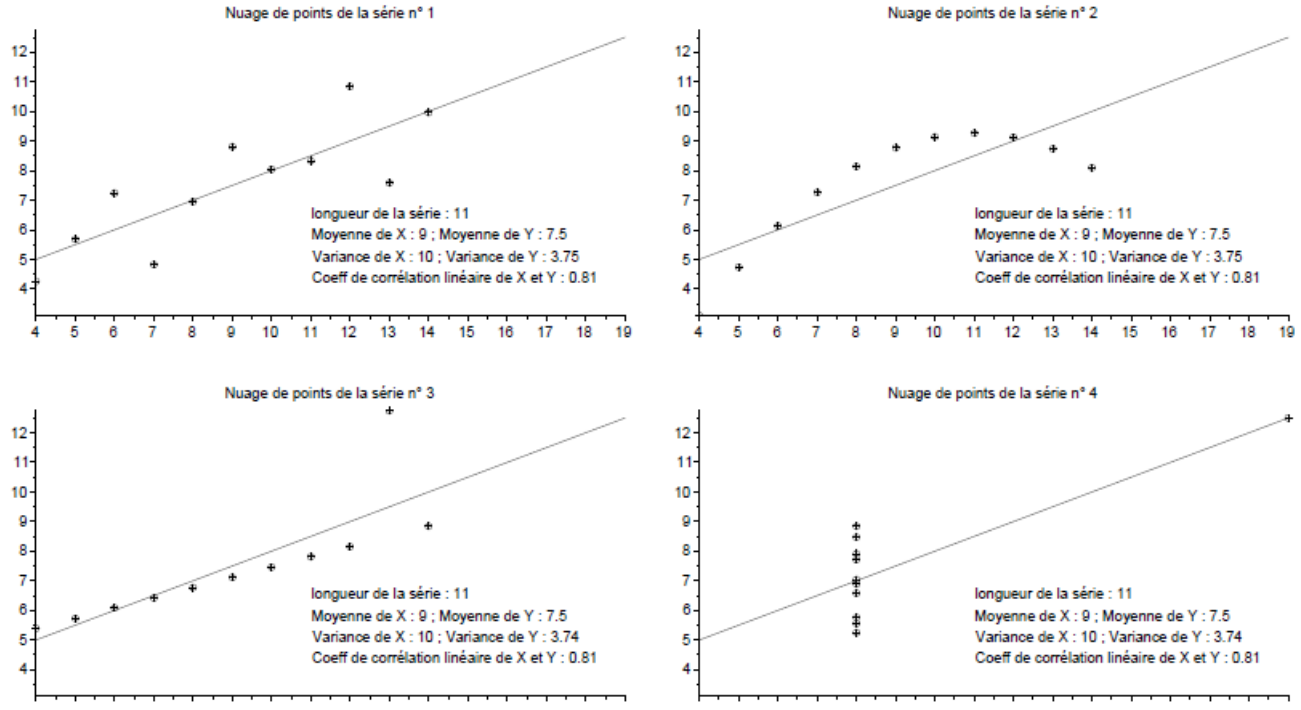


Remarque : Si la variable à expliquer est X et si Y est la variable explicative (mais que l'on a tracé X en abscisses et Y en ordonnée), on définit de même la droite de régression linéaire de X en Y de la série statistique (ou du nuage associé) : il s'agit de la droite d'équation $x = my + p$ telle que $G(m,p) = \sum_{k=1}^{n} \big(x_k - (my_k + p)\big)^2$ est minimal. De manière analogue, la droite de régression de X en Y a pour équation $x = \dfrac{\mathrm{Cov}(x,y)}{\sigma_y^2}(y - \overline{y}) + \overline{x}$. Il s'agit donc de la droite d'équation :

$$y = \frac{\sigma_y^2}{\mathrm{Cov}(x,y)}(x - \overline{x}) + \overline{y}$$

si la covariance ci-dessus est non nulle bien sûr.

⚠ Les points moyens, les variances, la covariance, la droite de régression ne sont que des indicateurs. Ils ne caractérisent eu aucun cas les variables X et Y ni un éventuel lien de cause à effet. Les exemples suivants sont assez parlant :



IV Exercices

Exercice 1 – D'après HEC ECE 2016. (★★) On s'intéresse à la fonction de production d'une entreprise qui produit un certain bien à une époque donnée. On note respectivement X et Y les quantités de travail et de capital requises pour produire une certaine quantité de ce bien, et l'on suppose que $X > 0$ et $Y > 0$.

On suppose que la production totale de l'entreprise est une variable aléatoire Q telle que $Q = BX^aY^{1-a}e^R$, où $a \in]0\,;1[$, $B > 0$ et R est une variable aléatoire suivant une loi normale centrée. Enfin, on pose :

$$b = \ln(B), \quad U = \ln(X) - \ln(Y), \quad \text{et} \quad T = \ln(Q) - \ln(Y).$$

On sélectionne $n \in \mathbb{N}^*$ entreprises qui produisent le bien considéré à l'époque donnée. On mesure pour chaque entreprise $i \in [\![1\,;n]\!]$ la quantité de travail x_i et la quantité de capital y_i utilisées ainsi que la quantité produite q_i. On suppose que, pour tout $i \in [\![1\,;n]\!]$, $x_i > 0$, $y_i > 0$ et $q_i > 0$.

Pour tout $i \in [\![1\,;n]\!]$, x_i, y_i et q_i sont des réalisations de variables aléatoires X_i, Y_i, Q_i ayant respectivement les mêmes lois que X, Y, et Q. On a $Q_i = BX_i^aY_i^{1-a}\exp(R_i)$ et $q_i = Bx_i^ay_i^{1-a}\exp(r_i)$. Ici $r_1, \ldots, r_n$ sont des réalisations de $R_1, R_2, ..., R_n$ qui sont des variables aléatoires supposées indépendantes et de même loi que R. On pose, pour tout $i \in [\![1\,;n]\!]$:

$$U_i = \ln X_i - \ln Y_i, \quad T_i = \ln Q_i - \ln Y_i \quad \text{et} \quad t_i = \ln q_i - \ln y_i$$

Ainsi, pour chaque entreprise $i \in [\![1\,;n]\!]$, t_i est une réalisation de la variable aléatoire T_i.

On a relevé pour $n = 16$ entreprises qui produisent le bien considéré à l'époque donnée, les deux séries statistiques $(u_i)_{1\leqslant i\leqslant n}$ et $(t_i)_{1\leqslant i\leqslant n}$ implémentées dans Python par les listes

```
u=[1.06,0.44,2.25,3.88,0.61,1.97,3.43,2.10,1.50,1.68,2.72,1.35,2.94,2.78,3.43,3.58]
```

```
t=[2.58,2.25,2.90,3.36,2.41,2.79,3.32,2.81,2.62,2.70,3.17,2.65,3.07,3.13,3.07,3.34]
```

1) Vérifier que $T = aU + b + R$ et, pour tout $i \in [\![1\,;n]\!]$, $t_i = au_i + b + r_i$.
2) Représenter sur un même graphique :
   - le nuage des points (u_1, t_1), (u_2, t_2), ..., (u_n, t_n).
   - la droite de régression de T en U.
   - la droite de régression de U en T.
3) Interpréter le point d'intersection des deux droites de régression.
4) Estimer graphiquement les moyennes empiriques $\overline{u}$ et $\overline{t}$.

Exercice 2 – Développement de bactéries. (★★) On étudie dans cet exercice le nombre de bactéries présentes dans un bouillon de culture au fur et à mesure du temps.

On dispose de la matrice de données suivante :

```
donnees=np.array([[0,1,2,3,4,5,6],[32,47,65,92,132,190,275]])
```

Cette matrice contient deux lignes, la seconde indiquant le nombre de bactéries (par unité de volume) présentes dans le bouillon de culture au cours de l'expérience : pour tout $i \in [\![0\,;6]\!]$, $x_i =$`donnees[0,i]`$= i$ et $y_i =$`donnees[1,i]` est le nombre de bactéries par unité de volume présentes dans le bouillon de culture après i heures.

1) Pour ce relevé statistique, quelle est la variable explicative X ? Quelle est la variable à expliquer Y ?
2) Représenter le nuage de points des données.
3) Avec Python calculer les moyennes $\overline{x}$ et $\overline{y}$, les variances σ_x^2 et σ_y^2 des séries statistiques X et Y, ainsi la covariance et le coefficient de corrélation linéaire r de la série statistique.
4) Superposer au nuage de points la droite de régression de y en x.

5) Le nuage de points de la série statistique peut laisser penser que Y est une fonction « exponentielle de X » : il semble qu'il existe α et λ deux réels strictement positifs tels que $Y \approx \lambda.e^{\alpha X}$. Remarquons que

$$Y \approx \lambda.e^{\alpha X} \qquad \Longleftrightarrow \qquad \ln(Y) \approx \alpha X + \ln \lambda.$$

Ainsi, il semble que la « nouvelle » variable à expliquer $\ln(Y)$ est une fonction « presque » affine de X. Nous allons alors étudier la série statistique bivariée de variable explicative X et de variable à expliquer $Z = \ln(Y)$, et nous allons déterminer la droite de régression linéaire de cette série.

a) Définir en Python un vecteur `Z` contenant $(s_0, \dots, z_6) = (\ln(y_0), \dots, \ln(y_6))$.

b) Déterminer l'équation de la droite de régression linéaire de Z en X, notée $z = \alpha x + \beta$.

c) Représenter alors le nuage de points de la série statistique $\big((x_0, z_0), (x_1, z_1), \dots, (x_6, z_6)\big)$ et lui superpose la droite en question. Commenter.

d) Superposer à la courbe de la question 1, la courbe d'équation $y = e^{\beta} e^{\alpha x}$. Commenter.

Exercice 3 – Corrélation ne veut pas dire causalité. (★)

Une bonne corrélation entre deux séries de données ne signifie pas pour autant qu'il existe un lien de cause à effet entre les deux. Voici deux exemples :

1) Ce tableau donne la consommation de margarine (en livre par personne) aux USA, et la taux de divorce dans l'état du Maine (en divorce pour 1000 personnes).

Année	2000	2001	2002	2003	2004	2005	2006	2997	2008	2009
Taux de divorce	8,2	7	6,5	5,3	5,2	4	4,6	4,5	4,2	3,7
Consommation de margarine	5	4,7	4,6	4,4	4,3	4,1	4,2	4,2	4,2	4,1

Calculer le coefficient de corrélation. Commenter.

2) Ce tableau donne la consommation de mozzarella (en livre par personne) et le nombre de doctorats en génie civil décernés aux USA.

Année	2000	2001	2002	2003	2004	2005	2006	2997	2008	2009
Consommation de mozzarella	9,3	9,7	9,7	9,7	9,9	10,2	10,5	11	10,6	10,6
Nombre de doctorats	480	501	540	552	547	622	655	701	712	708

Calculer le coefficient de corrélation. Commenter.

Exercice 4 – Autre démonstration de la droite de régression. (★★) Soient $x_1, \dots, x_n$ des réels qui ne sont pas tous égaux. Soient $y_1, \dots, y_n$ des réels. On note

$$\overline{x} = \frac{1}{n}\sum_{k=1}^{n} x_i, \quad \overline{y} = \frac{1}{n}\sum_{k=1}^{n} y_i \quad \overline{x^2} = \frac{1}{n}\sum_{k=1}^{n} x_i^2, \quad \overline{xy} = \frac{1}{n}\sum_{k=1}^{n} x_i y_i,$$

$$\sigma_x^2 = \frac{1}{n}\sum_{k=1}^{n}(x_i - \overline{x})^2 = \overline{x^2} - \overline{x}^2, \quad \text{et} \quad \mathrm{Cov}(x,y) = \frac{1}{n}\sum_{k=1}^{n}(x_i - \overline{x})(y_i - \overline{y}) = \overline{xy} - \overline{x}\,\overline{y}.$$

1) Montrer que $\sigma_x^2 > 0$.

2) Montrer que la fonction

$$F : (a,b) \in \mathbb{R}^2 \longmapsto \sum_{i=1}^{n}(y_i - ax_i - b)^2$$

est de classe C^1 sur $\mathbb{R}^2$ et admet pour unique point critique $(\widehat{a}, \widehat{b}) = \left(\dfrac{\mathrm{Cov}(x,y)}{\sigma_x^2}, \overline{y} - \dfrac{\mathrm{Cov}(x,y)}{\sigma_x^2}\overline{x}\right)$.

3) Montrer que, pour tout $(a,b) \in \mathbb{R}^2$,

$$F(a+\widehat{a}, b+\widehat{b}) - F(\widehat{a},\widehat{b}) = a^2\sigma_x^2 + (a\overline{x} + b)^2.$$

4) En déduire que F admet un unique minimum en $(\widehat{a}, \widehat{b})$.

Feuille de TP n° 3

Fonctions de plusieurs variables – Première partie

I Représentations de fonctions de deux variables

1) Courbe, surface

Définition. *Soit $n \in \mathbb{N}^*$. Soit $f : \mathbb{R}^n \longrightarrow \mathbb{R}$. On appelle graphe de f l'ensemble de $\mathbb{R}^{n+1}$ défini par*

$$\mathscr{C}_f = \{(x_1, x_2, \dots, x_{n+1}) \in \mathbb{R}^{n+1} \,|\, x_{n+1} = f(x_1, \dots, x_n)\}$$

- *Si $n = 1$, on retrouve la courbe d'une fonction de la variable réelle :*

$$\mathscr{C}_f = \{(x, y) \in \mathbb{R}^2 \,|\, y = f(x)\}$$

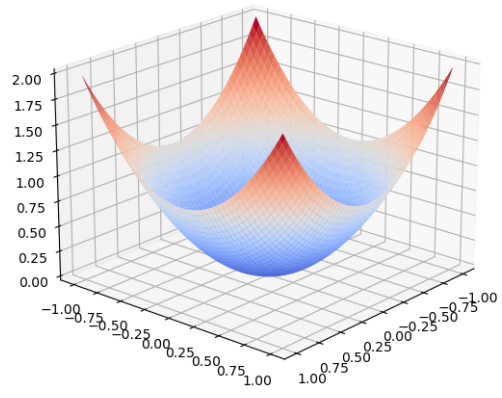
- *Si $n = 2$, alors la courbe de f est une surface tracée dans l'espace :*

$$\mathscr{C}_f = \{(x, y, z) \in \mathbb{R}^3 \,|\, z = f(x, y)\}.$$

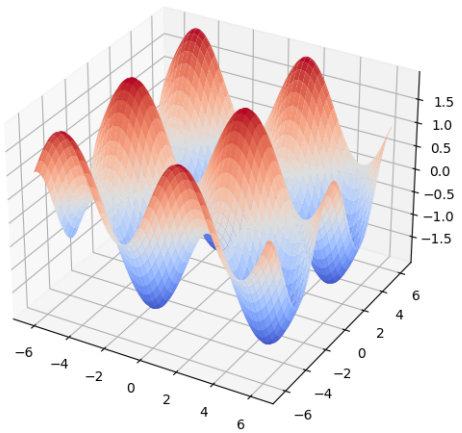
Remarque : Dans le cas où $n = 2$ (qui va nous intéresser dans ce chapitre), on peut voir les coordonnées x et y d'un point $(x, y) \in \mathbb{R}^2$ comme la latitude et la longitude et $z = f(x, y)$ comme l'altitude de ce point. Bien que ce soit une surface en 3 dimensions, on arrive à la représenter dans le plan en jouant sur la perspective et en utilisant des couleurs (les variations de couleurs représentent les variations d'altitude).

Exemples :

- *Courbe de $f : (x, y) \in \mathbb{R}^2 \longmapsto x^2 + y^2$:*



- *Courbe de $f : (x, y) \in \mathbb{R}^2 \longrightarrow \sin(x) + \cos(y)$:*



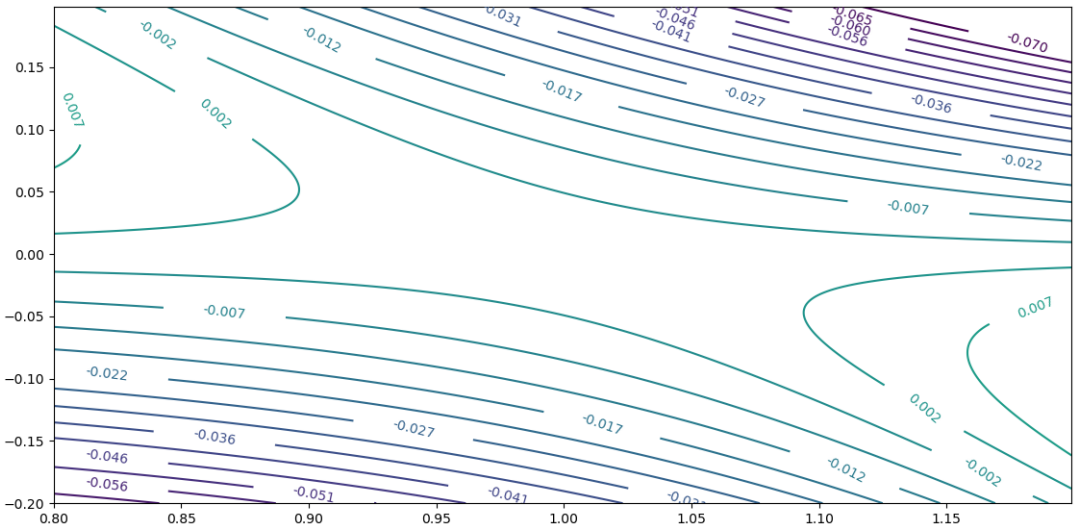
2) Lignes de niveaux

Définition. *Soit $n \in \mathbb{N}^*$. Soit $f : \mathbb{R}^n \longrightarrow \mathbb{R}$. Soit $\lambda \in \mathbb{R}$. On appelle ligne de niveau λ de f l'ensemble*

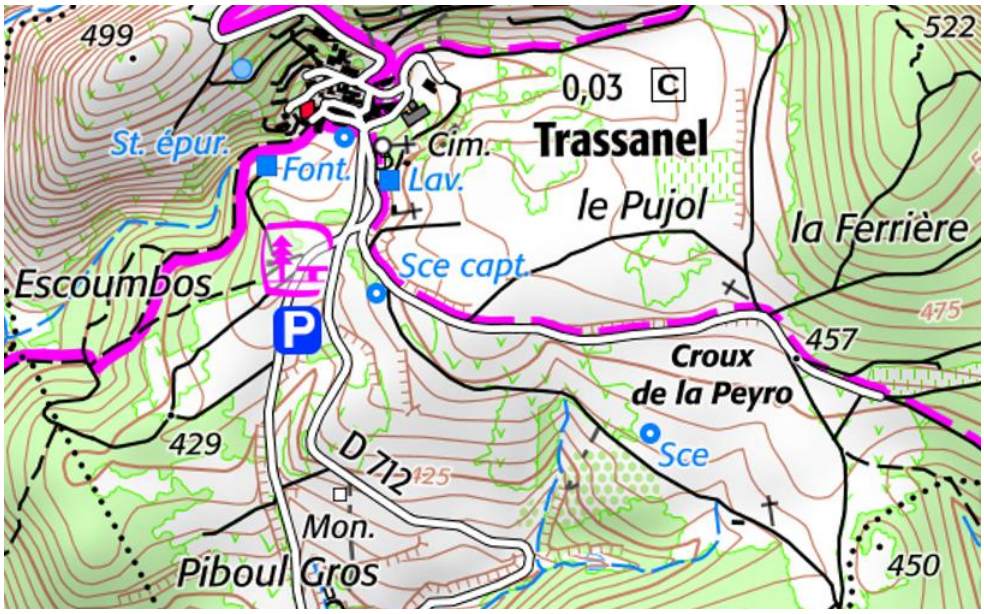
$$\{(x_1, \dots, x_n) \in \mathbb{R}^2 \,|\, f(x_1, \dots, x_n) = \lambda\}.$$

Remarque :

- Si $n = 2$, représenter des lignes de niveau consiste à se placer dans le plan, se donner une liste de niveaux puis, pour chacun des niveaux λ de la liste, représenter l'ensemble $\{(x, y) \in \mathbb{R}^2 \,|\, f(x, y) = \lambda\}$ en indiquant le niveau. Par exemple :



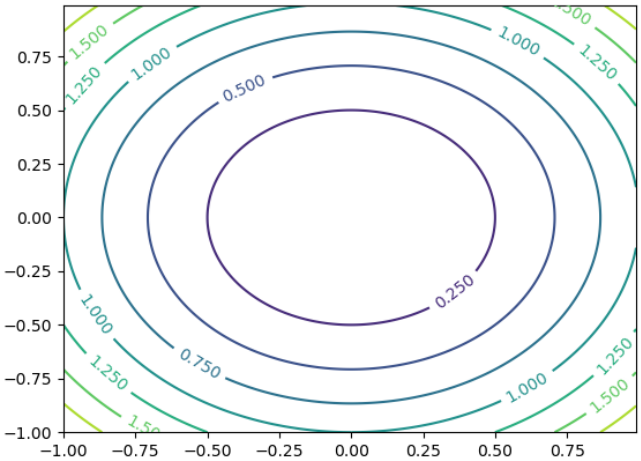
- Si $n = 2$ et si f est la fonction qui à la longitude et la latitude associe l'altitude, alors les lignes de niveau représentent les points qui sont à la même altitude : si on se promène sur une ligne de niveau, on ne monte pas et on ne descend pas : on reste au même niveau. Ce sont les lignes de niveau représentées sur les cartes topographiques. Par exemple :



Exemples :

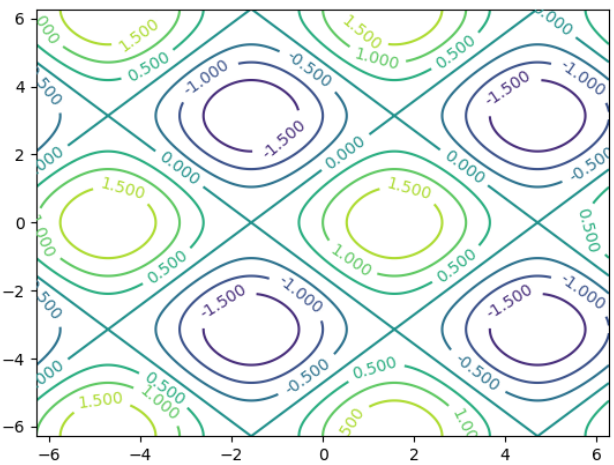
- *Lignes de niveau de*

$$f : (x, y) \in \mathbb{R}^2 \longmapsto x^2 + y^2 :$$

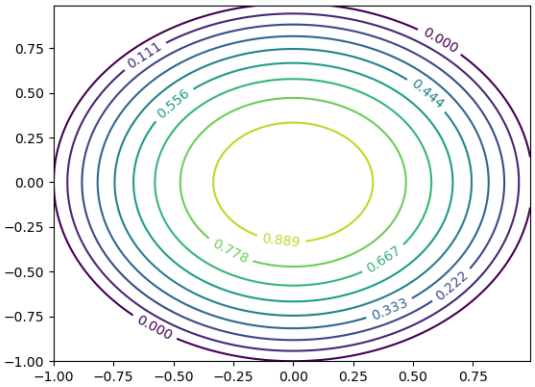


- *Lignes de niveau de*

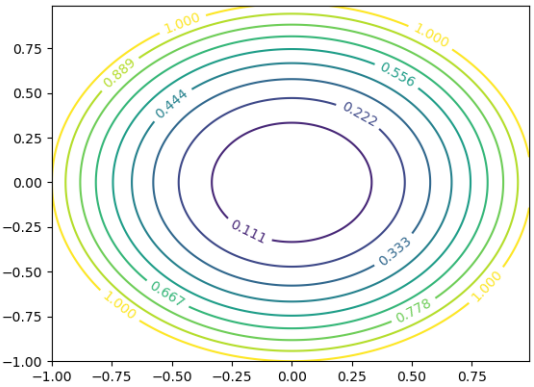
$$f : (x, y) \in \mathbb{R}^2 \longmapsto \sin(x) + \cos(y) :$$



Lecture de lignes de niveaux : Si au voisinage d'un point a, les lignes de niveaux ont tendance à se resserrer au fur et à mesure que le niveau augmente, cela traduit qu'il y a un maximum local en a :

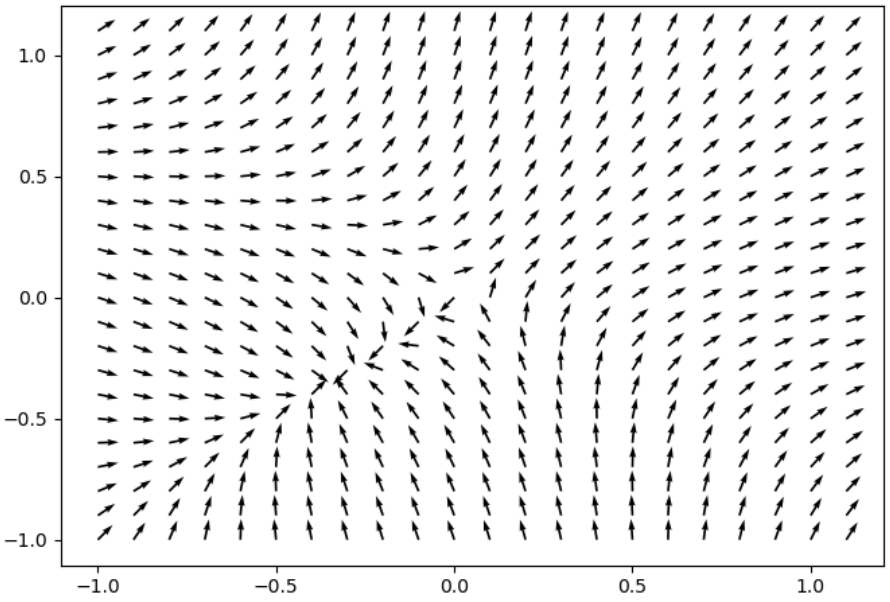


Si au voisinage d'un point a, les lignes de niveaux ont tendance à se resserrer au fur et à mesure que le niveau diminue, cela traduit qu'il y a un minimum local en a :



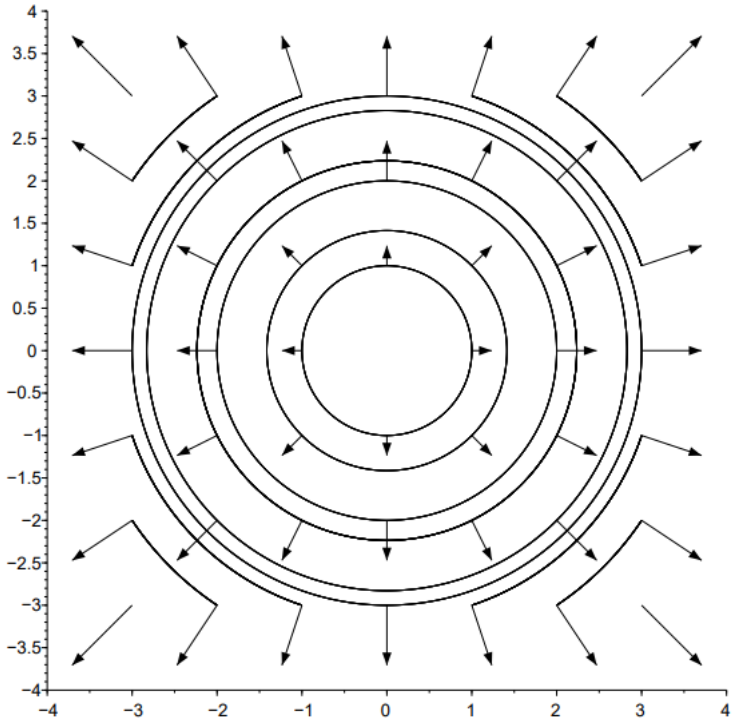
3) Champ de gradient

Si f est une fonction de classe C^1 de $\mathbb{R}^2$ dans $\mathbb{R}$, construire un champ de gradient de f revient à se donner plusieurs points du plan et à tracer une flèche correspondant au vecteur gradient de chacun de ses points. Par exemple :



Proposition. *Soit $f : \mathbb{R}^n \longrightarrow \mathbb{R}$ de classe C^1 sur $\mathbb{R}^n$. Soit $a \in \mathbb{R}^n$. Si le gradient $\nabla f(a) \neq 0$, alors :*

- *$\nabla f(a)$ est orthogonal à la tangente à la ligne de niveau $f(a)$ de f.*
- *$\nabla f(a)$ pointe vers les lignes de niveaux supérieurs (c'est-à-dire les lignes de niveau c pour tout $c > f(a)$).*



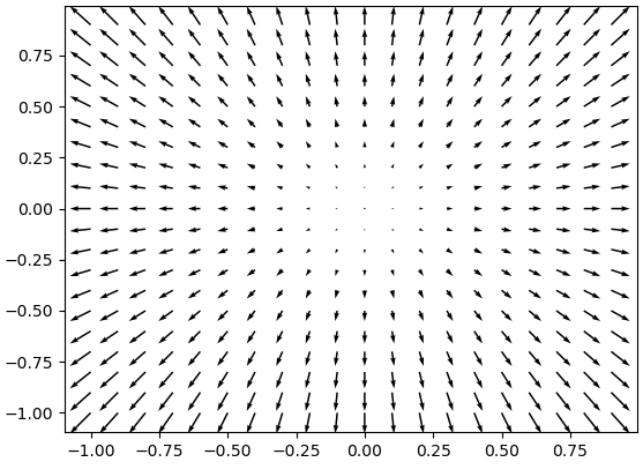
Esquisse de démonstration.

□

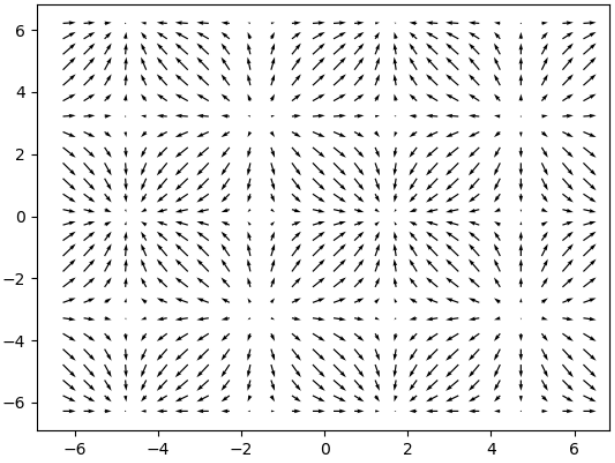
Démonstration dans le cas particulier où $f : (x, y) \in \mathbb{R}^2 \longmapsto 1 - x^2 - y^2$.
A faire sur une feuille séparée. □

Exemples :

- *Champ de gradient de* $f : (x, y) \in \mathbb{R}^2 \longmapsto x^2 + y^2$:

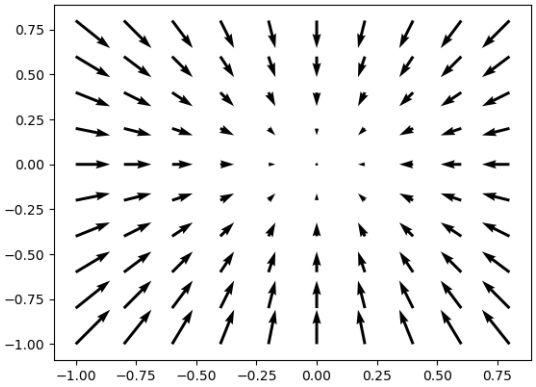


- *Champ de gradient de* $f : (x, y) \in \mathbb{R}^2 \longmapsto \sin(x) + \cos(y)$:

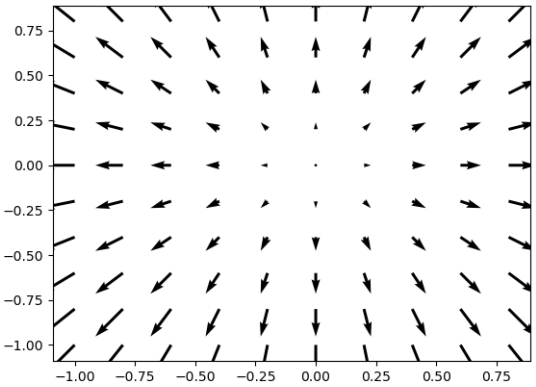


Lecture d'un champ de gradient : Dans un champ de gradient, on repère facilement les points critiques (en lesquels le gradient est nul par définition).

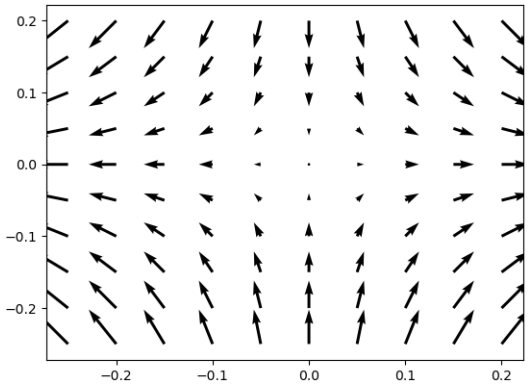
- Si au voisinage d'un point critique a, tous les gradients du champ pointent en direction de a, cela traduit qu'il y a un maximum local en a :



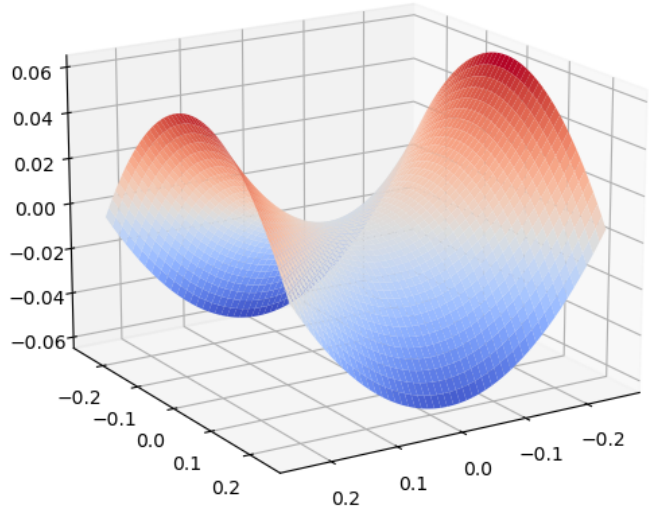
- Si au voisinage d'un point critique a tous les gradients du champ pointent dans la direction opposée à a, cela traduit qu'il y a un minimum local en a :



- Si au voisinage d'un point critique a, les gradients ne pointent pas tous en direction du point ou dans la direction opposée, alors cela traduit qu'il ne s'agit pas d'un extremum local :



On dit qu'il tel point est un point col ou un point selle, pour faire référence à une selle de cheval comme on peut le voir sur la surface ci-dessus (dont le champ de gradient ci-dessus correspond) :



Analyser le champ de gradient permet alors de conjecturer de domaines du plan dont la restriction de f à ces domaines présente en a un maximum local et d'autres domaines où c'est un minimum local. Dans l'exemple ci-dessus, on conjecture que sur $\{(x,y) \in \mathbb{R}^2 \,|\, y = 0\}$, f admet un minimum local (le long de cette droite, les gradients pointent dans la direction opposée de $a = (0,0)$) et sur $\{(x,y) \in \mathbb{R}^2 \,|\, x = 0\}$, f admet un maximum local (le long de cette droite, les gradients pointent vers $a = (0,0)$). Reste à le démontrer proprement.

II Représentation à l'aide de Python

La maitrise des fonctions Python permettant la représentation graphique de fonctions de plusieurs variables n'est pas exigible, conformément au programme.

La commande `np.meshgrid` est utilisée pour créer une grille rectangulaire à partir de deux tableaux unidimensionnels donnés représentant l'indexation cartésienne de la grille. En lien avec cette fonction, la commande

- `plt.contour` sert à tracer des lignes de niveau.
- `plt.quiver` sert à tracer des gradients.

Pour une représentation de surfaces, on utilise la commande `ax.plot_surface`, en ayant au préalable importé :

```
from mpl_toolkits.mplot3d import Axes3D
ax = Axes3D(plt.figure())
```

Rentrons dans les détails : soit f une fonction définie sur un domaine D de $\mathbb{R}^2$ et à valeurs réelles. Soient a, b, c, d des réels tels que $a < b$, $c < d$ et $[a\,;b] \times [c\,;d] \subset D$. On suppose que l'on implémente f, a, b, c, d en Python avec les variables `f,a,b,c,d` respectivement.

- Pour tracer la surface représentative de f sur $[a\,;b]\times[c\,;d]$, on utilise les commandes suivantes :

```
1  #On fait les importations requises.
2  import numpy as np
3  import matplotlib.pyplot as plt
4  from mpl_toolkits.mplot3d import Axes3D
5  ax = Axes3D(plt.figure())
6
7  #On discrétise les axes (ici avec un pas 0.01).
8  X = np.arange(a,b,0.01)
9  Y = np.arange(c,d,0.01)#On pourrait aussi utiliser np.linspace.
10
11 #On crée une grille rectangulaire à partir de X et Y.
12 X,Y=np.meshgrid(X,Y)
13
14 #On crée le tableau des images de X,Y par f.
15 f=np.vectorize(f)#On vectorise la fonction f si nécessaire.
16 Z=f(X,Y)
17
18 #On trace la surface.
19 ax.plot_surface(X,Y,Z)
20 plt.show()
```

Si on veut ajouter de la couleur à la surface, on peut remplacer la ligne 19 par :

```
1  from matplotlib.cm import coolwarm
2  ax.plot_surface(X, Y, Z,cmap=coolwarm)
```

- Pour tracer des lignes de niveaux, on commencer par lister les niveaux voulus (dans l'ordre croissant) dans une liste `L`. Ensuite on utilise les commandes suivantes :

```
1  #On fait les importations requises.
2  import numpy as np
3  import matplotlib.pyplot as plt
4
5  #On discrétise les axes (ici avec un pas 0.01).
6  X = np.arange(a,b,0.01)
7  Y = np.arange(c,d,0.01)#On pourrait aussi utiliser np.linspace.
8
9  #On crée une grille rectangulaire à partir de X et Y.
10 X,Y=np.meshgrid(X,Y)
11
12 #On crée le tableau des images de X,Y par f.
13 f=np.vectorize(f)#On vectorise la fonction f si nécessaire.
14 Z=f(X,Y)
15
16 #On trace les lignes de niveau.
17 plt.contour(X,Y,Z,L)
18 plt.show()
```

On peut aussi ajouter les indications des niveaux en remplaçant la ligne 17 par :

```
1  cp=plt.contour(X,Y,Z,L)
2  plt.clabel(cp,inline=True,fontsize=10)
```

Si on remplace `plt.contour(X,Y,Z,L)` par `plt.contour(X,Y,Z)`, les niveaux sont choisis par défaut.

- Pour tracer un champ de gradient, on utilise les commandes suivantes :

```
1  #On fait les importations requises.
2  import numpy as np
3  import matplotlib.pyplot as plt
4
5  #On discrétise les axes.
6  #Le pas (ici 0.1) ne doit pas être trop petit
7  #(sinon il y aura trop de flèches).
8  X = np.arange(a,b,0.1)
9  Y = np.arange(c,d,0.1)#On pourrait aussi utiliser np.linspace.
```

```
10
11 #On crée une grille rectangulaire à partir de X et Y.
12 X,Y=np.meshgrid(X,Y)
13
14 #On implémente le gradient de f.
15 def gradf(x,y):
16     return ..... , .....#gradient de f.
17 gradf=np.vectorize(gradf)
18 dx,dy=gradf(X,Y)
19
20 #On trace le champ de gradient.
21 plt.quiver(X,Y,dx,dy)
22 plt.show()
```

Exemple : *Prenons $a=-2$, $b=2$ et $f:(x,y)\longmapsto xye^{-(x^2+y^2)}$. Il s'agit d'une fonction de classe C^1 sur $\mathbb{R}^2$ et on a, pour tout $(x,y)\in\mathbb{R}^2$,*

$$\nabla f(x,y)=\left(y(1-2x^2)e^{-(x^2+y^2)},x(1-2y^2)e^{-(x^2+y^2)}\right).$$

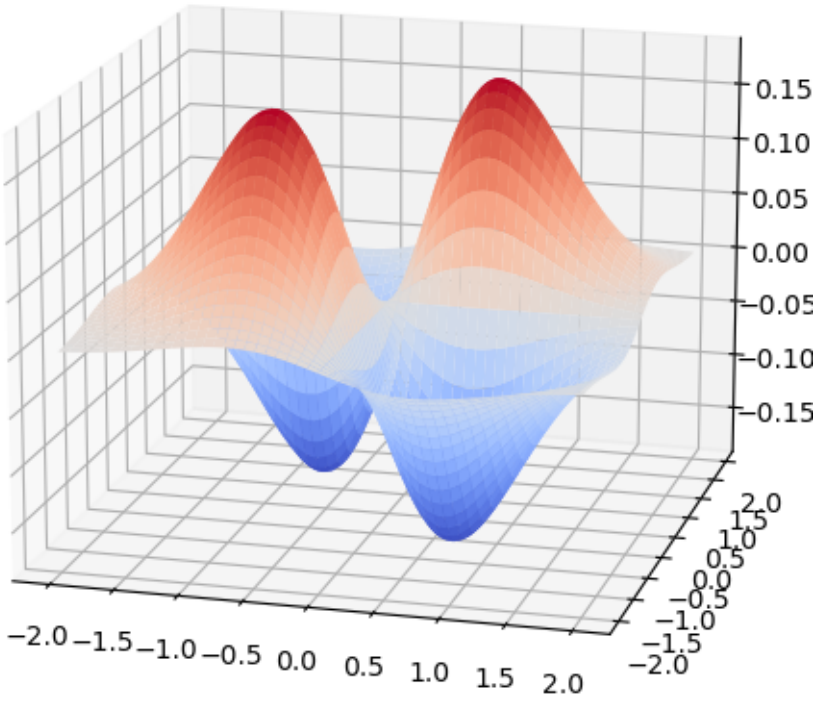
On implémente f et les deux dérivées partielles premières par :

```
1 def f(x,y) :
2     return x*y*np.exp(-(x**2+y**2))
3
4 def gradf(x,y):
5     g=np.exp(-(x**2+y**2))
6     return y*(1-2*x*x)*g,x*(1-2*y*y)*g
```

- *Éxécutons les commandes :*

```
1 from mpl_toolkits.mplot3d import Axes3D
2 ax=Axes3D(plt.figure())
3 X=np.arange(-2,2,0.01), Y=X
4 X,Y=np.meshgrid(X,Y)
5 Z=f(X,Y)
6 from matplotlib.cm import coolwarm
7 ax.plot_surface(X,Y,Z,cmap=coolwarm)
8 plt.show()
```

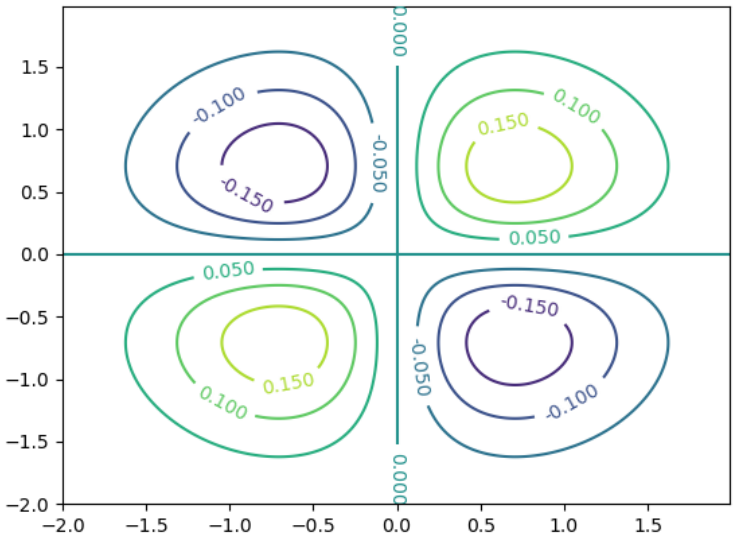
On obtient alors la surface suivante :



- *Éxécutons les commandes :*

```
X=np.arange(-2,2,0.01); Y=X
X,Y=np.meshgrid(X,Y)
Z=f(X,Y)
cp=plt.contour(X,Y,Z)
plt.clabel(cp,inline=True,fontsize=10)
plt.show()
```

On obtient alors les lignes de niveau suivantes :

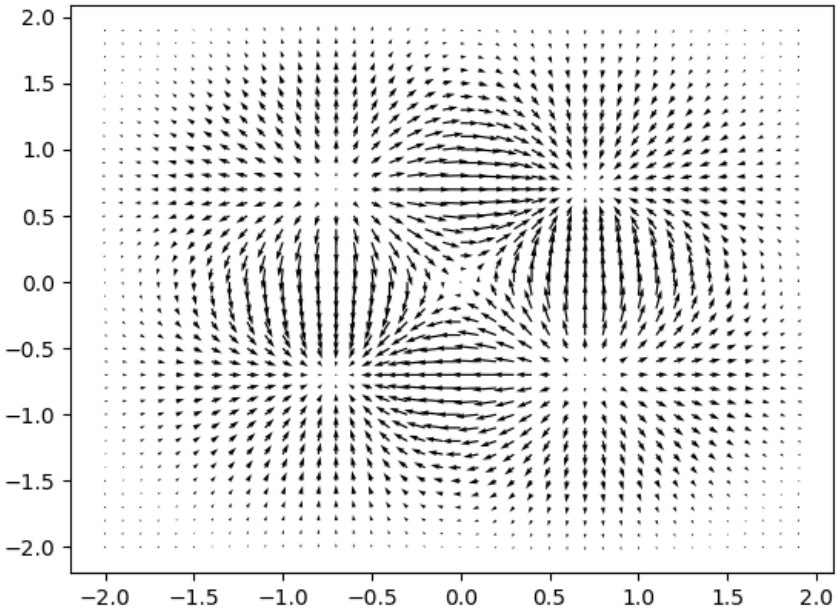


On remarque que, au voisinage des points de coordonnées approximatives $(0.7, 0.7)$ *et* $(-0.7, -0.7)$*, les lignes de niveaux se resserrent au fur et à mesure que le niveau augmente. Cela suggère qu'il y a un maximum local en ces points. Au voisinage des points de coordonnées approximatives* $(-0.7, 0.7)$ *et* $(0.7, -0.7)$*, les lignes de niveaux se resserrent au fur et à mesure que le niveau diminue. Cela suggère qu'il y a un minimum local en ces points.*

- *Éxécutons les commandes :*

```
X=np.arange(-2,2,0.1); Y=X
X,Y=np.meshgrid(X,Y)
dx,dy=gradf(X,Y)
plt.quiver(X,Y,dx,dy)
plt.show()
```

On obtient alors le champ de gradient suivant :



On remarque qu'il semble y avoir 5 points critiques. Localement, toutes les flèches pointent vers les points de coordonnées approximatives $(0.7, 0.7)$ *et* $(-0.7, -0.7)$*. Cela suggère qu'il y a un maximum local en ces points. Localement, toutes les flèches partent des points de coordonnées approximatives* $(-0.7, 0.7)$ *et* $(0.7, -0.7)$*. Cela suggère qu'il y a un minimum local en ces points. En revanche, en le point de coordonnées* $(0, 0)$*, il y a des flèches qui pointent et des flèches qui partent localement. Cela suggère que c'est un point selle.*

III Exercices

Exercice 1. (★★) Soit $f : (x, y) \in \mathbb{R}^2 \longmapsto xye^{-(x^2+y^2)}$. Il s'agit de la fonction de l'exemple ci-dessus.

1) Montrer que f est de classe C^1 sur $\mathbb{R}^2$ et déterminer ses points critiques.
2) Montrer que $(0, 0)$ est un point selle.
3) a) Étudier les extrema de la fonction $t \in \mathbb{R}_+ \longmapsto te^{-t}$.
   b) Justifier que, pour tout $(x, y) \in \mathbb{R}^2$, $xy \leqslant \dfrac{x^2 + y^2}{2}$.
   c) En déduire que f admet des extrema globaux en les quatre autre points critiques.

Exercice 2. (★) à (★★★)

1) Pour les fonctions suivantes, représenter la surface, des lignes de niveaux et un champ de gradient puis conjecturer la présence de points critiques et leur nature (point selle, point en lequel il y a un maximum/minimum local/global) :
   a) $(x, y) \longmapsto x^3 + xy + y^3$,
   b) $(x, y) \longmapsto \sin(x) + y^2 - 2y + 1$,
   c) $(x, y) \longmapsto xy - x^2y - xy^2$,
   d) $(x, y) \longmapsto x^4 + y^4 - 2(x - y)^2$,
   e) $(x, y) \longmapsto \dfrac{-3y}{x^2 + y^2 + 1}$,
   f) $(x, y) \longmapsto e^x(x + y^2 + e^x)$.
2) Démontrer ensuite ces conjectures.
   Pour la question 1f, on montrera au préalable que $\varphi : t \longmapsto 1 + t + 2e^t$ *s'annule en un unique point* α *et que* $\alpha \in [-2\,;-1]$.

Feuille de TP n° 4

Convergences et approximations de variables aléatoires

I Retour sur la loi faible des grands nombres

1) Approximation de probabilité ou d'espérance

Soient $(X_n)_{n\geqslant 1}$ une suite de variables aléatoires indépendantes admettant une espérance m et une variance. La loi faible des grands nombres assure que

$$\overline{X}_n = \frac{1}{n}\sum_{k=1}^{n} X_i \xrightarrow[n\to+\infty]{\mathbb{P}} m,$$

c'est-à-dire

$$\forall \varepsilon > 0, \qquad \mathbb{P}(|\overline{X}_n - m| \geqslant \varepsilon) \xrightarrow[n\to+\infty]{} 0.$$

Ainsi :

- Pour n assez grand (on peut même contrôler l'erreur grâce à l'inégalité de Bienaymé-Tchebychev ou le TCL, cf. paragraphe III.1), $\overline{X}_n$ est une approximation de $m = \mathbb{E}(X_1)$.
- On répète un grand nombre de fois une expérience indépendamment et on se demande le nombre de fois qu'un certain phénomène est réalisé. Plus précisément, pour tout $k \in \mathbb{N}^*$, notons A_k l'événement « l'événement qui nous intéresse est réalisé lors de la k-ième expérience ». On prend alors $X_k = \mathbb{1}_{A_k}$ pour tout $k \in \mathbb{N}^*$ et on a :

$$\frac{\mathrm{card}(\{k \in \mathbb{N}^* \mid A_k \text{ est réalisé}\})}{n} = \frac{1}{n}\sum_{k=1}^{n} \mathbb{1}_{A_k} \xrightarrow[n\to+\infty]{\mathbb{P}} \mathbb{E}(\mathbb{1}_{A_1}) = \mathbb{P}(A_1).$$

  Autrement dit, lorsqu'on réalise un grand nombre de fois une expérience indépendamment, la proportion de fois qu'un événement est réalisé lors de ces expériences, est une approximation de la probabilité de cet événement.

Nous avons vu de nombreux exemples d'application l'an passé et dans les TP n° 0 et 1.

Supposons que l'on ait implémenté en Python l'espérance m dans la variable `m` et que la liste en Python `X` contienne des réalisations $x_1, \dots, x_n$ des variables aléatoires $X_1, \dots, X_n$, avec $n \in \mathbb{N}^*$. La commande `np.cumsum(X)` renvoie donc un vecteur contentant

$$x_1, x_1 + x_2, x_1 + x_2 + x_3, \dots, x_1 + x_2 + \cdots + x_n$$

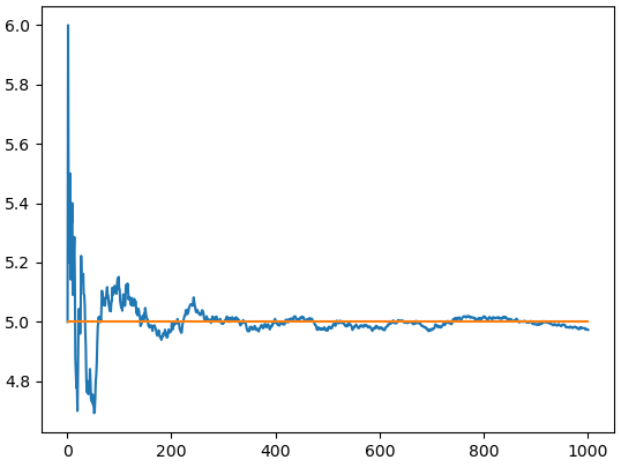
et donc `np.cumsum(X)/range(1,len(X)+1)` renvoie donc un vecteur contentant

$$x_1, \frac{x_1 + x_2}{2}, \frac{x_1 + x_2 + x_3}{3}, \dots, \frac{x_1 + x_2 + \cdots + x_n}{n}.$$

Ainsi le script suivant permet d'illustrer graphiquement la loi faible des grands nombres :

```python
import numpy as np
import matplotlib.pyplot as plt
n=len(X)
Xbar=np.cumsum(X)/range(1,n+1)
plt.plot(range(1,n+1),Xbar)
plt.plot([1,n],[m,m])
plt.show()
```

Ci contre, un exemple avec 1000 variables aléatoires indépendantes de loi $\mathcal{B}(20,1/4)$ donc avec `X=rd.binomial(20,1/4,1000)` et `m=5`.



Remarque : Pourquoi la loi *faible* des grands nombres ? Et bien parce qu'il existe une loi *forte* des grands nombres. On peut montrer (mais c'est bien plus difficile que pour la loi faible) que, si $(X_n)_{n\geqslant 1}$ est une suite de variable aléatoire définies sur un espace probabilisé $(\Omega,\mathcal{A},\mathbb{P})$, qui sont indépendantes et de même loi admettant une espérance m, alors

$$\mathbb{P}\left(\left\{\omega\in\Omega\,\middle|\,\overline{X}_n(\omega)\xrightarrow[n\to+\infty]{}m\right\}\right)=1.$$

Autrement dit, presque sûrement, $\overline{X}_n\xrightarrow[n\to+\infty]{}m$. On parle de convergence presque sûre (attention ce terme est hors-programme).

2) Histogramme et fonction de répartition empirique

Souvent cette année et l'an passé, nous avons superposé des histogrammes à des diagrammes à barres ou à des courbes de densité pour conjecturer la loi d'une variable aléatoire. Il s'agit d'applications de la loi des grands nombres. Démontrons-cela :

Soit $(X_1,\dots,X_n)$ des variables aléatoires indépendantes et de même loi qu'une variable aléatoire X (typiquement on connaît des réalisations de $X_1(\omega),\dots,X_n(\omega)$ qui peuvent donc être vues comme n réalisations indépendantes de X).

- Cas où X est discrète. On construit alors un histogramme ou plutôt un diagramme à barre empirique. Pour tout $x\in X(\Omega)$, on trace une barre dont la hauteur est la proportion de réalisations qui valent x. Il s'agit de

$$\frac{1}{n}\sum_{k=1}^{n}\mathbb{1}_{[X_i=x]}.$$

  Par la loi des grands nombres, cette quantité converge en probabilité vers $\mathbb{E}(\mathbb{1}_{[X=x]})=\mathbb{P}(X=x)$ quand n tend vers $+\infty$. Cela explique bien pourquoi, lorsque n est grand, le diagramme empirique se superpose au diagramme théorique.
- Cas où X est à densité. Notons F la fonction de répartition de X et f une densité de X. On sépare les observations en classes. Supposons que l'une des classe corresponde aux valeurs entre a et b. La barre de l'histogramme renormalisé correspondant à cette classe a pour hauteur

$$\frac{1}{n(b-a)}\sum_{k=1}^{n}\mathbb{1}_{[a<X_i\leqslant b]}.$$

  Par la loi des grands nombres, quand n tend vers $+\infty$, cette quantité converge en probabilité vers

$$\frac{\mathbb{E}(\mathbb{1}_{[a<X_i\leqslant b]})}{b-a}=\frac{\mathbb{P}(a<X\leqslant b)}{b-a}=\frac{F(b)-F(a)}{b-a}.$$

  Remarquons alors que, si F est dérivable en a et $F'(a)=f(a)$, on a $\dfrac{F(b)-F(a)}{b-a}\xrightarrow[b\to a]{}F'(a)=f(a)$. Ainsi lorsque n est grand et b proche de a (c'est le cas lorsqu'il y a beaucoup de classes), cela explique pourquoi la courbe de la densité épouse bien l'histogramme.

Dans l'exercice 8 du TP n° 1, nous avons également défini la fonction de répartition empirique d'un échantillon et démontré que, ponctuellement, elle converge en probabilité vers la fonction de répartition théorique.

3) Méthode de Monte-Carlo pour le calcul approché d'intégrales

La méthode de Monte-Carlo est une méthode permettant de calculer des valeurs numériques approchées en utilisant des variables aléatoires. On a déjà vu des exemples dans le paragraphe 1 de la partie II (calcul approché de probabilités d'événements ou d'espérances de variables aléatoires). Dans ce paragraphe, nous allons voir comment la loi faible des grands nombres permet de justifier un procédé de calcul approché d'intégrales.

Soient a et b des réels tels que $a < b$. Soit f une fonction continue sur $]a\,;b[$ et telle que $\displaystyle\int_a^b f(t)\,dt$ converge absolument. Le théorème de transfert assure que, si U est une variable aléatoire de loi $\mathcal{U}(]a\,;b[)$, alors

$$\mathbb{E}(f(U)) = \int_a^b f(x)\frac{dx}{b-a}.$$

et donc

$$\int_a^b f(x)\,dx = (b-a)\mathbb{E}(f(U)).$$

Notons que l'existence de l'espérance de $f(U)$ est immédiate (car l'intégrale de $x \longmapsto |f(x)|\,\dfrac{1}{b-a}\mathbb{1}_{]a\,;b[}(x)$ sur $]a\,;b[$ converge et que cette fonction est nulle en dehors de $]a\,;b[$). Il en est de même pour la variance de $f(U)$ si on suppose de plus que $\displaystyle\int_a^b (f(t))^2\,dt$ converge.

Si on se donne une suite $(U_n)_{n\in\mathbb{N}^*}$ de variables aléatoires indépendantes de loi uniforme sur $]a\,;b[$, alors $(f(U_n))_{n\in\mathbb{N}^*}$ est une suite de variables aléatoires indépendantes (par théorème des coalitions) et de même loi. Elles admettent une espérance et une variance donc la loi faible des grands nombres entraîne que

$$\frac{1}{n}\sum_{i=1}^{n} f(U_i) \xrightarrow[n\to+\infty]{\mathbb{P}} \mathbb{E}(f(U)).$$

Ainsi (puisque $x \longmapsto (b-a)x$ est continue sur $\mathbb{R}$), on obtient :

$$\frac{b-a}{n}\sum_{i=1}^{n} f(U_i) \xrightarrow[n\to+\infty]{\mathbb{P}} \int_a^b f(x)\,dx.$$

On en déduit que $\dfrac{b-a}{n}\displaystyle\sum_{i=1}^{n} f(U_i)$ est une approximation de $\displaystyle\int_a^b f(x)\,dx$.

Néanmoins la loi faible des grands nombres ne permet pas de mesurer la rapidité de la convergence. On assortit généralement le procédé ci-dessus d'une quantification des garanties d'approximation fournie par le Théorème Central Limite (cf. paragraphe 3 du III). On peut aussi utiliser l'inégalité de Bienaymé-Tchebychev (mais elle est beaucoup moins précise on trouvera une valeur de n garantissant une bonne approximation qui est bien exagérée). Supposons que la variance de $f(U)$ est majorée, disons par un réel M que l'on connaît. On a alors

$$\mathbb{P}\left(\left|\frac{b-a}{n}\sum_{i=1}^{n} f(U_i) - \int_a^b f(t)\,dt\right| \geqslant \varepsilon\right) \leqslant (b-a)^2\frac{M}{n\varepsilon^2}.$$

En effet :

Ainsi :

- À $n \in \mathbb{N}^*$ et $\alpha \in]0\,;1[$ fixés, si on prend $\varepsilon = \dfrac{(b-a)\sqrt{M}}{\sqrt{\alpha n}}$, alors $(b-a)^2\dfrac{M}{n\varepsilon^2} \leqslant \alpha$ et donc $\dfrac{b-a}{n}\displaystyle\sum_{i=1}^{n} f(U_i)$ est une approximation de $\displaystyle\int_a^b f(x)\,dx$ à $\dfrac{(b-a)\sqrt{M}}{\sqrt{\alpha n}}$ près.
- Inversement, si on se donne $\varepsilon > 0$ et $\alpha \in]0\,;1[$, on sait que, avec un **niveau de confiance**[1] de $1-\alpha$, une approximation $\dfrac{b-a}{n}\displaystyle\sum_{i=1}^{n} f(U_i)$ est une approximation de $\displaystyle\int_a^b f(x)\,dx$ à ε près dès que $n = \boxed{}$.
  En effet :

Exercice 1. Calculer des valeurs approchées des intégrales suivantes à 10^{-4} près (avec une niveau de confiance $1-\alpha$) avec un niveau de confiance de 95% et les comparer à les approximation de leurs valeurs exactes[2] fournies par Python :

$$\int_0^1 \frac{dt}{1+t}, \qquad \int_0^2 \frac{dt}{1+t}, \qquad \int_0^1 \frac{dt}{1+t^2}, \qquad \int_{-1}^1 \sqrt{1-t^2}\,dt, \qquad \int_0^1 \ln(t)\,dt.$$

Exercice 2 – Le cas d'une intégrale sur $\mathbb{R}_+$ ou $\mathbb{R}$. (★★) Soit f une fonction définie sur $\mathbb{R}_+$ et telle que $\displaystyle\int_0^{+\infty} f(x)\,dx$ converge absolument.

1) a) Justifier que
$$\int_0^{+\infty} f(x)\,dx = \int_0^1 f(-\ln(t))\,\frac{dt}{t}.$$
   b) A l'aide de la méthode de Monte-Carlo, déterminer un algorithme permettant de calculer une valeur approchée de $\displaystyle\int_0^{+\infty} f(x)\,dx$.
2) En remarquant que $\displaystyle\int_0^{+\infty} f(x)\,dx = \int_0^{+\infty} f(x)e^x \times e^{-x}\,dx$, proposer une autre méthode permettant de calculer une valeur approchée de $\displaystyle\int_0^{+\infty} f(x)\,dx$.
3) Tester ces deux méthodes avec Python pour calculer $\displaystyle\int_0^{+\infty} e^{-x^2/2}\,dx = \sqrt{\frac{\pi}{2}}$. Comparer avec la vraie valeur.
4) Comment faire pour approcher $\displaystyle\int_{-\infty}^{+\infty} f(x)\,dx$ avec Python lorsque celle-ci converge absolument ?
   On pourra s'aider d'une bijection de $]0\,;1[$ dans $\mathbb{R}$ (comme par exemple $t \longmapsto \ln(1/t-1)$, $t \longmapsto \ln(-\ln(t))$ ou $t \longmapsto \tan(\pi(1/2-t))$), ou encore couper l'intégrale en deux, ou utiliser une loi Normale.

1. C'est-à-dire avec probabilité supérieure ou égale à $1-\alpha$.
2. Il faut savoir les calculer bien sûr. Elles font respectivement $\ln(2)$, $\ln(3)$, $\frac{\pi}{4}$, $\frac{\pi}{2}$, -1.

Exercice 3 – Estimation de la fonction de répartition. (★★)

1) Soit X une variable aléatoire réelle ne prenant que des valeurs positives. En utilisant la méthode de Monte-Carlo, proposer une fonction qui prend en entrée x et qui calcule une valeur approchée de $F_X(x)$, la fonction de répartition de X en x.
2) Utiliser cette fonction pour illustrer (en superposant la fonction de répartition estimée avec la fonction de répartition théorique) le fait que :
   - $X_1 + X_2 \hookrightarrow \mathcal{B}(n_1 + n_2, p)$ lorsque X_1 et X_2 sont indépendantes de lois $\mathcal{B}(n_1, p)$ et $\mathcal{B}(n_2, p)$ respectivement.
   - $X_1 + X_2 \hookrightarrow \mathcal{P}(a_1 + a_2)$ lorsque X_1 et X_2 sont indépendantes de lois $\mathcal{P}(a_1)$ et $\mathcal{P}(a_2)$ respectivement.
   - $X_1 + X_2 \hookrightarrow \mathcal{N}(m_1 + m_2, \sigma_1^2 + \sigma_2^2)$ lorsque X_1 et X_2 sont indépendantes de lois $\mathcal{N}(m_1, \sigma_1^2)$ et $\mathcal{N}(m_2, \sigma_2^2)$ respectivement.
   - $X_1 + X_2 + \cdots + X_n \hookrightarrow \gamma(n)$ lorsque $X_1, \ldots, X_n$ sont indépendantes de loi $\mathcal{E}(1)$.
     Pour la fonction de répartition en un réel x de la loi $\gamma(n)$, on utilisera `sp.gdtr(1,n,x)` après `import scipy.special as sp`.

Exercice 4 – Produit de deux variables aléatoires de loi uniforme. (★★) Soient X et Y deux variables aléatoires indépendantes de même loi uniforme sur $]0\,;1[$. On note $Z = XY$.

1) Écrire une fonction Python qui prend en entrée t et qui calcule une valeur approchée de $F_Z(t)$ à l'aide de la méthode de Monte-Carlo.
2) Avec Python, représenter graphiquement la courbe de F_Z sur $[-1/2\,;3/2]$. Lui superposer la courbe de la fonction $t \longmapsto t(1 - \ln(t))\mathbb{1}_{]0\,;1[}(t) + \mathbb{1}_{[1\,;+\infty[}(t)$. Que conjecturer ?
3) Montrer la conjecture.
   On commencera par calculer la fonction de répartition de $-\ln(Z)$.

II Convergence en loi

Exercice 5 – D'après EDHEC 2017 et les oraux ESCP 2003. (★★) Soit $(X_n)_{n\geqslant 1}$ une suite de variables aléatoires définies sur un même espace probabilisé $(\Omega, \mathcal{A}, \mathscr{P})$ qui sont indépendantes et de même loi $\mathcal{U}([0\,;1])$. Pour tout $n \in \mathbb{N}^*$, on note $M_n = \max\{X_1, \ldots, X_n\}$. Plus rigoureusement

$$\forall \omega \in \Omega, \qquad M_n(\omega) = \max\{X_1(\omega), \ldots, X_n(\omega)\}.$$

On admet que M_n est que variable aléatoire. Enfin, on pose $Y_n = n(1 - M_n)$.

1) a) Soit $n \in \mathbb{N}^*$. Calculer F_{M_n} puis montrer que M_n est une variable aléatoire à densité.
   b) Établir l'existence de $\mathbb{E}(M_n)$ et $\mathbb{E}(M_n^2)$. Les calculer.
   c) Soit $\varepsilon > 0$. Donner un majorant (ne dépendant que de n et de ε) de $\mathbb{P}\big((M_n - 1)^2 \geqslant \varepsilon^2\big)$.
   d) Conclure que $\lim\limits_{n\to+\infty} \mathbb{P}(|M_n - 1| \geqslant \varepsilon) = 0$. Que signifie ce résultat ?
2) a) Écrire une fonction en Python appelée `SimulM` qui prend en entrée n et qui renvoie une simulation de M_n.
   b) Pour $n = 1000$, construire l'histogramme renormalisé de 10000 réalisations de Y_n avec 15 classes. Lui superposer la densité d'une variable aléatoire de loi $\mathcal{E}(1)$. Que peut-on conjecturer ?
   c) Pour tout $n \in \mathbb{N}^*$, calculer F_{Y_n}.
   d) Pour tout $x \in \mathbb{R}_+^*$, calculer $\lim\limits_{n\to+\infty} F_{Y_n}(x)$.
   e) Montrer la conjecture de la question 2b.

3) On considère les instruction suivantes en Python :

```
import numpy as np
import numpy.random as rd

def SimulZ(n):
    U=n*rd.random(n)
    return np.min(U)

m=5000; n=100; T=np.zeros(m)
for k in range(m):
    T[k]=SimulZ(n)
print(np.sum(T<=1)/m)
```

a) Montrer que la fonction `SimulZ`, de paramètre $n \in \mathbb{N}^*$ simule une variable aléatoire de même loi que Y_n.

b) En exécutant 10 fois le programme précédent, l'écran affiche les résultats suivants :

```
0.6418 0.6334 0.6452 0.629 0.6212 0.626 0.641 0.6282 0.63 0.6308
```

La valeur affichée à l'écran est une valeur approchée d'un nombre p. Préciser, à l'aide des questions précédentes, la valeur exacte de p.

Exercice 6 – Convergence vers la loi de Gumbel. (★★)

1) Montrer que $f : x \longmapsto \exp\left(-(t+e^{-t})\right)$ est une densité de probabilité. On dit qu'une variable aléatoire qui admet pour densité f suit une loi de Gumbel.

2) Soit U une variable aléatoire de loi $\mathcal{U}(]0\,;1[)$. On admet que $X = \ln\left(\dfrac{U}{1-U}\right)$ est une variable aléatoire. Calculer sa fonction de répartition et montrer que c'est une variable à densité.

3) Soient $(X_n)_{n\geqslant 1}$ une suite de variables aléatoires définies sur un même espace $(\Omega, \mathcal{A}, \mathscr{P})$ qui sont indépendantes et de même loi que X. Pour tout $n \in \mathbb{N}^*$, on note $Y_n = \max\{X_1, \dots, X_n\}$. Plus rigoureusement

$$\forall \omega \in \Omega, \qquad Y_n(\omega) = \max\{X_1(\omega), \dots, X_n(\omega)\}.$$

On pose $Z_n = Y_n - \ln(n)$.

a) Écrire une fonction en Python qui prend en entrée n et qui simule Y_n.

b) Pour $n = 1000$, tracer l'histogramme renormalisé de 10000 réalisations de Y_n avec 20 classes. Lui superposer le graphe de f. Que peut-on conjecturer ?

4) Montrer que la conjecture de la question précédente.

Exercice 7 – Simulation d'une loi de Poisson. (★★★)

1) À l'aide de l'approximation Binomiale/Poisson, écrire une fonction Python, appelée `Simul_1`, qui prend en entrée $a > 0$ et qui simule une variable aléatoire de loi de Poisson de paramètre a à l'aide d'une variable aléatoire de loi binomiale de premier paramètre 1000 en utilisant `rd.binomial()`.

2) On a vu dans l'exercice 11 du TP n° 1 que, si $a > 0$ et si $(U_i)_{i\geqslant 1}$ est une suite de variables aléatoires indépendantes et de même loi uniforme sur $]0\,;1[$, alors

$$Y = \min\{n \in \mathbb{N} \,|\, U_1 U_2 \dots U_{n+1} \leqslant e^{-a}\}$$

suit une loi de Poisson de paramètre a. Écrire une fonction Python, appelée `Simul_2`, qui prend en entrée $a > 0$ et qui simule une variable aléatoire loi de Poisson de paramètre a à l'aide de ce dernier résultat.

3) Soit $a > 0$. Soit U une variable aléatoire de loi uniforme sur $]0\,;1[$. Pour tout $k \in \mathbb{N}$, notons $f_k = \displaystyle\sum_{k=0}^{n} e^{-a}\frac{a^k}{k!}$.

a) Calculer $\mathbb{P}(f_{k-1} < U \leqslant f_k)$ pour tout $k \in \mathbb{N}^*$.

b) En déduire que $Y = \min\{k \in \mathbb{N} \,|\, U \leqslant f_k\}$ suit une loi de Poisson de paramètre a.

c) Compléter alors le programme suivant pour qu'il prenne en entrée $a > 0$ et simule une variable aléatoire de loi de Poisson de paramètre a à l'aide de ce dernier résultat.

```
import ......................
import ......................
def Simul_3(a):
    p=np.exp(-a)
    f=p
    k=0
    U=.........................
    ...........................
        k=k+1
        p=.....................
        f=f+p
    return ...................
```

4) Si on importe la librairie `time`, alors la commande `time.time()` renvoie un flottant contenant l'heure en secondes (avec une certaine précision). Si on exécute cette commande avant et après un script, l'écart entre les deux valeurs donne alors le temps d'exécution du script. Recopier et exécuter le script suivant :

```
import time
N=100000
a=4
s1=time.time(); X1=[Simul_1(a) for k in range(N)]; e1=time.time()
s2=time.time(); X2=[Simul_2(a) for k in range(N)]; e2=time.time()
s3=time.time(); X3=[Simul_3(a) for k in range(N)]; e3=time.time()
s4=time.time(); X4=[rd.poisson(a) for k in range(N)]; e4=time.time()
print(e1-s1,e2-s2,e3-s3,e4-s4)
M=np.max([np.max(X1),np.max(X2),np.max(X3),np.max(X4)])
valeurs=[i-0.5 for i in range(M+1)]
plt.hist([X1,X2,X3,X4],bins=valeurs,density=True)
plt.show()
```

Commenter.

III Le Théorème Central Limite

1) Énoncé

Soient $(X_n)_{n\geqslant 1}$ une suite de variables aléatoires indépendantes admettant une espérance m et une variance $\sigma^2 > 0$. Pour tout $n \in \mathbb{N}^*$, notons $\overline{X}_n = \dfrac{1}{n}\displaystyle\sum_{k=1}^{n} X_i$. Le Théorème Central Limite assure que, si Z désigne une variable aléatoire de loi $\mathcal{N}(0,1)$, alors

$$\overline{X}_n^* = \sqrt{n}\,\frac{\overline{X}_n - m}{\sigma} \xrightarrow[n\to+\infty]{\mathscr{L}} Z,$$

c'est-à-dire, pour tout $x \in \mathbb{R}$,

$$\mathbb{P}(\overline{X}_n^* \leqslant x) \xrightarrow[n\to+\infty]{} \Phi(x) = \int_{-\infty}^{x} e^{-t^2/2}\frac{dt}{\sqrt{2\pi}}$$

ou encore, pour tous $(a,b) \in \mathbb{R}^2$,

$$\mathbb{P}(a \leqslant \overline{X}_n^* \leqslant b) \xrightarrow[n\to+\infty]{} \int_a^b e^{-t^2/2}\frac{dt}{\sqrt{2\pi}}.$$

2) Représentation du TCL

a) Convergence des fonctions de répartition

Exercice 8. (★★)

1) Soit $p \in]0\,;1[$. Soit $(X_n)_{n\geqslant 1}$ une suite de variables aléatoires indépendantes de loi $\mathcal{B}(p)$. Notons m et σ^2 l'espérance et la variance d'une loi $\mathcal{B}(p)$.
   a) Justifier que, pour tous $n \in \mathbb{N}^*$ et $x \in \mathbb{R}$,
   $$\mathbb{P}\left(\sqrt{n}\,\frac{\overline{X}_n - m}{\sigma} \leqslant x\right) = F_{n,p}\left(x\sqrt{np(1-p)} + np\right),$$
   où $F_{n,p}$ désigne la fonction de répartition d'une variable aléatoire de loi $\mathcal{B}(n,p)$.
   b) Écrire une fonction Python qui prend en entrée n, p et un réel t et qui renvoie $F_{n,p}(t)$.
   c) Avec Python, tracer la courbe représentative de $x \longmapsto F_{n,p}\left(x\sqrt{np(1-p)} + np\right)$ sur l'intervalle $[-4\,;4]$ avec $n = 1000$ et $p = 0,3$.
   d) Lui superposer la courbe de Φ, la fonction de répartition d'une variable aléatoire de loi $\mathcal{N}(0,1)$. Commenter.
   On rappelle que, si on réalise l'importation suivante : `import scipy.special as sp`*, alors la commande* `sp.ndtr` *est une fonction Python qui implémente* Φ*.*
2) Soit $a \in \mathbb{R}_+^*$. Soit $(X_n)_{n\geqslant 1}$ une suite de variables aléatoires indépendantes de loi $\mathcal{P}(a)$. Notons m et σ^2 l'espérance et la variance d'une loi $\mathcal{P}(a)$.
   a) Justifier que, pour tous $n \in \mathbb{N}^*$ et $x \in \mathbb{R}$,
   $$\mathbb{P}\left(\sqrt{n}\,\frac{\overline{X}_n - m}{\sigma} \leqslant x\right) = F_{na}\left(x\sqrt{na} + na\right),$$
   où, pour tout $b \in \mathbb{R}_+^*$, F_b désigne la fonction de répartition d'une variable aléatoire de loi $\mathcal{P}(b)$.
   b) Écrire une fonction Python qui prend en entrée b et un réel t et qui renvoie $F_b(t)$.
   c) Avec Python, tracer la courbe représentative de $x \longmapsto F_{na}\left(x\sqrt{na} + na\right)$ sur l'intervalle $[-4\,;4]$ avec $n = 100$ et $a = 3$.
   d) Lui superposer la courbe de Φ, la fonction de répartition d'une variable aléatoire de loi $\mathcal{N}(0,1)$. Commenter.

b) Illustration à l'aide d'histogramme empirique

Exercice 9. (★★)

1) Écrire une fonction en Python qui prend en argument $n \in \mathbb{N}^*$ et le(s) paramètre(s) d'une certaine loi de probabilité et qui :
   - Stocke $N = 10000$ réalisations de $Y = \sqrt{n}\,\dfrac{\overline{X}_n - m}{\sigma}$, avec $\overline{X}_n = \dfrac{1}{n}\displaystyle\sum_{i=1}^{n} X_i$ où $X_1, \ldots, X_n$ sont des variables aléatoires indépendantes de la loi en question, dans un vecteur `Y`.
   - Trace l'histogramme renormalisé des valeurs de `Y` avec 20 classes via la commande : `plt.hist(Y,bins=20,density=True)`
   - Lui superpose la courbe de la densité de la loi Normale.
2) Tester avec $n = 100$, $n = 1000$, $n = 10000$ et les lois usuelles du programme (uniforme, Bernoulli, binomiale, géométrique, Poisson, exponentielle, gamma, Normale). Commenter.

IV Simulations d'une loi normale

1) Plusieurs méthodes de simulation

Voyons plusieurs méthodes permettant de simuler une variable aléatoire de loi $\mathcal{N}(0,1)$.

a) Utilisation de `rd.normal`

Pour simuler une une variable aléatoire de loi $\mathcal{N}(0,1)$, on peut utiliser la fonction `rd.normal(0,1)` du module `numpy`.

b) Utilisation du TCL

Si $(X_n)_{n\in\mathbb{N}^*}$ est une suite de variables aléatoires indépendantes de même loi admettant une espérance m et une variance $\sigma^2>0$, alors

$$\overline{X}_n^* = \frac{\sqrt{n}}{\sigma}\left(\frac{1}{n}\sum_{k=1}^{n} X_i - m\right)$$

converge en loi vers une variable aléatoire de loi $\mathcal{N}(0,1)$ quand n tend vers $+\infty$

Pour simuler une une variable aléatoire de loi $\mathcal{N}(0,1)$, on peut donc simuler $\overline{X}_n^*$ pour n assez grand et pour des variables aléatoires que l'on sait facilement simuler.

Avec des lois uniformes sur $[0\,;1]$: Pour n « assez grand » (mais pas tant que ça, comme on le verra, la convergence étant très rapide), si $X_1,\dots,X_n$ sont indépendantes de loi $\mathcal{U}([0\,;1])$ (on a $m=1/2$ et $\sigma^2=1/12$), alors

$$\overline{X}_n^* = \sqrt{12n}\left(\frac{1}{n}\sum_{k=1}^{n} X_i - \frac{1}{2}\right)$$

suit approximativement une loi $\mathcal{N}(0,1)$. Le programme suivant prend n en entrée et renvoie cette quantité :

```
import numpy.random as rd
def NormaleTCL(n):
    S=np.sum(rd.random(n))
    return np.sqrt(12*n)*(S/n-1/2)
```

Remarque : Si on prend $n=12$ (on verra ultérieurement que c'est déjà pas mal), on obtient que

$$\overline{X}_{12}^* = \sum_{k=1}^{12} X_i - 6$$

suit approximativement une loi $\mathcal{N}(0,1)$.

Avec des lois de Bernoulli de paramètre $p\in\,]0\,;1[$: Pour n « assez grand » (mais pas tant que ça, comme on le verra, la convergence étant très rapide), si $X_1,\dots,X_n$ sont indépendantes de loi $\mathcal{B}(p)$ (on a $m=p$ et $\sigma^2=p(1-p)$), alors

$$\overline{X}_n^* = \sqrt{\frac{n}{p(1-p)}}\left(\frac{1}{n}\sum_{k=1}^{n} X_i - p\right)$$

suit approximativement une loi $\mathcal{N}(0,1)$. Le programme suivant prend n et p en entrée et renvoie cette quantité :

```
import numpy.random as rd
def NormaleTCL2(n,p):
    S=rd.binomial(n,p)
    return np.sqrt(n/(p*(1-p)))*(S/n-p)
```

c) Utilisation de la méthode d'inversion de la fonction de répartition

Dans le TP n° 1, on a vu comment implémenter la fonction Φ, la fonction de répartition d'une variable aléatoire de loi $\mathcal{N}(0,1)$. On peut utiliser directement la commande `sp.ndtr` après `import scipy.special as sp`.

Exercice 10. (★)

1) Montrer que Φ est une bijection de $\mathbb{R}$ dans $]0\,;1[$.
2) Soit $U \hookrightarrow \mathcal{U}(]0\,;1[)$. Montrer que $Y = \Phi^{-1}(U)$ suit une loi $\mathcal{N}(0,1)$.
3) a) A l'aide d'un algorithme de dichotomie, écrire une fonction qui prend en entrée t et qui calcule $\Phi^{-1}(t)$. *On cherchera l'unique antécédent de t entre -5 et 5.*
   b) En déduire une fonction Python, intitulé `NormalInvRep`, sans argument renvoie une simulation d'une variable aléatoire de loi $\mathcal{N}(0,1)$ avec cette méthode.

Remarques :

- On a déjà vu cette méthode dans le TP n° 1. Il s'agit de la méthode d'inversion de la fonction de répartition qui est plus générale.
- La fonction Φ^{-1} est déjà implémentée dans Python. Il suffit d'utiliser la commande `sp.ndtri` après `import scipy.special as sp`. Mais celle-ci n'est pas au programme.

d) Utilisation de la méthode de Box-Muller

Exercice 11 – Méthode de Box-Muller. (★★★) On considère deux variables aléatoires U et V définies sur un même espace probabilisé $(\Omega, \mathcal{A}, \mathbb{P})$, indépendantes et de même loi uniforme sur $]0\,;1]$.

1) a) Montrer que $\sin$ réalise une bijection de $\left[-\frac{\pi}{2}\,;\frac{\pi}{2}\right]$ sur $[-1\,;1]$. On note Arcsin sa bijection réciproque.
   b) Montrer que Arcsin est dérivable sur $]-1\,;1[$ et que
   $$\forall x \in]-1\,;1[\,, \qquad \mathrm{Arcsin}'(x) = \frac{1}{\sqrt{1-x^2}}.$$
   c) Montrer que Arcsin est impaire.
2) On pose $X = \ln(\sqrt{-2\ln(U)})$. On admet que c'est une variable aléatoire sur $(\Omega, \mathcal{A}, \mathbb{P})$.
   a) Calculer la fonction de $-2\ln(U)$
   b) Montrer que X est une variable aléatoire à densité donc une densité est $g : x \longmapsto \exp\left(2x - \frac{e^{2x}}{2}\right)$.
3) On pose $Y = \sin(2\pi V)$ et $Z = \ln(|Y|)$. On admet que Y et Z sont des variables aléatoires sur $(\Omega, \mathcal{A}, \mathbb{P})$.
   a) Montrer que, pour tout $x \in]0\,;1]$, $[Y \geqslant x] = [\mathrm{Arcsin}(x) \leqslant 2\pi V \leqslant \pi - \mathrm{Arcsin}(x)]$.
   b) Montrer que, pour tout $x \in [-1\,;0]$, $[Y \leqslant x] = [\pi - \mathrm{Arcsin}(x) \leqslant 2\pi V \leqslant 2\pi + \mathrm{Arcsin}(x)]$.
   c) En déduire la fonction de répartition de Y.
   d) Montrer que Z admet pour densité la fonction
   $$h : x \longmapsto \begin{cases} \dfrac{2}{\pi}\dfrac{e^x}{\sqrt{1-e^{2x}}} & \text{si} \quad x < 0 \\ 0 & \text{si} \quad x \geqslant 0 \end{cases}$$
4) Justifier que la variable aléatoire $S = X + Z$ admet une densité et donner en une densité sous forme d'une intégrale.
5) On pose $T = \sqrt{-2\ln(U)}\sin(2\pi V)$. On admet que c'est une variable aléatoire sur $(\Omega, \mathcal{A}, \mathbb{P})$. Remarquons que $\ln(|T|) = S$.

a) Montrer qu'une densité de $|T|$ sur R est

$$f : x \longmapsto \begin{cases} \dfrac{2}{\pi}\displaystyle\int_{\ln(x)}^{+\infty} \frac{\exp\left(-e^{2t}/2\right)}{\sqrt{e^{2t}-x^2}} e^{2t}\,dt & \text{si} \quad x > 0 \\ 0 & \text{si} \quad x \leqslant 0 \end{cases}$$

b) Soit $x > 0$. A l'aide du changement de variable $t = \dfrac{1}{2}\ln(x^2+u^2)$, établir que $f(x) = \dfrac{2}{\sqrt{2\pi}} e^{-x^2/2}$.

6) Notons F et F_T les fonctions de répartition respectives de $|T|$ et T.

a) Montrer que $-T = \sqrt{-2\ln(U)}\sin(2\pi(1-V))$. En déduire que T et $-T$ ont la même loi.

b) En déduire que, pour tout $x \in \mathbb{R}$, $\mathbb{P}(T = x) = 0$.

c) Montrer que, pour tout $x \in \mathbb{R}_+$, $F_T(x) = \dfrac{1+F(x)}{2}$.

d) Montrer que, pour tout $x \in \mathbb{R}_-^*$, $F_T(x) = \dfrac{1-F(-x)}{2}$.

e) Conclure que T est bien une variable à densité.

f) En déduire que T suit une loi $\mathcal{N}(0,1)$.

7) Proposer une méthode de simulation d'une variable aléatoire de loi $\mathcal{N}(0,1)$.

2) Comparaison des méthodes

Exercice 12 – Comparaison des méthodes de simulation de variables aléatoires de loi $\mathcal{N}(0,1)$. (★★)
En s'inspirant du script de la dernière question de l'exercice 3, créer des vecteurs contenant 100000 réalisations de variables aléatoires indépendantes de loi $\mathcal{N}(0,1)$ obtenus chacun avec une des méthodes de simulations vues précédemment. Afficher un histogramme renormalisé et le temps d'exécution de chaque algorithme. Commenter.

Feuille de TP n° 5

Fonctions de deux variables – Deuxième partie

I Plan tangent

Soit Ω un ouvert de $\mathbb{R}^2$. Soit $a = (x_0, y_0) \in \Omega$. Si f est une fonction de classe C^1 sur Ω à valeurs dans $\mathbb{R}$, alors la formule de Taylor à l'ordre 1 assure que

$$f(x) \approx f(a) + \langle \nabla f(a), x - a \rangle,$$

pour tout x au voisinage de a. Cela motive la définition suivante :

Définition. *Soit Ω un ouvert de $\mathbb{R}^2$. Soit $(x_0, y_0) \in \Omega$. Soit $f : \Omega \longmapsto \mathbb{R}$ de classe C^1 sur Ω. On appelle plan tangent au graphe de f en (x_0, y_0) le plan d'équation*

$$z = f(x_0, y_0) + \partial_1 f(x_0, y_0)(x - x_0) + \partial_2 f(x_0, y_0)(y - x_0).$$

Ainsi, pour représenter le plan tangent, on représente localement le graphe de la fonction (de deux variables)

$$g : (x, y) \in \mathbb{R}^2 \longmapsto f(x_0, y_0) + \partial_1 f(x_0, y_0)(x - x_0) + \partial_2 f(x_0, y_0)(y - x_0).$$

Exemple : *Testons avec la fonction $f : (x, y) \in \mathbb{R}^2 \longmapsto xye^{-(x^2+y^2)}$ (cf. TP n° 2). On commence par les importations, l'implémentation de la fonction et du gradient et on représente la surface.*

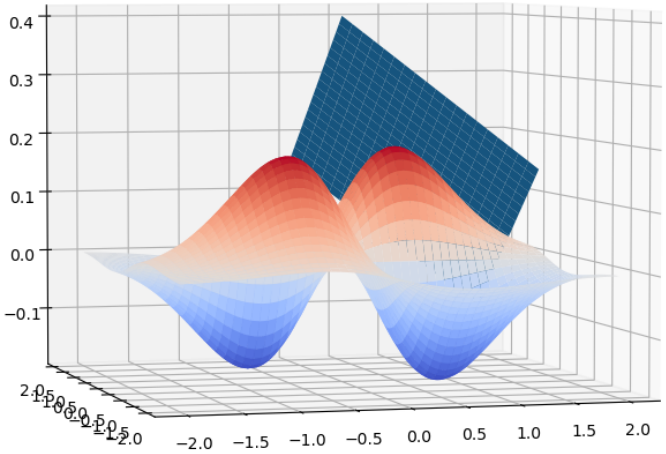
```
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
ax=Axes3D(plt.figure())
from matplotlib.cm import coolwarm

def f(x,y) :
    return x*y*np.exp(-(x**2+y**2))

def gradf(x,y):
    g=np.exp(-(x**2+y**2))
    return y*(1-2*x*x)*g,x*(1-2*y*y)*g

X=np.arange(-2,2,0.01)
Y=X
X,Y=np.meshgrid(X,Y)
Z1=f(X,Y)
ax.plot_surface(X,Y,Z1,cmap=coolwarm)
```

Ensuite, on peut définir la fonction qui implémente le plan :

```
def plan(x0,y0,x,y):
    a,b=gradf(x0,y0)
    return f(x0,y0)+a*(x-x0)+b*(y-y0)
```

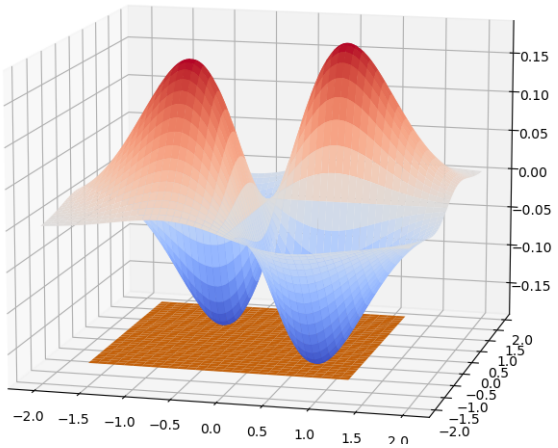
Représentons le plan tangent en $(1,1)$ *:*

```
X=np.arange(0,2,0.1)
Y=X
X,Y=np.meshgrid(X,Y)
Z2=plan(1,1,X,Y)
ax.plot_surface(X,Y,Z2)
plt.show()
```



Représentons le plan tangent en $(-1/\sqrt{2},1/\sqrt{2})$ *:*

```
a=1/2**(1/2)
X=np.arange(-a-1,-a+2,0.1)
Y=np.arange(a-2,a+1,0.1)
X,Y=np.meshgrid(X,Y)
Z2=plan(-a,a,X,Y)
ax.plot_surface(X,Y,Z2)
plt.show()
```



Supposons de plus que f soit de classe C^2 sur Ω. La formule de Taylor à l'ordre 2 entraîne alors que

$$f(x,y) = \underbrace{f(x_0,y_0) + \left\langle \nabla f(x_0,y_0), \begin{pmatrix} x-x_0 \\ y-x_0 \end{pmatrix} \right\rangle}_{=g(x,y)} + \frac{1}{2} q_{(x_0,y_0)}(x-x_0,y-y_0) + \|(x-x_0,y-y_0)\|^2 \varepsilon(x-x_0,y-y_0),$$

où $q_{(x_0,y_0)}$ est la forme quadratique associée à $\nabla^2 f(x_0,y_0)$, la Hessienne en (x_0,y_0) et où ε est une fonction définie au voisinage de $(0,0)$, continue en $(0,0)$ et nulle en $(0,0)$. Ainsi $f(x,y)-g(x,y)$ est du signe de $q_{(x_0,y_0)}(x-x_0,y-y_0)$ au voisinage de (x_0,y_0). Ainsi :

- Si $\nabla^2 f(x_0,y_0)$ ne possède que des valeurs propres strictement positives, $f(x,y)-g(x,y) \geqslant 0$ au voisinage de (x_0,y_0) et donc le plan tangent est en-dessous de la surface représentative de f au voisinage de (x_0,y_0).
- Si $\nabla^2 f(x_0,y_0)$ ne possède que des valeurs propres strictement négatives, $f(x,y)-g(x,y) \leqslant 0$ au voisinage de (x_0,y_0) et donc le plan tangent est au-dessus de la surface représentative de f au voisinage de (x_0,y_0).
- Si $\nabla^2 f(x_0,y_0)$ possède des valeurs propres de signes contraires, la surface représentative de f traverse le plan tangent au voisinage de (x_0,y_0).

Si (x_0,y_0) est un point critique, alors le plan tangent est horizontal. Dire que, au voisinage de (x_0,y_0), le plan tangent en (x_0,y_0) est en-dessous (respectivement au-dessus) de la surface représentative de f signifie que f admet un minimum (respectivement maximum) local en (x_0,y_0). On retrouve donc le résultat du cours.

Le code suivant permet de calculer les valeurs propres de la matrice Hessienne de f en (x_0,y_0) :

```
def Hessf(x,y):
    d11 = ..... #dérivée partielle d'ordre 2 d'indice (1,1)
    d22 = ..... #dérivée partielle d'ordre 2 d'indice (2,2)
    d12 = ..... #dérivée partielle d'ordre 2 d'indice (1,2)
    return np.array([[d11,d12],[d12,d22]])

import numpy.linalg as al
al.eig(Hessf(x0,y0))[0]
```

Exemple : *Reprenons l'exemple de la fonction* $f : (x,y) \in \mathbb{R}^2 \longmapsto xye^{-(x^2+y^2)}$. *C'est une fonction de classe* C^2 *sur* $\mathbb{R}^2$ *et, pour tout* $(x,y) \in \mathbb{R}^2$,

$$\nabla^2 f(x,y) = \begin{pmatrix} 2xy(2x^2-3)e^{-(x^2+y^2)} & (1-2x^2)(1-2y^2)e^{-(x^2+y^2)} \\ (1-2x^2)(1-2y^2)e^{-(x^2+y^2)} & 2xy(2y^2-3)e^{-(x^2+y^2)} \end{pmatrix}$$

Cela donne le code suivant :

```
def Hessf(x,y):
    g=np.exp(-(x**2+y**2))
    d11=2*x*y*(2*x*x-3)*g
    d22=2*x*y*(2*y*y-3)*g
    d12=(1-2*x*x)*(1-2*y*y)*g
    return np.array([[d11,d12],[d12,d22]])

import numpy.linalg as al
al.eig(Hessf(1/np.sqrt(2),1/np.sqrt(2)))[0]
al.eig(Hessf(1/np.sqrt(2),-1/np.sqrt(2)))[0]
al.eig(Hessf(-1/np.sqrt(2),-1/np.sqrt(2)))[0]
al.eig(Hessf(-1/np.sqrt(2),1/np.sqrt(2)))[0]
al.eig(Hessf(0,0))[0]
al.eig(Hessf(1,1))[0]
```

On trouve : `[-0.736,-0.736]`, `[0.736,0.736]`, `[-0.736,-0.736]`, `[0.736, 0.736]`, `[1.,-1.]` *et* `[-0.135,-0.406]`. *Cela nous permet de conjecturer que* f *admet un maximum local en* $(1/\sqrt{2}, 1/\sqrt{2})$, *un minimum local en* $(1/\sqrt{2}, -1/\sqrt{2})$, *un maximum local en* $(-1/\sqrt{2}, -1/\sqrt{2})$, *un minimum local en* $(-1/\sqrt{2}, 1/\sqrt{2})$, *un point selle en* $(0,0)$. *De plus, le plan tangent en* $(1,1)$ *est au-dessus de la surface représentative.*

Exercice 1. (★★) Reprendre l'exercice 2 du TP n° 2 et conjecturer la nature locale des points critiques des différentes fonctions à l'aide du signe des valeurs propres de la Hessienne obtenu avec Python. Puis le démontrer.

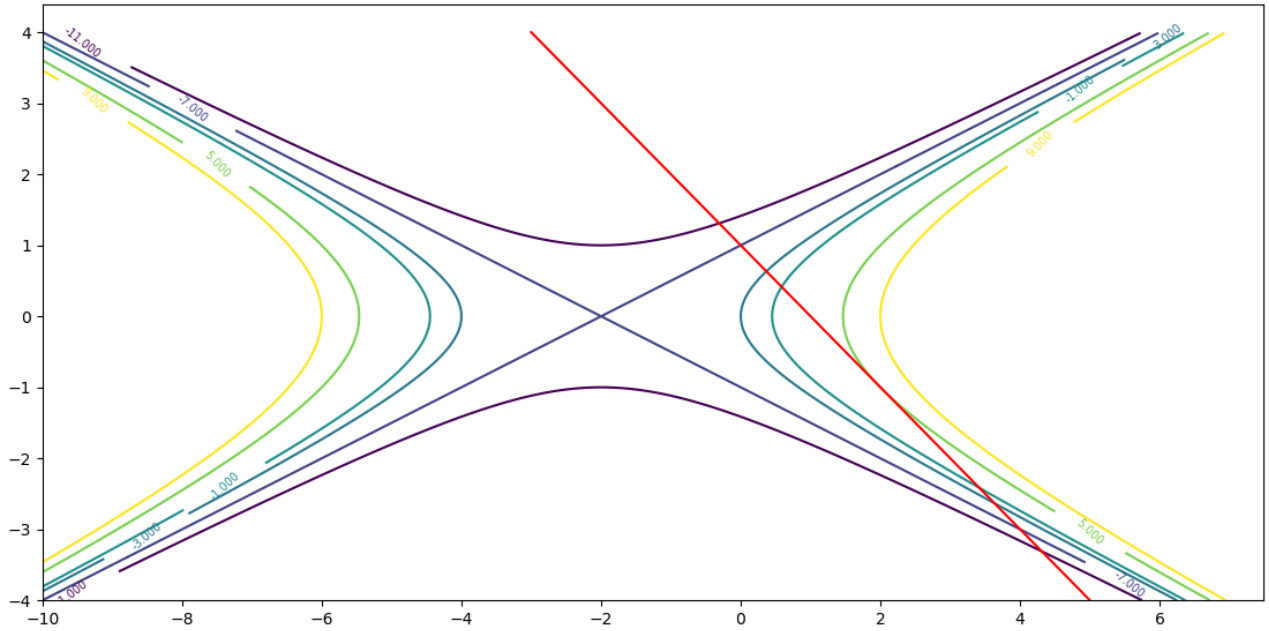
II Optimisation sous contrainte linéaire

Soit Ω un ouvert de $\mathbb{R}^2$. Soit $a = (x_0, y_0) \in \Omega$. Soit f est une fonction de classe C^1 sur Ω à valeurs dans $\mathbb{R}$. On étudie f sous la contrainte linéaire $\mathscr{C} = \{(x,y) \in \mathbb{R}^2 \,|\, g(x,y) = c\}$. Puisqu'il s'agit d'une contrainte linéaire, il existe deux réels a et b tels que $g : (x,y) \longmapsto ax + by$. On remarque que $\mathscr{C}$ est une droite affine d

On suppose que, sous la contrainte $\mathscr{C}$, f admet un extremum local en un point (x_0, y_0). Ce point appartient donc à la ligne de niveau $f(x_0, y_0)$ de f. Notons-la $\mathscr{L}_0$.

- On rappelle qu'une ligne de niveau est formée des points de même altitude. Ainsi, d'un côté de la ligne $\mathscr{L}_0$ se trouvent des points d'altitude strictement plus élevée que $f(x_0, y_0)$ et de l'autre des points d'altitude strictement moins élevée que $f(x_0, y_0)$.
- Le point (x_0, y_0) appartient à la contrainte $\mathscr{C}$, qui se trouve être une droite affine. Ainsi (x_0, y_0) est un point d'intersection de $\mathscr{L}_0$ et de $\mathscr{C}$.
- Si la droite affine $\mathscr{C}$ traverse la ligne de niveau $\mathscr{L}_0$, alors il existe, au voisinage de (x_0, y_0) des points de $\mathscr{C}$ en lesquels f prend des valeurs supérieurs à $f(x_0, y_0)$ et d'autres points en lesquels f prend des valeurs inférieures à $f(x_0, y_0)$, ce qui contredit que f admet un extremum local en (x_0, y_0) sous la contrainte $\mathscr{C}$. Ainsi $\mathscr{C}$ est confondue avec la tangente à $\mathscr{L}_0$ en (x_0, y_0).
- On peut retrouver ce dernier point avec les résultats vus en cours : $\nabla f(x_0, y_0)$ est orthogonale au noyau de g. De plus on a vu dans le TP n° 2 que $\nabla f(x_0, y_0)$ est orthogonal à la tangente à $\mathscr{L}_0$ en (x_0, y_0). Ainsi la tangente à $\mathscr{L}_0$ en (x_0, y_0) est parallèle au noyau de g, qui est lui même parallèle à la contrainte $\mathscr{C}$. Comme elles ont (x_0, y_0) pour point d'intersection, elles sont confondues.

Exemple : *Soit* $f : (x,y) \longmapsto x^2 + 4x - 4y^2 - 3$ *et la contrainte* $\mathscr{C} = \{(x,y) \in \mathbb{R}^2 \,|\, x + y = 1\}$. *Traçons des lignes de niveaux et la droite d'équation* $y = -x + 1$ *:*



On conjecture que $\mathscr{C}$ (en rouge) est tangente à la ligne de niveau 5 et passe par le point $(2,-1)$. Cela suggère que $(2,-1)$ est un point critique de f sous la contrainte $\mathscr{C}$.

Exercice 2. (★★) Soient $f : (x,y) \longmapsto x^2+4x-4y^2-3$ et $\mathscr{C} = \{(x,y) \in \mathbb{R}^2 \,|\, x+y=1\}$.

1) Reproduire la figure de l'exemple ci-dessus à l'aide de Python.
2) Démontrer que f admet bien $(2,-1)$ pour point critique de f sous la contrainte $\mathscr{C}$.
3) Montrer que f admet un maximum global en $(-2,1)$ sous la contrainte $\mathscr{C}$.

Exercice 3. (★★) Soient $f : (x,y) \longmapsto 2x^3-3x^2y-6x+y^3$ et $\mathscr{C} = \{(x,y) \in \mathbb{R}^2 \,|\, 2x+y=1\}$.

1) Représenter les lignes de niveaux de f et la droite $\mathscr{C}$. Conjecturer la présence d'un point critique pour f sous la contrainte $\mathscr{C}$.
   On pourra restreindre le tracé au rectangle $[-6\,;4]\times[-5\,;3]$ et on fera en sorte que le niveau -3 soit représenté.
2) Démontrer que f admet bien un unique point critique de f sous la contrainte $\mathscr{C}$
3) Montrer que f admet un unique minimum global sous la contrainte $\mathscr{C}$.

Exercice 4. (★★) Soient $f : (x,y) \longmapsto x^3+2x^2y-4y^3$ et $\mathscr{C} = \{(x,y) \in \mathbb{R}^2 \,|\, x-y=1\}$.

1) Représenter les lignes de niveaux de f et la droite $\mathscr{C}$. Conjecturer la présence de deux points critiques pour f sous la contrainte $\mathscr{C}$.
   On pourra restreindre le tracé au carré $[-8\,;8]^2$ et on fera en sorte que les niveaux $4/27$ et 76 soient représentés.
2) Démontrer que f admet bien deux points critiques de f sous la contrainte $\mathscr{C}$
3) Montrer que, sous la contrainte $\mathscr{C}$, f admet un minimum local et un maximum local mais qu'il ne s'agit pas de d'extrema globaux.

Feuille de TP n° 6

Estimation ponctuelle et par intervalle de confiance

Exercice 1. (★) Écrire une fonction en Python qui prend en entrée n et α et qui simule $N = 10000$ fois des variables aléatoires $X_1, \dots, X_n$ indépendantes et de même loi uniforme sur $]0\,;1[$ et qui renvoie la proportion de fois, parmi ces N fois, où $m = \dfrac{1}{2}$ appartient à l'intervalle de confiance asymptotique

$$\left[\overline{X}_n - t_{1-\alpha/2}\frac{S_n}{\sqrt{n}}\,;\overline{X}_n + t_{1-\alpha/2}\frac{S_n}{\sqrt{n}}\right].$$

La tester pour différentes valeurs de n et α. Commenter.

Pour la réciproque de Φ, on utilisera `sp.ndtri` après `import scipy.special as sp`.

Exercice 2. (★★) Soit $(X_n)_{n\geqslant 1}$ une suite de variables aléatoires admettant [2] un moment d'ordre 4. On note m leur espérance et σ^2 leur variance. Pour tout $n \in \mathbb{N}^*$, on pose

$$\overline{X}_n = \frac{1}{n}\sum_{i=1}^{n} X_i \qquad \text{et} \qquad S_n^2 = \frac{1}{n}\sum_{i=1}^{n}(X_i - \overline{X}_n)^2.$$

La quantité S_n^2 est appelée variance empirique de $X_1, \dots, X_n$.

1) Pour tout $n \in \mathbb{N}^*$, notons

$$T_n^2 = \frac{1}{n}\sum_{i=1}^{n} X_i^2 \qquad \text{et} \qquad t^2 = \mathbb{E}(X_1^2).$$

a) Justifier que $T_n^2 \xrightarrow[n\to+\infty]{\mathbb{P}} t^2$ et $(\overline{X}_n)^2 \xrightarrow[n\to+\infty]{\mathbb{P}} m^2$.

b) Vérifier que, pour tout $n \in \mathbb{N}^*$,

$$\forall n \in \mathbb{N}^*, \qquad S_n^2 - \sigma^2 = T_n^2 - t^2 - \big((\overline{X}_n)^2 - m^2\big).$$

c) Soient $n \in \mathbb{N}^*$ et $\varepsilon > 0$. En déduire que

$$\left[\left|S_n^2 - \sigma^2\right| \geqslant \varepsilon\right] \subset \left[\left|T_n^2 - t^2\right| \geqslant \frac{\varepsilon}{2}\right] \cup \left[\left|(\overline{X}_n)^2 - m^2\right| \geqslant \frac{\varepsilon}{2}\right].$$

2) En déduire que $S_n^2 \xrightarrow[n\to+\infty]{\mathbb{P}} \sigma^2$.

En vertu du résultat de l'exercice précédent, du Théorème Central Limite et d'un théorème (qui a disparu du nouveau programme mais qui était pourtant fort utile) appelée *lemme de Slutsky*, on obtient le résultat (hors-programme donc) suivant :

$$\sqrt{n}\,\frac{\overline{X}_n - m}{S_n} \xrightarrow[n\to+\infty]{\mathscr{L}} Z,$$

avec Z une variable aléatoire de loi $\mathcal{N}(0,1)$. Nous allons utiliser cette convergence dans l'exercice suivant.

2. En fait l'existence d'une variance suffit puisque la loi faible des grands nombres reste vrai si les variables aléatoires en jeu admettent seulement une espérance (mais, dans le programme, il faut un moment d'ordre 2 pour pouvoir le démontrer).

Exercice 3 – Intervalle de confiance pour l'espérance d'une loi Normale de variance connue. (★★)
Soient $m \in \mathbb{R}$, $\sigma > 0$ et $\alpha \in]0\,;1[$. Soit $(X_i)_{i\geqslant 1}$ une suite de variable aléatoires de loi $\mathcal{N}(m, \sigma^2)$ (on suppose que σ est connu). Pour tout $n \in \mathbb{N}^*$, notons $\overline{X}_n = \dfrac{1}{n}\displaystyle\sum_{k=1}^{n} X_i$.

1) Justifier que $\overline{X}_n$ est un estimateur sans biais et convergent de m.
2) On note u_α désigne l'unique antécédent de $1 - \dfrac{\alpha}{2}$ par Φ. Justifier que $\left[\overline{X}_n - \dfrac{u_\alpha \sigma}{\sqrt{n}}\,;\overline{X}_n + \dfrac{u_\alpha \sigma}{\sqrt{n}}\right]$ est un intervalle de confiance (non asymptotique) de niveau $1-\alpha$ pour m.
3) Pour $\alpha = 0.05$ et pour différentes valeurs de m, σ et n, simuler $N = 10000$ fois $\overline{X}_n$ et compter la proportion de fois où m tombe dans l'intervalle de confiance (on parle de niveau réel) et comparer cette proportion avec $1-\alpha$.

Exercice 4 – Intervalle de confiance asymptotique pour la paramètre d'une loi de Bernoulli. (★★)
Soient $p \in]0\,;1[$ et $\alpha \in]0\,;1[$. Soit $(X_i)_{i\geqslant 1}$ une suite de variable aléatoires de loi $\mathcal{B}(p)$. Pour tout $n \in \mathbb{N}^*$, notons $\overline{X}_n = \dfrac{1}{n}\displaystyle\sum_{k=1}^{n} X_i$.

1) Justifier que $\overline{X}_n$ est un estimateur sans biais et convergent de p.
2) Montrer que $p(1-p) \leqslant \dfrac{1}{4}$.
3) Montrer que $\left[\overline{X}_n - \dfrac{1}{2\sqrt{n\alpha}}\,;\overline{X}_n + \dfrac{1}{2\sqrt{n\alpha}}\right]$ est un intervalle de confiance (non asymptotique) de niveau $1-\alpha$ pour p.
4) On note u_α désigne l'unique antécédent de $1 - \dfrac{\alpha}{2}$ par Φ. Justifier que $\left[\overline{X}_n - \dfrac{u_\alpha}{2\sqrt{n}}\,;\overline{X}_n + \dfrac{u_\alpha}{2\sqrt{n}}\right]$ est un intervalle de confiance asymptotique de niveau $1-\alpha$ pour p.
5) Pour tout $n \in \mathbb{N}^*$, notons $S_n = \sqrt{\dfrac{1}{n}\displaystyle\sum_{k=1}^{n}(X_i - \overline{X}_n))^2}$. On admet que $\sqrt{n}\dfrac{\overline{X}_n - p}{S_n} \xrightarrow[n\to+\infty]{\mathscr{L}} N$ avec $N \hookrightarrow \mathcal{N}(0,1)$. Montrer alors que $\left[\overline{X}_n - \dfrac{u_\alpha S_n}{\sqrt{n}}\,;\overline{X}_n + \dfrac{u_\alpha S_n}{\sqrt{n}}\right]$ est un intervalle de confiance asymptotique de niveau $1-\alpha$ pour p.
6) Pour $\alpha = 0.05$ et pour différentes valeurs de p et de n, simuler $N = 10000$ fois $X_1, \ldots, X_n$ et compter :
   - l'amplitude des deux premiers intervalles de confiance et l'amplitude moyenne (on parle d'incertitude) du troisième.
   - la proportion de fois où p tombe dans ces intervalles de confiance (on parle de niveau réel) et comparer cette proportion avec $1-\alpha$.

   Quel intervalle de confiance est le meilleur ?

Problème – Adapté de ECRICOME 2021 et des oraux de l'ESCP 2005. (★★★) Toutes les variables aléatoires considérées dans cet exercice sont définies sur le même espace probabilisé $(\Omega, \mathcal{A}, \mathbb{P})$. Soit $\theta \in \mathbb{R}_+^*$. On considère X une variable aléatoire admettant pour densité

$$f_\theta : x \longmapsto \begin{cases} \dfrac{2x}{\theta^2} & \text{si} \quad x \in [0\,;\theta] \\ 0 & \text{sinon.} \end{cases}$$

1) a) Justifier que f_θ est bien une densité de probabilité.
   b) Expliciter la fonction de répartition F_θ de X.

On considère une suite de variables aléatoires $(X_n)_{n\geqslant 1}$ indépendantes et de même loi que X.

2) **Étude d'un premier estimateur de θ.** Pour tout $n \in \mathbb{N}^*$, on note $M_n = \max\{X_1; X_2; \dots; X_n\}$.

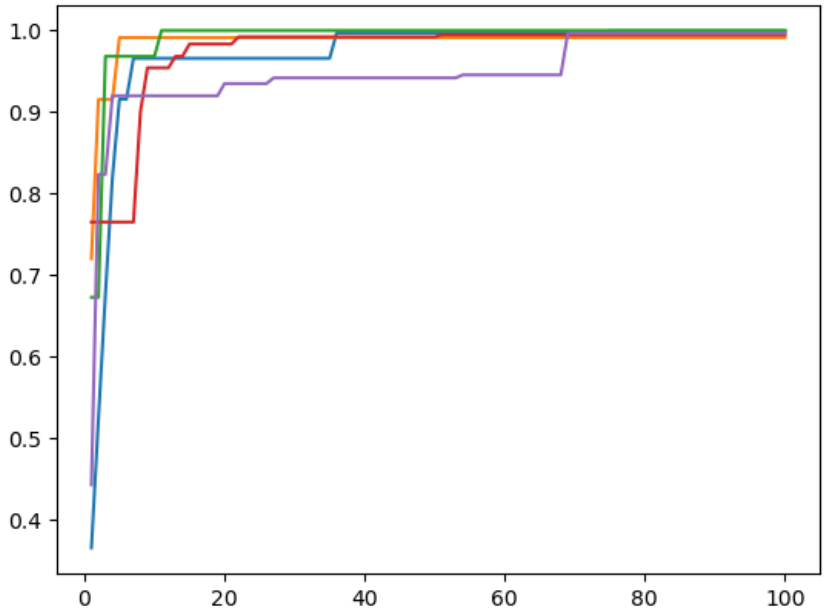
a) Montrer que, si $U \hookrightarrow \mathcal{U}([0\,;1])$, alors $\theta\sqrt{U}$ a la même loi que X.

b) Écrire une fonction en Python, appelée `Simul_M`, qui prend en entrée n et θ et qui renvoie une simulation de M_n.

c) Modifier la fonction précédente pour qu'elle renvoie une liste contenant une réalisation de $(M_1, \dots, M_n)$. On l'appellera `Simul_MM`.

On rappelle que, en Python, si `L` *désigne une liste ou un vecteur et si* `i` *désigne un entier non nul, alors* `L[:i]` *renvoie uniquement les* `i` *premières coordonnées de* `L`.

d) Écrire un script en Python permet d'obtenir le graphique ci-dessus consistant en cinq réalisations mutuellement indépendantes de $(M_1, \dots, M_{100})$ dans le cas où $\theta = 1$.



A partir de ce graphique, que peut-on conjecturer sur l'estimateur M_n ?

e) Montrer que, pour tout $n \in \mathbb{N}^*$, M_n est une variable aléatoire à densité. Donner en une densité.

f) Calculer les deux premiers moments de M_n.

g) Montrer que M_n est un estimateur biaisé de θ. Est-il asymptotiquement sans biais ?

h) Soit $\varepsilon > 0$ et $n \in \mathbb{N}^*$. Expliciter $\mathbb{P}(|M_n - \theta| \geqslant \varepsilon)$. L'estimateur M_n est-il convergent ?

i) Calculer $\mathbb{E}((M_n - \theta)^2)$ (*quantité appelée risque quadratique*[1] *de* M_n). Quel résultat retrouve-t-on ?

3) **Étude d'un second estimateur de θ.** Pour tout $n \in \mathbb{N}^*$, notons $\overline{X}_n = \dfrac{1}{n}\displaystyle\sum_{i=1}^{n} X_i$.

a) Donner l'espérance et la variance de $\overline{X}_n$.

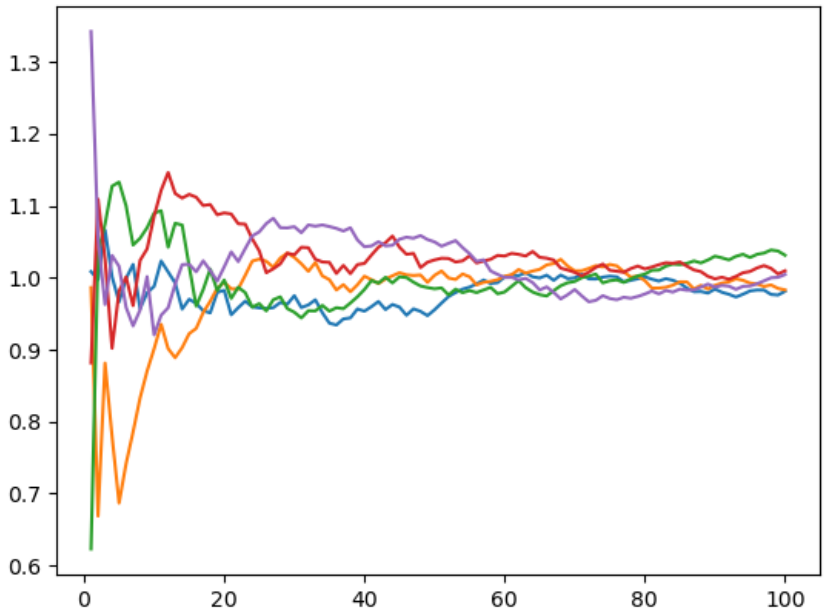
b) Déduire de la variable aléatoire $\overline{X}_n$ un estimateur sans biais Z_n de θ.

c) Écrire une fonction en Python, appelée `Simul_Z`, qui prend en entrée n et θ et qui renvoie une réalisation de Z_n.

d) Modifier la fonction précédente pour qu'elle renvoie une liste contenant une réalisation de $(Z_1, \dots, Z_n)$. On l'appellera `Simul_ZZ`.

e) Écrire un script en Python permet d'obtenir le graphique ci-dessus consistant en cinq réalisations mutuellement indépendantes de $(Z_1, \dots, Z_{100})$ dans le cas où $\theta = 1$.

1. Le risque quadratique d'un estimateur mesure l'écart moyen entre l'estimateur est le paramètre qu'il estime. Plus il est petit, plus l'estimateur est précis



A partir de ce graphique, que peut-on conjecturer sur l'estimateur M_n ?

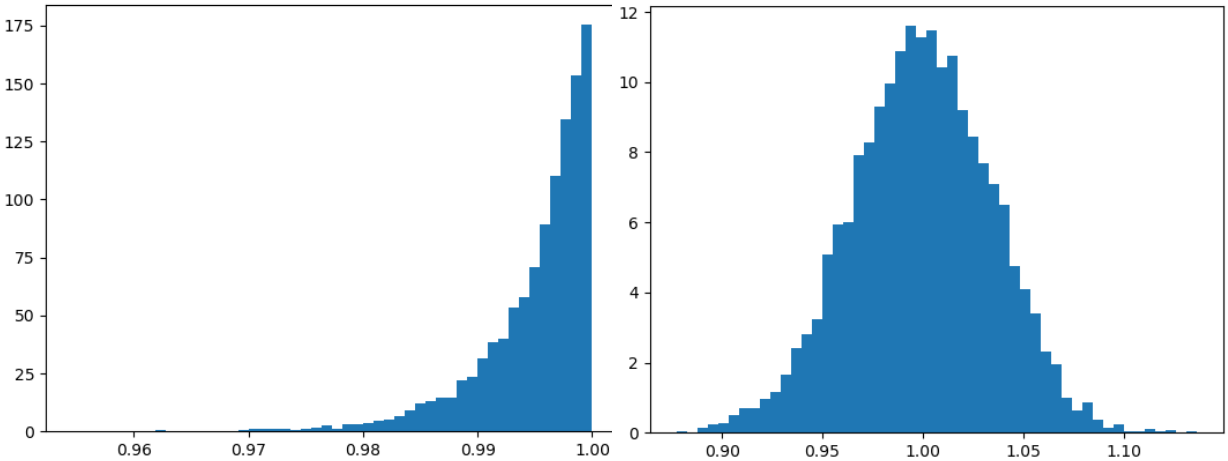
f) Calculer $\mathbb{E}((Z_n-\theta)^2)$ (*quantité appelée risque quadratique*[1] *de* Z_n). L'estimateur Z_n est-il convergent ?

4) **Comparaison des deux estimateurs.**

a) Comparer les risques quadratiques de M_n et Z_n. Quelle méthode choisiriez-vous pour estimer la valeur du paramètre θ ?

b) *On rappelle que, en Python, la commande* `plt.hist(L,bins=50,density=True` *trace un histogramme renormalisé avec 50 classes à partir des données de la liste* `L`.

Écrire un script Python qui permet d'obtenir les graphiques ci-dessus consistant en un histogramme de 5000 réalisation de M_{100} et un histogramme de 5000 réalisation de Z_{100} dans le cas où $\theta = 1$.



En quoi ces graphiques confortent-ils votre choix d'estimateur ?

5) **Un premier intervalle de confiance asymptotique pour θ.**

a) Montrer que la suite de variables aléatoires $\big(2n(\theta - M_n)\big)_{n\geqslant 1}$ convergence en loi vers une variable aléatoire dont on identifiera la loi et son (ses) paramètre(s).

b) Soit $\alpha \in]0\,;1[$. Déduire de la question précédente un intervalle de confiance asymptotique de niveau de confiance $1-\alpha$ pour le paramètre θ, construit à partir de M_n.

6) **Un second intervalle de confiance asymptotique pour θ.**

a) Montrer que la suite de variables aléatoires $\big(\sqrt{n}(Z_n - \theta)\big)_{n\geqslant 1}$ convergence en loi vers une variable aléatoire dont on identifiera la loi et son (ses) paramètre(s).

b) Soit $\alpha \in]0\,;1[$. Déduire de la question précédente un intervalle de confiance asymptotique de niveau de confiance $1-\alpha$ pour le paramètre θ, construit à partir de Z_n.

7) **Comparaison des deux intervalles de confiance asymptotiques.**

a) Lequel de ces deux intervalles de confiance asymptotique vous semble le meilleur ?

b) *En Python, si* `A` *désigne une matrice réelle, alors la commande* `np.max(A,0)` *renvoie un vecteur ligne contenant le maximum de chaque colonne de* `A`. *La commande* `np.mean(A,0)` *renvoie quand à elle un vecteur ligne contenant la moyenne de chaque colonne de* `A`.

Recopier et compléter le script Python suivant :

```
1  n=100
2  N=10000
3  theta=2
4  alpha=0.05
5  X=.....#Une matrice de taile nxN de réalisations de X
6  M=.....#N réalisations de M_n
7  Z=.....#N réalisations de Z_n
8
9  #Définition des bornes des intervalles de confiance :
10 aM=.....; bM=.....
11 import scipy.special as sp
12 u=sp.ndtri(1-alpha/2)#on a Phi(u)=1-alpha/2
13 aZ=.....; bZ=.....
14
15 #"Incertitude" : Largeur moyenne des deux intervalles de confiances :
16 LM=np.mean(bM-aM)
17 LZ=np.mean(bZ-aZ)
18
19 #"Niveau réel" : Proportion d'intervalles de confiances contenant theta :
20 pM=(np.sum(theta<=bM)-np.sum(theta<aM))/N
21 pZ=(np.sum(theta<=bZ)-np.sum(theta<aZ))/N
22
23 print("Estimation ponctuelle de theta par l'estimateur M_n : "+str(M[0]))
24 print("Intervalles de confiance obtenus avec l'estimateur M_n :")
25 print('Largeur moyenne : '+str(LM)+' et niveau réel : '+str(pM*100))
26 print("Estimation ponctuelle de theta par l'estimateur Z_n : "+str(Z[0]))
27 print("Intervalles de confiance obtenus avec l'estimateur Z_n :")
28 print('Largeur moyenne : '+str(LZ)+' et niveau réel : '+str(pZ*100))
```

Expliquer ce que font les commandes des lignes 20 et 21 du script.

c) Exécuter ensuite ce script. Commenter.

8) **Un intervalle de confiance pour θ.** On suppose dans cette question que n est un entier impair. Il existe donc $k \in \mathbb{N}$ tel que $n = 2k+1$. On admet l'existence de $2k+1$ fonctions $\varphi_1, \varphi_2, \dots, \varphi_{2k+1}$ continues sur $\mathbb{R}^{2k+1}$ à valeurs réelles telles que $Y_1, Y_2, \dots, Y_{2k+1}$ définies par

$$\forall i \in [\![1\,;2k+1]\!], \qquad Y_i = \varphi_i(X_1, X_2, \dots, X_{2k+1})$$

sont des variables aléatoires à densité de telle sorte que, pour tout $\omega \in \Omega$, $Y_1(\omega), Y_2(\omega), \dots, Y_{2k+1}(\omega)$ soit un réarrangement par ordre croissante de $X_1(\omega), X_2(\omega), \dots, X_{2k+1}(\omega)$:

$$\forall \omega \in \Omega, \qquad Y_1(\omega) \leqslant Y_2(\omega) \leqslant \dots \leqslant Y_{2k+1}(\omega).$$

En particulier Y_{k+1} est la médiane empirique de l'échantillon $(X_1, \dots, X_{2k+1})$. On pose $T_n = \sqrt{2}\,Y_{k+1}$.

On se donne $\alpha \in]0\,;1[$. On définit la variable aléatoire D_n

$$D_n = \operatorname{card}\left(\left\{i \in [\![1\,;n]\!] \,\middle|\, X_i < \frac{\theta}{\sqrt{2}}\right\}\right) = \sum_{i=1}^{n} \mathbb{1}_{X_i < \theta/\sqrt{2}}$$

a) Justifier que D_n suit une loi binomiale dont on précisera les paramètres.

b) Justifier que, pour tout $i \in [\![1\,;n]\!]$, $\left[Y_i \geqslant \dfrac{\theta}{\sqrt{2}}\right] = [D_n < i]$.

c) Montrer que, pour tout $\ell \in [\![0\,;n]\!]$, $\mathbb{P}(D_n \geqslant n-\ell) = \mathbb{P}(D_n \leqslant \ell)$. En déduire que $\mathbb{P}(D_n \geqslant k+1) = \dfrac{1}{2}$.

d) Montrer que l'ensemble $E_{n,\alpha} = \left\{i \in \mathbb{N} \,\middle|\, \mathbb{P}(D_n \geqslant i) \leqslant \dfrac{\alpha}{2}\right\}$ admet un minimum. On le note j_α.

e) Montrer que $j_\alpha > k+1$. En déduire que $Y_{n-j_\alpha+1} \leqslant Y_{j_\alpha}$ presque sûrement.

f) Déduire des questions précédentes que $\mathbb{P}\left(Y_{n-j_\alpha+1} < \dfrac{\theta}{\sqrt{2}} \leqslant Y_{j_\alpha}\right) \geqslant 1-\alpha$.

g) En déduire un intervalle de confiance pour θ au niveau de confiance $1-\alpha$.
Vérifier que l'estimateur T_n prend presque sûrement ses valeurs dans cet intervalle.

h) Justifier que la fonction Python suivante prend en entrée α et n et renvoie le minimum de $E_{n,\alpha}$.

```
def minE(n,alpha):
    i=n+1
    p=1/2**n
    s=p
    while s<=alpha/2:
        i=i-1
        p=p*i/(n-i+1)
        s=s+p
    return i
```

i) *En Python, si* `A` *désigne une matrice réelle, alors la commande* `np.sort(A,0)` *renvoie une copie de* `A` *triée colonne par colonne, par ordre croissant (chaque colonne est triée indépendamment).*

Compléter le script Python ci-dessous afin que son exécution retourne un vecteur `MedEmp` contenant p réalisation de la médiane empirique Y_{k+1}.

```
k=50
n=2*k+1
N=10000
theta=2
alpha=0.05
X=.....#Une matrice de taile nxN de réalisations de X
Y=np.sort(X,0)#Tri colonne par colonne
j=minE(n,alpha)#Calcul de j_alpha

#Définition des bornes des intervalles de confiance :
aT=.....; bT=.....

#"Incertitude" : Largeur moyenne des deux intervalles de confiances :
LT=np.mean(bT-aT)

#"Niveau réel" : Proportion d'intervalles de confiances contenant theta :
pT=(np.sum(theta<=bT)-np.sum(theta<aT))/N

print("Estimation ponctuelle de theta par l'estimateur T_n : "
                +str(.....))
print("Intervalles de confiance obtenus avec l'estimateur T_n :")
print('Largeur moyenne : '+str(LT)+ ' et niveau réel : '+str(pT*100))
```

Commenter et comparer avec les estimateurs asymptotiques. Si on prend une petite valeur de k (après tout c'est un intervalle de confiance non asymptotique), qu'obtient-on ? Est-ce plus ou moins intéressant ?