TP de probabilités : simulation de variables aléatoires en Python

1 Rappels

On utilise la librairie `numpy` pour les fonctions mathématiques et les générateurs de nombres aléatoires. La librairie `numpy` dispose du module `random` qui contient le nécessaire pour générer des nombres aléatoires. En particulier la fonction `rand` de ce module génère un nombre uniformément au hasard sur $[0, 1]$ (à précision machine bien sûr...).

On utilise le module `matplotlib.pyplot` pour les graphiques et les histogrammes. On commencera donc un fichier python par :

```
import matplotlib.pyplot as plt
import numpy as np
from numpy.random import rand
```

Par exemple, `np.cos(np.pi/3)` renvoie la valeur de $\cos(\pi/3)$. La commande `rand()` génère un nombre aléatoire suivant une loi uniforme sur $[0, 1]$. La commande `rand(n)` génère n nombres aléatoires de loi uniforme sur $[0, 1]$ indépendants, sous forme d'un vecteur `np.array`. La fonction `sum` renvoie la somme d'un vecteur.

Exercice 1.1. Lancer la commande :

```
n=1000
S = sum(rand(n))/n
print(S)
```

Expliquer pourquoi cette valeur est proche de 0.5.

La commande `np.linspace(x,y,n)` génère un vecteur de n points équidistants sur le segment $[x, y]$ (le premier point est x et le dernier est y). Par exemple :

```
S = linspace(0,1,11)
print(S)
>>> [0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0]
```

Pour tracer une fonction (qui peut prendre des vecteurs, comme `np.cos`, `np.exp`, etc.) on adaptera le bout de code suivant, qui trace la fonction cos sur l'intervalle $[0, 4\pi]$:

```
T = np.linspace(0,4*np.pi,300)
X = np.cos(T)
plt.plot(T,X)
plt.show()
```

Pour tracer un histogramme, on utilise la fonction `hist` du module `matplotlib.pyplot`. Par exemple, pour créer un histogramme avec 50 bâtons d'un échantillon de 10000 instances d'une loi normale de paramètre $(2, 3)$, on utilise le bout de code suivant (on fera attention au fait que la fonction `normal` prend en argument l'écart-type et non la variance) :

```
X = np.random.normal(2,np.sqrt(3),10000)
plt.hist(X,50,density=True)
plt.show()
```

Le bout de code suivant permet de définir la densité d'une loi normale de paramètre (m, σ^2) :

```
m=2
sigma2 = 3
fnorm = lambda x:1/np.sqrt(2*np.pi*sigma2) * np.exp(-(x-m)**2/(2*sigma2))
```

Exercice 1.2. Que remarquez vous lorsque vous superposez l'histogramme d'un échantillon d'une loi normale et sa densité ? Expliquer.

2 Simulation de variables aléatoires à partir d'une loi uniforme

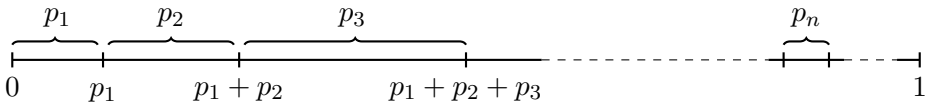
La machine dispose d'un générateur de nombres pseudo-aléatoires uniformes sur $[0, 1]$ (la fonction `rand`), ainsi que des générateurs pour d'autres lois (exponentielle, normale, binomiale, etc.). Comment faire si la loi que l'on cherche à simuler n'est pas une loi usuelle ?

2.1 Loi discrète

On suppose que la variable aléatoire X est à valeurs discrètes dans $x_1, x_2, \ldots$ (il peut y avoir une infinité de valeurs). On note :

$$p_n = \mathbb{P}(X = x_n)$$

On découpe le segment $[0, 1]$ en segments de taille $(p_n)_{n\geq 0}$:



Soit U une loi uniforme sur $[0, 1]$. On note Y la variable aléatoire défini par :

$$Y = \begin{cases} x_1 \text{ sur l'évènement } \{0 \leq U < p_1\} \\ x_2 \text{ sur l'évènement } \{p_1 \leq U < p_1 + p_2\} \\ \ldots \\ x_n \text{ sur l'évènement } \left\{\sum_{k=1}^{n-1} p_k \leq U < \sum_{k=1}^{n} p_k\right\} \\ \ldots \end{cases}$$

Alors on a :

$$\mathbb{P}(Y = x_n) = \mathbb{P}(X = x_n)$$

Autrement dit X et Y on la même loi. Pour simuler la variable aléatoire X il suffit donc de simuler la variable aléatoire Y. Voici une fonction Python qui simule une loi de Bernoulli de paramètre p :

```
def Bernoulli(p) :
    u = rand()
    if u < 1-p :
        return 0
    else :
        return 1
```

Où encore plus court :

```
Bernoulli = lambda p : int(rand()<p)
```

Exercice 2.1. Écrire une fonction qui simule une variable aléatoire uniforme sur $\{1, \dots, n\}$ à l'aide de la fonction `rand`. On pourra faire une boucle ou alors être malin est utiliser l'équivalence :

$$\forall u \in [0,1], \forall k \in \{0, \dots, n-1\}, \qquad \frac{k}{n} \leq u < \frac{k+1}{n} \quad \iff \quad k = \lfloor nu \rfloor$$

Exercice 2.2. A l'aide d'une boucle, écrire une fonction qui simule une loi de Poisson de paramètre λ à l'aide de la fonction `rand`. Pour rappel, X suit une loi de Poisson de paramètre λ si :

$$\mathbb{P}(X = k) = \frac{\lambda^k e^{-\lambda}}{k!}$$

On pourra utiliser la ligne de code suivante :

```
proba_poisson = lambda lb, k : lb**k * np.exp(-lb)/np.math.factorial(k)
```

Qui renvoie la probabilité qu'une loi de Poisson de paramètre `lb` prenne la valeur `k`.

2.2 Loi à valeurs réelles

Soit X une variable aléatoire à valeurs réelles. On note F_X sa fonction de répartition définie par :

$$F_X(x) = \mathbb{P}(X \leq x)$$

On suppose que la fonction F est continue et strictement croissante sur $\mathbb{R}$ (ou $\mathbb{R}_+$ si la variable aléatoire est à valeurs positives). Dans ce cas, la fonction F est une bijection croissante de $\mathbb{R}$ (ou $\mathbb{R}_+$) dans l'intervalle $]0,1[$. Notons F^{-1} son inverse, et U une variable aléatoire uniforme sur $]0,1[$. Puisque :

$$F^{-1} : \ ]0,1[\ \longrightarrow \ \mathbb{R} \ \text{ ou } \ \mathbb{R}_+$$

cela a un sens de considérer la variable aléatoire $Y = F^{-1}(U)$. Calculons la loi de Y :

$$\begin{aligned} \mathbb{P}(Y \leq x) &= \mathbb{P}(F^{-1}(U) \leq x) \\ &= \mathbb{P}(U \leq F(x)) \\ &= F(x) \end{aligned}$$

Où la dernière ligne est justifiée par le fait que U suit une loi uniforme sur $]0,1[$. Ainsi, X et Y ont la même loi. Pour simuler la variable aléatoire X il suffira donc de simuler la variable aléatoire $Y = F^{-1}(U)$. Il faut donc être capable de calculer l'inverse de F ce qui n'est pas toujours possible (penser à une loi normale). Il arrive donc que l'on doive calculer numériquement cet inverse.

Par exemple, considérons une variable aléatoire X à valeurs positives, de fonction de répartition :

$$F : x \mapsto 1 - \frac{1}{x^2}$$

Dans ce cas, son inverse est donné par :

$$F^{-1} : u \mapsto \frac{1}{\sqrt{1-u}}$$

La fonction Python suivante permet de simuler la loi de X :

```
X_sim = lambda : 1/np.sqrt(1-rand())
```

Dans le cas où F n'est plus continue ni strictement croissante, il est encore possible de construire un *inverse généralisé* de F pour appliquer la même méthode, mais nous n'en parlerons pas ici.

Exercice 2.3. Calculer l'inverse de la fonction de répartition d'une loi exponentielle de paramètre λ, puis simuler cette loi à l'aide de la fonction `rand`. Pour rappel, on a :

$$F(x) = 1 - e^{-\lambda x}$$

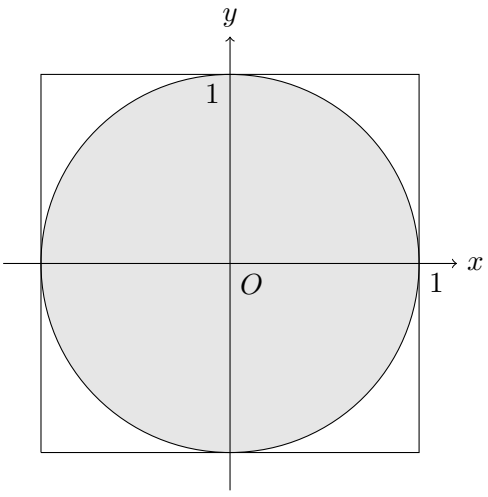
Exercice 2.4. On cherche a simuler une loi de Cauchy, dont la densité est donnée par :

$$f = \frac{1}{\pi}\frac{1}{1+x^2}$$

Tracer la densité f. Calculer la fonction de répartition de d'une loi de Cauchy, puis son inverse. Simuler la variable aléatoire X à l'aide de la fonction `rand`.

3 Calcul d'aire par la méthode de Monte-Carlo

Le but est d'approximer l'aire d'un disque D de rayon 1, c'est-à-dire π.



Pour cela, on génère N points au hasard suivant une loi uniforme dans le carré $C = [-1, 1] \times [-1, 1]$. La ligne de commande suivante permet de générer un tel point.

```
P = 2*rand(2)-1
print(P)
>>> [-0.64195673, 0.25580827]
```

On compte le nombre n de points qui tombent dans le cercle de rayon 1. Pour rappel, un point de coordonnées (x, y) se situe dans le cercle de rayon 1 ssi $x^2 + y^2 \leq 1$. Lorsque N est grand, la loi des grands nombres nous affirme que :

$$4\frac{n}{N} \simeq 4\frac{\mathcal{A}(D)}{\mathcal{A}(C)} = \pi$$

Exercice 3.1. Vérifier numériquement ce résultat avec $N = 10, 100, 1000, 10000, 100000$ et 1000000.

D'après le théorème central limite, la vitesse de convergence est en $\sqrt{N}$, c'est-à-dire que :

$$4\frac{n}{N} - \pi \simeq O\left(\frac{1}{\sqrt{N}}\right)$$

Exercice 3.2. Vérifier numériquement ce résultat. Combien de simulations faut-il faire pour espérer avoir 6 chiffres significatifs sur les décimales de π ?

Exercice 3.3. Calculer numériquement l'aire de :

$$E = \left\{(x, y) \in \mathbb{R} \;\middle|\; x^2 + y^4 + |\arctan(xy)| \leq 1\right\}$$

On utilisera le fait que $E \subset [-1, 1] \times [-1, 1]$. On pourra utiliser la ligne de commande :

```
f = lambda p: p[1]**2 + p[2]**2 + np.abs(np.arctan(p[1]*p[2]))
```

Qui définit la fonction $f : (x, y) \mapsto x^2 + y^4 + |\arctan(xy)|$. L'argument `p` est un vecteur coordonnées de taille 2.

Exercice 3.4 (Bonus)**.** Par une approche similaire, écrire une fonction qui donne le volume approché à 3 décimales près de la boule de rayon 1 dans $\mathbb{R}^3$, ou plus généralement dans $\mathbb{R}^n$ pour $n \geq 2$. Pour rappel, la boule B_n est définie par :

$$B_n = \left\{(x_1, x_2, \ldots, x_n) \in \mathbb{R}^n \;\middle|\; x_1^2 + x_2^2 + \ldots + x_n^2 \leq 1\right\}$$

On pourra comparer avec les valeurs exactes :

$$B_{2n} = \frac{\pi^n}{n!} \qquad \text{et} \qquad B_{2n+1} = \frac{2(n!)(4\pi)^n}{(2n+1)!}$$

4 Marche aléatoire et théorème central limite

On simule une marche aléatoire sur $\mathbb{Z}$. Pour cela on note $(X_n)_{n\geq 1}$ une suite de variables aléatoires iid telles que :

$$\mathbb{P}(X_k = -1) = \mathbb{P}(X_k = 1) = \frac{1}{2}$$

Et on note :

$$S_n = X_1 + X_2 + \ldots + X_n$$

On a évidemment $S_{n+1} = S_n + X_{n+1}$. Autrement dit à chaque instant, on monte ou on descend d'un cran, avec probabilité 1/2. On pourra utiliser la fonction suivante :

```
step = lambda n : 2*(rand(n)<0.5)-1
L = step(10)
>>> [1, 1, -1, 1, -1, -1, -1, 1, -1, 1]
```

qui génère une instance du vecteur aléatoire $(X_1, \dots, X_n)$. On pourra également utiliser la fonction `np.cumsum` qui prend en argument un vecteur et renvoie la somme cumulée des éléments du vecteur. Par exemple :

```
L = np.cumsum([1,2,3,4,5,6,7,8,9,10])
print(L)
>>> [1, 3, 6, 10, 15, 21, 28, 36, 45, 55]
```

Exercice 4.1. Ecrire une fonction qui renvoie une instance du vecteur $(S_1, \dots, S_n)$. Tracer la variable aléatoire S_n pour n entre 1 et 100. Montrer numériquement que :

$$\lim_{n \to +\infty} \frac{S_n}{n} = 0$$

On pourra utiliser le fait que si a et b sont deux vecteurs `np.array` de même longueur, la commande `a/b` renvoie la division coordonnées par coordonnées de `a` et `b`. Par exemple :

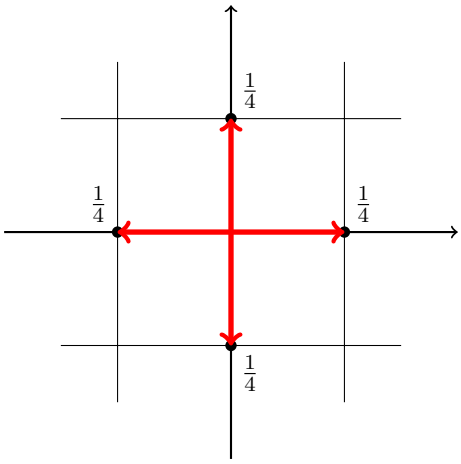
```
a = np.array([3,10,8])
b = np.array([2,5,3])
c = a/b
print(c)
>>> [1.5, 2, 2.6666666666666665]
```

On a $\mathrm{Var}(X_n) = 1$. D'après le TCL :

$$\frac{S_n}{\sqrt{n}} \underset{n \to +\infty}{\Longrightarrow} \mathcal{N}(0,1)$$

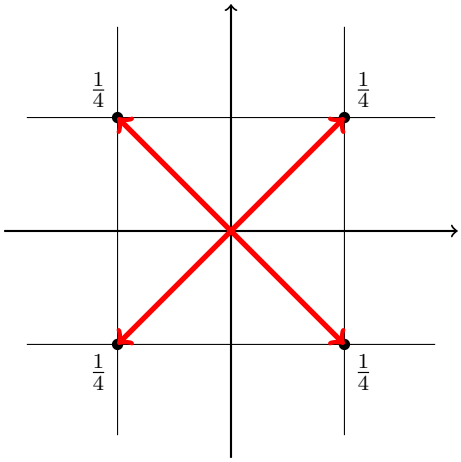
Exercice 4.2. Mettre en évidence le TCL en traçant l'histogramme d'un échantillon de 1000 instances de la variable aléatoire $S_n/\sqrt{n}$ pour $n = 10000$. Superposer à l'histogramme la densité d'une loi normale centrée réduite.

Il est possible de faire des marches aléatoires en dimension 2 sur $\mathbb{Z}^2$. Par exemple, on peut décider d'aller avec probabilité $1/4$ d'aller vers le haut, le bas, la gauche, ou la droite.



On peut également prendre deux marches aléatoires $(S_n)_{n \geq 1}$ et $(T_n)_{n \geq 1}$ sur $\mathbb{Z}$ indépendantes. Alors le couple de variables aléatoires $(S_n, T_n)_{n \geq 1}$ est une marche aléatoire sur :

$$\left\{(n,m) \in \mathbb{Z}^2 \;\middle|\; n+m \text{ est pair}\right\} \subset \mathbb{Z}^2$$



La deuxième méthode est plus rapide à simuler.

Exercice 4.3. Simuler une marche aléatoire dans $\mathbb{Z}^2$ avec la deuxième méthode. Pour cela on simulera deux marches aléatoires $(S_n)_{n\geq 1}$ et $(T_n)_{n\geq 1}$ sur $\mathbb{Z}$ indépendantes, et on tracera la marche (S_n, T_n) pour n entre 1 et 100, puis pour n entre 1 et 10000.

On note D_n la distance du point (S_n, T_n) à l'origine. Montrer que :

$$\mathbb{E}[D_n^2] = 2n$$

Le vérifier numériquement pour quelques valeurs de n par la loi des grands nombres. Vérifiez numériquement que pour n suffisamment grand on a l'équivalent :

$$\mathbb{E}[D_n] \simeq 2\sqrt{\frac{2n}{\pi}}$$

Exercice 4.4 (Bonus)**.** Soit $(X_n)_{n\geq 1}$ une suite de variables aléatoires iid de loi de Cauchy. On note :

$$S_n = X_1 + X_2 + \ldots + X_n$$

Montrer numériquement pour quelques valeurs de n que la variable aléatoire :

$$\frac{S_n}{n}$$

suit une loi de Cauchy. Mettre en évidence numériquement le fait que la loi de Cauchy ne vérifie pas la loi des grands nombres. Expliquer pourquoi.