I - Simulation des lois usuelles à densité

I.1 - Commandes Python

Python

La bibliothèque `"numpy.random"` permet de simuler les lois usuelles à densités.
Il faut d'abord importer cette bibliothèque : `"import numpy.random as rd"`.

- Loi uniforme :
  - ⋆ La commande `"rd.random()"` (ou `"rd.rand()"`) permet de simuler la loi uniforme sur [0, 1].
  - ⋆ La commande `"rd.random(n)"` permet de créer un vecteur contenant `n` simulations de la loi uniforme sur [0, 1].
  - ⋆ La commande `"rd.random([n,p])"` permet de créer une matrice `(n,p)` dont les coefficients sont des simulations de la loi uniforme sur [0, 1].
- Loi exponentielle :
  - ⋆ La commande `"rd.exponential(1/a)"` permet de simuler la loi exponentielle de paramètre `"a"`.
    Attention : : le paramètre "1/a" dans la commande correspond à l'espérance.
  - ⋆ La commande `"rd.exponential(1/a, n)"` permet de créer un vecteur contenant `n` simulations de la loi exponentielle de paramètre `"a"`.
  - ⋆ La commande `"rd.exponential(1/a, [n,p])"` permet de créer une matrice `(n,p)` dont les coefficients sont des simulations de la loi exponentielle de paramètre `"a"`.
- Loi normale :
  - ⋆ La commande `"rd.normal(mu, sigma)"` permet de simuler la loi normale de paramètre `mu` et `sigma`^{2}.
    Attention : le second paramètre dans la commande correspond à l'écart-type.
  - ⋆ La commande `"rd.normal(mu, sigma,n)"` permet de créer un vecteur contenant `n` simulations de la loi normale de paramètre `mu` et `sigma`^{2}.
  - ⋆ La commande `"rd.normal(mu, sigma, [n,p])"` permet de créer une matrice `(n,p)` dont les coefficients sont des simulations la loi normale de paramètre `mu` et `sigma`^{2}.

I.2 - Histogrammes

Un histogramme permet de visualiser la répartition des valeurs d'un échantillon E entre différentes classes (intervalles). Plus précisément, si les valeurs de E sont comprises dans un intervalle $[a, b]$, on découpe cet intervalle en intervalles disjoints appelés classes (souvent de même longueur) et on compte le nombre d'éléments de E dans chaque classe.

Python

Pour tracer des histogrammes, on importe d'abord la bibliothèque : `"import matplotlib.pyplot as plt"`.

- La commande `"plt.hist(L,n)"` trace l'histogramme des valeurs contenues dans la liste `L` en `n` classes de même longueurs (Python crée lui même les intervalles correspondants aux classes).
- La commande `"plt.hist(L,C)"` trace l'histogramme des valeurs contenues dans la liste `L`, les différentes intervalles des classes étant contenues dans la liste `C`.
- En ajoutant comme troisième argument `"density=True"` à la commande `"plt.hist"`, on trace les fréquences à la place des effectifs.

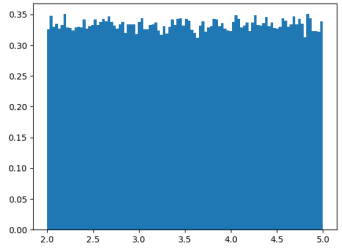
Exemple 1

On importe les bibliothèques.

```
1 import numpy as np
2 import numpy.random as rd
3 import matplotlib.pyplot as plt
```

- Loi uniforme : $X \hookrightarrow \mathcal{U}(2,5)$.

```
1 n = 100000
2 X = 3*rd.random(n)+2
3 C = np.linspace (2 ,5 ,100)
4 plt.hist(X,C,density=True)
5 plt.show()
```

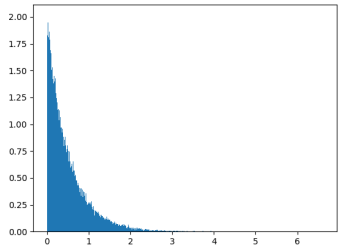


Remarque : On peut aussi simuler `X` avec une boucle for.

```
1 X = np.zeros(n)
2 for i in range(n) :
3     X[i] = 3*rd.rand()+2
```

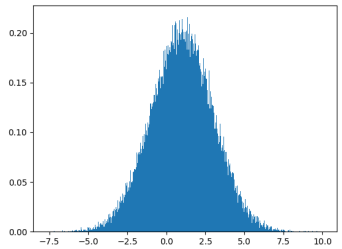
- Loi exponentielle : $Y \hookrightarrow \mathcal{E}(2)$.

```
1 n = 100000
2 Y = rd.exponential(1/2,n)
3 plt.hist(Y,100,density=True)
4 plt.show()
```



- Loi normale : $Z \hookrightarrow \mathcal{N}(1,4)$.

```
1 n = 100000
2 Z = rd.normal(1,2,n)
3 plt.hist(Z,1000,density=True)
4 plt.show()
```



II - Simulation de variables aléatoires à densité

Exercice 1

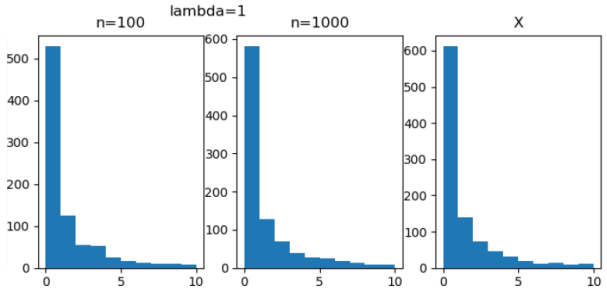
Soit Y une variable aléatoire suivant la loi exponentielle de paramètre 2.

Soit $\lambda > 0$. On pose : $X = \left(\frac{Y}{\lambda}\right)^2$ et on admet que X est une variable aléatoire à densité.

Soit $(X_n)_{n\in\mathbb{N}^*}$ une suite de variables aléatoires indépendantes et de même loi que X.

On pose pour tout $n \in \mathbb{N}^*$: $M_n = \min(X_1, \dots, X_n)$ et $J_n = n^2 M_n - \frac{1}{n}$ et on admet que M_n et J_n sont des variables aléatoires à densité.

1. (a) Écrire une fonction Python d'entête `"def VarX(n,lamb) :"` afin qu'elle renvoie un tableau `numpy` contenant `n` réalisations de la variable aléatoire X (`lamb` représente le paramètre λ).
   (b) On prend $\lambda = 1$. Écrire une fonction Python d'entête `"def VarJ(n) :"` qui renvoie une liste de 1000 valeurs simulant la variable aléatoire J_n.
2. On prend $\lambda = 1$. On donne ci-dessous les histogrammes des variables aléatoires J_{100}, J_{1000} et X créés à partir de 1000 valeurs réparties entre les classes $[k,\ k+1[)_{k\in[\![0;9]\!]}$.



   (a) Quelle programme permet d'obtenir le premier histogramme ?
   (b) Que peut-on conjecturer ?

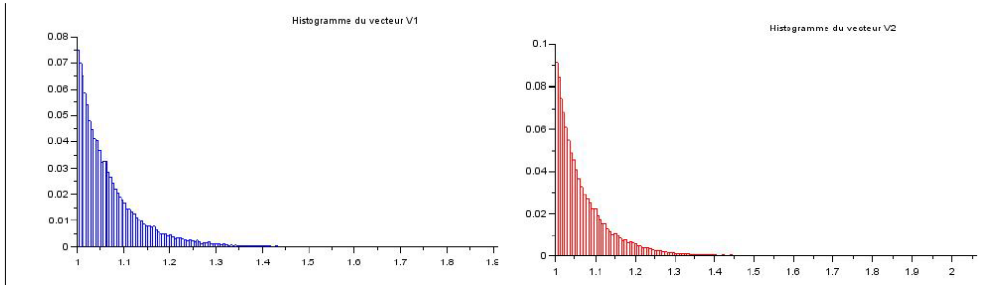
Exercice 2 (*Loi de Pareto*)

Soient a et b deux réels strictement positifs. On considère une variable aléatoire X admettant pour fonction de répartition la fonction : $F_X : x \mapsto \begin{cases} 0 & \text{si } x < b \\ 1 - \left(\frac{b}{x}\right)^a & \text{si } x \geqslant b \end{cases}$.

On admet que X est une variable à densité. On dit que X suit la loi de Pareto de paramètres a et b.

1. Soit U une variable aléatoire suivant la loi uniforme sur $]0,\ 1[$.
   Montrer que la variable aléatoire $Y = bU^{-\frac{1}{a}}$ suit la loi de Pareto de paramètres a et b.
2. En déduire une fonction d'en-tête `"def Pareto(a,b) :"` qui prend en arguments deux réels `a` et `b` strictement positifs et qui renvoie une simulation de la variable aléatoire X.
3. On considère une suite de variables aléatoires $(Y_i)_{i\in\mathbb{N}^*}$, indépendantes et suivant toutes la même loi de Pareto de paramètres a et 1 et on pose pour tout entier n non nul : $Z_n = \min(Y_1, \dots, Y_n)$.
   Écrire une fonction d'en-tête `"def SimuZ(a,n) :"` qui simule la variable Z_n, les valeurs de `a` et `n` étant données en paramètre.
4. On considère le programme suivant et on donne les histogrammes associés. Que peut-on conjecturer ?

```
a = float(input("Donner la valeur de a : "))
n = int(input("Donner la valeur de n : "))
V1 = [Simu_Z(a,n) for i  in range(10000)]
V2 = [Pareto(a*n,1) for i  in range(10000)]
plt.hist(V1, 100, density=True)
plt.hist(V2, 100, density=True)
plt.show()
```

III - Méthode d'inversion

Exercice 3 (*Loi exponentielle*)

Soit $\lambda > 0$ et X une variable aléatoire suivant la loi exponentielle de paramètre λ.

1. (a) Rappeler la fonction de répartition F_X de X.
   (b) Montrer que la fonction F_X définit une bijection de $\mathbb{R}_+^*$ sur $]0;1[$ et déterminer l'expression de sa fonction réciproque.
2. (a) Soient U une variable aléatoire suivant la loi uniforme sur $]0,\ 1[$ et $Y = -\frac{1}{\lambda}\ln(1-U)$.
   Montrer que Y suit la même loi que X.
   (b) Simuler une réalisation d'une variable aléatoire X suivant la loi exponentielle de paramètre 2 à l'aide de la commande `"rd.rand()"`.
   (c) Quelle valeur approchée est affichée par le programme suivant ?

```
n = 1000
Y = np.zeros(n)
for i in range(n) :
    Y[i] = - 2 * np.log( 1 - rd.rand() )
print(np.mean(Y))
```

Exercice 4 (*Loi de Gumbel*)

On considère une variable aléatoire X suivant la loi de Gumbel, dont une densité est :

$$\forall x \in \mathbb{R}, \quad f(x) = \mathrm{e}^{-x}\mathrm{e}^{-\mathrm{e}^{-x}}$$

1. (a) Déterminer la fonction de répartition F_X de X.
   (b) Montrer que la fonction F_X est bijective et déterminer l'expression de sa fonction réciproque.
2. (a) Soit U une variable aléatoire suivant la loi uniforme sur $]0,\ 1[$ et $Y = -\ln(-\ln(U))$.
   Montrer que Y suit la même loi que X.
   (b) Écrire une fonction d'entête `"def LoiGumbel() :"` qui renvoie une réalisation d'une variable aléatoire X suivant la loi de Gumbel.
   (c) A l'aide de la fonction `"LoiGumbel"`, écrire un programme qui renvoie une approximation de l'espérance de la variable aléatoire X.

Exercice 5 (*Généralisation : Méthode d'inversion*)

Soit X une variable aléatoire à densité dont la fonction de répartition F définit une bijection de $]a;b[$ sur $]0;1[$.
Soit U une variable aléatoire suivant la loi uniforme sur $]0;1[$.
Montrer que la variable $Z = F^{-1}(U)$ suit la même loi que X.