TP 5

Simulation de variables aléatoires à densité

Exercice 1

1. Écrire une fonction `norcentreereduite()` simulant la loi $\mathcal{N}(0,1)$ à l'aide de la fonction `rd.random`.
2. Écrire une fonction `normale(m,s)` simulant la loi $\mathcal{N}(m,s^2)$ à partir de la fonction `norcentreereduite()`.
3. Écrire une fonction `Normale(m,s,N)` donnant un vecteur contenant N réalisations de la loi $\mathcal{N}(m,s^2)$.

1. En utilisant le premier point de la propriété précédente :

```
def norcentreereduite():
    u = rd.random(12)
    x = np.sum(u)-6
    return x
```

2. En utilisant le deuxième point de la propriété précédente :

```
def normale(m,s):
    x = norcentreereduite()
    y = s*x+m
    return y
```

3. On peut procéder ainsi :

```
def Normale (m,s,N):
    X = np.zeros(N)
    for k in range(N):
        X[k] = normale(m,s)
    return X
```

Exercice 2

1. Rappeler l'expression de la fonction de répartition de $U \hookrightarrow \mathscr{U}(]0,1[)$.
2. Démontrer le théorème précédent.

1. On a $F_U : x \in \mathbb{R} \mapsto \begin{cases} 0 & \text{si } x \leq 0 \\ x & \text{si } 0 < x < 1 \\ 1 & \text{si } x \geq 1 \end{cases}$.
2. Posons $Y = F^{-1}(U)$. On a $Y(\Omega) = F^{-1}(]0,1[) =]a,b[= X(\Omega)$. Et pour tout $x \in]a,b[$, on a :

$$F_Y(x) = P(Y \leq x) = P(F^{-1}(U) \leq x) = P(U \leq F(x)) \underset{F(x)\in]0,1[}{=} F(x)$$

. Les fonctions de répartitions de X et Y sont égales : elles suivent donc la même loi.

Exercice 3

1. Rappeler l'expression de la fonction de répartition d'une loi exponentielle, et montrer qu'elle réalise une bijection de $\mathbb{R}_+^*$ sur $]0,1[$.

   Déterminer sa bijection réciproque.

2. (a) Écrire une fonction `exponentielle(lambda)` simulant une loi $\mathscr{E}(\lambda)$ à partir de la fonction `rd.random()`.

   (b) Écrire une fonction `Exponentielle(lambda,N)` donnant un échantillon de taille N de la loi $\mathscr{E}(\lambda)$.

3. (a) Créer un vecteur de taille 10000 contenant 10000 simulations d'une variable aléatoire suivant la loi $\mathscr{E}(1/2)$.

   (b) En utilisant les commandes `np.mean` et `np.std` de la librairie `numpy`, vérifier que la moyenne et l'écart-type empiriques (c'est-à-dire de ce vecteur) sont bien conformes à ce qu'on attend.

4. (a) Écrire une fonction `gamma(n)` simulant la loi $\gamma(n)$ pour tout $n \in \mathbb{N}^*$.

   (b) Écrire une fonction `Gamma(n,N)` renvoyant un vecteur contenant N réalisations de la loi $\gamma(n)$.

1. Notons F la fonction de répartition de la loi $\mathscr{E}(\lambda)$. On a :

$$F : x \in \mathbb{R} \mapsto \begin{cases} 0 & \text{si } x \leq 0 \\ 1 - e^{-\lambda x} & \text{si } x > 0 \end{cases}.$$

   F est continue sur $\mathbb{R}_+^*$, strictement croissante sur cet intervalle (car F est dérivable et $F'(x) = \lambda e^{-\lambda x} > 0$) et on a $\lim\limits_{0^+} F = 0$ et $\lim\limits_{+\infty} F = 1$. Par le théorème de la bijection, F réalise une bijection de $\mathbb{R}_+^*$ dans $]0,1[$.

   Déterminons F^{-1} :$]0,1[\to \mathbb{R}_+^*$. Soit pour cela $x \in \mathbb{R}_+^*$ et $y \in]0,1[$. On résout :

$$\begin{aligned} y = F(x) &\Leftrightarrow y = 1 - e^{-\lambda x} \Leftrightarrow e^{-\lambda x} = 1 - y \\ &\Leftrightarrow x = -\frac{1}{\lambda}\ln(1-y) \end{aligned}$$

   Ainsi on a $F^{-1} : y \in]0,1[\to -\frac{1}{\lambda}\ln(1-y)$.

2. (a) En utilisant le principe d'inversion :

```
def exponentielle(lbd):
    u = rd.random()
    x = -(1/lbd)*np.log(1-u)
    return x
```

   (b) En adaptant la fonction `Normale` ici :

```
def Exponentielle(lbd,N):
    X = np.zeros(N)
    for k in range(N):
        X[k] = exponentielle(lbd)
    return X
```

3. (a) On utilise les lignes de codes suivantes :

```
E = Exponentielle(0.5,10000)
print(np.mean(E),np.std(E))
```

(b) Rappelons que les commandes `np.mean` et `np.std` calculent respectivement la moyenne et l'écart-type d'une série statistique. On obtient ici `2.0344615` et `2.0258905`. C'est bien conforme à ce qu'on attend, puisque l'espérance et la variance d'une loi $\mathscr{E}(\lambda)$ sont égales à $\dfrac{1}{\lambda}$, c'est-à-dire 2 ici.

4. (a) Rappelons pour commencer que si $X_1, \dots, X_n$ sont indépendantes et de même loi $\mathscr{E}(1) = \gamma(1)$, alors $X_1 + \cdots + X_n$ suit une loi $\gamma(n)$ par stabilité de la loi gamma. En utilisant ce résultat, on en déduit la fonction suivante pour simuler une loi $\gamma(n)$:

```
def gamma(n):
    X = Exponentielle(1,n)
    y = np.sum(X)
    return y
```

(b) En adaptant toujours le même programme que dans `Normale` ou `Exponentielle` :

```
def Gamma(n,N):
    X = np.zeros(N)
    for k in range(N):
        X[k] = gamma(n)
    return X
```

Exercice 4

On rappelle qu'une variable X suit une loi de Cauchy si elle admet pour fonction de répartition la fonction :

$$F : x \in \mathbb{R} \mapsto \frac{1}{\pi}\left(\arctan(x) + \frac{\pi}{2}\right).$$

Une densité de X est alors la fonction $f : t \mapsto \dfrac{1}{\pi}\dfrac{1}{1+t^2}$.

1. Montrer que F réalise une bijection de $\mathbb{R}$ sur $]0,1[$, et déterminer F^{-1}.
2. (a) Écrire une fonction `cauchy()` simulant une loi de Cauchy à partir de la fonction `rd.random`.
   (b) Écrire une fonction `Cauchy(N)` donnant un échantillon de taille N de la loi de Cauchy.
3. (a) Créer un vecteur de taille 10000 contenant 10000 simulations d'une variable aléatoire suivant la loi de Cauchy.
   (b) Utiliser la commande `np.mean` avec ce vecteur pour évaluer l'espérance d'une loi de Cauchy. Recommencer avec plusieurs échantillons. Que constatez-vous ? Une variable suivant la loi de Cauchy admet-elle une espérance ?

1. F est continue et strictement croissante car la fonction arctangente l'est, et on a $\lim\limits_{-\infty} F = 0$ et $\lim\limits_{+\infty} F = 1$. Par le théorème de la bijection, F réalise une bijection de $\mathbb{R}$ dans $]0,1[$.

Déterminons $F^{-1} :]0,1[\to \mathbb{R}$. Soit pour cela $x \in \mathbb{R}$ et $y \in]0,1[$. On résout :

$$y = F(x) \Leftrightarrow y = \frac{1}{\pi}\left(\arctan(x) + \frac{\pi}{2}\right) \Leftrightarrow \pi y - \frac{\pi}{2} = \arctan(x)$$
$$\Leftrightarrow x = \tan\left(\pi y - \frac{\pi}{2}\right)$$

Ainsi on a $F^{-1} : y \in]0,1[\to \tan\left(\pi y - \frac{\pi}{2}\right)$.

2. (a) En utilisant la méthode d'inversion, on obtient :

```
def cauchy():
    u = rd.random()
    c = np.tan(np.pi*u-np.pi/4)
    return c
```

(b) Toujours sur le même principe :

```
def Cauchy(N):
    C = np.zeros(N)
    for k in range(N):
        C[k] = cauchy()
    return C
```

3. (a) On utilise la commande :

```
C = Cauchy(10000)
```

(b) On utilise :

```
print(np.mean(C))
```

On obtient `0.3452672`. En exécutant plusieurs fois ce code, on obtient `- 1.7578416`, `13.079537`, ... Les valeurs obtenues sont très dispersées, alors qu'on pourrait s'attendre à ce qu'elles soient voisine d'une valeur qui serait l'espérance d'une loi de Cauchy. Cela suggère qu'une variable suivant une loi de Cauchy n'admet pas d'espérance. On peut vérifier que c'est effectivement le cas à l'aide de la fonction de répartition (faites le !).

Exercice 5

1. Simuler avec la fonction `Exponentielle` $N = 10000$ valeurs de la loi $\mathscr{E}(0.5)$.
2. Tracer la courbe représentative de la densité f de la loi $\mathscr{E}(0.5)$.
3. Tracer l'histogramme des fréquences de l'échantillon obtenu (on prendra pour cela une subdivision c de l'intervalle $[0,10]$ en $p = 100$ intervalles de même longueur).

   Comparer l'histogramme des fréquences de l'échantillon à la courbe représentative de f. Qu'en pensez vous ?
4. Procéder de même pour la loi $\gamma(3)$.

On procède ainsi :

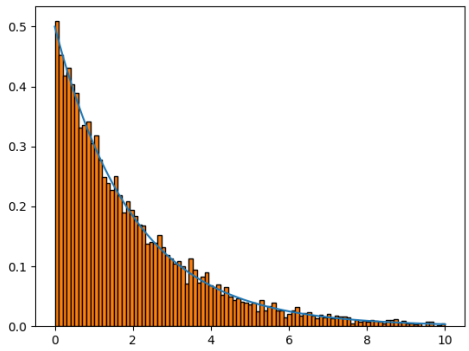
```
# Echantillon
N = 10000
lbd = 1/2
x = Exponentielle(lbd,N)

# densité
def f(t):
    y = lbd*np.exp(-lbd*t)
    return y
t = np.linspace(0,10,100)
y = f(t)
plt.plot(t,y)

# histogramme des fréquences
c = np.linspace(0,10,100)
plt.hist(x,c,density='True',edgecolor =
 'k')

plt.show()
```

On obtient :



On observe graphiquement que les aires des rectangles formant l'histogramme des fréquences sont proches des aires délimitées par la courbe représentative de la densité. On peut donc conclure que `Exponentielle(0.5)` renvoie bien des simulations suivant la loi $\mathscr{E}(0.5)$.

Procédons de même avec la loi $\gamma(3)$.

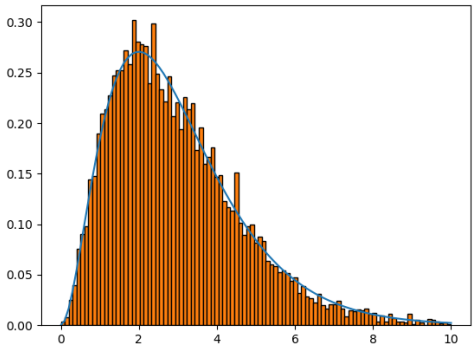
```
# Echantillon
N = 10000
n = 3
x = Gamma(n,N)

# densité
def f(t):
    y = (t**2)*np.exp(-t)/2
    return y
t = np.linspace(0,10,100)
y = f(t)
plt.plot(t,y)

# histogramme des fréquences
c = np.linspace(0,10,100)
plt.hist(x,c,density='True',edgecolor =
 'k')

plt.show()
```

On obtient :



On observe là aussi que les aires des rectangles formant l'histogramme des fréquences sont proches des aires délimitées par la courbe représentative de la densité : la simulation de la loi $\gamma(3)$ par la commande `gamma(3)` est donc pertinente.

Exercice 6

1. Expliquer les lignes de commandes proposées pour tracer la fonction de répartition empirique.
2. Simuler avec la fonction `Exponentielle` $N = 10000$ réalisations de la loi $\mathscr{E}(1)$.
3. Tracer la fonction de répartition empirique de l'échantillon obtenu et la fonction de répartition théorique de cette loi. Comparer.

1. Le vecteur `u` contient 100 éléments en progression arithmétique, de la plus petite valeur de l'échantillon `x` à sa plus grande valeur. À chaque passage dans la boucle `for`, on attribue à la k-ème composante du vecteur `v` la fréquence des éléments de l'échantillon `x` qui sont plus petites que `u[k]`. Cela devrait correspondre approximativement à la probabilité que la variable X soit plus petite que `u[k]`. La commande `plt.plot((u,v)` trace donc bien la

fonction de répartition empirique de l'échantillon `x`.

2. On le fait avec la commande suivante :

```
# Echantillon
N = 10000
lbd = 1
x = Exponentielle(lbd,N)
```

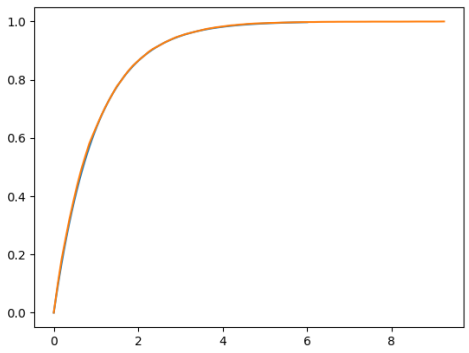
3. On utilise le code suivant :

```
# Fct de rép théorique
def F(x):
    y = 1-np.exp(-lbd*x)
    return y
u = np.linspace(0,6,100)
v = F(u)
plt.plot(u,v)

# Fct de rép empirique
n = 100
u = np.linspace(np.min(x),np.max(x),n)
v = np.zeros(n)
for k in range(n):
    v[k] = np.mean(x <= u[k])
plt.plot(u,v)

plt.show()
```

On obtient :



Ces fonctions sont très proches l'une de l'autre, notre simulation semble donc pertinente.

Exercice 7

1. Calculer $\Phi(0), \Phi(1), \Phi(1.96)$.
2. Soit X une variable aléatoire suivant la loi $\mathcal{N}(3, 2^2)$. Calculer $P(X > 10)$, $P(0 \leq X < 3)$.
3. Tester les simulations obtenues des lois $\mathcal{N}(0, 1)$ et $\mathcal{N}(2, 3^2)$ en traçant les fonctions de répartition théoriques et empiriques.
4. Comparer ces résultats avec ceux obtenus par la fonction `rd.normal`.

1. On utilise le code suivant :

```
import scipy.special as sp
p1 = sp.ndtr(0)
p2 = sp.ndtr(1)
p1 = sp.ndtr(1.96)
print(p1,p2,p3)
```

   On obtient respectivement les valeurs `0.5`, `0.8413447` et `0.9750021`.

2. On a :

$$P(X > 10) = 1 - P(X \leq 10) = 1 - P(X - 3 \leq 7) = 1 - P\left(\frac{X-3}{2} \leq \frac{7}{2}\right) = 1 - \Phi(3.5)$$

   À l'aide de la commande :

```
p = sp.ndtr(3.5)
disp(1-p)
```

   on obtient que cette probabilité vaut `0.0002326290790355401` . On procède de même pour l'autre probabilité.

3. Pour la loi $\mathcal{N}(0,1)$, avec un échantillon de taille $N = 10000$:

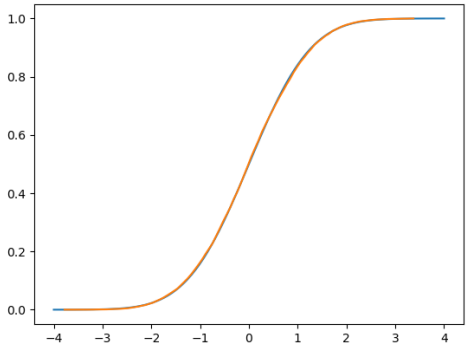
```
# Echantillon
N = 10000
m = 0
s = 1
x = Normale(m,s,N)

# Fct de rép théorique
import scipy.special as sp
u = np.linspace(-4,4,100)
v = sp.ndtr(u)
plt.plot(u,v)

# Fct de rép empirique
n = 100
u = np.linspace(np.min(x),np.max(x),n)
v = np.zeros(n)
for k in range(n):
    v[k] = np.mean(x <= u[k])
plt.plot(u,v)

plt.show()
```

   On obtient :



Ainsi la simulation de la loi $\mathcal{N}(0,1)$ par la fonction `Normale` semble pertinente.

De même pour la loi $\mathcal{N}(2,9)$:

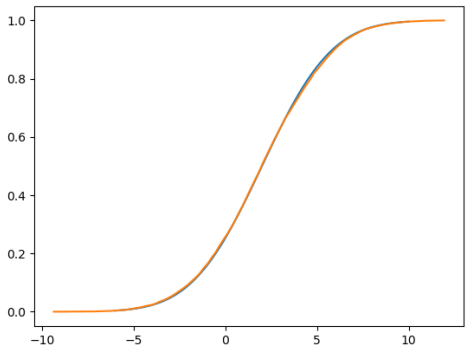
```
# Echantillon
N = 10000
m = 2
s = 3
x = Normale(m,s,N)

# Fct de rép théorique
import scipy.special as sp
u = np.linspace(-6,10,100)
v = sp.ndtr((u-m)/s)
plt.plot(u,v)

# Fct de rép empirique
n = 100
u = np.linspace(np.min(x),np.max(x),n)
v = np.zeros(n)
for k in range(n):
    v[k] = np.mean(x <= u[k])
plt.plot(u,v)

plt.show()
```

On obtient :



Là aussi, notre simulation semble pertinente.

4. Avec la fonction `rd.normal` :

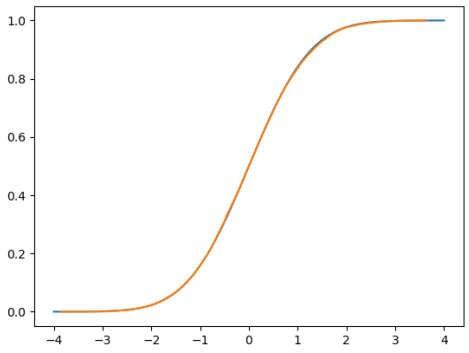
```
# Echantillon
N = 10000
m = 0
s = 1
x = rd.normal(m,s,N)

# Fct de rép théorique
import scipy.special as sp
u = np.linspace(-4,4,100)
v = sp.ndtr(u)
plt.plot(u,v)

# Fct de rép empirique
n = 100
u = np.linspace(np.min(x),np.max(x),n)
v = np.zeros(n)
for k in range(n):
    v[k] = np.mean(x <= u[k])
plt.plot(u,v)

plt.show()
```

On obtient :



Les résultats obtenus avec `rd.normal` sont similaires de ceux obtenus avec la fonction `Normale`.

Exercice 8 (★★)

Soit $a \in \mathbb{R}_+^*$ et $(X_1, \dots, X_n)$ une famille de variables aléatoires indépendantes, identiquement distribuées suivant une loi uniforme sur $[0, a]$. On pose :

$$U = \min(X_1, \dots, X_n) \quad \text{et} \quad V = \max(X_1, \dots, X_n).$$

1. Déterminer les lois de U et V.

2. On considère la fonction suivante :

```
def simulation(a,n):
```

```
    nb_sim = 10000
    R = a*rd.random([nb_sim,n])
    couple = np.array([np.min(R,0), np.max(R,0)])
    return couple
```

Expliquer la fonction `simulation`.

3. Prenons $n = 2$ et $a = 1$. Comparer graphiquement la qualité de cette simulation (fonction de répartition empirique/théorique).

Exercice 9 (★★★ - Différentes simulations de la loi de Pareto)

Soit $k \in \mathbb{N}^*$ et $\lambda > 0$.

1. Déterminer la valeur de r pour laquelle $f_\lambda : x \to \begin{cases} 0 & \text{si } x \leq \lambda \\ \dfrac{r}{x^{k+1}} & \text{sinon} \end{cases}$ est une densité de probabilité.

   Si X admet pour densité f_λ, on dit que X suit la loi de Pareto de paramètre λ et k.

2. Déterminer la fonction de répartition d'une variable suivant la loi de Pareto de paramètres λ et k.

3. En utilisant la méthode d'inversion, simuler une variable aléatoire suivant une loi de Pareto de paramètres λ et k.

4. Soient $X_1, \cdots, X_k$ k variables aléatoires indépendantes suivant toutes la loi uniforme sur $]0, 1]$. On pose alors $Y = \dfrac{\lambda}{\max(X_1, \cdots, X_k)}$.

   (a) Montrer que Y suit une loi de Pareto de paramètres λ et k.

   (b) En déduire une autre méthode pour simuler la loi de Pareto.

5. Comparons à présent les deux méthodes obtenues de simulation d'une loi de Pareto.

   (a) En simulant $N = 100000$ réalisations d'une loi de Pareto à l'aide de chacune de ces méthodes, déterminer laquelle des deux est la plus rapide.

   On pourra à cet effet utiliser la commande `time.clock()` *de la librairie* `time`*. Pour cela, on exécute la commande* `tps1 = time.clock()` *juste avant le début de la simulation, puis la commande* `tps2 = time.clock()` *juste après. La différence entre ces différents temps donnera le temps d'exécution de la portion de code encadrée (en secondes).*

   (b) Prenons $\lambda = 1$. Comparer graphiquement la qualité de chacune des deux simulations d'une loi de Pareto (histogramme des fréquences/densité théorique).
