TP 5

Simulation de variables aléatoires à densité

Exercice 1

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

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

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

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

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

3. On peut procéder ainsi :

```python
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 $F^{-1} : y \in]0,1[\to -\dfrac{1}{\lambda}\ln(1-y)$.

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

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

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

```python
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 :

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

(b) Toujours sur le même principe :

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

3. (a) On utilise la commande :

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

(b) On utilise :

```python
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.

Vérifions que c'est effectivement le cas. Une variable X suivant une loi de Cauchy admet une espérance si et seulement si l'intégrale $\displaystyle\int_{-\infty}^{+\infty} |t| f(t)\,\mathrm{d}t$ converge, soit par parité si et seulement si $I = \displaystyle\int_0^{+\infty} \frac{t}{\pi(1+t^2)}\,\mathrm{d}t$ converge. La fonction $t \in \mathbb{R}_+ \mapsto \dfrac{t}{\pi(1+t^2)}$ étant continue sur $[0,+\infty[$, l'intégrale I est généralisée en $+\infty$. Et on a :

- $\dfrac{t}{\pi(1+t^2)} \underset{t\to+\infty}{\sim} \dfrac{1}{\pi t}$.
- $\frac{1}{t} \geq 0$ pour tout $t > 0$.
- $\displaystyle\int_1^{+\infty} \frac{\mathrm{d}t}{t}$ diverge (intégrale de Riemann en $+\infty$ d'exposant $1 \leq 1$).

Par théorème de comparaison, l'intégrale I diverge, et $E(X)$ n'existe pas.

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 $\gamma(3)$.

1. On procède ainsi :

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

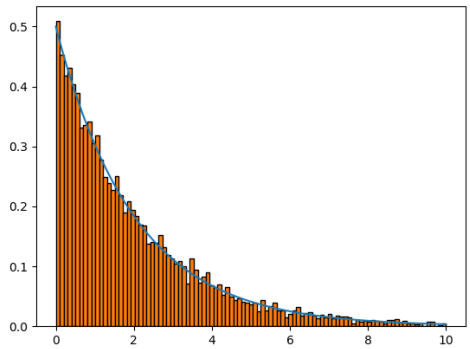
2. Traçons la courbe représentative de la fonction $f : t \in \mathbb{R}_+ \mapsto (1/2)\exp(-t/2)$.

```
# 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)
```

3. On utilise la commande `plt.hist` pour tracer l'histogramme des fréquences de l'échantillon `x`.

```
# 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 $\mathscr{E}(0.5)$.

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

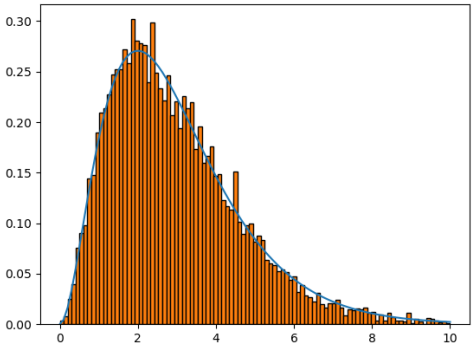
```
# Echantillon
N = 10000
n = 3
x = Gamma(n,N)

# 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. Expliquons les lignes de commandes suivantes permettant de tracer la fonction de répartition empirique associée à l'échantillon `x` de taille `N`.

```
plt.step(np.sort(x),np.arange(0,1,1/N))
plt.show()
```

Le vecteur `np.sort(x)` est de même longueur `N` que `x` et contient les valeurs (ou modalités) de `x` triées dans l'ordre croissant. Notons que l'échantillon `x` provenant d'une simulation d'une loi continue, aucune valeur n'apparait plusieurs fois (toutes les modalités sont d'effectif 1).

Le vecteur `np.arange(0,1,1/N)` est lui aussi de longueur `N`, et correspond à une progression arithmétique de 0 à 1 (exclu) de raison `1/N`.

La commande `plt.step(np.sort(x),np.arange(0,1,1/N))` trace alors la fonction G en escalier d'abscisse `np.sort(x)` et d'ordonnée `np.arange(0,1,1/N)`. Pour tout $x \in \mathbb{R}$, il existe $k \in [\![0, N-2]\!]$ tel que x soit compris entre la k-ème modalité y_k et la $(k+1)$-ème modalité y_{k+1} de `np.sort(x)`. Et $G(x)$ est alors égale à k/N, qui est la fréquence des modalités inférieures ou égales à x. G est donc bien la fonction de répartition empirique associée à l'échantillon `x`.

2. On le fait avec la commande suivante :

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

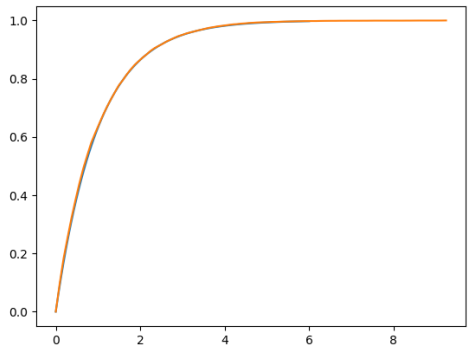
3. On utilise le code suivant :

```
# 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
plt.step(np.sort(x),np.arange(0,1,1/N))

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 $N = 10000$:

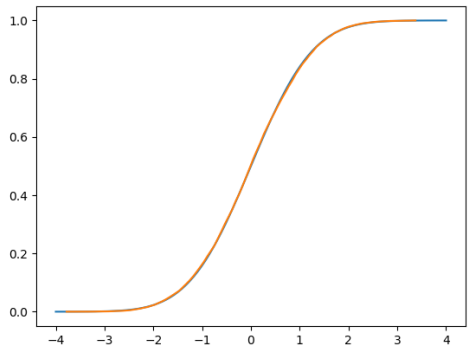
```
# Echantillon
N = 10000
m = 0
s = 1
x = Normale(m,s,N)

# 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
plt.step(np.sort(x),np.arange(0,1,1/N))

plt.show()
```

On obtient :



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

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

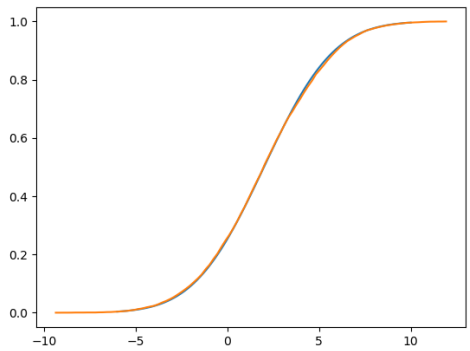
```
# Echantillon
N = 10000
m = 2
s = 3
x = Normale(m,s,N)

# 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
plt.step(np.sort(x),np.arange(0,1,1/N))

plt.show()
```

On obtient :



Là aussi, notre simulation semble pertinente.

4. Avec la fonction `rd.normal` :

```
# Echantillon
```

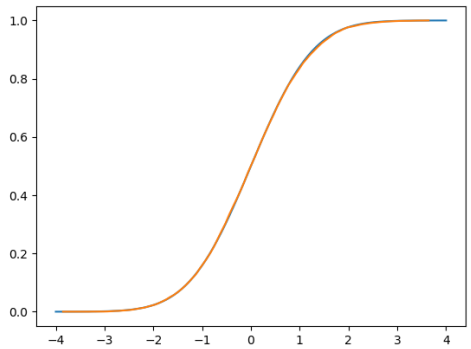
```
N = 10000
m = 0
s = 1
x = rd.normal(m,s,N)

# 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
plt.step(np.sort(x),np.arange(0,1,1/N))

plt.show()
```

On obtient :



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

Exercice 8 (★★)

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

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

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,1), np.max(R,1)])
    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).

1. Nous l'avons traité en TD. On obtient les fonctions de répartitions suivantes pour U et V respectivement :

$$F_U : x \in \mathbb{R} \mapsto \begin{cases} 0 & \text{si } x < 0 \\ 1 - \left(1 - \frac{x}{a}\right)^n & \text{si } x \in [0, a] \\ 1 & \text{si } x > a \end{cases} \quad \text{et} \quad F_V : x \in \mathbb{R} \mapsto \begin{cases} 0 & \text{si } x < 0 \\ \left(\frac{x}{a}\right)^n & \text{si } x \in [0, a] \\ 1 & \text{si } x > a \end{cases}.$$

2. La commande `a*rd.random([nb_sim,n])` crée une matrice à `np_sim = 10000` lignes et `n` colonnes dont chaque coefficient a été pris aléatoirement comme réalisation d'une loi uniforme continue sur $[0, a]$.

   La commande `np.min(R,1)` renvoie le minimum de chaque ligne de la matrice `R`, c'est-à-dire le minimum de n réalisations d'une loi uniforme sur $[0, a]$. Ainsi, `np.min(R,1)` est une matrice colonne contenant 10000 réalisations de la variable U. De même, `np.max(R,1)` renvoie une matrice colonne contenant 10000 réalisations de la variable V.

   La fonction `simulation` renvoie donc un échantillon de taille 10000 pour chacune des lois des variables U et V.

3. On peut procéder comme suit pour U (on procèderait de même pour V).

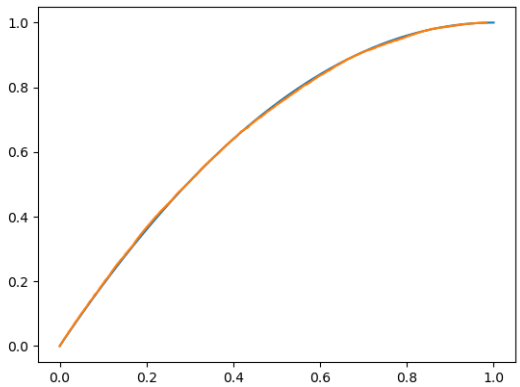
```python
# échantillon
n = 2
a = 1
U = simulation(a,n)[0]

# Fonction de répartion théorique
def F(x):
    return 1-(1-x/a)**n
x = np.linspace(0,1,100)
y = F(x)
plt.plot(x,y)

# Fonction de répartition empirique
plt.step(np.sort(U), np.arange(0,1,1/10000))

plt.show()
```

   On obtient la représentation graphique suivante :



- Les courbes étant pratiquement confondues, cela valide cette simulation de la variable U.

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

Soient $k \in \mathbb{N}^*$ 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).

1. La fonction f_λ est bien continue sur $\mathbb{R}$ sauf éventuellement en λ, et positive. De plus :

$$\int_{-\infty}^{+\infty} f_\lambda(x)\,\mathrm{d}x = \int_{\lambda}^{+\infty} \frac{r}{x^{k+1}}\,\mathrm{d}x = \frac{r}{k\lambda^k}.$$

   Et donc cette intégrale vaut 1 si, et seulement si, $r = k\lambda^k$.

2. Pour $x \leq \lambda$, on a $P(X \leq x) = 0$, et si $x > \lambda$:

$$P(X \leq x) = \int_{-\infty}^{x} f_\lambda(t)\,\mathrm{d}t = k\lambda \int_{\lambda}^{x} \frac{1}{t^{k+1}}\,\mathrm{d}t = 1 - \left(\frac{\lambda}{x}\right)^k.$$

3. F_X réalise une bijection croissante de $]\lambda, +\infty[$ sur $]0,1[$. Pour trouver sa bijection, prenons $x \in]\lambda, +\infty[$ et $y \in]0,1[$ et résolvons :

$$F_X(x) = y \Leftrightarrow 1 - \left(\frac{\lambda}{x}\right)^k = y \Leftrightarrow x = \frac{\lambda}{(1-y)^{\frac{1}{k}}}.$$

   Ainsi, $F_X^{-1}(y) = \dfrac{\lambda}{(1-y)^{\frac{1}{k}}}$.

   En appliquant la méthode d'inversion, on peut simuler une loi de Pareto de paramètres λ et k comme suit.

```python
def pareto1(lbd,k):
    y = rd.random()
    return lbd/((1-y)**(1/k))
```

4. (a) On vérifie que $Z = \max(X_1, \dots, X_k)$ est une variable à densité, avec pour fonction de répartition :

$$F_Z : x \in \mathbb{R} \mapsto \begin{cases} 0 & \text{si } x \leq 0 \\ x^n & \text{si } x \in]0,1[\\ 1 & \text{si } x \geq 0 \end{cases}.$$

Cherchons la fonction de répartition de $Y = \frac{\lambda}{Z}$. Puisque $Z(\Omega) =]0,1]$, alors $Y(\Omega) =]\lambda, +\infty[$ et $F_Y(x) = 0$ pour tout $x \leq \lambda$. Et pour $x > \lambda$:

$$\begin{aligned} P(Y \leq x) = P\left(\frac{\lambda}{Z} \leq x\right) = P\left(\frac{\lambda}{x} \leq Z\right) &= 1 - P(Z < \frac{\lambda}{x}) \\ &= 1 - P(Z \leq \frac{\lambda}{x}) \text{ car } Z \text{ est à densité} \\ &= 1 - \left(\frac{\lambda}{x}\right)^k. \end{aligned}$$

Donc Y suit une loi de Pareto de paramètres λ et k.

(b) On peut procéder comme suit, en suivant la définition de la variable Y.

```python
def pareto2(lbd,k):
    Z = np.max(rd.random(k))
    return lbd/Z
```

Exercice 10 (★★★ - Méthode de simulation par rejet)

Soient X une variable à densité et f une densité de X. On suppose que f est bornée et à support borné, c'est-à-dire qu'il existe $K \in \mathbb{R}_+$, $a, b \in \mathbb{R}$ avec $a < b$ tels que :

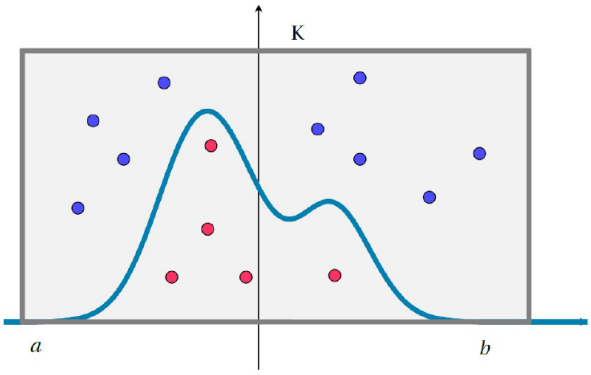
$$\forall x \in \mathbb{R}, \quad 0 \leq f(x) \leq K \quad \text{et} \quad f(x) = 0 \text{ si } x \notin [a,b].$$

Rappelons que pour tout $\alpha < \beta$, l'aire $\mathscr{A}_{\alpha,\beta}$ comprise entre la courbe, l'axe des abscisses et les droites d'équations $x = \alpha$ et $x = \beta$ correspond exactement à la probabilité que la variable aléatoire X prenne les valeurs comprises entre α et β :

$$\mathscr{A}_{\alpha,\beta} = \int_\alpha^\beta f(t)\,\mathrm{d}t = P(\alpha \leq X \leq \beta).$$

Pour simuler la variable X, on peut tirer un point au hasard sous la courbe représentative de f de manière uniforme. On prend alors pour réalisation l'abscisse de ce point.

Précisons que pour tirer un point sous la courbe, on tire un point au hasard dans le rectangle $[a,b] \times [0,K]$ à l'aide de deux lois uniformes. Si ce dernier est sous la courbe, on le garde. Sinon, on recommence.



1. Quelle est la loi du nombre d'essais avant de tirer un point qui est bien situé sous la courbe de f ?

2. Soit X une variable aléatoire dont une densité f est définie sur $\mathbb{R}$ par :

$$f(x) = \begin{cases} 6x(1-x) & \text{si } x \in [0,1], \\ 0 & \text{sinon.} \end{cases}$$

   (a) Tracer le graphe de f à l'aide de `Python`. Quelles valeurs choisir pour a, b et K ?

   (b) Écrire une fonction `Python` d'en-tête `def simulation()` qui renvoie une réalisation de la variable X.

   (c) Créer un vecteur contenant $N = 10000$ réalisations de la loi de X. Comparer l'histogramme des fréquences de cet échantillon à la courbe représentative de f. Conclure.

1. On répète une même épreuve à deux issues (succès ou échec) de manière identique et indépendante, et on compte le nombre d'essais pour obtenir le premier succès. Il s'agit donc d'une loi géométrique de paramètre p la probabilité de succès.

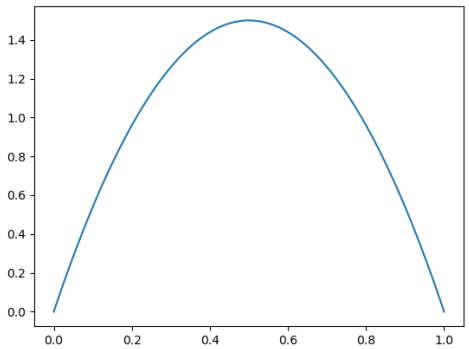
   La probabilité de succès est égale au rapport de l'aire sous la courbe représentative de f, qui est égale à 1 puisque f est une densité, avec l'aire du rectangle $[a,b] \times [0,K]$, égale à $(b-a) \times K$. Ainsi, $p = \frac{1}{(b-a)\times K}$.

2. (a) On peut procéder comme suit.

```python
def f(x):
    return 6*x*(1-x)

x = np.linspace(0,1,100)
y = f(x)
plt.plot(x,y)
plt.show()
```

   On obtient la représentation graphique suivante.



   On peut prendre $a = 0$, $b = 1$, $K = 3/2$.

   (b) On prend un point (x,y) au hasard dans le rectangle $[0,1] \times [0,3/2]$ à l'aide de la commande `rd.random`. Puis on teste si $y > f(x)$, c'est-à-dire si le point (x,y) est sous la courbe de f ou non. Tant que ce point est au dessus, on recommence en prenant de nouveau un point au hasard dans le rectangle. Et lorsque la boucle s'arrête, le point (x,y) est en dessous de la courbe, et on renvoie x.

   On peut procéder comme suit :

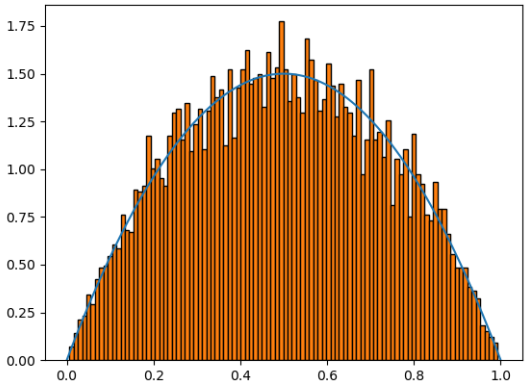
```python
def simulation():
    x = rd.random()
```

```
    y = (3/2)*rd.random()
    while y>f(x) :
        x = rd.random()
        y = (3/2)*rd.random()
    return x
```

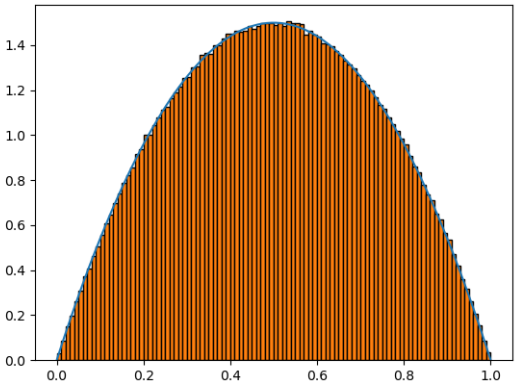
(c) On procède comme d'habitude.

```
N = 10000
x = np.zeros(N)
for k in range(N):
    x[k] = simulation()
plt.hist(x,100,density='True',edgecolor='k')
plt.show()
```

Voici la représentation graphique obtenue :



En prenant $N = 1000000$, on obtient :



Cette simulation de la variable X semble donc tout à fait pertinente.