TD

Thème : Simulations des variables aléatoires

I. Estimation d'une espérance, d'une variance

Soit X, une variable aléatoire réelle admettant une espérance.
En utilisant une nouvelle fois la loi faible des grands nombres, on peut approximer $\mathbf{E}(X)$ en simulant un grand nombre de fois la variable aléatoire X et en effectuant la moyenne arithmétique de l'échantillon obtenu.

$$\mathbf{E}(X) \simeq \frac{x_1 + \cdots x_N}{N} \quad \text{où N représente la taille de l'échantillon } (x_1, \ldots, x_N).$$

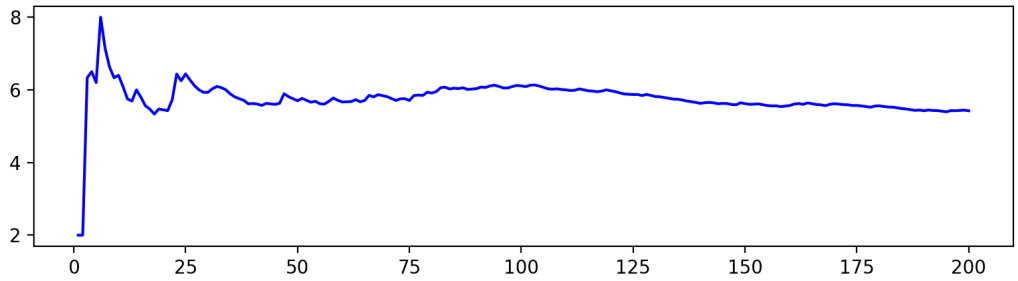
Exercice 1. ✦ ✎ *d'après Ecricome 2021, voie E*

On lance indéfiniment une pièce équilibrée. On s'intéresse au rang du lancer auquel on obtient pour la première fois deux «Pile» consécutifs.

1. Recopier et compléter la fonction Python ci-contre afin qu'elle simule les lancers de la pièce jusqu'à l'obtention de deux «Pile» consécutifs, et qu'elle renvoie le nombre de lancers effectués.
2. Écrire une fonction `moyenne` d'argument n qui simule n fois l'expérience ci-dessus et renvoie la moyenne des résultats obtenus.
3. On calcule `moyenne(n)` pour chaque entier n de $[\![1, 200]\!]$, et on trace les résultats obtenus dans le graphe suivant. Que pouvez-vous conjecturer sur la variable aléatoire X ?

Editeur

```
def simulX():
    tirs=0
    pile=0
    while pile ... :
        if rd.random()<1/2 :
            pile ...
        else :
            pile ...
        tirs ...
    return ...
```



Exercice 2. ✦✦ **Estimation de l'espérance et de la variance**

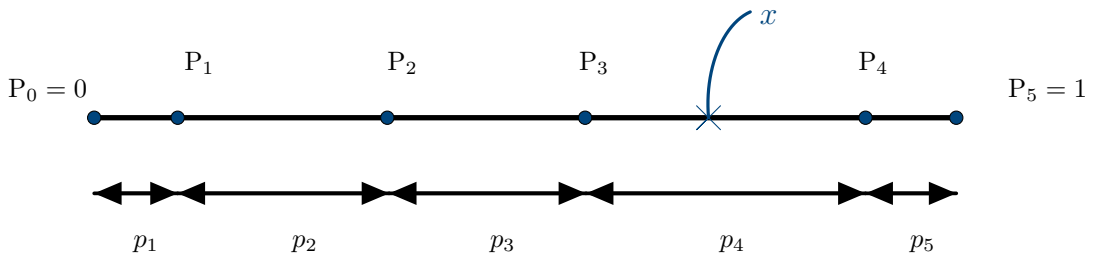
Une urne contient trois boules : une rouge, une bleue et une blanche. On tire avec remise dans l'urne jusqu'à l'apparition de la première boule blanche. Soit X, la variable aléatoire égale au nombre de boules rouges tirées. Écrire un programme qui permet d'approximer l'espérance et la variance de X.

II. Méthode d'inversion dans le cas discret

Soit X une variable aléatoire finie. Pour simplifier le programme, on suppose $X(\Omega) = [\![1; n]\!]$. On note

$$\forall\, i \in [\![1; n]\!], \qquad p_i = \mathbf{P}\big([X = i]\big) \quad \text{et} \quad P_i = \sum_{k=1}^{i} p_k.$$

On souhaite simuler la variable X uniquement avec la commande `rand()`. L'idée est la suivante : on simule la loi de X en tirant un réel x, au hasard dans $[0;1]$ et en renvoyant l'entier k si x appartient à l'intervalle $[\mathrm{P}_{k-1};\mathrm{P}_k[$. La probabilité que l'entier k soit choisi est alors $\mathrm{P}_k - \mathrm{P}_{k-1} = p_k = \mathbf{P}\big([\mathrm{X}=k]\big)$ (avec la convention $\mathrm{P}_0 = 0$).



Exercice 3. ✦✦ **Simulation des v.a finies**

1. Que représente les réels P_i en termes de probabilités et de X ?
2. Écrire un programme qui prend en argument la matrice-ligne $(p_i)_{i\in[\![1;n]\!]}$ correspond à la loi de X et simule X.
3. Tester votre programme sur la loi uniforme $\mathscr{U}([\![1;10]\!])$ et vérifier la cohérence du programme à l'aide d'un histogramme.

On peut toutefois étendre la méthode précédente au cas où X prend ses valeurs dans $\mathbb{N}$. Prenons le cas de $\mathrm{X} \hookrightarrow \mathcal{P}(\lambda)$. Notons pour $k \in \mathbb{N}$,

$$p_k = \mathbf{P}\big([\mathrm{X}=k]\big) = \mathrm{e}^{-\lambda}\frac{\lambda^k}{k!} \quad \text{et} \quad \mathrm{P}_k = \sum_{j=0}^{k} p_j.$$

En particulier, on a $$\mathrm{P}_{k+1} = \mathrm{P}_k + \frac{\lambda}{k+1}p_k.$$

Exercice 4. ✦✦ **Simulation d'une loi de Poisson**

1. Adapter la méthode précédente pour simuler la variable $\mathrm{X} \hookrightarrow \mathcal{P}(\lambda)$.
2. Comparer l'histogramme obtenu aux valeurs théoriques.

III. Méthode d'inversion dans le cas continu

L'idée repose sur l'énoncé suivant :

Si
- ⇀ U suit la loi uniforme sur $]0;1[$.
- ⇀ X est une variable aléatoire à densité dont la fonction de répartition F est une bijection de $]a;b[$ avec $-\infty \leqslant a < b \leqslant +\infty$ sur $]0;1[$.

Alors La variable $\mathrm{Y} = \mathrm{F}^{-1}(\mathrm{U})$ suit la même loi que X.

Ainsi pour simuler la variable X, on explicite (si possible) la fonction réciproque F^{-1}, on simule la variable U et on renvoie $\mathrm{F}^{-1}(\mathrm{U})$.

Exemples.

- La fonction de répartition d'une loi uniforme sur $[a;b]$ est définie sur $\mathbb{R}$ par

$$\mathrm{F}(x) = \begin{cases} 0 & \text{si } x < a \\ \frac{x-a}{b-a} & \text{si } a \leqslant x < b \\ 1 & \text{si } x \geqslant b \end{cases}$$

On inverse la relation $\mathrm{F}(x) = t$ pour $x \in]a;b[$. On obtient alors

$$\mathrm{F}^{-1}(\mathrm{U}) = a + (b-a)\mathrm{U} \hookrightarrow \mathcal{U}([a;b]).$$

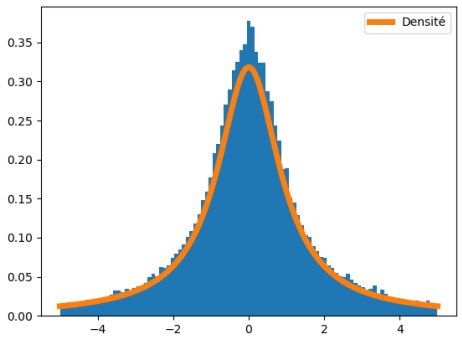
- On dit qu'une variable aléatoire X à densité suit une loi de Cauchy si une densité est donnée par $f : t \in \mathbb{R} \mapsto 1/\big(\pi(1+t^2)\big)$. On a vu que la fonction de répartition est définie sur $\mathbb{R}$ par $\mathrm{F}(x) = \arctan(x)/\pi + 1/2$. La fonction F est strictement croissante, continue de $\mathbb{R}$ dans $]0;1[$. Le théorème de la bijection s'applique

$$\forall\, t \in]0;1[, \qquad \mathrm{F}^{-1}(t) = \tan\big(\pi(t-1/2)\big).$$

D'après l'exercice précédent, si U suit une loi uniforme continue sur $]0;1[$, alors la variable $\mathrm{F}^{-1}(\mathrm{U})$ suit une loi de Cauchy. Ci-dessous, un histogramme d'un échantillon obtenu par simulation par la méthode d'inversion. On vérifie bien que l'histogramme obtenu est très proche d'une densité de la loi de Cauchy.

Editeur

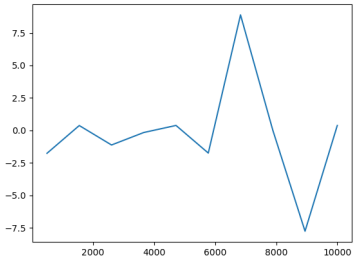
```
U= np.random.rand(50000)
L=np.tan(np.pi*(U-1/2))
inter=np.linspace(-5,5,100)
plt.hist(L, bins=inter, density=True)
# Création de l'histogramme
x=np.linspace(-5,5,200)
y=1/(np.pi*(1+x**2))
plt.plot(x,y,linewidth=5,label="Densité")
plt.legend()
plt.show()
```



Exercice 5. ✦ ✎ Expliquer l'intérêt du code suivant ? Qu'illustre-t-on sur la loi de Cauchy ?

Editeur

```
Liste=100*np.linspace(5,100,10)
E=np.zeros(10)
for i in range(10) :
    U= np.random.rand(Liste[i])
    L=np.tan(np.pi*(U-1/2))
    e=sum(L)/Liste[i]
    E[i]=e
plt.plot(Liste,E)
plt.show()
```



Exercice 6. ✦ ✎ Une variable aléatoire X suit la loi de Weibull de paramètres $\alpha > 0$ et $\lambda > 0$ si elle admet pour densité f définie sur $\mathbb{R}$ par

$$f(t) = \begin{cases} \alpha\lambda \mathrm{e}^{-\lambda t^{\alpha}} t^{\alpha-1} & \text{si } t > 0 \\ 0 & \text{sinon.} \end{cases}$$

Adapter la méthode précédente pour simuler une loi de Weibull. Comment tester votre simulation ? (*On pourra choise* $\alpha = 2$, $\lambda = 3$.

IV. Compléments

Exercice 7. ✦ ✎ Pour tout entier naturel $n \geqslant 2$, on pose $\mathrm{J}_n = \displaystyle\int_0^1 \frac{\ln(t)}{1+t^n}\,\mathrm{d}t$. L'objectif est d'obtenir une valeur approchée de l'intégrale J_n à l'aide de Python.

1. Soit X une variable aléatoire suivant la loi exponentielle de paramètre 1. On pose : $\mathrm{Y}_n = \dfrac{-\mathrm{X}}{1+\mathrm{e}^{-n\mathrm{X}}}$.
   Vérifier que Y_n admet une espérance, et exprimer $\mathbf{E}(\mathrm{Y}_n)$ à l'aide de J_n.

2. On rappelle que dans la bibliothèque Python `numpy.random` importée par `rd` se trouve l'instruction `rd.exponential(a)` qui renvoie une réalisation d'une variable aléatoire suivant la loi exponentielle de paramètre `1/a`.
Écrire une fonction qui prend en argument n et permet d'approximer J_n.

Exercice 8. ✧ Simulation d'une loi de Pareto

Soient $\theta \in \mathbb{R}_*^+$ et la fonction f définie sur $\mathbb{R}$ par :

$$f(x) = \begin{cases} \frac{1}{\theta x^{1+1/\theta}} & \text{si } x \geqslant 1 \\ 0 & \text{si } x < 1. \end{cases}$$

1. Montrer que f est une densité de probabilité.
On considère dans la suite une variable aléatoire X strictement positive de densité f et on note F sa fonction de répartition.

2. On pose $Y = \ln(X)$ et on note G sa fonction de répartition. Justifier que Y suit une loi exponentielle dont on précisera le paramètre.

3. On rappelle qu'en Python, la commande `rd.exponential(a)` simule une variable aléatoire suivant la loi exponentielle de paramètre $1/a$. Écrire une fonction Python `SimuP` prenant en argument θ et permettant de simuler la variable X.

4. Commenter le résultats des lignes suivantes.

Editeur

```
def mystere(theta):
    A=np.zeros(5)
    for p in range(2,7):
        S=0
        for k in range(1,10**p):
            S+=simuP(theta)
        A[p-2]=round(S/10**p,3)
        # round(..,3) arrondit
        # le résultat à 10**(-3)
    return A
```

Console

```
>>> mystere(1/2)
array([2.197, 2.103, 2.011, 1.99, 2. ])

>>> mystere(1)
array([ 4.892,  8.204, 15.007, 10.577,
    13.309])

>>> mystere(1.2)
array([  7.318,  12.865,  29.35 , 502.559,
    187.655])
```

5. Soient $X_1, \ldots, X_k$, k variables aléatoires mutuellement indépendantes et toute de même loi uniforme sur $]0;1]$. On pose alors $Z = 1/\max(X_1, \ldots, X_k)$. Reconnaître la loi de Z et simuler cette variable aléatoire.

Exercice 9. ✦✦ ✎ Simulation d'une loi géométrique à partir d'une loi exponentielle

Soit $\lambda \in \mathbb{R}_+^*$. Soit X une variable aléatoire sur un espace probabilisé $(\Omega, \mathcal{A}, \mathbf{P})$ de loi $\mathcal{E}(\lambda)$. On pose $Y = \lfloor X \rfloor + 1$.

1. Montrer $Y \hookrightarrow \mathcal{G}(1 - e^{-\lambda})$.

2. En déduire un programme qui simule une loi géométrique de paramètre p en utilisant la commande `np.exponential(1/lbda)`.

ECG 2

Éléments de solutions

Exercice 1 p. 1

1.

```
def simulX():
    tirs=0
    pile=0
    while pile<2:
        if rd.random()<1/2 :
            pile+=1
        else :
            pile=0
        tirs+=1
    return tirs
```

2.

```
def moyenne(n):
    s=0
    for i in range(n):
        s+=simulX()
    return s/n
```

3. Comme la simulation donne une courbe qui se rapproche de 6, on peut conjecturer que la variable X admet une espérance et que cette dernière est proche de 6 (voir la loi des grands nombres).

Exercice 2 p. 1

Un premier programme pour simuler la variable X :

```
def simulationUrne():
    # 1 pour rouge
    # 2 pour bleu
    # 3 pour blanc
    X=0
    Tirage=rd.randint(1,4)

    while Tirage<3:
        if Tirage==1:
            X+=1
        Tirage=rd.randint(1,4)
    return X
```

On en déduit deux programmes.

⇀ Pour approximer l'espérance :

```
m=50000
S=0
for i in range(m):
    S+=simulationUrne()
print('espérance :', S/m)

#  Test :
espérance : 0.9939
```

On conjecture que l'espérance est de 1.

⇀ Pour approximer la variance, on commence par approximer le moment d'ordre 2 puis on utilise la formule de Koenig-Huygens :

```
m=50000
e2=0
for i in range(m):
    e2+=simulationUrne()**2
# e2/m est une approximation du moment d'
    ordre 2.
print('Variance :', e2/m-1)
```

Exercice 3 p. 2

1. Pour tout indice k

$$\mathrm{P}_k = \mathbf{P}(\mathrm{X} \leqslant k).$$

2. La variable `Somme` correspond à P_k que l'on calcule à l'aide d'une boucle while.

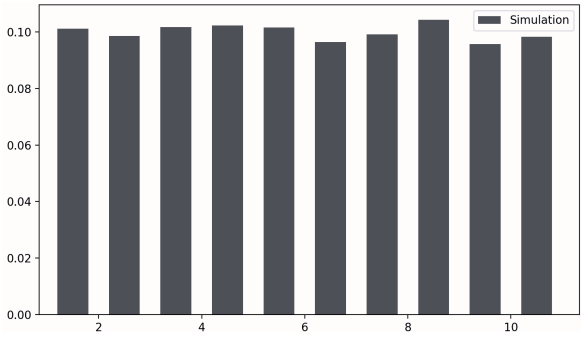
```
def Simulation(Loi) :
    n=len(Loi)
    k=1
    Somme=Loi[0]
    x=rd.rand()
    while Somme<x :
        Somme+=Loi[k]
        k+=1
    return k
```

3.

```
def TestSimulation(m) : # m représente le
    nombre de simulation, ou encore la
    taille de l'échantillon m=1000 par
    exemple
    Loi=np.ones(10)/10
    ech=np.zeros(m)
    for i in range(m):
        ech[i]=Simulation(Loi)
    classe =np.linspace(0,11,12)
    plt.hist(ech,bins=classe,density='true
        ')
    plt.show()
```

Il ne reste plus qu'à tester

```
TestSimulation(10000)
```



On pourra tester avec des valeurs de `m` de plus en plus grandes.

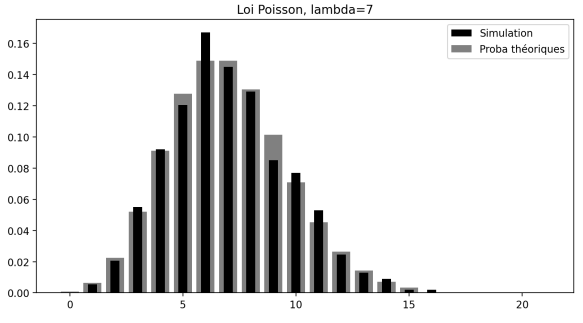
1.

```
def SimulationPoisson(lbda) :
    p=np.exp(-lbda)
    Somme=p
    k=0
    x=rd.random()
    while Somme<x :
        k+=1
        p=p*lbda/k
        Somme+=p
    return k
```

2.

```
def TestSimulationPoisson(m,lbda) :
    # m représente le nombre de
        simulations, ou encore la taille
        de l'échantillon m=1000 par
        exemple
    ech=np.zeros(m)
    for i in range(m):
        ech[i]=SimulationPoisson(lbda)
    classe=np.linspace(0,22,23)-0.5
    plt.hist(ech,bins=classe,density='true
        ')
    plt.show()
```

En rajoutant le diagramme en bâtons, on obtient par exemple :



Exercice 5 p. 3

Ce code illustre le fait que la loi de Cauchy n'a pas d'espérance (ni de variance). En effet, si la loi admet une variance, on s'attend à une convergence des moyennes empiriques vers l'espérance.

Exercice 6 p. 3

- Si on pose pour tout $t \in \mathbb{R}_x^+$

$$g(t) = \lambda t^\alpha$$

alors $$f(t) = g'(t)\mathrm{e}^{-g(t)}.$$

Ainsi pour $x \in \mathbb{R}_x^+$

$$\begin{aligned} \mathrm{F}_x(x) &= \int_{-\infty}^{x} f(t)\,\mathrm{d}t \\ &= \int_0^x f(t)\,\mathrm{d}t = \int_0^x g'(t)\mathrm{e}^{-g(t)}\,\mathrm{d}t \\ &= \left[-\mathrm{e}^{-g(t)}\right]_0^x = 1-\mathrm{e}^{-g(x)}. \end{aligned}$$

Soit $u \in]0;1[$

$$\begin{aligned} \mathrm{F}_x(x) = 0 &\iff 1-\mathrm{e}^{-\lambda x^\alpha} = u \\ &\iff 1-u = \mathrm{e}^{-\lambda x^\alpha} \\ &\iff \ln(1-u) = -\lambda x^\alpha \\ &\iff \left(-\frac{1}{\lambda}\ln(1-u)\right)^{1/\alpha} = x. \end{aligned}$$

La fonction de répartition $\mathrm{F_X}$ définit une bijection de $\mathbb{R}_*^+$ dans $]0;1[$ avec pour tout $t \in]0;1[$,

$$\mathrm{F}^{-1}(t) = \left(-\frac{1}{\lambda}\ln(1-t)\right)^{1/\alpha}.$$

Dès lors, un code possible est :

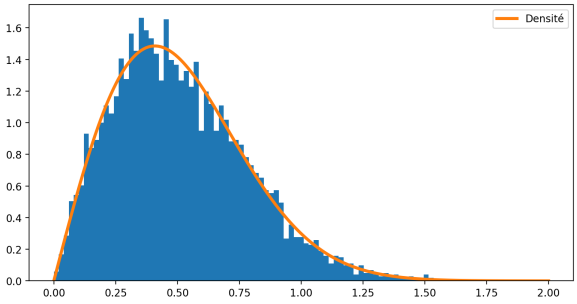
```
def simuWeibull(alpha,lbda):
    u=rd.random()
    return (-np.log(1-u)/lbda)**(1/alpha)
```

En remarquant que si $\mathrm{U} \hookrightarrow \mathcal{U}([0;1])$, alors $1-\mathrm{U} \hookrightarrow \mathcal{U}([0;1])$, on peut simplifier la dernière ligne par

```
    return (-np.log(u)/lbda)**(1/alpha)
```

- On peut tester le code avec :

```
alpha=2
lbda=3
m=5000 # taille échantillon
Ech=np.zeros(m)
for i in range(m):
    Ech[i]=simuWeibull(alpha,lbda)
inter=np.linspace(0,2,100)
plt.hist(Ech,bins=inter,density=True)
# Création de l'histogramme
x=np.linspace(0,2,200)
y=alpha*np.exp(-lbda*x**alpha)*lbda*x**(
    alpha-1)
plt.plot(x,y,label="Densité",linewidth=3)
plt.legend()
plt.show()
```



L'histogramme épouse l'allure de la densité, le test est concluant.

Exercice 7 p. 3

1. D'après le théorème de transfert, Y_n admet une espérance si et seulement si l'intégrale

$$\int_0^{+\infty} \frac{-u}{1+\mathrm{e}^{-nu}} f_X(u)\,\mathrm{d}u$$

est absolument convergente. Or pour $u \in [0,+\infty[$.

$$\left|\frac{-u}{1+\mathrm{e}^{-nu}} f_X(u)\right| = \frac{|-u|}{|1+\mathrm{e}^{-nu}|}\,|f_X(u)| = \frac{u}{1+\mathrm{e}^{-nu}}\mathrm{e}^{-u}.$$

Par les croissances comparées, on a

$$\frac{u}{1+\mathrm{e}^{-nu}}\mathrm{e}^{-u} = o_{+\infty}\left(\frac{1}{u^2}\right)$$

et la convergence de l'intégrale par les critères de Riemann et de négligeabilité. Ainsi, Y_n admet une espérance et à l'aide du changement de variable strictement croissant et $\mathcal{C}^1$

$$u = -\ln(t) \quad \text{et} \quad \mathrm{d}u = -\frac{1}{t}\,\mathrm{d}t, \qquad \mathrm{e}^{-u}\,\mathrm{d}u = \mathrm{d}t,$$

on montre que

$$\int_0^{+\infty} \frac{u}{1+\mathrm{e}^{-nu}}\mathrm{e}^{-u}\,\mathrm{d}u = -\int_0^1 \frac{\ln(t)}{1+t^n}\,\mathrm{d}t$$

Finalement, il y a convergence absolue, l'espérance existe et

$$\mathbf{E}(Y_n) = \int_0^{+\infty} \frac{-u}{1+\mathrm{e}^{-nu}}\mathrm{e}^{-u}\,\mathrm{d}u = \int_0^1 \frac{\ln(t)}{1+t^n}\,\mathrm{d}t = J_n.$$

2. Pour approximer J_n, on approxime $\mathbf{E}(Y_n)$. Et, à l'aide de la loi faible des grands nombres, une approximation est donnée par les moyennes empiriques.

```
def ApproxJn(n):
    m=5000
    s=0
    for i in range(m):
        x=rd.exponential(1)
        y=-x/(1+np.exp(n*x))
        s+=y
    return s/m
```

Exercice 8 p. 4

1. La fonction f est continue sur $\mathbb{R}\setminus\{1\}$ et positive. De plus, pour $A \in [1;+\infty[$

$$\int_{-\infty}^{A} f(t)\,\mathrm{d}t = \int_1^A f(t)\,\mathrm{d}t = \int_1^A \frac{1}{\theta}\cdot x^{-1-\frac{1}{\theta}}\,\mathrm{d}x$$
$$= \left[-x^{-1/\theta}\right]_1^A = 1 - A^{-1/\theta} \underset{A\to+\infty}{\longrightarrow} 1.$$

On a donc convergence et l'égalité

$$\int_{-\infty}^{+\infty} f(t)\,\mathrm{d}t = 1.$$

En conclusion, f est une densité de probabilité.

2. La variable X est presque sûrement à valeurs dans $[1;+\infty[$, $\ln(X)$ est donc presque sûrement à valeurs dans $\mathbb{R}^+$.
Soit $x \in \mathbb{R}$. Pour $x < 0, G(x) = 0$. Pour $x \geqslant 0$, on a par stricte croissance de l'exponentielle

$$G(x) = \mathbf{P}\big(\ln(X) \leqslant x\big) = \mathbf{P}\left(X \leqslant \mathrm{e}^x\right) = F\left(\mathrm{e}^x\right).$$

Or pour $t \in [1;+\infty[$

$$F(t) = \int_{-\infty}^{t} f(t)\,\mathrm{d}t = 1 - t^{-1/\theta}$$

en reprenant le calcul de la question 1. D'où

$$G(x) = 1 - (\mathrm{e}^x)^{-1/\theta} = 1 - \mathrm{e}^{-x/\theta}.$$

On reconnaît pour G la fonction de répartition d'une loi exponentielle de paramètre $1/\theta$. Comme la fonction de répartition caractérise la loi

$$\ln(X) \hookrightarrow \mathcal{E}(1/\theta).$$

3. Inversement si $Y \hookrightarrow \mathcal{E}(1/\theta)$ alors $\exp(X)$ suit une loi de Pareto de paramètre θ. On en déduit le code :

```
def simuP(theta):
    y=rd.exponential(theta)
    return np.exp(y)
```

4. Lorsque le paramètre vaut 1/2, on constate une convergence des premiers termes. Ce qui ne semble pas le cas dès que le paramètre dépasse 1. Ce code illustre le fait qu'une variable qui suit une loi de Pareto n'a pas d'espérance dès que la paramètre

dépasse 1.

5. Notons H, la fonction de répartition de Z et F_U celle d'une loi uniforme.
La variable Z est strictement supérieure à 1, donc H est nulle sur $]-\infty;1]$. De plus, pour $x \in [1;+\infty[$

$$H(x) = \mathbf{P}(Z \leqslant x) = \mathbf{P}\big(\max(X_1,\dots,X_k) \geqslant 1/x\big)$$

car la fonction inverse est décroissante sur $\mathbb{R}_*^+$. On poursuit par passage au complémentaire

$$\begin{aligned} H(x) &= 1 - \mathbf{P}\big(\max(X_1,\dots,X_k) < 1/x\big) \\ &= 1 - \mathbf{P}\Big(\bigcap_{i=1}^{k}[X_i < 1/x]\Big) \\ &= 1 - \prod_{i=1}^{k}\mathbf{P}\Big([X_i < 1/x]\Big) \quad \text{(indépendance)} \\ &= 1 - \prod_{i=1}^{k}\mathbf{P}\Big([X_i \leqslant 1/x]\Big) \quad \text{(à densité)} \\ &= 1 - F_U(1/x)^k = 1 - 1/x^k \quad (\text{car } 1/x \in [0;1]). \end{aligned}$$

On reconnaît l'expression d'une variable aléatoire de loi de Pareto de paramètre $\theta = 1/k$.

- Pour simuler la variable, on peut écrire

```
u=rd.random(k)
z=1/np.max(u)
```

Exercice 9 p. 4

1. Comme X est à valeurs dans $\mathbb{R}^+$, Y est une variable aléatoire à valeurs dans $\mathbb{N}^*$. Soit $k \in \mathbb{N}^*$, on a

$$\begin{aligned} \mathbf{P}\big([Y=k]\big) &= \mathbf{P}\big(\lfloor X\rfloor + 1 = k\big) \\ &= \mathbf{P}\big([k-1 \leqslant X < k]\big) \\ &= \mathbf{P}\big([k-1 \leqslant X \leqslant k]\big). \end{aligned}$$

On peut maintenant faire le calcul explicite à l'aide de la densité de X.

$$\begin{aligned} \mathbf{P}\Big(\big[\,k-1 \leqslant X \leqslant k\,\big]\Big) &= \int_{k-1}^{k} \lambda e^{-\lambda t}\,dt \\ &= e^{-\lambda(k-1)} - e^{-\lambda k} \\ &= \big(e^{-\lambda}\big)^{k-1}\big(1-e^{-\lambda}\big). \end{aligned}$$

Si on pose $p = 1 - e^{-\lambda}$ et $q = 1 - p = e^{-\lambda}$, on a prouvé

$$\forall\, k \in \mathbb{N}^*, \qquad \mathbf{P}\big([Y=k]\big) = q^{k-1}p.$$

Finalement, Y suit une loi géométrique $\mathcal{G}\big(1-e^{-\lambda}\big)$.

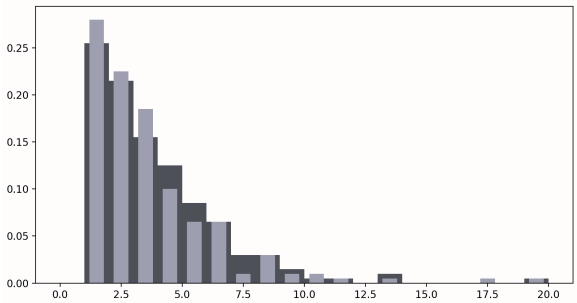
2. Posons $\lambda = -\ln(1-p)$ de sorte que $1 - e^{-\lambda} = p$. Un code possible est alors :

```
def geoP(p):
    lbda=-np.log(1-p)
    x=rd.exponential(1/lbda)
    # attention à la convention :
    # il faut
    # l'inverse du paramètre
    return np.floor(x)+1
```

Remarque. Pour tester le programme, on peut afficher un histogramme d'un échantillon obtenu par le script précédent et un histogramme obtenu directement à l'aide de la commande python `rd.geometric`.

```
p=1/3
m=200 # taille échantillon
Ech=np.zeros(m)
for i in range(m):
    Ech[i]=geoP(p)

Ech2=rd.geometric(p,m)
inter=np.linspace(0,20,21)
plt.hist(Ech,bins=inter,density=True)
plt.hist(Ech2,bins=inter,density=True,
    rwidth=0.6)
plt.show()
```



Précisons qu'on peut éviter la commande `rd.exponential(1/lbda)` *car il facile de simuler une variable de loi exponentielle par la méthode d'inversion.*