Correction - TP 11

Méthode de Monte-Carlo

Exercice 1

1. (a) Vérifions les hypothèses de la loi faible des grands nombres. Les U_i étant indépendantes, les variables $g(U_i)$ le sont aussi par lemme de coalition.
De plus si U suit une loi $\mathcal{U}([0,1])$, et en notant

$$f_U : x \mapsto \begin{cases} 1 & \text{si } 0 \leq x \leq 1 \\ 0 & \text{sinon,} \end{cases},$$

on a :

- $g(U)$ admet une espérance si et seulement si $\displaystyle\int_{-\infty}^{+\infty} g(t)f_U(t)dt = \int_0^1 g(t)dt$ converge absolument par le théorème de transfert. Or c'est l'intégrale d'une fonction continue sur un segment, donc c'est bien le cas, et on a :
$$E(g(U)) = \int_0^1 g(t)dt.$$
- $g(U)$ admet une variance si et seulement si $\displaystyle\int_{-\infty}^{+\infty} g(t)^2 f_U(t)dt = \int_0^1 g(t)^2 dt$ converge absolument toujours par le théorème de transfert. Ce qui est bien le cas car là aussi, il s'agit d'une intégrale d'une fonction continue sur un segment.

On peut donc conclure par la loi faible des grands nombres que :

$$S_n \xrightarrow{P} \int_0^1 g(x)\mathrm{d}x.$$

(b) On cherche une fonction g continue sur $[0,1]$ telle que :

$$\int_0^1 g(x)\mathrm{d}x = \int_0^1 \ln(1+x^2)\mathrm{d}x.$$

Prenons donc $g : x \mapsto \ln(1+x^2)$. On peut alors compléter le programme proposé :

```
def g(x) :
    return np.log(1+x**2)

n = 1000
S = 0
for k in range(n):
    u = rd.random()
    S = S + g(u)
print(S/n)
```

On obtient par exemple pour $n = 1000$ en exécutant deux fois le programme `0.2641198` et `0.2661799` respectivement.

(c) La valeur obtenue à l'aide des commandes de l'énoncé donne `0.2639435`. On obtient donc une valeur approchée de cette intégrale. On pourrait augmenter n pour préciser notre estimation de l'intégrale.

2. Essayons d'adapter le raisonnement de l'exercice précédent. Pour la première intégrale J, on cherche U une variable dont une densité est f_U et g une fonction continue sur un intervalle à déterminer tels que :

$$\int_{-\infty}^{+\infty} g(x)f_U(x)dx = \int_0^{+\infty} \frac{e^{-x}}{1+x^4}dx.$$

Cela suggère de prendre

$$f_U : x \in \mathbb{R} \mapsto \begin{cases} e^{-x} & \text{si } x \geq 0 \\ 0 & \text{sinon,} \end{cases},$$

et donc $U \hookrightarrow \mathcal{E}(1)$, et $g : x \in \mathbb{R}_+ \mapsto \dfrac{1}{1+x^4}$. On obtient donc le code suivant :

```
def g(x) :
    return 1/(1+x**4)

n = 1000
S = 0
for k in range(n):
    u = rd.exponential(1/1)
    S = S + g(u)
print(S/n)
```

On obtient `0.6129161` comme estimation ponctuelle de la valeur de J.

Pour K en procédant toujours par analogie avec ce qui a été fait avant, on est amené à prendre

$$f_U : x \in \mathbb{R} \mapsto \begin{cases} \frac{1}{6} & \text{si } -2 \leq x \leq 4 \\ 0 & \text{sinon.} \end{cases},$$

et donc $U \hookrightarrow \mathcal{U}([-2,4])$, et $g : x \in \mathbb{R}_+ \mapsto 6e^{-x^2}$. On obtient donc le code suivant :

```
def g(x) :
    return 6*np.exp(-x**2)

n = 1000
S = 0
for k in range(n):
    u = 6*rd.random()-2
    S = S + g(u)
print(S/n)
```

On obtient `1.7047088` comme estimation ponctuelle de la valeur de K.

Exercice 2

On adapte ce qui a été vu dans l'exercice précédent dans le cas discrète (pour obtenir une somme).

1. Soit $X \hookrightarrow \mathcal{B}(10, 1/2)$ et $g(x) = \dfrac{1}{\sqrt{1+x^4}}$. Alors avec le théorème de transfert (X est à support fini) :

$$E(g(X)) = \sum_{k \in X(\Omega)} g(k)P(X=k) = \sum_{k=0}^{10} \frac{1}{\sqrt{1+k^4}} \binom{10}{k} \left(\frac{1}{2}\right)^k \left(\frac{1}{2}\right)^{10-k} = \frac{1}{2^{10}} \sum_{k=0}^{10} \frac{\binom{10}{k}}{\sqrt{1+k^4}}.$$

On obtient donc le code suivant :

```
def g(x) :
    return 1/np.sqrt(1+x**4)

n = 1000
S = 0
for k in range(n):
    u = rd.binomial(10, 1/2)
    S = S + g(u)
print(2**10*S/n)
```

On obtient `64.4528891` comme estimation ponctuelle de la première somme.

2. Soit $X \hookrightarrow \mathcal{G}(1/2)$ et $g(x) = \dfrac{1}{x^3}$. Alors avec le théorème de transfert (sous réserve de convergence) :

$$E(g(X)) = \sum_{k \in X(\Omega)} g(k)P(X=k) = \sum_{k=1}^{+\infty} \frac{1}{x^3} \left(1-\frac{1}{2}\right)^{k-1} \left(\frac{1}{2}\right) = \sum_{k=1}^{+\infty} \frac{1}{k^3 2^k}.$$

Cette dernière série converge bien par comparaison de séries à terme général positif (car par exemple $\frac{1}{k^3 2^k} \leq \frac{1}{k^3}$ qui est le terme général d'une série de Riemann convergente car $3 > 1$). On obtient donc le code suivant :

```
def g(x) :
    return 1/x**3

n = 1000
S = 0
for k in range(n):
    u = rd.geometric(1/2)
    S = S + g(u)
print(S/n)
```

On obtient `0.5407447` comme estimation ponctuelle de la deuxième somme.

Exercice 3

En suivant le procédé expliqué au début de cette section, on prend des points au hasard dans $[0,1] \times [0,1]$, et on teste si ces points sont à l'intérieur de la boucle ou non. On compte alors le nombre de points satisfaisant cette condition, et on renvoie leur fréquence. Ce qui donne :

```
n = 10000
X = rd.random(n)
Y = rd.random(n)
S = 0
for i in range(n):
    if X[i]**3+Y[i]**3 <= (3/2)*X[i]*Y[i] :
        S = S+1
print(S/n)
```

On obtient `0.382` comme estimation ponctuelle de l'aire de cette boucle.

Exercice 4

1. $\mathscr{D}$ correspond à un quart du disque de centre $(0,0)$ et de rayon 1. Il est d'aire $\dfrac{\pi \times 1^2}{4} = \dfrac{\pi}{4}$.

2. (a) T_i suit une loi de Bernoulli de paramètre p où p représente la probabilité qu'un point pris au hasard dans $C = [0,1] \times [0,1]$ appartienne à $\mathscr{D}$, soit :

$$p = \frac{\mathscr{A}_{\mathscr{D}}}{\mathscr{A}_C} = \frac{\pi}{4}.$$

Ainsi T_i suit la loi $\mathcal{B}(\pi/4)$.

(b) On a par linéarité de l'espérance,

$$E(4\overline{T_n}) = \frac{4}{n}\sum_{i=1}^{n} E(T_i) = \frac{4}{n} \times n \times \frac{\pi}{4} = \pi.$$

Comme les T_i sont indépendants, car les X_i et Y_i le sont, on a par les propriétés de la variance :

$$V(4\overline{T_n}) = \frac{16}{n^2}\sum_{i=1}^{n} V(T_i) = \frac{16}{n^2} \times n \times \frac{\pi}{4}\left(1 - \frac{\pi}{4}\right) = \frac{\pi(4-\pi)}{n}.$$

On peut alors appliquer l'inégalité de Bienaymé-Tchebychev que, pour tout $\varepsilon > 0$:

$$P(|4\overline{T_n} - \pi| \geq \varepsilon) \leq \frac{\pi(4-\pi)}{n\varepsilon^2} \underset{n\to+\infty}{\longrightarrow} 0.$$

Ainsi, $4\overline{T_n}$ converge en probabilité vers π.

3. On adapte le programme de l'exercice précédent :

```
n = 10000
X = rd.random(n)
Y = rd.random(n)
S = 0
for k in range(n):
    if X[k]**2+Y[k]**2 <= 1 :
        S = S+1
print(4*S/n)
```

On obtient `3.1216` comme valeur approchée de π.

4. (a) Les calculs ont déjà été effectués en cours : on a obtenu qu'un intervalle de confiance de $\frac{\pi}{4}$ au niveau de risque $\alpha = 0.05$ est donné par $\left[\overline{T_n} - \frac{t_\alpha}{2\sqrt{n}}; \overline{T_n} + \frac{t_\alpha}{2\sqrt{n}}\right]$. Ainsi un intervalle de confiance de π au niveau de risque $\alpha = 0.05$ est donné par $\left[4\overline{T_n} - \frac{2t_\alpha}{\sqrt{n}}; 4\overline{T_n} + \frac{2t_\alpha}{\sqrt{n}}\right]$. À l'aide de l'estimation observée précédente, on obtient l'un intervalle de confiance observé $[3.0824, 3.1608]$. On notera qu'il contient bien π, on avait théoriquement une probabilité de 0.95 que ça soit effectivement le cas.

(b) Il faut prendre n de sorte que :

$$\frac{2t_\alpha}{\sqrt{n}} \leq 0.01 \quad \Leftrightarrow \quad n \geq 153664.$$

On doit prendre $n = 153664$ pour obtenir un intervalle de confiance de π au niveau de confiance de 0.95. On notera que cette valeur est très élevée, et que notre méthode ne permet pas de converger très rapidement vers le nombre π.

Exercice 5

1. On va créer un échantillon `E` de taille N grand de la loi normale $\mathcal{N}(0,1)$. On cherchera ensuite parmi cet échantillon la proportion des modalités `E(k)` plus petites qu'un réel donné x, en comptant le nombre `C` de telles modalités. On renverra alors cette proportion `p` qui est une estimation de $P(X \leq x) = \Phi(x)$ où $X \hookrightarrow \mathcal{N}(0,1)$. Ce qui donne la fonction suivante :

```
def phi_empirique(x):
    n = 10000
    X = rd.normal(0,1,n)
    S = 0
    for k in range(n):
        if X[k]<=x :
            S = S+1
    return(S/n)
```

On pouvait aussi procéder comme suit :

```
def phi_empirique(x):
    n = 10000
    X = rd.normal(0,1,n)
    return np.mean(X<=x)
```

2. Comparons par exemple les valeurs en 1, en ajoutant les lignes de commandes suivantes :

```
p = sp.ndtr(1)
print(phi_empirique(1),p)
```

On obtient `0.8413447` avec la fonction `sp.ndtr` et `0.847` avec `phi_empirique`.

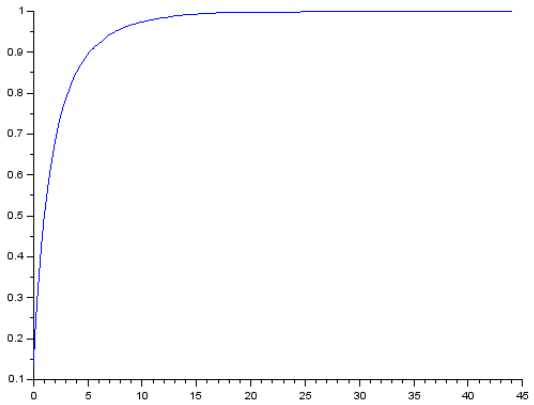
Exercice 6

En utilisant les commandes données au TP 9, on obtient :

```
N = 10000 #taille de l' echantillon
X = rd.poisson(2, N)
Y = rd.exponential(1/1, N)
Z = X*Y  #echantillon de la loi de Z

n = 100
u = np.linspace(np.min(Z), np.max(Z), n)
v = np.zeros(n)
for k in range(n) :
    v[k] = np.mean(Z <= u[k])
plt.plot(u, v)
plt.show()
```

qui génère le graphe suivant :



Exercice 7

1. La variable `E` contient une matrice de taille $n \times 1000$ de réalisations de la loi $\mathscr{E}(1)$. La commande `np.std(E,0)` calcule l'écart-type de chacune des colonnes (grâce à l'argument `0`) de `E`. `S` contient donc 1000 réalisations de S_n. `np.mean(S)` contient la moyenne de ces réalisations, ce qui correspond à une estimation de $E(S_n)$. Ainsi la variable `bS` contient une estimation de l'erreur moyenne que commet S_n en estimant θ. De même, `rS` contient une estimation de $V(S_n) + (E(S_n) - \theta)^2$.

   En exécutant ce code, on obtient `bS = -0.0018237` et `rS = 0.001871`.

2. On adapte ce qui a été fait plus haut :

```
lbd = 1 ; theta = 1/lbd ; n = 100 ; m = 1000
E = rd.exponential(1,[n,m])
Xbar = np.mean(E,0)
bXbar = np.mean(Xbar) - theta
rXbar = np.std(Xbar)**2 + (np.mean(Xbar)-theta)**2
print(bXbar,rXbar)
```

   On obtient ici `bXbar = 0.0015157` et `rXbar = 0.0009914`. Notons que la valeur de `bXbar`, proche de 0, n'est pas étonnante ici, puisque $\overline{X_n}$ est un estimateur sans biais de $\theta = \dfrac{1}{\lambda}$.

3. Exécutons ce programme pour différentes valeurs de λ. On obtient :

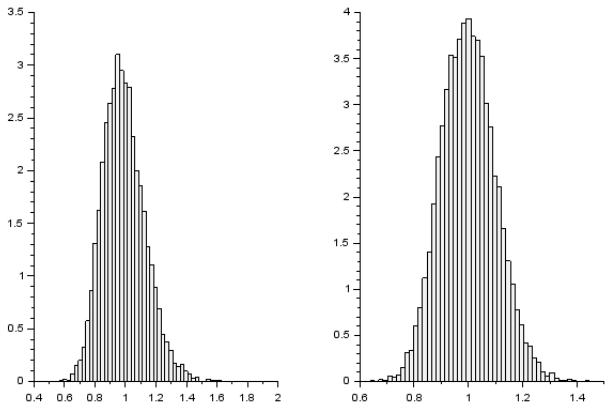
λ	Estimation de $V_\theta(\overline{X_n}) + (E_\theta(\overline{X_n}) - \theta)^2$	Estimation de $V_\theta(S_n) + (E_\theta(S_n) - \theta)^2$
1	0.0009914	0.001871
1/2	0.004065	0.0082213
1/5	0.0258101	0.0478365
1/10	0.1025539	0.2082655

On obtient des résultats analogues en faisant varier n. On observe que $r_\theta(\overline{X_n}) \approx \dfrac{r_\theta(S_n)}{2}$ pour les différentes valeurs de λ et n testées. On peut donc conclure, sur la base de ces observations, que $\overline{X_n}$ est un meilleur estimateur de $\theta = \frac{1}{\lambda}$ que S_n.

4. On peut utiliser le programme suivant :

```
lbd = 1
n = 100
theta = 1/lbd
m = 10000
E = rd.exponential(1/lbd, [n, m])
S = np.std(E, 0)
X = np.mean(E, 0)
c = np.arange(0, 3.1, 0.02)
plt.subplot(1, 2, 1)
plt.hist(S, 50, density='True', edgecolor='k')
plt.subplot(1, 2, 2)
plt.hist(X, 50, density='True', edgecolor='k')
plt.show()
```

On obtient les histogramme suivants dans le cas où $\lambda = 1$ et $n = 100$:



On obtient des histogrammes analogues pour d'autres valeurs de λ et n. Ces histogrammes confirment les résultats numériques obtenus à la question précédente : l'histogramme associé à $\overline{X_n}$ est plus "concentré" autour de la valeur de $\theta = \frac{1}{\lambda}$ que celui associé à S_n. $\overline{X_n}$ semble donc être un meilleur estimateur de $\theta = \frac{1}{\lambda}$ que S_n.

Exercice 8

On utilise le code suivant :

```
alpha = 0.05
p = 1-alpha/2
x = sp.ndtri(p)
print(x)
```

qui renvoie la valeur `1.959964`. Pour $\alpha = 0.01$, on obtient la valeur `2.5758293`.

Exercice 9

1. (a) La fonction suivante convient :

```
def etendue(n,alpha):
    p = 1-alpha/2
    t = sp.ndtri(p)
    return(2*t/sqrt(n))
```

   (b) On constate qu'à n fixé, l'étendue augmente si α diminue. Et à α fixé, l'étendue diminue si n augmente.

2. On choisit le paramètre `theta` au hasard à l'aide d'une loi exponentielle $\mathscr{E}(1)$. On crée ensuite $m = 1000$ échantillons de taille $n = 1000$ de la loi $\mathscr{N}(\theta, 1)$. On calcule alors 1000 réalisations de $\overline{X_n}$ à l'aide de la commande `Xbar = np.mean(E,0)`. La boucle `for` permet alors de compter sur les 10000 intervalles de confiance observés, le nombre d'intervalles contenant effectivement `theta`. On stocke ce nombre dans la variable `c`, et on renvoie la fréquence des intervalles de confiances qui contiennent `theta`. On obtient ainsi une estimation du niveau de confiance réelle de l'intervalle.

3. On obtient en exécutant plusieurs fois le code :

θ **réel**	**Fréquence d'IdC contenant** θ
0.4844947	0.946
0.9987862	0.937
0.7165302	0.944
2.7665696	0.956

   On obtient un niveau de confiance réel proche de 95%, conforme à ce qu'on attendait.