TP 8

Méthode de Monte Carlo

Exercice 1 (★★)

Soit (U_n) une suite de variables aléatoires indépendantes suivant toutes la loi uniforme sur $[0,1]$, et soit $g : [0,1] \to \mathbb{R}$ une fonction continue. Pour tout $n \geq 1$, on pose $S_n = \dfrac{1}{n}\sum_{i=1}^{n} g(U_i)$.

1. En utilisant la loi faible des grands nombres, montrer que :

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

   À quelle propriété de cours ce résultat vous fait-il penser ?

2. On souhaite déterminer une valeur approchée de l'intégrale $I = \displaystyle\int_0^1 \cos(x^2)\mathrm{d}x$.

   Compléter le programme suivant afin qu'il retourne une valeur approchée de I.

```
def g(x) :
      return ...

n = 1000 ; S = 0
for k in range(n):
        u = ...
        S = S + ...
print(...)
```

   Essayer ce programme avec différentes valeurs de n, en faisant plusieurs essais pour chaque valeur de n. Comparer les valeurs obtenues avec celle donnée par le code suivant :

```
from scipy.integrate import quad
res, err = quad(g,0,1)
print("Résultat de l'intégrale :", res)
```

1. Vérifions les hypothèses de la LFGN. Les U_i étant indépendantes, les variables $g(U_i)$ le sont aussi par lemme de coalition. De plus si U suit une loi $\mathscr{U}([0,1])$, 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 $\displaystyle\int_{-\infty}^{+\infty} g(t) f_U(t)\,\mathrm{d}t = \int_0^1 g(t)\,\mathrm{d}t$ 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 :

$$E(g(U)) = \int_0^1 g(t)\,\mathrm{d}t.$$

   - $g(U)$ admet une variance si et seulement si $\displaystyle\int_{-\infty}^{+\infty} g(t)^2 f_U(t)\,\mathrm{d}t = \int_0^1 g(t)^2\,\mathrm{d}t$ 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 :

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

Ce résultat est comparable au théorème des sommes de Riemann : si $g : [0,1] \to \mathbb{R}$ est continue, alors on a (méthode des rectangles à gauche) :

$$\frac{1}{n}\sum_{k=0}^{n-1} g(k/n) \underset{n\to+\infty}{\longrightarrow} \int_0^1 g(x)\mathrm{d}x.$$

Ici les points de $[0,1]$ pour constituer la somme sont pris de façon équidistribuée sur $[0,1]$. Ils le sont de façon aléatoire dans cet exercice.

2. On cherche une fonction g continue sur $[0,1]$ telle que :

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

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

```
def g(x) :
      return np.cos(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 `0.9118163` et `0.8988760` respectivement. La valeur obtenue à l'aide des commandes de l'énoncé donne `0.9045242`. On obtient donc une valeur approchée de cette intégrale. On pourrait augmenter n pour préciser notre estimation de l'intégrale.

Exercice 2 (★)

Estimer la valeur des intégrales suivantes à l'aide de la méthode de Monte-Carlo.

- $I = \displaystyle\int_0^{+\infty} \frac{e^{-x}}{1+x^4}\,\mathrm{d}x$
- $J = \displaystyle\int_{-2}^{4} e^{-x^2}\,\mathrm{d}x.$

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

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

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 $U \hookrightarrow \mathscr{E}(1)$, 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 I.

Pour J 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 $U \hookrightarrow \mathscr{U}([-2,4])$, 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 J.

Exercice 3 (★★)

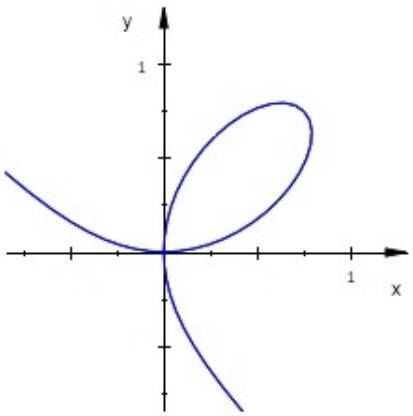
Estimer la valeur des sommes suivantes à l'aide de la méthode de Monte-Carlo :

- $\displaystyle\sum_{k=0}^{10} \frac{\binom{10}{k}}{\sqrt{1+k^4}}$.
- $\displaystyle\sum_{k=1}^{+\infty} 1$

Exercice 4 (★)

Considérons la courbe ci-contre (appelée folium de Descartes), dont on cherche à calculer une valeur approchée de l'aire formée par la boucle. On admet que l'intérieur de cette boucle est l'ensemble :

$$\mathscr{B} = \{(x,y) \in \mathbb{R}^2 : x \geq 0,\ y \geq 0,\ x^3 + y^3 \leq \frac{3xy}{2}\}.$$



Folium de Descartes.

Compléter le programme suivant afin qu'il retourne une valeur approchée de l'aire de $\mathscr{B}$.

```
n = 10000
X = rd.random(n)
Y = ...
S = 0
for i in range(n):
    if ............................. :
        S = S+1
print(.........)
```

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 5 (★★ - Calcul d'une valeur approchée de π)
Soit $\mathscr{D}=\{(x,y)\in\mathbb{R}^2,\ x,y\geq 0,\ x^2+y^2\leq 1\}$

1. Représenter $\mathscr{D}$. Quelle est son aire ?
2. Soient $(X_i)_{1\leq i\leq n}$ et $(Y_i)_{1\leq i\leq n}$ des variables i.i.d. suivant la loi $\mathscr{U}([0,1])$. On note T_i la variable qui vaut 1 si le point (X_i,Y_i) appartient à $\mathscr{D}$, et 0 sinon.
   (a) Déterminer la loi de T_i.
   (b) On pose $\overline{T_n}=\dfrac{1}{n}\displaystyle\sum_{i=1}^{n}T_i$. Montrer que $4\overline{T_n}$ converge en probabilité vers π.
3. Écrire un programme qui calcule une valeur approchée de π.
4. (a) Déterminer un intervalle de confiance asymptotique de π, au niveau de risque $\alpha=0.05$, dont les bornes dépendent de $\overline{T_n}$.
   (b) Quelle valeur de n faut-il choisir pour obtenir une approximation de π à $\varepsilon=0.01$ près, au niveau de risque $\alpha=0.05$?

1. $\mathscr{D}$ correspond à un quart du disque de centre $(0,0)$ et de rayon 1. Il est d'aire $\dfrac{\pi 1^2}{4}=\dfrac{\pi}{4}$.
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 $\mathscr{B}(\pi/4)$.
   (b) Les T_i étant indépendants, car les X_i et Y_i le sont, et admettant une variance, on a par la loi faible des grands nombres que $\overline{T_n}$ converge en probabilité vers $E(T_1)=\dfrac{\pi}{4}$. En composant par $f:x\in\mathbb{R}\mapsto 4x$ continue, on en déduit que $4\overline{T_n}$ converge en probabilité vers $E(4T_1)=\pi$.
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 6 (★★ - Fonction de répartition empirique de la loi $\mathcal{N}(0,1)$)

1. Écrire une fonction `phi_empirique` qui, à un réel x entré par l'utilisateur, renvoie une estimation de $\Phi(x)$ par la méthode de Monte-Carlo.
2. On rappelle que la fonction de répartition Φ de la loi $\mathcal{N}(0,1)$ est accessible dans la librairie `scipy.special` à l'aide de la commande `sp.ndtr`. Comparer les résultats obtenus à l'aide de la fonction `phi_empirique` avec ceux renvoyés par la commande `sp.ndtr`.

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 7 (★)

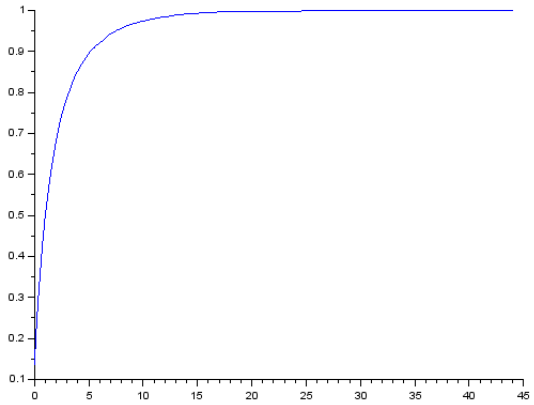
Tracer la fonction de répartition empirique de $Z = XY$ où $X \hookrightarrow \mathscr{P}(2)$ et $Y \hookrightarrow \mathscr{E}(1)$ sont indépendantes.

En utilisant les commandes données au TP5, 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

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

qui génère le graphe suivant :



Exercice 8 (★★)

Soit $(X_1, \dots, X_n)$ un échantillon de loi mère $\mathscr{E}(\lambda)$. Nous disposons des deux estimateurs de $\theta = \dfrac{1}{\lambda}$:

- la moyenne empirique $\overline{X_n} = \dfrac{1}{n}\sum_{i=1}^{n} X_i$;
- l'écart type empirique $S_n = \sqrt{\dfrac{1}{n-1}\sum_{k=1}^{n}(X_k - \overline{X_n})^2}$.

On souhaite comparer ces deux estimateurs. Prenons pour commencer $\lambda = 1$ et $n = 100$.

1. On considère les instructions suivantes :

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

Que contiennent les variables `S`, `bS` et `rS` ?

2. Estimer de même le biais et le risque quadratique de $\overline{X_n}$ comme estimateur de θ.
3. Recommencer avec d'autres valeurs de λ et de n. Quel estimateur privilégieriez vous ?
4. On souhaite enfin comparer ces deux estimateurs à l'aide de leur histogramme des fréquences. Écrire pour cela un programme qui :
   - simule $m = 10000$ n-échantillons de la loi $\mathscr{E}(\lambda)$;
   - calcule, pour chaque échantillon observé, l'estimation correspondante obtenue à l'aide de chacun des estimateurs $\overline{X_n}$ et S_n ;
   - trace l'histogramme des 10000 estimations obtenues avec chacun des estimateurs.

     On rappelle pour cela que la commande `plt.hist(x,50)` trace l'histogramme de la série statistique `x`, en répartissant ses éléments en 50 classes équiréparties entre la plus petite et la plus grande valeur de `x`.

     On pourra également utiliser les commandes `plt.subplot(1,2,1)` et `plt.subplot(1,2,2)` pour tracer les histogrammes sur la même fenêtre graphique l'un à côté de l'autre.

Comparer les deux histogrammes. Recommencer pour d'autres valeurs de λ. Quel semble être le meilleur estimateur ?

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 :

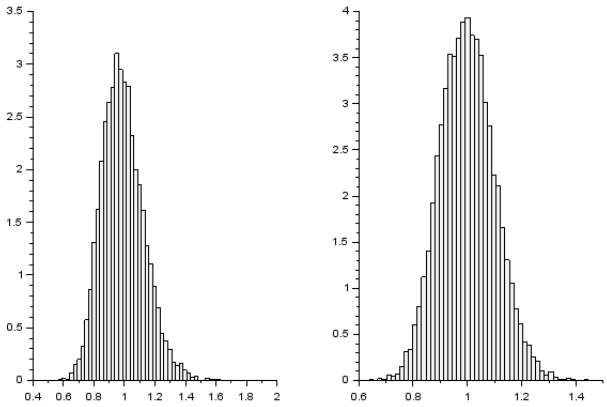
λ	Estimation de $V_\theta(\overline{X_n}) + (E_\theta(\overline{X_n}) - \theta)^2$	Estimation de $V_\theta(S_n) + (E_\theta(S_n) - \theta)^2$
1	0.0009914	0.001871
1/2	0.004065	0.0082213
1/5	0.0258101	0.0478365
1/10	0.1025539	0.2082655

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)
M = np.mean(E,0)
T = M
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(T,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 9 (★ - Des valeurs classiques)

Avec les notations habituelles, déterminer les valeurs de t_α pour $\alpha = 0.05$ et $\alpha = 0.01$.

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 10 (★★ - Intervalle de confiance de l'espérance d'une loi normale)

Nous avons montré en TD que si $X_1, \dots, X_n$ sont des variables i.i.d. suivant la loi normale $\mathcal{N}(\theta, 1)$, alors $\left[\overline{X_n} - \frac{t_\alpha}{\sqrt{n}}, \overline{X_n} + \frac{t_\alpha}{\sqrt{n}}\right]$ est un intervalle de confiance de θ au niveau de confiance $1-\alpha$.

1. **Étendue de l'intervalle de confiance.**
   (a) Écrire une fonction `y=etendue(n,alpha)` qui prend en paramètres un entier n et un réel $\alpha \in]0,1[$, et renvoie l'étendue de l'intervalle de confiance de θ correspondant.
   (b) Essayer ce programme pour $\alpha = 0.05$ en faisant varier n. Puis à n fixé, étudier l'étendue de l'intervalle pour $\alpha = 0.05$, $\alpha = 0.01$, $\alpha = 0.001$ et $\alpha = 0.0001$. Que constate-t-on ?

2. **Niveau de confiance réel.**
   Prenons dans la suite $\alpha = 0.05$ et $n = 1000$. On souhaite déterminer le niveau de confiance réel de l'intervalle de confiance par la méthode de Monte-Carlo.
   (a) Qu'effectue le code suivant :

```
theta = rd.exponential(1/1) ; n = 1000 ; m = 10000 ; t=sp.ndtri(0.975)
E = rd.normal(theta,1,[n,m])
Xbar = np.mean(E,0)
c = 0
for k in range(m):
    if np.abs(Xbar[k]-theta)<t/np.sqrt(n):
        c = c+1
print("theta :",theta) ;
print("Estimation du niveau de conf de l'IdC :",c/m)
```

   (b) Exécuter plusieurs fois ce code. Est-ce conforme à ce que vous vous attendiez ?

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 $\mathcal{E}(1)$. On crée ensuite $m = 1000$ échantillons de taille $n = 1000$ de la loi $\mathcal{N}(\theta, 1)$. 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.