FEUILLE DE TD/TP 3

Exercice 1. Le but de cet exercice est de tester la méthode de Monte Carlo sur un exemple simple. On souhaite vérifier la qualité de l'approximation de

$$\mathbb{P}\big(U \leq \lambda\big),$$

où U est une variable aléatoire de loi uniforme sur $]0,1[$ et λ est un réel donné dans l'intervalle $]0,1[$.

(1) Montrer que la variance de la variable aléatoire $\mathbb{1}_{U\leq\lambda}$ est $\lambda(1-\lambda)$ et est toujours inférieure à $1/4$. Mettre en place une représentation graphique de la fonction $x \in [0,1] \mapsto x(1-x)$ permettant de le vérifier.

La variable $\mathbb{1}_{U\leq\lambda}$ prend ses valeurs dans l'ensemble $\{0;1\}$, elle suit donc une loi de Bernoulli de paramètre :

$$\mathbb{P}(\mathbb{1}_{U\leq\lambda} = 1) = \mathbb{P}(U \leq \lambda) = \lambda$$

car $U \sim \mathcal{U}([0;1])$ et sa variance est bien $\lambda(1-\lambda)$. La fonction $x \to x(1-x)$ admet pour dérivée $x \to 1-2x$ et atteint donc son maximum sur $[0;1]$, $1/4$, en $x = 1/2$.

Soit $(U_n)_{n\in\mathbb{N}}$ une suite de v.a.i.i.d. de même loi que U. On pose pour tout $n \in \mathbb{N}^*$, $X_n = \mathbb{1}_{U_n\leq\lambda}$ et $\overline{X}_n$ la moyenne empirique des n premières variables :

$$\overline{X}_n = \frac{1}{n}\sum_{i=1}^{n} X_i = \frac{1}{n}\sum_{i=1}^{n} \mathbb{1}_{U_i\leq\lambda}.$$

Le théorème central limite dit que l'intervalle aléatoire

$$\Big]\overline{X}_n - 1,96\frac{\sqrt{\lambda(1-\lambda)}}{\sqrt{n}}; \overline{X}_n + 1,96\frac{\sqrt{\lambda(1-\lambda)}}{\sqrt{n}}\Big[$$

est un intervalle de confiance asymptotique pour $\lambda = \mathbb{P}(U \leq \lambda)$ au niveau de confiance $0,95$. En particulier, l'intervalle de confiance $]\overline{X}_n - \frac{1.96}{2}n^{-1/2}; \overline{X}_n + \frac{1,96}{2}n^{-1/2}[$ est un intervalle de confiance asymptotique pour λ au niveau de confiance supérieur ou égal à $0,95$.
Si on choisit $n = \lceil(\frac{1.96}{2})^2\varepsilon^{-2}\rceil$, pour $\varepsilon > 0$ donné, alors l'intervalle de confiance ci-dessus est contenu dans l'intervalle $]\bar{x}_n - \varepsilon, \bar{x}_n + \varepsilon[$.

(2) Mettre en place une fonction `Python`, dépendant de deux paramètres λ et n, permettant de simuler les réalisations de n variables aléatoires indépendantes de même loi que $\mathbb{1}_{U\leq\lambda}$.

(3) Mettre en place une boucle permettant de simuler les réalisations de M v.a. indépendantes de même loi que $\overline{X}_n$.

(4) On choisit $n = \lceil(\frac{1.96}{2})^2\varepsilon^{-2}\rceil$ pour $\varepsilon = 0,01$. Calculer la proportion de réalisations dans la boucle ci-dessus à l'intérieur de l'intervalle $]\lambda - \varepsilon, \lambda + \varepsilon[$ et comparer à 0,95. On choisira successivement $\lambda = 0,75$ et $\lambda = 0,5$.

```
## Question 1
x=np.arange(0,1,.01)
plt.plot(x,x*(1-x))

## Question 2
def MC_p(lambd,n):
    return(np.mean((np.random.rand(n)<=lambd).astype(int)))

## Question 3
def MC_test(M,lambd,n):
    res=np.zeros(M)
    for i in range(M):
        res[i]=MC_p(lambd,n)
    return(res)
```

```
## Question 4
#lambda=.75
lambd=.75
n=int(np.floor(1.96**2/(4*.01**2))+1)
M=10**4
res=MC_test(M,lambd,n)
np.mean((np.abs(res-lambd)<=.01).astype(int))

#lambda=.5
lambd=.5
n=int(np.floor(1.96**2/(4*.01**2))+1)
M=10**4
res=MC_test(M,lambd,n)
np.mean((np.abs(res-lambd)<=.01).astype(int))
```

Exercice 2. Le but de cet exercice est d'étudier deux approches Monte-Carlo pour donner une approximation de la quantité :

$$I = \int_0^1 \cos(x^3)e^{-x}\,\mathrm{d}x.$$

(1) Montrer que I peut être écrit sous les formes suivantes :

$$I = \mathbb{E}\left[\cos(U^3)e^{-U}\right], \quad I = \mathbb{E}\left[\cos(X^3)\mathbb{1}_{X\in[0,1]}\right],$$

où U est une variable aléatoire de loi uniforme sur $[0,1]$ et X est une variable aléatoire de loi exponentielle de paramètre 1.

Comme la loi de U a pour densité la fonction $x \mapsto \mathbb{1}_{[0,1]}(x)$, on a directement que

$$\mathbb{E}\left[\cos(U^3)e^{-U}\right] = \int_0^1 \cos(x^3)e^{-x}\,\mathrm{d}x,$$

ce qui est précisément le résultat attendu.

Pour la seconde expression, comme la loi de X a pour densité la fonction $x \mapsto \mathbb{1}_{x>0}e^{-x}$, on a que

$$\mathbb{E}\left[\cos(X^3)\mathbb{1}_{[0,1]}(X)\right] = \int_0^\infty \cos(x^3)\mathbb{1}_{[0,1]}(x)e^{-x}\,\mathrm{d}x = \int_0^1 \cos(x^3)e^{-x}\,\mathrm{d}x$$

qui est aussi le résultat attendu.

(2) Écrire une fonction `f_MC` permettant de calculer en `Python` les valeurs de la fonction $x \mapsto \cos(x^3)e^{-x}$.

(3) En utilisant l'instruction `integrate.quad(f_MC,0,1)` du package `scipy.integrate`, donner une valeur approchée de I à l'aide de l'intégrateur numérique de `Python` (fondée sur une méthode déterministe).

(4) Écrire une fonction `MC_1` renvoyant l'approximation Monte-Carlo avec n tirages, l'écart-type empirique associé, ainsi que la largeur et les bornes de l'intervalle de confiance asymptotique de niveau 0,95 lorsque I est représentée à l'aide de U. Tester avec $n = 10^4$.

Répéter M fois l'approximation et calculer la proportion de fois où l'intervalle de confiance contient la valeur calculée dans la question (4) pour $M = n = 10^4$.

(5) Écrire une fonction `MC_2` renvoyant l'approximation Monte-Carlo avec n tirages, l'écart-type empirique associé, ainsi que la largeur et les bornes de l'intervalle de confiance asymptotique de niveau 0,95 lorsque I est représentée à l'aide de X. Tester avec $n = 10^4$.

Répéter M fois l'approximation et calculer la proportion de fois que l'intervalle de confiance contient la valeur calculée dans la question (4) pour $M = n = 10^4$.

(6) Quelle est la meilleure des deux méthodes ?

```
import scipy.integrate as integrate

## Question 2
def f_MC(x):
    return(np.cos(x**3)*np.exp(-x))

##Question 3

I=integrate.quad(f_MC, 0,1)
print(I[0])

##Question 4

def MC_1(n):
    tirages=f_MC(np.random.rand(n))
    approximation=np.mean(tirages)
    ecart_type=np.std(tirages,ddof=1)
    erreur=1.96*ecart_type*n**(-1/2)
    b_inf=approximation-erreur
    b_sup=approximation+erreur
    return([approximation,ecart_type,erreur,b_inf,b_sup])

print(MC_1(int(10**4)))

n=int(10**4)
M=int(10**4)
res=np.zeros(5*M).reshape(M,5)
for i in range(M):
    res[i,:]=MC_1(n)

#on recupere la premiere et la troisieme colonnes
np.mean((np.abs(I[0]-res[:,0])<res[:,2]).astype(int))
```

```
##Question 5

def MC_2(n):
    expo=-np.log(np.random.rand(n))
    tirages=np.cos(expo**3)*(expo<1).astype(int)
    approximation=np.mean(tirages)
    ecart_type=np.std(tirages,ddof=1)
    erreur=1.96*ecart_type*n**(-1/2)
    b_inf=approximation-erreur
    b_sup=approximation+erreur
    return([approximation,ecart_type,erreur,b_inf,b_sup])

print(MC_2(int(10**4)))

n=int(10**4)
M=int(10**4)
res=np.zeros(5*M).reshape(M,5)
for i in range(M):
    res[i,:]=MC_1(n)

#on recupere la premiere et la troisieme colonnes
np.mean((np.abs(I[0]-res[:,0])<res[:,2]).astype(int))
```

Exercice 3. *Aiguilles de Buffon.* Sur un parquet constitué de lattes de largeur 1, on lance des aiguilles ayant elles-mêmes 1 pour largeur. On compte alors la proportion d'aiguilles touchant les bords d'une latte. On se restreint à 10 lattes. On oriente le repère de manière à ce que les lattes soient orientées verticalement avec les bords aux abscisses entières. On appelle (X_1, X_2) le vecteur aléatoire modélisant le centre de l'aiguille et θ l'angle que fait l'aiguille avec l'axe des ordonnées. On suppose que les variables X_1 et X_2 sont indépendantes et de loi uniforme $\mathcal{U}([0;10])$. et que la variable θ est indépendante de (X_1, X_2) et de loi uniforme $\mathcal{U}([0;\pi])$.

(1) Faire un dessin.

(2) Montrer que la probabilité qu'une aiguille touche le bord d'une latte est $2/\pi$.

L'aiguille touche le bord d'une latte si les deux extrêmités de l'aiguille ne sont pas sur la même latte, c'est-à-dire si il existe un bord de latte, numéroté $k \in \{0,\dots,10\}$ telle que l'abscisse de l'extrémité la plus à gauche soit inférieure à k et l'abscisse de l'extémité la plus à droite soit supérieure à k. En notant A l'événement d'intérêt, on a donc :

$$\begin{aligned} A &:= \left\{ \exists k \in \{0,\dots,10\},\ X_1 - \frac{1}{2}\sin\theta < k \leq X_1 + \frac{1}{2}\sin\theta \right\} \\ &= \bigcup_{k=0}^{10} \left\{ -\frac{1}{2}\sin\theta < k - X_1 \leq \frac{1}{2}\sin\theta \right\}. \end{aligned}$$

Les ensembles ci-dessus sont deux à deux disjoints. On a donc

$$\begin{aligned} \mathbb{P}(A) &= \sum_{k=0}^{10} \mathbb{P}\left(-\frac{1}{2}\sin\theta < k - X_1 \leq \frac{1}{2}\sin\theta \right) \\ &= \sum_{k=0}^{10} \frac{1}{\pi}\frac{1}{10} \int_0^{10}\int_0^{\pi} \mathbb{1}_{\{-\frac{1}{2}\sin t < k-x \leq \frac{1}{2}\sin t\}} \,\mathrm{d}x\,\mathrm{d}t. \end{aligned}$$

De fait, en intégrant d'abord en x,

$$\mathbb{P}(A) = \sum_{k=0}^{10} \frac{1}{\pi}\frac{1}{10} \int_0^{\pi} \sin t \,\mathrm{d}t = \frac{1}{\pi}\int_0^{\pi} \sin t \,\mathrm{d}t = \frac{2}{\pi}.$$

(3) Écrire un code `python` permettant de représenter dans le carré $[-1,11]^2$ onze segments verticaux d'ordonnées allant de 0 à 10 aux abscisses suivantes : 0, 1, 2, ..., 10. On pourra utiliser l'instruction `plt.grid()`.

(4) Écrire un code `python` pour simuler le lancer d'une aiguille. La fonction doit renvoyer 1 si l'aiguille touche le bord et 0 sinon. Elle doit aussi représenter l'aiguille sur le plancher : de couleur rouge si elle touche le bord et de couleur noire sinon.
(5) Lancer $n = 10^4$ aiguilles (on peut faire plus, mais attention la représentation graphique ralentit beaucoup les choses). Calculer la proportion d'aiguilles touchant les bords d'une latte. Comparer cette proportion à $2/\pi$.

```
fig=plt.figure()
# On représente le plancher
plt.plot([0, 0], [0, 10], 'r-', xlim=(-1, 11))
for i in range(1, 11):
    plt.plot([i, i], [0, 10], 'r-')

# Fonction pour lancer une aiguille
def une_aiguille():
    centre = np.random.uniform(0, 10, 2)
    angle = np.random.uniform(0, np.pi, 1)
    extremite_1 = centre + 0.5 * np.array([np.sin(angle), np.cos(angle)])
    extremite_2 = centre - 0.5 * np.array([np.sin(angle), np.cos(angle)])
    color = "black"
    res = int(np.floor(extremite_1[0]) != np.floor(extremite_2[0]))
    if res == 1:
        color = "red"
    plt.plot([extremite_1[0], extremite_2[0]], [extremite_1[1], extremite_2[1]], color, linewidth=2)
    return res

# Lancer de n aiguilles
n=int(10**4)
res=np.zeros(n)
for i in range(n):
    res[i]=une_aiguille()
plt.grid()
plt.show()


# Calculer la valeur de Pi
estimated_pi = 2 / np.mean(res)
print(estimated_pi)
```

Buffon_python.jpg

FIGURE 1. Aiguilles de Buffon avec 10^4 lancers

Exercice 4. Dans tout l'exercice, U désigne une v.a. de loi uniforme sur $]0,1[$.

(1) On considère l'intégrale $I = \int_0^1 e^{-x^2}\,\mathrm{d}x$.
   (a) Montrer, que pour tout $u \in [0;1]$,
   $$(e^{-u^2} - e^{-1/4})(e^{-(1-u)^2} - e^{-1/4}) \leq 0\,.$$
   (b) Déduire de la question précédente que
   $$\mathrm{Cov}(e^{-U^2}, e^{-(1-U)^2}) = \mathrm{Cov}(e^{-U^2} - e^{-1/4}, e^{-(1-U)^2} - e^{-1/4}) \leq 0\,.$$
   puis que
   $$\mathbb{V}\left[\frac{1}{2}e^{-U^2} + \frac{1}{2}e^{-(1-U)^2}\right] \leq \frac{1}{2}\mathbb{V}\left[e^{-U^2}\right].$$
   (c) Mettre en place deux méthodes de Monte-Carlo fondées sur les représentations :
   $$I = \mathbb{E}\left[e^{-U^2}\right] \quad \text{et} \quad I = \frac{1}{2}\mathbb{E}\left[e^{-U^2} + e^{-(1-U)^2}\right].$$
   Comparer les variances associées et vérifier numériquement (en utilisant une troisième méthode de Monte-Carlo) les deux bornes ci-dessus.
(2) On considère maintenant l'intégrale $J = \int_0^1 e^{-(x-\frac{1}{2})^2}\,\mathrm{d}x$.

(a) Vérifier mathématiquement que la méthode de la variable antithétique ne diminue pas la variance.
(b) Vérifier numériquement que la méthode de la variable antithétique ne diminue pas la variance.

(3) On définit la fonction φ par :

$$\varphi(u) = (\tfrac{1}{2} - u)\mathbb{1}_{[0,\frac{1}{2})}(u) + (\tfrac{3}{2} - u)1_{[\frac{1}{2},1]}(u)\,.$$

(a) Vérifier que $\varphi(U)$ a même loi que U.
(b) Coder la fonction φ en `Python`.
(c) Montrer que, pour tout $u \in [0;1]$,

$$(e^{-(u-1/2)^2} - e^{-1/16})(e^{-(\varphi(u)-1/2)^2} - e^{-1/16}) \leq 0\,.$$

En déduire que

$$\mathbb{V}\left[\frac{1}{2}e^{-(U-\frac{1}{2})^2} + \frac{1}{2}e^{-(\varphi(U)-\frac{1}{2})^2}\right] \leq \frac{1}{2}\mathbb{V}\left[e^{-(U-\frac{1}{2})^2}\right].$$

(4) Mettre en place deux méthodes de Monte-Carlo fondées sur les représentations suivantes :

$$J = \mathbb{E}\left[e^{-(U-\frac{1}{2})^2}\right] \quad \text{et} \quad J = \frac{1}{2}\mathbb{E}\left[e^{-(U-\frac{1}{2})^2} + e^{-(\varphi(U)-\frac{1}{2})^2}\right].$$

Comparer les variances associées et vérifier numériquement (en utilisant une troisième méthode de Monte-Carlo) les deux bornes de la question ci-dessus.

Exercice 5. Le but de ce problème est d'illustrer, sur un exemple simple, le principe suivant :
Afin de réduire la variance dans une méthode de Monte-Carlo, il est souhaitable d'écrire la quantité à approcher sous la forme de l'espérance d'une fonction aussi proche que possible d'une constante contre la loi d'une variable aléatoire simulable.

A cet égard, nous allons voir qu'il est en particulier souhaitable que la fonction soit aussi proche que possible d'une constante **aux valeurs typiques** de la variable aléatoire. A contrario, nous allons également voir que, dans certains cas, les petites variations peuvent avoir une contribution non négligeable parce qu'accumulées sur des événements de probabilité trop importante. Ceci peut par exemple être le cas si la fonction à intégrer est proche de 0 : le cas échéant, l'erreur relative commise par la méthode de Monte-Carlo peut être significative si trop de variations, même petites, sont accumulées.

Tout au long du problème, nous cherchons à mettre en évidence ces idées sur l'exemple suivant :

$$I_{a,b} = \int_a^{a+b} e^{-x^2}\,\mathrm{d}x,$$

où $a > 0$ et $b \geq 1$, a étant pensé comme grand.
Finalement, on note $f(a) \underset{a\to\infty}{\lesssim} g(a)$ si f est asymptotique inférieur à g :

$$\forall \epsilon > 0,\, \exists A > 0,\, \forall a \geq A, \quad f(a) \leq g(a)(1+\epsilon)$$

soit, si g est strictement positive pour a assez grand, $\limsup_{a\to\infty} f(a)/g(a) \leq 1$.

(1) Montrer que

$$e^{-(a^2+1)} \int_0^1 e^{-2ax}\,\mathrm{d}x \leq I_{a,b} \leq e^{-a^2} \int_0^b e^{-2ax}\,\mathrm{d}x\,.$$

Et en déduire que

$$\frac{1}{2a}e^{-(a^2+1)}\left(1 - e^{-2a}\right) \leq I_{a,b} \leq \frac{1}{2a}e^{-a^2}\left(1 - e^{-2ab}\right)$$

et donc que

$$\frac{1}{2a}e^{-(a^2+1)} \underset{a\to\infty}{\lesssim} I_{a,b} \underset{a\to\infty}{\lesssim} \frac{1}{2a}e^{-a^2}$$

En effectuant le changmeent de variable $x = a + y$, on voit que

$$\begin{aligned} I_{a,b} &= \int_0^b e^{-(a+y)^2}\,\mathrm{d}y \\ &= e^{-a^2}\int_0^b e^{-2ay-y^2}\,\mathrm{d}y. \end{aligned}$$

Comme pour tout $y \in [0;b]$, $e^{-y^2} \leq 1$, on obtient la borne supérieure demandée. Pour la borne inférieure, il suffit de remarquer que, comme on intègre une fonction positive,

$$\int_0^b e^{-2ay-y^2}\,\mathrm{d}y \geq \int_0^1 e^{-2ay-y^2}\,\mathrm{d}y \geq e^{-1}\int_0^1 e^{-2ay}\,\mathrm{d}y\,.$$

Le calcul des intégrales intervenant dans les bornes supérieure et inférieure donne les autres bornes demandées.

(2) (a) Montrer que $I_{a,b}$ peut être mis sous la forme

$$I_{a,b} = \sqrt{\pi}\mathbb{E}\left[\mathbb{1}_{[a,a+b]}(X)\right]\,,$$

où X suit la loi gaussienne $\mathcal{N}(0,1/2)$.

Par définition de la loi $\mathcal{N}(0,\frac{1}{2})$, on a

$$\sqrt{\pi}\mathbb{E}\left[\mathbb{1}_{[a,a+b]}(X)\right] = \sqrt{\pi}\int_{\mathbb{R}} \mathbb{1}_{[a,a+b]}(x)\frac{1}{\sqrt{2\pi\frac{1}{2}}}e^{-\frac{x^2}{2\times 1/2}}\,\mathrm{d}x = I_{a,b}\,.$$

(b) Montrer que la variance σ^2 associée à l'estimateur Monte-Carlo ci-dessus vérifie

$$\frac{\sqrt{\pi}}{2a}e^{-(a^2+1)}\left(1-e^{-2a}\right) \leq \sigma^2 + I_{a,b}^2 \leq \frac{\sqrt{\pi}}{2a}e^{-a^2}\left(1-e^{-2ab}\right)\,.$$

Montrer que l'erreur relative $\frac{\sigma}{\sqrt{n}I_{a,b}}$ vérifie, pour b et n fixés, quand a est grand :

$$\frac{\pi^{1/4}}{\sqrt{n}}\sqrt{2a}e^{\frac{a^2+1}{2}} \underset{a\to\infty}{\lesssim} \frac{\sigma}{\sqrt{n}I_{a,b}} \underset{a\to\infty}{\lesssim} \frac{\pi^{1/4}}{\sqrt{n}}\sqrt{2a}e^{\frac{a^2}{2}+1}$$

On s'intéresse à la variance $\sigma^2 = \mathbb{V}\left[\sqrt{\pi}\mathbb{1}_{[a,a+b]}(X)\right]$ où $X \sim \mathcal{N}(0,1/2)$. Ainsi,

$$\sigma^2 = \mathbb{E}\left[\pi\mathbb{1}_{[a,a+b]}(X)^2\right] - \mathbb{E}\left[\sqrt{\pi}\mathbb{1}_{[a,a+b]}(X)\right]^2 = \pi\mathbb{E}\left[\mathbb{1}_{[a,a+b]}(X)\right] - I_{a,b}^2 = \sqrt{\pi}I_{a,b} - I_{a,b}^2\,.$$

Les bornes demandées dans la question sont alors des conséquences directes du résultat de la question (1).

Le rapport de l'écart-type de la moyenne empirique sur son espérance est obtenu en utilisant la borne pécédente et encore une fois le résulat de la question (1) pour $I_{a,b}$.

(c) Mettre en place sous la forme d'une fonction `Python` une méthode de Monte-Carlo à n tirages renvoyant la valeur estimée ainsi que la variance associée. Calculer le rapport écart-type / valeur estimée pour $b = 1$ et a variant de 1 à 2 par pas de 1/10 et vérifier que le rapport est alors une fonction croissante de a.

```
def MC1(n,a,b):
    r=np.sqrt(.5)*np.random.normal(0,1,n)
    tirages=np.sqrt(np.pi)*(r>=a).astype(int)*(r<=(a+b)).astype(int)
    return([np.mean(tirages),np.std(tirages,ddof=1)])
```

(3) (a) A l'aide d'un changement de variable, montrer que $I_{a,b}$ peut être mis sous la forme

$$I_{a,b} = \sqrt{\pi}e^{-a^2}\mathbb{E}\left[e^{-2aX}\mathbf{1}_{[0,b]}(X)\right]\,,$$

où X suit la loi gaussienne $\mathcal{N}(0,1/2)$.

Cette nouvelle écriture permet de ramener l'intervalle d'intégration à des valeurs qui sont typiques de la loi gaussienne.

En utilisant la densité de la loi $\mathcal{N}(0,1/2)$, on obtient

$$\begin{aligned} \sqrt{\pi}e^{-a^2}\mathbb{E}\left[e^{-2aX}\mathbf{1}_{[0,b]}(X)\right] &= \sqrt{\pi}e^{-a^2}\int_0^b e^{-2ax}\frac{1}{\sqrt{\pi}}e^{-x^2}\,\mathrm{d}x \\ &= \int_0^b e^{-x^2-2ax-a^2}\,\mathrm{d}x = \int_0^b e^{-(x+a)^2}\,\mathrm{d}x\,. \end{aligned}$$

Le changement de variable $y = x + a$ nous permet alors bien de retrouver $I_{a,b}$.

(b) Montrer que la variance σ^2 associée à l'estimateur Monte-Carlo ci-dessus vérifie

$$\sigma^2 + I_{a,b}^2 \leq \frac{\sqrt{\pi}}{4a} e^{-2a^2}.$$

En déduire que l'erreur relative vérifie :

$$\frac{\sigma}{\sqrt{n} I_{a,b}} \underset{a\to\infty}{\lesssim} \frac{e\pi^{1/4}}{n}\sqrt{2a}\,.$$

On s'intéresse ici à la variance $\sigma^2 = \mathbb{V}\left[\sqrt{\pi} e^{-a^2-2aX} \mathbb{1}_{[0,b]}(X)\right]$ où $X \sim \mathcal{N}(0, 1/2)$. Ainsi, on a

$$\begin{aligned}\sigma^2 + I_{a,b}^2 &= \mathbb{E}\left[\pi e^{-2a^2-4aX} \mathbb{1}_{[0,b]}(X)\right] \\ &= \sqrt{\pi} e^{2a^2}\left(\sqrt{\pi} e^{-(2a)^2} \mathbb{E}\left[e^{-2(2a)X} \mathbb{1}_{[0,b]}(X)\right]\right) \\ &= \sqrt{\pi} e^{2a^2} I_{2a,b}\,.\end{aligned}$$

Les résultats de la question (1) donne alors la majoration voulue.
Le rapport de l'écart-type de la moyenne empirique sur son espérance est obtenu en utilisant la borne précédente et encore une fois le résulat de la question (1) pour $I_{a,b}$.

(c) Mettre en place sous la forme d'une fonction `Python` une méthode de Monte-Carlo avec n tirages renvoyant la valeur estimée ainsi que la variance associée. Comparer (pour $n = 10^4$) les résultats obtenus avec ceux de la question précédente : on prendra $b = 1$ et on fera varier a de 1 à 2 par pas de 1/5.

```
def MC2(n,a,b):
      r=np.sqrt(.5)*np.random.normal(0,1,n)
      tirages=np.sqrt(np.pi)*np.exp(-a**2)*np.exp(-2*a*r)*(r<=b).astype(int)*(r>=0).astype(int)
      return([np.mean(tirages),np.std(tirages,ddof=1)])


# representation graphique


res=np.zeros(12).reshape(2,6)
i=0
for a in np.arange(2,3.1,.2):
       res[0,i]=MC1(int(1E4),a,1)[1]
       res[1,i]=MC2(int(1E4),a,1)[1]
       i=i+1
plt.figure()
plt.plot(np.arange(2,3.1,.2),res[0,],c="r")
plt.plot(np.arange(2,3.1,.2),res[1,],c="b")
plt.show()

#ratio ecart-type/estimation
i=0
res=np.zeros(11)
for a in np.arange(1,2.05,.1):
      res[i]=MC1(int(1E4),a,1)[1]/MC1(int(1E4),a,1)[0]
      i=i+1

print(res)
```

(4) (a) À l'aide d'une méthode d'échantillonnage préférentiel, montrer que $I_{a,b}$ peut être mis sous la forme

$$I_{a,b} = \frac{1}{2a} e^{-a^2} \mathbb{E}\left[e^{-Y^2} \mathbb{1}_{[0,b]}(Y)\right],$$

où Y suit une loi exponentielle de paramètre $2a$.
Cette nouvelle écriture est intéressante en particulier lorsque a est grand : le cas échéant, les valeurs typiques de Y sont concentrées au voisinage de 0 et la fonction à intégrer peut être vue comme très proche d'une constante.

La définition de la loi exponentielle nous montre que :

$$\frac{1}{2a}e^{-a^2}\mathbb{E}\left[e^{-Y^2}\mathbb{1}_{[0,b]}(Y)\right] = \frac{1}{2a}e^{-a^2}\int_0^b e^{-y^2}2ae^{-2ay}\,\mathrm{d}y$$
$$= \int_0^b e^{-(y+a)^2}\,\mathrm{d}y\,.$$

On retrouve alors l'intégrale $I_{a,b}$ en effectuant le changement de variable $x = y + a$.

(b) Montrer que la variance σ^2 associée à l'estimateur Monte-Carlo ci-dessus vérifie

$$\sigma^2 + I_{a,b}^2 \leq \frac{1}{4a^2}e^{-2a^2}.$$

En déduire que l'erreur relative vérifie

$$\frac{\sigma}{\sqrt{n}I_{a,b}} \underset{a\to\infty}{\lesssim} \frac{e}{\sqrt{n}}$$

On a

$$\sigma^2 + I_{a,b}^2 = \frac{1}{4a^2}e^{-2a^2}\mathbb{E}\left[e^{-2Y^2}\mathbb{1}_{[0,b]}(Y)\right].$$

L'expression à l'intérieur de l'espérance étant inférieure à 1, on obtient bien le résultat demandé. La borne asymptotique s'obtient alors aisément.

(c) Mettre en place sous la forme d'une fonction `Python` une méthode de Monte-Carlo avec n tirages renvoyant la valeur estimée ainsi que la variance associée. Comparer (pour $n = 10^4$) les résultats obtenus avec ceux de la question précédente : on prendra $b = 1$ et on fera varier a de 1 à 2 par pas de 1/5.
Calculer par ailleurs le rapport écart-type / valeur estimée pour $b = 1$ et a variant de 1 à 2 par pas de 1/10.

```
def MC3(n,a,b):
	r=-np.log(np.random.rand(n))/(2*a)
	tirages=np.exp(-a**2)*(1/(2*a))*np.exp(-r**2)*(r<=b).astype(int)
	return([np.mean(tirages),np.std(tirages,ddof=1)])

# representation graphique

res=np.zeros(12).reshape(2,6)
i=0
for a in np.arange(2,3.1,.2):
	res[0,i]=MC2(int(1E4),a,1)[1]
	res[1,i]=MC3(int(1E4),a,1)[1]
	i=i+1
fig=plt.figure()
plt.plot(np.arange(2,3.1,.2),res[0,],c="r")
plt.plot(np.arange(2,3.1,.2),res[1,],c="b")
plt.show()


#ratio ecart-type/estimation
i=0
res=np.zeros(11)
for a in np.arange(1,2.1,.1):
	i=i+1
	res[i]=MC3(int(1E4),a,1)[1]/MC3(int(1E4),a,1)[0]
```

(5) (a) À l'aide d'une autre méthode d'échantillonnage préférentiel, montrer que $I_{a,b}$ peut être mis sous la forme

$$I_{a,b} = be^{-a^2}\mathbb{E}\left[e^{-U^2-2aU}\right],$$

où U suit une loi uniforme sur $[0, b]$.
Cette nouvelle écriture est particulièrement intéressante lorsque b et a sont petits: le cas échéant, l'exponentielle dans l'espérance ne varie pas trop.

On écrit cette fois :

$$be^{-a^2}\mathbb{E}\left[e^{-U^2-2aU}\right] = be^{-a^2}\int_0^1 e^{-u^2-2au}\,\mathrm{d}u = b\int_0^1 e^{-(u+a)^2}\,\mathrm{d}u\,.$$

Le changement de variable $x = bu + a$ permet alors de retomber sur $I_{a,b}$.

(b) Montrer que la variance σ^2 associée à l'estimateur Monte-Carlo ci-dessus vérifie

$$\sigma^2 + I_{a,b}^2 \leq \frac{b}{4a}e^{-2a^2}.$$

On a maintenant

$$\sigma^2 + I_{a,b}^2 = b^2e^{-2a^2}\mathbb{E}\left[e^{-2U^2-4aU}\right] \leq b^2e^{-2a^2}\mathbb{E}\left[e^{-4aU}\right].$$

Il suffit alors de calculer

$$\mathbb{E}\left[e^{-4aU}\right] = \frac{1}{b}\int_0^b e^{-4au}\,\mathrm{d}u = \frac{1-e^{-4ab}}{4ab} \leq \frac{1}{4ab},$$

pour obtenir le résultat demandé. On obtient alors la borne suivante pour l'erreur relative :

$$\frac{\sigma}{\sqrt{n}I_{a,b}} \leq \frac{e\sqrt{ab}}{1-e^{-2a}}\,.$$

(c) Mettre en place sous la forme d'une fonction `Python` une méthode de Monte-Carlo avec n tirages renvoyant la valeur estimée ainsi que la variance associée. Comparer (pour $n = 10^4$) les résultats obtenus avec ceux de la question précédente : on prendra $a = 0,5$ et on fera varier b de 1 à 3 par pas de $1/5$.

```
def MC4(n,a,b):
      r=b*np.random.rand(n)
      tirages=b*np.exp(-a**2)*np.exp(-2*a*r-r**2)
      return([np.mean(tirages),np.std(tirages,ddof=1)])

# representation graphique

res=np.zeros(22).reshape(2,11)
i=0
for b in np.arange(1,3.1,.2):
       res[0,i]=MC3(int(1E4),.5,b)[1]
       res[1,i]=MC4(int(1E4),.5,b)[1]
       i=i+1

fig=plt.figure()
plt.ylim(np.min(res),np.max(res))
plt.plot(np.arange(1,3.1,.2),res[0,],c="r")
plt.plot(np.arange(1,3.1,.2),res[1,],c="b")
plt.show()
```