UM6P Probabilités appliquées et modélisation aléatoire MSD1

Issam El Kadiri
Issam.Elkadiri@um6p.ma

Exercice 1 :

Utilisation des méthodes Monte Carlo pour l'approximation des intégrales suivantes :

$$I = \int_{-2}^{2} e^{x+x^2} dx \qquad J = \int_{-\infty}^{+\infty} e^{-x^2} dx$$

In [137…
```python
import numpy as np
import matplotlib.pyplot as plt
def f_I(x) :
    return np.exp(x+x**2)

def f_J(x) :
    return np.exp(-x**2)

def mc_integrate(func, a, b, n ) : # Approximation d'une intégrale de func entre a et b avec n tirages.
    U = np.random.uniform(a, b, n)
    y = [func(val) for val in U]
    y_mean = np.sum(y)/n
    integ = (b-a) * y_mean
    return integ
```

In [43]:
```python
mc_integrate(f_I, -2, 2, 10000000)
```

Out[43]:
```
93.14012948984592
```

In [142…
```python
def simpson(f,x0,xn,n):
    # calculating step size
    h = (xn - x0) / n

    # Finding sum
    integration = f(x0) + f(xn)

    for i in range(1,n):
        k = x0 + i*h

        if i%2 == 0:
            integration = integration + 2 * f(k)
        else:
            integration = integration + 4 * f(k)

    # Finding final integration value
    integration = integration * h/3

    return integration

simpson(f_I, -2, 2, 1000)
```

Out[142…
```
93.1627533818421
```

Ci-dessus nous avons une valeur approchée de la valeur de I grâce à une approxmiation déterministe par la méthode Simpson.

Pour approximer J, ayant pour borne supérieur $+\infty$, on peut se contenter de prendre comme borne supérieure 100, ou 1000, car la fonction $f(x) = e^{-x^2}$ décroit très rapidement vers 0. Numériquement, $f(100)$ est de l'ordre de 10^{-44}. On utilise également la parité de f pour ne calculer l'intégrale qu'à partir de 0.

In [55]:
```python
2*mc_integrate(f_J, 0, 1000, 10000000)
```

Out[55]:
```
1.770058412606884
```

La valeur exact de J est connue (classique), nous avons :

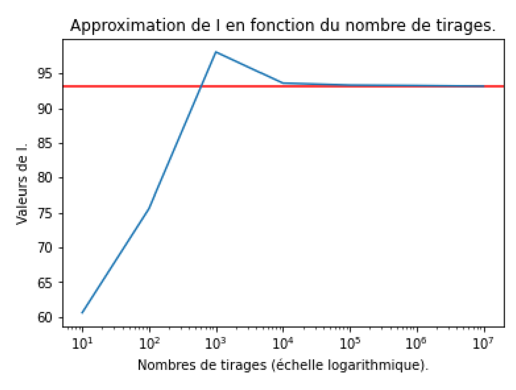
$$J = \sqrt{\pi} \approx 1.77245385091$$

In [93]:
```python
def comparaisonI():
    L=[10, 100, 1000, 10000, 100000, 1000000, 10000000]
    compI=[]
    for i in range(len(L)) :
        compI.append(mc_integrate(f_I, -2, 2, L[i]))
    return compI
```

In [128…
```python
comp1=comparaisonI()
```

In [143…
```python
L=[10, 100, 1000, 10000, 100000, 1000000, 10000000]
plt.xscale("log")
plt.axhline(y=93.16, color="red")
plt.title("Approximation de I en fonction du nombre de tirages.")
plt.xlabel("Nombres de tirages (échelle logarithmique).")
plt.ylabel("Valeurs de I.")
plt.plot( L,comp1)
```

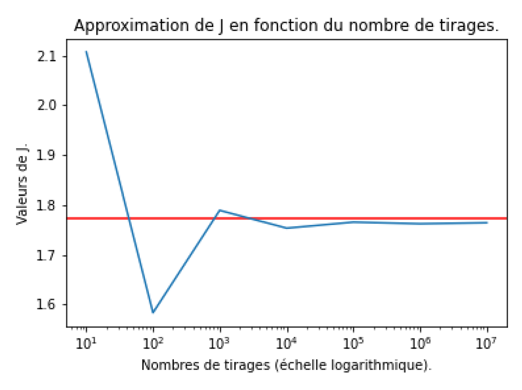
Out[143…
```
[<matplotlib.lines.Line2D at 0x1dc068ba9a0>]
```



In [132…
```python
def comparaisonJ():
    L=[10, 100, 1000, 10000, 100000, 1000000, 10000000]
    compJ=[]
    for i in range(len(L)) :
        compJ.append(mc_integrate(f_J, -2, 2, L[i]))
    return compJ

comp2=comparaisonJ()
```

In [135…
```python
plt.xscale("log")
plt.axhline(y=np.sqrt(np.pi), color="red")
plt.title("Approximation de J en fonction du nombre de tirages.")
plt.xlabel("Nombres de tirages (échelle logarithmique).")
plt.ylabel("Valeurs de J.")
plt.plot( L,comp2)
```

Out[135…
```
[<matplotlib.lines.Line2D at 0x1dc07220730>]
```



Exercice 2 :

Considérons une variable aléatoire réelle continue X de densité de probabilité

$$f(x) = \theta x^{\theta-1} 1_{[0,1[}, \quad x \in \mathbb{R}$$

où $\theta > 0$ est un paramètre.

1) Espérance et variance de X :

$$\mathbb{E}(X) = \int_{-\infty}^{+\infty} x f(x) dx = \int_0^1 \theta x^{\theta} dx = \frac{\theta}{\theta+1}$$

.

$$\mathbb{V}(X) = \int_{-\infty}^{+\infty} x^2 f(x) dx - \mathbb{E}(X)^2 = \int_0^1 \theta x^{\theta+1} dx - \mathbb{E}(X)^2 = \frac{\theta}{\theta+2} - \frac{\theta^2}{(\theta+1)^2}$$

.

2) Génération de n réalisations de la variable aléatoire X :

Algorithme d'acceptation-rejet : g densité d'une unifrome sur $[0, 1[$ avec comme constante de majoration θ. Pour θ fixé, on a : Pour tout x dans $[0, 1[$:

$$f(x) \leq \theta g(x)$$

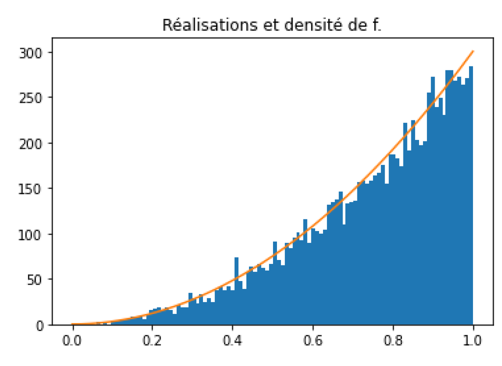
In [194…
```python
def fx(theta, x):
    return theta*x**(theta-1)

def AR(theta) :
    U1=np.random.uniform(0,1)
    G=theta*np.random.uniform(0,1)
    r=-8
    if G<=fx(theta, U1) :
        r=U1
    else :
        while G>fx(theta, U1) :
            U1=np.random.uniform(0,1)
            G=theta*np.random.uniform(0,1)
        r=U1
    return r
def esperance(theta):
    return theta/(theta+1)
def variance(theta) :
    return theta/(theta+2) - (theta**2)/((1+theta)**2)
```

3) Tracer l'histogramme des réalisations et le confronter à la densité de f pour $\theta = 3$ et $n = 10000$.

In [201…
```python
n=10000
theta=3
realisations=[]
for i in range(n):
    realisations.append(AR(theta))
```

In [240…
```python
plt.hist(realisations, bins=100)
intervalle=np.linspace(0,1,1000)
y=[]
for i in range(len(intervalle)):
    y.append(fx(theta, intervalle[i]))
plt.title("Réalisations et densité de f.")
plt.plot(intervalle,100*np.array(y))
```

Out[240…
```
[<matplotlib.lines.Line2D at 0x1dc16bde1f0>]
```



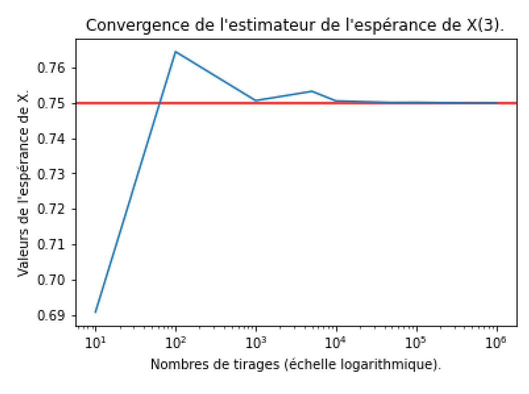
4) Montrer graphiquement la convergence des estimateurs de l'espérance et la variance de X vers leurs valeurs théoriques pour $\theta = 0.9$ et $\theta = 3$.

In [259…
```python
N=[10, 100, 1000, 5000, 10000, 50000, 100000, 500000, 1000000]
esp1=[]
for i in range (len(N)) :
    esp1.append(np.mean([AR(3) for i in range(N[i])]))
esp1
```

Out[259…
```
[0.6908878723399823,
 0.7643936416128787,
 0.7505761927449662,
 0.7531881055697779,
 0.7504700570574039,
 0.7500265045814494,
 0.7500684851627792,
 0.7498746808697127,
 0.7499129499025124]
```

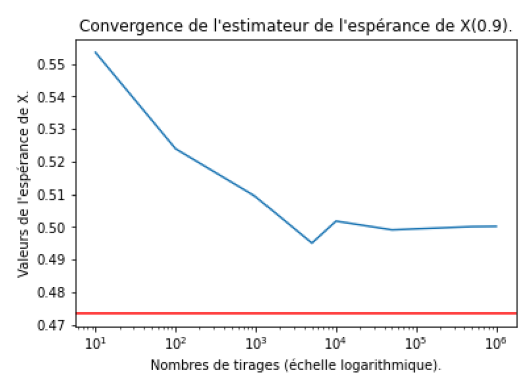
In [260…
```python
plt.xscale("log")
plt.axhline(y=esperance(3), color="red")
plt.title("Convergence de l'estimateur de l'espérance de X(3).")
plt.xlabel("Nombres de tirages (échelle logarithmique).")
plt.ylabel("Valeurs de l'espérance de X.")
plt.plot( N,esp1)
```

Out[260…
```
[<matplotlib.lines.Line2D at 0x1dc18110d00>]
```



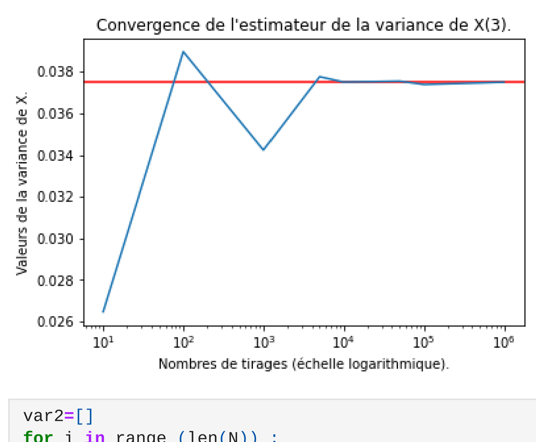
In [263…
```python
esp2=[]
for i in range (len(N)) :
    esp2.append(np.mean([AR(0.9) for i in range(N[i])]))
esp2
plt.xscale("log")
plt.axhline(y=esperance(0.9), color="red")
plt.title("Convergence de l'estimateur de l'espérance de X(0.9).")
plt.xlabel("Nombres de tirages (échelle logarithmique).")
plt.ylabel("Valeurs de l'espérance de X.")
plt.plot( N,esp2)
```

Out[263…
```
[<matplotlib.lines.Line2D at 0x1dc1a77e070>]
```



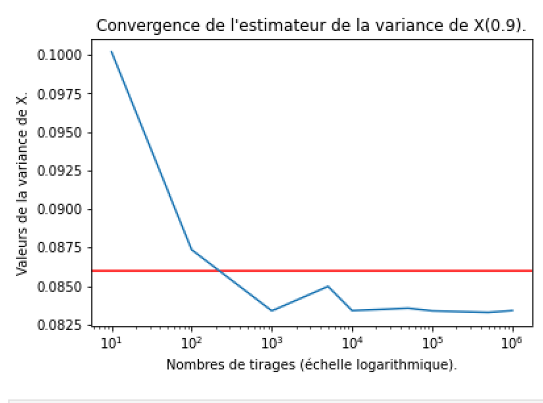
In [271…
```python
var1=[]
for i in range (len(N)) :
    var1.append(np.var([AR(3) for i in range(N[i])]))
var1

plt.xscale("log")
plt.axhline(y=variance(3), color="red")
plt.title("Convergence de l'estimateur de la variance de X(3).")
plt.xlabel("Nombres de tirages (échelle logarithmique).")
plt.ylabel("Valeurs de la variance de X.")
plt.plot( N,var1)
```

Out[271…
```
[<matplotlib.lines.Line2D at 0x1dc1a957910>]
```



In [272…
```python
var2=[]
for i in range (len(N)) :
    var2.append(np.var([AR(0.9) for i in range(N[i])]))
var2

plt.xscale("log")
plt.axhline(y=variance(0.9), color="red")
plt.title("Convergence de l'estimateur de la variance de X(0.9).")
plt.xlabel("Nombres de tirages (échelle logarithmique).")
plt.ylabel("Valeurs de la variance de X.")
plt.plot( N,var2)
```

Out[272…
```
[<matplotlib.lines.Line2D at 0x1dc1a6fd130>]
```



In [ ]: