Sorbonne Université **3M246 : Processus et simulation**

Contrôle TP
4 mai 2018

On commence par importer les librairies nécessaires :

```
from random import random
import numpy as np
import matplotlib.pyplot as plt
```

Exercice 1

On commence par calculer la fonction de répartition associée à X :

$$F(x) = \int_{-\infty}^{x} f(t)dt = \int_{0}^{x} te^{-t^2/2}dt = -\int_{0}^{x} (-te^{-t^2/2})dt = -[e^{-t^2/2}]_0^x = 1 - e^{-x^2/2}$$

Il suffit ensuite d'inverser F :

$$F(x) = u \Leftrightarrow 1 - e^{-x^2/2} = u \Leftrightarrow x = \sqrt{-2\ln(1-u)}$$

Pour tirer X, il suffit de tirer $U \sim \mathcal{U}([0,1])$ puis de retourner $X = \sqrt{-2\ln(1-U)}$. Comme U et $1-U$ ont même loi, on peut retourner $X = \sqrt{-2\ln U}$.

```
def simuX():
    return np.sqrt(-2*np.log(random()))
```

On peut tirer un échantillon de taille $N = 10000$.

```
N = 10000
l = [simuX() for i in range(N)]
plt.hist(l, normed=True)

lx = np.arange(0, 10., 0.05)
lfx = [x*np.exp(-x**2/2) for x in lx]
plt.plot(lx, lfx, 'r')

plt.show()
```

Exercice 2

On souhaite simuler une variable X dont la densité est donnée par :

$$f(x) = \frac{3}{4}(1-x^2)\mathbf{1}_{[-1,1]}(x)$$

On commence par tracer la fonction f et on se rend compte que f est bornée par $y = \frac{3}{4}$.
Il suffit de simuler un couple (X,Y) dont la loi est uniforme sur $G = \{(x,y) \in [-1,1] \times [0,\frac{3}{4}]; y < f(x)\}$.
On sait, alors que X a pour densité f.
Pour simuler une loi uniforme sur G, il suffit de simuler un couple (X,Y) suivant une loi uniforme sur le rectangle $[-1,1] \times [0,\frac{3}{4}]$, et vérifier si (X,Y) est dans G. Si ce n'est pas le cas, on recommence.

```
def simu_rejet():
    while True:
        x, y = 2.*random()-1., 3./4.*random()
        if y < 3./4.*(1.-x**2):
            return x
```

On peut tirer un échantillon de taille $N = 10000$.

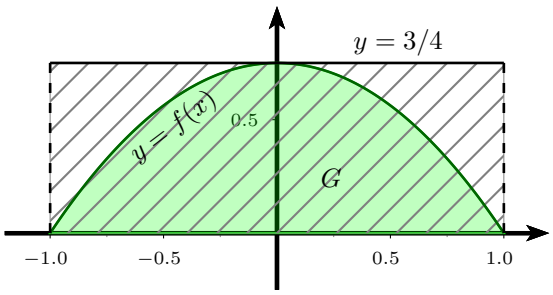

FIGURE 1 – On tire les couples (X, Y) dans la zone hachurée $[-1, 1] \times [0, \frac{3}{4}]$, et on ne garde que ceux qui sont dans la zone verte G

```
N = 10000
l = [simu_rejet() for i in range(N)]
plt.hist(l, normed=True)

lx = np.arange(-1., 1., 0.01)
lfx = [3./4.* (1.-x**2) for x in lx]
plt.plot(lx, lfx, 'r')

plt.show()
```

Exercice 3

On remarque que :

$$\begin{aligned} I &= \int_{-\infty}^{\infty} 3\sqrt{|x|}\mathrm{e}^{-3|x-2|}dx \\ &= \int_{-\infty}^{0} 3\sqrt{-x}\mathrm{e}^{-3(2-x)}dx + \int_{0}^{2} 3\sqrt{x}\mathrm{e}^{-3(2-x)}dx + \int_{2}^{\infty} 3\sqrt{x}\mathrm{e}^{-3(x-2)}dx \\ &= \int_{2}^{\infty} 3\sqrt{u-2}\mathrm{e}^{-3u}du + \int_{0}^{2} 3\sqrt{2-u}\mathrm{e}^{-3u}du + \int_{0}^{\infty} 3\sqrt{u+2}\mathrm{e}^{-3u}du \\ &= \int_{0}^{\infty} 3(\sqrt{u+2} + \sqrt{|2-u|})e^{-3u}du \end{aligned}$$

On souhaite calculer $I = \int_0^\infty 3(\sqrt{x+2} + \sqrt{|2-x|})e^{-3x}dx$. I peut se réécrire :

$$I = \int_0^\infty (\sqrt{x+2} + \sqrt{|2-x|})f(x)dx = \int_0^\infty g(x)f(x)dx$$

où $f : x \mapsto 3e^{-3x}$ est la densité de la loi exponentielle de paramètre 3 et $g : x \mapsto \sqrt{x+2} + \sqrt{|2-x|}$.

Il suffit de tirer un échantillon $X_1, \dots X_N$ de densité f, puis de calculer $\widehat{I}(N) = \dfrac{1}{N}\displaystyle\sum_{k=1}^{N} g(X_k)$.

Pour tirer suivant f, on utilise la méthode d'inversion. La fonction de répartition associée est $F : x \mapsto 1 - e^{-3x}$, d'inverse $F^{-1} : u \mapsto -\dfrac{1}{3}\ln(1-u)$. Ainsi, Si $U \sim \mathcal{U}([0,1])$, alors $X = -\dfrac{1}{3}\ln U$ suit une loi exponentielle de paramètre 3.

La variance σ^2 de l'estimateur I est $\sigma^2 = \dfrac{1}{N}\displaystyle\sum_{k=1}^{N} g(X_k)^2 - \left(\frac{1}{N}\sum_{k=1}^{N} g(X_k)\right)^2$.

L'intervalle de confiance à 95% est donné par $\left[\widehat{I}(N) - 1.96\dfrac{\sigma}{\sqrt{N}}, \widehat{I}(N) + 1.96\dfrac{\sigma}{\sqrt{N}}\right]$.

```
def g(x):
    return np.sqrt(x+2) + np.sqrt(np.abs(x-2))

def integrale(N):
   sum = 0
   sum_square = 0
   for k in range(N):
      x = -1/3.*np.log(random())
      sum += g(x)
      sum_square += g(x)**2
   I = sum/N
   var = sum_square/N - I**2
   intervalle = [I - 1.96*np.sqrt(var/N), I + 1.96*np.sqrt(var/N)]
   return I, var, intervalle
```

On peut estimer $\widehat{I}(N)$ à partir d'un échantillon de taille $N = 10000$.

```
integrale_full(10000)
```

Exercice 4

Pour tirer une variable S_N suivant une loi binomiale de paramètres (N, p), on tire N variables X_k i.i.d. suivant une loi de Bernoulli de paramètre p et on calcule $S_N = \sum_{k=1}^{N} X_k$.

```
def bernoulli(p):
    return 1*(random()<p)

def binom(N,p):
    return sum([bernoulli(p) for i in range(N)])
```

Si on considère être dans l'état i à un instant n, alors l'état suivant est déterminé suivant une loi binomiale de paramètres $(N, i/N)$.
La fonction `WrightFisher` prend en argument un entier k et un entier i et renvoie la trajectoire de k pas de (X_n), avec $X_0 = i$.

```
N = 100

def WrightFisher(k, i):
    Xn = i
    traj = [Xn]            # contient les positions successives de la chaine.
    for j in range(k):
        Xn = binom(N, Xn/N)
        traj.append(Xn)
    return traj
```

On teste cette fonction sur les cas suivants :

```
plt.plot(WrightFisher(k=100, i=50))
plt.show()

plt.plot(WrightFisher(k=500, i=50))
plt.show()
```

On remarque qu'au bout d'un certain nombre de générations, il n'y a plus qu'un seul type dominant.