1 Introduction

Le module `random` de `python` contient la fonction `random()` permettant de simuler une variable aléatoire continue qui suit une loi uniforme. Il contient aussi des fonctions prédéfinies permettant de simuler des variables aléatoires non-uniformes :

`random.expovariate(lambda)`

permet de simuler une variable aléatoire suivant la loi exponentielle de paramètre `lambda` : $\mathscr{E}(\lambda)$.

`random.normalvariate(mu, sigma)`

permet de simuler une variable aléatoire suivant la loi normale de paramètres : espérance `mu`, écart-type `sigma`, c'est à dire : $\mathscr{N}(\mu, \sigma^2)$.

2 Exemple : Simulation d'une loi exponentielle

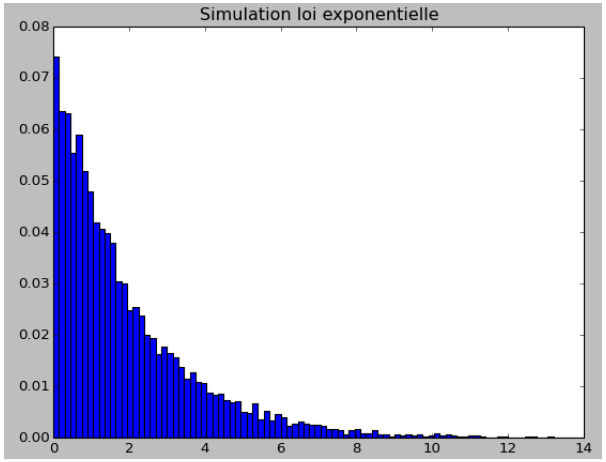
On reprend l'exercice 2 du TD7 qui simulait la loi exponentielle $\mathscr{E}(0,5)$. Mais plutôt que de simuler la loi uniforme $\mathscr{U}[0,1]$ et d'appliquer la méthode de la fonction inverse, on utilise la fonction `expovariate(0.5)` pour simuler directement notre loi exponentielle ; comme d'habitude on répète N fois l'expérience aléatoire et on trace les fréquence des résultats obtenus.

```
from random import expovariate
def varExp(param):
    return expovariate(param)
Max = 15
def simulExp(N,n, param):
    FreqX = [0] * n
    Nbre = 0
    for k in range(N):
        x = varExp(param)
        if x < Max:
            i = int(x/Max * n)
            FreqX[i] += 1
            Nbre += 1
    return [x/Nbre for x in FreqX]

# Simulation et tracé des fréquences obtenues
N = 10000    # Nombre de répétitions
n = 100    # Nombre d'intervalles considérés
FreqX = simulExp(N, n, 0.5)
```

```
import numpy as np
import matplotlib.pyplot as plt
I = np.arange(0,Max,Max/n)
plt.title("Simulation loi exponentielle")
plt.bar(I,FreqX,width = Max/n)
plt.show()
```

Ce qui produit le graphique :



On pourrait alors, de la même façon que dans le TD7, produire les histogrammes de la fonction de répartition et de la fonction densité.

3 Simulation d'une loi normale

Sur le même modèle on utilise la fonction `random.normalvariate(0,1)` pour simuler la loi normale $\mathscr{N}(0,1)$:

```
from random import normalvariate
def varNormale(mu,sigma):
    return normalvariate(mu,sigma)
```

Exercice 1
A l'aide de l'inégalité de Bienaymé-Tchebychev déterminer un réel a tel que si $X \hookrightarrow \mathscr{N}(0,1)$, alors $P(-a \leqslant X \leqslant a) \geqslant 0,96$.

Réponse. D'après l'inégalité de Bienaymé-Tchebychev : $P(|X-E(X)| \geqslant a) \leqslant \dfrac{\sigma^2}{a^2}$. Ainsi si $X \hookrightarrow \mathscr{N}(0,1)$ alors :

$$P(-a \leqslant X \leqslant a) = 1 - P(|X| > a) = 1 - P(|X| \geqslant a) \geqslant 1 - \frac{1}{a^2}$$

Il suffit donc de prendre $a = 5$, en effet $1 - \dfrac{1}{5^2} = 1 - 0.04 = 0.96$.

Exercice 2

1. Ecrire une fonction `simulNormale(N,n,mu,sigma)` qui simulera le résultat de la var $X \hookrightarrow \mathscr{N}(0,1)$ pour N expériences aléatoires et retournera la fréquence d'obtention de chaque résultat.
   – On ne tiendra compte que des résultats dans l'intervalle $I = [-5,5]$.
   – Les fréquences seront stockés dans une liste `freqX` de longueur $2n+1$. Le résultat $X = x \in [-5,5]$ augmentera l'élément de `freqX` d'indice $\lfloor n * x/5 \rfloor + n$ (la partie entière s'obtenant grâce à la fonction `floor` du module `math`).
2. Effectuer la simulation paramètres : `N= 100 000`, `n=100`, `mu=0` et `sigma = 1`
   Puis effectuer le tracé de l'histogramme des fréquences obtenues.
3. En déduire le tableau des valeurs de la fonction densité, puis effectuer le tracé de son histogramme.
4. En déduire le tableau des valeurs de la fonction de répartition, puis effectuer le tracé de son histogramme.