# TP Informatique n° 8
# Simulation de variables aléatoires à densité

Le but de ce TP est de mettre en oeuvre la simulation des différentes variables à densité rencontrées dans le chapitre, et d'apprendre à utiliser les bibliothèques `numpy.random` et `scipy.stats`.

Listing 1 – Bibliothèques utiles dans le TP

```
import numpy as np
import math
import matplotlib.pyplot as plt
import random as rd
import scipy.stats as sps
```

## 1 Inversion de la fonction de répartition

Le théorème utilisé dans cette partie pour simuler des variables aléatoires à densité est le suivant :

**Théorème 1** Soit $X$ une variable aléatoire à densité de fonction de répartition $F$ strictement croissante sur $]a,b[$, nulle sur $]-\infty,a]$ et égale à 1 sur $[b,+\infty[$. Alors $F$ est une bijection de $]a,b[$ sur $]0,1[$ et

$$F(X) \hookrightarrow \mathcal{U}[0,1]$$

**Remarque 1** Ainsi, si $U \hookrightarrow \mathcal{U}[0,1]$, alors $F^{-1}(U)$ a même loi que $X$.

**Exercice 1** Ecrire une fonction `unif(a,b)` qui simule une variable aléatoire de loi $\mathcal{U}[a,b]$ , $(a<b)$. ■

**Exercice 2** Ecrire une fonction `expo(mu)` qui simule une variable aléatoire de loi $\mathcal{E}(\mu)$, $(\mu>0)$. ■

**Exercice 3** Ecrire une fonction `cauchy()` qui simule une variable aléatoire de densité $f : x \mapsto \frac{1}{\pi(1+x^2)}$. ■

## 2 Utilisation des bibliothèques

Les deux bibliothèques de fonctions permettant de travailler avec les lois usuelles sont `numpy.random` et `scipy.stats`, la seconde étant plus complète puisqu'elle contient aussi les densités, fonctions de répartition et leur inverse.

Listing 2 – Densité et fonction de répartition de la loi exponentielle

```
X=np.linspace(0,8,200)

# densite de la loi expo(1)
plt.figure()
plt.plot(X,sps.expon.pdf(X,scale=1)) # ATTENTION : scale=1/mu (esperance)
plt.grid()
plt.show()

#fonction de repartition de la loi expo(1)
plt.figure()
plt.plot(X,sps.expon.cdf(X,scale=1))
plt.grid()
plt.show()
```

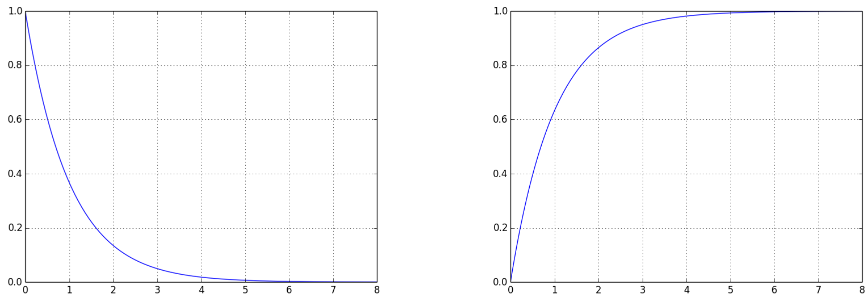


Figure 1 – Densité et fonction de répartition de la loi $\mathcal{E}(1)$

■ **Exercice 4** Vérifier que `np.random.exponential(scale=1.0,size=10)` et `sps.expon.rvs(scale=1,size=10)` permettent de simuler $n$ variables aléatoires indépendantes de loi exponentielle. Illustrer la loi des grands nombres. ■

**Remarque 2** Dans ces deux fonctions, la loi exponentielle $\mathcal{E}(\mu)$ est paramétrée par son espérance $\frac{1}{\mu}$ ! (Convention anglo-saxonne)

En cas de doute, se souvenir que $X \hookrightarrow \mathcal{E}(1) \Leftrightarrow \frac{1}{\mu}X \hookrightarrow \mathcal{E}(\mu)$ et ne simuler que des variable de loi $\mathcal{E}(1)$ ...

Listing 3 – Histogramme associé à un échantillon de 1000 simulations

```
S=sps.expon.rvs(scale=1,size=1000)

plt.figure()
plt.plot(X,sps.expon.pdf(X,scale=1),color="red")
plt.hist(S,30,normed=True)
plt.show()
```

Commenter le résultat obtenu.

# 3 Simulation de la loi $\mathcal{N}(0,1)$

**Proposition 1** Soit $\Phi$ la fonction de répartition d'une variable aléatoire de loi $\mathcal{N}(0,1)$. Alors $\Phi$ est continue et strictement croissante sur $\mathbb{R}$. C'est donc une bijection de $\mathbb{R}$ sur $]0,1[$.

En particulier, pour tout $\alpha \in ]0,1[$, il existe un unique réel $u_\alpha$ appelé quantile d'ordre $\alpha$ tel que $\Phi(u_\alpha) = \alpha$.

**Remarque 3**
1. $\Phi^{-1}$ est parfois appelée la fonction quantile.
2. La difficulté liée à la simulation de la loi normale est que l'inverse de sa fonction de répartition ne s'exprime pas à l'aide de fonctions mathématiques élémentaires.
3. Néanmoins, on peut obtenir grâce à Python les valeurs numériques approchées de $\Phi^{-1}$ via `sps.norm.ppf`. Ainsi, par la méthode d'inversion :

   ```
   >>> sps.norm.ppf(rd.random())
   ```

   simule une variable aléatoire de loi $\mathcal{N}(0,1)$.
4. Si $X \hookrightarrow \mathcal{N}(0,1)$, alors $m + \sigma X \hookrightarrow \mathcal{N}(m,\sigma^2)$. On peut donc simuler toutes les lois normales dès que l'on sait simuler la loi normale centrée réduite.

## 3.1 Utilisation d'une bibliothèque

La fonction `np.random.normal(m,sigma)` permet de simuler une variable aléatoire d'espérance $m$ et **d'écart-type** $\sigma$. Là encore, il faut faire attention à la convention de paramétrage de la loi normale. Dans le doute, simuler une variable aléatoire de loi $\mathcal{N}(0,1)$ et effectuer une transformation affine !

Une autre solution est d'utiliser les fonctions de la bibliothèque `Scipy`.

Listing 4 – Simulation de la loi $\mathcal{N}(0,1)$ via la bibliothèque Scipy

```
X=np.linspace(-5,5,400)
S=sps.norm.rvs(loc=0,scale=1,size=1000)
```

```

plt.figure()
plt.plot(X,sps.norm.pdf(X,loc=0,scale=1),color="red")
plt.hist(S,20,normed=True)
plt.show()
```

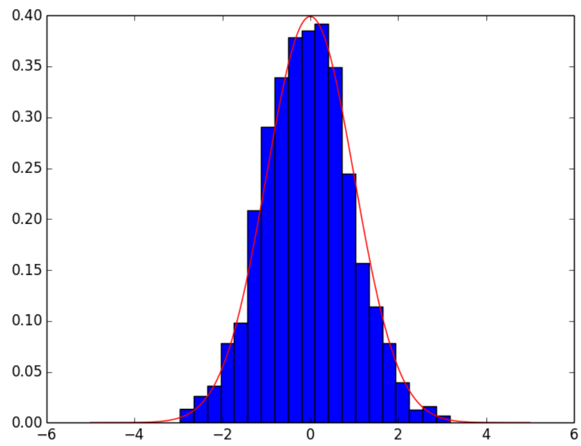


FIGURE 2 – Densité et histogramme des valeurs simulées de la loi $\mathcal{N}(0,1)$

**Remarque 4** Ainsi, pour mettre en évidence la loi d'une variable aléatoire $X$ lorsqu'on dispose d'observations $(x_1, \ldots, x_n)$ provenant d'un échantillon $(X_1, \ldots, X_n)$ de variables aléatoires indépendantes de même loi que $X$, on trace l'histogramme des valeurs observées. (Au fait, pourquoi ? ...)

## 3.2 Algorithme de Box-Muller

Le théorème suivant permet encore la simulation exacte d'une variable aléatoire de loi $\mathcal{N}(0,1)$ dès que l'on sait simuler la loi uniforme, il est donc simple à mettre en oeuvre.

**Théorème 2** Soient $U$ et $V$ deux variables aléatoires indépendantes de loi uniforme sur $[0;1]$. La variable aléatoire $\sqrt{-2\ln(U)}\cos(2\pi V)$ suit une loi normale $\mathcal{N}(0,1)$.

■ **Exercice 5** Ecrire une fonction qui simule une variable aléatoire de loi normale centrée réduite à l'aide de cette méthode. Effectuer 100 simulations et calculer la moyenne et l'écart type des résultats obtenus. ■

## 3.3 Le théorème limite central

On va utiliser l'approximation de la loi normale centrée réduite fournie par le théorème limite central pour simuler une variable aléatoire suivant **approximativement** cette loi.

**Proposition 2** Soit $(X_n)$ une suite de variables aléatoires mutuellement indépendantes de loi $\mathcal{U}[0,1]$. Alors :

$$\sqrt{\frac{12}{n}}\sum_{i=1}^{n} X_i - \sqrt{3n} \xrightarrow[n\to+\infty]{\mathcal{L}} Z \hookrightarrow \mathcal{N}(0,1)$$

■ **Exercice 6** Ecrire une fonction simulant la variable aléatoire centrée réduite associée à la somme de $n = 12$ variables aléatoires indépendantes de loi $\mathcal{U}[0,1]$. Effectuer $N = 1000$ simulations et comparer l'histogramme des valeurs obtenues avec la densité usuelle de la loi $\mathcal{N}(0,1)$. ■

Réciproquement, on souhaite illustrer la convergence en loi du théorème limite central : pour cela on considère $N$ variables aléatoires $X_1, \ldots, X_N$ indépendantes et de loi $\mathcal{B}(n,p)$.

■ **Exercice 7**

1. Soit $k \in [\![1, N]\!]$. Ecrire la variable $X_k^*$ centrée réduite associée à $X_k$ et justifier que pour $a < b$ :
$$\lim_{n\to+\infty} \mathbf{P}(a \leqslant X_k^* \leqslant b) = \Phi(b) - \Phi(a)$$
où $\Phi$ est la fonction de répartition de la loi $\mathcal{N}(0,1)$.
2. Illustrer graphiquement ce résultat (on prendra $n = 100$, $p = 0.5$ et $N = 1000$).
3. Soit $X \hookrightarrow \mathcal{B}(100, 0.5)$. Comment calculer $\mathbf{P}(X = k)$ en utlisant l'approximation du théorème limite central ? Evaluer numériquement l'erreur commise.

■

# 4 Méthode de Monte-Carlo

■ **Exercice 8** Le but est de calculer une valeur approchée de l'intégrale $I = \int_0^1 \frac{1}{1+t^2}dt$ à l'aide d'une méthode probabiliste.

1. On définit, pour $t \in [0;1]$, la fonction $g(t) = \frac{1}{1+t^2}$. Ecrire une fonction `g` qui prend en paramètre un réel `t` et qui renvoie la valeur de $g(t)$.
2. Soit $U$ une variable aléatoire de loi uniforme sur $[0;1]$. Justifier que $I = \mathbf{E}(g(U))$. Justifier que la variable aléatoire $g(U)$ admet une variance $\sigma^2$ non nulle.

3. Soit $(U_n)$ une suite de variables aléatoires indépendantes de loi uniforme sur $[0; 1]$. Rappeler la loi des grands nombres et justifier que

$$S_n = \frac{1}{n}\sum_{k=1}^{n} g(U_k)$$

converge en probabilité vers $I$.

4. Ecrire une fonction demandant à l'utilisateur un entier `n` et calculant une valeur approchée de $I$ à l'aide de la question précédente.
5. Comparer le résultat avec la valeur exacte de $I$. Comment évaluer l'erreur d'approximation commise ?