# Feuille de TP n°1 – Simulation par inversion

L'objectif de ce TP est d'implémenter des méthodes de simulations de variables aléatoires vues en cours. On dispose donc d'une loi de probabilité $\mathbb{P}$ et on chercher à simuler un tirage de $X_1, X_2, \ldots, X_n$ où les $X_i$ sont des v.a. i.i.d. de loi $\mathbb{P}$ et de fonction de répartition $F$.

## 1 Quelques lois classiques par inversion

La brique de départ est la simulation de la loi uniforme sur $[0, 1]$. On pourra y accéder de deux façons dans cette première partie :

— `numpy.random.rand(N)`

— `scipy.stats.uniform.rvs(size=N)`

Pour tous les exemples suivants, on créera des scripts (puis éventuellement des fonctions) qui renvoient en sortie un nombre N de réalisations de la loi donnée, le nombre N pouvant être fixé par l'utilisateur.

Par ailleurs vous mettrez au point un (ou plusieurs) script qui permette de voir que vous simulez bien la loi demandée. Vous pourrez utiliser les commandes :

— `hist` de la librairie `matplotlib.pyplot`, avec option `normed` : pour trace l'histogramme normalisé d'un échantillon X.

— les commandes `numpy.sort` et `numpy.linspace` permettent de tracer la fonction de répartition empirique de l'échantillon X.

**Exercice 1.1** (Fonction de répartition empirique)**.** Créer une fonction Python qui à une liste (ou un tableau) de données $x = (x_1, \ldots, x_n)$ et à un nombre $z$, renvoie la valeur de la fonction de répartition empirique au point $z$ :

$$F_n(z) = \frac{1}{n} \sum_{i=1}^{n} \mathbf{1}_{]-\infty,z]}(x_i).$$

Elle renvoie donc le nombre d'éléments $x_i$ inférieurs ou égaux à $z$, divisé par $n$.

Enfin vous vérifierez numériquement que si $F$ est la fonction de répartition de $X$, alors $F(X)$ suit la loi uniforme sur $[0, 1]$.

**Exercice 1.2** (Lois à densité)**.** Implémenter les lois suivantes :

1. loi uniforme sur $[a, b]$ (densité $f(x) = \frac{1}{b-a}\mathbf{1}_{(a,b)}(x)$) ;
2. loi exponentielle de paramètre $\lambda > 0$ (densité $f(x) = \lambda \exp(-\lambda x)\mathbf{1}_{\mathbb{R}_+}(x)$) ;
3. loi de Weibull de paramètres $(\lambda, \alpha) \in (\mathbb{R}_+^*)^2$ (densité $f(t) = \lambda\alpha t^{\alpha-1}e^{-\lambda t^\alpha},\ t > 0$) ;
4. loi de Pareto de paramètre $\lambda > 0$ (densité $f(x) = \frac{\lambda}{x^{\lambda+1}}\mathbf{1}_{[1,+\infty[}(x)$),
5. loi de Cauchy de paramètre $c > 0$ (densité $f(x) = \frac{c}{\pi(c^2 + x^2)}$).

**Exercice 1.3** (Lois discrètes)**.** Simuler la :

1. loi de Bernoulli de paramètre $p \in ]0, 1[$ ;
2. loi binômiale de paramètres $n \in \mathbb{N}^*$ et $p \in ]0, 1[$. On rappelle que si $U_1, \ldots, U_n$ sont $n$ v.a. i.i.d. de loi de Bernoulli de paramètre $p$, alors $X = U_1 + \ldots + U_n$ suit une loi binômiale de paramètres $n$ et $p$.

**Exercice 1.4.** Comprendre et commenter la fonction suivante :

```
def realis(x,p):
    n=len(x)
    U=scipy.stats.uniform.rvs(size=1)
    a=0
    b=p[0]
    for i in range(n-1):
        if U>=a and U<b:
            return x[i]
        else:
            a=b
            b=b+p[i+1]
    return x[n-1]
```

**Exercice 1.5.** Comprendre et commenter la fonction suivante :

```
def realis2(n):
    U=scipy.stats.uniform.rvs(size=1)
    return(int(numpy.ceil(n*U)))
```

## 2 Utilisation des générateurs de Python

On utilisera principalement les modules Python `numpy.random` et `scipy.stats`. Le premier permet de simuler la plupart des lois connues. En revanche il ne fait que de la simulation. Le second contient, outre la simulation de la plupart des lois connues, les densités, fonctions de répartition, quantiles, etc... des lois classiques.

Beaucoup de lois sont déjà implémentées dans ces deux modules. Pour accéder à la liste, vous pourrez consulter :

`https://docs.scipy.org/doc/scipy/reference/stats.html`
`https://docs.scipy.org/doc/numpy-1.14.0/reference/routines.random.html`

Pour chaque loi simulée précédemment, comparer avec les fonctions de ces modules. Attention aux paramètres, Python n'ayant pas forcément les mêmes notations que vous.

### Loi normale ou gaussienne

La loi normale $\mathcal{N}(\mu, \sigma^2)$, de paramètres $\mu \in \mathbb{R}$ et $\sigma^2 > 0$, ne se simule pas par inversion. On pourra utiliser `scipy.stats.norm.rvs`.

**Exercice 2.1.** Prouver que si $X$ suit une loi $\mathcal{N}(\mu, \sigma^2)$, alors $Y = (X - \mu)/\sigma$ suit une loi $\mathcal{N}(0, 1)$, et réciproquement. Le vérifier également numériquement.