ENSEIRB-MATMECA 1A **MA105**
Année 2023-2024 **TP**

Contenu très inspiré d'une feuille de TP python d'Igor Kortchemski.

1 Simulation de lois

1.1 Loi uniforme

Comment Python simule-t-il une suite de variables aléatoires indépendantes et uniformes sur $[0,1]$? En quelques mots, cela revient à générer des entiers aléatoires indépendants uniformes dans $E = \{1, 2, \dots, M\}$, puis à les diviser par M. Pour cela, Python génère une suite de nombres pseudo-aléatoires en partant dun état $x_0 \in E$ (la graine, ou « seed » en anglais) et en appliquant successivement une même fonction $f : E \to E$ (par exemple $f(x) = ax + c$ modulo M avec a et c bien choisis). Cette transformation doit bien sûr vérifier plusieurs propriétés : on veut que la suite générée « ressemble » à une suite de variables aléatoires uniformes et indépendantes sur E. Que veut dire « ressemble » ? La suite doit réussir à passer toute une batterie de tests statistiques dindépendances et dadéquation de loi. En particulier, la période, cest-à-dire le plus petit entier k tel que $f^{(k)}(x) = x$ (avec $f^{(k)}$ litérée k-ième), doit être (très (très)) grande.

Python utilise lalgorithme appelé « Mersenne Twister » (développé par Makoto Matsumoto et Takuji Nishimura en 1997) possédant une période de $2^{19937}-1$ (qui est un nombre premier de Mersenne).

Ainsi : existe-t-il des vrais générateurs de nombre aléatoire ?
NON : Pas vraiment, tous les générateurs sont déterministes !!
MAIS : ils sont construits de telle sorte à passer les tests statistiques.

Exercice 1 Exécuter plusieurs fois le code python suivant

```
import numpy.random as npr

print(npr.rand())
npr.seed(seed=1)
print(npr.rand())
print(npr.rand())
npr.seed(seed=1)
print(npr.rand())
print(npr.rand())
```

On suppose ainsi quon a à notre disposition une « boîte noire » qui permet de simuler une suite de variables aléatoires indépendantes et uniformes sur $[0,1]$. Pour générer dautres aléas que ceux de loi uniforme, on fait subir des transformations astucieuses (algorithme de simulation).

Exemple (simulation dune variable uniforme sur un segment quelconque). Si $a < b$ et U est une variable aléatoire uniforme sur $[0,1]$, on peut démontrer que $a + (b-a)U$ suit une loi uniforme sur $[a,b]$. Ainsi, pour simuler une variable aléatoire uniforme sur $[a,b]$, on simule une variable aléatoire U uniforme sur $[0,1]$ et on renvoie $a + (b-a)U$.

1.2 Simulation par inversion de la fonction de répartition

On a vu en TD que si U est uniforme sur $[0,1]$ et $\lambda > 0$, alors $V = -\frac{1}{\lambda}\ln(U)$ est une loi exponentielle de paramètre λ. Plus généralement, on a le résultat suivant (démontré aussi en TD).

Théorème 1 *Soit X une variable aléatoire réelle. On suppose que sa fonction de répartition F est strictement croissante (F est donc bijective de $\mathbf{R}$ sur $[0,1]$ et on peut noter F^{-1} son inverse). Soit U une variable aléatoire uniforme sur $[0,1]$. Alors $F^{-1}(U)$ a même loi que X.*

Si F n'est pas strictement croissante (et donc pas injective), le théorème précédent reste vrai à condition de définir $F^{-1}(u)$ comme $F^{-1}(u) = \inf\{x \in \mathbf{R} : F(x) \geq u\}$ (F^{-1} est appelé linverse continue à droite de F).

Exercice 2 Simuler une variable aléatoire de Cauchy de paramètre 1 dont une densité est $\frac{2}{\pi}\frac{1}{1+x^2}$. On pourra compléter le code suivant :

```
from math import *
import numpy.random as npr

def Cauchy():
  U=BLA
  return tan(BLA)

print(Cauchy())
```

1.3 Loi géométrique

On va comparer plusieurs manières de simuler une variable géométrique de paramètre $p \in]0,1[$ à partir dune variable aléatoire uniforme sur $[0,1]$ en se fondant sur les résultats théoriques suivants :

1. De manière générale, pour simuler une variable aléatoire X à valeurs entières telle que $\mathbf{P}(X = i) = p_i$ pour tout $i \geq 0$, on tire une variable uniforme U sur $[0,1]$ et on renvoie l'entier k tel que $p_0 + \ldots + p_{k-1} < U < p_0 + \ldots + p_k$.
2. On tire des variables aléatoires de Bernoulli de paramètre p (on renvoie 1 si une variable aléatoire uniforme U sur $[0,1]$ est plus petite que p, 0 sinon) et on s'arrête à la première fois quon tombe sur 1.
3. On pose $\lambda = -\frac{1}{\ln(1-p)}$ et on renvoie $\lceil \lambda X \rceil$ où X est une variable aléatoire exponentielle de paramètre 1.
4. Faire appel à une fonction intégrée de Python.

Exercice 3 Comparer l'efficacité de ces trois méthodes pour simuler N variables géométrique de paramètre p en complétant le code suivant. Commenter l'influence des paramètres.

```
from __future__ import division
from math import *
import numpy as np
import numpy.random as npr
from time import time

p=0.1 #parametre de la geometrique
N=10000 #nombre de fois qu'on simule la geometrique

def methode1():
  k=1
  tmp=p
  U=npr.rand()
  while U>tmp:
    tmp=tmp+BLA
    k=k+1
  return BLA

def methode2():
  tmp=0
  k=0
  while tmp==0:
    k=k+1
    if BLA:
      tmp=1
  return k
```

```python
def methode3():
  X=-log(npr.rand())
  return int(ceil(BLA))

def methode4():
  return npr.BLA(p)

t1 = time()
[methode1() for i in range(N)]
t2 = time()
temps1 = t2 - t1

print("La methode 1 a pris ", temps1, " secondes")

t1 = time()
[methode2() for i in range(N)]
t2 = time()
temps1 = t2 - t1
print("La methode 2 a pris ", temps1, " secondes")

t1 = time()
[methode3() for i in range(N)]
t2 = time()
temps1 = t2 - t1
print("La methode 3 a pris ", temps1, " secondes")

t1 = time()
[methode4() for i in range(N)]
t2 = time()
temps1 = t2 - t1
print("La methode 4 a pris ", temps1, " secondes")
```

2 Convergence de variables aléatoires

2.1 Approximation dune loi de Poisson par une loi binomiale

Vous avez sûrement déjà démontré le résultat suivant (si ce n'est pas le cas, faites le !) :

Théorème 2 *Soit* $\lambda > 0$. *Soit* X_n*une variable aléatoire binomiale de paramètre* $(n, \lambda/n)$. *Alors* X_n *converge en loi vers une loi de Poisson de paramètre* λ *lorsque* $n \to \infty$.

Exercice 4 Illustrer ce théorème en complétant le code suivant :

```python
from __future__ import division
import numpy as np
import numpy.random as npr
import scipy.stats as sps
import matplotlib.pyplot as plt

param=3 #parametre
n=100
N=5000 #nombre de tirages effectue pour tracer l'histogramme en batons de la loi de
                                          X_n
X=npr.binomial(BLA)

BLA
BLA
BLA
BLA
BLA
BLA
```

2.2 Approximation dune loi exponentielle par une loi géométrique

Nous avons démontré le résultat suivant en TD.

Théorème 3 *Soit $\lambda > 0$. Si X_n est une variable aléatoire de loi géométrique de paramètre λ/n, alors X_n converge en loi vers une variable exponentielle de paramètre λ lorsque $n \to \infty$.*

Exercice 5 Illustrer ce théorème en complétant le code suivant :

```
from __future__ import division
import numpy as np
import numpy.random as npr
import scipy.stats as sps
import matplotlib.pyplot as plt

param=2 #parametre
n=1000
N=5000 #nombre de tirages effectue pour tracer une densite de X_n

X=npr.geometric(BLA)

plt.hist(BLA, normed=True, label="Densite empirique", bins=int(sqrt(N)))
x = np.linspace(BLA, BLA, 100)
f_x = param*np.exp(-BLA)
plt.plot(x, f_x, "r", label="Densite theorique")
plt.legend()
```

2.3 Encore la loi exponentielle...

En utilisant un théorème du cours (lequel à votre avis ?) on peut démontrer que si $(X_n, n \geq 2)$ est une suite de variables aléatoires exponentielles de paramètre 1, définies sur le même espace de probabilité, alors presque sûrement la plus grande valeur d'adhérence de la suite $(X_n/\ln(n), n \geq 2)$ vaut 1.

Exercice 6 Illustrer ce résultat en complétant le code suivant

```
from __future__ import division
import numpy as np
import numpy.random as npr
import scipy.stats as sps
import matplotlib.pyplot as plt

n=100000

X=np.arange(2,n+2)
Y=BLA

plt.plot(X, Y)
plt.plot(X, BLA, "r", label="Droite d'equation y=1")

plt.legend()
```

Exercice 7 Etudier numériquement la convergence en loi de la suite $X_n/\ln(n)$.

3 Approximation de π par la méthode de Monte-Carlo

Considérons un couple (X, Y) de variables aléatoires indépendantes telles que chacune soit uniforme sur $[0, 1]$. Soit $((X_i, Y_i)), i \geq 1$ une suite de vecteurs aléatoires indépendants et de même loi que (X, Y).

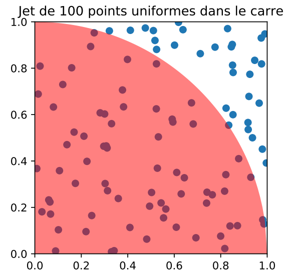

FIGURE 1 – Dans cet exemple, $S_{100} = \frac{4}{100} \times 69$ (il y a 69 points dans le quart de disque rouge).

Exercice 8 Montrer que $\mathbf{P}(X^2 + Y^2 \leq 1) = \frac{\pi}{4}$.

Ceci est intuitif au moins : (X, Y) représente un point « uniforme » dans le carré $[0,1]^2$, et le probabilité qu'il appartienne à un quart de disque centré en 0 est l'aire de ce quart de disque (cest-à-dire $\pi/4$) divisé par l'aire totale du carré, qui vaut 1.
Posons $Z_i = 1$ si $X_i^2 + Y_i^2 \leq 1$ et $Z_i = 0$ sinon. Ainsi, $(Z_i)_{i \geq 1}$ sont des variables aléatoires indépendantes de Bernoulli de paramètre $\pi/4$. En posant

$$S_n = \frac{4}{n} \sum_{i=1}^{n} Z_i,$$

on en déduit daprès la loi forte des grands nombres que S_n converge presque sûrement vers π.

On veut maintenant un intervalle de confiance sur la valeur de π. Pour cela, on remarque que

$$\mathbf{E}(S_n) = \pi, \quad \mathbf{V}(S_n) = \frac{16}{n} \frac{\pi}{4} \left(1 - \frac{\pi}{4}\right) \leq \frac{4}{n}.$$

Daprès l'inégalité de Bienaymé-Tchebychev, on a donc

$$\mathbf{P}(|S_n - \pi| \geq \alpha) \leq \frac{4}{n\alpha^2}.$$

Exercice 9 Simuler un intervalle de confiance de π damplitude au plus 10^{-2} à 99% en complétant le code suivant :

```
from __future__ import division
import numpy as np
import numpy.random as npr
import scipy.stats as sps
import matplotlib.pyplot as plt

n=BLA

X=npr.rand(n)
Y=npr.rand(n)

Sn=4/n*np.sum(BLA) #sum compte le nombre d'elements non nuls (ou vrais) dans un
tableau
print("Intervalle de confiance pour Pi au niveau 0.99 : ["+str(BLA)+","+str(BLA)+"]")
#On utilise str pour convertir un entier en chaine de caracteres et on utilise le +
                                              pour concatener des chaines de caracteres
print("En vrai, Pi vaut "+str(np.pi))
```

4 Réglement de comptes à OK Corall

Soit $n \geq 1$ un entier. Deux groupes de bandits, chacun constitué de n personnes, se retrouvent à OK Corral pour un règlement de comptes. Tant qu'il reste au moins un bandit vivant dans chaque groupe, à chaque seconde un bandit (choisi uniformément au hasard parmi ceux encore vivants, indépendamment de tout ce qui sest passé avant) abat un bandit de lautre groupe. On note $V(n)$ le nombre (aléatoire) de survivants à lissue de la fusillade. Une étude théorique permet de montrer que $V(n)/n^{3/4}$ converge en loi lorsque $n \to \infty$ vers une variable aléatoire réelle à densité dont une densité est

$$\sqrt{\frac{3}{\pi}} x e^{-\frac{3x^4}{16}} 1_{[0,\infty(}(x).$$

Exercice 10 Illustrer cette convergence en loi par des simulations.

5 Echantillonage préférentiel

On reprend un exercice de TD.

Exercice 11 On souhaite calculer numériquement l'intégrale suivante $I = \int_0^1 e^{-x^4} \mathrm{d}x$.

1. Soit X suivant une loi uniforme sur $[0, 1]$. Exprimer I en fonction de X et appliquer la méthode de Monte-Carlo afin de donner une approximation de I.
2. Soit B une variable aléatoire suivant une loi bêta de paramètre $(1 - \varepsilon, 1 + \varepsilon)$ (avec $\varepsilon \geq 0$ un paramètre réel), c'est-à-dire dont la densité est donnée, pour $x \in \mathbf{R}$ par

$$b(x) = \begin{cases} \frac{1}{\beta(1-\varepsilon,1+\varepsilon)} x^{\varepsilon}(1-x)^{-\varepsilon} & \text{si } x \in [0,1], \\ 0 & \text{sinon,} \end{cases}$$

   où $\beta(1 - \varepsilon, 1 + \varepsilon)$ est une constante de renormalisation.
3. Écrire I en fonction de B.
4. Faire afficher la courbe qui a ε associe une approximation de l'écart-type de $\frac{e^{-B^4}}{b(B)}$. Qu'en pensez-vous ?