⋆ Simulations de Variables Aléatoires ⋆
Méthode d'inversion

Compétences attendues

- ✓ Savoir simuler une loi discrète ou continue usuelle avec les outils fournis par la bibliothèque Numpy, ou uniquement à partir d'une variable de loi uniforme et la méthode d'inversion
- ✓ Vérifier graphiquement la pertinence d'une simulation d'une loi (histogrammes, boxplot)

Liste des commandes Python exigibles aux concours

- Dans la librairie `numpy.random` :
  - Lois discrètes : `binomial`, `randint`, `geometric`, `poisson` ;
  - Lois continues : `random`, `uniform`, `exponential`, `normal`, `gamma`
- Dans la librairie `matplotlib.pyplot` : `hist`, `show`

Dans toute cette feuille technique, on supposera l'import de la bibliothèque `numpy.random` avec le préfixe `rd`.

Introduction et motivation

Malgré l'existence de commandes sous `numpy.random` qui permettent la simulation immédiate de [presque] toutes les lois, nous avons vu au cours/TP précédent que de nombreuses lois étaient simulables à partir d'une simple variable de loi $\mathcal{U}([0;1])$. C'est d'ailleurs d'elle seule que l'on vous demandera bien souvent de partir pour générer d'autres lois plus complexes.
Nous allons voir ici, que c'est même le cas pour n'importe quelle loi (discrète ou continue) dont on connaîtrait la fonction de **répartition inverse**. Ceci a notamment permis de baptiser le procédé en **méthode d'inversion** (dû à l'utilisation d'une fonction de répartition inverse).

1 Méthode d'inversion discrète

1.1 Principe

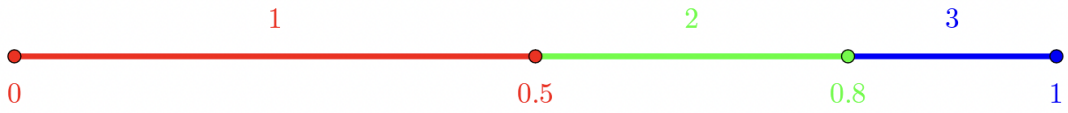
La **méthode d'inversion** est une méthode générale pour simuler la loi d'une variable X à partir d'une variable aléatoire $U \sim \mathcal{U}([0;1])$. Elle consiste à "inverser" la fonction de répartition de X. L'exemple suivant constitue un point de méthode qui s'applique parfaitement à n'importe quel cas de génération de lois discrètes.

Exemple 1.1. On souhaite simuler une variable aléatoire X ayant pour support (ensemble image) $X(\Omega) = \{1,2,3\}$ et pour probabilités ponctuelles

$$\mathbb{P}(X=1) = 0.5, \quad \mathbb{P}(X=2) = 0.3, \quad \mathbb{P}(X=3) = 0.2.$$

Rappelons que pour générer une v.a. de Bernoulli $\mathcal{B}(p)$ (à 2 issues possibles) on découpait le segment $[0;1]$ (à voir comme les probabilités) en 2 parties : à gauche de p (de "volume" p) et à droite de p (de "volume" $1-p$).

Pour notre cas, on va procéder d'une manière analogue en découpant cette fois-ci le segment $[0;1]$ en 3 intervalles (correspondant aux 3 issues possibles), de longueurs 0.5, 0.3 et 0.2 correspondant aux probabilités de chaque issue.



Pour simuler la variable X, on tire alors au hasard un nombre $u \in [0;1]$ (avec `rd.random()` ou `rd.uniform(0,1)`) et on retourne :

- 1 si $u \in [0;0.5]$, ce qui se produit avec probabilité $0.5 - 0 = 0.5 = \mathbb{P}(X=1)$;
- 2 si $u \in [0.5;0.8]$, ce qui se produit avec probabilité $0.8 - 0.5 = 0.3 = \mathbb{P}(X=2)$;
- 3 si $u \in [0.8;1]$, ce qui se produit avec probabilité $1 - 0.8 = 0.2 = \mathbb{P}(X=3)$.

On peut ainsi simuler la variable X à l'aide de la fonction suivante :

```
def simu_X(): #pas besoin d'argument, les probas sont fixes
    u = rd.random() # ou rd.uniform(0,1,1)
    if u <= 0.5 :
        return 1
    elif u <= 0.8 :
        return 2
    else :
        return 3
```

Exercice 1 [Loi Uniforme Discrète] On souhaite simuler une loi uniforme $\mathcal{U}(\llbracket 1;6 \rrbracket)$ (un dé équilibré).

1. Déterminer le découpage correspondant de l'intervalle $[0;1]$ pour cette loi.
2. Écrire une fonction `unif()` simulant la loi $\mathcal{U}(\llbracket 1;6 \rrbracket)$ à l'aide de la méthode d'inversion.

1.2 Cas général de la méthode d'inversion

Soit à présent X une variable discrète qui prend les valeurs $x_1 < x_2 < \cdots < x_n < \cdots \in X(\Omega)$ qui est un ensemble dénombrable (fini ou infini).
Rappelons que sa fonction de répartition est en escalier (croissante et constante par morceaux), et est donnée par :

$$\forall x \in \mathbb{R}, \quad F(x) = \mathbb{P}(X \leq x) = \begin{cases} 0 & \text{si } x < x_1, \\ \sum_{i=1}^{n} \mathbb{P}(X = x_i) & \text{si } x_n \leq x < x_{n+1}. \end{cases}$$

La méthode d'inversion présentée sur les exemples précédents se généralise de la manière suivante :

Méthode 1.0 [Simulation d'une variable aléatoire discrète par la méthode d'inversion]

Pour simuler une variable aléatoire discrète X définie comme précédemment, on procédera comme suit :

1. on tire $u =$ `rd.random()` aléatoirement dans $[0;1]$;
2. on détermine l'intervalle $I_k =]F(x_{k-1}); F(x_k)]$ tel que $u \in I_k$ (si $k = 1$, $I_1 = [0; F(x_1)]$) ;
3. on retourne x_k.

Remarque 1.1. Le programme est alors sensé retourner x_k avec une probabilité $\mathbb{P}(X = x_k)$ pour tout k.
Vérifions le pour $k \geq 2$: x_k est retourné si et seulement si $u \in I_k =]F(x_{k-1}); F(x_k)]$ où u est une réalisation d'une variable aléatoire uniforme $U \sim \mathcal{U}([0;1])$. Or, ceci se réalise avec probabilité (les inégalités strictes et larges étant équivalentes comme U est une variable aléatoire à densité) :

$$\mathbb{P}(U \in I_k) = \mathbb{P}(F(x_{k-1}) \leq U \leq F(x_k)) = \mathbb{P}(U \leq F(x_k)) - \mathbb{P}(U \leq F(x_{k-1})) = F_U(F(x_k)) - F_U(F(x_{k-1})).$$

Comme de plus, la fonction de répartition de U est telle que $F_U(x) = x$ si $x \in [0;1]$, et que $F(x_k), F(x_{k-1}) \in [0;1]$, on obtient :

$$\mathbb{P}(F(x_{k-1}) \leq U \leq F(x_k)) = F(x_k) - F(x_{k-1}) = \sum_{i=1}^{k} \mathbb{P}(X = x_i) - \sum_{i=1}^{k-1} \mathbb{P}(X = x_i) = \mathbb{P}(X = x_k).$$

Le même argument vaut aussi pour $k = 1$. Ainsi, la probabilité qu'un tel programme retour x_k est bien de $\mathbb{P}(X = x_k)$.

2 Exercices

Exercice 2 [Simulation de la loi de Poisson par la méthode d'inversion]

On souhaite simuler une loi de Poisson $\mathcal{P}(\lambda)$ (pour $\lambda > 0$) par la méthode d'inversion.

1. On tire au hasard un nombre $u \in [0;1]$. Dans quel intervalle I_k le nombre u doit se trouver pour qu'on retourne k ?
2. On se donne le code suivant :

```
def poisson(lamb):
    u = rd.random()
    k = 0
    s = np.exp(-lamb)
    t = s
    while t<u :
        k = k+1
        s = s*(lamb/k)
        t = t+s
    return k
```

   Expliquer le fonctionnement de la fonction `poisson(lamb)`.
3. Écrire une fonction `Poisson(lamb,N)` renvoyant un vecteur Numpy contenant N réalisations indépendantes de la loi $\mathcal{P}(\lambda)$.
4. À l'aide d'un diagramme en bâtons, juger de la pertinence de cette simulation pour $\lambda = 5$ en la comparant avec la distribution théorique. On admet pour cela que les cas où la variable prend une valeur supérieure à 10 sont négligeables.

Exercice 3 [Simulation de la loi géométrique par méthode d'inversion]

Écrire une fonction `geom` qui prend en paramètre un nombre $p \in]0;1[$, puis qui simule une loi géométrique de paramètre p en utilisant la méthode d'inversion.

Exercice 4

On dispose d'une urne contenant 7 boules indiscernables au toucher. 3 sont blanches et 4 sont noires. On effectue dans cette urne 2 tirages successifs d'une boule, sans remise. On appelle X (respectivement Y) la variable aléatoire prenant la valeur 1 si la première boule tirée (respectivement la deuxième boule tirée) est blanche et 0 sinon.
Compléter le programme suivant afin qu'il simule l'expérience et affiche la valeur du couple (X, Y).

```
if rd.random() < 3/7 :
    X = ...
else :
    X = ...
if rd.random() < ... :
    Y = ...
else :
    Y = ...
print(X,Y)
```

Exercice 5

À un guichet, des clients peuvent venir expédier ou retirer un colis. Au cours d'une journée, le nombre de clients N qui s'y présentent suit la loi de Poisson de paramètre $\lambda \in \mathbb{R}_+^*$. Chaque client a une probabilité $p \in]0;1[$ de venir pour expédier un colis et $1-p$ pour en retirer un. On note C le nombre de colis expédiés dans la journée.

1. Pour tout $k \in \mathbb{N}$, déterminer l'espérance de C conditionnellement à l'évènement $\{N = k\}$.
2. En déduire l'espérance de C.
3. On suppose dans cette question que $p = 0.3$ et $\lambda = 2.5$.
   (a) Écrire un programme renvoyant un vecteur C contenant $n = 10000$ réalisations de la variable aléatoire C.
   (b) Comparer alors la moyenne empirique sur cet échantillon de réalisations de C avec son espérance théorique calculée en 2.
   (c) On souhaite représenter le diagramme en bâtons des fréquences de l'échantillon.
      i. Exécuter la commande `np.mean(C<=4)`. Que dire des valeurs prises par C ?
      ii. Représenter le diagramme en bâtons des fréquences de l'échantillon.

Exercice 6 [Loi arbitraire]

1. Vérifier que si $(p_0, p_1, \ldots, p_N)$ est une probabilité et U est une variable aléatoire uniforme sur $[0;1]$, alors les évènements $A_i = \{p_0 + \ldots p_{i-1} < U \leq p_0 + \cdots + p_i\}$ forment une partition de l'univers Ω et vérifient $\mathbb{P}(A_i) = p_i$ (avec $0 \leq i \leq N$).
2. Utiliser la question précédente pour simuler une variable aléatoire X à valeurs dans $\{0,1,2,3\}$ telle que
$$\mathbb{P}(X=0) = \frac{1}{2},\ \mathbb{P}(X=1) = \frac{1}{4},\ \mathbb{P}(X=2) = \frac{1}{6},\ \mathbb{P}(X=3) = \frac{1}{12}.$$
3. Afficher un histogramme de 1000 simulations de la variable X.
4. Écrire une fonction prenant en argument un tableau Numpy de réels $(p_i)_{1 \leq i \leq N-1}$ compris entre 0 et 1 et dont la somme est inférieure à 1. Cette fonction devra renvoyer une simulation d'une variable aléatoire à valeurs dans $\{0, \ldots, N\}$ telle que $\mathbb{P}(X = k) = p_k$. En particulier, on posera $p_N = 1 - \sum_{i=0}^{N-1} p_k$.