Exercices - Fiche 2

* * *

Exercice 1. *HEC 2017.* Pour tout entier $n \in \mathbb{N}^*$, on considère une suite infinie d'épreuves de Bernoulli indépendantes telles que pour chaque épreuve la probabilité de succès est égale à $\frac{1}{n}$. On suppose que cette expérience est modélisée par un espace probabilisé $(\Omega, \mathcal{A}, \mathbb{P})$.

On note X_n la variable aléatoire égale au rang du premier succès et on pose $U_n = \frac{X_n}{n}$.

1. Écrire un programme PYTHON qui simule la variable aléatoire U_n .
2. En ayant recours à un histogramme, comparer la loi de U_n avec la loi exponentielle $\mathcal{E}(1)$ pour $n = 10, n = 50, n = 100$. Que semble indiquer ce qu'on observe ?
3. Vérifier cette conjecture.

Exercice 2. *HEC 2006.*

1. Montrer que la fonction $g_n : x \mapsto \left(\sum_{k=0}^n \frac{x^k}{k!} \right) e^{-x}$ est dérivable sur $\mathbb{R}$ et calculer sa dérivée.
2. Montrer que pour tout $n \in \mathbb{N}$, l'équation

$$\frac{e^x}{2} = \sum_{k=0}^n \frac{x^k}{k!}$$

admet une unique solution dans $\mathbb{R}_+$. Dans la suite, on notera a_n cette solution.

3. Écrire une fonction PYTHON $\mathbf{G}(n, x)$ permettant de calculer $g_n(x)$, puis une autre permettant d'obtenir une valeur approchée de a_n à 10^{-2} près.

Exercice 3. *HEC.* Au moyen de trois dés équilibrés à 6 faces, proposer une méthode pour choisir de manière aléatoire une personne dans un groupe de 24 personnes, en assurant l'équiprobabilité.

Écrire un programme PYTHON simulant ce tirage au sort à l'aide de cette méthode.

Exercice 4. *HEC 2009.* On considère la suite réelle $(u_n)_{n \in \mathbb{N}}$ définie par

$$u_0 = 3, \quad u_1 = \frac{29}{9}, \quad \text{et} \quad \forall n \in \mathbb{N}, \quad u_{n+2} = 9 - \frac{26}{u_{n+1}} + \frac{24}{u_n u_{n+1}}.$$

1. Écrire une fonction PYTHON permettant de calculer la valeur du terme u_n pour tout $n \in \mathbb{N}$.
2. On pose $a_0 = 3$, et pour tout $n \in \mathbb{N}$, $a_{n+1} = u_n a_n$. Montrer que (a_n) vérifie la relation de récurrence $a_{n+3} = 9a_{n+2} - 26a_{n+1} + 24a_n$.
3. Montrer que pour tout $n \in \mathbb{N}$, $a_n = 2^n + 3^n + 4^n$.
4. Expliciter u_n en fonction de n , puis $\lim_{n \rightarrow \infty} u_n$.

Exercice 5. *ESCP 2019.* On définit la suite (C_n) par

$$C_0 = 1, \quad \text{et pour tout } n \in \mathbb{N} \quad C_{n+1} = \sum_{k=0}^n C_k C_{n-k}.$$

Écrire une fonction `C(n)` en PYTHON qui renvoie la valeur de $C(n)$ en fonction de $n \in \mathbb{N}$.

Exercice 6. On considère la fonction PYTHON suivante :

```
import numpy as np
def X(m):
    N=np.floor(1+m*rd.rand())
    y=0
    for i in range(int(N)):
        y=y+np.floor(2*rd.rand())
    return y
```

1. On fixe $m \in \mathbb{N}^*$. Quelle est la loi suivie par la variable locale N ? Préciser $X(\Omega)$. Pour tout couple d'entiers (i, k) , calculer la probabilité $\mathbb{P}_{N=k}(X = i)$. En déduire la loi de X (on ne cherchera pas à simplifier la somme obtenue).
 2. Calculer $\mathbb{E}(X)$.
-

Exercice 7. *ESCP, 2021.* Soient a et b deux réels positifs. On définit les suites (u_n) et (v_n) par récurrence de la manière suivante :

$$\begin{cases} u_0 = a, v_0 = b, \\ u_{n+1} = \frac{u_n + v_n}{2}, \forall n \in \mathbb{N} \\ v_{n+1} = \sqrt{u_n v_n}, \forall n \in \mathbb{N} \end{cases}$$

1. Montrer que les suites (u_n) et (v_n) sont bien définies.
 2. Montrer que $(u_n)_{n \in \mathbb{N}^*}$ est décroissante, $(v_n)_{n \in \mathbb{N}^*}$ est croissante, et $v_n \leq u_n$ pour tout $n \in \mathbb{N}^*$.
 3. En déduire que (u_n) et (v_n) convergent vers une même limite, que l'on notera $M(a, b)$.
 4. Écrire une fonction PYTHON qui prend en entrée deux réels positifs a et b et un réel strictement positif `eps`, et qui renvoie une valeur approchée `M` de $M(a, b)$ à `eps` près. Autrement dit, `M` doit satisfaire $|M(a, b) - \mathbf{M}| \leq \text{eps}$.
-

Exercice 8. *ESCP, 2018.* Une pièce truquée donne Pile avec la probabilité $p \in]0, 1[$ et Face avec la probabilité $q = 1 - p$. On propose l'expérience suivante pour "rééquilibrer" la pièce. L'expérience est constituée d'une suite de parties consécutives. Chaque partie consiste à lancer la pièce deux fois de suite.

- Si à l'issue d'une partie on obtient deux fois le même résultat, on refait une partie,
- Si à l'issue d'une partie on obtient deux résultats différents, alors on arrête l'expérience et on rend le résultat du dernier lancer.

L'expérience est modélisée à l'aide d'un espace probabilisé $(\Omega, \mathcal{A}, \mathbb{P})$. Soient T la variable aléatoire égale au nombre de lancers nécessaires pour que l'expérience se termine, et R la variable aléatoire égale au résultat de l'expérience.

1. Écrire une fonction PYTHON $T(p)$ qui prend p en entrée, et renvoie une simulation de la variable aléatoire T .
 2. Écrire une fonction PYTHON $R(p)$ qui prend p en entrée, et renvoie une simulation de la variable aléatoire R .
 3. En utilisant la fonction R , proposer une méthode pour vérifier que l'expérience a bien pour effet de "rééquilibrer la pièce".
-

Exercice 9. *ESCP, 2021.* On considère une urne qui contient trois boules : une blanche, une noire et une rouge. On effectue des tirages au hasard d'une boule avec remise dans l'urne. On note X le numéro du tirage où pour la première fois on a obtenu la boule blanche, et Y le numéro du tirage où on a obtenu la noire pour la première fois.

Écrire une fonction PYTHON qui renvoie une simulation du couple (X, Y) .

Exercice 10. *ESCP, 2018.* Soit $f : \mathbb{R} \rightarrow \mathbb{R}$ définie par

$$f(x) = \begin{cases} \frac{x}{2} & \text{si } x \in [0, 1[, \\ -\frac{1}{6}(x-4) & \text{si } x \in [1, 4], \\ 0 & \text{sinon.} \end{cases}$$

1. Montrer que f est une densité de probabilité. On considère pour la suite une variable aléatoire X qui admet f pour densité. On note F sa fonction de répartition.
2. Montrer que F est bijective de $[0, 4]$ sur $[0, 1]$ et que sa réciproque est donnée par

$$G(x) = \begin{cases} \sqrt{4y} & \text{si } y \in [0, \frac{1}{4}[, \\ 4 - \sqrt{12 - 12y} & \text{si } y \in [\frac{1}{4}, 1]. \end{cases}$$

3. On considère la variable aléatoire $Y = F(X)$. Déterminer la loi de Y et en déduire le rôle des fonctions `mystere()` et `mysterebis(n)` ci-dessous.

```
def mystere():
    y=rd.rand()
    if y<1/4:
        return np.sqrt(4*y)
    else:
        return 4-np.sqrt(12-12*y)
```

```
def mysterebis(n):
    E=0;
    for i in range(n):
        x=mystere()
        E=E+x
    return E/n
```