ECG2 - 2023-2024

DS N°3
MATHÉMATIQUES
(4H)

La qualité de la rédaction, le soin porté à la copie, la lisibilité, l'orthographe, la rigueur du vocabulaire ainsi que la clarté des raisonnements sont des critères importants d'évaluation.

Quelques précisions :
. les résultats finaux doivent être clairement mis en évidence (soulignés ou encadrés),
. les questions de chaque exercice doivent être présentées dans l'ordre du sujet.

L'usage de tout matériel électronique est interdit. Aucun document n'est autorisé.

EXERCICE 1 :

Toutes les variables aléatoires rencontrées dans cet exercice sont supposées définies sur un même espace probabilisé $(\Omega, \mathscr{A}, P)$ que l'on ne cherchera pas à déterminer.

1. Soit f la fonction qui à tout réel x associe

$$f(x) = \begin{cases} x e^{-\frac{x^2}{2}} & \text{si } x \geqslant 0 \\ 0 & \text{sinon} \end{cases}$$

   (a) Vérifier que la fonction $g : x \mapsto -e^{-\frac{x^2}{2}}$ est une primitive de la fonction f sur $[0; +\infty[$.

   (b) Montrer que f peut être considérée comme densité d'une certaine variable aléatoire X.

   (c) Rappeler la valeur du moment d'ordre 2 d'une variable aléatoire suivant la loi normale centrée réduite.

   (d) En déduire, par des considérations de parité, que X a une espérance et que $E(X) = \sqrt{\dfrac{\pi}{2}}$.

2. On note F_X la fonction de répartition de X. Déterminer $F_X(x)$ selon que $x < 0$ ou $x \geqslant 0$.

3. **Simulation**

   (a) On pose $Z = X^2$ et on note F_Z sa fonction de répartition. Déterminer $F_Z(x)$ dans chacun des cas $x < 0$ et $x \geqslant 0$ et montrer que Z suit une loi exponentielle dont on précisera le paramètre.

   (b) En déduire une fonction Python d'en-tête `def simulx():` qui renvoie une simulation de X.

4. Pour tout entier naturel n non nul, on pose $Y_n = \dfrac{X}{\sqrt{n}}$ et on note G_n la fonction de répartition de Y_n.

   Montrer que l'on a :

$$G_n(x) = \begin{cases} 1 - e^{-nx^2/2} & \text{si } x \geqslant 0 \\ 0 & \text{si } x < 0 \end{cases}$$

5. On considère une suite $(X_n)_{n \in \mathbb{N}^*}$ de variables aléatoires mutuellement indépendantes, et suivant toutes la même loi que X. Pour tout entier naturel n non nul, on pose $M_n = \min(X_1, \ldots, X_n)$.

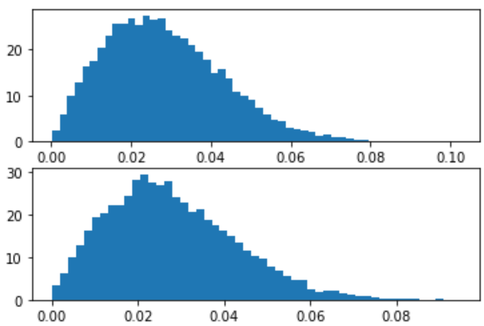
   (a) Exprimer, pour tout réel x la probabilité $P(M_n > x)$ à l'aide de la fonction F_X, puis en déduire que M_n suit la même loi que la variable Y_n présentée à la question 4).

   (b) Compléter la fonction Python suivante afin qu'elle renvoie une simulation de M_n à l'appel de `simulM(n)`.

```
def simulM(n):
    X = [ .................. for k in range(n) ]
    M = .................
    return M
```

   (c) Donner une interprétation du script suivant ainsi que les graphiques ci-après, dans le contexte de l'exercice :

```
plt.clf()
n=1000
nbsim=10000
fig, axs = plt.subplots(2)
x=[simuly(n) for _ in range(1,nbsim+1)]
axs[0].hist(x,bins='auto',density='True')
y=[simulM(n) for _ in range(1,nbsim+1)]
axs[1].hist(y,bins='auto',density='True')
plt.show()
```



EXERCICE 2 :

Dans tout l'exercice, la lettre n désigne un entier naturel.

On pose $u_n = \displaystyle\int_0^1 \frac{x^n}{4-x^2}\,dx$ et on a en particulier $u_0 = \displaystyle\int_0^1 \frac{1}{4-x^2}\,dx$.

1. (a) Déterminer les réels a et b tels que :

$$\forall x \in [0;1], \quad \frac{1}{4-x^2} = \frac{a}{2-x} + \frac{b}{2+x}.$$

   (b) En déduire que $u_0 = \dfrac{1}{4}\ln(3)$.

2. Calculer u_1.

3. (a) Montrer à l'aide d'un calcul d'intégrale que : $\forall n \in \mathbb{N}, \quad 4u_n - u_{n+2} = \dfrac{1}{n+1}$.

   (b) Compléter la fonction Python ci-dessous afin qu'elle renvoie la valeur de u_n à l'appel de `suite(n)`.

```python
def suite(n):
    if (-1)**n == 1 :
        u = np.log(3) / 4
        for k in range(2,n + 1,2) :
            u = 4 * u - ...
    else :
        u = np.log(2 / np.sqrt(3))
        for k in range(3,n + 1,2) :
            u = 4 * u - ...
    return u
```

4. (a) Utiliser la définition de la suite $(u_n)_{n\in\mathbb{N}}$ pour établir l'encadrement suivant :

$$\forall n \in \mathbb{N}, \quad \frac{1}{4(n+1)} \leqslant u_n \leqslant \frac{1}{3(n+1)}$$

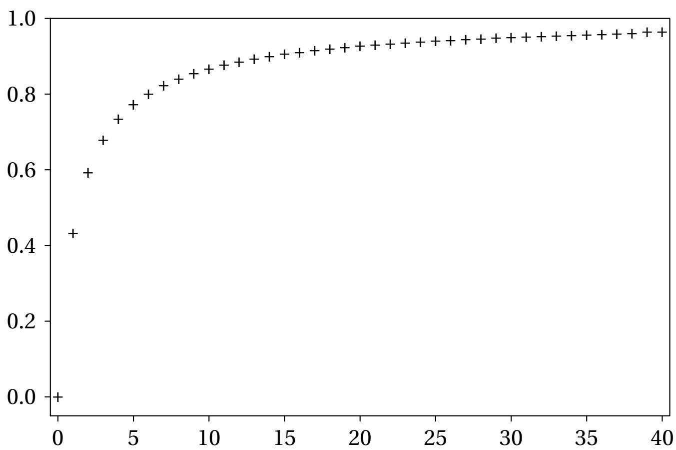
   (b) En déduire la convergence de la suite $(u_n)_{n\in\mathbb{N}}$ ainsi que la valeur de $\lim\limits_{n\longrightarrow+\infty} u_n$.

   (c) La série de terme général u_n est-elle convergente ou divergente ? Pour quelle raison ?

5. (a) On considère le script suivant qui utilise la fonction déclarée plus haut :

```python
x = np.arange(0,41)
u = [] # liste vide
for n in range(41):
    u.append(3 * n * suite(n))
plt.plot(x,u,'+')
plt.show()
```

   Ce script renvoie le graphique suivant :



Laquelle des quatre conjectures suivantes peut-on émettre quant au comportement de la suite $(u_n)_{n\in\mathbb{N}}$ au voisinage de $+\infty$?

❶ $u_n \underset{+\infty}{\sim} 3n$ ❷ $\lim\limits_{n\longrightarrow+\infty} u_n = 1.$ ❸ $u_n \underset{+\infty}{\sim} \dfrac{1}{3n}.$ ❹ $u_n \underset{+\infty}{\sim} \dfrac{1}{n}.$

(b) Établir, grâce à une intégration par parties, l'égalité suivante :

$$\forall n\in\mathbb{N}, u_n = \frac{1}{3(n+1)} - \frac{2}{n+1}\int_0^1 \frac{x^{n+2}}{\left(4-x^2\right)^2}\,\mathrm{d}x$$

(c) Montrer par encadrement que :

$$\lim_{n\longrightarrow+\infty}\int_0^1 \frac{x^{n+2}}{\left(4-x^2\right)^2}dx = 0$$

(d) Vérifier la conjecture établie à la question 5a).

PROBLÈME

Partie I : réduction d'une matrice

On considère la matrice

$$M = \begin{pmatrix} 0 & 1 & 0 \\ \frac{1}{4} & \frac{1}{2} & \frac{1}{4} \\ 0 & 1 & 0 \end{pmatrix}.$$

1. Calculer M^2 et M^3, vérifier que $2M^3 = M^2 + M$, puis en déduire un polynôme annulateur de M.

2. On donne l'instruction Python suivante :

```
IPython 7.22.0 -- An enhanced Interactive Python.

In [1]: import numpy as np

In [2]: M=np.array([[0,1,0],[1/4,1/2,1/4],[0,1,0]])

In [3]: np.linalg.matrix_rank(M-np.eye(3,3))
Out[3]: 2
```

En déduire une racine du polynôme annulateur trouvé à la question précédente puis factoriser ce polynôme.

3. En déduire $Sp(M) = \left\{-\frac{1}{2};0;1\right\}$ et enfin donner une base de chacun des sous-espaces propres de M.

4. En déduire l'existence d'une matrice inversible $P\in\mathcal{M}_3(\mathbb{R})$ dont la première ligne est $\begin{pmatrix} 2 & -1 & 1 \end{pmatrix}$ et une matrice $D\in\mathcal{M}_3(\mathbb{R})$ telles que $M = PDP^{-1}$.

5. En déduire : $\forall n\in\mathbb{N},\quad M^n = PD^nP^{-1}$.

6. Déterminer P^{-1} puis en déduire que la deuxième ligne de M^n est donnée par :

$$\begin{pmatrix} \dfrac{-\left(-\frac{1}{2}\right)^n+1}{6} & \dfrac{-\left(-\frac{1}{2}\right)^n+2}{3} & \dfrac{-\left(-\frac{1}{2}\right)^n+1}{6} \end{pmatrix}.$$

Partie II : étude d'une chaîne de Markov

On considère deux urnes A et B contenant initialement une boule blanche et une boule noire chacune.
On procède à une suite d'épreuves vérifiant le protocole suivant :

• chaque épreuve consistant à tirer au hasard une boule dans chaque urne
• la boule tirée de A étant remise dans B et la boule tirée de B étant remise dans A.

Pour tout entier naturel n, on note X_n la variable aléatoire égale au nombre de boules blanches présentes dans A avant la $(n+1)^{\text{e}}$ épreuve.
On pose: $a_n = P(X_n = 0)\,, b_n = P(X_n = 1)\,, c_n = P(X_n = 2)$ et $U_n = \begin{pmatrix} a_n & b_n & c_n \end{pmatrix}$.

1. (a) Donner U_0.

   (b) Déterminer la loi de X_1 et en déduire les valeurs de a_1, b_1 et c_1.

   (c) Justifier rapidement que, pour tout entier naturel n supérieur ou égal à 1, $\big((X_n = 0)\,, (X_n = 1)\,, (X_n = 2)\big)$ est un système complet d'événements.

On admet dans la suite que, pour tout n de $\mathbb{N}^*$, les probabilités a_n, b_n et c_n sont non nulles.

2. **Simulation.**

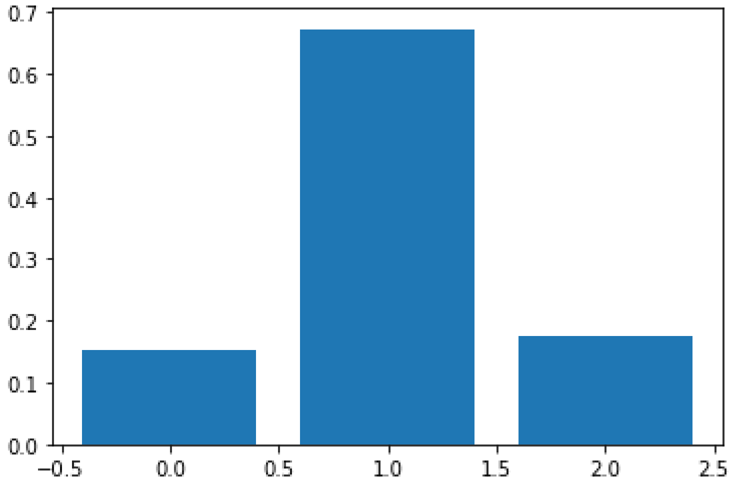
   (a) Ecrire une fonction d'en-tête `tirage(n,b):` qui prend en argument deux entiers naturels n et b et simule le tirage d'une bouel dans un urne contenant n boules noires et b boules blanches et qui renvoie 1 si la boule tirée est noire et 0 sinon.

   (b) On suppose que la fonction précédente a été correctement écrite. Recopier et compléter la fonction suivante qui permet de simuler la variable X_n :

```python
def simulX(n):
    nA,bA=1,1
    nB,bB=1,1
    for _ in range(n):
        tirageA=tirage(nA,bA)
        tirageB=tirage(nB,bB)
        if tirageA== ...... and tirageB == ........:
            bA=bA-1
            nA=nA+1
            bB=bB+1
            nB=nB-1
        elif tirageA==1 and tirageB ==0:
            bA= ....
            nA= ....
            bB= ....
            nB= ....
    return nA
```

   (c) On ajoute les commandes suivantes. Expliquer précisément ce qu'elles font. On joint la figure obtenue après leur exécution. Que peut-on conjecturer ?

```python
n=100
freq_etats=np.zeros(3)
for k in range(1000):
  freq_etats[simulX(n)-1]+=1
freq_etats=freq_etats/1000
plt.bar([1,2,3],freq_etats)
plt.show()
```



3. Soit n un entier naturel non nul.

   (a) Déterminer, en les justifiant, les probabilités conditionnelles $P_{(X_n=i)}\left(X_{n+1}=j\right)$, pour tout $(i,j)\in [\![0,2]\!]^2$.

   (b) En déduire le graphe représentant le processus de Markov décrit ci-dessus.

   (c) Utiliser la formule des probabilités totales pour exprimer a_{n+1}, b_{n+1} et c_{n+1} en fonction de a_n, b_n et c_n.

4. Vérifier que les relations trouvées à la question 2c) restent valables pour $n=0$.

5. (a) Établir la relation suivante :

$$\forall n\in\mathbb{N}, U_{n+1}=U_nM$$

   (b) En déduire la relation, valable pour tout entier naturel n :

$$U_n=U_0M^n$$

   (c) Déduire d'un résultat de la Partie I, la loi de X_n.

   (d) Déterminer les limites $\lim\limits_{n\longrightarrow+\infty} a_n$, $\lim\limits_{n\longrightarrow+\infty} b_n$ et $\lim\limits_{n\longrightarrow+\infty} c_n$. Comment interpréter ces résultats ?

   (e) Compléter la fonction suivante prenant en argument un entier n et renvoyant la matrice ligne U_n :

```python
def loiX(n):
    a,b,c= ... , ... , ...
    U=np.array([[a,b,c]])
    M=np.array([[0,1,0],[1/4,1/2,1/4],[0,1,0]])
    for _ in range(n):
        U = np.dot( ... , .... )
    return U
```

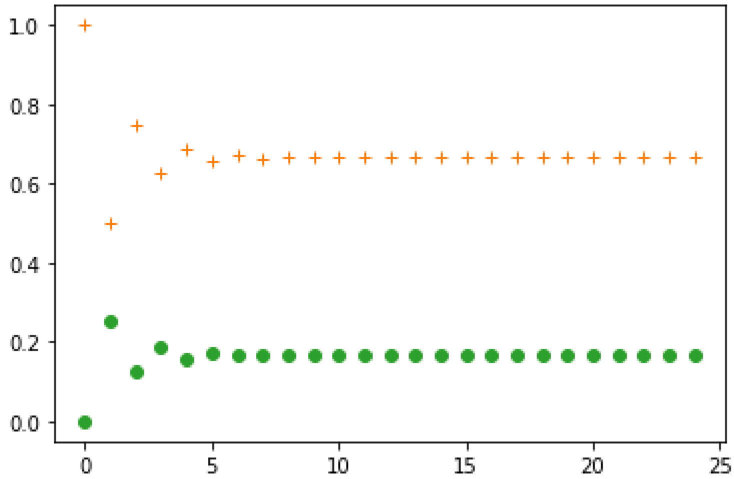
   On souhaite représenter graphiquement ces trois suites pour n allant de 0 à 25. Compléter le script suivant :

```python

n=25
a=[loiX(k)[ ... ,...] for k in range(n)]
b=[loiX(k)[ ... ,...]  for k in range(n)]
c=[loiX(k)[ ... ,...]  for k in range(n)]

plt.plot( ... ,'.')
plt.plot( ... ,'+')
plt.plot( ... ,'o')
plt.show()
```

   (f) Le script suivant permet d'obtenir la figure suivante :



Expliquer ce résultat graphique et notamment le fait qu'on ne voit que deux suites affichées.