Math ECG 2. 2022-2023

Mathématiques Appliquées - F. Gaunard
http://frederic.gaunard.com
ENC Bessières, Paris 17e.

Ecricome 2023

Mathématiques Appliquées
Solution

Merci à Mélanie Bronn, Tom Dutilleul et Romain Meurant pour leur relecture et leurs conseils afin d'améliorer cette solution.

Exercice 1

Soit n un entier naturel non nul.
Une urne contient n boules indiscernables au toucher numérotées de 1 à n.On tire une boule au hasard dans l'urne. Si cette boule tirée porte le numéro k, on place alors dans une seconde urne toutes les boules suivantes : une boule numérotée 1, deux boules numérotées 2 et plus généralement, pour tout $j \in [\![1,k]\!]$, j boules numérotées j, jusqu'à k boules numérotées k. Les boules de cette deuxième urne sont indiscernables au toucher. On effectue alors un tirage au hasard dans cette seconde urne.
On note X la variable aléatoire égale au numéro de la première boule tirée et Y la variable aléatoire égale au numéro de la deuxième boule tirée.

(1) Le premier tirage se fait dans une urne qui contient n boules indiscernables au toucher et chaque numéro est alors équiprobable; on a reconnu la loi uniforme

$$X \hookrightarrow \mathcal{U}([\![1,n]\!]).$$

En particulier, les formules du cours donnent directement $E(X) = \dfrac{n+1}{2}$, $\quad V(X) = \dfrac{n^2-1}{12}$.

(2) Dans le *pire* des cas, on pioche lors du premier tirage la boule numérotée 1 et il n'y a dans la deuxième urne qu'une boule numérotée 1. Si on pioche au premier coup la boule numérotée n, il y a ensuite des boules numérotées de 1 à n dans la deuxième urne. Ainsi, la valeur de la deuxième boule piochée est toujours une valeur entre 1 et n et toutes les valeurs sont possibles. On a donc $Y(\Omega) = [\![1,n]\!]$.

(3) Soit $k \in [\![1,n]\!]$.

(a) On suppose que l'évènement $[X = k]$ est réalisé.
D'après la description de l'expérience, on a disposé dans la deuxième urne un nombre de boules égal à

$$1 + 2 + ... + k = \sum_{j=1}^{k} j = \frac{k(k+1)}{2}.$$

(b) Si $j \in [\![1,k]\!]$, il y a dans l'urne des boules numérotées j (et il y en a exactement j). Sinon, les boules j ne font pas partie de l'urne (et la probabilité de les piocher alors nulle). On a

donc par équiprobabilité et avec la formule ci-dessus pour le total de boules :

$$P_{[X=k]}(Y=j) = \begin{cases} \dfrac{2j}{k(k+1)}, & \text{si } 1 \leq j \leq k \\ \\ 0, & \text{si } k < j \end{cases}$$

(4) (a) Commençons par mettre au même dénominateur.

$$\frac{a}{k} + \frac{b}{k+1} = \frac{a(k+1)+bk}{k(k+1)} = \frac{(a+b)k+a}{k(k+1)}.$$

Ainsi, par principe d'identification des polynômes

$$\left(\forall k \in \mathbb{N}^*, \qquad \frac{a}{k} + \frac{b}{k+1} = \frac{1}{k(k+1)}\right) \iff \begin{cases} a+b &= 0 \\ a &= 1 \end{cases}$$
$$\iff \begin{cases} b &= -1 \\ a &= 1 \end{cases}$$

On peut alors écrire, pour tout $k \in \mathbb{N}^*$,

$$\frac{1}{k(k+1)} = \frac{1}{k} - \frac{1}{k+1}.$$

(b) Soit $j \in [\![1, n]\!]$. D'après la formule des probabilités totales appliquée au s.c.e $\{[X=k] : k \in [\![1,n]\!]\}$, on a

$$\begin{aligned} P(Y=j) &= \sum_{k=1}^{n} P_{[X=k]}(Y=j)P(X=k) \\ &= \sum_{k=j}^{n} P_{[X=k]}(Y=j)P(X=k) \\ &= \sum_{k=j}^{n} \frac{2j}{k(k+1)} \times \frac{1}{n} = \frac{2j}{n} \sum_{k=j}^{n} \frac{1}{k(k+1)} \\ &= \frac{2j}{n} \sum_{k=j}^{n} \left(\frac{1}{k} - \frac{1}{k+1}\right) \\ &= \frac{2j}{n}\left(\frac{1}{j} - \frac{1}{n+1}\right) \\ &= \frac{2(n+1-j)}{n(n+1)}, \end{aligned}$$

ce qui est bien le résultat demandé.

(5) Y a un univers image fini, elle admet donc une espérance. De plus,

$$\begin{aligned}
E(Y) &= \sum_{j=1}^{n} jP(Y=j) = \sum_{j=1}^{n} j\frac{2(n+1-j)}{n(n+1)} \\
&= \frac{2}{n(n+1)}\sum_{j=1}^{n} j(n+1-j) = \frac{2}{n(n+1)}\left((n+1)\sum_{j=1}^{n} j - \sum_{j=1}^{n} j^2\right) \\
&= \frac{2}{n(n+1)}\left((n+1)\frac{n(n+1)}{2} - \frac{n(n+1)(2n+1)}{6}\right) \\
&= (n+1) - \frac{2n+1}{3} \\
&= \frac{n+2}{3},
\end{aligned}$$

ce qui est bien le résultat attendu.

(6) Soit $n \geq 2$. Si X et Y étaient indépendantes alors, pour tout couple $(k,j) \in [\![1,n]\!]^2$, on aurait

$$P_{[X=k]}(Y=j) = P(Y=j) \neq 0.$$

Or, la probabilité de gauche est nulle dès que $j > k$. Les variables X et Y ne sont donc pas indépendantes.
Pour $n = 1$, les deux variables aléatoires X et Y sont certaines et égales à 1... elles sont donc indépendantes.

(7) (a) Par le théorème de transfert, on a

$$\begin{aligned}
E(XY) &= \sum_{k=1}^{n}\sum_{j=1}^{n} kjP([X=k]\cap[Y=j]) = \sum_{k=1}^{n}\sum_{j=1}^{n} kjP([X=k])P_{[X=k]}([Y=j]) \\
&= \sum_{k=1}^{n}\sum_{j=1}^{k} kj \times \frac{1}{n} \times \frac{2j}{k(k+1)} \\
&= \frac{1}{n}\sum_{k=1}^{n}\frac{2}{k+1}\sum_{j=1}^{k} j^2 = \frac{1}{n}\sum_{k=1}^{n}\frac{2}{k+1} \times \frac{k(k+1)(2k+1)}{6} \\
&= \frac{1}{3n}\sum_{k=1}^{n} k(2k+1) = \frac{1}{3n}\left(2\sum_{k=1}^{n} k^2 + \sum_{k=1}^{n} k\right) \\
&= \frac{1}{3n}\left(\frac{n(n+1)(2n+1)}{3} + \frac{n(n+1)}{2}\right) \\
&= \frac{n+1}{18}\left(2(2n+1)+3\right) \\
&= \frac{(n+1)(4n+5)}{18},
\end{aligned}$$

ce qui fait plaisir car c'est ce qu'on demande!

(b) Par la formule de König-Huyguens,

$$\begin{aligned}
\mathrm{cov}(X,Y) &= E(XY) - E(X)E(Y) = \frac{(n+1)(4n+5)}{18} - \frac{n+1}{2} \times \frac{n+2}{3} \\
&= \frac{n+1}{18}\left(4n+5-3(n+2)\right) \\
&= \frac{(n+1)(n-1)}{18} = \frac{n^2-1}{18}
\end{aligned}$$

et c'est tout bon.
(On remarque que, pour $n = 1$, la covariance est nulle, ce qui est cohérent avec ce qu'on a mentionné ci-avant quant à l'indépendance de X et Y.)

(8) (a) Il suffit d'utiliser l'instruction `append` pour compléter la liste avec deux boucles `for`.

```
def seconde_urne(k):
    L=[ ]
    for j in range(1, k+1) :
        for i in range(j):
            L.append(j)
    return L
```

(b) La variable X se simule grâce à la commande `rd.randint(1, n+1)`. On crée l'urne 2 à l'aide de la valeur de X. On prend ensuite le terme de la liste `urne2` à la position i (où i est choisi aléatoirement uniformément parmi le nombre de boules disponibles) pour Y. Ceci donne

```
import numpy.random as rd

def simul_XY(n) :
    X = rd.randint(1, n+1)
    urne2=seconde_urne(X)
    nb=len(urne2) # nombre de boules dans l'urne 2
    i=rd.randint(0, nb)
    Y=urne2[i]
    return X,Y
```

(c) Le programme proposé est le suivant

```
def fonction(n):
    liste=[0]*n
    for i in range(10000):
        j = simul_XY[1]
        liste[j-1]=liste[j-1]+1/10000
    return liste
```

La variable `liste` contient la liste des fréquences de chaque valeur prise par Y lors de 10000 simulations de celle-ci. C'est à dire qu'on *estime* la loi de Y (les fréquences observées donnent des valeurs approchées des valeurs théoriques $P(Y = j)$).
On commence, avec la commande `liste=[0]*n` par créer une liste de n zéros qui va être actualisé. Le $j-$ième terme de la liste (indexé en Python par $j - 1$) contient la fréquence de passage par la valeur j.
En effet, la commande `j = simul_XY[1]` simule Y (c'est la deuxième composante du couple (X, Y) car on indexe en commençant par 0...) et les composantes de `liste` sont les fréquences qui augmente de 1 divisé par l'effectif total dès lors que Y a pris la valeur de la composante en question.

(9) Dans toute cette question, on suppose $n = 20$.

(a) Le point moyen a pour coordonnées $(\overline{x}, \overline{y})$ où $\overline{x}$ et $\overline{y}$ désignent respectivement les moyennes des 20 valeurs obtenues par les simulations de X et de Y. Lorsque n devient grand, la *loi faible des grands nombres* permet d'affirmer (enfin il faudrait dire que X et Y admettent une variance mais c'est le cas car les univers images sont finis) que la moyenne empirique d'un

$n-$échantillon fournit une bonne[1] estimation de l'espérance. Donc le point moyen devrait avoir des coordonnées

$$\overline{x} \simeq E(X) = \frac{21}{2} = 10,5, \qquad \text{et } \overline{y} \simeq E(Y) = \frac{22}{3} \simeq 7,33.$$

(b) Les valeurs prises par X et Y sont entre 1 et 20: on peut éliminer la figure 1 (le nuage de points ne correspond pas). La covariance de X et Y est positive, la droite de régression est donc croissante. On peut éliminer la 4. La droite de régression passe par le point moyen, donc elle doit passer par $(10,5; 7,33)$. On peut donc éliminer les figures 2 et 4 où c'est la droite de régression qui ne correpond pas.
C'est donc la figure 3 qui correspond au nuage de points étudié.

À titre de remarque et pour le plaisir seulement, on se permet d'ajouter quelques lignes de code qui permettent de générer un tel nuage de points. On observera que certains points sont d'ailleurs confondus.

```
import matplotlib.pyplot as plt

nuageX = []
nuageY = []
for i in range(50):
    X, Y = simul_XY(20)
    nuageX.append(X)
    nuageY.append(Y)

EX = np.mean(nuageX)
VX = np.mean(np.multiply(nuageX,nuageX)) - EX**2
EY = np.mean(nuageY)
EXY = np.mean(np.multiply(nuageX,nuageY))
covXY = EXY-EX*EY
a = covXY/VX
b = EY-a*EX

plt.plot(nuageX,nuageY,'x')
plt.plot([b,20],[0,20*a+b],'--')
plt.show()
```

Exercice 2

On considère la fonction f définie sur $]0,+\infty[$ par

$$\forall x \in]0,+\infty[, \qquad f(x) = \frac{e^{\frac{x}{2}}}{\sqrt{x}}.$$

On rappelle que $2 < e < 3$.

(1) (a) La fonction $x \mapsto \sqrt{x}$ est dérivable sur $]0,+\infty[$ et ne prend, sur cet intervalle que des valeurs strictement positives (elle ne s'annule donc pas). La fonction $x \mapsto x/2$ est affine donc dérivable sur $\mathbb{R}$ (et *a fortiori* sur $]0,+\infty[$) et la fonction exponentielle est dérivable partout. Par composition puis quotient, f est donc dérivable sur $]0,+\infty[$. De plus, pour tout $x > 0$,

[1] c'est un estimateur convergent et non biaisé mais ces notions ne sont plus vraiment au programme

on a

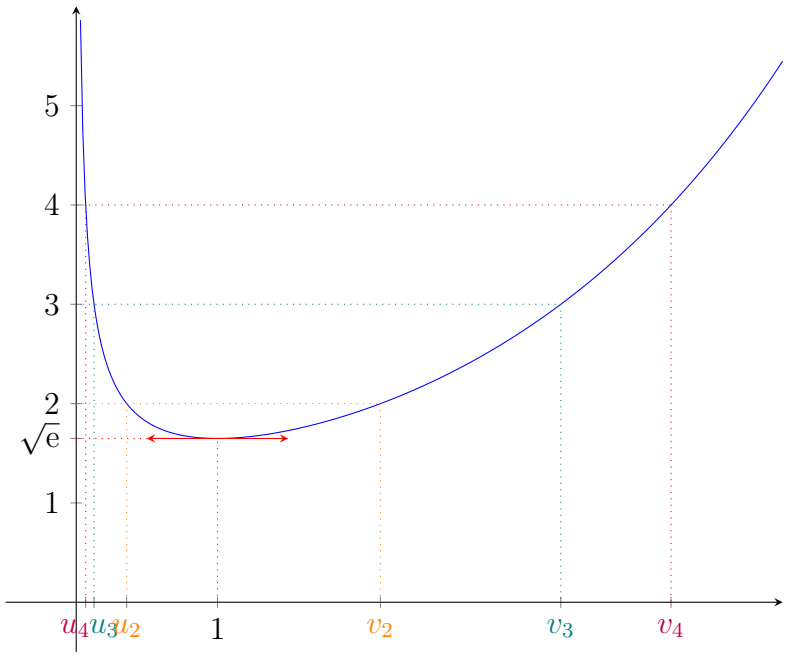
$$f'(x) = \frac{\frac{1}{2}e^{\frac{x}{2}}\sqrt{x} - \frac{e^{\frac{x}{2}}}{2\sqrt{x}}}{x} = \frac{x-1}{2x\sqrt{x}}e^{\frac{x}{2}} = \frac{x-1}{2x}f(x).$$

(b) Pour $x > 0$, $f(x) = x^{-1/2}e^{x/2} \underset{x\to+\infty}{\longrightarrow} +\infty$ par croissance comparée.
En 0, on a $e^{x/2} \to 1$, et $\sqrt{x} \to 0^+$ donc, par algèbre des limites, $f(x) \underset{x\to 0}{\longrightarrow} +\infty$.

Pour tracer l'allure de la courbe, il nous faut les variations qui sont données sans la moindre difficulté par le signe de la dérivée ci-dessus.

x	0		1		$+\infty$
$f'(x)$	‖	$-$	0	$+$	
f	‖ $+\infty$	↘	$\sqrt{e}$	↗	$+\infty$

(c) On obtient la figure



(On a fait figurer les trois termes des deux suites (u_n) et (v_n) étudiées ci après par plaisir, ce n'était pas demandé.)

(d) Il s'agit bien sûr d'appliquer le théorème de bijection deux fois. Commençons par observer que, comme $2 < e < 3 < 4$, on a $\sqrt{e} < \sqrt{4} = 2$ et donc, pour tout entier $n \geq 2$, $n \in]\sqrt{e}, +\infty[$. Comme f est continue et strictement décroissante sur $]0, 1[$, elle réalise (par le théorème de bijection) une bijection de $]0, 1[$ sur $]\sqrt{e}, +\infty[$. En particulier, n qui est un élement de l'intervalle image admet donc un unique antécédent par f, noté u_n dans $]0, 1[$.
De même, sur $]1, +\infty[$, f est continue et strictement croissante et réalise une bijection de cet intervalle sur $]\sqrt{e}, +\infty[$ et n admet alors un unique antécédent par f sur $]1, +\infty[$ noté v_n. Au final, l'équation $f(x) = n$ admet bien exactement deux solutions u_n et v_n telles que

$$0 < u_n < 1 < v_n.$$

(2) (a) Toujours par le théorème de bijection, la bijection réciproque de la restriction de f à $]1,+\infty[$ est également strictement croissante. Notons-là g. Comme $f(v_n) = n$, alors $v_n = g(n)$. Ayant $n < n+1$, en appliquant g strictement croissante, on a $v_n = g(n) < g(n+1) = v_{n+1}$ et (v_n) est (strictement) croissante.

(b) Par le théorème de convergence monotone, une suite croissante est soit majorée et alors elle converge, soit non majorée et elle diverge vers $+\infty$. Supposons que (v_n) soit majorée. Alors elle converge vers une certaine limite réelle ℓ qui vérifie $\ell \geq 1$. Mais alors, comme $f(v_n) = n$ et que f est continue

$$+\infty = \lim_{n\to+\infty} n = \lim_{n\to+\infty} f(v_n) = f\left(\lim_{n\to+\infty} v_n\right) = f(\ell),$$

ce qui est absurde. Donc (v_n) n'est pas majorée et diverge vers $+\infty$.

(3) (a) On raisonne de manière complètement analogue à la question précédente. La bijection réciproque h de la restriction de f à $]0,1[$ est strictement décroissante donc

$$u_{n+1} = h(n+1) < h(n) = u_n$$

et la suite (u_n) est bien décroissante.

(b) Comme (u_n) est minorée par 0 (et décroissante donc) le théorème de convergence monotone permet d'affirmer qu'elle converge vers une certaine limite réelle $\ell \in [0;1]$.

(c) Supposons que $\ell \in]0,1]$. Alors comme précédemment,

$$+\infty = \lim_{n\to+\infty} n = \lim_{n\to+\infty} f(u_n) = f\left(\lim_{n\to+\infty} u_n\right) = f(\ell),$$

ce qui est absurde. Donc $\ell = 0$.

(d) Par définition de u_n, on a

$$e^{\frac{u_n}{2}} = \sqrt{u_n} n \iff e^{u_n} = n^2 u_n.$$

Comme $u_n \to 0$, par continuité de l'exponentielle, on a $e^{u_n} \to 1$ ce qui donne bien

$$\lim_{n\to+\infty} n^2 u_n = 1$$

et permet d'écrire l'équivalence

$$u_n \sim \frac{1}{n^2}, \qquad n \to +\infty.$$

(4) (a) Le sujet nous rappelle comment fonctionne l'algorithme de recherche d'une solution par dichotomie. On divise donc l'intervalle de recherche en 2 et poursuivant la recherche à gauche ou à droite, selon que $f(c) > n$ ou non, tant que la taille de l'intervalle $(b-a)$ est supérieure à la précision voulue pour l'estimation. Si $f(c) < n$, comme f est strictement décroissante sur $]0,1[$ où se trouve u_n, c'est que notre solution est dans la moitié de gauche. Ceci donne le programme suivant

```
import numpy as np
def approx_u(n, eps) :
    a=0
    b=1
    while (b-a) > eps :
        c=(a+b)/2 # milieu de l'intervalle
        if np.exp(c/2)/np.sqrt(c) < n :  # si f(c) < n
            b=c # alors on cherche à gauche
        else :
            a=c # sinon on cherche à droite
    return (a+b)/2
```

(b) Cette question nécessite de calculer la somme des valeurs approchées avec une boucle `for`...sauf que.. si on veut une erreur totale de l'ordre de `eps`, une solution pour que cette erreur totale ne soit pas trop grosse est de prendre une erreur pour chaque terme sommé de l'ordre de `eps`$/(N-1)$ (car il y a $N-1$ termes dans la somme).

```
def sp(eps, N):
    s=approx_u(2, eps/(N-1))
    for k in range(3, N+1) :
        s=s+approx_u(k, eps/(N-1))
    return s
```

Exercice 3

Partie 1

On considère la matrice $A = \begin{pmatrix} 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{pmatrix}$. On note f l'endomorphisme de $\mathbb{R}^4$ représenté par A dans la base canonique $\mathcal{C} = (e_1, e_2, e_3, e_4)$ de $\mathbb{R}^4$.

(1) On considère les vecteurs suivants de $\mathbb{R}^4$:

$$u_1 = (-1, 1, 0, 1), \quad u_2 = (0, -1, 1, 0), \quad u_3 = (0, 1, 1, 0), \quad u_4 = (1, 0, 0, 1).$$

On note $\mathcal{B} = (u_1, u_2, u_3, u_4)$.

(a) La famille $\mathcal{B}$ est constituée de 4 vecteurs de $\mathbb{R}^4$ qui est un espace vectoriel de dimension 4. Il suffit donc de montrer qu'elle est libre pour qu'elle en forme une base.
Soient $a, b, c, d \in \mathbb{R}$.

$$au_1 + bu_2 + cu_3 + du_4 = 0 \iff \begin{cases} -a+d &= 0 \\ a-b+c &= 0 \\ b+c &= 0 \\ a+d &= 0 \end{cases} \iff \begin{cases} a-b+c &= 0 \\ b+c &= 0 \\ -b+c+d &= 0 \\ b-c+d &= 0 \end{cases}$$

$$\iff \begin{cases} a-b+c &= 0 \\ b+c &= 0 \\ 2c+d &= 0 \\ -2c+d &= 0 \end{cases} \iff \begin{cases} a-b+c &= 0 \\ b+c &= 0 \\ 2c+d &= 0 \\ 2d &= 0 \end{cases}$$

$$\iff a = b = c = d = 0$$

Ainsi, la famille $\mathcal{B}$ est bien libre et forme une base de $\mathbb{R}^4$.

(b) Il faut commencer par exprimer les images par f des quatre vecteurs de $\mathcal{B}$, qu'on ne manque pas d'exprimer en fonction des vecteurs de $\mathcal{B}$.

- $A \begin{pmatrix} -1 \\ 1 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$ donc $f(u_1) = 0$.

- $A\begin{pmatrix}0\\-1\\1\\0\end{pmatrix} = \begin{pmatrix}0\\0\\0\\0\end{pmatrix}$ donc $f(u_2) = 0$.
- $A\begin{pmatrix}0\\1\\1\\0\end{pmatrix} = \begin{pmatrix}0\\2\\2\\0\end{pmatrix}$ donc $f(u_3) = 2u_3$.
- $A\begin{pmatrix}1\\0\\0\\1\end{pmatrix} = \begin{pmatrix}2\\1\\1\\2\end{pmatrix}$ donc $f(u_4) = 2u_4 + u_3$.

Ceci permet d'écrire

$$\mathrm{Mat}(f, \mathcal{B}) = \begin{pmatrix}0&0&0&0\\0&0&0&0\\0&0&2&1\\0&0&0&2\end{pmatrix}$$

(c) En notant P la matrice de passage de la base canonique $\mathcal{C}$ vers $\mathcal{B}$ (qui est donc bien inversible), la formule de changement de base donne bien

$$A = PTP^{-1},$$

où

$$T = \mathrm{Mat}(f, \mathcal{B}) = \begin{pmatrix}0&0&0&0\\0&0&0&0\\0&0&2&1\\0&0&0&2\end{pmatrix}$$

est bien triangulaire (supérieure) et

$$P = \begin{pmatrix}-1&0&0&1\\1&-1&1&0\\0&1&1&0\\1&0&0&1\end{pmatrix}.$$

(2) (a) C'est un calcul.

$$A^2 = \begin{pmatrix}1&0&0&1\\1&1&1&0\\1&1&1&0\\1&0&0&1\end{pmatrix}\begin{pmatrix}1&0&0&1\\1&1&1&0\\1&1&1&0\\1&0&0&1\end{pmatrix} = \begin{pmatrix}2&0&0&2\\3&2&2&1\\3&2&2&1\\2&0&0&2\end{pmatrix}$$

Il suit que

$$A^3 = A^2 \cdot A = \begin{pmatrix}2&0&0&2\\3&2&2&1\\3&2&2&1\\2&0&0&2\end{pmatrix}\begin{pmatrix}1&0&0&1\\1&1&1&0\\1&1&1&0\\1&0&0&1\end{pmatrix} = \begin{pmatrix}4&0&0&4\\8&4&4&4\\8&4&4&4\\4&0&0&4\end{pmatrix}$$

On constate qu'on a bien

$$\begin{aligned} 4A^2-4A &= 4\begin{pmatrix}2&0&0&2\\3&2&2&1\\3&2&2&1\\2&0&0&2\end{pmatrix} - 4\begin{pmatrix}1&0&0&1\\1&1&1&0\\1&1&1&0\\1&0&0&1\end{pmatrix} \\ &= \begin{pmatrix}8&0&0&8\\12&8&8&4\\12&8&8&4\\8&0&0&8\end{pmatrix} - \begin{pmatrix}4&0&0&4\\4&4&4&0\\4&4&4&0\\4&0&0&4\end{pmatrix} \\ &= \begin{pmatrix}4&0&0&4\\8&4&4&4\\8&4&4&4\\4&0&0&4\end{pmatrix} \\ &= A^3. \end{aligned}$$

(b) Procédons donc par récurrence en construisant les termes de la suite de proche en proche.

- <u>initialisation</u>. Pour $n=1$, $A = 0\times A^2 + 1\times A$. En posant $a_1 = 0$ et $b_1 = 1$, la relation est vraie pour $n=1$.

- <u>hérédité</u>. Supposons que, pour un certain $n\in\mathbb{N}^*$, il existe deux réels a_n et b_n tels que $A^n = a_nA^2 + b_nA$. Il suit que

$$\begin{aligned} A^{n+1} &= A\cdot A^n \\ &\underset{\text{H.R}}{=} A\left(a_nA^2+b_nA\right) \\ &= a_nA^3 + b_nA^2 \\ &= a_n(4A^2-4A) + b_nA^2 \\ &= (4a_n+b_n)A^2 - 4a_nA. \end{aligned}$$

  En posant $a_{n+1} = 4a_n + b_n$ et $b_{n+1} = -4a_n$, on a bien

$$A^{n+1} = a_{n+1}A^2 + b_{n+1}A,$$

  et la récurrence est terminée.

(3) (a) Soit $n\geq 1$. D'après ce qui précède

$$\begin{aligned} a_{n+2} &= 4a_{n+1} + b_{n+1} \\ &= 4a_{n+1} - 4a_n \end{aligned}$$

et la suite (a_n) est récurrente linéaire d'ordre 2.

(b) On associe à la suite son équation caractéristique $q^2-4q+4=0$ qui admet une unique solution $q=2$. D'après le cours, on peut alors affirmer qu'il existe deux constantes $\lambda,\mu\in\mathbb{R}$ telles que

$$a_n = (\lambda+\mu n)2^n.$$

Pour déterminer λ et μ, on injecte les valeurs des deux premières termes : $a_1 = 0$ et $a_2 = 4a_1 + b_1 = 1$ ce qui donne

$$\left\{\begin{array}{rcl} 2(\lambda+\mu) &=& 0 \\ 4(\lambda+2\mu) &=& 1 \end{array}\right. \iff \left\{\begin{array}{rcl} \lambda &=& -1/4 \\ \mu &=& 1/4 \end{array}\right.$$

et on peut conclure que, pour tout $n\geq 1$,

$$a_n = \frac{1}{4}(-1+n)2^n = (n-1)2^{n-2}.$$

(c) Pour tout $n \geq 2$, on a $b_n = -4a_{n-1}$ et donc

$$b_n = -4(n-1-1)2^{n-1-2} = -(n-2)2^{n-1}.$$

On remarque que cette formule est encore valide pour $n = 1$.

(4) En faisant le bilan de ce qui précède, pour $n \in \mathbb{N}^*$,

$$\begin{aligned}
A^n &= a_n A^2 + b_n A \\
&= (n-1)2^{n-2} \begin{pmatrix} 2 & 0 & 0 & 2 \\ 3 & 2 & 2 & 1 \\ 3 & 2 & 2 & 1 \\ 2 & 0 & 0 & 2 \end{pmatrix} - (n-2)2^{n-1} \begin{pmatrix} 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{pmatrix} \\
&= 2^{n-2} \left((n-1) \begin{pmatrix} 2 & 0 & 0 & 2 \\ 3 & 2 & 2 & 1 \\ 3 & 2 & 2 & 1 \\ 2 & 0 & 0 & 2 \end{pmatrix} - (n-2) \begin{pmatrix} 2 & 0 & 0 & 2 \\ 2 & 2 & 2 & 0 \\ 2 & 2 & 2 & 0 \\ 2 & 0 & 0 & 2 \end{pmatrix} \right) \\
&= 2^{n-2} \begin{pmatrix} 2 & 0 & 0 & 2 \\ n+1 & 2 & 2 & n-1 \\ n+1 & 2 & 2 & n-1 \\ 2 & 0 & 0 & 2 \end{pmatrix} \\
&= \begin{pmatrix} 2^{n-1} & 0 & 0 & 2^{n-1} \\ (n+1)2^{n-2} & 2^{n-1} & 2^{n-1} & (n-1)2^{n-2} \\ (n+1)2^{n-2} & 2^{n-1} & 2^{n-1} & (n-1)2^{n-2} \\ 2^{n-1} & 0 & 0 & 2^{n-1} \end{pmatrix},
\end{aligned}$$

ce qui est le résultat attendu. Ouf.

Partie 2

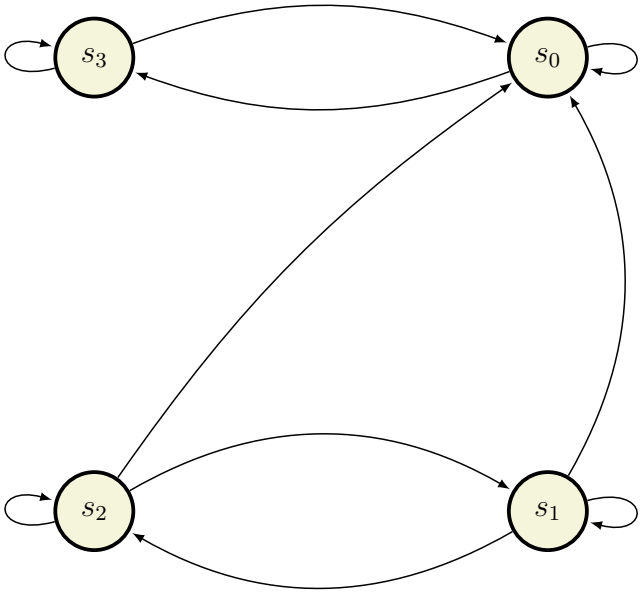
Soit p un entier naturel non nul et G un graphe non pondéré orienté à p sommets . On note $s_0, s_1, ..., s_{p-1}$ les sommets de G.

(5) (a) On appelle matrice d'adjacence associée à G la matrice $M = (a_{i,j})$ dont chaque terme $a_{i,j}$ est égal à 1 ou à 0 selon qu'il existe une arrête orientée allant du sommet i vers le sommet j.

(b) Soient n un entier non nul, i un entier de $[\![1, p]\!]$ et j un entier de $[\![1, p]\!]$.
Le coefficient situé à la $i-$ème ligne et $j-$ème colonne de M^n est le nombre de chemins de longueur n du sommet i au sommet j.

(6) Dans cette question uniquement, on suppose $p = 4$ et que la matrice d'adjacence du graphe G est la matrice A de la Partie 1.

(a) Le diagramme correspondant est le suivant



(b) Il n'existe pas de chemin orienté de s_3 vers s_2: le graphe n'est pas connexe (on dira plutôt qu'il n'est pas *fortement connexe*). On peut aussi aller chercher un résultat du cours : "*Si A est la matrice d'adjacence d'un graphe orienté G à n sommets, G est connexe si et seulement si la matrice $I_n + A + \cdots + A^{n-1}$ a tous ses coefficients* ***strictement*** *positifs.*" Ce qui n'est pas le cas dans l'exercice car il y a des coefficients nuls quand on calcule $I + A + A^2 + A^3$.
En revanche, le graphe est *faiblement connexe*; il existe bien un chemin menant de n'importe quel sommet à n'importe quel autre sommet sans tenir compte de l'orientation des arcs.

(c) Soit n un entier naturel non nul.
Comme énoncé ci-avant le nombre de chemins de longueur n du sommet s_3 au sommet s_0 est donné par le coefficients à la $4-$ième ligne et première colonne de la matrice A^n. D'après les résultats de la Partie 1, il y en a donc 2^{n-1}.

(7) Dans cette question et les suivantes, on revient au cas général décrit au début de la Partie 2.
Soit s un sommet de G. On dit que le sommet t est un voisin de s quand $s \neq t$ et (s, t) est une arête du graphe.
Comme le graphe est orienté, si t est voisin de s, alors s n'est pas forcément voisin de t.
On appelle liste d'adjacence du graphe G une liste de p sous-listes telle que, pour tout entier $k \in [\![0, p-1]\!]$, la sous-liste située à la position k contient tous les numéros des sommets voisins de s_k.

☞ On observe que la définition de sommets voisins ci-dessus empêche *a priori* un sommet d'être voisin de lui-même, alors que ceci est contredit par la liste d'adjacence ci-dessus... On poursuit donc la résolution de cet exercice en autorisant un sommet à être voisin de lui même s'il existe une arête du sommet vers lui-même.

Cela dit ce choix rentre en conflit avec le résultat sur le nombre de chemins de longueurs n et la matrice d'adjacence à la puissance n utilisé ci-avant.

Nul doute que les correctrices et correcteurs sauront êtres souples dans l'intérêt des candidat.e.s. Malgré tout, on ne peut que signaler des problèmes à plusieurs reprises dans ce sujet sur les graphes...

Pour écrire la fonction demandée, il faut donc parcourir les lignes de la matrice d'adjacence et tester si le coefficient $a_{i,j}$ est nul ou non ce qui indiquera si le sommet s_{i-1} est voisin du sommet s_{j-1}. On propose alors le programme suivant

```
def matrice_vers_liste(A) :
    p=len(A)
    L=[ ]
    for i in range(p) : # on parcourt la ligne i
        L.append([ ]) # une nouvelle sous-liste
        for j in range(p) : # on parcourt la colonne j
            if A[i][j] ==1 :
                L[i].append(j) # le sommet j est voisin du sommet i
    return L
```

(8) On chercher à écrire une fonction en langage Python permettant d'obtenir la longueur du plus court chemin menant d'un sommet de départ s_i à chaque sommet du graphe.
On aura reconnu un algorithme de *parcours en largeur*.

(a) En suivant les étapes de l'algorithme décrit, on obtient la liste `[1, 0, 1, 2]`.

(b) On complète l'algorithme

```
def parcours(L, i0):
    p = len(L)
    distances= [p]*p # liste de p éléments valant p
    distances[i0]=0
    a_explorer=[i0]
    marques = [i0]
    while a_explorer != [ ] :
        s = a_explorer[0] # on prend le premier terme de la liste
        del a_explorer[0] # on l'enlève de la liste
        for v in L[s] : # pour tous les sommets voisins de s
            if v not in marques : # si il n'est pas marqué
                marques.append(v)
                a_explorer.append(v)
                distances[v]=distances[s]+1
    return distances
```

(c) En remplaçant le `return distances` par

```
return marques
```

on renvoie bien la liste de tous les sommets accessibles depuis $i0$.
On peut aussi voir que le programme précédent renvoie la liste des distances mais elle est initialisée à p. Si c'est toujours p après exécution de l'algorithme c'est qu'aucun chemin n'a été trouvé. Sinon la longueur serait inférieure ou égale à $p-1$ vu qu'il y a p sommets. Ainsi, ue réponse alternative serait de remplacer le `return distances` par les commandes suivantes

```
chemins=[ ]
for k in range(p) :
    if distances[k] < p :
        chemins.append(k)
return chemins
```