Exercice 1: [Correction]

Soit (A, B) un couple de variables aléatoires discrètes, définies sur un espace probabilisé $(\Omega, \mathcal{B}, P)$, à valeurs dans $\mathbb{N} \times \mathbb{N}$, telles que, pour tout (i, j) de $\mathbb{N}^2$, on a :

$$P[(A = i) \bigcap (B = j)] = C \times \frac{e^{-i}}{j^2 + 3j + 2}.$$

1. **a)** Montrer que $C = 1 - e^{-1}$.
   b) Déterminer la loi de A, préciser son espérance et sa variance.
   c) Les variables aléatoires A et B sont-elles indépendantes ?
2. Soit Z la variable aléatoire égale à $5A + 7B$.
   a) Écrire un programme qui, n étant donné, calcule $P(Z = n)$.
   b) En déduire la probabilité de l'événement $(Z = 23)$. Démontrer le résultat.

Exercice 2: [Correction]

Un sac contient n jetons numérotées de 1 à n. On tire un jeton au hasard, on note son numéro et on le remet dans le sac. On appelle X la variable aléatoire qui prend pour valeur ce numéro.

Lorsque ce numéro est k, on distribue au hasard k billes dans p boîtes $B_1, \dots, B_p$. On désigne par Y_i la variable aléatoire égale au nombre de billes reçues par la boîte B_i $(i \in \{1, \dots, p\})$.

1. Écrire un programme qui permette de simuler informatiquement la p-liste $(Y_1, \dots, Y_p)$.
2. **a)** Étant donné une variable aléatoire Z, et un nombre entier N grand. Donnez une variable qui approche de $\mathbb{E}[Z]$ et rappelez le théorème permettant de faire cette affirmation.
   b) Écrire un programme permettant de donner une approximation de l'ensemble des $\mathbb{E}[Y_i]$ pour n, p donnés.
   c) Testez ce programme pour diverses valeurs de n et p. À n et p fixés, comparez les $\mathbb{E}[Y_i]$ pour $i \in [\![1, p]\!]$. Le résultat est-il cohérent ?
3. Déterminer la loi du couple (X, Y_i) pour $i \in \{1, \dots, p\}$.
4. On cherche maintenant à établir *a priori* une relation entre $\mathbb{E}[Y_1]$ et n.
   a) Écrire un programme permettant de dessiner, à p fixé, le graphique des valeurs approchées de $\mathbb{E}[Y_1]$ pour une abscisse n valant $10, 20, \dots, 100$.
   b) Testez ce programme pour diverses valeurs de p. Qu'en déduisez-vous ?
   c) Vérifions maintenant l'observation faite dans la question précédente. En vous servant de la loi du couple (X, Y_i), calculer l'espérance de Y_i pour $i \in \{1, \dots, p\}$.
5. Déterminer l'espérance de la variable aléatoire $\frac{Y_i}{X}$.

Exercice 3: [Correction]

Une puce se déplace dans un plan muni d'un repère orthonormé. Elle part de l'origine O des coordonnées à l'instant 0. Si à l'instant $t = k, k \in \mathbb{N}$ il se situe au point de coordonnées (X_k, Y_k), alors à l'instant $t = k + 1$ il se trouve au point de coordonnées (X_{k+1}, Y_{k+1}) de sorte que $A_{k+1} = X_{k+1} - X_k$ et $B_{k+1} = Y_{k+1} - Y_k$ suivent la loi normale $\mathcal{N}(0, 1)$. On suppose les A_i et les B_j mutuellement indépendantes. Soit $\Phi_{m,\sigma}$ la fonction de répartition de la loi normale $\mathcal{N}(m, \sigma^2)$ et $\varphi_{m,\sigma}$ une densité de cette loi.

Si Z est une variable suivant une loi normale $\mathcal{N}(0, 1)$, on donne/rappelle les commandes suivantes en python :

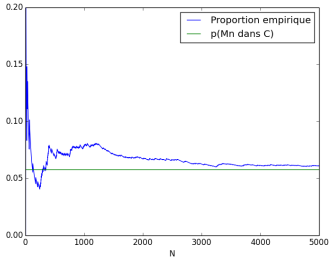
```
1  import scipy.stats as sc
2
3  sc.norm.rvs()   # simule une valeur aléatoire de Z.
4  sc.norm.pdf(x)  # renvoie Phi_{0,1}(x)
```

La commande `hist` du module `matplotlib.pyplot` permet de réaliser des histogrammes de la façon suivante :

```
1  plt.hist(L)     # dessine un histogramme des
      données correspondant à la liste des caractères
      L de chaque individu. Par défaut, la découpe se
      fait en 10 classes
2  plt.hist(L,density=True}    # rend un histogramme d
      'aire totale = 1
```

1. Simuler informatiquement une série de n déplacements sous forme de graphique.
2. **a)** Quelle est la loi suivie par X_n ?
   b) Soit M_n le point de coordonnées (X_n, Y_n). Exprimer, à l'aide de la fonction $\Phi_{0,1}$, la probabilité qu'à l'instant n, le point M_n se trouve dans le carré $C = [-1, 1]^2$.
3. Soit n fixé. On se donne un N échantillon du couple (X_n, Y_n).
   Le graphique ci-dessous exprime, pour $n = 10$, la courbe donnant la proportion empirique de points M_n présents dans C et la probabilité théorique $p(M_n \in C)$.

Que peut-on déd héoriquement ce résultat.



4. Soit D_n la distance de M_n à l'origine : $D_n = \sqrt{X_n^2 + Y_n^2}$.
- **a)** Établir un histogramme pour N réalisations de D_n^2 pour N grand.
- **b)** Fixez un n et observez. De quelle loi connue semble se rapprocher (graphiquement) la loi de D_n^2 ?

5. Considérons ici que l'on acccepte la conjecture faite dans la question précédente.
- **a)** Par quel moyen peut-on obtenir une valeur approchée du paramètre de cette loi ?
- **b)** En déduire un programme `inv_paramDn2(n)` permettant, pour un n fixé, de déterminer une approximation de l'inverse du paramètre de la loi de D_n^2.
- **c)** Dessinez le graphique des `[n,inv_paramDn2(n)]` pour n variant de 1 à 100 et proposez finalement une possible loi pour D_n^2.

Exercice 4: [Correction] Oral Agro-Veto 2018

On pourra utiliser pour les programmes ***Python*** *les fonctions* ***matrix*** *et* ***linalg.matrix_rank()*** *du module* ***numpy****, qui permettent respectivement de définir une matrice et de déterminer le rang d'une matrice, comme le montre l'exemple suivant :*

```
1  import numpy as np
2  A=np.matrix( [ [1 ,2 ,1] , [2 ,3 ,2], [3 ,5 ,3]] )
```

Dans le shell,

```
np.linalg.matrix_rank (A)
```

donne le rang de $A = \begin{pmatrix} 1 & 2 & 1 \\ 2 & 3 & 2 \\ 3 & 5 & 3 \end{pmatrix}$*, c'est-à-dire* 2.

On considère la matrice :

$$A = \begin{pmatrix} 3 & 1 & 1 \\ 1 & 0 & 1 \\ -3 & 0 & -1 \end{pmatrix}.$$

1.
- **a)** Écrire une fonction Python prenant en arguments deux vecteurs de taille 3 et renvoyant un booléen (`True` ou `False`) indiquant s'ils sont colinéaires. *(On pourra représenter les vecteurs par des listes).*
- **b)** Écrire une fonction Python `vecteurs_propres(u)` prenant en argument un vecteur de taille 3 et renvoyant un booléen (`True` ou `False`) indiquant s'il est un vecteur propre de A.

2.
- **a)** Vérifier (à l'aide de Python par exemple) que -1, 1, 2 sont valeurs propres de A et préciser ensuite pour chacune un vecteur propre associé.
- **b)** La matrice A est-elle diagonalisable ?

3. Soient $X_1, \cdots, X_n$, n variables aléatoires indépendantes suivant la loi de Bernoulli de paramètre $p \in]0;1[$. On note : $M_n = \dfrac{1}{n}\sum_{k=1}^{n} X_k$ et $M_n^\star = \dfrac{M_n - p}{\sqrt{\dfrac{p(1-p)}{n}}}$.
- **a)** Donner, pour $\alpha \in \mathbb{R}_+^\star$, l'approximation de la probabilité

$$P([-\alpha < M_n^\star < \alpha])$$

donnée par le théorème central limite.
- **b)** En déduire que $P\left(M_n - \dfrac{1}{\sqrt{n}} \leqslant p \leqslant M_n + \dfrac{1}{\sqrt{n}}\right) \geqslant 0,95$.

On pourra admettre que, $\forall x \in [0;1],\ x(1-x) \leq \dfrac{1}{4}$ *et si* Φ *désigne la fonction de répartition d'une variable suivant une loi normale centrée réduite, alors* $\Phi(1,96) \approx 0,975$.

4. On note N_V le nombre de vecteurs propres de A dont les cœfficients sont des entiers de $[\![-5;5]\!]$.
- **a)** Expliquer comment le programme suivant permet d'estimer la valeur de N_V :

```
1  def simul ():
2      u = [ randint ( -5 ,5) for k in range (3) ]
3      return vecteurs_propres (u)
4  n = 10000            # Valeur de n a definir.
5  nb = 0
6  for k in range (n):
7      if simul ():
8          nb += 1
9  print ( round (nb/n *11**3)) # round (x) = l'entier
       le plus proche de x.
```

- **b)** Comment choisir n pour que l'on soit sûr à 95% de la valeur affichée ?
- **c)** Commenter le résultat obtenu.

FE 16 - Probabilités Solutions

Exercice 1

1. a) On a :

$$1 = C\sum_{j=0}^{\infty}\frac{1}{j^2+3j+2}\left(\sum_{i=0}^{\infty}e^{-i}\right) = \frac{C}{1-e^{-1}}\sum_{j=0}^{\infty}\left(\frac{1}{j+1}-\frac{1}{j+2}\right)$$

et comme $\sum_{j=0}^{n}\left(\frac{1}{j+1}-\frac{1}{j+2}\right) = 1-\frac{1}{n+2} \xrightarrow[n\to\infty]{} 1$,
il vient :

$$C = 1-e^{-1}$$

b) La loi de A est donnée par, pour tout $i \in \mathbb{N}$:

$$P(A=i) = \sum_{j=0}^{\infty}P(A=i\cap B=j) = e^{-i}(1-e^{-1})$$

Donc A suit la loi géométrique à valeur dans $\mathbb{N}$ de paramètre $p = 1-e^{-1}$, et :

$$\boxed{E(A) = \frac{1}{e-1}-1 \quad \text{et} \quad V(A) = \frac{e}{(e-1)^2}}$$

c) On a $B(\Omega) = \mathbb{N}$ et pour tout $j \in \mathbb{N}$:

$$P(B=j) = \frac{1}{j^2+3j+2}$$

ce qui entraîne que A et B sont indépendantes.

2. a) On a

$$\begin{aligned} P(Z=n) &= P(5A+7B=n) \\ &= \sum_{i=0}^{+\infty}P(5A+7B=n, A=i) \\ &= \sum_{i=0}^{+\infty}P(7B=n-5i, A=i) \\ &= \sum_{i=0}^{+\infty}P(7B=n-5i)P(A=i) \end{aligned}$$

Pour que le terme $P(7B=n-5i, A=i)$ soit non nul, il faut que

$$n-5i \geqslant 0$$

i.e.

$$i \leqslant \frac{n}{5}$$

$$P(Z=n) = \sum_{i=0}^{\lfloor \frac{n}{5} \rfloor}P(7B=n-5i)P(A=i)$$

somme pour laquelle les termes non nuls sont ceux tels que $n-5i$ sont multiples de 7.

```python
import numpy as np

def ProbaZ(n):
    '''Calcul P(Z=n) = somme de i entier positif inférieur ou égal à n/5 de P(B=(n-5i)/7, A=i) pour les (n-5i)/7 entiers.'''

    s=0  # initialisation de la somme
    i=0  # valeur de A
    max_des_i=n/5
    C=1-np.exp(-1)
    while i<=max_des_i:
        if int((n-5*i)/7)==(n-5*i)/7:   # si cette valeur est entière
            j=(n-5*i)/7
            s+=C*np.exp(-i)/(j**2+3*j+2)
        i+=1
    return(s)
```

b) On a $\boxed{P(Z=23)=0}$: en effet, il suffit d'essayer les valeurs possibles de B qui sont $0,\dots,3$, pour voir que $23-7B$ n'est pas un multiple de 5 ou si $23-5A$ n'est pas mutliple de 7.

Exercice 2

1. On propose le programme suivant :

```
def SimulY(n,p):
    X=randint(1,n) # Simule le numéro X du jeton
       tiré en première étape.
    Y=[0]*p      # Initialisation de la liste Y=[Y1
       ,...,Yp]
    for i in range(X): # on prend une bille après l
       'autre
        # choix du numéro de la boite pour la bille
           i
        b=randint(1,p)
        # les indices de python vont de 0 à p-1 au
           lieu de 1 à p. On rajoute une bille dans
           la boite d'indice b-1
        Y[b-1]+=1
    return(Y)
```

2. **a)** Étant donné un n-échantillon de Z, la loi faible des grands nombres permet d'affirmer que la moyenne empirique $Z^* = \dfrac{A_1 + \ldots + Z_N}{N}$ est une valeur approchée de $\mathbb{E}[Z]$.

b) On propose le programme suivant :

```
def ApprocheE(n,p):
    '''Donne la liste des valeurs approchées de E[
       Zi] par TCL.'''
    nb_simul=1000 # nb de simulations
    L=[0]*p # initialisatoin de la liste L des
       sommes pour chaque Yi
    for k in range(nb_simul):
        Y=SimulY(n,p)
        for i in range(p):
            L[i]+=Y[i]
    E=[l/nb_simul for l in L]
    return(E)
```

c) Il semblerait que

$$\mathbb{E}[Y_1] = \ldots = \mathbb{E}[Y_p]$$

ce qui semble cohérent puisque les boites sont choisies au hasard.

3. • La variable X suit la loi uniforme sur $[\![1,n]\!]$ et Y_i est à valeursdans $[\![0,n]\!]$.

• Soit donc $(k,\ell) \in [\![1,n]\!] \times [\![0,n]\!]$, on a

⋆ Si $\ell > k$, il est clair que $P((X=k) \cap (Y_i = \ell)) = 0$

⋆ Si $\ell \leqslant k$,

$$P((X=k) \cap (Y_i = \ell)) = P(X=k) P_{(X=k)}(Y_i = \ell)$$

Or

$$P(X=k) = \frac{1}{n} \quad \text{et} \quad P_{(X=k)}(Y_i = \ell) = \binom{k}{\ell} \left(\frac{1}{p}\right)^{\ell} \left(1 - \frac{1}{p}\right)^{k-\ell}$$

(succession de k expériences indépendantes à deux issues : tomber dans B_i ou tomber dans une autre boîte). Ainsi :

$$P((X=k) \cap (Y_i = \ell)) = \frac{1}{n} \binom{k}{\ell} \left(\frac{1}{p}\right)^{\ell} \left(1 - \frac{1}{p}\right)^{k-\ell}$$

4. **a)**

```
import matplotlib.pyplot as plt

def DessineE(p):
    ab=[k*10 for k in range(1,11)] # valeurs de n
       =10,20,...,100 en abscisse

    E1=[] # liste des E[Y1] pour n=10,20,...,100
    for k in range(10):
        E1+=[ApprocheE((k+1)*10,p)[1]]
    # Dessinons le graphes de EY1
    plt.clf()
    plt.plot(ab,E1)
```

b) Il semblerait qu'il existe $a, b \in \mathbb{R}$ tels que $\mathbb{E}[Y_1] = an + b$, où a et b dépendent de p.

c) Y_i est une variable fini, donc son espérance existe. On a

$$\begin{aligned} E(Y_i) &= \sum_{l=0}^{n} lP(Y_i = l) = \sum_{l=0}^{n} l \sum_{k=1}^{n} P(X=k) P_{X=k}(Y_i = l) \\ &= \sum_{k=1}^{n} \underbrace{P(X=k)}_{\frac{1}{n}} \underbrace{\sum_{l=0}^{n} lP_{X=k}(Y_i = l)}_{\text{espérance de } \mathcal{B}(k, \frac{1}{p}) = \frac{k}{p}} \\ &= \sum_{k=1}^{n} \frac{1}{n} \frac{k}{p} = \frac{1}{np} \frac{n(n+1)}{2} \end{aligned}$$

D'où

$$\boxed{E(Y_i) = \frac{n+1}{2p}}$$

5.
$$\begin{aligned} E(\tfrac{Y_i}{X}) &= \sum_{k=1}^{n}\sum_{l=0}^{n}\frac{l}{k}P(X=k,Y_i=l) \\ &= \sum_{k=1}^{n}\sum_{l=0}^{n}\frac{l}{k}P(X=k)P_{X=k}(Y_i=l) \\ &= \sum_{k=1}^{n}\underbrace{\frac{P(X=k)}{k}}_{\frac{1}{kn}}\underbrace{\sum_{l=0}^{n}lP_{X=k}(Y_i=l)}_{\frac{k}{p}} \\ &= \sum_{k=1}^{n}\frac{1}{np} \\ &= \frac{1}{p} \end{aligned}$$

D'où

$$\boxed{E(\tfrac{Y_i}{X})=\frac{1}{p}}$$

Exercice 3

1. On propose le programme suivant :

```
1  import scipy.stats as sc
2
3  def SimulVoyage(n):
4      '''Renvoie la liste des abscisses et ordonnées
           du point sur lequel se trouve la puce. n est
           le nombre de déplacements effectués'''
5      va=sc.norm()
6      # Abs est constitué de toutes les abscisses des
           points par lesquels on passe
7      # Ord est l'ensemble des ordonnées
           correspondantes
8      X=[va.rvs() for _ in range(n)]
9      Y=[va.rvs() for _ in range(n)]
10     Abs=[0]
11     Ord=[0]
12     for i in range(n):
13         Abs+=[Abs[-1]+X[i]]
14         Ord+=[Ord[-1]+Y[i]]
15     plt.close('all')
16     plt.plot(Abs,Ord)
17     plt.show()
```

2. a) Par télescopage $X_n = \sum_{k=1}^{n} A_k$. Les variables $A_i, i \in \mathbb{N}$ étant indépendantes et de même loi $\mathcal{N}(0,1)$, on sait que $\boxed{X_n \hookrightarrow \mathcal{N}(0,n).}$

b) M est dans le carré voulu si et seulement si $-1 \leqslant X_n \leqslant 1$ et $-1 \leqslant Y_n \leqslant 1$,
soit par indépendance des variables X_n et Y_n (les variables A_i sont aussi indépendantes des variables B_j) :

$$\begin{aligned} P(M_n \in C) &= P(-1 \leqslant X \leqslant 1)P(-1 \leqslant Y \leqslant 1) \\ &= \left(\Phi_{0,\sqrt{n}}(1) - \Phi_{0,\sqrt{n}}(-1)\right)^2 \\ &= \left(2\Phi_{0,\sqrt{n}}(1) - 1\right)^2 \end{aligned}$$

Or $\Phi_{0,\sqrt{n}}(1) = P(X_n \leqslant 1) = P\left(\frac{X_n}{\sqrt{n}} \leqslant \frac{1}{\sqrt{n}}\right) = \Phi_{0,1}\left(\frac{1}{\sqrt{n}}\right)$,
d'où :

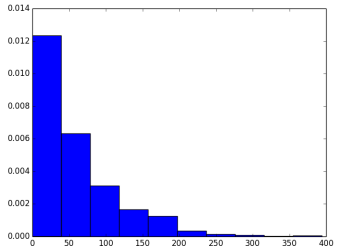
$$P(M_n \in C) = \left(2\Phi\left(\frac{1}{\sqrt{n}}\right) - 1\right)^2$$

3. La proportion empirique se rapproche de la théorique. C'est le TCL

4. a)

```
1  def histoM(n,N):
2      # On simule N fois les n voyages
3      VoyageX=[] # abscisse des points d'arrivée
4      VoyageY=[] # ordonnée des points d'arrivée
5      for i in range(N):
6          [Abs,Ord]=SimulVoyage(n) # totalité du
               voyage
7          # on s'intéresse à l'arrivée
8          Xn=Abs[-1]
9          Yn=Ord[-1]
10         VoyageX+=[Xn]
11         VoyageY+=[Yn]
12     Dn=[VoyageX[i]**2+VoyageY[i]**2 for i in range(
           N)]
13     clf()
14     plt.hist(Dn,normed=True)
```

b) Il semblerait que ceci se rapproche d'une loi exponentielle :



5. a) Si $Z \hookrightarrow \mathcal{E}(\lambda)$, on a

$$\mathbb{E}[Z] = \frac{1}{\lambda}.$$

On peut donc approcher le paramètre en donnant une valeur approchée de son espérance, qui s'obtient elle-même en effectuant le moyenne d'un n-échantillon pour n grand.

b)

```
def inv\_paramDn2(n):
    '''Donne une approximation du paramètre de Dn
       ^2.'''
    N=1000
    Distance_echantillon=[]
    # on fabrique autant d'échantillons que
       demandés :
    for _ in range(N):
        [Abs,Ord]=SimulVoyage(n) # totalité du
           voyage
        # on s'intéresse à l'arrivée
        Xn=Abs[-1]
        Yn=Ord[-1]
        Distance_echantillon+=[Xn**2+Yn**2]

    moyenne=np.mean(Distance_echantillon)
    # on sait que E[variable]=1/parametre
    return(1/moyenne)
```

c) On trouve une droite pour laquelle on conjecture que l'inverse du paramètre vaut $2n$ et donc que

$$D_n^2 \hookrightarrow \mathcal{E}\left(\frac{1}{2n}\right)$$

Exercice 4

1. a) Deux vecteurs u, v sont colinéaires si et seulement si le rang de la matrice constituée de u et v est égal à 2, d'où le programme ci-dessous :

```
import numpy as np

def coli(u,v):
    L=np.matrix([u, v])
    r=np.linalg.matrix_rank (L)
    return(not(r==2))
```

b) u est un vecteur propre de A si et seulement si u et Au sont colinéaires. On propose donc le programme suivant :

```
def vp(u):
    '''Détermine si u est vecteur propre de la
       matrice A précédemment construite.'''
    # On vérifie si u est le vecteur nul :
    if u[0]==0 and u[1]==0 and u[2]==0:
        return False
    # Si u n'est pas nul, on fait le produit de A
       par u
    else:
        V=np.array([[u[0]], [u[1]], [u[2]] ]) # V
           vecteur colonne de u
        A = np. array ( [ [3,1 ,1] , [1,0,1] ,
           [-3,0,-1] ] )
        Y=np.dot(A, V)            # Y est le vecteur
            AV
        w=[Y[0][0],Y[1][0],Y[2][0]] # w est la
           liste des coordonnées de Y
    return(coli(u,w))
```

2. a) • Méthode 1 :

On peut vérifier que $rg(A - \lambda Id) < 3$ pour $\lambda = -1, 1, 2$ afin de vérifier qu'elles sont bien valeurs propres, mais comme on cherche à connaître également les espaces propres, on peut également déterminer $E_\lambda = \ker(A - \lambda Id)$ pour chacune de ces trois valeurs. Si l'espace est non nul, cela confirmera que λ est valeur propre, et E_λ sera l'espace propre associé.

• Cas de $\lambda = -1$

$$\begin{aligned}(x,y,z) \in E_{-1} &\Leftrightarrow \begin{cases} 4x + y + z = 0 \\ x + y + z = 0 \\ -3x = 0 \end{cases} \\ &\Leftrightarrow \begin{cases} y + z = 0 \\ x = 0 \end{cases} \\ &\Leftrightarrow (x,y,z) \in Vect((0,1,-1))\end{aligned}$$

D'où -1 est valeur propre et son espace propre est

$$\boxed{E_{-1} = Vect((0,1,-1))}$$

- Cas de $\lambda = 1$

$$
\begin{aligned}
(x,y,z) \in E_1 &\Leftrightarrow \begin{cases} 2x+y+z=0 \\ x-y+z=0 \\ -3x-2z=0 \end{cases} \\
&\Leftrightarrow \begin{cases} 2x+y+z=0 \\ 3x+2z=0 \qquad L_2+L_1 \\ -3x-2z=0 \end{cases} \\
&\Leftrightarrow \begin{cases} 2x+y+z=0 \\ 3x+2z=0 \end{cases} \\
&\Leftrightarrow (x,y,z) \in Vect((-2,1,3))
\end{aligned}
$$

D'où 1 est valeur propre et son espace propre est

$$\boxed{E_1 = Vect((-2,1,3))}$$

- Cas de $\lambda = 2$

$$
\begin{aligned}
(x,y,z) \in E_2 &\Leftrightarrow \begin{cases} x+y+z=0 \\ x-2y+z=0 \\ -3x-3z=0 \end{cases} \\
&\Leftrightarrow \begin{cases} x+y+z=0 \\ 3x+3z=0 \qquad L_2+2L_1 \\ -x-z=0 \end{cases} \\
&\Leftrightarrow \begin{cases} y=0 \qquad L_1+L_3 \\ x+z=0 \end{cases} \\
&\Leftrightarrow (x,y,z) \in Vect((1,0,-1))
\end{aligned}
$$

D'où 2 est valeur propre et son espace propre est

$$\boxed{E_2 = Vect((1,0,-1))}$$

- Méthode 2 :

On peut afficher les valeurs approchées des valeurs propres

```
print('vap : ',np.linalg.eigvals(A))
```

On vérifie que ce sont bien les valeurs exactes avec le rang de (A-lambda Id)

```
Id=np. matrix( [ [1,0,0] , [0,1,0] , [0,0,1 ] ])

lamb=-1
r=np. linalg . matrix_rank (A-lamb*Id)
print(r)
```

Pour les vecteurs propres :

```
print('vep : ',np.linalg.eig(A)[1])
```

mais les résultats ne sont pas très évidents car ce sont des valeurs approchées, donc non utilisable en l'état. En revanche, on extrapole les valeurs en observant qu'il semblerait que :
- un vecteur propre de vap -1 s'écrit $[0, a, -a]$ d'où on propose un vecteur propre de -1 : $vm = [0, 1, -1]$
- un vecteur propre de vap 2 s'écrit $[a, 0, -a]$ d'où on propose $v2 = [1, 0, -1]$
- un vecteur propre de vap 1 s'écrit $[-2a, a, 3a]$ d'où on propose $v1 = [-2, 1, 3]$

puis on vérifie s'ils sont effectivement vep :

```
print('-1 :',vp(A,vm))
print('2 :',vp(A,v2))
print('1 :',vp(A,v1))
```

et on obtient `True` pour les 3, ce qui signifie qu'ils sont bien vecteurs propres.

b) On dispose de 3 valeur propres distinctes pour une matrice A de taille 3. A est donc diagonalisable.

3. **a)** Comme n est une moyenne, d'après le théorème central limite, on sait que, pour n grand, $\frac{M_n - \mathbb{E}[M_n]}{\sigma(M_n)}$ suit à peu près une loi normale centrée réduite. Or

$$\mathbb{E}[M_n] = \frac{\mathbb{E}[X_1] + \ldots + \mathbb{E}[X_n]}{n} = \frac{np}{n} = p$$

et, par indépendance des X_k,

$$\mathbb{V}(M_n) = \frac{\mathbb{V}[X_1] + \ldots + \mathbb{V}[X_n]}{n^2} = \frac{np(1-p)}{n^2} = \frac{p(1-p)}{n}.$$

On a donc bien

$$M_n^\star = \frac{M_n - \mathbb{E}[M_n]}{\sigma(M_n)}$$

D'où

$$\boxed{P([-\alpha < M_n^\star < \alpha]) = \Phi(\alpha) - \Phi(-\alpha) = 2\Phi(\alpha) - 1}$$

où Φ est la fonction de répartition de la loi normale centrée réduite.

b)

$$\begin{aligned}
P\left(M_n - \frac{1}{\sqrt{n}} \leqslant p \leqslant M_n + \frac{1}{\sqrt{n}}\right) &= P\left(-1 \leqslant \frac{p - M_n}{\frac{1}{\sqrt{n}}} \leqslant 1\right) \\
&= P\left(-1 \leqslant \frac{M_n - p}{\frac{1}{\sqrt{n}}} \leqslant 1\right) \\
&= P\left(-\frac{1}{\sqrt{p(1-p)}} \leqslant \frac{M_n - p}{\frac{\sqrt{p(1-p)}}{\sqrt{n}}} \leqslant \frac{1}{\sqrt{p(1-p)}}\right) \\
&= P\left(-\frac{1}{\sqrt{p(1-p)}} \leqslant M_n^\star \leqslant \frac{1}{\sqrt{p(1-p)}}\right)
\end{aligned}$$

Or, comme

$$\sqrt{p(1-p)} \leqslant \sqrt{\frac{1}{4}}$$

on trouve

$$\frac{1}{\sqrt{p(1-p)}} \geqslant 2$$

et donc

$$P\left(M_n - \frac{1}{\sqrt{n}} \leqslant p \leqslant M_n + \frac{1}{\sqrt{n}}\right) \geqslant \quad P\left(-2 \leqslant M_n^\star \leqslant 2\right) = 2\Phi(2) - 1$$

D'après l'énoncé, on sait que

$$\Phi(2) \simeq \Phi(1,96) \simeq 0,975$$

d'où

$$\boxed{P\left(M_n - \frac{1}{\sqrt{n}} \leqslant p \leqslant M_n + \frac{1}{\sqrt{n}}\right) \geqslant 0,95}$$

4. **a)** De manière générale, le nombre de vecteurs quelconques dont les coefficients sont compris entre -5 et 5 sont au nombre de

$$(5 - (-5) + 1)^3 = 11^3$$

(nombre de 3-liste d'éléments dans $[\![-5, 5]\!]$.)
Ainsi, si on note p la proportion de vecteurs propres parmi l'ensemble de toutes ces possibilités, on a

$$N_V = p * 11^3$$

Si on prend un vecteur au hasard parmi ceux dont dont les coefficients sont compris entre -5 et 5, la probabilité de tomber sur un vecteur propre est donc de p.
Or, d'après les résultats précédents, on sait que p peut être approché par M_n pour n grand.
Le programme donné calcul d'abord M_n puis effectue

$$M_n * 11^3,$$

ce qui permet d'approcher

$$p * 11^3 = N_V$$

En effet, `simul()` permet de simuler un vecteur au hasard dont les coefficients sont des entiers compris dans $[\![-5, 5]\!]$ et rend `True` si c'est un vecteur propre et `False` sinon.
Ainsi, la boucle qui suit permet, sur 10 000 simulations de vecteurs, de compter le nombre de vecteurs u qui sont vecteurs propres (`nb`). À la fin, `nb/n` est la proportion de vecteurs u qui sont vecteurs propres parmi tous ceux générés, c'est-à-dire

$$\frac{nb}{n} \simeq p$$

puis enfin

$$N_V \simeq \frac{nb}{n} * 11^3$$

b) En testant, on trouve des valeurs différentes à chaque essai : 21, 22,24,25, etc... La valeur de n doit être insuffisante.

La valeur affichée est `round (nb/n *11**3)` c'est-à-dire l'entier le plus proche de $11^3\frac{nb}{n} = 11^3 M_n$.
Si on veut être sûr que `round (nb/n *11**3)` $= N_V$ à 95%, il faudrait que

$$N_V \in \big[11^3 M_n - 0.5;\ 11^3 M_n + 0.5\big[$$

Ainsi, l'entier le plus proche de `nb/n *11**3` sera bien N_V (l'intervalle étant ouvert à droite car `round` arrondi les ",5' à l'entier supérieur.)
Or $N_V = 11^3 p$, d'où

$$\begin{aligned} N_V \in \big[11^3 M_n - 0.5;\ 11^3 M_n + 0.5\big[&\Leftrightarrow 11^3 M_n - 0.5 \leqslant 11^3 p < 11^3 M_n + 0.5 \\ &\Leftrightarrow M_n - \frac{0.5}{11^3} \leqslant p < M_n + \frac{0.5}{11^3} \end{aligned}$$

On sait que

$$P\left(M_n - \frac{1}{\sqrt{n}} \leqslant p \leqslant M_n + \frac{1}{\sqrt{n}}\right) \simeq 0,95$$

D'où, si

$$\frac{0.5}{11^3} \geqslant \frac{1}{\sqrt{n}}$$

on a

$$\begin{aligned} P(N_V = round(11^3 M_N)) &= P\left(M_n - \frac{0.5}{11^3} \leqslant p < M_n + \frac{0.5}{11^3}\right) \\ &\geqslant P\left(M_n - \frac{1}{\sqrt{n}} \leqslant p \leqslant M_n + \frac{1}{\sqrt{n}}\right) \simeq 0,95 \end{aligned}$$

Or

$$\frac{0.5}{11^3} \geqslant \frac{1}{\sqrt{n}} \quad \Leftrightarrow \quad \sqrt{n} \geqslant 2.11^3 \quad \Leftrightarrow \quad n \geqslant 4.11^6 = 7\,086\,244$$

En conclusion, pour être certain de la valeur affiché, il faudrait prendre

$$\boxed{n = 7\,086\,244}$$

Après un peu d'attente (!!), la valeur de N_V ainsi obtenue est

$$\boxed{N_V = 22}$$

c) On constate qu'ici "n" grand prend toute sa signification !

Un même calcul aurait pu être effectué à la main. En effet, il suffit de compter le nombre de vecteurs propres parmi les espaces propres :

$$E_{-1} = Vect((0,1,-1)) \quad E_1 = Vect((-2,1,3)) \quad E_2 = Vect((1,0,-1))$$

Les trois vecteurs formant une base, on est certain de ne pas trouver de vecteurs communs à ces trois espaces, mis à part le vecteur nul. L'ensemble des vecteurs cherchés est donc

$$(E_{-1} \cup E_1 \cup E_2) \cap [\![-5,5]\!]^3$$

Dont le cardinal peut donc se calculer par

$$\big(card\big(E_{-1} \cap [\![-5,5]\!]^3\big) - 1\big) + \big(card\big(E_1 \cap [\![-5,5]\!]^3\big) - 1\big) + \big(card\big(E_2 \cap [\![-5,5]\!]^3\big) - 1\big) + 1$$

- Cas de $E_{-1} \cap [\![-5,5]\!]^3$

$$E_{-1} \cap [\![-5,5]\!]^3 = \left\{ \begin{pmatrix} 0 \\ \alpha \\ -\alpha \end{pmatrix} \mid \alpha \in [\![-5,5]\!] \right\}$$

Donc

$$card\big(E_{-1} \cap [\![-5,5]\!]^3\big) = 11$$

- Cas de $E_2 \cap [\![-5,5]\!]^3$

$$E_2 \cap [\![-5,5]\!]^3 = \left\{ \begin{pmatrix} \alpha \\ 0 \\ -\alpha \end{pmatrix} \mid \alpha \in [\![-5,5]\!] \right\}$$

Donc

$$card\big(E_2 \cap [\![-5,5]\!]^3\big) = 11$$

- Cas de $E_{-1} \cap [\![-5,5]\!]^3$

$$E_{-1} \cap [\![-5,5]\!]^3 = \left\{ \begin{pmatrix} -2\alpha \\ \alpha \\ 3\alpha \end{pmatrix} \mid 3\alpha \in [\![-5,5]\!] \right\}$$

Or,

$$-5 \leqslant 3\alpha \leqslant 5 \quad \Leftrightarrow \quad -\frac{5}{3} \leqslant \alpha \leqslant \frac{5}{3}$$

Les seules possibilités sont

$$\alpha = -1, 0, 1$$

Donc

$$card\big(E_2 \cap [\![-5,5]\!]^3\big) = 3$$

Autrement dit, le cardinal cherché est donc

$$(11-1)+(3-1)+(11-1) = 22$$