Exercice 1

Soit ρ un endomorphisme de $E = \mathbb{R}^3$ dont la matrice dans une base orthonormée directe $(\vec{\imath}, \vec{\jmath}, \vec{k})$ est

$$M = \frac{1}{9}\begin{bmatrix} 8 & 1 & -4 \\ -4 & 4 & -7 \\ 1 & 8 & 4 \end{bmatrix}.$$

(1) **a.** Écrire en langage `Python` une fonction permettant de calculer le produit scalaire de deux vecteurs de $\mathbb{R}^3$ appelés a et b à l'aide d'une boucle. La fonction renvoyant un scalaire pourra avoir pour entête :
`def monProd(a,b)`

b. Comparer les résultats avec le produit scalaire fourni par la bibliothèque numpy :
`numpy.vdot(a,b)`
sur les vecteurs $a = \frac{1}{9}\begin{pmatrix} 8 \\ -4 \\ 1 \end{pmatrix}$ et $b = \frac{1}{9}\begin{pmatrix} 1 \\ 4 \\ 8 \end{pmatrix}$

(2) Montrer que ρ est un automorphisme orthogonal direct.

(3) Montrer que $\omega = \begin{pmatrix} 3 \\ -1 \\ -1 \end{pmatrix}$ est invariant par ρ.
En déduire la nature de l'isométrie.

(4) Montrer que le vecteur $u = \frac{1}{\sqrt{10}}\begin{pmatrix} 1 \\ 3 \\ 0 \end{pmatrix}$ est orthogonal à ω.

(5) Donner les éléments caractéristiques de cette transformation.

Solution de l'exercice 1

(1) Soit ρ un endomorphisme de $E = \mathbb{R}^3$ dont la matrice dans une base orthonormée directe (i, j, k) est

$$M = \frac{1}{9}\begin{bmatrix} 8 & 1 & -4 \\ -4 & 4 & -7 \\ 1 & 8 & 4 \end{bmatrix}$$

(2) **a.**
```
def monProd(a,b):
    #a vecteur reel de dimension 3
    #b vecteur reel de dimension 3
    #renvoie le produit scalaire des vecteurs a et b
    scal=0
    for i in range(0,3):
        scal=scal+a[i]*b[i]
    return scal
```

b.
```
#console
>>> a=1/9*numpy.array([8,-4,1])
>>> b=1/9*numpy.array([1,4,8])

>>> monProd(a,b)
>>> numpy.vdot(a,b)
```

Les vecteurs sont orthogonaux.

(3) ρ est un endomorphisme orthogonal puisque les colonnes de la matrice associée forment une base orthonormée, et il est direct puisque de déterminant 1.

(4) $\omega = (3i - j - k)$ est invariant par ρ par calcul.
Dès la question précédente, étant donné que la matrice associée n'est pas l'identité, la nature de l'isométrie était une rotation, cette rotation s'effectue autour du vecteur ω.

(5) Le vecteur $u = \dfrac{1}{\sqrt{10}}(i + 3j)$ est orthogonal à ω en effectuant le produit scalaire.

(6) En calculant $u \cdot \rho(u) = \cos(\theta)$ et le produit mixte de u, $\rho(u)$ et w : $\det(u, \rho(u), w) = \|u\|^2\|w\| \sin(\theta) = \dfrac{5\sqrt{11}}{18}$. La matrice dans une base correctement choisie est

$$M = \frac{1}{9}\begin{bmatrix} \frac{7}{18} & \frac{5\sqrt{11}}{18} & 0 \\ -\frac{5\sqrt{11}}{18} & \frac{7}{18} & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Exercice 2

On considère la surface (S) d'équation cartésienne :

$$x^2 + y^2 + z^2 + 2xyz - 1 = 0.$$

(1) Écrire en langage `Python` une fonction permettant de renvoyer vrai (True) si le point M passé en paramètre appartient à la surface (S). La fonction renvoyant un booléen pourra avoir pour entête :

```
def appartient(M)
```

où `M` peut être vu comme une liste ou comme un triplet ou sous la forme de son choix, le choix est laissé au candidat.

(2) Le point de coordonnées $(0, 0, 0)$ appartient-il à (S) ?

(3) On pose $f(x, y, z) = x^2 + y^2 + z^2 + 2xyz - 1$ avec x, y et z des réels.

a. Justifier que f est de classe C^∞ sur $\mathbb{R}^3$.

b. Calculer le gradient de f au point (x_0, y_0, z_0).

(4) Déterminer l'ensemble des points non réguliers de (S).

(5) Donner l'équation cartésienne du plan tangent à (S) à un point régulier de (S) dont les coordonnées seront notées (x_0, y_0, z_0).

(6) Déterminer l'intersection de la surface (S) avec le plan d'équation $x = a$ dans le cas $a = 1$ ou $a = -1$.

(7) Déduire à l'aide de la question précédente que (S) contient exactement 6 droites.

Solution de l'exercice 2

On considère la surface (S) d'équation cartésienne :

$$x^2 + y^2 + z^2 + 2xyz - 1 = 0$$

(1)
```
def appartient(M):
    # discussion sur le test d'egalite avec les flottants
    # fonctionne avec M sous forme de liste ou de array Numpy
    # M point a des coordonnees reelles en dimension 3
    # renvoie un booleen indiquant l'appartenance du point  a la surface (S)
    # Si l'on souhaite verifier la valeur de M  :    print(M)
        reponse=(M[0]**2+M[1]**2+M[2]**2+2*M[0]*M[1]*M[2]-1==0)
        return reponse
```

(2)
```
>>> print(appartient([0,0,0]))
False
```

Le point de coordonnées $(0, 0, 0)$ n'appartient pas à la surface (S).

(3) On pose $f(x, y, z) = x^2 + y^2 + z^2 + 2xyz - 1$ avec x, y et z des réels.

a. f est de classe $C^{+\infty}$ sur $\mathbb{R}^3$ comme fonction polynomiale.

b. Gradient de f au point (x_0, y_0, z_0) : $\boxed{\overrightarrow{grad} f(x, y, z) = {}^t(2x + 2yz, 2y + 2xz, 2z + 2xy).}$

(4) Ensemble des points non réguliers de (S) : $\boxed{\begin{cases} y = & \pm 1 \\ x = & \pm 1 \\ z = & -xy \end{cases}}$ Quatre points annulent le gradient et ces quatre points appartiennent à (S), leurs coordonnées sont $\boxed{(-1, -1, -1), (-1, 1, 1), (1, -1, 1), (1, 1, -1).}$

(5) Une équation cartésienne du plan tangent à (S) à un point régulier de (S) dont les coordonnées seront notées (x_0, y_0, z_0) :

$$(x_0 + y_0z_0)X + (y_0 + x_0z_0)Y + (z_0 + y_0x_0)Z - x_0y_0z_0 - 1 = 0$$

(6) L'intersection de la surface (S) avec le plan d'équation $x = a$ dans le cas $a = 1$ ou $a = -1$.
Si $a = 1$ ou $a = -1$, l'intersection est une droite d'équation : $\begin{cases} x = & a \\ y + az = & 0 \end{cases}$

(7) En partant d'une forme où x n'est pas constant $\begin{cases} x &= t \\ y &= a + bt \\ z &= c + dt \end{cases}$
Injecter dans (S) pour obtenir un polynôme de degré 3 en t et on obtient le système

$$\begin{cases} 2bd & = 0 \\ b^2 + d^2 + 1 + 2ad + 2cb & = 0 \\ 2ab + 2bd + 2ac & = 0 \\ a^2 + c^2 - 1 & = 0 \end{cases}$$

Alors on a $b = 0$ ou $d = 0$ par conséquent, z ou y sont constants et alors on peut conclure.
Dans les cas où $a \neq 1$ et $a \neq -1$, il s'agit d'équation polynomiale de degré 2, l'intersection est alors une ellipse pour $a \in]-1, 1[$ et une hyperbole pour $a < -1$ ou $a > 1$. Ainsi ce que l'on a fait pour $x = a$, peut être fait de la même manière pour $y = a$ et $z = a$ et il vient que (S) contient exactement 6 droites.

Exercice 3

Soit f la fonction $2\pi-$périodique définie par

$$\forall x \in [-\pi, \pi], \quad f(x) = \frac{\pi^2}{12} - \frac{x^2}{4}.$$

(1) Tracer à la main en justifiant la courbe représentative de la fonction f sur l'intervalle $[-2\pi, 2\pi]$.

(2)
- **a.** Calculer, pour tout entier naturel n, les coefficients réels de Fourier b_n (associés aux fonctions sinus) de la fonction f.
- **b.** Calculer le coefficient de Fourier a_0.
- **c.** Calculer les coefficients réels de Fourier a_n (associés aux fonctions cosinus) pour n un entier non nul à l'aide de deux intégrations par parties.

(3) Montrer que pour tout x de l'intervalle $[-\pi, \pi]$,

$$\frac{\pi^2}{12} - \frac{x^2}{4} = \sum_{n=1}^{+\infty} (-1)^{n+1} \frac{\cos(nx)}{n^2}.$$

(4) Déduire de la question précédente que

$$\sum_{n=1}^{+\infty} \frac{(-1)^{n+1}}{n^2} = \frac{\pi^2}{12}.$$

(5)
- **a.** Illustrer à l'aide du langage `Python` le résultat précédent en écrivant une fonction prenant en paramètre un entier naturel N non nul et renvoyant la valeur de la somme partielle $\sum_{k=1}^{N} \frac{(-1)^{k+1}}{k^2}$.

  La fonction pourra avoir comme entête :

  ```
  def sommeP(N)
  ```

  Pour calculer $(-1)^k$*, on prendra garde d'utiliser la bibliothèque* `math` *et*

  ```
  math.pow(-1,k)
  ```

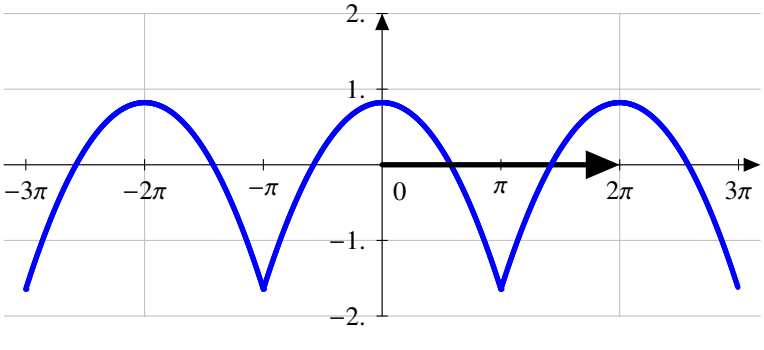
- **b.** Calculer les sommes partielles au rang 10, 100 et 1000 et comparer avec $\frac{\pi^2}{12}$.

  On rappelle que la valeur π *est obtenue à l'aide de la commande*

  ```
  math.pi
  ```

Solution de l'exercice 3

(1) Tracer la courbe représentative de la fonction f sur l'intervalle $[-2\pi, 2\pi]$.



(2) **a.** Pour tout entier naturel n les coefficients de Fourier b_n de la fonction f sont nuls car la fonction f est paire.

b. $\boxed{a_0 = 0}$.

c. Les coefficients de Fourier a_n pour n un entier non nul à l'aide de deux intégrations par parties :

$$\boxed{a_n(f) = \frac{2}{\pi}\int_0^{\pi} f(t)\cos(nt)dt = \frac{(-1)^{n+1}}{n^2}.}$$

(3) Pour tout x de l'intervalle $[-\pi, \pi]$, f étant 2π-périodique, continue et C^1 par morceaux sur $\mathbb{R}$:

$$\boxed{\frac{\pi^2}{12} - \frac{x^2}{4} = \sum_{n=1}^{+\infty}(-1)^{n+1}\frac{\cos(nx)}{n^2}.}$$

(4) Avec la question précédente en prenant $x = 0$

$$\boxed{\sum_{n=1}^{+\infty}\frac{(-1)^{n+1}}{n^2} = \frac{\pi^2}{12}.}$$

(5) **a.**

```
import math

def sommeP(N):
#N entier indiquant le rang de la somme partielle
#renvoie la valeur de la somme partielle au rang N
    somme=0
    for k in range(1,N+1):
            somme=somme+math.pow(-1,k+1)/math.pow(k,2)
    return somme
```

b.

```
>>> math.pi # pour comparer et afficher la valeur de Pi renvoyee par
3.141592653589793

>>> abs(sommeP(10)-(math.pi)*(math.pi)/12)  # ecart entre S_10 et Pi
-0.004504857813128038
>>> abs(sommeP(100)-(math.pi)*(math.pi)/12)  # ecart entre S_100 et P
-4.9500049984607664e-05
>>> abs(sommeP(1000)-(math.pi)*(math.pi)/12)  # ecart entre S_1000 et
-4.995000018048756e-07
```

Exercice 4

Pour $n \in \mathbb{N}^*$, on considère la somme :

$$S_n = \sum_{k=1}^{n} \frac{1}{\sqrt{n^2 + 2kn}}.$$

(1) Écrire, en `Python`, une fonction `somme(n)` prenant en paramètre un entier naturel non nul `n`, et renvoyant la valeur de S_n.

(2) Donner la valeur de S_{10}, de S_{100} et de S_{1000}.

(3) Convergence de $(S_n)_{n\geq 1}$.

a. Calculer la valeur de $\displaystyle\int_0^1 \frac{1}{\sqrt{1+2x}}\,dx$.

b. Montrer que $(S_n)_{n\geq 1}$ converge vers $\sqrt{3}-1$.

Solution de l'exercice 4

(1)

```
import math

def somme(n):
  s=0
  for k in range(1,n+1):
    s+=1/math.sqrt(n**2+2*k*n)
  return s
```

(2) En faisant appel à la fonction précédente on trouve :

```
>>> somme(10)
0.7115892706266376

>>> somme(100)
0.7299442882935635

>>> somme(1000)
0.731839549999278
```

(3) **a.** On a :

$$\begin{aligned}\int_0^1 \frac{1}{\sqrt{1+2x}}\,dx &= \left[\sqrt{1+2x}\right]_0^1 \\ &= \sqrt{3} - \sqrt{1}.\end{aligned}$$

On trouve donc $\boxed{\displaystyle\int_0^1 \frac{1}{\sqrt{1+2x}}\,dx = \sqrt{3}-1}$.

b. Pour $n \in \mathbb{N}^*$, on a :

$$\begin{aligned}S_n = \sum_{k=1}^{n} \frac{1}{\sqrt{n^2+2kn}} &= \sum_{k=1}^{n} \frac{1}{n\sqrt{1+\frac{2k}{n}}} \\ &= \frac{1}{n}\sum_{k=1}^{n} \frac{1}{\sqrt{1+\frac{2k}{n}}}.\end{aligned}$$

On reconnaît une somme de Riemann associée à la fonction $f : x \longmapsto \dfrac{1}{1+2x}$. Cette fonction f étant continue sur $[0,1]$, on a par le théorème de convergence des sommes de Riemann :

$$\lim_{n\to+\infty}(S_n) = \int_0^1 f(x)\,dx$$

et d'après la question précédente on en déduit que $\boxed{\lim_{n\to+\infty}(S_n) = \sqrt{3}-1}$.

Exercice 5

Pour $a \in \mathbb{R}$, on considère la matrice :

$$A(a) = \begin{pmatrix} 1 & a-2 & a & 1 \\ a & -1 & 1 & a \\ 0 & 0 & -a & 1 \\ 0 & 0 & -1 & 0 \end{pmatrix}.$$

(1) Déterminer le polynôme caractéristique de $A(a)$.

(2) Soit $B \in \mathcal{M}_{4,1}(\mathbb{R})$. Déterminer pour quelle(s) valeur(s) de a le système linéaire $A(a)X = B$ admet une unique solution.

(3) On s'intéresse au système différentiel suivant, où les inconnues sont x, y, z, w fonctions de t :

$$(S) : \begin{cases} x' &= x + 0,5y + 2,5z + w \\ y' &= 2,5x - y + z + 2,5w \\ z' &= -2,5z + w \\ w' &= -z \end{cases}$$

a. On note A la matrice associée au système (S). Montrer qu'il existe deux matrices P et D avec P inversible et D diagonale, telles que $A = PDP^{-1}$. On déterminera ces matrices à l'aide de `Python` . *On pourra utiliser les deux fonctions suivantes de la bibliothèque* `numpy` *:*

```
numpy.linalg.eig          numpy.diag
```

b. Déterminer à la main la solution Y_0 du système différentiel $Y' = DY$ telle que $Y_0(0) = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}$.

c. Écrire, en Python, une fonction `solution(t)` prenant en paramètre un nombre réel `t`, et renvoyant la valeur de la solution X de (S) en `t` tel que $P^{-1}X = Y_0$.

Solution de l'exercice 5

(1) On calcule le polynôme caractéristique en développant deux fois de suite par rapport à la dernière ligne :

$$\begin{aligned} \det(\lambda I_4 - A) &= \begin{vmatrix} \lambda-1 & 2-a & -1 \\ -a & \lambda+1 & -a \\ 0 & 0 & -1 \end{vmatrix} + \lambda \begin{vmatrix} \lambda-1 & 2-a & -a \\ -a & \lambda+1 & -1 \\ 0 & 0 & \lambda+a \end{vmatrix} \\ &= \begin{vmatrix} \lambda-1 & 2-a \\ -a & \lambda+1 \end{vmatrix} + \lambda(\lambda+a) \begin{vmatrix} \lambda-1 & 2-a \\ -a & \lambda+1 \end{vmatrix} \\ &= -\left(\lambda^2 - (1-a)^2\right)\left(1 + a\lambda + \lambda^2\right). \end{aligned}$$

Ainsi $\boxed{\det(\lambda I_4 - A) = -\left(\lambda^2 - (1-a)^2\right)\left(1 + a\lambda + \lambda^2\right)}$.

(2) La matrice $A(a)$ étant une matrice carrée, on a :

$$\begin{aligned} \text{Le système linéaire } A(a)X = B \text{ admet une unique solution} &\iff \text{la matrice } A(a) \text{ est inversible} \\ &\iff 0 \text{ n'est pas valeur propre de } A(a) \\ &\iff \det(-A) \neq 0 \\ &\iff a \neq 1. \end{aligned}$$

On conclut $\boxed{A(a)X = B \text{ admet une unique solution si et seulement si } a \neq 1}$.

(3) **a.** On remarque que $A = A(2.5)$. Sur Python, on peut écrire :

```
>>> import numpy as np  # les deux bibliotheques necessaires
>>> import numpy.linalg as alg
>>> a=2.5
>>> A=np.array([[1,a-2,a,1],[a,-1,1,a],[0,0,-a,1],[0,0,-1,0]])
>>> L=alg.eig(A)
>>> L
(array([ 1.5, -1.5, -2. , -0.5]), array([[ 0.70710678, -0.19611614,
0.66683134,  0.72370779],
       [ 0.70710678,  0.98058068, -0.26673253,  0.31016048],
       [ 0.        ,  0.        , -0.62237591, -0.27569821],
       [ 0.        ,  0.        , -0.31118796, -0.55139641]]))
```

Cela montre que $A(2.5)$ admet quatre valeurs propres distinctes : $1.5, -1.5, -2, -0.5$. On en déduit que $A(2.5)$ est diagonalisable sur $\mathbb{R}$. Pour trouver P et D on peut écrire à la suite :

```
>>> P=L[1]
>>> D=np.diag(L[0])
```

b. Le système $Y' = DY$ se résout "à la main" et on trouve, pour tout $t \in \mathbb{R}$:

$$Y(t) = \begin{pmatrix} C_1 e^{1.5t} \\ C_2 e^{-1.5t} \\ C_3 e^{-2t} \\ C_4 e^{-0.5t} \end{pmatrix}$$

où C_1, C_2, C_3 et C_4 sont des constantes réelles. La contrainte $Y(0) = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}$ impose $C_1 = C_2 = C_3 = C_4 = 1$. Finalement $\boxed{Y_0(t) = \begin{pmatrix} e^{1.5t} \\ e^{-1.5t} \\ e^{-2t} \\ e^{-0.5t} \end{pmatrix}}$.

c. On sait que $X = PY$.

```
def x(t): # on definit deja les fonctions solutions de Y'=DY
    return np.exp(1.5*t)

def y(t):
    return np.exp(-1.5*t)

def z(t):
    return np.exp(-2*t)

def w(t):
    return np.exp(-0.5*t)

def solution(t):
    return P.dot([[x(t)],[y(t)],[z(t)],[w(t)]])
```

Exercice 6

Le plan $\mathcal{P}$ est rapporté à un repère orthonormé direct $\mathcal{R} = (O, \vec{\imath}, \vec{\jmath})$.
On considère la courbe Γ de représentation paramétrique

$$\begin{cases} x(t) = \dfrac{2t-1}{t^2-1} \\ y(t) = \dfrac{t^2+3}{t-1} \end{cases}$$

On pose $M(t)$ le point de coordonnées $(x(t), y(t))$.

(1) Donner les ensembles de définition des fonctions x et y.

(2) Étudier la courbe paramétrée Γ.

(3) Justifier l'existence d'asymptotes verticales et horizontales et préciser leur équation respective.

(4) **a.** Définir en `Python` les fonctions `abscisse(t)` et `ordonnee(t)` où t est le paramètre et renvoyant respectivement la valeur de $x(t)$ et de $y(t)$.

b. Tracer la courbe $t \mapsto \dfrac{y(t)}{x(t)}$ sur l'intervalle $I_1 = \left[\dfrac{2}{3}; 1\right[$, la commande `linspace` pourra être utilisée.

c. Tracer la courbe $t \mapsto \dfrac{y(t)}{x(t)}$ sur l'intervalle $I_2 = \left]1; \dfrac{4}{3}\right]$, la commande `linspace` pourra être utilisée.

d. Que constate-t-on ? Justifier votre réponse par le calcul.

e. Tracer la courbe $t \mapsto y(t) - 8x(t)$ sur les intervalles I_1 et I_2.

f. Que constate-t-on ? En déduire l'équation d'une droite asymptote à la courbe. Vous démontrerez votre conjecture.

Solution de l'exercice 6

(1) La fonctions x est définie sur l'ensemble des réels privé de -1 et 1.
La fonctions y est définie sur l'ensemble des réels privé de 1.

(2) Étude de la courbe paramétrée Γ.

Pour tout t réel différent de 1 et -1 : $x'(t) = \dfrac{-2t^2+2t-2}{(t^2-1)^2}$.

Pour tout t réel différent de 1 : $y'(t) = \dfrac{(t+1)(t-3)}{(t-1)^2}$.

t	$-\infty$		-1		1		3		$+\infty$
$x'(t)$		$-$		$-$	‖	$-$			
$x(t)$	0	↘	$+\infty$ ‖ $-\infty$	↘	$+\infty$ ‖ $-\infty$	↘	$\frac{5}{8}$	↘	0

t	$-\infty$		-1		1		3		$+\infty$
$y'(t)$		$+$	0	$-$	‖	$-$	0	$+$	
$y(t)$	$-\infty$	↗	-2	↘	$+\infty$ ‖ $-\infty$	↘	6	↗	$+\infty$

(3) $\lim\limits_{t\to\pm\infty} x(t) = 0$ et $\lim\limits_{t\to\pm\infty} y(t) = \pm\infty$ donc $\boxed{x = 0 \text{ est asymptote.}}$ $\lim\limits_{t\to-1} x(t) = \pm\infty$ et $\lim\limits_{t\to\pm\infty} y(t) = -2$ donc $\boxed{y = -2 \text{ est asymptote.}}$

(4) **a.**

```
def x(t):
    return (2*t-1)/(t**2-1)

def y(t):
    return (t**2+3)/(t-1)
```

b.

```
import numpy
import matplotlib.pyplot as plt

epsilon=1E-6
t1=numpy.linspace(2/3,1-epsilon,1000)
plt.plot(t1,y(t1)/x(t1),color='r')
plt.show()
```

c.

```
t2=numpy.linspace(1+epsilon,4/3,1000)
plt.plot(t2,y(t2)/x(t2),color='g')
plt.show()
```

d. La fonction semble prolongeable par continuité en 1 et $\boxed{\lim_{t\to 1} \frac{y(t)}{x(t)} = \lim_{t\to 1} \frac{(t^2+3)(t+1)}{2t-1} = 8.}$

e.

```
t=numpy.linspace(2/3,4/3,1000)
plt.plot(t,y(t)-8*x(t))
plt.show()
```

f. La fonction semble prolongeable par continuité en 1 et $\boxed{\lim_{t\to 1} y(t) - 8x(t) = -4.}$

$y = 8x - 4$ est asymptote à la courbe.

Exercice 7

(1) Rappeler la dimension et la base canonique de $\mathbb{R}_2[X]$.

(2) On note : $P_0 = 1$; $P_1 = X + 1$ et $P_2 = (X + 2)^2$. Vérifier que $\mathcal{B}$ la famille constituée de ces trois polynômes est une base de $\mathbb{R}_2[X]$.

(3) Pour P et Q dans $\mathbb{R}_2[X]$, on définit $\varphi(P, Q)$ comme étant $\sum_{k=-1}^{1} P(k) \times Q(k)$.
Montrer que l'application φ est un produit scalaire sur $\mathbb{R}_2[X]$.

(4) Écrire en `Python` une fonction renvoyant la valeur de $\varphi(P_i, P_j)$ en fonction des paramètres entiers i et j.

(5) $\mathcal{B}$ est-elle une base orthonormée (relativement à φ) ?

(6) À partir de $\mathcal{B}$, construire une base orthonormée de $(\mathbb{R}_2[X], \varphi)$.

(7) On note H le sous-espace vectoriel de $\mathbb{R}_2[X]$ engendré par P_1 et P_2.
Déterminer la distance du polynôme X^2 à H puis la distance de 1 à H.

Solution de l'exercice 7

(1) $\mathbb{R}_2[X]$ est un espace vectoriel de dimension 3 et $\boxed{\text{sa base canonique est } (1; X; X^2).}$

(2) Le cardinal de la famille $\mathcal{B}$ est égal à la dimension de $\mathbb{R}_2[X]$, il suffit de vérifier que cette famille est libre pour obtenir qu'il s'agit d'une base de $\mathbb{R}_2[X]$.
Elle l'est car de degrés échelonnés. Donc $\mathcal{B}$ est libre dans $\mathbb{R}_2[X]$.
Donc $\boxed{\mathcal{B} \text{ est une base de } \mathbb{R}_2[X].}$

(3) φ est une application de $\mathbb{R}_2[X] \times \mathbb{R}_2[X]$ à valeur dans $\mathbb{R}$.
La commutativité de la multiplication dans $\mathbb{R}$ entraîne que φ est symétrique.
Pour P, Q_1 et Q_2 dans $\mathbb{R}_2[X]$, λ et μ dans $\mathbb{R}$, on a :

$$\varphi(P, \lambda Q_1+\mu Q_2) = \sum_{k=-1}^{1} P(k)\times(\lambda Q_1+\mu Q_2)(k) = \lambda \sum_{k=-1}^{1} P(k)\times Q_1(k)+\mu \sum_{k=-1}^{1} P(k)\times Q_2(k) = \lambda\varphi(P, Q_1)+\mu\varphi(P, Q_2).$$

Donc φ est symétrique par rapport à la seconde variable. Par symétrie, elle est donc bilinéaire.

Pour P dans $\mathbb{R}_2[X]$, les évaluations $P(k)$ sont réelles et leurs carrés sont dans $[0; +\infty[$.
Donc $\varphi(P, P) \geq 0$.

Soit P dans $\mathbb{R}_2[X]$ tel que $\varphi(P, P) = 0$. Par le même argument de positivité des $(P(k))^2$, on en déduit que pour k dans $[\![-1; 1]\!]$, $(P(k))^2$ est nul.
Donc le polynôme P possède au moins 3 racines réelles (-1, 0 et 1). P ayant un degré inférieur ou égal à trois, on en déduit que P est le polynôme nul.
$\boxed{\varphi \text{ étant une forme bilinéaire symétrique définie positive, c'est un produit scalaire sur } \mathbb{R}_2[X].}$

(4) On peut écrire en `Python` pour fonction renvoyant la valeur de $\varphi(P_i, P_j)$ ceci :

```
def notreps(i, j):
    res = 0
    for k in range(-1, 2):
        res = res + (k + i) ** i * (k + j) ** j
    return res
```

(5) Par calcul, on observe que $\varphi(P_0, P_0)$ vaut 3 et non 1. Donc la base $\mathcal{B}$ n'est pas orthonormée relativement au produit scalaire φ.

(6) Pour construire une base orthonormée de $(\mathbb{R}_2[X], \varphi)$ à partir de $\mathcal{B}$, on peut utiliser le procédé d'orthonormalisation de Gram-Schmidt. On obtient les quatres nouveaux $\tilde{P}_k$ suivants :

— $\boxed{\tilde{P}_0 = \dfrac{1}{\sqrt{3}} = \dfrac{\sqrt{3}}{3}}$

— $\boxed{\tilde{P}_1 = \dfrac{1}{\sqrt{2}}X = \dfrac{\sqrt{2}}{2}X}$

— $\boxed{\tilde{P}_2 = \dfrac{\sqrt{3}}{\sqrt{2}}\left(X^2 - \dfrac{2}{3}\right) = \dfrac{\sqrt{6}}{6}\left(3X^2 - 2\right)}$

(7) On observe la relation de liaison : $P_2 = X^2 + 4P_1$. Donc X^2 est dans H et la distance de X^2 à H est nulle.

Le polynôme 1 n'est pas dans H, on recherche une base orthonormée de H.

À l'aide des calculs fournis par 4), on obtient une base constituée de

$Q_1 = \dfrac{1}{\sqrt{5}}P_1$ et $Q_2 = \dfrac{\sqrt{5}}{\sqrt{6}}\left(X^2 - \dfrac{2}{5}X - \dfrac{2}{5}\right)$.

En notant P_H le projeté orthogonal de P_0 sur H, la distance de P_0 à H vaut $\sqrt{\varphi(P_0 - P_H, P_0 - P_H)}$.

Ici, on a : $\boxed{P_H = \varphi(P_0, Q_1)Q_1 + \varphi(P_0, Q_2)Q_2.}$

Donc : $P_H = \dfrac{2}{3}X^2 - \dfrac{1}{3}X - \dfrac{1}{3}$ et $\boxed{\text{la distance de } P_0 \text{ à } H \text{ vaut } \dfrac{\sqrt{6}}{3}.}$

Exercice 8

Un message doit transiter par un certain nombre de personnes. Chaque personne est susceptible soit de transmettre le message fidèlement à la suivante, soit de mentir et de lui transmettre son exact contraire. La probabilité qu'une personne mente est notée $p \in]0,1[$. On suppose que les personnes ne se concertent pas entre elles. On note p_n la probabilité que le message transmis par la n^{e} personne soit identique au message initial. On conviendra que $p_0 = 1$.

1. (a) Proposer une fonction `Python` d'en-tête `chaine(n,p)` qui simule la réalisation de n transmissions, n étant un entier naturel non nul fourni en paramètre et p la probabilité définie dans l'énoncé. Elle renverra 1 si le message transmis par la n^{e} personne est identique au message initial et 0 sinon.

   (b) On répète N chaînes toutes de mêmes paramètres n et p. À l'aide de la fonction précédente, proposer une fonction d'en-tête `simul(N,n,p)` qui renvoie la proportion de messages correctement transmis par la n^{e} personne au cours de N réalisations de cette même chaîne.

   Tester la fonction pour différentes valeurs de p (on pourra prendre $n = 200$ et $N = 3000$). Que constate-t-on ?

2. À l'aide de la formule des probabilités totales, montrer que pour tout entier naturel n :

$$p_{n+1} = p + (1-2p)p_n$$

3. (a) Trouver un réel x tel que $x = p + (1-2p)x$.

   (b) Montrer que la suite u de terme général $u_n = p_n - x$ est géométrique.

   (c) En déduire l'expression de p_n en fonction de n.

4. Déterminer $\lim\limits_{n\to+\infty} p_n$. Interpréter et commenter.

Solution de l'exercice 8

1. (a)
```
import random as rd

def chaine(n,p):
    msg=1
    for i in range(1,n+1) :
        if rd.random() < p:
            msg=1-msg
    return msg
```

   (b)
```
def simul(N,n,p):
    S=0
    for i in range(1,N+1):
        S=S+chaine(n,p)
    return S/N
```

   Et on teste en console :

```
>>> simul(3000,200,0.1)
0.5006666666666667

>>> simul(3000,200,0.5)
0.49266666666666664

>>> simul(3000,200,0.9)
0.49233333333333335
```

2. Soit $n \in \mathbb{N}$. $\left\{A_n, \bar{A_n}\right\}$ est un système complet d'événements. Par la formule des probabilités totales on a donc :

$$\begin{aligned} p_{n+1} = \mathbb{P}(A_{n+1}) &= \mathbb{P}(A_n)\mathbb{P}_{A_n}(A_{n+1}) + \mathbb{P}\left(\bar{A_n}\right)\mathbb{P}_{\bar{A_n}}(A_{n+1}) \\ &= p_n(1-p) + (1-p_n)p = p + (1-2p)p_n \end{aligned}.$$

3. Pour tout naturel n, on note A_n l'événement « le signal transmis par la personne n est identique au signal émis ». (p_n) est arithmético-géométrique.

   On cherche un réel x tel que : $x = p + (1-2p)x \iff x = \dfrac{1}{2}$.

   La suite de terme général $u_n = p_n - \dfrac{1}{2}$ est géométrique de raison $1-2p$.

   Pour tout naturel n, $u_n = (1-2p)^n u_0 = \dfrac{1}{2}(1-2p)^n$, d'où $\boxed{p_n = \dfrac{1}{2}\left(1 + (1-2p)^n\right)}$.

4. $0 < p < 1$ donc $0 < 2p < 2$ puis $-2 < -2p < 0$ et $-1 < 1-2p < 1$ donc $\lim\limits_{n\to+\infty} (1-2p)^n = 0$ puis $\lim\limits_{n\to+\infty} p_n = \dfrac{1}{2}$.

   À l'issue d'un grand nombre de transmissions, il y a une chance sur deux que le message transmis soit déformé, et ce, quelle que soit la probabilité de mensonge.

Exercice 9

Vous disposez d'une trousse contenant 10 stylos dont un seul fonctionne.

1. Vous les essayez l'un après l'autre jusqu'à trouver celui qui fonctionne. Combien devrez vous effectuer d'essais de stylo en moyenne ?
2. Même question si l'on suppose qu'à chaque essai infructueux, vous remettez (à tort) le stylo dans la trousse et que vous tirez au hasard à nouveau.
3. Vous en essayez un (au hasard), puis s'il y a échec, un deuxième puis, s'il y a échec, vous remettez le premier et vous tirez (au hasard) le troisième puis, s'il y a échec, vous remettez le deuxième et vous tirez (au hasard) le quatrième, puis...

   Combien devrez vous effectuer d'essais de stylo en moyenne pour trouver le bon ?
4. (a) Proposer dans chacune des trois situations, une fonction `Python` d'en-tête `simul1(s)` (pour la première, puis `simul2(s)` et `simul3(s)` pour les suivantes) renvoyant le nombre d'essais nécessaires à la découverte du bon stylo, s étant le nombre de stylos (ici 10).

   (b) À l'aide des fonctions précédentes, écrire une fonction `esp(N, s)` qui fournit la moyenne du nombre d'essais nécessaires au cours de N réalisations de la même expérience où `s` correspond au nombre de stylos. Confronter les résultat expérimentaux aux valeurs trouvées dans les questions précédentes.

Solution de l'exercice 9

On note S_i l'événement « le i^{e} stylo testé fonctionne » et on note X la variable aléatoire égale au nombre de tirages nécessaires pour trouver le « bon » stylo. Il s'agit à chaque fois de déterminer l'espérance de X.

1. Il n'y a pas remise. X à valeurs dans $[\![1, 10[\![$

   $\mathbb{P}(X = 1) = \mathbb{P}(S_1) = 1/10$

   $\mathbb{P}(X = 2) = \mathbb{P}\left(\overline{S_1} \cap S_2\right) = 9/10 \times \dfrac{1}{9} = \dfrac{1}{10}$

   $\mathbb{P}(X = 3) = \mathbb{P}\left(\overline{S_1} \cap \overline{S_2} \cap S_3\right) = 9/10 \times \dfrac{8}{9} \times 1/8 = \dfrac{1}{10}$

   etc.

   $X \hookrightarrow \mathscr{U}([[1, 10]])$ donc $\boxed{\mathbb{E}(X) = 5,5.}$
2. X suit la loi géométrique de paramètre $1/10$ donc $\boxed{\mathbb{E}(X) = 10.}$
3. À partir du deuxième tirage, il y a toujours 9 stylos dans l'urne dont un seul est bon.

   $\mathbb{P}(X = 1) = \mathbb{P}(S_1) = \dfrac{1}{10}$

   $\mathbb{P}(X = 2) = \mathbb{P}\left(\overline{S_1} \cap S_2\right) = \dfrac{9}{10} \times \dfrac{1}{9} = \dfrac{1}{10}$

   $\mathbb{P}(X = 3) = \mathbb{P}\left(\overline{S_1} \cap \overline{S_2} \cap S_3\right) = \dfrac{9}{10} \times \dfrac{8}{9} \times \dfrac{1}{9}$

   $\mathbb{P}(X = 4) = \mathbb{P}\left(\overline{S_1} \cap \overline{S_2} \cap \overline{S_3} \cap S_4\right) = \dfrac{9}{10} \times \dfrac{8}{9} \times \dfrac{8}{9} \times \dfrac{1}{9}$ etc.

   Pour tout $k \in \mathbb{N}^*$, $\mathbb{P}(X = k) = \begin{cases} \dfrac{1}{10} & \text{si } k = 1 \\ \dfrac{1}{10}\left(\dfrac{8}{9}\right)^{k-2} & \text{si } k \geq 2 \end{cases}$

$$
\begin{aligned}
\mathbb{E}(X) &= \frac{1}{10} + \frac{1}{10}\sum_{k=2}^{+\infty} k\left(\frac{8}{9}\right)^{k-2} \quad \text{qui est une série convergente} \\
&= \frac{1}{10}\left(1 + \sum_{k=1}^{+\infty}(k+1)\left(\frac{8}{9}\right)^{k-1}\right) \quad \text{en réindexant} \\
&= \frac{1}{10}\left(1 + \sum_{k=1}^{+\infty} k\left(\frac{8}{9}\right)^{k-1} + \sum_{k=1}^{+\infty}\left(\frac{8}{9}\right)^{k-1}\right) \\
&= \frac{1}{10}\left(1 + 9 \underbrace{\sum_{k=1}^{+\infty} k\frac{1}{9}\left(\frac{8}{9}\right)^{k-1}}_{\text{espérance d'une géométrique de paramètre } \frac{1}{9}} + \frac{1}{1-\frac{8}{9}}\right) \\
&= \frac{1}{10}(1 + 81 + 9) = \frac{91}{10}
\end{aligned}
$$

4.

```
import random as rd

# Rercherche sans remise
def simul1(s):
    test=False
    n=0
    while not(test):
        test=(rd.random() < 1/(s-n))
        # il reste s-n stylos dans la trousse tous equiprobables
        n+=1
    return n

# Rercherche avec remise
def simul2(s):
    test=False
    n=0
    while not(test):
        test=(rd.random() < 1/s)
        n+=1
    return n

# Rercherche avec remise partielle
def simul3(s):
    if rd.random() < 1/s: # on a trouve le stylo du premier coup
        n=1
    else:
    # sinon on se ramene au cas numero 2
    # avec un stylo en moins sans oublier
    # qu'on en a deja  extrait un
        n=simul2(s-1)+1
    return n

stylos=10

# Estimateur empirique de l'esperance
def esp(N, s):
    S=0
```

```
for i in range(1, N+1):
    S=S+simul3(s)
return S/N
```
