Correction des exercices

Exercice 1

1. *a)* On définit la fonction :

```
def A(a, b, c, d):
    return np.array([[0, a, b, c, d],
                     [a, 0, b, c, d],
                     [a, b, 0, c, d],
                     [a, b, c, 0, d],
                     [a, b, c, d, 0]])
```

b) On exécute le script suivant :

```
for (a, b, c, d) in ((1, 2, 3, 4), (4, 2, 3, 1), (-3, -1, 1, 2)):
    print(alg.eigvals(A(a, b, c, d)))
```

```
[-1. 10. -2. -3. -4.]
[-4. 10. -2. -1. -3.]
[-1.  3.  1. -1. -2.]
```

Il semble que les valeurs propres de $A(a,b,c,d)$ soient $-a$, $-b$, $-c$, $-d$ et $a+b+c+d$.

c) Si on réalise le script suivant :

```
for (a, b, c, d) in ((1, 2, 3, 4), (4, 2, 3, 1), (-3, -1, 1, 2)):
    print(alg.eig(A(a, b, c, d)))
```

on constate que les vecteurs propres sont difficiles à lire. On va donc les normaliser en divisant à chaque fois par la dernière des coordonnées.

```
for (a, b, c, d) in ((1, 2, 3, 4), (4, 2, 3, 1), (-3, -1, 1, 2)):
    print('a b c d = ', a, b, c, d)
    vals, vps = alg.eig(A(a, b, c, d))
    for j in range(5):
        vps[:, j] = vps[:, j] / vps[4, j]
        print('valeur propre : ', vals[j], ' vecteur propre : ', vps[:, j])
```

On observe que les deux premières matrices sont diagonalisables, avec :

$$\mathrm{A}(1,2,3,4) = \mathrm{P}\begin{pmatrix} -1 & & & & \\ & -2 & & & \\ & & -3 & & \\ & & & -4 & \\ & & & & 10 \end{pmatrix}\mathrm{P}^{-1} \quad \text{avec} \quad \mathrm{P} = \begin{pmatrix} -10 & -3 & -7/6 & -2/5 & 1 \\ 1 & -3 & -7/6 & -2/5 & 1 \\ 1 & 1 & -7/6 & -2/5 & 1 \\ 1 & 1 & 1 & -2/5 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{pmatrix}$$

$$\mathrm{A}(4,2,3,1) = \mathrm{P}\begin{pmatrix} -4 & & & & \\ & -2 & & & \\ & & -3 & & \\ & & & -1 & \\ & & & & 10 \end{pmatrix}\mathrm{P}^{-1} \quad \text{avec} \quad \mathrm{P} = \begin{pmatrix} -5/2 & -1 & -4/9 & -1/10 & 1 \\ 1 & -1 & -4/9 & -1/10 & 1 \\ 1 & 1 & -4/9 & -1/10 & 1 \\ 1 & 1 & 1 & -1/10 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{pmatrix}$$

En revanche la troisième, pour laquelle -1 est valeur propre double, ne l'est pas ; on obtient les vecteurs propres :

$$\text{pour } \lambda = 3, \begin{pmatrix} -1/3 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}, \quad \text{pour } \lambda = 1, \begin{pmatrix} 1/2 \\ 1/2 \\ 1 \\ 1 \\ 1 \end{pmatrix}, \quad \text{pour } \lambda = -1, \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}, \quad \text{pour } \lambda = -2, \begin{pmatrix} 2 \\ 2 \\ 2 \\ 2 \\ 1 \end{pmatrix}$$

2. On calcule sans peine $\mathrm{A}\begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix} = \left(\sum_{i=1}^{n} a_i\right)\begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix}$ donc $\sum_{i=1}^{n} a_i$ est valeur propre et $\begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix}$ est un vecteur propre associé.

On calcule maintenant $A\begin{pmatrix}\alpha\\ \vdots\\ \alpha\\ \beta\\ \vdots\\ \beta\end{pmatrix} = \begin{pmatrix}\alpha(a_1+\cdots+a_{k-1})+\beta(a_k+\cdots+a_n)\\ \vdots\\ \alpha(a_1+\cdots+a_{k-1})+\beta(a_k+\cdots+a_n)\\ \alpha(a_1+\cdots+a_k)+\beta(a_{k+1}+\cdots+a_n)\\ \vdots\\ \alpha(a_1+\cdots+a_k)+\beta(a_{k+1}+\cdots+a_n)\end{pmatrix}$.

Cherchons α et β pour que ce vecteur soit propre pour la valeur propre $-a_k$; il faut que

$$\begin{cases}\alpha(a_1+\cdots+a_{k-1})+\beta(a_k+\cdots+a_n)=-a_k\alpha\\ \alpha(a_1+\cdots+a_k)+\beta(a_{k+1}+\cdots+a_n)=-a_k\beta\end{cases} \iff \alpha(a_1+\cdots+a_k)+\beta(a_k+\cdots+a_n)=0$$

Lorsque $(a_1+\cdots+a_k, a_k+\cdots+a_n)\neq(0,0)$, $-a_k$ est bien valeur propre pour le vecteur propre défini si dessus avec $\alpha=(a_k+\cdots+a_n)$ et $\beta=-(a_1+\cdots+a_k)$.

3. Lorsque les a_i sont strictement positifs on dispose de $n+1$ valeurs propres deux à deux distinctes : les $-a_k$ et $s=\sum_{i=1}^{n} a_i = 1$. Il existe donc $P\in\mathcal{M}_{n+1}(\mathbb{R})$ tel que $A(a_1,\dots,a_n)=P\,\mathrm{diag}(-a_1,\dots,-a_n,1)P^{-1}$.

Ainsi, $A(a_1,\dots,a_n)m = P\,\mathrm{diag}((-a_1)^m,\dots,(-a_n)^m,1)P^{-1}$ et puisque les a_i sont dans $]0,1[$, $\lim_{m\to+\infty} A(a_1,\dots,a_n)^m = P\,\mathrm{diag}(0,\dots,0,1)P^{-1}$; on reconnait la matrice du projecteur spectral sur le sous-espace propre associé à la valeur propre 1, autrement dit $\mathrm{Vect}\begin{pmatrix}1\\ \vdots\\ 1\end{pmatrix}$.

Exercice 2 Dans cet exercice on utilise les importations suivantes :

```
import numpy as np
import matplotlib.pyplot as plt
import numpy.random as rd
```

1. On a $|X_n|\leqslant\sum_{k=1}^{n}\frac{1}{2^k}\leqslant\sum_{k=1}^{+\infty}\frac{1}{2^k}=1$ donc $X_n\in[-1,1]$ et donc $\mathbb{P}(X_n\in[-1,1])=1$.
2. D'après le lemme des coalitions les variables $(\cos(tX_{n,k}))_{k\geqslant1}$ sont indépendantes donc d'après la loi faible des grands nombres, $\mathbb{P}\left(\left|\frac{1}{N}\sum_{k=1}^{N}\cos(tX_{n,k})-\mathbb{E}(\cos(tX_n))\right|\geqslant\epsilon\right)\leqslant\frac{\mathbb{V}(tX_n)}{N^2\epsilon^2}$. Cette quantité tend donc vers 0 lorsque N tend vers $+\infty$.
3. On définit les fonctions :

```
def X(n):
    s = 0
    for k in range(1, n+1):
        s += (2 * rd.randint(2) - 1) / 2**k
    return s

def f(n, t, N=1000):
    s = 0
    for k in range(1, N+1):
        s += np.cos(t * X(n))
    return s / N
```

puis on exécute le script :

```
T = np.linspace(-10, 10, 128)
for n in range(3, 11):
    Y = [f(n, t) for t in T]
    plt.plot(T, Y)
plt.grid()
plt.show()
```

La question précédente et cette simulation laissent présager que $\lim_{n\to+\infty}\mathbb{E}(\cos(tX_n))=\frac{\sin t}{t}$.

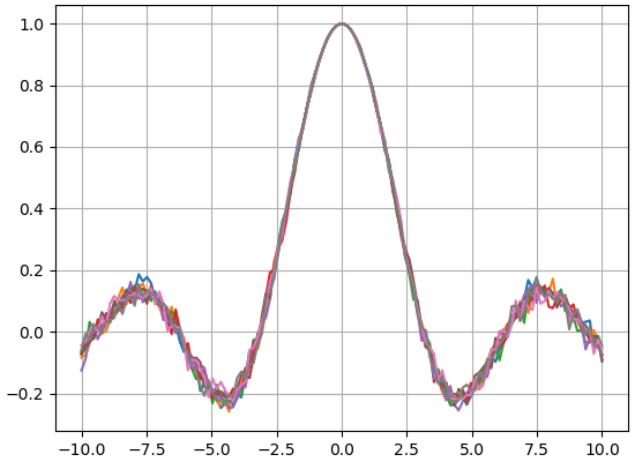

FIGURE 1 – Le graphe de la question 3

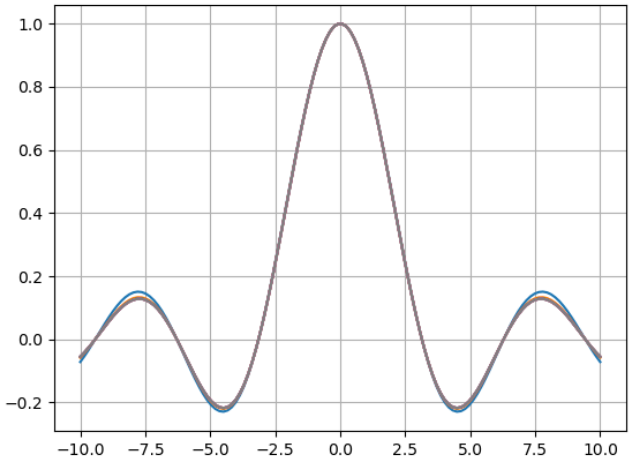

FIGURE 2 – Le graphe de la question 5

4. On a $e^{itX_n} = \prod_{k=1}^{n} e^{\frac{it\epsilon_k}{2^k}}$ et par le lemme des coalitions, $\Phi_{X_n}(t) = \prod_{k=1}^{n} \mathbb{E}\left(e^{\frac{it\epsilon_k}{2^k}}\right)$. On calcule $\mathbb{E}\left(e^{\frac{it\epsilon_k}{2^k}}\right) = \frac{e^{\frac{it}{2^k}} + e^{-\frac{it}{2^k}}}{2} = \cos\left(\frac{t}{2^k}\right)$ donc $\Phi_{X_n}(t) = \prod_{k=1}^{n} \cos\left(\frac{t}{2^k}\right)$.

   L'égalité $\cos\left(\frac{t}{2^k}\right) = \frac{1}{2}\frac{\sin\left(\frac{t}{2^{k-1}}\right)}{\sin\left(\frac{t}{2^k}\right)}$ permet de calculer ce produit par télescopage : $\Phi_{X_n}(t) = \frac{\sin(t)}{2^n \sin\left(\frac{t}{2^n}\right)}$.

5. On définit la fonction puis on réalise le script suivant :

```
def Phi(n, t):
    return np.sin(t) / 2**n / np.sin(t / 2**n)

T = np.linspace(-10, 10, 128)
for n in range(3, 11):
    Y = [Phi(n, t) for t in T]
    plt.plot(T, Y)
plt.grid()
plt.show()
```

   Il se pourrait bien que $\mathbb{E}(\cos(tX_n)) = \Phi_{X_n}(t)$.

6. Déjà traité à la question précédente.

7. Puisque $-\epsilon_k$ et ϵ_k suivent la même loi, il en est de même de X_n et de $-X_n$.

8. On en déduit que $\Phi_{X_n}(t) = \mathbb{E}(e^{itX_n}) = \mathbb{E}(e^{-itX_n})$ donc par linéarité de l'espérance, $\Phi_{X_n}(t) = \mathbb{E}\left(\frac{e^{itX_n} + e^{-itX_n}}{2}\right) = \mathbb{E}(\cos(tX_n))$; ainsi, $\mathbb{E}(\cos(tX_n)) = \frac{\sin(t)}{2^n \sin\left(\frac{t}{2^n}\right)}$ et $\lim_{n\to+\infty} \mathbb{E}(\cos(tX_n)) = \frac{\sin t}{t}$.

9. X_n est à valeurs finies donc on peut dériver dans l'espérance pour obtenir :

$$-\mathbb{E}(X_n \sin(tX_n)) = \frac{\cos t}{2^n \sin(\frac{t}{2^n})} - \frac{\sin(t)\cos(\frac{t}{2^n})}{4^n \sin(\frac{t}{2^n})^2}$$

   puis $\lim_{n\to+\infty} \mathbb{E}(X_n \sin(tX_n)) = \frac{\sin t - t\cos t}{t^2}$. Vérifions-le expérimentalement :

```
def g(n, t, N=1000):
    s = 0
    for k in range(1, N+1):
        x = X(n)
        s += x * np.sin(t * x)
    return s / N

T = np.linspace(-10, 10, 128)
Y = [g(10, t) for t in T]
plt.plot(T, Y)
Z = [(np.sin(t) - t * np.cos(t)) / t**2 for t in T]
plt.plot(T, Z)
plt.grid()
plt.show()
```

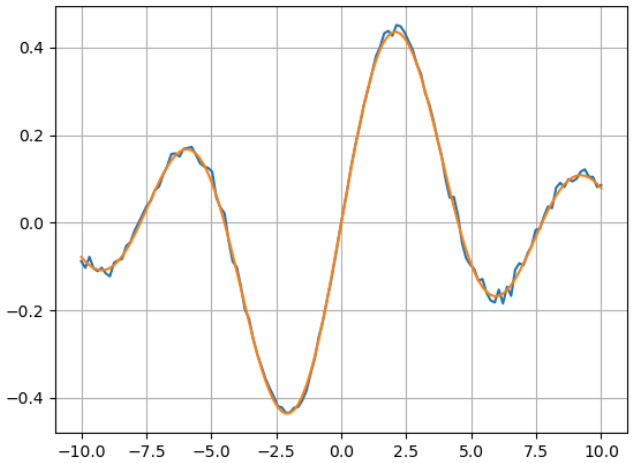

FIGURE 3 – Le graphe de la question 9

Exercice 3 Dans cet exercice on utilise les importations suivantes :

```
import numpy as np
import matplotlib.pyplot as plt
```

1. $f_1(x) = 2x$, $f_2(x) = \dfrac{4\sqrt{3}}{3}x^{3/2}$.

   Supposons $f_n(x) = \alpha_n x^{\beta_n}$. Alors $f_{n+1}(x) = \dfrac{4\sqrt{\alpha_n}}{\beta_n + 2} x^{\frac{\beta_n}{2}+1}$ donc les suites (α_n) et (β_n) sont définies par $\alpha_0 = 1$, $\beta_0 = 0$ et les relations $\alpha_{n+1} = \dfrac{4\sqrt{\alpha_n}}{\beta_n + 2}$ et $\beta_{n+1} = \dfrac{\beta_n}{2} + 1$.

2. On a $\beta_n = 2 - \dfrac{1}{2^{n-1}}$ donc $\lim \beta_n = 2$.

3. On définit la fonction :

```
def beta(n):
    return 2 - 1 / 2**(n-1)

def alpha(n):
    if n == 0:
        return 1
    return 4 * np.sqrt(alpha(n-1)) / (beta(n-1) + 2)
```

puis on réalise le script :

```
N = range(1, 21)
A = [alpha(n) for n in N]
plt.plot(N, A, '.')
plt.grid()
plt.show()
```

ce qui permet de conjecturer que la suite (α_n) converge vers 1.

4. On réalise maintenant le script suivant :

```
X = np.linspace(0, 1, 256)
for n in range(1, 21):
    Y = [alpha(n) * x**(beta(n)) for x in X]
    plt.plot(X, Y)
plt.grid()
plt.show()
```

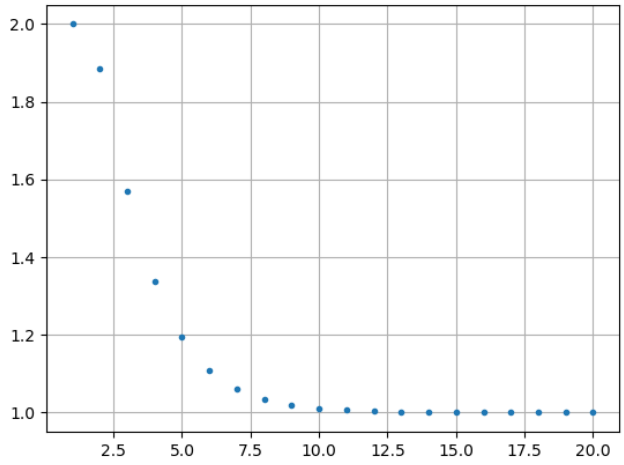

FIGURE 4 – La suite (α_n)

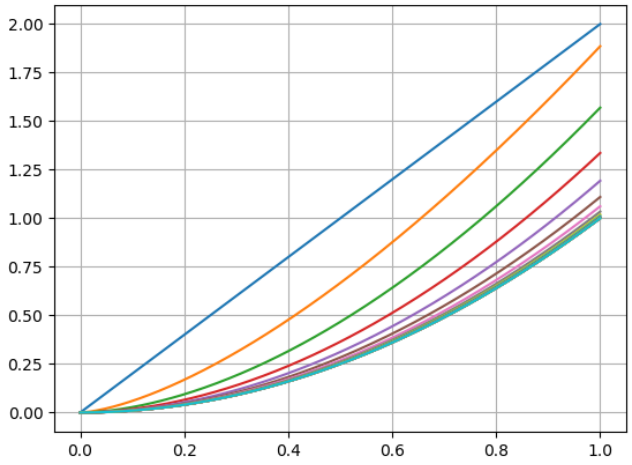

FIGURE 5 – Les fonctions (f_n)

La suite (f_n) semble converger uniformément vers $x \mapsto x^2$.

5. On a $\alpha_{k+1} = \dfrac{\sqrt{\alpha_k}}{1-1/2^{k+1}}$ donc $\ln(\alpha_{k+1}) = \dfrac{1}{2}\ln(\alpha_k) - \ln\left(1 - \dfrac{1}{2^{k+1}}\right)$.

Ainsi, $2^{k+1}\ln(\alpha_{k+1}) - 2^k \ln(\alpha_k) = -2^{k+1}\ln\left(1 - \dfrac{1}{2^{k+1}}\right)$ et par télescopage, $2^n \ln(\alpha_n) = -\displaystyle\sum_{k=1}^{n} 2^k \ln\left(1 - \frac{1}{2^k}\right)$.

Notons pour $x \in]0,1[$, $h(x) = -\dfrac{1}{x}\ln(1-x) = \displaystyle\sum_{n=1}^{+\infty} \frac{x^{n-1}}{n} \leqslant \sum_{n=1}^{+\infty} x^{n-1} = \frac{1}{1-x}$. En particulier, $h\left(\dfrac{1}{2^k}\right) \leqslant \dfrac{1}{1-1/2^k} \leqslant 2$ donc $0 \leqslant 2^n \ln(\alpha_n) \leqslant \displaystyle\sum_{k=1}^{n} 2$ donc $0 \leqslant \ln(\alpha_n) \leqslant \dfrac{n}{2^{n-1}}$. On en déduit que (α_n) converge vers 1.

6. Notons $f(x) = x^2$. D'après ce qui précède la suite (f_n) converge simplement vers f.

On a $f_n'(x) - f'(x) = x\left(\alpha_n\beta_n x^{\beta_n - 2} - 2\right) = x\left(\alpha_n\beta_n x^{-1/2^{n-1}} - 2\right)$. Posons $x_n = \left(\dfrac{\alpha_n\beta_n}{2}\right)^{2^{n-1}}$. Il s'agit de savoir si x_n appartient au segment $[0,1]$.

Cependant, les formules de récurrence donnent $\alpha_{n+1}\beta_{n+1} = 2\sqrt{\alpha_n} \geqslant 2$ car $\alpha_n \geqslant 1$, donc $x_n \geqslant 1$. On en déduit que $f_n - f$ est positive et croissante sur $[0,1]$ et donc que $\|f_n - f\|_\infty = f_n(1) - f(1)$ et puisque $\lim f_n(1) = f(1)$ (on a déjà prouvé la convergence simple) on en déduit que la convergence est bien uniforme sur $[0,1]$.