Exercices Centrale 2 - PSI

Exercice 1

2018 - Centrale 2 - PSI

Une matrice $A \in \mathcal{M}_n(\mathbb{R})$ vérifie la propriété $\mathcal{H}_n$ si ses coefficients appartiennent tous à $\{-1, 1\}$ et si les colonnes de A forment une famille orthogonale.

1. A l'aide de l'ordinateur, dénombrer les matrices vérifiant $\mathcal{H}_2$.
2. Soit $A \in \mathcal{M}_n(\mathbb{R})$ dont tous les coefficients appartiennent à $\{-1, 1\}$. Montrer que A vérifie $\mathcal{H}_n$ si et seulement si $\frac{1}{\sqrt{n}}A$ est orthogonale.
3. Décrire les transformations du plan associées aux matrices vérifiant $\mathcal{H}_2$.
4. Soit $A \in \mathcal{M}_n(\mathbb{R})$ vérifiant la propriété $\mathcal{H}_n$. Montrer que la transposée de A vérifie aussi $\mathcal{H}_n$.
5. Soit $A \in \mathcal{M}_n(\mathbb{R})$ vérifiant la propriété $\mathcal{H}_n$. Montrer que la matrice déduite de A en changeant tous les signes sur une ligne ou sur une colonne vérifie $\mathcal{H}_n$.
6. A l'aide de l'ordinateur, dénombrer les matrices vérifiant $\mathcal{H}_4$ et dont la première ligne et la première colonne ne sont composées que de 1.
7. En déduire le nombre de matrices vérifiant $\mathcal{H}_4$.[a]

a. Cette dernière question ne figurait pas dans l'énoncé originel, mais c'est la suite logique de la question précédente.

Solution

1 Un premier essai naïf : On va énumérer toutes les matrices $\begin{pmatrix} a & b \\ c & d \end{pmatrix}$ dont les coefficients valent tous -1 ou 1 et on va vérifier à chaque fois si $\begin{pmatrix} a \\ c \end{pmatrix}$ et $\begin{pmatrix} b \\ d \end{pmatrix}$ sont orthogonaux, auquel cas on incrémente un compteur :

```
cpt = 0

for a in [-1,1]:

    for b in [-1,1]:

        for c in [-1,1]:

            for d in [-1,1]:

                if a*c + b*d == 0 :

                    cpt += 1


print(cpt)
8
```

Ça marche, mais ce n'est pas très satisfaisant car si on veut augmenter n on va se retrouver à imbriquer n^2 boucles, ce qui va vite devenir fastidieux. Et on aimerait bien plutôt avoir une fonction qu'on pourrait invoquer pour n'importe quelle valeur de n au lieu de devoir réécrire le code à chaque fois.

Commençons par écrire une fonction qui détermine si une matrice a toutes ses colonnes deux à deux orthogonales (les matrices seront représentées sous forme d'un array python) :

```
import numpy as np

def ortho(M):

    n = np.shape(M)[0]

    for i in range(n-1):

        for j in range(i+1,n):

            s = 0

            for k in range(n):

                s += M[k,i] * M[k,j]

            if s != 0 :

                return False

                # quand on rencontre deux colonnes
                #  non orthogonales,
                # on sait que la propriete n'est pas verifiee

    return True
```

Pour énumérer toutes les matrices de taille n à coefficients ± 1, il « suffit »d'énumérer toutes les listes de longueur n^2 à coefficients ± 1 puis de les « reshaper »en array.
Et pour énumérer les listes en questions, une fonction récursive est fort tentante, selon le principe suivant :
Pour énumérer toutes les listes de longueur k :

- Si $k = 0$, il n'y a rien à faire (terminaison de récursion).
- sinon, je vais :
  - Écrire un 1 puis ordonner à un esclave de compléter avec toutes les listes de longueur $k - 1$.
  - Écrire un -1 puis ordonner à un esclave de compléter avec toutes les listes de longueur $k - 1$.

```
def denombre(n):

    L = np.zeros([n**2])

    # array unidimesionnel pour enumerer
    # toutes les listes en question

    cpt = [0]

    # compteur du nombre de matrices qui conviennent
    # (sous forme d'une liste, pour que la fonction annexe
```

```
    # puisse en modifier la valeur !)

    def enumere(k) :

        # fonction recursive annexe qui gere
        # le k-ieme coefficient de L

        if k == n**2  :

            # terminaison : la liste L est complete
            # On verifie l'orthogonalite
            # de la matrice associee

            M = L.reshape([n,n])

            if ortho(M):

                cpt[0] += 1

        else :

            # On met sucessivement -1 et 1 dans la case
            # numero k et on lance notre
            #  esclave sur la case suivante

            for x in [-1,1]:

                L[k] = x

                enumere(k+1)

    enumere(0)

    return cpt[0]
```

Ok, je me suis fait plaisir avec cette fonction récursive mais elle ne permet pas de traiter beaucoup de valeurs de n. Il faut énumérer 2^{n^2} matrices, ce qui prend rapidement beaucoup de temps. On peut aller jusqu'à $n = 5$ en s'armant d'un peu de patience :

```
denombre(2)
Out[]: 8
denombre(3)
Out[]: 0
denombre(4)
Out[]: 768
denombre(5)
Out[]: 0
```

Je ne pense pas que le jury attendait un tel degré de sophistication sur cette question ! L'essai « naïf »devait suffire à le satisfaire. Mais si on a fini l'exo lors de la préparation et s'il nous reste cinq minutes, on peut peut-être raisonnablement se lancer là-dedans pour parfaire l'approche informatique ...

2 • Si A vérifie $\mathcal{H}_n$, alors toutes ses colonnes sont deux à deux orthogonales. Cette

propriété est conservée quand on multiplie la matrice par un scalaire, en particulier par $\frac{1}{\sqrt{n}}$. Donc les colonnes de $\frac{1}{\sqrt{n}}A$ sont deux à deux orthogonales.
de plus toutes les colonnes de $\frac{1}{\sqrt{n}}A$ sont normées car :

$$\sum_{k=1}^{n}\left(\frac{\pm 1}{\sqrt{n}}\right)^2 = \sum_{k=1}^{n}\frac{1}{n} = 1$$

Donc $\frac{1}{\sqrt{n}}A$ est orthogonale.

- Si $\frac{1}{\sqrt{n}}A$ est orthogonale, alors en particulier toutes ses colonnes sont deux à deux orthogonales et cette propriété est conservée lorsqu'on multiplie par $\sqrt{n}$, donc A vérifie $\mathcal{H}_n$.

Ceci prouve bien l'équivalence attendue.

3 On sait d'après la classification des isométries vectorielles du plan que les matrices de $O_2(\mathbb{R})$ sont toutes de la forme $\begin{pmatrix}\cos\theta & -\sin\theta\\ \sin\theta & \cos\theta\end{pmatrix}$ (rotation d'angle θ) ou bien $\begin{pmatrix}\cos\theta & \sin\theta\\ \sin\theta & -\cos\theta\end{pmatrix}$ (réflexion d'axe $\text{Vect}\left(\begin{pmatrix}\cos\frac{\theta}{2}\\ \sin\frac{\theta}{2}\end{pmatrix}\right)$).

D'après la question précédente, une matrice vérifiant $\mathcal{H}_n$ doit donc être de la forme $\sqrt{2}\begin{pmatrix}\cos\theta & -\sin\theta\\ \sin\theta & \cos\theta\end{pmatrix}$ ou bien $\sqrt{2}\begin{pmatrix}\cos\theta & \sin\theta\\ \sin\theta & -\cos\theta\end{pmatrix}$.

Donc ces matrices représentent la composition d'une homothétie de rapport $\sqrt{2}$ avec une rotation ou une symétrie. Ses coefficients devant en plus être égaux à -1 ou 1, les sinus et cosinus doivent valoir $\frac{\sqrt{2}}{2}$ ou $-\frac{\sqrt{2}}{2}$, ce qui nous donne les possibilités suivantes :

- L'homothétie de rapport $\sqrt{2}$ composée avec les rotations d'angles respectifs $\frac{\pi}{4}, \frac{3\pi}{4}, \frac{5\pi}{4}$ et $\frac{7\pi}{4}$.
- L'homothétie de rapport $\sqrt{2}$ composée avec les réflexions par rapport aux droites vectorielles dont les angles avec l'axe des abscisses valent respectivement $\frac{\pi}{8}, \frac{3\pi}{8}, \frac{5\pi}{8}$ et $\frac{7\pi}{8}$.

4 La matrice A n'étant composée que de 1 et -1, il en va de même pour sa transposée.
De plus la matrice $B = \frac{1}{\sqrt{n}}A$ est orthogonale (cf question **2.**) donc $BB^T = I_n$.
On a donc aussi $B^TB = I_n$, c'est à dire que B^T est elle aussi orthogonale.
Et donc A^T vérifie $\mathcal{H}_n$ (car $\frac{1}{\sqrt{n}}A^T = B^T$).

5 Si on change les signes d'une colonne de A, on obtient encore une matrice constituée de 1 et -1, et le changement de signe n'affecte pas l'orthogonalité des colonnes. Donc la matrice obtenue vérifie elle aussi $\mathcal{H}_n$.
Si on change les signes d'une ligne, cela revient à changer les signes d'une colonne de A^T. D'après ce qu'on vient de voir, la matrice ainsi obtenue à partir de A^T vérifie encore $\mathcal{H}_n$, et donc la matrice obtenue à partir de A la vérifie aussi.

6 On peut encore s'amuser à imbriquer neuf boucles, mais je préfère réutiliser mon énumération récursive :

```
def denombrebis():

    L = np.zeros([9])

    M = np.ones([4,4])
```

```
    cpt = [0]

    def enumere(k) :

        if k == 9  :

            M[1:4,1:4] = L.reshape([3,3])

            if ortho(M):

                cpt[0] += 1

        else :

            for x in [-1,1]:

                L[k] = x

                enumere(k+1)

    enumere(0)

    return cpt[0]


denombrebis()
Out[]: 6
```

Il y a donc six matrices de ce type.

7 A partir d'une matrice vérifiant $\mathcal{H}_4$ on peut obtenir une unique matrice avec des 1 sur la première ligne et la première colonne en changeant éventuellement les signes des quatre lignes et des trois dernières colonnes (attention ici : si on s'autorise à changer aussi le signe de la première colonne, il n'y a plus unicité du procédé et le dénombrement qui suit tombe un peu à l'eau ...).

Donc chacune des six matrices décrites à la question précédente est associée à 2^7 matrices qui vérifient $\mathcal{H}_4$. Il y a donc 6×2^7 matrices qui vérifient $\mathcal{H}_4$.

On peut vérifier à l'aide de la fonction codée au tout début :

```
denombre(4)
Out[]: 768
6*2**7
Out[]: 768
```

Remarque : Travers symptomatique du concours Centrale : proposer des exercices très longs qui peuvent décourager les candidats (ledit travers étant ici accentué par mon penchant naturel pour la récursivité et par l'irrépressible envie de rajouter la question qui conclue naturellement l'exercice)

Exercice 2

2018 - Centrale 2 - PSI

Soit $(a_n) \in (\mathbb{R}^{+*})^{\mathbb{N}}$. On considère la suite (u_n) définie par $u_0 = 1, u_1 = a_1$ et pour tout $n \geq 2$, $u_n = a_n u_{n-1} + u_{n-2}$.

1 Écrire une fonction Python qui prend en argument les p premiers termes de la suite (a_n) donnés sous forme de liste et renvoie les p premiers termes de la suite (u_n).

2 Écrire une fonction Python qui prend en argument les p premiers termes de la suite (a_n) donnés sous forme de liste et renvoie les premières sommes partielles (S_n) de la série numérique de terme général $\dfrac{(-1)^k}{u_k u_{k-1}}$.

3 Tracer les 20 premiers points (n, S_n) dans les cas $a_n = \frac{1}{n^2}$, $a_n = \frac{1}{\sqrt{n}}$ et $a_n = 1$. Que peut-on conjecturer ?

4 On suppose que la série de terme général a_n converge. Montrer que pour tout $n \geq 1$, $u_n \leq \prod_{k=1}^{n}(1 + a_k)$. Étudier la nature de la série de terme général $\dfrac{(-1)^k}{u_k u_{k-1}}$.

5 On suppose que la série de terme général a_n diverge. Étudier la nature de la série de terme général $\dfrac{(-1)^k}{u_k u_{k-1}}$.

Indication : introduire la série de terme général $u_{n-1}(u_n - u_{n-2})$.

Solution

1 L'énoncé est un peu bizarre puisque la suite (a_n) commence à a_0, mais a_0 ne sert à rien. On va faire comme si on n'avait rien vu :

```
def U(a):

    p = len(a)

    U = [1,a[1]]

    for n in range(2,p):

        U.append( a[n] * U[n-1] + U[n-2] )

    return U
```

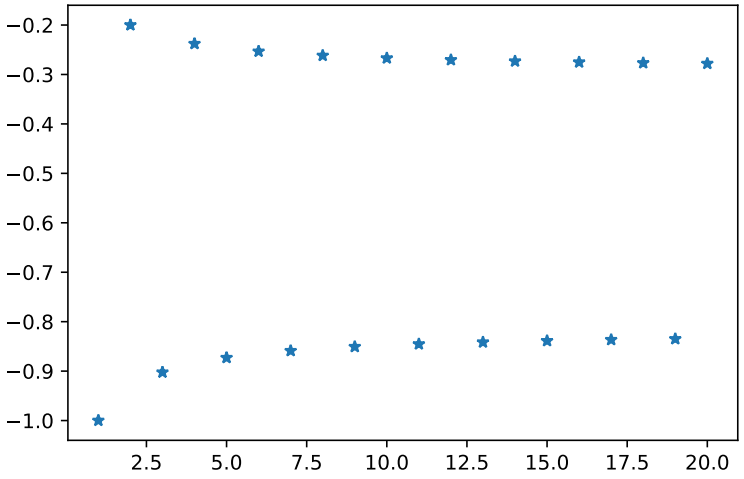
2

```
def S(a):

    u = U(a)

    S = [ (-1) / ( u[0] * u[1] ) ]

    for k in range(2,len(u)):

        S.append( S[-1] + (-1)**k / ( u[k-1] * u[k] ) )

    return S
```

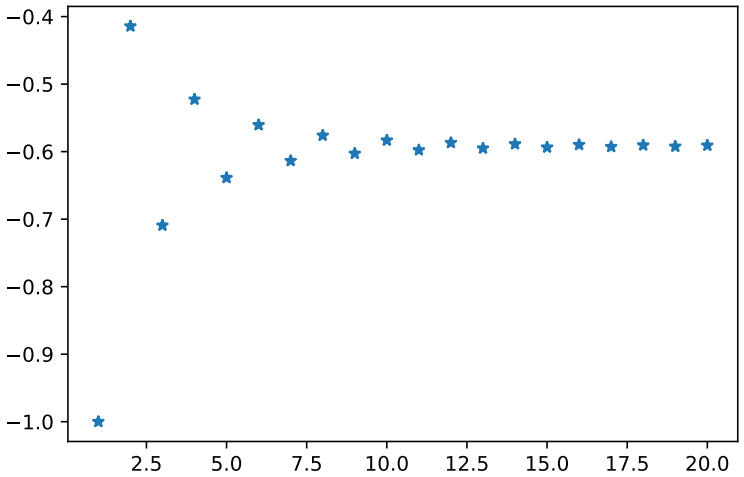
3 Pour la première suite proposée :

```
import matplotlib.pyplot as plt

a =[0] + [ 1/n**2 for n in range(1,21)]

les_Sn = S(a)

les_n = [n for n in range(1,21)]

plt.plot(les_n,les_Sn,'*')

plt.show()
```

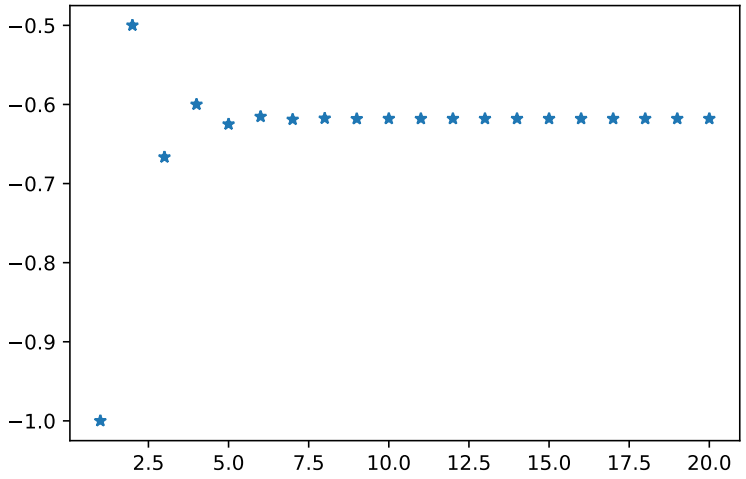
On obtient :



On dirait que les suites extraites (S_{2n}) et (S_{2n+1}) convergent, mais vers des limites différentes.
Avec la deuxième suite on obtient :



Et avec la dernière :



Dans les deux derniers cas, la série semble converger.

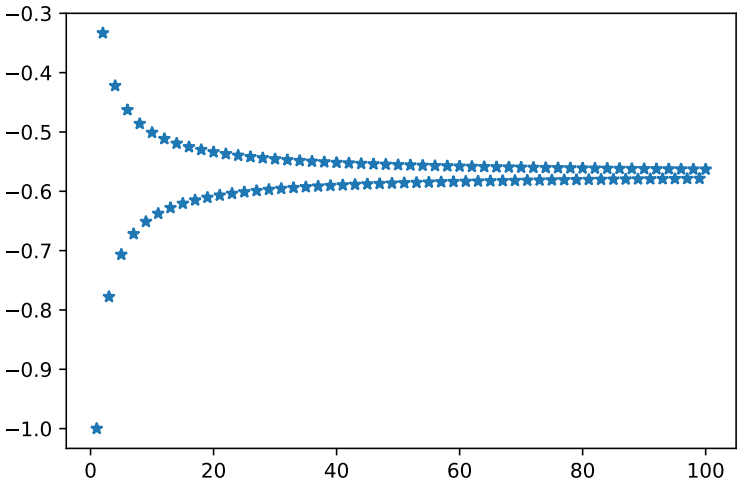
Qualitativement, plus les a_n sont grands, plus les u_n seront grands et plus la série des $\frac{(1)^k}{u_{k-1}u_k}$ aura des chances de converger.

Avec $a_n = 1$ et $a_n = 1/\sqrt{n}$, les a_n sont assez grands pour faire converger la série.

En revanche quand on descend à $a_n = 1/n^2$ les a_n sont devenus trop petits et la série diverge.

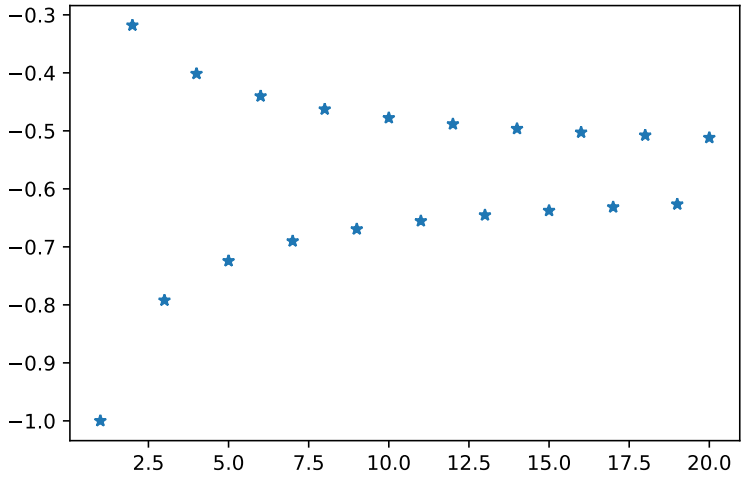
Essayons des suites de (a_n) intermédiaires pour mieux cerner le problème :

- Avec $a_n = 1/n$:

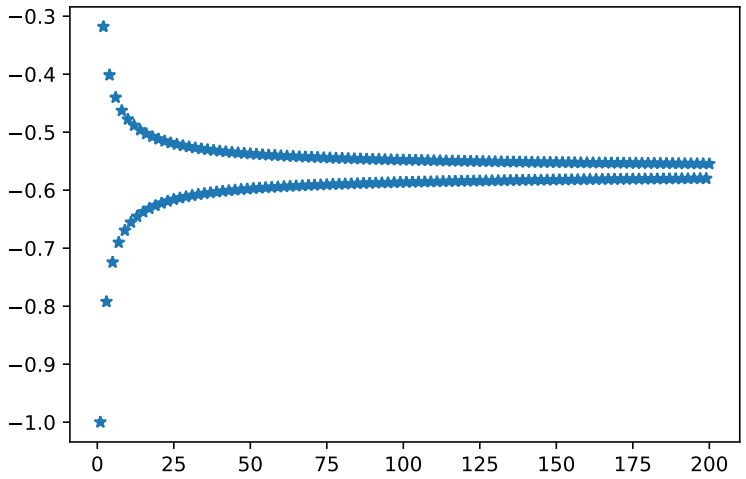


La série semble encore converger.

4 Avec $a_n = 1/n^{1.1}$:



Difficile de dire si ça converge lentement ou bien si ça diverge. Essayons avec des valeurs de n plus grandes :



C'est pas super clair... La bascule semble tout de même se faire autour de $a_n = 1/n$. Peut-être il a-t-il un lien avec la convergence de la série $\sum a_n$?
(oui, j'ai lorgné sur la suite de l'exo avant de courageusement avancer ma conjecture...)

5 Montrons le résultat par récurrence sur n :

- pour $n = 1$ on a $u_1 = a_1 \leq 1 + a_1$.
- pour $n = 2$ on a $u_2 = a_2 a_1 + 1 \leq 1 + a_2 a_2 + a_1 + a_2 = (1 + a_1)(1 + a_2)$

- Supposons la propriété vraie pour n et $n+1$. Alors :

$$\begin{aligned}
u_{n+2} &= a_{n+2}u_{n+1} + u_n \\
&\leq a_{n+2}\prod_{k=1}^{n+1}(1+a_k) + \prod_{k=1}^{n}(1+a_k) \\
&\leq \prod_{k=1}^{n}(1+a_k)\Big(a_{n+2}(1+a_{n+1}) + 1\Big) \\
&\leq \prod_{k=1}^{n}(1+a_k)\Big(a_{n+2} + a_{n+2}a_{n+1} + 1\Big) \\
&\leq \prod_{k=1}^{n}(1+a_k)\Big((a_{n+1}+1)(a_{n+2}+1)\Big) \\
&\leq \prod_{k=1}^{n+2}(1+a_k)
\end{aligned}$$

Et donc la propriété est vraie pour tout $n \geq 1$.
u_n étant de plus clairement positif, on obtient l'encadrement :

$$0 \leq u_n \leq \exp\Big(\sum_{k=1}^{n}\ln(1+a_k)\Big)$$

Puisque la série $\sum a_n$ converge, son terme général tend vers 0 et alors $\ln(1+a_n) \sim a_n$.
De plus, les a_n sont positifs, et donc les $\ln(1+a_n)$ aussi.
Donc la série de terme général $\ln(1+a_n)$ est de même nature que la série de terme général a_n, à savoir convergente.
Notons donc S la somme de cette série. Comme les $\ln(1+a_n)$ sont positifs, toutes les sommes partielles de cette série sont majorées par S.
Il vient alors que :

$$0 \leq u_n \leq e^S$$

Et donc que :

$$\forall k \in \mathbb{N}^*, \frac{1}{u_{k-1}u_k} \geq e^{-2S} > 0$$

Et donc la série de terme général $\dfrac{(-1)^k}{u_{k-1}u_k}$ est grossièrement divergente.

6 Posons $v_n = u_{n-1}(u_n - u_{n-2})$.
D'après la relation qui définit la suite u_n, on a $v_n = a_n u_{n-1}^2$.
La suite u_n est minorée par $m = \mathrm{Min}(1, a_1) > 0$ (récurrence limpide).
On a donc $v_n \geq m^2 a_n$ et donc la série de terme général v_n diverge vers $+\infty$.
Mais d'autre part cette série est télescopique :

$$\begin{aligned}
\sum_{k=2}^{n} v_k &= \sum_{k=2}^{n}(u_{k-1}u_k - u_{k-1}u_{k-2}) \\
&= \sum_{k=2}^{n} u_{k-1}u_k - \sum_{k=1}^{n-1} u_{k-1}u_k \\
&= u_{n-1}u_n - u_0u_1
\end{aligned}$$

Donc la suite $(u_{n-1}u_n)$ tend vers $+\infty$.
D'autre part cette suite est croissante :

$$u_{n-1}u_n - u_{n-2}u_{n-1} = u_{n-1}(u_n - u_{n-2}) = a_n u_{n-1}^2 > 0$$

Donc la suite $\left(\dfrac{1}{u_{k-1}u_k}\right)_{k\in\mathbb{N}^*}$ est positive, décroissante et tend vers 0.

Le théorème spécial des séries alternées assure alors que la série de terme général $\dfrac{(-1)^k}{u_{k-1}u_k}$ est convergente.

Exercice 3

2018 - Centrale 2 - PSI

Soit $\omega = e^{\frac{2i\pi}{3}}$. On pose, pour tout $n \in \mathbb{N}^*$, $S_n = \sum_{k=1}^{n} \dfrac{\omega^k}{k}$.

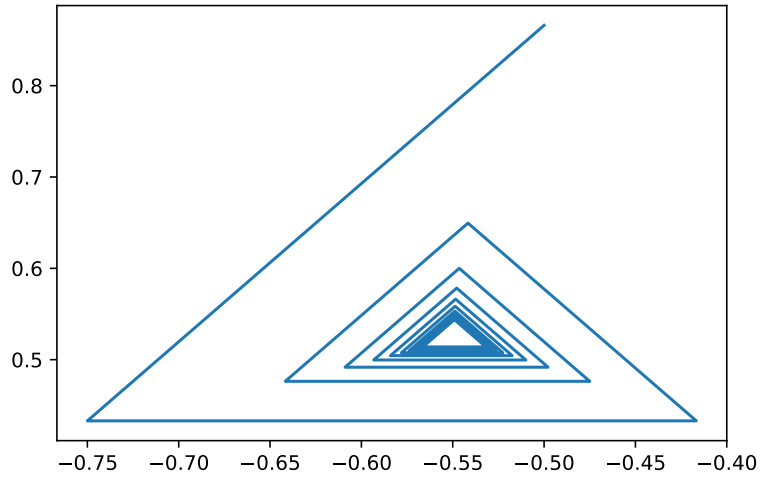
1. Étudier avec Python la convergence de la suite (S_n).
2. Calculer numériquement puis de manière exacte $I = \displaystyle\int_0^1 \frac{\omega - t}{1+t+t^2}\,\mathrm{d}t$.
3. Montrer que (S_n) converge vers I.

Solution

1. Représentons dans le plan complexes les quelques premières valeurs de S_n :

```
import matplotlib.pyplot as plt

from math import cos , sin , pi

omega = cos(2*pi/3) + 1.j * sin(2*pi/3)

nbpts = 30

les_s = [omega]

for k in range(2,nbpts+1):

    les_s.append( les_s[-1] + omega**k/k)

reelle = [z.real for z in les_s]

imag = [z.imag for z in les_s]

plt.plot(reelle,imag)

plt.show()
```

On obtient :



La suite semble converger vers une limite proche de $-0.55 + 0.55i$.

2 En séparant partie imaginaires et partie réelle de la fonction à intégrer :

```
def f(t):

    return (omega.real -t ) / ( 1 + t + t**2 )

def g(t):

    return (omega.imag ) / ( 1 + t + t**2 )

print( quad(f,0,1)[0]+quad(g,0,1)[0]*1.j)

(-0.5493061443340547+0.5235987755982989j)
```

Pour le calcul exact, séparons à nouveau partie réelle et partie imaginaire :

$$\begin{aligned}\int_0^1 \frac{-\frac{1}{2}-t}{1+t+t^2}\,\mathrm{d}t &= -\frac{1}{2}\int_0^1 \frac{1+2t}{1+t+t^2}\,\mathrm{d}t \\ &= -\frac{1}{2}\Big[\ln(1+t+t^2)\Big]_0^1 \\ &= -\frac{1}{2}\ln(3)\end{aligned}$$

$$\begin{aligned}
\int_0^1 \frac{\frac{\sqrt{3}}{2}}{1+t+t^2}\,\mathrm{d}t &= \frac{\sqrt{3}}{2}\int_0^1 \frac{\mathrm{d}t}{(\frac{1}{2}+t)^2+\frac{3}{4}} \\
&= \frac{2}{\sqrt{3}}\int_0^1 \frac{\mathrm{d}t}{1+[\frac{2}{\sqrt{3}}(\frac{1}{2}+t)]^2} \\
&= \left[\arctan\left(\frac{2}{\sqrt{3}}(\frac{1}{2}+t)\right)\right]_0^1 \\
&= \arctan(\sqrt{3}) - \arctan(\frac{1}{\sqrt{3}}) \\
&= \frac{\pi}{3} - \frac{\pi}{6} \\
&= \frac{\pi}{6}
\end{aligned}$$

Donc :

$$I = -\frac{1}{2}\ln(3) + \frac{i\pi}{6}$$

On vérifie avec Python que cela correspond bien à la valeur numérique obtenue précédemment :

```
from math import log

print( -log(3)/2 + pi/6*1.j)

(-0.5493061443340549+0.5235987755982988j)
```

3 L'idée est ici d'interpréter S_n comme une intégrale sur $[0\,;1]$ en exploitant le fait que $\frac{1}{k} = \int_0^1 t^{k-1}\,\mathrm{d}t$, ce qui devrait permettre de confronter S_n à I :

$$\begin{aligned}
I - S_n &= \int_0^1 \frac{\omega - t}{1+t+t^2}\,\mathrm{d}t - \sum_{k=1}^{n}\omega^k \int_0^1 t^{k-1}\,\mathrm{d}t \\
&= \int_0^1 \left(\frac{\omega - t}{1+t+t^2} - \sum_{k=0}^{n-1}\omega(\omega t)^k\right)\mathrm{d}t \\
&= \int_0^1 \left(\frac{\omega - t}{1+t+t^2} - \omega\frac{1-(\omega t)^n}{1-\omega t}\right)\mathrm{d}t \\
&= \int_0^1 \left(\frac{\omega - t}{1+t+t^2} - \frac{1-(\omega t)^n}{\overline{\omega} - t}\right)\mathrm{d}t \\
&= \int_0^1 \left(\frac{\omega - t}{1+t+t^2} - \frac{(1-(\omega t)^n)(\omega - t)}{(\overline{\omega} - t)(\omega - t)}\right)\mathrm{d}t \\
&= \int_0^1 \frac{(\omega - t)(\omega t)^n}{1+t+t^2}\,\mathrm{d}t
\end{aligned}$$

Compte tenu du fait que :

- $|\omega| = 1$.
- $|\omega - t| \leq |\omega| + |t| \leq 2$.
- $\left|\frac{1}{1+t+t^2}\right| \leq 1$.

on obtient la majoration :

$$|I - S_n| \leq 2\int_0^1 t^n\,\mathrm{d}t = \frac{2}{n+1} \underset{n\to+\infty}{\longrightarrow} 0$$

Et donc (S_n) tend bien vers I.

Exercice 4

2018 - Centrale 2 - PSI

Une urne contient initialement n boules numérotées de 1 à n. On pioche une boule, son numéro est k :

- Si $k = 1$, l'expérience s'achève.
- sinon, on recommence en ayant enlevé de l'urne toutes les boules avec un numéro supérieur ou égal à k.

On note X_n la variable aléatoire égale au nombre de tirages de l'expérience.

1. Écrire une fonction Python *experience* effectuant l'expérience et renvoyant une réalisation de X_n.
2. Écrire une fonction Python *moyenne* renvoyant la moyenne de 10^3 réalisations de X_n.
3. Tracer les valeurs de $moyenne(n) - \ln n$ pour différentes valeurs de n. Que peut-on conjecturer ?
4. Montrer que pour $n > 2$ et $j \in [\![1\,;n]\!]$ on a $P(X_n = j) = \dfrac{1}{n}\displaystyle\sum_{k=1}^{n} P(X_{k-1} = j-1)$ puis que :

$$P(X_n = j) = \frac{n-1}{n}P(X_{n-1} = j) + \frac{1}{n}P(X_{n-1} = j-1)$$

On note, pour tout n, G_n la fonction génératrice de X_n.

5. Calculer $G_1(t)$ puis montrer que $G_n(t) = \dfrac{n-1+t}{n}G_{n-1}(t)$.
6. Donner la loi de X_n.
7. Calculer $E(X_n)$ en fonction de $E(X_{n-1})$ puis démontrer la conjecture.

Solution

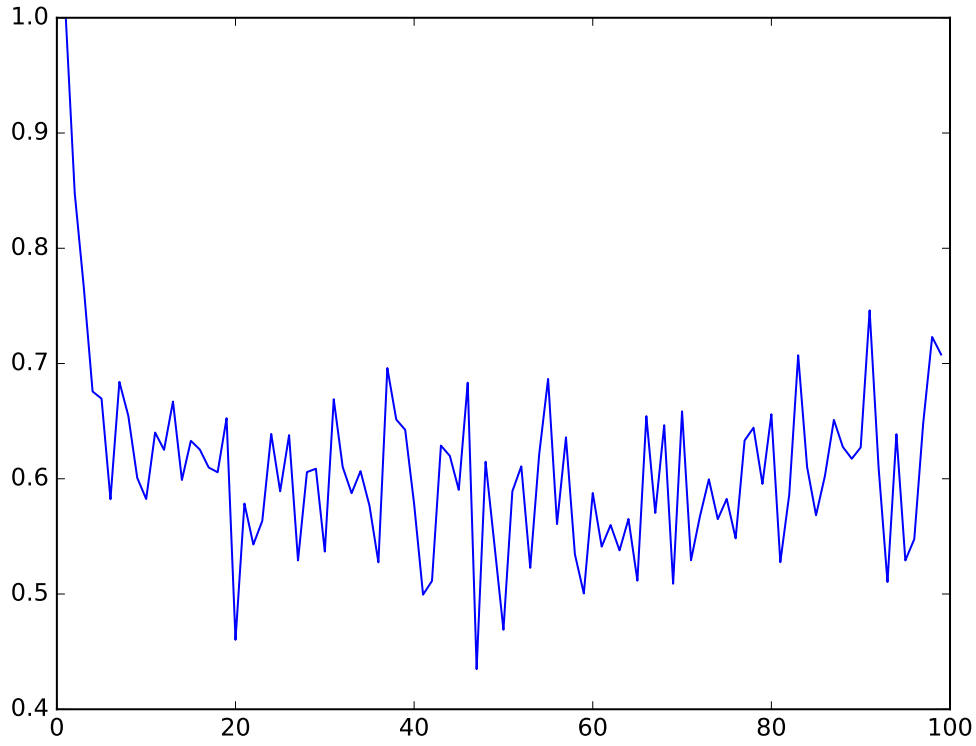
1.
```
import numpy.random as rd

def experience(n):

    maxurne = n

    cpt = 0

    while maxurne != 0 :

        cpt += 1

        maxurne = rd.randint(1,maxurne+1) - 1

    return cpt
```

2.
```
def moyenne(n):

    s = 0
```

```
    for k in range(10**3):
        s += experience(n)
    return s / 10**3
```

```
from math import log
import matplotlib.pyplot as plt
les_n = [n for n in range(1,100)]
les_m = [moyenne(n) - log(n) for n in les_n]
plt.plot(les_n,les_m)
plt.show()
```

On obtient :



3

On dirait que la différence entre $E(X_n)$ et $\ln n$ est bornée, et donc que $E(X_n)$ est équivalente à $\ln n$ lorsque n tend vers $+\infty$.

4 L'énoncé est un peu imprécis puisqu'il invoque la variable aléatoire X_0 qui n'est pas vraiment définie ! On complète cette définition en disant que X_0 est une variable aléatoire certaine de valeur 0 (c'est cette convention qui fera marcher la formule qui suit).

Introduisons les événements A_k = « la première boule piochée porte le numéro k »pour tout $k \in [\![1\,;n]\!]$.

$\{A_k; k \in [\![1\,;n]\!]\}$ est un système complet d'événements. La formule des probabilités totales

donne :

$$\begin{aligned} P(X_n = j) &= \sum_{k=1}^{n} P(A_k) P_{A_k}(X_n = j) \\ &= \sum_{k=1}^{n} \frac{1}{n} P(X_{k-1} = j-1) \\ &= \frac{1}{n} \sum_{k=1}^{n} P(X_{k-1} = j-1) \end{aligned}$$

En effet, si A_k est réalisé au premier tirage, on se retrouve avec une urne qui contient les boules 1 à $k-1$ et la probabilité (conditionnelle) qu'on effectue alors $j-1$ tirages est donc égale à $P(X_{k-1} = j-1)$.
Le cas $k = 1$ marche aussi grâce à la convention choisie pour X_0 car $P_{A_1}(X_n = 1) = 1$ et $P_{A_1}(X_n = j) = 0$ pour $j > 1$.
En exploitant la formule précédente (d'abord pour n puis pour $n-1$) :

$$\begin{aligned} P(X_n = j) &= \frac{1}{n} \sum_{k=1}^{n} P(X_{k-1} = j-1) \\ &= \frac{1}{n} \Big(\sum_{k=1}^{n-1} P(X_{k-1} = j-1) + P(X_{n-1} = j-1) \Big) \\ &= \frac{1}{n} \times \frac{n-1}{n-1} \sum_{k=1}^{n-1} P(X_{k-1} = j-1) + \frac{1}{n} P(X_{n-1} = j-1) \\ &= \frac{n-1}{n} P(X_{n-1} = j) + \frac{1}{n} P(X_{n-1} = j-1) \end{aligned}$$

5 X_1 suit la loi certaine de valeur 1, donc $G_1(t) = t$.
Pour tout $n \geq 2$:

$$\begin{aligned} G_n(t) &= \sum_{j=1}^{n} P(X_n = j) t^j \\ &= \sum_{j=1}^{n} \Big(\frac{n-1}{n} P(X_{n-1} = j) + \frac{1}{n} P(X_{n-1} = j-1) \Big) t^j \\ &= \frac{n-1}{n} \sum_{j=1}^{n} P(X_{n-1} = j) t^j + \frac{1}{n} \sum_{j=1}^{n} P(X_{n-1} = j-1) t^j \\ &= \frac{n-1}{n} G_{n-1}(t) + \frac{1}{n} \sum_{j=0}^{n-1} P(X_{n-1} = j) t^{j+1} \\ &= \frac{n-1}{n} G_{n-1}(t) + \frac{t}{n} G_{n-1}(t) \\ &= \frac{n-1+t}{n} G_{n-1}(t) \end{aligned}$$

On a utilisé au passage que $P(X_{n-1} = 0) = P(X_{n-1} = n) = 0$.

6 La question précédente permet de d'exprimer la fonction génératrice de X_n en fonction de n :

$$G_n(t) = G_1(t) \prod_{k=2}^{n} \frac{k-1+t}{k} = \frac{1}{n!} \prod_{i=0}^{n-1} (i+t)$$

Puisque la fonction génératrice d'une variable aléatoire caractérise la loi de cette variable aléatoire, trouver G_n suffit à pouvoir prétendre avoir trouvé la loi de X_n.

7 On sait que $G_n(1) = 1$ et que $E(X_n) = G_n'(1)$ (il n'y a ici aucun problème de convergence puisque X_n est une variable aléatoire finie).
En dérivant la relation $G_n(t) = \frac{n-1+t}{n} G_{n-1}(t)$ on obtient :

$$G_n'(t) = \frac{1}{n} G_{n-1}(t) + \frac{n-1+t}{n} G_{n-1}'(t)$$

Et en évaluant cette relation en 1 il vient :

$$E(X_n) = \frac{1}{n} + E(X_{n-1})$$

Puisque $E(X_1) = 1$, une récurrence immédiate assure que :

$$E(X_n) = \sum_{k=1}^{n} \frac{1}{k}$$

Or on sait que $\displaystyle\sum_{k=1}^{n} \frac{1}{k} = \ln n + \gamma + O(1)$ (avec γ la constante d'Euler), ce qui confirme l'équivalent en $\ln n$ pressenti à la question **3.**.