Correction des exercices

Exercice 1 Dans cet exercice on utilise les importations suivantes :

```
import numpy as np
import matplotlib.pyplot as plt
from numpy.polynomial import Polynomial
from scipy.optimize import fsolve
```

1. On réalise le script :

```
def P(n):
    return Polynomial([1] * (2*n+1))

X = np.linspace(-2, 2, 256)
for n in range(1, 11):
    Y = [P(n)(x) for x in X]
    plt.plot(X, Y)
plt.axis([-2, 2, 0, 5])
plt.grid()
plt.show()
```

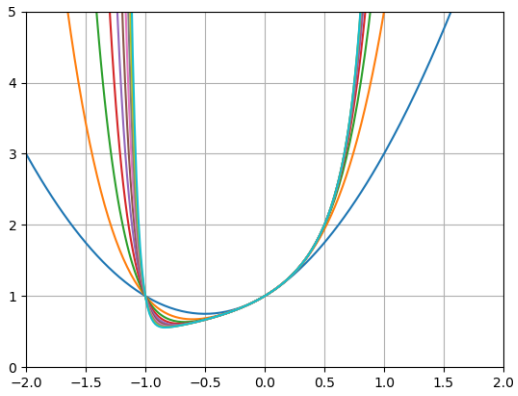


Figure 1 – Le graphe de la question 1

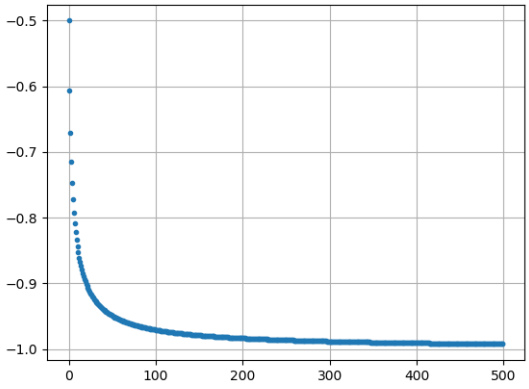


Figure 2 – Le graphe de la question 5

On constate que les minima de ces fonctions polynomiales semblent être compris entre -1 et $-1/2$.

2. $\mathrm{P}_n(x) = \dfrac{x^{2n+1}-1}{x-1}$ soit en dérivant : $\mathrm{P}_n'(x) = \dfrac{u_n(x)}{(x-1)^2}$ avec $u_n(x) = 2nx^{2n+1} - (2n+1)x^{2n} + 1$.

3. On calcule $u_n'(x) = 2n(2n+1)x^{2n-1}(x-1)$ donc les variations de u_n sont données par :

x	$-\infty$		0		1		$+\infty$
$u_n'(x)$		$+$	0	$-$	0	$+$	
$u_n(x)$	$-\infty$	↗	1	↘	0	↗	$+\infty$

On constate que u_n s'annule et change de signe pour une unique valeur $a_n < 0$. Les variations de P_n sont donc données par :

x	$-\infty$		a_n		1		$+\infty$
$\mathrm{P}_n'(x)$		$-$	0	$+$	0	$+$	
$\mathrm{P}_n(x)$	$+\infty$	$\searrow$	$\mathrm{P}_n(a_n)$	$\nearrow$			$+\infty$

4. On définit la fonction :

```
def A(n):
    def u(x):
        return 2*n * x**(2*n+1) - (2*n+1) * x**(2*n) + 1
    return fsolve(u, -1)
```

5. On réalise le script suivant :

```
a = [A(n) for n in range(1, 501)]
plt.plot(a, '.')
plt.grid()
plt.show()
```

qui laisse présager que (a_n) décroit et tend vers -1.

6. On a $u_n(-1) = -4n < 0$ et $u_n(0) = 1 > 0$ donc $a_n \in]-1,0[$. Ainsi, $2n+1 \leqslant 2n+1-2na_n \leqslant 4n+1$, ce qui conduit à l'équivalent $\ln(2n+1-2na_n) \sim \ln(n)$.

On a $u_n(a_n) = 0$, soit $a_n^{2n}\big(2n+1-2na_n\big) = 1$, et donc $\ln(2n+1-2na_n) = -2n\ln|a_n|$. On en déduit que $\ln|a_n| \sim -\dfrac{\ln(n)}{2n}$ puis que $\lim a_n = -1$.

7. $|a_n| = 1 - h_n$ donc de l'équivalent précédent on tire $h_n \sim \dfrac{\ln(n)}{2n}$.

Exercice 2 Dans cet exercice on utilisera les importations suivantes :

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

1. On a $\mathbb{E}(\mathrm{X}_i) = 0$ et $\mathbb{V}(\mathrm{X}_i) = 1$ donc par linéarité de l'espérance, $\mathbb{E}(\mathrm{S}_n) = 0$, et par indépendance des (X_i), $\mathbb{V}(\mathrm{S}_n) = n$. Par indépendance des (X_i) on a $\mathbb{E}(\mathrm{P}_n) = 0$, et sachant que $\mathrm{P}_n^2 = 1$ on en déduit que $\mathbb{V}(\mathrm{P}_n) = 1$.

2. Posons $\mathrm{V}_n = \begin{pmatrix} \mathbb{P}(\mathrm{P}_n = 1) \\ \mathbb{P}(\mathrm{P}_n = -1) \end{pmatrix}$. Les événements $[\mathrm{X}_n = 1]$ et $[\mathrm{X}_n = -1]$ forment un système complet d'événements donc

$$\begin{cases} \mathbb{P}(\mathrm{P}_n = 1) = \mathbb{P}(\mathrm{P}_{n-1} = 1)\mathbb{P}(\mathrm{X}_n = 1) + \mathbb{P}(\mathrm{P}_{n-1} = -1)\mathbb{P}(\mathrm{X}_n = -1) \\ \mathbb{P}(\mathrm{P}_n = -1) = \mathbb{P}(\mathrm{P}_{n-1} = 1)\mathbb{P}(\mathrm{X}_n = -1) + \mathbb{P}(\mathrm{P}_{n-1} = -1)\mathbb{P}(\mathrm{X}_n = 1) \end{cases} \iff \mathrm{V}_n = \frac{1}{2}\begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}\mathrm{V}_{n-1}$$

On calcule $\begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}^n = 2^{n-1}\begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}$ donc $\mathrm{V}_n = \dfrac{1}{2}\begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}\mathrm{V}_0$ avec $\mathrm{V}_0 = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$ donc $\mathbb{P}(\mathrm{P}_n = 1) = \mathbb{P}(\mathrm{P}_n = -1) = \dfrac{1}{2}$.

On a $\mathbb{P}(\mathrm{S}_n = n$ et $\mathrm{P}_n = -1) = 0$ alors que $\mathbb{P}(\mathrm{S}_n = n)\mathbb{P}(\mathrm{P}_n = -1) = \dfrac{1}{2^{n+1}}$ donc P_n et S_n ne sont pas indépendantes.

3. On définit les trois fonctions, puis une fonction d'affichage :

```
def S(X):
    s = 1
    for x in X:
        s += x
    return s

def P(X):
    p = 1
    for x in X:
        p *= x
    return p
```

```
def D(X):
    n = len(X)
    M = np.ones((n, n))
    for k in range(n):
        M[k, k] += X[k]
    return alg.det(M)

def affiche(X):
    print(S(X), P(X), D(X))
```

4. On réalise le script suivant :

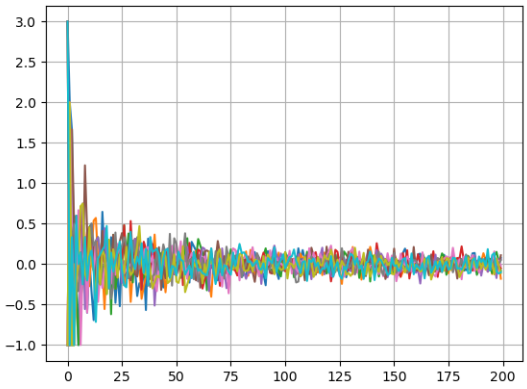
```
for _ in range(20):
    X = 2 * rd.randint(2, size=10) - 1
    affiche(X)
```

qui nous convainc que $A_n = P_n(S_n + 1)$. Pour le prouver on développe en utilisant la n-linéarité du déterminant :

$$D(X_1,\dots,X_n) = \begin{vmatrix} X_1 & & \\ & \ddots & \\ & & X_n \end{vmatrix} + \sum_{k=1}^{n} \begin{vmatrix} X_1 & & 1 & & \\ & \ddots & \vdots & & \\ & & 1 & & \\ & & \vdots & \ddots & \\ & & 1 & & X_n \end{vmatrix} = P_n + \sum_{k=1}^{n} \prod_{i\neq k} X_i = P_n + \sum_{k=1}^{n} X_k P_n = P_n(1+S_n).$$

5. On réalise le script :

```
for _ in range(10):
    U = []
    for n in range(1, 201):
        X = 2 * rd.randint(2, size=n) - 1
        U.append(P(X) * (S(X) + 1) / n)
    plt.plot(U)
plt.grid()
plt.show()
```



6. $\mathbb{E}(A_n) = \mathbb{E}(P_n(1+S_n)) = \mathbb{E}(P_n) + \mathbb{E}(P_nS_n) = \mathbb{E}(P_nS_n) = \sum_{k=1}^{n} \prod_{i\neq k} \mathbb{E}(X_i) = 0.$

$\mathbb{E}(A_n^2) = \mathbb{E}((1+S_n)^2) = 1 + 2\mathbb{E}(S_n) + \mathbb{E}(S_n^2) = 1 + \mathbb{E}(S_n^2) = 1 + \mathbb{V}(S_n) = n+1$ donc $\mathbb{V}(A_n) = n+1$.

D'après l'inégalité de Bienaymé-Tchebychev, $\mathbb{P}\big(|A_n| \geqslant \epsilon n\big) \leqslant \dfrac{\mathbb{V}(A_n)}{\epsilon^2 n^2} = \dfrac{n+1}{\epsilon^2 n^2}$ donc $\lim\limits_{n\to+\infty} \mathbb{P}\big(|A_n| \geqslant \epsilon n\big) = 0.$

Exercice 3 Dans cet exercice on utilisera les importations suivantes :

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

1. Application directe du critère spécial relatif aux séries alternées.
2. *a)* On réalise le script suivant :

```
s = 0
for n in range(1, 1001):
    s += (-1)**n * np.log(1 + 1 / n)
print(s, np.log(2 / np.pi))
```

```
-0.45108320487303755 -0.4515827052894548
```

b) On sépare les termes pairs et impairs :

$$\sum_{n=1}^{2N} a_n = \sum_{k=1}^{N} \ln\left(1+\frac{1}{2k}\right) - \sum_{k=1}^{N} \ln\left(1+\frac{1}{2k-1}\right) = \sum_{k=1}^{N} \ln\left(\frac{(2k-1)(2k+1)}{(2k)^2}\right) = \ln\left(\frac{1}{2N+1}\prod_{k=1}^{N}\left(\frac{2k+1}{2k}\right)^2\right) = \ln\left(\frac{(2N+1)!(2N)!}{2^{4N}(N!)^4}\right)$$

La formule de Stirling permet de conclure :

$$\frac{(2N+1)!(2N)!}{2^{4N}(N!)^4} \sim \frac{(2N+1)4\pi N\,e^{-4N}(2N)^{4N}}{2^{4N}(2\pi N)^2\,e^{-4N}\,N^{4N}} = \frac{2N+1}{\pi N} \sim \frac{2}{\pi} \qquad \text{donc} \qquad \sum_{n=1}^{+\infty} a_n = \ln\left(\frac{2}{\pi}\right).$$

3. $|a_n| \sim \dfrac{1}{n}$ donc $\lim\left|\dfrac{a_{n+1}}{a_n}\right| = 1$ et d'après le critère de d'Alembert le rayon de convergence est égal à 1.

4. *a)* On réalise le script suivant :

```
def f(x):
    s = 0
    for n in range(1, 1001):
        s += (-1)**n * np.log(1 + 1 / n) * x**n
    return s

X = np.linspace(-1, 1, 256)
Y = [f(x) for x in X]
plt.plot(X, Y)
plt.plot([1], [np.log(2 / np.pi)], 'o')
plt.grid()
plt.show()
```

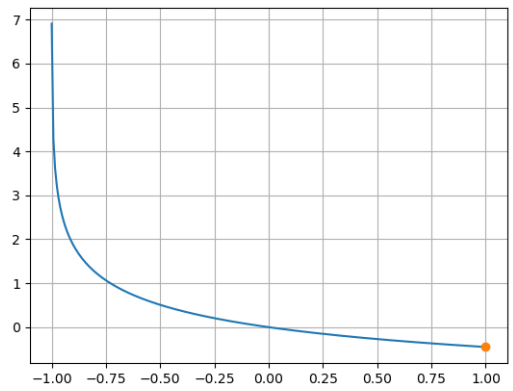


Figure 3 – Le graphe de la question 4.a

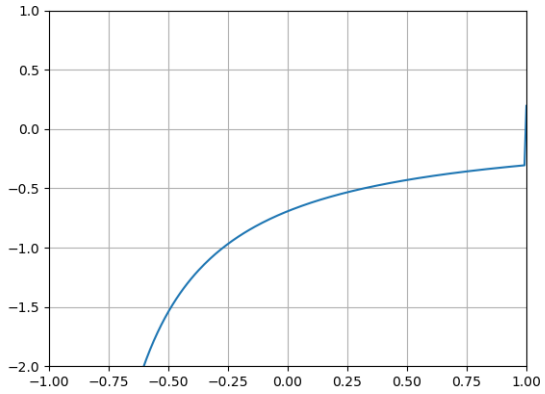


Figure 4 – Le graphe de la question 5.a

b) On note $f_n(x) = (-1)^n a_n x^n$. Le critère spécial relatif aux séries alternées s'applique pour $x \in [-1,1]$ et fournit la majoration du reste $|R_n(x)| \leqslant |f_{n+1}(x)| \leqslant |a_n|$. On en déduit que $\|R_n\|_\infty \leqslant |a_n|$ et ainsi que la convergence est uniforme sur $[0,1]$. La fonction f est donc continue sur cet intervalle, et en particulier $\lim\limits_{x\to 1} f(x) = f(1) = \ln\left(\dfrac{2}{\pi}\right)$.

5. *a)* On réalise le script suivant :

```
def df(x):
    s = 0
    for n in range(1, 1001):
        s += (-1)**n * n * np.log(1 + 1 / n) * x**(n-1)
    return s

X = np.linspace(-1, 1, 256)
Y = [df(x) for x in X]
plt.plot(X, Y)
plt.axis([-1, 1, -2, 1])
plt.grid()
plt.show()
```

On constate que bien que la série définissant f' ne soit pas convergente pour $x = 1$ (le terme général na_n ne tend pas vers 0), il semble que f' possède une limite en 1.