PlanchesPreparationOralCentraleMaths2_1718

June 19, 2018

1 Planche 2

```
In [5]: from numpy.polynomial import Polynomial
        from math import floor  # Partie entière
        import matplotlib.pyplot as plt
```

1.0.1 Question a

On sait très bien que l'espérance de U_1 vaut $\frac{N+1}{2}$. Sa fonction génératrice est donnée par

$$\forall t \in [-1,1], \qquad G_{U_1}(t) = \sum_{k=1}^{N} t^k \mathbb{P}(U_1 = k) = \frac{1}{N}\sum_{k=1}^{N} t^k.$$

1.0.2 Question b

L'indépendance des U_k donne $G_{S_n} = G_{U_1} \times \cdots \times G_{U_n} = (G_{U_1})^n$ donc

$$\forall t \in [-1,1], \qquad G_{U_n}(t) = \left(\frac{1}{N}\sum_{k=1}^{N} t^k\right)^n.$$

1.0.3 Question c

```
In [2]: P = [0] + [1]*4
        P = Polynomial(P)**6
        print(sum(list(P)[0:10]))
```

```
84.0
```

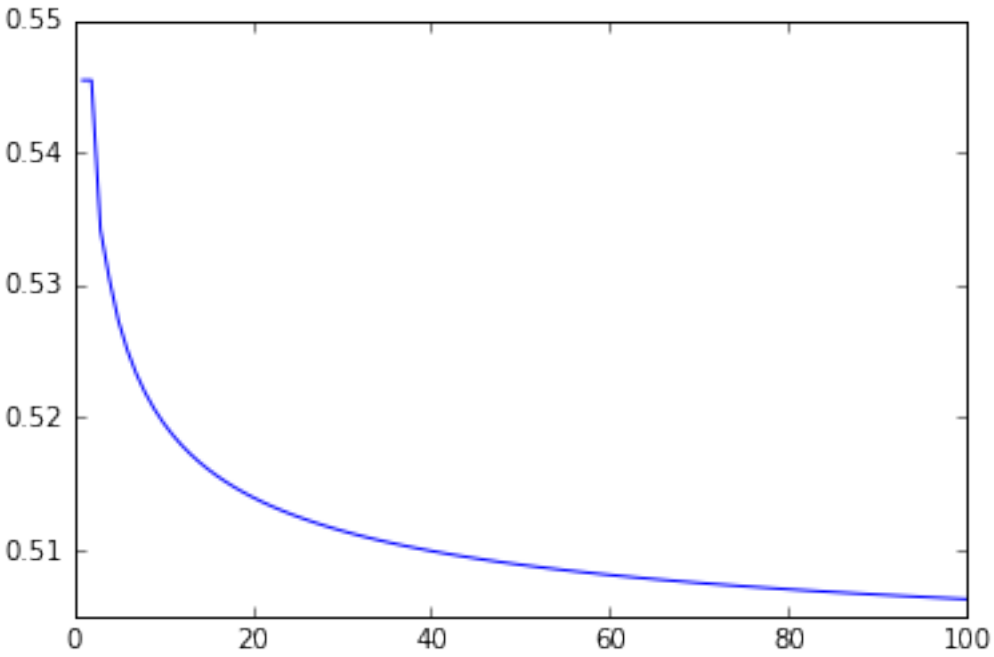
1.0.4 Question d

```
In [8]: def calcul_u_n(n_min=1, n_max=100, N=11):
            """ Renvoie la liste des u_n pour n variant de n_min à n_max.
            """
            m = (N+1)/2
            resultat = [0] * (n_max-n_min +1)
            gU1 = [0] + [1/N]*N
```

```
    gU1 = Polynomial(gU1)
    gSn = gU1  # Initialisation
    for n in range(n_min, n_max+1):
        borne = floor(n*m)
        resultat[n-n_min] = sum(list(gSn)[0:borne+1])
        gSn = gSn * gU1
    return resultat

# Passons à la représentation graphique

n_min = 1
n_max = 100
x = range(n_min, n_max+1)
y = calcul_u_n(n_min, n_max)
plt.plot(x, y)
plt.show()
```



Ça semble décroître et tendre vers $\frac{1}{2}$.

1.0.5 Question e

Posons $V_n = N + 1 - U_n$. On remarque que V_n a la même loi que U_n. De plus, les V_n sont mutuellement indépendantes. Posons ensuite $T_n = V_1 + \cdots + V_n$. On observe alors l'égalité

$$T_n = \sum_{k=1}^{n} (N + 1 - u_k) = n(N+1) - S_n = 2nm - S_n$$

puis

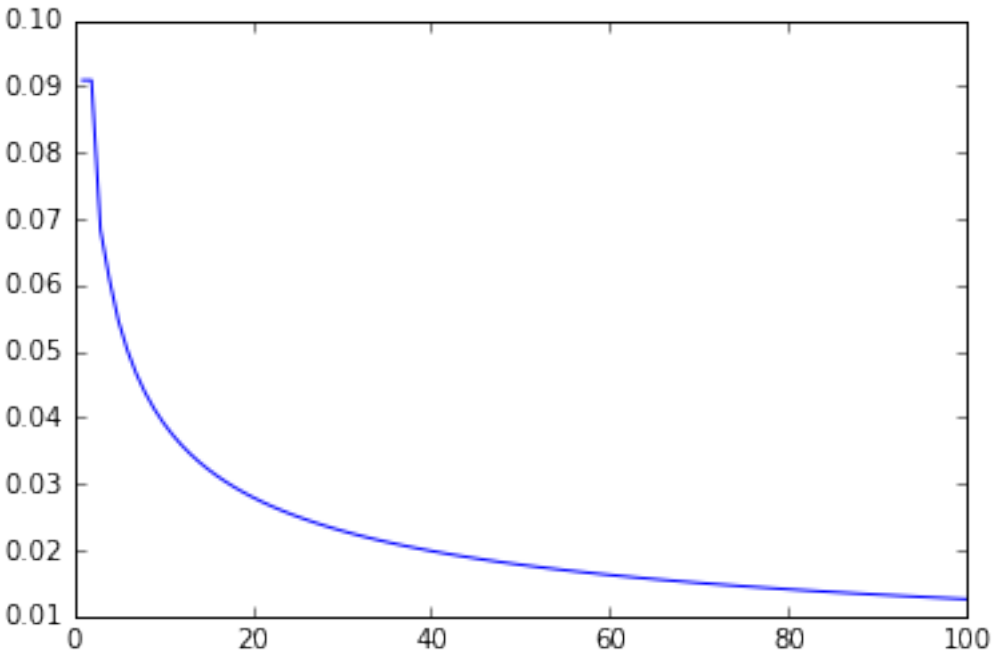
$$u_n = \mathbb{P}(T_n \geq nm) = v_n + w_n.$$

1.0.6 Question f

```
In [11]: def calcul_w_n(n_min=1, n_max=100, N=11):
             """ Renvoie la liste des w_n pour n variant de n_min à n_max.
             """
             m = (N+1)//2  # Cette fois, on part du principe que c'est un entier.
             resultat = [0] * (n_max-n_min +1)
             gU1 = [0] + [1/N]*N
             gU1 = Polynomial(gU1)
             gSn = gU1  # Initialisation
             for n in range(n_min, n_max+1):
                 resultat[n-n_min] = list(gSn)[n*m]
                 gSn = gSn * gU1
             return resultat

         # Passons à la représentation graphique

         n_min = 1
         n_max = 100
         x = range(n_min, n_max+1)
         y = calcul_w_n(n_min, n_max)
         plt.plot(x, y)
         plt.show()
```



Il semble que w_n tende vers 0.

1.0.7 Question g

Remarquons l'égalité $u_n + v_n = 1$, qui donne ensuite $u_n = \frac{1-w_n}{2}$. En admettant que w_n tend vers 0, on en déduit que u_n tend vers $\frac{1}{2}$.

1.0.8 Question h

Une suite $(i_1, \ldots, i_n)$ d'entiers strictement positifs tels que $i_1 + \ldots + i_n = k)$ peut être représentée par un diagramme points-bâtons : on constitue des groupes de i_1 points puis i_2 points et ainsi de suite, en délimitant deux groupes de points consécutifs par un bâton.

Ainsi, la suite $(2, 4, 1, 3, 2)$, dont la somme vaut 12, est représentée par le diagramme

$$\cdot\cdot|\cdot\cdot\cdot\cdot|\cdot|\cdot\cdot\cdot|\cdot\cdot$$

On se convaincra facilement que cette représentation est bijective. Le cardinal $S_{n,k}$ est donc le nombre de diagrammes points-bâtons comportant k points et $n-1$ bâtons, avec la contrainte que les bâtons ne soient pas au début ni à la fin et qu'ils ne soient pas consécutifs.

Ainsi, les $n-1$ bâtons doivent être placés à des emplacements distincts parmi les $k-1$ interstices entre les points. Cela donne le cardinal

$$S_{n,k} = \binom{k-1}{n-1}.$$

1.0.9 Question i

On développe la formule de la question b.

$$G_{S_n}(t) = \frac{1}{N^n}\left(\sum_{i_1=1}^{N} t^{i_1}\right) \times \cdots \times \left(\sum_{i_n=1}^{N} t^{i_n}\right) = \frac{1}{N^n} \sum_{1 \leq i_1, \ldots, i_n \leq N} t^{i_1 + \cdots + i_n}.$$

On connaît aussi l'expression

$$G_{S_n}(t) = \sum_{k=0}^{+\infty} \mathbb{P}(S_n = k) t^k.$$

Par unicité des coefficients du développement en série entière (ou d'un polynôme), on obtient

$$\mathbb{P}(S_n = k) \leq \frac{S_{n,k}}{N^n} = \frac{1}{N^n}\binom{k-1}{n-1}.$$

1.0.10 Question j

Extrapolons la question manquante : démontrer les conjectures précédentes.

Il s'agit de prouver que $\mathbb{P}(S_n = n(N+1)/2)$ tend vers 0. J'imagine que ça se fait bien en exploitant l'équivalent de Stirling mais j'estime ne pas avoir le temps de rentrer dans le détail de ce calcul.

2 Planche 6

```
In [2]: import numpy as np
        import matplotlib.pyplot as plt
        import scipy.integrate as integr
```

2.0.1 Question a

Il s'agit bien sûr de la fonction $x : t \mapsto \frac{a}{\omega}\sin(\omega t)$.

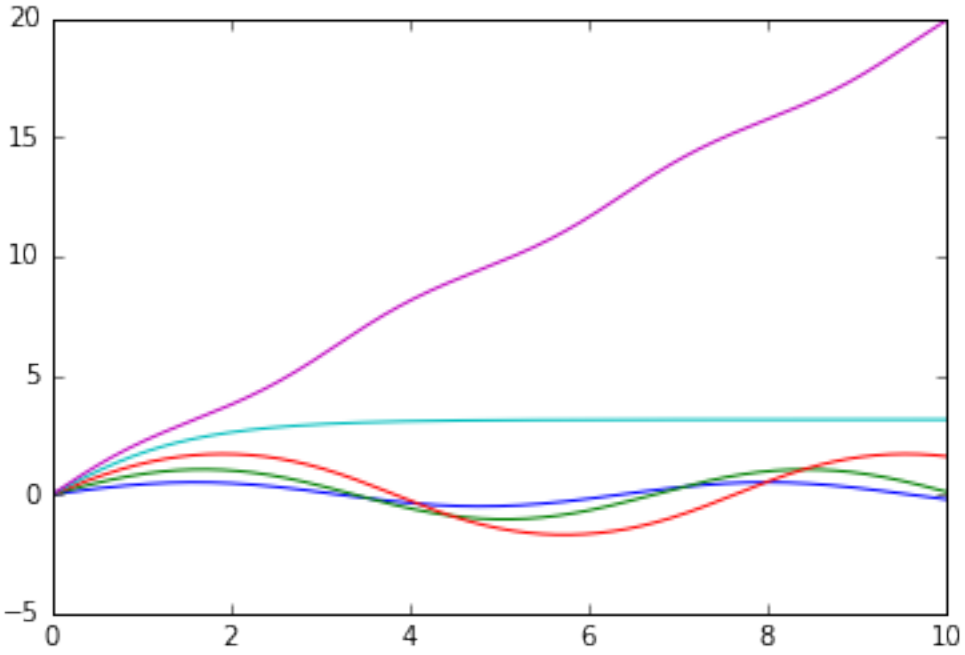
2.0.2 Question b

En posant $X(t) = \begin{pmatrix} f(t) \\ f'(t) \end{pmatrix}$ et $g(x,y) = (y, -\omega^2 \sin(x))$, le problème se reformule en

$$\forall t \in I, \quad X'(t) = g(X(t)), \qquad X(0) = \begin{pmatrix} 0 \\ a \end{pmatrix}.$$

```
In [3]: # On prend omega = 1 pour les applications numériques.
        def g(x, t):
            return np.array([x[1], -np.sin(x[0])])

        t = np.arange(0, 10.01, 0.01)
        liste_a = [0.5, 1, 1.5, 2, 2.5]
        for a in liste_a:
            x = integr.odeint(g, np.array([0, a]), t)
            #plt.plot(x[:, 0], x[:, 1])
            plt.plot(t, x[:, 0])
        plt.show()
```



2.0.3 Question c.i

La fonction F est une primitive sur $\mathbb{R}$ de la fonction $h : u \mapsto \frac{1}{\sqrt{1-r^2\sin^2(\omega u)}}$, qui est de classe $\mathcal{C}^\infty$ sur $\mathbb{R}$, donc la fonction F est de classe $\mathcal{C}^\infty$ sur $\mathbb{R}$.

La fonction h est strictement positive donc F est strictement croissante. La fonction h est paire, donc la fonction F, qui est sa primitive nulle en 0, est impaire.

La monotonie de F entraîne l'existence d'une limite en $+\infty$. La fonction $u \mapsto \sin^2(\omega u)$ est π/ω-périodique. On en déduit l'égalité suivante pour tout entier naturel n

$$F(n\pi/\omega) = nF(\pi/\omega).$$

Le nombre $F(\pi/\omega)$ est strictement positif (intégrale d'une fonction continue, positive, différente de la fonction nulle) donc la suite de terme général $F(n\pi/\omega)$ tend vers $+\infty$.

La limite de F en $+\infty$ est donc $+\infty$. Par imparité, la limite de F en $-\infty$ est $-\infty$.

La continuité, la stricte croissance et les limites aux bornes prouvent que F est une bijection de $\mathbb{R}$ sur $\mathbb{R}$.

2.0.4 Question c.ii

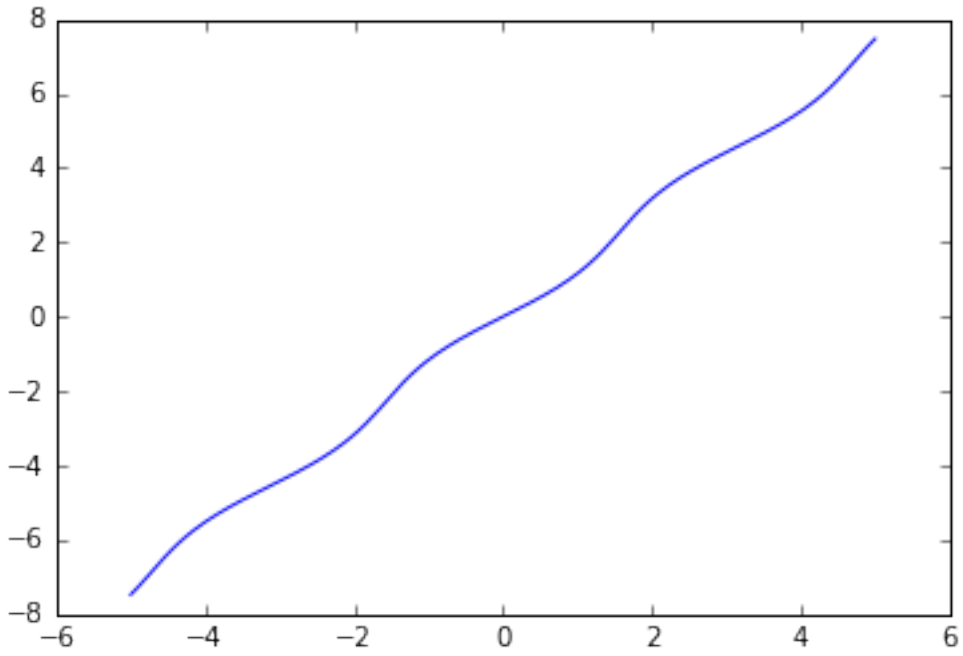
La classe $\mathcal{C}^\infty$ de F a déjà été justifiée à la question précédente. Le fait que la fonction F' ne s'annule pas permet de conclure que F^{-1} est également de classe $\mathcal{C}^\infty$.

2.0.5 Question c.iii

Pour la construction graphique, on prend $\omega = 1$.

```
In [5]: def F(t, r=0.9, omega=1):
            def h(u):
                return (1-(r*np.sin(omega*u))**2)**(-0.5)
            valeur, precision = integr.quad(h, 0, t)
            return valeur

        x = np.linspace(-5, 5, 200)
        y = [F(t) for t in x]
        plt.plot(x, y)
        plt.show()
```




```
In [57]: F(3.569584225335946)  # J'ai un peu honte, mais j'ai fait une dichotomie à la main pour

Out[57]: 5.0
```

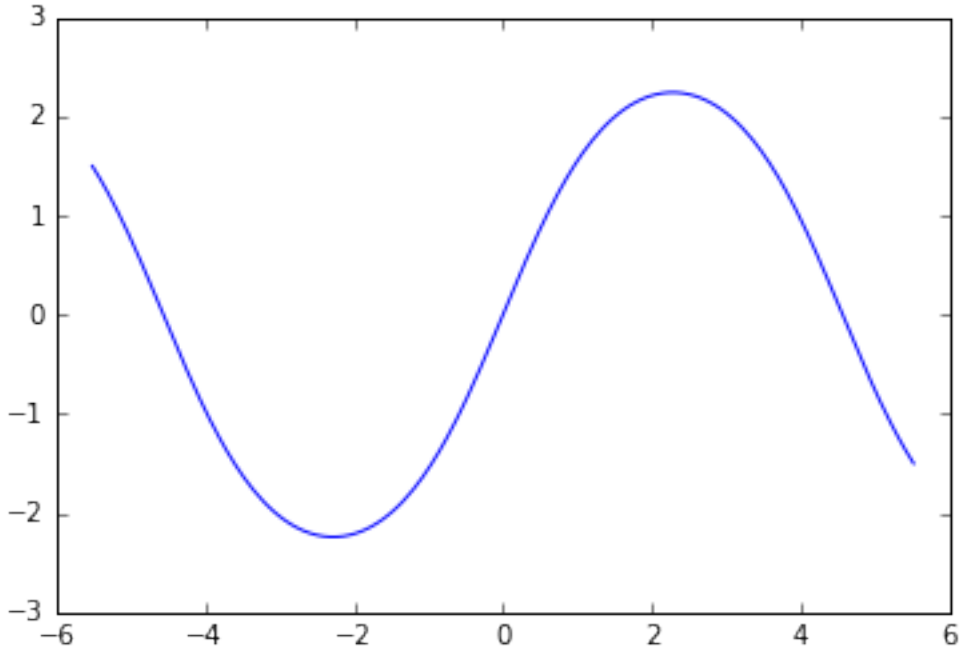
2.0.6 Question d

En fait, il est hors de question de programmer le calcul de la fonction F^{-1}. En posant $u = F^{-1}(t)$, le graphe de la fonction φ est paramétré par

$$x(u) = F(u), \qquad y(u) = 2\text{Arcsin}(r\sin(\omega u)).$$

Pour représenter la fonction φ sur $[-5,5]$, en toute rigueur, il faudrait calculer $F^{-1}(5)$, mais ne chipotons pas et prenons u dans $[-4,4]$. Ça devrait faire le boulot.

```
In [59]: r = 0.9
         omega = 1
         u = np.linspace(-4, 4, 200)
         x = [F(s) for s in u]
         y = 2*np.arcsin(r*np.sin(omega*u))
         plt.plot(x, y)
         plt.show()
```



Ça ressemble un peu à une sinusoïde mais ça n'en est pas strictement une. Pas évident de savoir exactement ce qu'on est censé observer.

2.0.7 Question e

L'égalité $\varphi(0) = 0$ s'obtient directement.

Une première dérivation donne

$$\forall t \in \mathbb{R}, \quad \varphi'(t) = 2\omega(F^{-1})'(t) \times r\cos(\omega F^{-1}(t)) \times \frac{1}{\sqrt{1 - r^2\sin^2(\omega F^{-1}(t))}}.$$

La règle de dérivation des fonctions réciproques donne

$$(F^{-1})'(t) = \frac{1}{F'(F^{-1}(t))} = \sqrt{1 - r^2\sin^2(\omega F^{-1}(t))}.$$

Il reste donc

$$\forall t \in \mathbb{R}, \quad \varphi'(t) = 2\omega \times r \times \cos(\omega F^{-1}(t)) = a \times \cos(\omega F^{-1}(t)).$$

On en tire notamment l'égalité $\varphi'(0) = a$.

Une deuxième dérivation donne

$$\forall t \in \mathbb{R}, \quad \varphi''(t) = -a\omega \times (F^{-1})'(t) \times \sin(\omega F^{-1}(t)) = -a\omega \times \sqrt{1 - r^2\sin^2(\omega F^{-1}(t))} \times \sin(\omega F^{-1}(t)).$$

Un peu de trigonométrie donne ensuite

$$\forall t \in \mathbb{R}, \quad \sin(\varphi(t)) = 2r\sin(\omega F^{-1}(t)) \times \sqrt{1 - r^2\sin^2(\omega F^{-1}(t))}.$$

On utilise pour cela les identités $\sin(2\theta) = 2\sin(\theta)\cos(\theta)$ et $\cos(\mathrm{Arcsin}(s)) = \sqrt{1 - s^2}$.

Bref, tout ceci prouve que la fonction φ est une solution du problème de la question b.

3 Planche 7

```
In [60]: import numpy as np
```

3.0.1 Question a

```
In [61]: def matrice_An(n):
             mat = np.diag(range(1, n+1))
             mat[0, :] = 1  # Première ligne
             mat[:, 0] = 1  # Première colonne
             return mat

In [62]: matrice_An(4)

Out[62]: array([[1, 1, 1, 1],
                [1, 2, 0, 0],
                [1, 0, 3, 0],
                [1, 0, 0, 4]])
```

3.0.2 Question b

En développant selon la dernière colonne puis selon la dernière ligne, on obtient

$$\chi_n = (X-n)\chi_{n-1} - (X-2)\cdots(X-(n-1)).$$

3.0.3 Question c

La formule précédente donne la relation de récurrence $F_n = F_{n-1} - \frac{1}{X-n}$. Un télescopage donne ensuite

$$F_n = F_2 + \sum_{k=3}^{n}(F_k - F_{k-1}) = F_2 - \sum_{k=3}^{n}\frac{1}{X-k}.$$

Le calcul donne $\chi_2 = (X-2)(X-1) - 1$ donc $F_2 = X - 1 - \frac{1}{X-2}$ puis

$$F_n = X - 1 - \sum_{k=2}^{n}\frac{1}{X-k}.$$

On en déduit l'expression

$$\chi_n = \prod_{k=1}^{n}(X-k) - \sum_{k=2}^{n}\prod_{\substack{2\leq j\leq n\\ j\neq k}}(X-j).$$

3.0.4 Question d

La matrice A_n est symétrique, à coefficients réels, donc, d'après le théorème spectral, cette matrice est diagonalisable dans $\mathcal{M}_n(\mathbb{R})$. En particulier, son spectre est inclus dans $\mathbb{R}$.

3.0.5 Question e

```
In [67]: def max_vp(n):
             return max(np.linalg.eigvals(matrice_An(n)))

In [68]: # Testons tout de même cette fonction
         for n in range(10, 110, 10):
             print(n, max_vp(n))
```

```
10 10.1506261283
20 20.0638339217
30 30.0397224698
40 40.028697459
50 50.0224217999
60 60.0183817953
70 70.0155674486
80 80.0134961897
90 90.0119088774
100 100.010654091
```

Tout ceci suggère un développement asymptotique en $\lambda_n = n + \frac{1}{n} + o\left(\frac{1}{n}\right)$ mais l'énoncé n'en demande pas tant.

3.0.6 Question f

Les formules de la question c montrent que $2, \dots, n$ sont des pôles de F_n. Ce ne sont donc pas des racines de χ_n. Les valeurs propres de A_n sont donc exactement les racines de la fraction rationnelle F_n.

Les limites de F_n aux bornes des intervalles $]-\infty, 2[,]2, 3[, \dots,]n-1, n[,]n, +\infty[$ sont à chaque fois $-\infty$ (à la borne de gauche) et $+\infty$ (à la borne de droite). Par continuité de F_n sur ces intervalles, on peut appliquer le théorème des valeurs intermédiaires. Chacun de ces n intervalles contient donc au moins une valeur propre de A_n. On sait qu'il y en a au plus n donc chacun de ces intervalles contient exactement une valeur propre de A_n.

En particulier, on obtient l'inégalité $\lambda_n > n$.

Par ailleurs, on trouve $F_n(n+1) = n - \sum_{k=2}^{n} \frac{1}{n+1-k} = n - \sum_{\ell=1}^{n-1} \frac{1}{\ell}$.

On sait que la somme $\sum_{\ell=1}^{n-1} \frac{1}{\ell}$ est de l'ordre de $\ln(n)$. Elle est donc négligeable devant n. On en déduit que $F_n(n+1)$ est positif à partir d'un certain rang n_0.

Comme on l'a déjà remarqué, la limite de F_n en n par valeurs supérieures est $-\infty$. Le théorème des valeurs intermédiaires donne donc, si $n \geq n_0$, l'existence d'une racine de F_n dans l'intervalle $]n, n+1[$. Cette racine ne peut être que λ_n.

Cet encadrement donne $\lambda_n = n + \mathcal{O}(1)$.

4 Planche 3

```
In [6]: import numpy as np
```

4.0.1 Question a

```
In [40]: def decomp(a):
             n, _ = a.shape
             d_inv = np.diag([1/a[i, i] for i in range(n)])
             e = np.zeros((n, n))
             f = np.zeros((n, n))
             for i in range(1, n):
                 for j in range(i):
                     e[i, j] = -a[i, j]
                     f[j, i] = -a[j, i]
             return d_inv, e, f
```

4.0.2 Question b

```
In [41]: def jacobi(A, B, X0, n):
             xn = X0
             d_inv, e, f = decomp(A)
             d1 = d_inv.dot(e+f)
             d2 = d_inv.dot(B)
             for i in range(n):
                 xn = d1.dot(xn) + d2
             return xn
```

4.0.3 Question c

```
In [42]: a = np.array([[2, 3, 1],
                       [-2, 5, 4],
                       [0, 1, 8]])
         b = np.array([[1], [1], [1]])
         X0 = np.copy(b)
         n = 100
         xn_1 = jacobi(a, b, X0, n)
         xn_2 = np.linalg.solve(a, b)
         print(xn_1)
         print(xn_2)
         print(np.sum((xn_1-xn_2)**2))
```

```
[[ 0.16949153]
 [ 0.18644068]
 [ 0.10169492]]
[[ 0.16949153]
 [ 0.18644068]
 [ 0.10169492]]
5.58981907059e-30
```

4.0.4 Question d.i

Les coefficients diagonaux de la matrice A sont non nuls donc voilà.

4.0.5 Question d.ii

Exercice classique. On le trouve même sur Wikipédia. https://fr.wikipedia.org/wiki/Matrice_%C3%A0_diagona

4.0.6 Question d.iii

On connaît l'égalité $(D-E-F)Y = B$. On en tire l'égalité $Y = D^{-1}(E+F)Y + D^{-1}B = MY + D^{-1}B$. On connaît aussi l'égalité $X_{k+1} = MX_k + D^{-1}B$.

Par soustraction, on obtient $X_{k+1} - Y = M(X_k - Y)$.

On remarquera qu'il s'agit exactement de la méthode employée pour exprimer le terme général d'une suite arithmético-géométrique. Et pour cause : la relation de récurrence est du même type.

4.0.7 Question d.iv

Pour tout indice de ligne i, posons $S_i = \sum_{j \neq i} |A(i,j)|$.

Étant donné une colonne U de $\mathcal{M}_{n,1}(\mathbb{R})$, le i-ième coefficient de la colonne MU est donné par

$$(MU)(i) = \frac{1}{A(i,i)} \sum_{j \neq i} (-A_{i,j}\, U(j)).$$

On en déduit la majoration

$$|(MU)(i)| \leq \frac{S_i}{|A_{i,i}|} \times ||U||.$$

Notons ρ le plus grand des nombres $\frac{S_i}{|A_{i,i}|}$ quand i décrit $\{1, \ldots, n\}$. On a alors

$$\forall i \in \{1, \ldots, n\}, \quad |(MU)(i)| \leq \rho \times ||U||$$

donc $||MU|| \leq \rho ||U||$. L'hypothèse de diagonale strictement dominante fait que la constante ρ appartient à $[0, 1[$.

On obtient en particulier la majoration $||X_{k+1} - Y|| \leq \rho\, ||X_k - Y||$.

4.0.8 Question d.v.

En itérant cette inégalité, on obtient $||X_k - Y|| \leq \rho^k\, ||X_0 - Y||$ pour tout entier k.

Le fait que ρ soit dans $[0, 1[$ fait que ρ^k tend vers 0 quand k tend vers $+\infty$. On en déduit que la suite vectorielle $(X_k)_{k \in \mathbb{N}}$ converge vers le vecteur Y et que cette convergence est rapide.

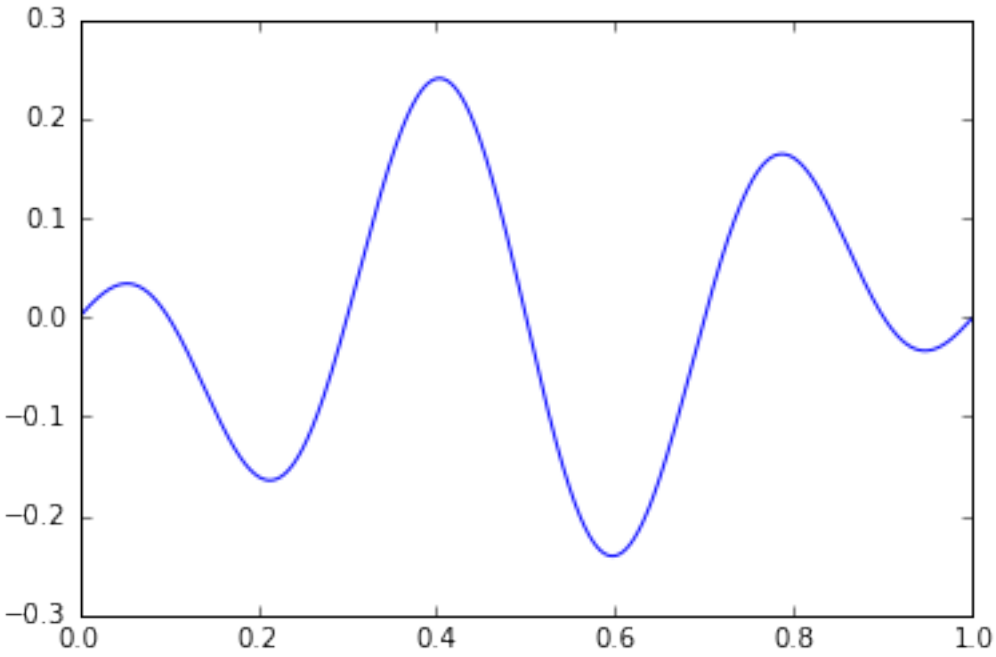
5 Planche 11

```
In [4]: import numpy as np
        import matplotlib.pyplot as plt
        import scipy.integrate as integr
```

```
In [2]: def f(x):
            return x*(1-x)*np.cos(5*np.pi*x)
```

5.0.1 Question a

```
In [7]: t = np.linspace(0, 1, 1000)
        x = f(t)
        plt.plot(t, x)
        plt.show()
        m = max(x)
        print(m)
```




```
0.240325259815
```

5.0.2 Question b

```
In [5]: n = 100
        valeurs_In = [0]*(n+1)
        for k in range(n+1):
            def fk(t):
                return f(t)**k
            valeur, precision = integr.quad(fk, 0, 1)
            valeurs_In[k] = valeur
            print("n={}, In={}".format(k, valeur))
```

```
n=0, In=1.0
n=1, In=-1.9014457691887807e-17
n=2, In=0.016654347487961046
```

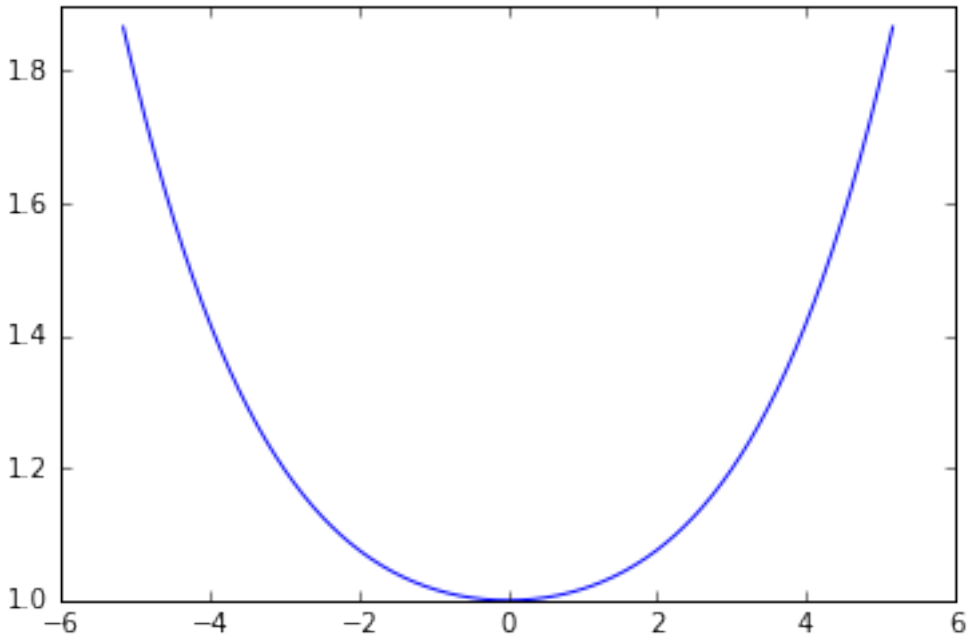
n=3, In=-4.450082874753596e-19
n=4, In=0.0005957180555963488
n=5, In=1.3683065466445082e-20
n=6, In=2.5992751245865593e-05
n=7, In=2.0777251189305275e-21
n=8, In=1.2506931419381345e-06
n=9, In=1.9161248338450193e-22
n=10, In=6.348392615547812e-08
n=11, In=1.2662591350091162e-23
n=12, In=3.328021926998407e-09
n=13, In=8.51214928511243e-25
n=14, In=1.775049098962484e-10
n=15, In=4.9497904006799055e-26
n=16, In=9.568280567114367e-12
n=17, In=2.918227801913279e-27
n=18, In=5.190729820470777e-13
n=19, In=1.6950946165901531e-28
n=20, In=2.8269992754230414e-14
n=21, In=9.493954039086006e-30
n=22, In=1.5434509040783065e-15
n=23, In=5.225682365524988e-31
n=24, In=8.440158922311196e-17
n=25, In=2.9078733503459104e-32
n=26, In=4.620297069729591e-18
n=27, In=1.5916623197137988e-33
n=28, In=2.5311099867562147e-19
n=29, In=8.530887218023643e-35
n=30, In=1.3873572570431e-20
n=31, In=4.666600388816826e-36
n=32, In=7.607620475145411e-22
n=33, In=2.500336217313792e-37
n=34, In=4.173107906185433e-23
n=35, In=1.3457795827121698e-38
n=36, In=2.2898122110399165e-24
n=37, In=7.326151202634649e-40
n=38, In=1.2567715185486313e-25
n=39, In=3.815794906346382e-41
n=40, In=6.899543545840815e-27
n=41, In=2.0669182028740958e-42
n=42, In=3.7886646398438424e-28
n=43, In=1.0930900790702318e-43
n=44, In=2.080892816461796e-29
n=45, In=5.796713981462076e-45
n=46, In=1.1431627138379713e-30
n=47, In=3.0730418972260324e-46
n=48, In=6.281435479540346e-32
n=49, In=1.6173925831815906e-47
n=50, In=3.45223535465069e-33

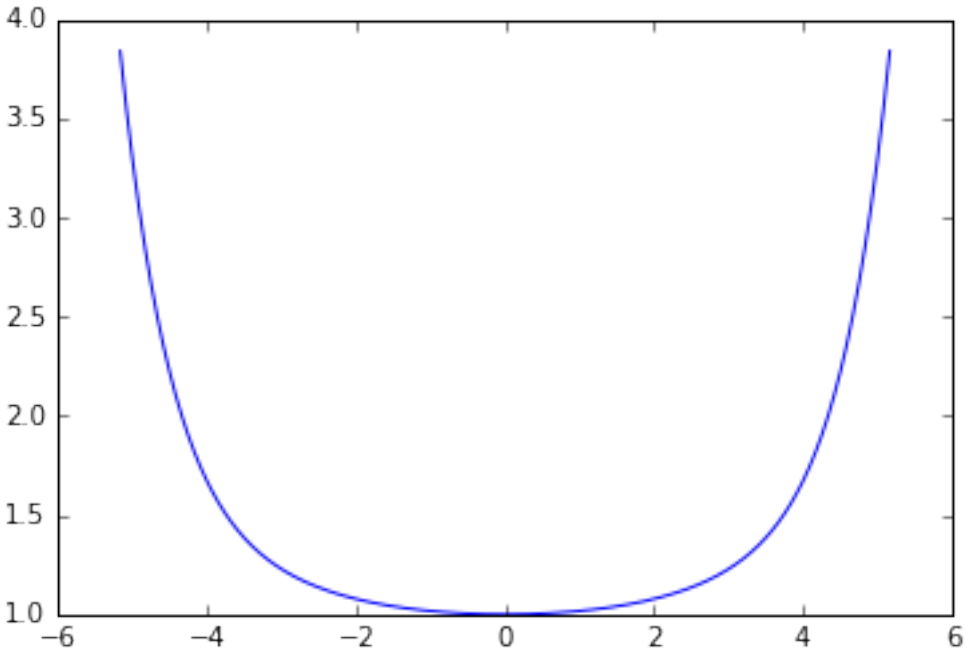
n=51, In=8.592404241198334e-49
n=52, In=1.897714623988622 3e-34
n=53, In=4.50153705375577e-50
n=54, In=1.0433954840693035e-35
n=55, In=2.3692305510259414e-51
n=56, In=5.737902343208905e-37
n=57, In=1.249399075039394e-52
n=58, In=3.1560375695077386e-38
n=59, In=6.459085149420171e-54
n=60, In=1.7362596636546643e-39
n=61, In=3.374149085802018e-55
n=62, In=9.55365132676451e-41
n=63, In=1.7699219906346794e-56
n=64, In=5.2578113852961896e-42
n=65, In=9.296797798418861e-58
n=66, In=2.894144607281656e-43
n=67, In=4.817565345290035e-59
n=68, In=1.59335946119479e-44
n=69, In=2.528302432736675e-60
n=70, In=8.773730934337187e-46
n=71, In=1.3072472831490791e-61
n=72, In=4.832040531198279e-47
n=73, In=6.868435252237374e-63
n=74, In=2.6616517288141302e-48
n=75, In=3.5632813626291605e-64
n=76, In=1.4663747853196721e-49
n=77, In=1.8587983492543467e-65
n=78, In=8.079984509592791e-51
n=79, In=9.589908938132938e-67
n=80, In=4.452937021974754e-52
n=81, In=4.966202843490897e-68
n=82, In=2.4544362288179356e-53
n=83, In=2.590131655597303e-69
n=84, In=1.3530842054223313e-54
n=85, In=1.3378779213236136e-70
n=86, In=7.460440467000329e-56
n=87, In=6.982050402038191e-72
n=88, In=4.114048357361328e-57
n=89, In=3.579868656769771e-73
n=90, In=2.2690199028083058e-58
n=91, In=1.861793005812606e-74
n=92, In=1.251612601956796e-59
n=93, In=9.696838571972072e-76
n=94, In=6.904988893813568e-61
n=95, In=5.007906762507643e-77
n=96, In=3.809923120141227e-62
n=97, In=2.571735558774381e-78
n=98, In=2.1024629715752533e-63

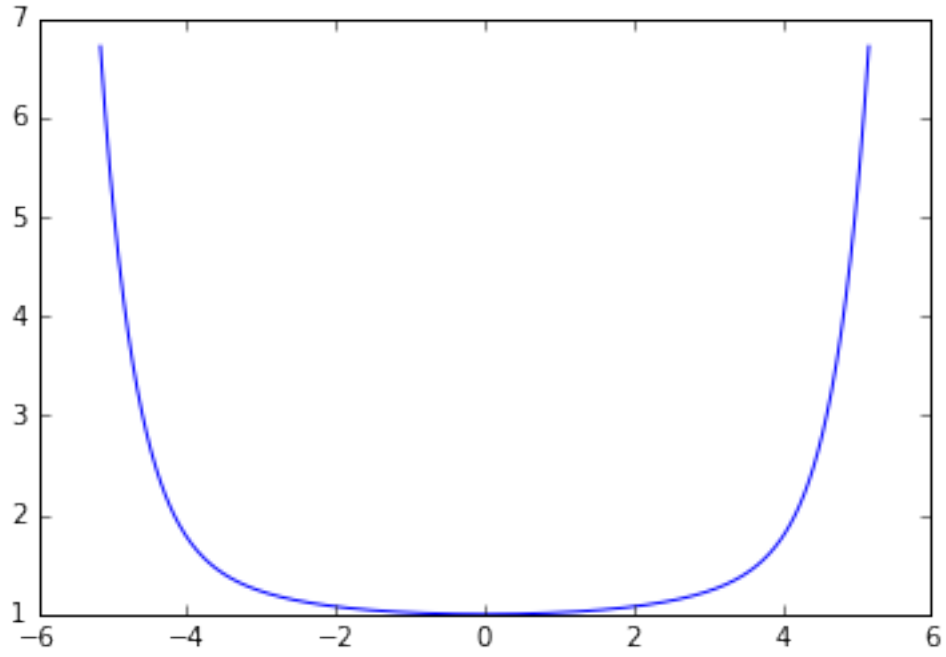
```
n=99, In=1.3332156239777788e-79
n=100, In=1.1603744850285505e-64
```

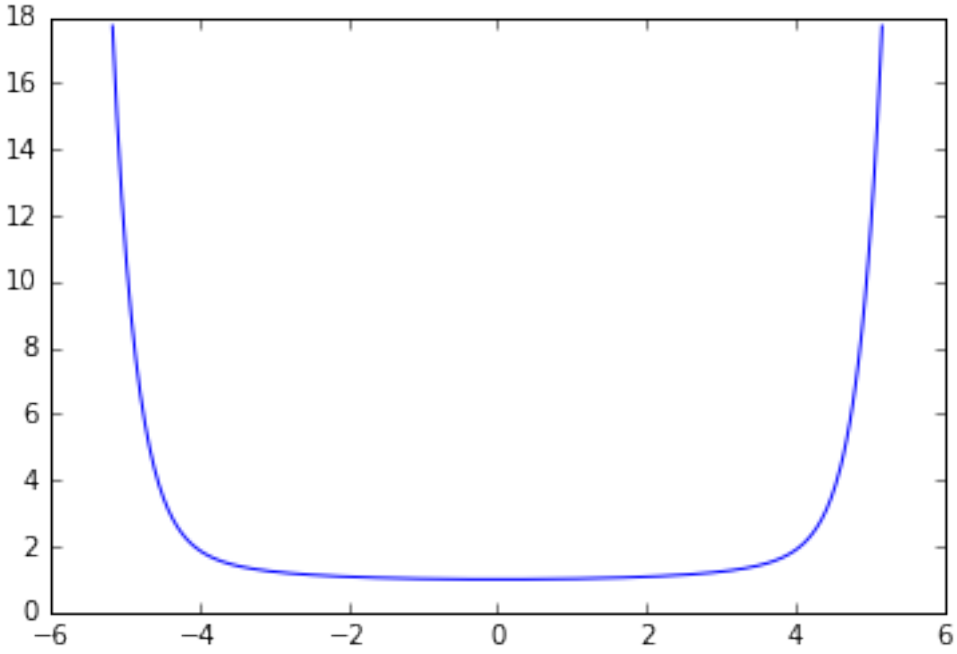
5.0.3 Question c

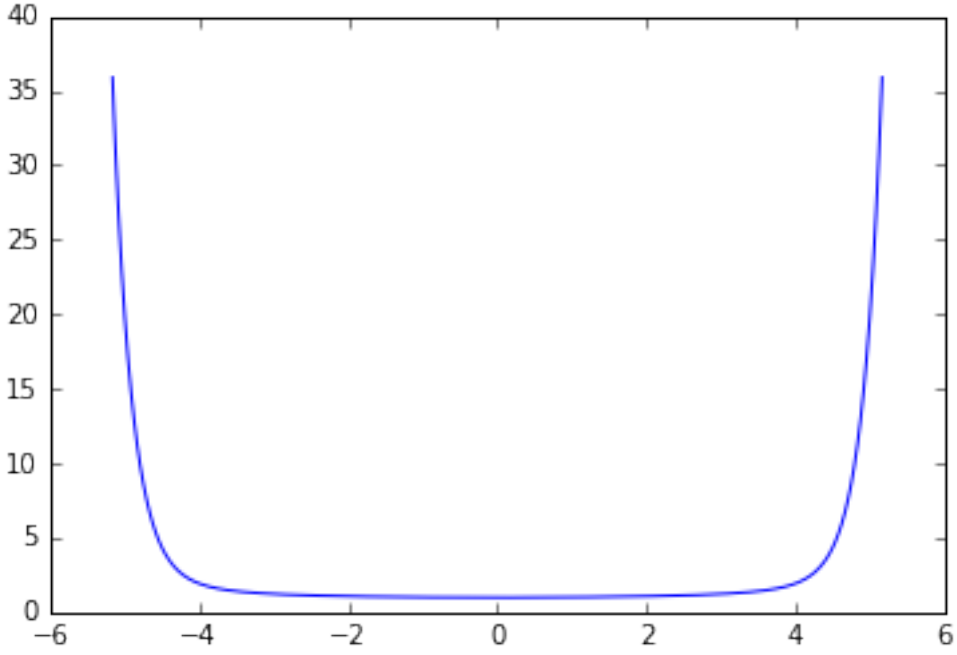
```
In [12]: a = 1 + 1/m
         for n in range(5, 31, 5):
             def Sn(x):
                 return sum(valeurs_In[k]*x**k for k in range(n+1))
             t = np.linspace(-a, a, 500)
             plt.plot(t, Sn(t))
             plt.show()
```

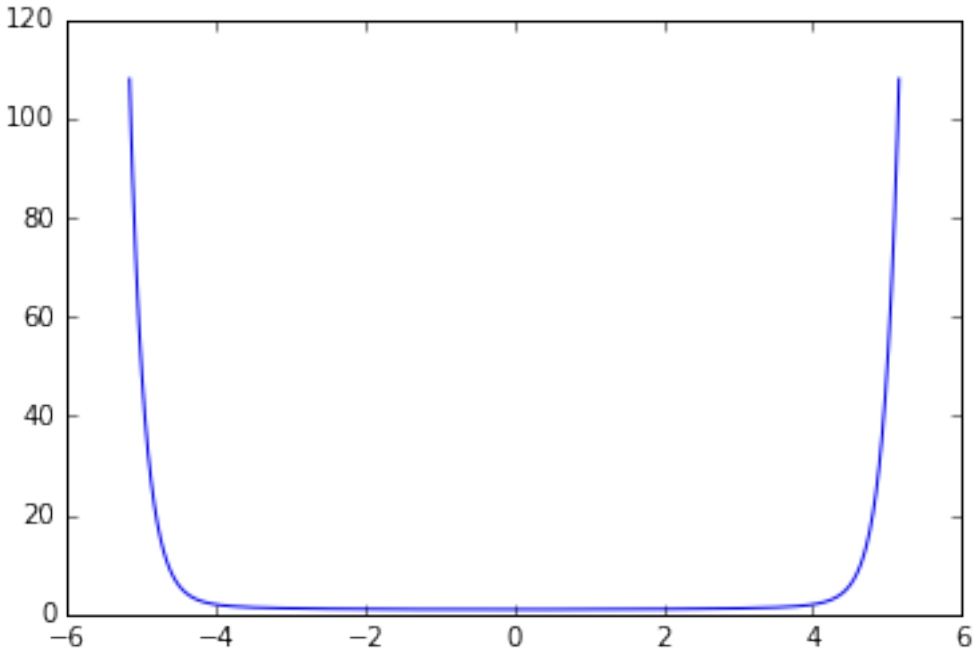



4.0
3.5
3.0
2.5
2.0
1.5
1.0
−6
−4
−2
0
2
4
6


7
6
5
4
3
2
1
−6
−4
−2
0
2
4
6


18
16
14
12
10
8
6
4
2
0
−6
−4
−2
0
2
4
6


40
35
30
25
20
15
10
5
0
−6
−4
−2
0
2
4
6



5.0.4 Question d

Notons R le rayon de convergence cherché. Admettons l'égalité $M = \max\{|f(x)|\,;\, x \in [0,1]\}$.

On connaît déjà l'inégalité $|I_n| \leq M^n$, qui donne l'inégalité

$$R \geq \text{rayon}\left(\sum_{n\geq 0} M^n\, x^n\right) = \frac{1}{M}.$$

Ensuite, prenons ε dans $]0, M[$. Notons y le point où f atteint son maximum sur $[0,1]$. Par continuité de f, il existe α dans $]0, y[$ tel qu'on ait

$$\forall x \in [y-\alpha, y+\alpha], \quad f(x) \geq M - \varepsilon.$$

Pour éviter les embêtements liés aux changements de signe de la fonction f, limitons-nous aux exposants pairs. Pour tout entier n, on remarque la minoration

$$I_{2n} \geq \int_{y-\alpha}^{y+\alpha} f(x)^{2n}\,\mathrm{d}x \geq 2\alpha(M-\varepsilon)^{2n}.$$

On en déduit que si $x = 1/(M-\varepsilon)$, alors la suite de terme général $I_{2n}x^{2n}$ ne tend pas vers 0, donc $R \leq 1/(M-\varepsilon)$.

On a donc prouvé pour tout ε dans $]0, M[$ l'encadrement $\frac{1}{M} \leq R \leq \frac{1}{M-\varepsilon}$.

En faisant tendre ε vers 0, on obtient finalement l'égalité $R = 1/M$.

6 Planche 5

```
In [14]: import numpy as np
         import matplotlib.pyplot as plt
         from math import floor
```

6.0.1 Question 1

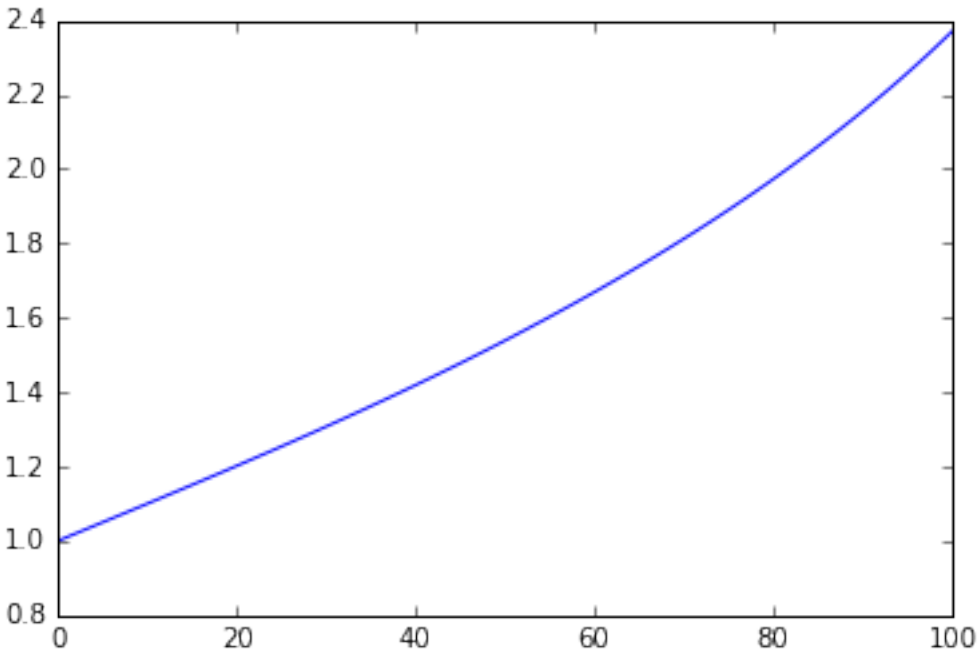
```
In [15]: def indice(x, h=0.01):
             return floor(x/h)
```

6.0.2 Question 2

```
In [17]: h = 0.01
         n = indice(1)
         fti = np.ones(n+1)
         for x in range(0, n):
             i = indice((x/n)**2)
             fti[x+1] = fti[x] + h*fti[i]
```

6.0.3 Question 3

```
In [18]: plt.plot(fti)
         plt.show()
```



6.0.4 Question 4.a

Pour tout x dans $[0,1]$, on trouve, par dérivation terme à terme d'une part, par substitution d'autre part,

$$f'(x) = \sum_{n=1}^{+\infty} n a_n x^{n-1} = \sum_{k=0}^{+\infty} (k+1) a_{k+1} x^k \quad \text{et} \quad f(x^2) = \sum_{n=0}^{+\infty} a_n x^{2n}.$$

Pour tout $n \in \mathbb{N}$, l'unicité du coefficient devant x^{2n} donne en particulier $a_n = (2n+1)a_{2n+1}$.

Pour tout $n \in \mathbb{N}$, l'unicité du coefficient devant x^{2n+1} donne également $a_{2n+2} = 0$, ce qui fait $a_{2k} = 0$ pour tout $k \in \mathbb{N}^*$.

6.0.5 Questions 4.b et 4.c

Récurrences directes.

6.0.6 Question 5

On pose $b_0 = 1$ et $b_p = \prod_{i=1}^{p} \frac{1}{2^i - 1}$ pour tout $p \in \mathbb{N}^*$.

La règle de d'Alembert donne la convergence absolue de la série numérique $\sum b_p x^{2^p - 1}$ pour tout x dans $[-1,1]$.

On note f sa somme et on peut alors vérifier directement que cette fonction est solution du problème posé.

7 Planche 9

```
In [20]: import numpy as np
         from numpy.polynomial import Polynomial
         import scipy.integrate as integr
```

7.0.1 Question a

Le fait que Φ soit un produit scalaire est très classique. La bilinéarité doit être balayée à coup de linéarité de l'intégrale. La symétrie est évidente, de même que la positivité.

Pour le caractère défini positif, on remarque que si $\Phi(P,P)$ est nul, alors la fonction $t \mapsto P(t)^2$ est continue, positive sur $[-2,2]$, d'intégrale nulle, donc cette fonction est identiquement nulle ; on en déduit que P possède une infinité de racines, si bien que c'est le polynôme nul.

```
In [21]: def phi(p, q):
             valeur, precision = integr.quad(p*q, -2, 2)
             return valeur
```

Testons cette fonction en calculant $\Phi(X^2 - 2, X^4 + 3)$.

```
In [22]: p = Polynomial([-2, 0, 1])
         q = Polynomial([3, 0, 0, 0, 1])
         print(phi(p, q))
```

```
2.971428571428573
```

7.0.2 Question b

C'est un calcul direct : si P et pair et Q est impair, le polynôme $P \times Q$ est impair donc son intégrale sur $[-2, 2]$ est nulle.

7.0.3 Question c

On applique bien sûr le procédé de Gram-Schmidt. Pour gagner du temps, on l'applique séparément aux familles $(1, X^2, X^4)$ et (X, X^3), puisqu'elles engendrent des sous-espaces supplémentaires et orthogonaux.

```
In [23]: def normaliser(p):
             """ Renvoie p / || p ||.
             """
             return p / np.sqrt(phi(p, p))
```

```
In [25]: x0 = Polynomial([1])
         x2 = Polynomial([0, 0, 1])
         x4 = Polynomial([0, 0, 0, 0, 1])
         p0 = normaliser(x0)
         p2 = normaliser(x2 - phi(p0, x2) * p0)
         p4 = normaliser(x4 - phi(p2, x4) * p2 - phi(p0, x4) * p0)

         x1 = Polynomial([0, 1])
         x3 = Polynomial([0, 0, 0, 1])
         p1 = normaliser(x1)
         p3 = normaliser(x3 - phi(p1, x3) * p1)

         print([list(p) for p in [p0, p1, p2, p3, p4]])
```

```
[[0.5], [0.0, 0.43301270189221935], [-0.55901699437494734, 0.0, 0.41926274578121053], [0.0, -0.9
```

8 Planche 12

```
In [1]: import numpy as np
        import matplotlib.pyplot as plt
        import scipy.integrate as integr
```

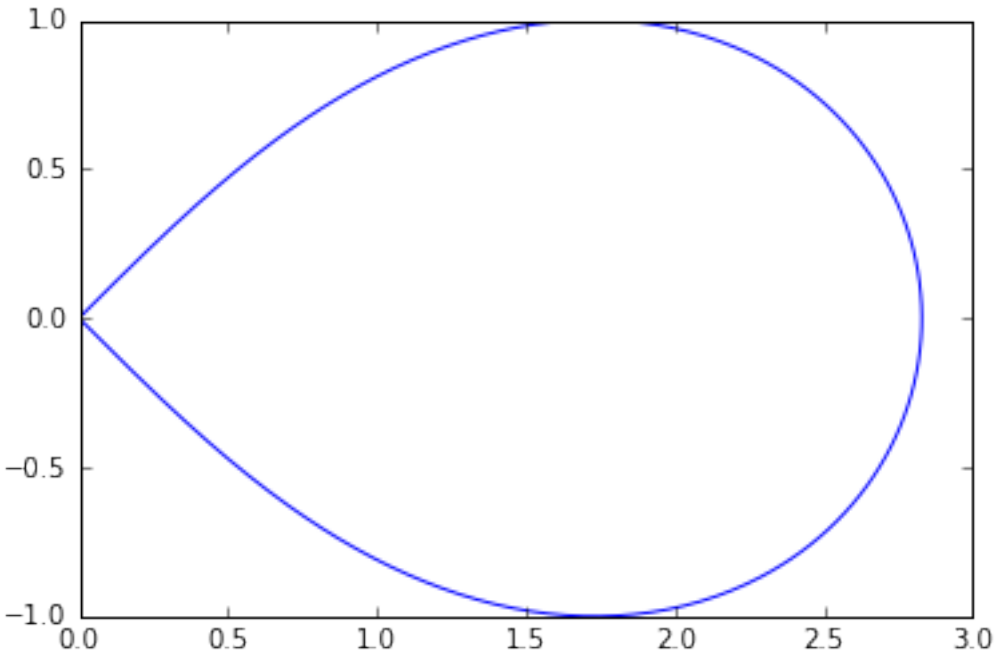
8.0.1 Question a

On trouve la factorisation $\cos^2(t) + 4\cos(t) + 3 = (3 + \cos(t))(1 + \cos(t))$, qui montre que cette quantité est positive pour tout t réel. Les fonctions x et y sont donc définies sur $\mathbb{R}$. Elles sont de plus 2π-périodiques, ce qui permet de réaliser le tracé complet de la courbe en limitant le paramètre au segment $[-\pi, \pi]$.

```
In [2]: def x(t):
            c = np.cos(t)
            return np.sqrt(c**2 + 4*c + 3)
```

```
        def y(t):
            return np.sin(t)
```

```
In [3]: t = np.linspace(-np.pi, np.pi, 500)
        plt.plot(x(t), y(t))
        plt.axis('equal')  # Repère orthonormé
        plt.show()
```



8.0.2 Question b

Les identités $x(-t) = x(t)$ et $y(-t) = -y(t)$ justifient que l'axe des abscisses est un axe de symétrie de cette courbe.

On peut ainsi restreindre l'étude au segment $[0, \pi]$. En traçant la restriction de la courbe à ce segment, on complète alors la trajectoire en lui applique la symétrie orthogonale par rapport à l'axe des abscisses.

8.0.3 Question c

La fonction y est croissante sur $[0, \pi/2]$ puis décroissante sur $[\pi/2, \pi]$.

La fonction cos est décroissante sur $[0, \pi]$. Les fonctions $t \mapsto 1 + \cos(t)$ et $t \mapsto 3 + \cos(t)$ sont décroissantes et à valeurs positives, donc leur produit l'est aussi. La fonction x est donc décroissante sur $[0, \pi]$.

Le calcul des dérivées donne

$$\forall t \in [0, \pi[, \quad x'(t) = -\sin(t) \times \frac{\cos(t) + 2}{x(t)}$$

et

$$\forall t \in [0, \pi], \quad y'(t) = \cos(t).$$

Les fonctions x' et y' n'ont aucun point d'annulation : cette courbe ne présente aucun point singulier.

8.0.4 Question d

Pour tout t dans $]0, \pi[$, la pente de la tangente au point de paramètre t vaut

$$m(t) = \frac{y'(t)}{x'(t)} = -\frac{\sqrt{(1+\cos(t))(3+\cos(t))}}{\tan(t)(2+\cos(t))}.$$

Dans cette expression, le quotient $\frac{-\sqrt{3+\cos(t)}}{2+\cos(t)}$ tend vers $-\sqrt{2}$ quand t tend vers π. Posons maintenant $q(t) = \frac{\sqrt{1+\cos(t)}}{\tan(t)}$.

Une substitution donne $q(\pi + s) = \frac{\sqrt{1-\cos(s)}}{\tan(s)}$. Quand s tend vers 0, ce quotient est équivalent à $\frac{|s|/\sqrt{2}}{s}$. Le quotient $q(t)$ tend donc vers $-1/\sqrt{2}$ lorsque t tend vers π par valeurs inférieures et la pente $m(t)$ tend vers 1.

La demi-tangente correspondante en l'origine a pour équation $y = x$. L'autre demi-tangente a pour équation $y = -x$ par symétrie.

8.0.5 Question e

La longueur de l'arc L est donnée par

$$\ell = \int_{-\pi}^{\pi} \sqrt{x'(t)^2 + y'(t)^2}\, \mathrm{d}t.$$

```
In [4]: def dx(t):
            return -np.sin(t) * (np.cos(t) + 2) / x(t)

        def dy(t):
            return np.cos(t)

        def ds(t):
            return np.sqrt(dx(t)**2 + dy(t)**2)
```

```
In [5]: longueur = integr.quad(ds, -np.pi, np.pi)
        print(longueur)
```

```
(7.416298709205447, 1.4331409044279414e-11)
```

9 Planche 10

9.0.1 Question a

```
In [6]: def binaire(n):
            resultat = []
```

```
            quotient = n
            while quotient > 0:
                reste = quotient % 2
                quotient = quotient // 2
                resultat.append(reste)
            return resultat
```

```
In [7]: # Tests
        print(binaire(8))
        print(binaire(237))
```

```
[0, 0, 0, 1]
[1, 0, 1, 1, 0, 1, 1, 1]
```

9.0.2 Question b

Le développement binaire de n est de la forme $\sum_{i=0}^{d} \varepsilon_i 2^i$, où chaque ε_i vaut 0 ou 1.

On obtient alors

$$2n = \sum_{i=0}^{d} \varepsilon_i 2^{i+1} = \sum_{j=1}^{d+1} \varepsilon_{j-1} 2^j \quad \text{et} \quad 2n+1 = 1 + \sum_{j=1}^{d+1} \varepsilon_{j-1} 2^j.$$

Donc $\sigma_2(2n) = \sigma_2(n)$ et $\sigma_2(2n+1) = \sigma_2(n) + 1$ puis $s_{2n} = s_n$ et $s_{2n+1} = -s_n$.

9.0.3 Question c

```
In [22]: def liste_s(n):
             resultat = [0]*(n+1)
             resultat[0] = 1
             m = n // 2
             r = n % 2
             for k in range(0, m):
                 resultat[2*k+1] = -resultat[k]
                 resultat[2*k+2] = resultat[k+1]
             if r == 1:
                 resultat[n] = -resultat[m]
             return resultat
```

```
In [23]: # Test
         print(liste_s(10))
         print(liste_s(11))
```

```
[1, -1, -1, 1, -1, 1, 1, -1, -1, 1, 1]
[1, -1, -1, 1, -1, 1, 1, -1, -1, 1, 1, -1]
```

9.0.4 Question d

```
In [24]: n = 100
         liste_sommes = [1]*(n+1)
         s = liste_s(n)
         for k in range(1, n+1):
             liste_sommes[k] = liste_sommes[k-1] + s[k]
         print(liste_sommes)
```

```
[1, 0, -1, 0, -1, 0, 1, 0, -1, 0, 1, 0, 1, 0, -1, 0, -1, 0, 1, 0, 1, 0, -1, 0, 1, 0, -1, 0, -1,
```

Il semble que S_{2k+1} soit nul et que S_{2k} soit égal à 1 ou -1. En regardant de plus près, la suite $(S_{2k})_{k\in\mathbb{N}}$ ressemble fortement à la suite $(s_k)_{k\in\mathbb{N}}$.

Remarquons déjà que l'identité $s_{2n} + s_{2n+1} = 0$ donne $S_{2n+1} = S_{2n-1}$. La somme S_1 est nulle donc S_{2n+1} est nul pour tout entier n.

Si $n \geq 1$, on a ensuite $S_{2n} = S_{2n-1} + s_{2n} = s_n$. La conjecture est prouvée.

9.0.5 Question e

J'espère que la question n'a pas été posée telle quelle, car elle est infaisable sans indication. Ça passe par une réécriture des sommes partielles.

$$\sum_{n=0}^{N} \varepsilon_n v_n = \varepsilon_0 v_0 + \sum_{n=1}^{N} \varepsilon_n (V_n - V_{n-1}) = \varepsilon_0 V_0 + \sum_{n=1}^{N} \varepsilon_n V_n - \sum_{n=0}^{N-1} \varepsilon_{n+1} V_n = \sum_{n=0}^{N} (\varepsilon_n - \varepsilon_{n+1}) V_n + \varepsilon_{N+1} V_N.$$

On introduit une constante C telle que

$$\forall n \in \mathbb{N}, \quad |V_n| \leq C.$$

La majoration $|\varepsilon_{N+1} V_N| \leq C\, \varepsilon_{N+1}$ montre que $\varepsilon_{N+1} V_N$ tend vers 0 quand N tend vers $+\infty$.

La décroissance de la suite $(\varepsilon_n)_{n\in\mathbb{N}}$ donne la majoration $|(\varepsilon_n - \varepsilon_{n+1}) V_n| \leq C\, (\varepsilon_n - \varepsilon_{n+1})$.

Le fait que la suite $(\varepsilon_n)_{n\in\mathbb{N}}$ converge donne la convergence de la série de terme général $\varepsilon_n - \varepsilon_{n+1}$. Par domination, la série de terme général $(\varepsilon_n - \varepsilon_{n+1}) V_n$ converge absolument. Elle converge donc.

Ainsi, la somme partielle $\sum_{n=0}^{N} \varepsilon_n v_n$ tend vers $\sum_{n=0}^{+\infty} (\varepsilon_n - \varepsilon_{n+1}) V_n$ quand N tend vers $+\infty$. On a prouvé que la série de terme général $\varepsilon_n v_n$ est convergente.

9.0.6 Question f

Posons $P_N = \prod_{n=1}^{N} \left(\frac{n}{n+1}\right)^{s_n}$.

On trouve $\ln(P_N) = -\sum_{n=1}^{N} s_n \ln\left(1 + \frac{1}{n}\right)$.

La suite $(s_n)_{n\in\mathbb{N}}$ vérifie les hypothèses de la suite $(v_n)_{n\in\mathbb{N}}$ de la question précédente. La suite de terme général $\ln(1 + 1/n)$ vérifie les hypothèses de la suite $(\varepsilon_n)_{n\in\mathbb{N}}$.

On en déduit que la série de terme général $s_n \ln\left(1 + \frac{1}{n}\right)$ est convergente. Ainsi, la suite de terme général $\ln(P_N)$ est convergente.

Par continuité de l'exponentielle, la suite de terme général P_N est convergente et sa limite est strictement positive.

10 Planche 14

```
In [25]: import numpy as np
```

10.0.1 Question a

Une possibilité est de remarquer que la matrice A est à diagonale strictement dominante (voir exercice 3). C'est particulièrement difficile pour une première question, mais aucune autre méthode ne me saute aux yeux.

10.0.2 Question b

$T(X^*) = X^*$.

Le problème $AX = B$ est donc ramené à une recherche de point fixe.

Comme dans la planche numéro 3, on brode ici sur un théorème classique qui fut jadis au programme de nos classes.

Théorème du point fixe (Banach). Soit $(E, ||||)$ un espace vectoriel normé de dimension finie (ce théorème a un cadre plus général, la complétude, qui n'est pas au programme de nos classes).

Soit $T : E \to E$ une application. On suppose qu'il existe ρ dans $[0,1[$ tel que T soit ρ-lipschitzienne.

Alors l'application T possède exactement un point fixe X^*. De plus, toute suite $(X_n)_{n\in\mathbb{N}}$ vérifiant la relation de récurrence $X_{n+1} = T(X_n)$ converge vers X^*.

10.0.3 Question c

```
In [26]: def matBande(p, b, a):
             mat = a*np.eye(p)
             for i in range(p-1):
                 mat[i, i+1] = b
                 mat[i+1, i] = b
             return mat

         def T(p, X, B):  # Il manque la référence à la colonne B dans l'énoncé.
             C3 = matBande(p, 1, 0)
             return (C3.dot(X) + B) / 3
```

```
In [27]: # Test
         print(matBande(5, 1, 0))
```

```
[[ 0.  1.  0.  0.  0.]
 [ 1.  0.  1.  0.  0.]
 [ 0.  1.  0.  1.  0.]
 [ 0.  0.  1.  0.  1.]
 [ 0.  0.  0.  1.  0.]]
```

10.0.4 Question d

```
In [30]: p = 5
         B = np.ones((5, 1))
         A = matBande(p, -1, 3)
         print(A)
```

```
[[ 3. -1.  0.  0.  0.]
 [-1.  3. -1.  0.  0.]
 [ 0. -1.  3. -1.  0.]
 [ 0.  0. -1.  3. -1.]
 [ 0.  0.  0. -1.  3.]]
```

```
In [33]: Ainv = np.linalg.inv(A)
         print(Ainv)
```

```
[[ 0.38194444  0.14583333  0.05555556  0.02083333  0.00694444]
 [ 0.14583333  0.4375      0.16666667  0.0625      0.02083333]
 [ 0.05555556  0.16666667  0.44444444  0.16666667  0.05555556]
 [ 0.02083333  0.0625      0.16666667  0.4375      0.14583333]
 [ 0.00694444  0.02083333  0.05555556  0.14583333  0.38194444]]
```

```
In [34]: Xetoile = Ainv.dot(B)
         print(Xetoile)
```

```
[[ 0.61111111]
 [ 0.83333333]
 [ 0.88888889]
 [ 0.83333333]
 [ 0.61111111]]
```

10.0.5 Question g

Testons tout de suite le procédé d'itération. On fera les maths après.

Notons tout de même que la fonction T recalcule la matrice C à chaque appel, ce qui est très mauvais en termes de complexité.

```
In [35]: nb_etapes = 100
         xn = B  # Initialisation
         for _ in range(nb_etapes):
             xn = T(p, xn, B)
         print(xn)
         print("Évaluation de l'erreur : {}".format(np.sqrt(np.sum((xn-Xetoile)**2))))
```

```
[[ 0.61111111]
 [ 0.83333333]
 [ 0.88888889]
```

```
 [ 0.83333333]
 [ 0.61111111]]
Évaluation de l'erreur : 1.9229626863835638e-16
```

Sur cet exemple, on observe une bonne convergence du procédé.

10.0.6 Questions e, f et g

Même principe que la fin de la planche 3. Et pour cause : c'est la même méthode.

11 Planche 8

11.0.1 Question a

```
In [36]: def nombre9(n):
             quotient = n
             reste = quotient % 10
             total = 0
             while reste == 9:
                 total += 1
                 quotient = quotient // 10
                 reste = quotient % 10
             return total
```

```
In [37]: # Test
         donnees = [9, 1, 19, 91, 919, 9999, 29999]
         attendus = [1, 0, 1, 0, 1, 4, 4]
         resultat_test = [nombre9(n) for n in donnees]
         print(resultat_test)
         print(resultat_test == attendus)
```

```
[1, 0, 1, 0, 1, 4, 4]
True
```

11.0.2 Question b

```
In [38]: n = 10
         u = 9
         resultats = [1]
         for _ in range(n):
             u = u**3 * (4 + 3*u)
             resultats.append(nombre9(u))
         print(resultats)
```

```
[1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024]
```

11.0.3 Question c

On reconnaît de bien belles puissances de 2. On conjecture naturellement l'égalité $c_n = 2^n$.

Plus précisément, l'énoncé H_n de récurrence va se formuler ainsi : ń Il existe un nombre a_n non multiple de 5 tel que $u_n = a_n \times 10^{2^n} - 1$. ż

L'égalité $c_0 = 9$ prouve que H_0 est vrai (on prend $a_0 = 1$).

Soit maintenant un entier n pour lequel H_n est vrai. Quelques lignes de calcul donnent

$$u_{n+1} = 3(a_n)^4 10^{4.2^n} - 8(a_n)^3 10^{3.2^n} + 6(a_n)^2 10^{2.2^n} - 1 = a_{n+1} \times 10^{2^{n+1}} - 1$$

en ayant posé $a_{n+1} = 3(a_n)^4 10^{2.2^n} - 8(a_n)^3 10^{2^n} + 6(a_n)^2$.

Le nombre a_{n+1} est congru à $6(a_n)^2$ modulo 5 donc ce n'est pas un multiple de 5. On a prouvé H_{n+1}.

Par récurrence, on a tout ce qu'il nous faut.

11.0.4 Question d

```
In [39]: def avant9(n):
             quotient = n
             reste = n % 10
             while reste == 9:
                 quotient = quotient // 10
                 reste = quotient % 10
             return reste
```

```
In [41]: # Test
         donnees = [9, 1, 19, 91, 919, 9999, 29999]
         attendus = [0, 1, 1, 1, 1, 0, 2]
         resultat_test = [avant9(n) for n in donnees]
         print(resultat_test)
         print(resultat_test == attendus)
```

```
[0, 1, 1, 1, 1, 0, 2]
True
```

11.0.5 Question e

```
In [42]: n = 10
         u = 9
         resultats = [0]
         for _ in range(n):
             u = u**3 * (4 + 3*u)
             resultats.append(avant9(u))
         print(resultats)
```

```
[0, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5]
```

Si on appelle b_n le chiffre en question augmenté de 1, il est égal au reste a_n modulo 10. On a vu dans la démonstration de la question c que a_{n+1} est congru à $6(a_n)^2$ modulo 10. On sait que b_0 vaut 1. On en déduit que b_1 vaut 6 et comme 6×6^2 est congru à 6 modulo 10, les b_n ultérieurs valent tous 6, ce qui explique le comportement observé ci-dessus.

In [ ]: