TP 2

Correction du TP

Exercice 1 (★)
Sans rentrer les cœfficients un à un, déclarer les matrices $A = \begin{pmatrix} 5 & 3 & 3 \\ 3 & 5 & 3 \\ 3 & 3 & 5 \end{pmatrix}$ et $B = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{pmatrix}$.

> On peut utiliser les commandes suivantes :
>
> ```
> >>> A = 3*np.ones((3,3))+2*np.eye(3,3) ; A
> >>> B = np.eye(4,4) ; B[:,0:1] = np.ones((4,1)) ; B
> ```
>
> À noter que la commande :
>
> ```
> >>> B = np.eye(4,4) ; B[:,0:1] = 1 ; B
> ```
>
> fonctionne également.

Exercice 2 (★)
Que vaut le produit d'un vecteur colonne $\begin{pmatrix} a_1 \\ \vdots \\ a_n \end{pmatrix}$ par le vecteur ligne $\begin{pmatrix} 1 & \dots & 1 \end{pmatrix}$?

En déduire une ligne de commande créant la matrice $\begin{pmatrix} 1 & 1 & \dots & 1 \\ 2 & 2 & \dots & 2 \\ \vdots & \vdots & & \vdots \\ 10 & 10 & \dots & 10 \end{pmatrix} \in \mathscr{M}_{10}(\mathbb{R})$.

> On a :
>
> $$\begin{pmatrix} a_1 \\ \vdots \\ a_n \end{pmatrix} \times \begin{pmatrix} 1 & \dots & 1 \end{pmatrix} = \begin{pmatrix} a_1 & a_1 & \dots & a_1 \\ a_2 & a_2 & \dots & a_2 \\ \vdots & \vdots & & \vdots \\ a_{10} & a_{10} & \dots & a_{10} \end{pmatrix}.$$
>
> On en déduit les commandes suivantes :
>
> ```
> >>> C = np.transpose([np.arange(1,11)]) ; L = np.ones(1,10) ; np.dot(C,L)
> ```
>
> À noter les crochets autour de `np.arange(1,11)` pour qu'il le considère bien comme une matrice ligne, et non comme un vecteur (afin de pouvoir le transposer).

Exercice 3 (★)

1. On définit la matrice $A = \begin{pmatrix} 1 & 1 & 2 & 0 \\ 1 & -1 & 0 & -2 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & -1 \end{pmatrix}$.
   (a) Déclarer la matrice A.
   (b) Appliquer l'algorithme du pivot de Gauss à l'aide de `Python` (on effectuera chacune des opérations élémentaires sur `Python`). En déduire le rang.
   (c) Vérifier votre résultat à l'aide de la fonction `al.matrix_rank`.
2. On considère à présent la matrice $B = \begin{pmatrix} 1 & 1 & 0 \\ 1 & -1 & 1 \\ 1 & 0 & 1 \end{pmatrix}$.

(a) Déclarer la matrice B. Vérifier que B est inversible et calculer son inverse à l'aide de `Python`.

(b) Résoudre à l'aide de `Python` le système $\begin{cases} x+y & =1 \\ x-y+z & =3 \\ x+z & =-2 \end{cases}$.

1. On peut procéder ainsi pour cette question :

```
>>> A = np.array([[1,1,2,0],[1,-1,0,-2],[0,1,1,1],[1,0,1,-1]])
>>> B = A.copy() # On affecte dans B une copie indépendante de A
>>> B[1,:] = B[1,:]-B[0,:] ; B[3,:] = B[3,:]-B[0,:]; B
>>> B[1,:] = (-1/2)*B[1,:]; B
>>> B[2,:] = B[2,:]-B[1,:]; B[3,:] = B[3,:]+B[1,:]; B
>>> al.matrix_rank(A)
```

2. On procède ainsi :

```
>>> B = np.array([[1,1,0],[1,-1,1],[1,0,1]])
>>> al.matrix_rank(B)
3
>>> al.inv(B)
array([[ 1.,  1., -1.],
       [-0., -1.,  1.],
       [-1., -1.,  2.]])
>>> al.solve(B,[[1],[3],[2]])
array([[ 2.],
       [-1.],
       [ 0.]])
```

Exercice 4 (★★)

1. (a) En une seule ligne de commande, créer le vecteur $x = \left(1, \frac{1}{4}, \frac{1}{9}, \frac{1}{16}, \ldots, \frac{1}{100}\right)$ sans saisir un à un les éléments.

   (b) Compléter la commande précédente pour qu'elle renvoie $\sum_{k=1}^{10} \frac{1}{k^2}$.

2. En une seule ligne de commande, calculer la somme $\sum_{n=0}^{10} \frac{1}{2^n}$.

3. Que calculent les commandes suivantes :

   (a) `x = np.ones(10) ; y = np.cumsum(x)`

   (b) `x = np.ones(10) ; y = np.sum(np.cumsum(x))`

   (c) `x = np.ones(10) ; y = np.sum(np.cumsum(np.cumsum(x)))`

1. (a) On pourrait proposer l'instruction suivante, en faisant des opérations coefficients par coefficients.

```
>>> x = np.arange(1,11)**(-2)
```

   Cependant, `Python` renvoie un message d'erreur, car on part d'un vecteur `np.arange(1,11)` d'entiers pour obtenir un vecteur contenant a priori des flottants, ce que `Python` ne permet pas. On va donc modifier le vecteur de départ pour qu'il contienne dès le début des flottants.

```
>>> x = np.arange(1.,11.)**(-2)
```

   (b) On applique la fonction `np.sum` :

```
>>> np.sum(x)
```

2. Sur le modèle de la question précédente :

```
>>> x = np.sum((1/2)**np.arange(1,11))
```

3. (a) `x` est le vecteur de taille 10 contenant que des 1. `y` est un vecteur de taille 10 dont la i-ème composante contient :
$$y_i = \sum_{j=1}^{i} x_i = \sum_{j=1}^{i} 1 = i.$$
(b) `y` contient le réel :
$$\sum_{i=1}^{10} i = \frac{10 \times 11}{2} = 55.$$
(c) `y` contient cette fois le réel :
$$\sum_{i=1}^{10}\sum_{j=1}^{i} j = \sum_{i=1}^{10} \frac{i(i+1)}{2} = \frac{1}{2}\sum_{i=1}^{10}(i+i^2) = \frac{10\times 11}{4} + \frac{10\times 11\times(2\times 10+1)}{12}.$$

Exercice 5 (★★)

Le nombre de véhicules passant sur une petite route de campagne en une journée est une variable aléatoire X qui suit une loi de Poisson $\mathscr{P}(5)$.

On va simuler une année de circulation sur cette route. Pour cela, importons la bibliothèque `numpy.random` à l'aide de l'instruction `import numpy.random as rd`, et utilisons la commande `A = rd.poisson(5,[52,7])`. La i-ème ligne de A correspondant à la i-ème semaine de circulation. On obtient ainsi une matrice A qui contient 52×7 nombres tirés suivant la loi $\mathscr{P}(5)$.

1. Déterminer le nombre maximal de véhicules circulant sur la route en une journée durant l'année. Et le nombre maximal de véhicules un mercredi durant l'année ?
2. Déterminer les numéros des semaines où la route à connu son maximum de fréquentation.
3. Déterminer le nombre de jours où la route a connu son minimum de fréquentation.
4. En une seule ligne de commande, évaluer la probabilité qu'une journée voit passer plus de 8 véhicules. Sauriez-vous calculer la valeur exacte de cette probabilité ?

1. On obtient le nombre maximal de véhicules circulant en une journée grâce à la commande `np.max(A)`.
Pour obtenir le mercredi où il y a eu le plus de circulation, on utilise la commande `np.max(A(:,2))`.
2. Il faut pour cela sommer suivant les lignes de A pour obtenir le nombre de véhicule pour chaque semaine. On

```
S = np.sum(A,1)
```

On parcourt ensuite `S` afin de renvoyer les indices des lignes où le coefficient de S est maximal :

```
m = np.max(S)
for k in range(52) :
    if S[k] = m :
        print(k+1)
```

3. On parcourt toute la matrice `A` et on compte le nombre de jours où la fréquentation est minimale :

```
m = np.min(A)
k = 0
for i in range(52) :
    for j in range(7) :
        if A[i,j] == m :
```

```
                k = k+1
print(k)
```

4. Nous n'aurons qu'une approximation de cette probabilité, mais on peut compter le nombre de jours où 8 voitures sont passées et diviser ce nombre par 364.

```
k = 0
for i in range(52) :
    for j in range(7) :
        if A[i,j] >= 8 :
            k = k+1
print(k/364)
```

Exercice 6 (★★)

Écrire une fonction d'en-tête `def trace(A)` prenant en entrée une matrice `A`, et renvoyant la trace de `A` si la matrice entrée est carrée, et un message d'erreur sinon.

On peut utiliser le programme suivant :

```
def trace(A)
    n,p = np.shape(A)
    if n != p :
        return "A n'est pas carrée"
    else :
        T = 0
        for k in range(n):
            T = T + A[k,k]
        return T
```

Exercice 7 (★★★)

1. (a) Établir que, lorsque k est inférieur ou égal à n, on a :

$$\binom{n}{k} = \prod_{i=1}^{k} \left(\frac{n+1}{i} - 1 \right).$$

   (b) Écrire une fonction d'en-tête `def coeffbin(k,n):` permettant de calculer $\binom{n}{k}$ à partir des vecteurs `np.arange(1,k+1)` et `np.ones(k)`.

   (c) Retrouver ce résultat avec la commande suivante (qu'on justifiera) :

`np.cumprod(np.arange(n,0,-1)/np.arange(1,n+1))`

2. On désire créer une matrice de taille $n \times n$ ($n \geq 1$) dont l'élément de la ligne $i+1$ et de la colonne $j+1$ est égal à $\binom{i}{j}$, avec i et j dans $[\![0, n-1]\!]$.

   (a) Donner une ligne de commande permettant de créer une matrice C de taille $n \times n$ dont les éléments de la première colonne sont égaux à 1 (les autres éléments étant nuls).

   (b) Écrire une fonction d'en-tête `def triangle_Pascal(n)` qui remplit la matrice C à l'aide de la formule de Pascal et renvoie ainsi la matrice voulue (triangle de Pascal).

1. (a) On a :

$$\binom{n}{k} = \frac{n!}{k!(n-k)!} = \frac{n(n-1)\dots(n-k+1)}{1\times 2\times\dots\times(k-1)k}$$
$$= \frac{n+1-1}{1}\frac{n+1-2}{2}\dots\frac{n+1-(k-1)}{k-1}\frac{n+1-k}{k} = \prod_{i=1}^{k}\left(\frac{n+1}{i}-1\right).$$

(b) On propose la fonction suivante :

```
def coeffbin(k,n):
    u = np.arange(1,k+1)
    v = (n+1)*np.ones(k)
    return int(np.prod(v/u-1))
```

On utilise la fonction `int()` afin que le résultat retourné soit de type entier et non flottant.

(c) `np.arange(n,0,-1)` est le vecteur `[n,n-1,...,2,1]`. Et donc :

`np.arange(n,0,-1)/np.arange(1,n+1)`

est le vecteur `[n/1,(n-1)/2,...,2/(n-1),1/n]`. Par conséquent, si on applique à ce vecteur la fonction `np.cumprod`, le vecteur obtenu a pour k-ème composante :

$$\frac{n}{1}\times\dots\times\frac{n-k+1}{k} = \frac{n(n-1)\dots(n-k+1)}{k!} = \binom{n}{k}.$$

On peut en déduire la fonction suivante :

```
def coeffbin(k,n):
    if k==0 :
        return 1
    else :
        A = np.cumprod(np.arange(n,0,-1)/np.arange(1,n+1))
        return int(A[k-1])
```

2. (a) On peut procéder ainsi :

```
C = np.zeros((n,n))
C[:,0:1] = ones((n,1))
```

(b) On rappelle que $\binom{n}{k} = \binom{n-1}{k-1} + \binom{n-1}{k}$. Reste à remplir la matrice ligne par ligne à l'aide de cette relation :

```
def triangle_Pascal(n):
    C = np.zeros((n,n))
    C[:,0:1] = np.ones((n,1))
    for k in range(1,n):
        for j in range(1,k+1):
            C[k,j] = C[k-1,j-1]+C[k-1,j]
    return C
```

Voici la réponse de `Python` lorsqu'on entre `triangle_Pascal(6)` :

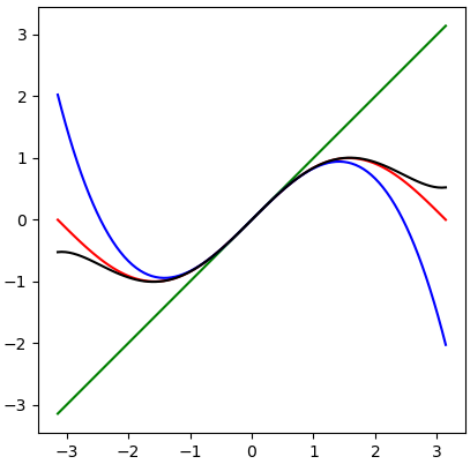
$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0\\ 1 & 1 & 0 & 0 & 0 & 0\\ 1 & 2 & 1 & 0 & 0 & 0\\ 1 & 3 & 3 & 1 & 0 & 0\\ 1 & 4 & 6 & 4 & 1 & 0\\ 1 & 5 & 10 & 10 & 5 & 1 \end{pmatrix}.$$

Exercice 8 (★)
Représenter sur une même figure les représentations graphiques des fonctions sin, $x \mapsto x$, $x \mapsto x - \dfrac{x^3}{3!}$, $x \mapsto x - \dfrac{x^3}{3!} + \dfrac{x^5}{5!}$ sur l'intervalle $[-\pi, \pi]$. Que remarque-t-on ?

On utilise le code suivant :

```
import matplotlib.pyplot as plt

def P1(x):
    return x

def P2(x):
    return x-(x**3)/6

def P3(x):
    return x-(x**3)/6+(x**5)/120

x=np.linspace(-np.pi,np.pi,100)
plt.plot(x,np.sin(x),'r')
plt.plot(x,P1(x),'g')
plt.plot(x,P2(x),'b')
plt.plot(x,P3(x),'k')

plt.axis("scaled")
plt.show()
```

On obtient le graphe suivant :

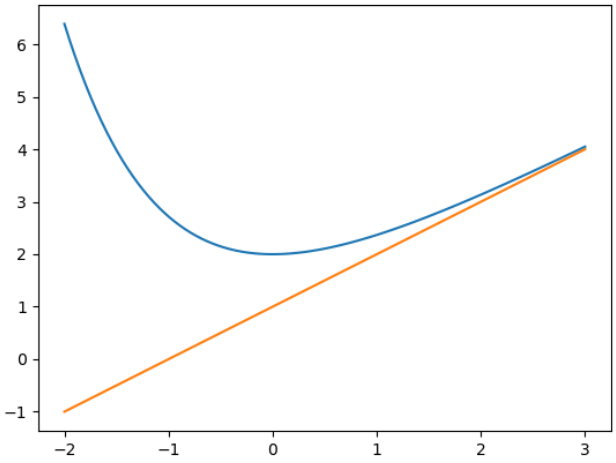


Exercice 9 (★)
Soit la fonction $f : x \mapsto x+1+e^{-x}$. On note $\mathscr{D}$ la droite d'équation $y = x+1$. Comme $\lim\limits_{x\to+\infty} f(x)-(x+1) = 0$, on dit que $\mathscr{D}$ est asymptote à la courbe $\mathscr{C}_f$ de f au voisinage de $+\infty$.
Illustrer graphiquement cette situation en écrivant des instructions qui permettent de tracer sur un même graphique $\mathscr{C}_f$ et la droite $\mathscr{D}$. On fera varier x dans $[-2, 3]$.

Voici une proposition :

```
x = np.linspace(-2,3,100)
plt.plot(x,x+1+np.exp(-x))
plt.plot(x,x+1)
plt.show()
```

Ce qui renvoie :



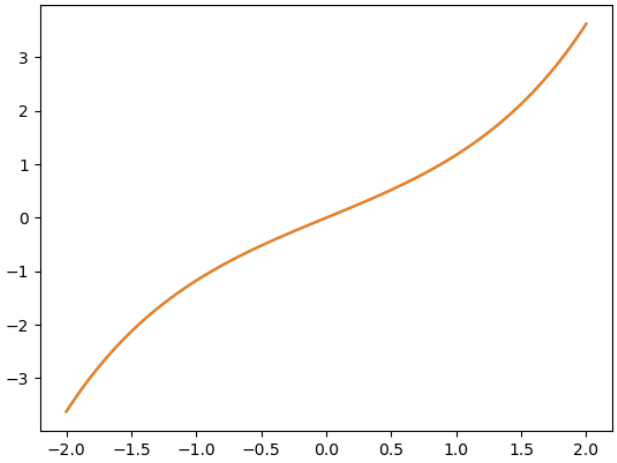
Exercice 10 (★★ - Fonction réciproque)

1. On considère la fonction $f : \mathbb{R} \to \mathbb{R}$ définie par $f(x) = \dfrac{e^x - e^{-x}}{2}$ (appelée *sinus hyperbolique*, et souvent notée *sh*). Montrer que f réalise une bijection de $\mathbb{R}$ dans $\mathbb{R}$.
2. Définir une subdivision de l'intervalle $I = [-2, 2]$ en 100 sous-intervalles de même longueur, puis écrire des instructions permettant de tracer la courbe de f définie sur I.
3. Ajouter des instructions pour tracer sur le même graphique la courbe de la bijection réciproque de f.

1. f est continue sur $\mathbb{R}$ comme composée de fonctions continues. Elle est de plus dérivable pour les mêmes raisons, et on a :
$$\forall x \in \mathbb{R}, \quad f'(x) = \frac{e^x + e^{-x}}{2} > 0.$$
Donc f est strictement monotone sur $\mathbb{R}$, et on a $\lim\limits_{x \to \pm\infty} f(x) = \pm\infty$. Par le théorème de la bijection, f réalise une bijection de $\mathbb{R}$ sur $\mathbb{R}$.
2. On peut utiliser les instructions :

```
x = np.linspace(-2,2,100)

def f(x):
    return (np.exp(x)-np.exp(-x))/2

y = f(x)
plt.plot(x,y)
```

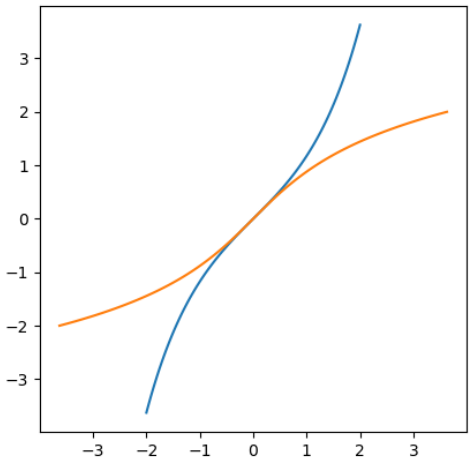
Ce qui renvoie :



3. On sait que le graphe de f^{-1} est le symétrique de celui de f par rapport à la droite $y = x$. On utilise ce résultat ici en ajoutant à ce qui précède l'instruction :

```
plt.plot(y,x)
```

On obtient :



Exercice 11 (★★)

1. Créer un vecteur ligne `u`, tel que pour tout $1 \leq i \leq 30$, `u[i]` soit égal à $\dfrac{(-1)^i}{i^2}$.
2. Créer un vecteur `v` tel que pour $1 \leq i \leq 30$, `v[i]` soit égal à $\displaystyle\sum_{k=1}^{i} \frac{(-1)^k}{k^2}$.
3. Représenter graphiquement les points M_i de coordonnées $\left(i, \displaystyle\sum_{k=1}^{i} \frac{(-1)^k}{k^2}\right)$ pour $i = 1, \ldots, 30$. Que peut-on conjecturer à partir de cette représentation ?

1. On peut procéder ainsi :

```
u = ((-1)**np.arange(1,31))/(np.arange(1,31)**2)
```

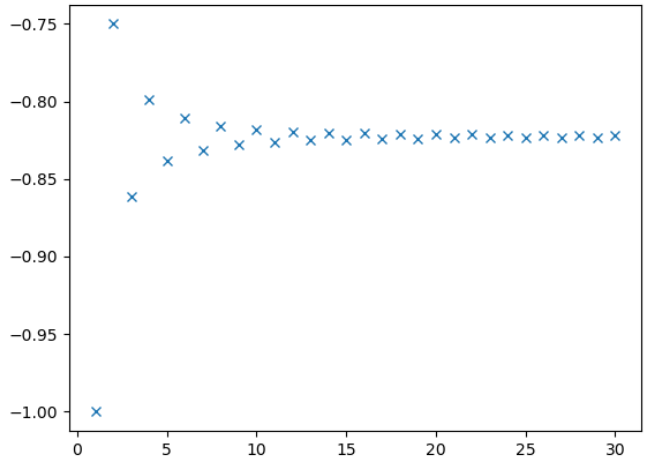
2. On utilise la commande `np.cumsum` :

```
v = np.cumsum(u)
```

3. On utilise pour cela la ligne de commandes suivante :

```
x = np.arange(1,31)
plt.plot(x,v,'x')
```

On obtient le graphe suivant :



On observe que la série converge et que sa somme est environ égale à 0,82. Plus précisément, on observe que les suites extraites paires et impaires de la suite des sommes partielles de la série sont adjacentes et converge vers la somme de la série.

On vérifie facilement à la main que la série converge. En effet, elle est absolument convergente (car la série $\sum \frac{1}{n^2}$ converge en tant que série de Riemann avec $\alpha = 2 > 1$), donc convergente.

> **Déjà vu ?**
> Nous avons plusieurs fois rencontré des séries alternées en TD (série harmonique alternée) ou en DM. On a à chaque fois montré que les séries extraites paires et impaires de (S_n) sont adjacentes, l'une croissante, l'autre décroissante, et qu'elles tendent vers la somme de la série. C'est ce qu'on remarque aussi ici graphiquement. Pour plus de détails sur les séries alternées, je vous renvoie au :
>
> ☞ **Complément de cours 1. Autour des séries alternées.**

Exercice 12 (★★ - Étude d'une suite implicite)

On note, pour tout $n \in \mathbb{N}^*$, (E_n) l'équation $x^3 + \frac{1}{n}x^2 + x - 2 = 0$.

1. Soit $n \in \mathbb{N}^*$. Étudier les variations sur $\mathbb{R}_+$ de la fonction $f_n : x \mapsto x^3 + \frac{1}{n}x^2 + x - 2$.
   En déduire que l'équation (E_n) admet une unique solution u_n sur $\mathbb{R}_+$, et que u_n appartient à $[0, 1]$.

2. On considère le script suivant :

```
def f(n,x):
    return x**3+(1/n)*x**2+x-2

def suite(n):
```

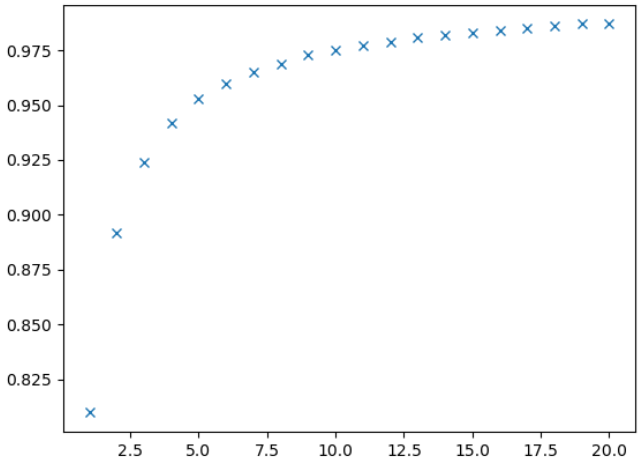
```
    v = np.linspace(0,1,1000)
    i = 0
    while f(n,v[i])<0 :
        i = i+1
    return v[i-1]
```

Que renvoie la commande `suite(n)` ?

3. Compléter le programme suivant afin qu'il permette de représenter les 20 premiers termes de la suite $(u_n)_{n\in\mathbb{N}^*}$:

```
U = np.zeros(20)
for ...
    U[k-1] = ...
x = np.arange(1,21)
plt.plot(...,...,'x')
```

4. Le programme précédent complété permet d'obtenir la représentation graphique suivante :



Quelles conjectures peut-on émettre sur la monotonie et la limite de la suite $(u_n)_{n\in\mathbb{N}^*}$?

5. Démontrer les résultats annoncés à la question précédente.

1. On pose $f_n(x) = x^3 + \frac{1}{n}x^2 + x - 2$. f_n est strictement croissante (on l'observe en la dérivant), continue et telle que $f_n(0) = -2$, $f_n(1) = \frac{1}{n}$. Par le théorème de la bijection, l'équation (E_n) admet une unique solution x_n sur $\mathbb{R}_+$. De plus $x_n \in]0,1[$.
2. La fonction `suite(n)` parcourt l'intervalle $[0,1]$ avec un pas de 10^{-3} jusqu'à ce que la condition `f(n,v[i])<0` ne soit plus satisfaite. Étant donné la croissance de f_n, u_n appartient à l'intervalle $[v(i-1), v(i)]$, et la valeur `v[i-1]` retournée par `suite(n)` constitue donc une approximation de u_n à 10^{-3} près.
3. On peut procéder ainsi :

```
U = np.zeros(20)
for k in range(1,21)
    U[k-1] = suite(k)
x = np.arange(1,21)
plt.plot(x,U,'x')
```

4. On peut conjecturer que la suite (u_n) est croissante et qu'elle converge vers 1.

5. Pour tout $n \geq 1$, on a $f_{n+1}(x_n) \leq f_n(x_n) = 0 = f_{n+1}(x_{n+1})$. Puisque f_{n+1} est strictement croissante, on en déduit que $x_n \leq x_{n+1}$ et donc (x_n) est croissante. Par le théorème de la limite monotone, elle converge donc vers une limite $\ell \in]0, 1]$. Or en passant à la limite dans (E_n), on obtient :
$$\ell^3 + \ell - 2 = 0.$$
Et 1 est racine évidente de $x^3 + x - 2 = (x - 1)(x^2 + x + 2)$, et c'est l'unique racine réelle de ce polynôme. Ainsi, on obtient bien $\ell = 1$.