TD 19 -
Systèmes linéaires

Informatique
MPSI-PCSI - Lycée Thiers

Exercice 1.

Exercice 1. Après avoir vérifié que leur solution est unique, résoudre les systèmes d'équations linaires suivants d'inconnues x, y, z de 2 façons différentes, avec les fonctions `solve` et `inv` du sous module `linalg` de `numpy`.

1.
$$\begin{cases} x + y + z = 1 \\ x - 2y + z = 0 \\ 2x - y + z = 2 \end{cases}$$

2.
$$\begin{cases} 2x + y - 2z + t = -2 \\ x + 2y + z - t = 1 \\ x - y + z - t = 4 \\ -x + y + 2z + t = -1 \end{cases}$$

Exercice 1. Corrigé

1.

```
import numpy as np
A = np.array([[1,1,1],[1,-2,1],[2,-1,1]])
B = np.array([1,0,2])
print("rang :",np.linalg.matrix_rank(A))
```

```
rang : 3
```

Le système a 3 équations et 3 inconnues. Le rang de sa matrice est maximal, le système admet une unique solution.

```
Sol = np.linalg.solve(A,B)
print(Sol)
```

Solution :

```
[ 1.66666667 0.33333333 -1. ]
```

Exercice 1. Corrigé

2.

```
A = np.array([
[2,1,-2,1],
[1,2,1,-1],
[1,-1,1,-1],
[-1,1,2,1]])
B = np.array([-2,1,4,-1])
print("rang :",np.linalg.matrix_rank(A))
```

```
rang : 4
```

Le système a 4 équations et 4 inconnues. Le rang de sa matrice est maximal, le système admet une unique solution.

```
Sol = np.linalg.solve(A,B)
print(Sol)
```

Solution :

```
[ 1. -1. 1. -1.]
```

Exercice 2 : Enoncé

Exercice 2. Nous considérons dans les exercices suivants les trois suites imbriquées $(u_n), (v_n), (w_n)$ définies à l'aide de la relation de récurrence :

$$u_0 = v_0 = w_0 = 1, \qquad \forall\, n \in \mathbb{N} \quad \begin{cases} u_{n+1} = u_n + v_n + w_n \\ v_{n+1} = 2.u_n - v_n + w_n \\ w_{n+1} = u_n + 2.v_n + w_n \end{cases}$$

1. Ecrire une fonction `uvw(n)` prenant en paramètre un entier positif `n` et qui retourne la valeur du triplet (u_n, v_n, w_n). On utilisera un produit matriciel après avoir re-écrit la relation de récurrence à l'aide de l'écriture matricielle :

$$\begin{pmatrix} u_0 \\ v_0 \\ w_0 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \qquad \forall\, n \in \mathbb{N}, \quad \begin{pmatrix} u_{n+1} \\ v_{n+1} \\ w_{n+1} \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 \\ 2 & -1 & 1 \\ 1 & 2 & 1 \end{pmatrix} \times \begin{pmatrix} u_n \\ v_n \\ w_n \end{pmatrix}$$

.

Pour vérification, `uvw(10)` retournera `array([70632, 56813, 89072])`.

2. Pour trois suites $(u'_n), (v'_n), (w'_n)$ vérifiant la même relation de récurrence que ci-dessus, on a obtenu $u'_{10} = 33752$, $v'_{10} = 26101$, $w'_{10} = 43038$. Déterminer les valeurs de u_0, v_0, w_0.

Exercice 2. Corrigé

1.

```
import numpy as np
def uvw(n):
    U = np.array([1,1,1])
    A = np.array([[1,1,1],[2,-1,1],[1,2,1]])
    for k in range(n):
        U = np.dot(A,U)
    return U
```

```
In [1]: uvw(10)
Out[1]: array([70632, 56813, 89072])
```

Exercice 2. Corrigé

2.

```
A = np.array([[1,1,1],[2,-1,1],[1,2,1]])
A10 = np.eye(3) # Matrice Identité
U = np.array([33752,26101,43038])
for k in range(10):
    A10 = np.dot(A10,A)
print(np.linalg.solve(A10,U))
```

```
[ 0.99991635 -1.00003788 1.00013596]
```

En fait $(u_0, v_0, w_0) = (1, -1, 1)$ comme le montre :

```
In [6]: np.dot(A10,[1,-1,1])
Out[6]: array([ 33752., 26101., 43038.])
```

Exercice 3.

Déterminer tous les polynômes de degré au plus 4 prenant pour valeurs -1, -5, 1, 5 et 19 respectivement en -2, -1, 0, 1 et 2.

Correction : Il y a au plus un polynôme de degré 4 prenant ces valeurs :

En effet supposons que P et Q soient deux polynomes de degré au plus 4 vérifiant cette condition, alors $P - Q$ est un polynôme de degré au plus 4 ayant 5 racines : c'est donc le polynôme identiquement nul, donc $P = Q$.

Soit $P = a_0 + a_1X + a_2X^2 + a_3X^3 + a_4X^4$:

$$\begin{cases} P(-2) = a_0 - 2a_1 + 4a_2 - 8a_3 + 16a_4 & = -1 \\ P(-1) = a_0 - a_1 + a_2 - a_3 + a_4 & = -5 \\ P(0) = a_0 & = 1 \\ P(1) = a_0 + a_1 + a_2 + a_3 + a_4 & = 5 \\ P(2) = a_0 + 2a_1 + 4a_2 + 8a_3 + 16a_4 & = 19 \end{cases}$$

On est amené à résoudre ce système d'inconnues a_0, a_1, a_2, a_3, a_4.

Exercice 3. Corrigé

```
A = [
[1,-2,4,-8,16],
[1,-1,1,-1,1],
[1,0,0,0,0],
[1,1,1,1,1],
[1,2,4,8,16]]
print(np.linalg.matrix_rank(A)
```

```
5
```

Il y a une unique solution.

```
B = [-1,-5,1,5,19]
Coef = np.linalg.solve(A,B)
```

```
In [1]: Coef
Out[1]: array([ 1., 5., -2., -0., 1.])
```

C'est le polynôme $P(X) = 1 + 5X - 2X^2 + X^4$.

Exercice 3. Corrigé

Ecrivons une fonction P prenant en paramète un nombre x et qui renvoie la valeur polynomiale de P en x :

```
def P(x):
    n = len(Coef)
    S = 0
    for k in range(n):
        S += Coef[k] * x**k
    return S
```

```
In [2]: P(-2),P(-1),P(0),P(1),P(2)
Out[2]: (-1.0, -5.0, 1.0, 5.0, 19.0)
```