ECG 2

TP informatique

Thème : Algèbre bilinéaire

Exercice 1. ✦✦ **Méthode de la puissance**

Dans la suite, A désigne une matrice de $\mathcal{M}_n(\mathbb{R})$ non nulle et diagonalisable. On pose λ, la plus grande valeur propre de A en valeur absolue.

Le but de cet exercice est de déterminer une approximation d'un vecteur propre de A associé à la valeur propre λ. Pour cela, on définit la suite de vecteurs $(\mathrm{X}_k)_{k\in\mathbb{N}}$ par

$$\mathrm{X}_0 \in \mathcal{M}_{n,1}(\mathbb{R}) \quad \text{et} \quad \forall\, k \in \mathbb{N}, \quad \mathrm{X}_{k+1} = \frac{\mathrm{AX}_k}{\|\mathrm{AX}_k\|}$$

où $\|\cdot\|$ désigne la norme associée au produit scalaire canonique sur $\mathcal{M}_{n,1}(\mathbb{R})$. On admet que la suite $(\mathrm{X}_k)_{k\in\mathbb{N}}$ converge vers un vecteur propre associé à λ, c'est-à-dire que, pour tout indice $i \in [\![1; n]\!]$, la suite de la i-ème coordonnée de X_k converge.

1. Écrire une fonction **norme** qui prend en argument une matrice colonne X et renvoie sa norme $\|\mathrm{X}\|$.
2. Écrire une fonction **puissance(A, X0, k)** qui renvoie le terme X_k.
3. *Test.* On pose
$$\mathrm{A} = \begin{bmatrix} 0.5 & 0.5 \\ 0.2 & 0.8 \end{bmatrix} \quad \text{et} \quad \mathrm{X}_0 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}.$$
Calculer (sans Python) les valeurs propres de A et donner un vecteur propre de norme 1 associé à la plus grande valeur propre. Comparer les résultats obtenus avec les résultats obtenus avec la fonction **puissance**.
4. Modifier le programme précédent pour construire une nouvelle fonction qui prend en arguments A, X_0, λ et affiche la plus petite valeur k telle que $\|\mathrm{AX}_k - \lambda \mathrm{X}_k\| \leqslant 10^{-6}$.
5. On suppose maintenant la matrice A inversible. Comment obtenir une approximation d'un vecteur propre d'une matrice associé à μ, la plus petite valeur propre de A en valeur absolue ?

Exercice 2. ✦✦ **Construction d'une famille orthonormée avec deux vecteurs**

1. Soient u, $v \in \mathbb{R}^n$. Comment construire deux vecteurs e_1, $e_2 \in \mathbb{R}^n$ tels que
$$\|e_1\| = 1, \quad \|e_2\| = 1 \quad \text{et} \quad \mathrm{Vect}(u, v) = \mathrm{Vect}(e_1, e_2) \ ?$$
2. En déduire un programme qui prend en arguments deux vecteurs de $\mathbb{R}^3$ puis teste si les deux vecteurs sont non colinéaires et, dans le cas où ils ne sont pas colinéaires, renvoie une base orthonormée du sous-espace vectoriel engendré par ces deux vecteurs.
On pourra identifier un vecteur de $\mathbb{R}^3$ avec une matrice de $\mathcal{M}_{3,1}(\mathbb{R})$.

ECG 2

Éléments de solutions

Exercice 1

Avant de commencer, on importe :

```
import numpy as np
```

1. Si les coefficients de U sont donnés par u_i, on sait que

$$\|\mathrm{U}\| = \sqrt{\sum_{i=1}^{n} {u_i}^2}.$$

On en déduit le programme :

```
def norme(U):
    n=len(U)
    s=0
    for i in range(n):
        s+=U[i]**2
    return s**(1/2)

# Un petit test
U=np.array([3,0,4])
print(norme(U))
5.0
```

2.

```
def puissance(X0,A,k):
    X=X0
    for i in range(k):
        X=np.dot(A,X)
        X=X/norme(X)
    return X
```

3. Dans un premier temps, on vérifie que la matrice A est diagonalisable avec deux valeurs propres 1 et 0.3. De plus,

$$\mathrm{E}_1(\mathrm{A}) = \mathrm{Vect}\left(\begin{bmatrix}1\\1\end{bmatrix}\right) \quad \text{et} \quad \mathrm{E}_{0.3}(\mathrm{A}) = \mathrm{Vect}\left(\begin{bmatrix}5\\-2\end{bmatrix}\right).$$

On teste :

```
X0=np.array([1,0])
A=np.array([[1/2,1/2],[1/5,4/5]])

print(puissance(X0,A,10))
[0.70711409 0.70709947]
```

4.

```
def puissance2(X0,A,lbda):
    X=X0
    Y=np.dot(A,X)
    c=0
    while norme(Y-lbda*X)>10**(-6):
        Y=np.dot(A,X)
        X=Y/norme(Y)
        c+=1
    return c
```

5. Si A est inversible alors 0 n'est pas valeur propre et l'équation $\mathrm{AX} = \lambda \mathrm{X}$ est équivalente à $\lambda^{-1}\mathrm{X} = \mathrm{A}^{-1}\mathrm{X}$. On en déduit que la plus petite valeur propre de A en valeur absolue est donc la plus grande valeur propre en valeur absolue de A^{-1} et surtout les espaces propres restent identiques. On peut donc appliquer la méthode de la puissance à A^{-1}. CE qui donne

```
Ainv=np.linalg.inv(A)
puissance(X0,Ainv,k)
```

Optimisons un peu le code en rajoutant le cas où la matrice n'est pas inversible. Dans ce cas, on sait que 0 est la plus petite valeur propre en valeur absolue.

```
def puissance3(X0,A,k):
    [n,p]=np.shape(A)
    if np.linalg.matrix_rank(A)!=n or n!=p
        :
        return 0
    else :
        Ainv=np.linalg.inv(A)
        return puissance(X0,Ainv,k)
```

Et le test :

```
>>> puissance3(X0,A,10))
[0.92847669 -0.37139068]

>> X=np.array([5,-2])
>>> X/norme(X)
array([ 0.92847669, -0.37139068])
```

Exercice 2

1. En reprenant le procédé de Schmidt, on peut poser

$$e_1 = \frac{u}{\|u\|}, \quad w = v - \frac{\langle u, v\rangle}{\|u\|^2}u, \quad e_2 = \frac{w}{\|w\|}.$$

2. Les vecteurs u et v sont liées si $u = 0_{\mathrm{E}}$ ou $w = 0_{\mathrm{E}}$. On en déduit le code :

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


def scal(u,v):
    n=len(u)
    s=0
    for i in range(n):
        s+=u[i]*v[i]
    return s

def orth(u,v):

    if scal(u,u)==0:
        return 'u est nul'
```

```
    e1=u/scal(u,u)**(1/2)
    w=v-scal(e1,v)*e1
    if scal(w,w)==0:
        return  'les vecteurs sont liés'
    else : return e1, w/scal(w,w)**(1/2)


# et le test

u=np.array([3,0,4])
v=np.array([0,4,3])
[e1,e2]=orth(u,v)


>>> e1
array([0.6, 0. , 0.8])

>>> e2
array([-0.32829174,  0.91192151,
    0.24621881])

>>> scal(e1,e2)
-8.326672684688674e-17

>>> scal(e1,e1)
1.0

>>> scal(e2,e2)
1.0000000000000002
```