Python 8
Chaînes de Markov

Matrices stochastiques

1. Écrire une fonction `Python` prenant en argument d'entrée une matrice et renvoyant `True` si elle est stochastique, `False` sinon.

```python
import numpy as np

def stochastique(A):
    n,p=np.shape(A)
    for i in range(0,n):
        for j in range(0,p):
            if A[i,j]<0:
                return False
        if sum(A[i,:])!=1:
            return False
    return True
```

2. En déduire une fonction `Python` permettant de savoir si une matrice est bistochastique.

```python
def bistochastique(A):
    if stochastique(A)==True and stochastique(np.transpose(A))==True:
        return True
    else:
        return False
```

Au boulot !!

On classifie les étudiants de CPGE ECG en trois groupes :

- Groupe 1 : étudiants qui ne travaillent pas ;
- Groupe 2 : étudiants qui travaillent mais avec des résultats encore faibles ;
- Groupe 3 : étudiants qui travaillent et avec des résultats très encourageants.

On choisit un étudiant au hasard en CPGE ECG et on note, pour tout $n \in \mathbb{N}^*$, X_n la variable aléatoire égale au groupe dans lequel se situe l'étudiant après n semaines passées en CPGE. On considère $X_0 = 1$.
On considère que, d'une semaine à l'autre :

- 20% des étudiants du groupe 1 passent dans le groupe 2, les autres restent dans le groupe 1 ;
- 20% des étudiants du groupe 2 passent dans le groupe 3, 10% retournent dans le groupe 1 et les autres restent dans le groupe 2 ;
- 10% des étudiants du groupe 3 retournent dans le groupe 2 et les autres restent dans le groupe 3.

On note, pour tout $n \in \mathbb{N}$, $U_n = \begin{pmatrix}\mathbb{P}([X_n = 1]) & \mathbb{P}([X_n = 2]) & \mathbb{P}([X_n = 3])\end{pmatrix}$.

1. Donner la matrice de transition associée à la chaîne de Markov $(X_n)_{n\in\mathbb{N}}$.

2. Écrire une fonction `Python` prenant en argument d'entrée un entier naturel `n`, une matrice `A` et un état initial `U0` et renvoyant en sortie la matrice U_n associée à la chaîne de Markov d'état initial `U0` et de matrice de transition `A`.

```python
import numpy as np
import numpy.linalg as al
import matplotlib.pyplot as plt

def etat(n,A,U0):
    U=np.dot(U0,al.matrix_power(A,n))
    return U
```

☞ Rappels...
- Produit matriciel AB :
- Puissance A^n :

3. Écrire une fonction **Python** de sorte que l'exécution de **graphique(N,A,U0)** renvoie le graphique représentant **N** premiers états probabilistes de la chaîne de Markov initialisée à **U0** et de matrice de transition **A**.

```
def graphique(N,A,U0):
    l,c=np.shape(U0)
    for i in range(0,c):
        L=[etat(n,A,U0)[0,i] for n in range(N)]
        plt.plot(range(N),L,"+",label='$\mathbb{P}([X_n=$'+str(i+1)+'])')
    plt.legend()
    plt.show()
```

4. A l'aide de la commande **al.eig**, écrire une fonction prenant en argument une matrice stochastique A et renvoyant en sortie un état stable de la chaîne de Markov associée.

```
import numpy as np
import numpy.linalg as al

def stable(A):
    Sp,VP=al.eig(np.transpose(A))
    k=0
    while np.round(Sp[k],5)!=1:
        k=k+1
    V=VP[:,k]
    V=V/sum(V)
    return V.real
```

✘ **Attention !**
$U = UA \iff {}^tA\,{}^tU = {}^tU$...