Calcul matriciel

Multiplication par un scalaire

somme

Récapitulatif séance précédente

Ecrire les fonctions suivantes :

ligne(m,i)

- **en entrée :** une matrice **m** et un indice de ligne **i**
- **en sortie :** la ième ligne sous forme de liste

```
M=[[1,2],[3,4],[5,6]]        ligne(M,1)→[3,4]
```

colonne(m,j)

- **en entrée :** une matrice **m** et un indice de colonne **j**
- **en sortie :** la jème colonne sous forme de liste

```
M=[[1,2],[3,4],[5,6]]        colonne(M,0)→[1,3,5]
```

Récapitulatif séance précédente

Nouvelle fonction

parcourir(m)

- **en entrée :** une matrice **m** (liste de listes)
- **en sortie :** la liste de toutes les éléments de **m**

```
M=[[1,2],[3,4],[5,6]]
```

```
parcourir(M)→[1,2,3,4,5,6]
```

mult(m,x)

- **en entrée :** une matrice **m** (liste de listes) et un réel **x**
- **en sortie :** la liste de toutes les éléments de **m** multipliés par **x**

```
M=[[1,2],[3,4],[5,6]]
```

```
mult(M,2)→[2,4,6,8,10,12]
```

Récapitulatif séance précédente

Nouvelle fonction

multi_scal(m,x)

- **en entrée :** une matrice **m** (liste de listes) et un réel **x**
- **en sortie :** une liste de liste, de même dimensions que **M**, contenant ses valeurs multipliées par **x**.

```
M=[[1,2],[3,4],[5,6]]   multi_scal(M,2)→[[2,4],[6,8],[10,12]]
```

Multiplication par un scalaire

Si $M = \begin{pmatrix} 5 & -3 \\ 1 & 2 \\ 0 & 1 \end{pmatrix}$ alors $2M = \begin{pmatrix} 10 & -6 \\ 2 & 4 \\ 0 & 2 \end{pmatrix}$

Définition mathématique

Soit $M = (m_{ij})_{\substack{1 \le i \le n \\ 1 \le J \le p}}$ une matrice d'ordre (n, p) et x un réel.

La matrice xM ou $x \times M$ est la matrice de terme général $xm_{ij} = x \times m_{ij}$: $xM = (xm_{ij})_{\substack{1 \le i \le n \\ 1 \le j \le p}}$

Addition de matrices

Si $M = \begin{pmatrix} 5 & -3 \\ 1 & 2 \\ 0 & 1 \end{pmatrix}$ et $N = \begin{pmatrix} 1 & -1 \\ 2 & -2 \\ 0 & -1 \end{pmatrix}$ alors $M + N = \begin{pmatrix} 5+1 & -3-1 \\ 1+2 & 2-2 \\ 0+0 & 1-1 \end{pmatrix} = \begin{pmatrix} 6 & -4 \\ 3 & 0 \\ 0 & 0 \end{pmatrix}$

Définition mathématique

Soit $M = (m_{ij})_{\substack{1 \le i \le n \\ 1 \le j \le p}}$ et $N = (n_{ij})_{\substack{1 \le i \le n \\ 1 \le j \le p}}$ deux matrices de même ordre (n, p).

La somme $M + N$ est la matrice de terme général $m_{ij} + n_{ij}$: $M + N = (m_{ij} + n_{ij})_{\substack{1 \le i \le n \\ 1 \le j \le p}}$

Multiplication par un scalaire

Si $M = \begin{pmatrix} 5 & -3 \\ 1 & 2 \\ 0 & 1 \end{pmatrix}$ alors $2M = \begin{pmatrix} 10 & -6 \\ 2 & 4 \\ 0 & 2 \end{pmatrix}$

Fonction ***multscal(M,x)*** :
- prend en paramètres une liste de listes M représentant une matrice et un réel x
- renvoie la liste de listes associée au produit xM

Définition mathématique

Soit $M = (m_{ij})_{\substack{1 \le i \le n \\ 1 \le J \le p}}$ une matrice d'ordre (n, p) et x un réel.

La matrice xM ou $x \times M$ est la matrice de terme général $xm_{ij} = x \times m_{ij}$: $xM = (xm_{ij})_{\substack{1 \le i \le n \\ 1 \le j \le p}}$

Addition de matrices

Si $M = \begin{pmatrix} 5 & -3 \\ 1 & 2 \\ 0 & 1 \end{pmatrix}$ et $N = \begin{pmatrix} 1 & -1 \\ 2 & -2 \\ 0 & -1 \end{pmatrix}$ alors $M + N = \begin{pmatrix} 5+1 & -3 \\ 1+2 & 2 \\ 0+0 & 1 \end{pmatrix}$

Fonction ***som_mat(M,N)*** :
- prend en paramètres deux listes de listes M et N représentant deux matrices de mêmes dimensions
- renvoie la liste de listes associée à la somme M+N

Définition mathématique

Soit $M = (m_{ij})_{\substack{1 \le i \le n \\ 1 \le j \le p}}$ et $N = (n_{ij})_{\substack{1 \le i \le n \\ 1 \le j \le p}}$ deux matrices de même ordre (n, p).

La somme $M + N$ est la matrice de terme général $m_{ij} + n_{ij}$: $M + N = (m_{ij} + n_{ij})_{\substack{1 \le i \le n \\ 1 \le j \le p}}$

Python

```
def multiscal(mat,x) :
    n=len(mat)
    p=len(mat[0])
    # Initialisation de « prod », la matrice résultat, par une liste vide
    prod=[]
    # Double boucle pour parcourir tous les éléments de la matrice et les multiplier par x
    for i in range(n):
        prod.append([])
        for j in range(p):
            prod[i].append(x*mat[i][j])
    return(prod)
```

Python

$$M = \begin{pmatrix} 5 & -3 \\ 1 & 2 \\ 0 & 1 \end{pmatrix} \Rightarrow 2M = \begin{pmatrix} 10 & -6 \\ 2 & 4 \\ 0 & 2 \end{pmatrix}$$

```
def multiscal(mat,x) :
    n=len(mat)
    p=len(mat[0])
    prod=[]
    for i in range(n):
        prod.append([])
        for j in range(p):
            prod[i].append(x*mat[i][j])
    return(prod)
```

mat=[[5,3],[1,2],[0,1]] x=2 Initialisation n=3 P=2 prod=[]	**i=0**	**i=1**	**i=2**
	prod=[**[]**] Pour j de 0 à 1 : prod[**0**].append(x*mat[**0**][j]) Résultat prod=[[10,6]]	prod=[[10,6],**[]**] Pour j de 0 à 1 : prod[**1**].append(x*mat[**1**][j]) Résultat prod=[[10,6],[2,4]]	prod=[[10,6]],[2,4],**[]**] Pour j de 0 à 1 : prod[**2**].append(x*mat[**2**][j]) Résultat prod=[[10,6],[2,4],[0,2]]

Python

```
def multiscal(mat,x) :
    n=len(mat)
    p=len(mat[0])
    # Initialisation de « prod », la matrice résultat : on crée une
    # matrice de dimension(n,p) remplie de valeurs manquantes

    prod=[[None for j in range(p)] for i in range(n)]
    # Double boucle pour parcourir tous les éléments de la matrice et les multiplier par x
    for i in range(n):
        for j in range(p):
            prod[i][j]=x*mat[i][j]
    return(prod)
```

Python

$$M = \begin{pmatrix} 5 & -3 \\ 1 & 2 \\ 0 & 1 \end{pmatrix} \Rightarrow 2M = \begin{pmatrix} 10 & -6 \\ 2 & 4 \\ 0 & 2 \end{pmatrix}$$

```
def multiscal(mat,x) :
    n=len(mat)
    p=len(mat[0])
    prod=[[None for j in range(p)] for i in range(n)]
    for i in range(n):
        for j in range(p):
            prod[i][j]=x*mat[i][j]
    return(prod)
```

`mat=[[5,3],[1,2],[0,1]]`
`x=2`
 	
Initialisation
`n=3`
`P=2`
`prod=[[0,0],[0,0],[0,0]]`
	i=0
	`Pour j de 0 à 1 :`
	`prod[0][j]=2*mat[0][j]`
	
	Résultat
	`prod=[[10,6],[0,0],[0,0]]`
		i=1
		`Pour j de 0 à 1 :`
		`prod[1][j]=2*mat[1][j]`
		
		Résultat
		`prod=[[10,6],[2,4],[0,0]]`
			i=
			`Pour j de 0 à 1 :`
			`prod[2][j]=`
			`2*mat[2][j]`
			
			Résultat
			`prod=[[10,6],[2,4],`
			`[0,2]]`

Python

```
def multiscal(mat,x) :
    n=len(mat)
    p=len(mat[0])
    # définition "en compréhension" du produit
    return [[x*mat[i][j] for j in range(p)] for i in range(n)]
```

Python

```
def multiscal(mat,x) :
    n=len(mat)
    p=len(mat[0])
    # définition "en compréhension" du produit
    return [[x*mat[i][j] for j in range(p)] for i in range(n)]
```

$$M = \begin{pmatrix} 5 & -3 \\ 1 & 2 \\ 0 & 1 \end{pmatrix} \Rightarrow 2M = \begin{pmatrix} 10 & -6 \\ 2 & 4 \\ 0 & 2 \end{pmatrix}$$

mat=[[5,3],[1,2],[0,1]] x=2 Initialisation n=3 P=2	**i**	[x*mat[i][j] pour j allant de 0 à 1]	**Résultat**
	i=0	[x*mat[**0**][j] pour j allant de 0 à 1]	**[10,6]**
	i=1	[x*mat[**1**][j] pour j allant de 0 à 1]	**[2,4]**
	i=2	[x*mat[**2**][j] pour j allant de 0 à 1]	**[0,2]**
Résultat [[10,6],[2,4],[0,2]]			

Python

```
def multiscal(mat,x) :
    n=len(mat)
    p=len(mat[0])
    prod=[[None for j in range(p)] for i in range(n)]
    for i in range(n):
        for j in range(p):
            prod[i][j]=x*mat[i][j]
    return(prod)
```

```
def multiscal(mat,x) :
    n=len(mat)
    p=len(mat[0])
    prod=[]
    for i in range(n):
        prod.append([])
        for j in range(p):
            prod[i].append(x*mat[i][j])
    return(prod)
```

```
def multiscal(mat,x) :
    n=len(mat)
    p=len(mat[0])
    # définition "en compréhension" du produit
    return [[x*mat[i][j] for j in range(p)] for i in range(n)]
```

Addition de matrices

Si $M = \begin{pmatrix} 5 & -3 \\ 1 & 2 \\ 0 & 1 \end{pmatrix}$ et $N = \begin{pmatrix} 1 & -1 \\ 2 & -2 \\ 0 & -1 \end{pmatrix}$ alors $M + N = \begin{pmatrix} 5+1 & -3-1 \\ 1+2 & 2-2 \\ 0+0 & 1-1 \end{pmatrix} = \begin{pmatrix} 6 & -4 \\ 3 & 0 \\ 0 & 0 \end{pmatrix}$

Définition mathématique

Soit $M = (m_{ij})_{\substack{1 \le i \le n \\ 1 \le j \le p}}$ et $N = (n_{ij})_{\substack{1 \le i \le n \\ 1 \le j \le p}}$ deux matrices de même ordre (n, p).

La somme $M + N$ est la matrice de terme général $m_{ij} + n_{ij}$: $M + N = (m_{ij} + n_{ij})_{\substack{1 \le i \le n \\ 1 \le j \le p}}$

```
def som_mat(mat1,mat2):
n=len(mat1)
    p=len(mat1[0])
    # définition "en compréhension" de la somme
    return [[mat1[i][j]+mat2[i][j] for j in range(p)] for i in range(n)]
```

Transposition

La transposée d'une matrice est la matrice obtenue en échangeant les lignes et les colonnes.

Par exemple, si $M = \begin{pmatrix} 5 & -3 \\ 1 & 2 \\ 0 & 1 \end{pmatrix}$ alors ${}^tM = \begin{pmatrix} 5 & 1 & 0 \\ -3 & 2 & 1 \end{pmatrix}$

Définition mathématique

Soit $M = (m_{ij})_{\substack{1 \le i \le n \\ 1 \le J \le p}}$ une matrice d'ordre (n, p).

La transposée de M est la matrice ${}^tM = (t_{ij})_{\substack{1 \le i \le p \\ 1 \le j \le m}}$ d'ordre (p, n), telle que : $t_{ij} = m_{ji}$

Transposition

La transposée d'une matrice est la matrice obtenue en échangeant les lignes et les colonnes.

Par exemple, si $M = \begin{pmatrix} 5 & -3 \\ 1 & 2 \\ 0 & 1 \end{pmatrix}$ alors ${}^tM = \begin{pmatrix} 5 & 1 & 0 \\ -3 & 2 & 1 \end{pmatrix}$

Définition mathématique

Soit $M = (m_{ij})_{\substack{1 \le i \le n \\ 1 \le J \le p}}$ une matrice d'ordre (n, p).

La transposée de M est la matrice ${}^tM = (t_{ij})_{\substack{1 \le i \le p \\ 1 \le j \le m}}$ d'ordre (p, n), telle que : $t_{ij} = m_{ji}$

```
def transpose(mat):
    n=len(mat)
    p=len(mat[0])
    # définition "en compréhension" de la transposition
    return [[mat[i][j] for i in range(n)] for j in range(p)]
```