L3 E - S5 2018-19 --- TD ordinateurs --- séance 5

Contents

1 Calcul matriciel 1
- 1.1 Addition et multiplication(s) . . . 2
- 1.2 Sélections d'éléments . . . 2
- 1.3 Des matrices particulières . . . 3
  - 1.3.1 Matrices nulles . . . 3
  - 1.3.2 Matrices " identité " . . . 4
- 1.4 Fonctions basiques (et diagonalisation) . . . 4

2 Exercice 1 4

3 Résolution d'un système linéaire 5

4 Exercice 2 5

5 Exercice 3 5

6 Exercice 4 6

7 Exercice 5 6

8 Exercice 6 : la méthode du pivot de Gauss 6
- 8.1 Transformations élémentaires . . . 6
- 8.2 Remarques sur ces transformations . . . 6
- 8.3 Enoncé de l'exercice . . . 7

1 Calcul matriciel

```
In [ ]: from pylab import *
```

Voici des exemples de matrices (et de vecteurs, un vecteur colonne étant une matrice à une seule colonne et un vecteur ligne étant une matrice à une seule ligne) :

```
In [ ]: v= array([4,7,9])                    ### a est un vecteur ligne
        print('v=',v,'\n')
        w=array([[2],[-5],[1],[9]])              ### w est un vecteur colonne
        print("w=",w,'\n')
        m = array([[1, 2, 3], [4, 5, 6]])             ### m est une matrice (2,3)
        print ("m=",m,'\n')
        M = array([[1, 2, 3,4,5], [6,7,8,9,10],[11,12,13,14,15]])      ### M est une matrice (3,
        print ("M=",M,'\n')
```

Quelques fonctions élémentaires :

```
In [ ]: print (size(v),shape(v))
        print (size(w),shape(w))
        print (size(m),shape(m))
        print (size(M),shape(M))
```

1.1 Addition et multiplication(s)

On peut bien entendu additionner des matrices de même type (ou shape) :

```
In [ ]: A = array([[-1, 0, 3.5], [1, 1, -4]])
        m+A
```

Par contre, l'addition :

```
In [ ]: m+M
```

n'est bien entendu pas possible. De la même façon que l'addition matricielle se fait terme à termes entre matrices de même taille, python accepte une multiplication terme à terme :

```
In [ ]: print('m=',m,'\n')
        print('A=',A,'\n')
        print('m*A=',m*A)
```

La multiplication matricielle usuelle se note avec la fonction dot :

```
In [ ]: dot(m,M)   ## m est de type (2,3) et M de type (3,5)
```

```
In [ ]: print(M)
        print('\n')           ### Pour une présentation plus aérée, on saute une ligne
        N=M.T                 ### N est la transposée de M
        print(N)
        print('\n')
        print ('MN=\n',dot(M,N),'\n')
        print ('NM=\n',dot(N,M))
```

1.2 Sélections d'éléments

```
In [ ]: c = array([[1, 2, 3,4,5], [6,7,8,9,10],[11,12,13,14,15]])
        print('c=\n',c)


        print('\n\n')      ## je saute 2 lignes pour rendre l'affichage plus clair

        print (c[1])       ## j'affiche la ligne d'indice 1 (pour python)
        print (type (c[1]))
        print (shape(c[1]))


        ####### ATTENTION !!!! Les indexations avec python commencent toujours en 0 avec python
```

```
        ####### Donc c[1] est en fait, pour nous, L2, la deuxième ligne
        ####### Si on veut L1, la 1ère ligne de la matrice, il faut demander d'imprimer c[0]

        print ('\n')
        print ('la ligne L1 de la matrice M est ',c[0])
        print ('\n')
        print (c[2:],'\n')

        print (c[0,0])           ### RAPPEL : le premier indice est toujours 0
        print (c[2,4])

        print (c[:1],'\n')       ### affichage de la 1ère  ligne de la matrice
        print (c[:2])            ### affichage des 2 premières lignes
        print ('\n')

        print (c[2:])            ### On enlève les 2 premières lignes
        print ('\n')
        print (c[1:2])           ### On enlève la première et la dernière lignes
        print('\n')

        print (len(c[1]))        ### nombre d'éléments dans une ligne

        print ('\n')
```

Et pour afficher les colonnes :

```
In [ ]: print('c=\n',c,'\n')
        print (c[:,:1],'\n')      ### affichage de la 1ère colonne
        print (c[:,2:])           ### on enlève les 2 premières colonnes
        print ('\n')
        print (c[:,4:])           ### on enlève les 4 premières colonnes
        print ('\n')

        print (c[:,2:4],'\n')           ### on garde les colonnes :
                                        ### à partir de l'indice 2 et avant l'indice 4

        print (c[:,0:1])        ### on enlève des colonnes : avant 0 et à partir de 1
                                ### il ne reste donc que la colonne 0 !!
        print ('\n')
```

1.3 Des matrices particulières

1.3.1 Matrices nulles

```
In [ ]: print (zeros(4),'\n')

        print (zeros((3,5)) ,'\n')
```

```
In [ ]: print (ones (3))  ## comme zeros... en remplaçant 0 par 1...
        print ('\n')
        print (ones((2,5)))
```

1.3.2 Matrices " identité "

```
In [ ]: print(eye(3),'\n')
        eye(5)
```

1.4 Fonctions basiques (et diagonalisation)

```
In [ ]: A=array([[-7,1,9],[-12,3,12],[-6,1,8]])

        print ('A=\n',A, '\n')

        ##### Calcul de l'inverse (il faut que la matrice soit inersible !!!)

        print ('l\'inverse de A est :\n',inv(A), '\n')

        ### calcul du déterminant

        print ('det(A)=',det(A))

        ##### Diagonalisation de A

        D,P=eig(A)
        print ('D=',D,'\n')     ### les valeurs propres
        Diag=zeros((3,3))
        for i in range(3):      ### on écrit la matrice diagonale associée à X
            Diag[i,i]=D[i]
        print ('La matrice diagonale est\n',Diag,'\n')

        print ('La matrice de changement de base est P=\n',P,'\n') #### les vecteurs propres


        ##### Formule de changement de base

        B=dot(dot(P,Diag),inv(P))  ### B=P.Diag.inv(P)
        print (B)                  ### on vérifie que B=A...
        print(A-B)                 ### cependant...
```

Bien entendu, le calcul précédent a fonctionné parce qu'on avait pris une matrice diagonalisable...

2 Exercice 1

Cherchez par vous-même (i.e. avec une feuille et un stylo) les vecteurs propres de la matrice

$$A = \begin{pmatrix} -7 & 1 & 9 \\ -12 & 3 & 12 \\ -6 & 1 & 8 \end{pmatrix}$$

et comparez avec les vecteurs propres trouvés par python.

3 Résolution d'un système linéaire

Python est équipé pour résoudre les systèmes linéares :

```
In [ ]: M = array([[2,1,1],[1,1,1], [1,2,1]])
        X = array([1,2,3])
        Y = solve(M, X)
        print (Y)
```

```
In [ ]: MM = array([[2,1,1],[1,1,1], [5,2,2]])
        X = array([1,2,3])
        Y = solve(MM, X)
        print (Y)
```

```
In [ ]: MM = array([[2,1,1],[1,1,1], [5,2,2]])
        X = array([2,-1,7])
        Y = solve(MM, X)
        print (Y)
```

4 Exercice 2

Dans les derniers exemples, on demande à python d'étudier les systèmes :

$$\begin{cases} 2x + y + z = 1 \\ x + y + z = 2 \\ 5x + 2y + 2z = 3 \end{cases}$$

puis

$$\begin{cases} 2x + y + z = 2 \\ x + y + z = -1 \\ 5x + 2y + 2z = 7 \end{cases}$$

Dans les deux cas, il renvoie que ce n'est pas possible. Etes-vous d'accord avec lui ?

5 Exercice 3

Notations : Si $M = (m_{ij})_{1 \leq i \leq n, 1 \leq j \leq p}$ est une matrice à n lignes et p colonnes, on notera L_i (pour $1 \leq i \leq n$) la i-ème ligne de M et C_j (pour $1 \leq j \leq p$), la j-ème colonne de M (Evidemment, s'il ya un risque de confusion, on pourra préciser $L_i(M)$ ou $C_j(M)$ pour préciser qu'il s'agit bien des lignes et colonnes de la matrice M.

Ecrire une fonction `multiplie(M,i,a)` qui renvoie la matrice M avec la ligne L_i multipliée par le nombre a.

6 Exercice 4

1- Si M est une matrice, l'instruction `M[2]` va donner la ligne L_3 de M (si cette ligne existe...). Et, de façon générale, l'instruction `M[i]` va donner la ligne L_{i+1} de M (si cette ligne existe...). Pouvez-vous expliquer pourquoi ?

2- Ecrire une fonction `echange(M,i,j)` qui échange les lignes L_i et L_j de la matrice M.

3- En déduire une instruction qui retourne la colonne i d'une matrice et une fonction qui échange les colonnes C_i et C_j d'une matrice.

7 Exercice 5

Pour i et j fixés et distincts, $E_{ij} = (e_{ij})$ est la matrice dont l'entrée e_{ij} vaut 1 et toutes les autres entrées sont nulles. Pour tout nombre réel a et pour $1 \leq i, j \leq n$, la matrice carrée d'ordre n égale à $I_n + aE_{ij}$ est appelée matrice de transvection~; par exemple~:

$$I_4 - 3E_{42} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & -3 & 0 & 1 \end{pmatrix}$$

1- Si M est une matrice carrée d'ordre n, quel est le résultat du produit $(I_n + aE_{ij}).M$ (avec $i \neq j$ et $1 \leq i, j \leq 4$) ?

2- Ecrire une fonction `tranvection(M,i,j,a)` qui renvoie la matrice $(I_n + aE_{ij}).M$ si M est une matrice carrée d'ordre n (et $i \neq j$, $1 \leq i, j \leq n$ et $a \in \mathbf{R}$).

8 Exercice 6 : la méthode du pivot de Gauss

8.1 Transformations élémentaires

• $T_1 : L_i \to L_i + \alpha L_j$ (avec $i \neq j$: ajouter à une ligne un multiple d'une <u>autre</u> ligne),

•• $T_2 : L_i \to \alpha L_i$, $\alpha \neq 0$ (multiplier une ligne par un scalaire non nul),

••• $T_3 : L_i \leftrightarrow L_j$ (échanger deux lignes).

Si M est inversible, ces transformations élémentaires permettent de passer de M à I_n :

$$\begin{pmatrix} * & * & * \\ * & * & * \\ & * & * \end{pmatrix} \xrightarrow{T_1} \begin{pmatrix} * & * & * \\ 0 & * & * \\ 0 & * & * \end{pmatrix} \xrightarrow{T_1} \begin{pmatrix} * & * & * \\ 0 & * & * \\ 0 & 0 & * \end{pmatrix} \xrightarrow{T_1} \begin{pmatrix} * & * & 0 \\ 0 & * & 0 \\ 0 & 0 & * \end{pmatrix} \xrightarrow{T_1} \begin{pmatrix} * & 0 & 0 \\ 0 & * & 0 \\ 0 & 0 & * \end{pmatrix} \xrightarrow{T_2} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

8.2 Remarques sur ces transformations

1- Cette méthode n'aboutit que si la matrice M de départ est inversible...

2- La transformation T_1 nécessite d'avoir un `pivot` (l'élément diagonal sur lequel on s'appuie) non nul ; lorsqu'il est nul, on utilise la transformation T_3 pour faire apparaître un pivot non nul.

3- Le schéma ci-dessus est théorique : dans la pratique, il y aura souvent intérêt à utiliser T_2 pour se ramener à des matrices d'entiers ou à effectuer directement des manipulations du type "remplacer L_i par $\alpha L_i + \beta L_j$".

On note que les transformations T_i sont inversibles et que si

$$I_n = T_k T_{k-1} \dots T_2 T_1 M$$

cela signifie que

$$M^{-1} = T_k T_{k-1} \dots T_2 T_1 I_n$$

Avec $B = \begin{pmatrix} 1 & -1 & 2 \\ 2 & 0 & 4 \\ 1 & 1 & 3 \end{pmatrix}$, on trouve $B^{-1} = \begin{pmatrix} -2 & 5/2 & -2 \\ -1 & 1/2 & 0 \\ 1 & -1 & 1 \end{pmatrix}$.

- Cela permet aussi de résoudre des systèmes ; par exemple, $BX = (7, 10, 4)$ a pour solution $X = (3, -2, 1)$.
- Cette méthode permet aussi de calculer le déterminant (en faisant attention aux cas où on a multiplié une ligne par un scalaire) ; par exemple, ici on voit que $\det B = 2$.
- Cette méthode s'applique également de la même façon en agissant sur les colonnes. Par contre, il ne faudra pas l'appliquer en mélangeant les 2 méthodes (par les lignes et par les colonnes).

8.3 Enoncé de l'exercice

1- Ecrire une fonction `gauss(M)` qui par suites de transformations élémentaires transforme la matrice carrée M en la matrice de l'identité.

2- Ecrire une fonction `GaussInv(M)` qui retourne l'inverse de la matrice carrée M en appliquant la méthode de Gauss.

3- Tester votre programme sur la matrice B donnée en exemple ci-dessus.