R1.07 TP 3

Parcours d'une matrice

9 janvier 2026 – B. Colombel

Un compte-rendu de ce TP (réponses aux exercices et aux questions) est à rédiger sur le notebook *Jupyter* et votre compte rendu doit être déposé sous le nom

`prenom.nom-TP3.ipynb`

dans le dossier relatif à ce TP sur Ametice (R1.07, TP4).

1 Algorithme de parcours d'une matrice

Nous allons commencer à programmer des fonctions *matricielles* avec *sagemath*. Il s'agit de fonctions de la forme :

$$\begin{array}{rccc} f: & \mathcal{M}_{n,p}(\mathbb{R}) & \to & M_{n',p'}(\mathbb{R}) \\ & A & \mapsto & B \end{array}$$

En résumé :
- f est le nom de la fonction ;
- elle prend en entrée une matrice $A \in \mathcal{M}_{n,p}(\mathbb{R})$;
- elle renvoie en sortie une matrice $B \in \mathcal{M}_{n',p'}(\mathbb{R})$ (si elle existe).

Un grand nombre de fonctions matricielles reposent sur l'algorithme permettant de parcourir l'ensemble des coefficients d'une matrice :

Parcours d'une matrice A

Entrées : Une matrice A de dimension $n \times p$
début
  pour i *allant de 1 à* n **faire**
    pour j *allant de 1 à* p **faire**
      instructions à faire sur le coefficient a_{ij}
    fin
  fin
fin

Exemple 1. La fonction ci-contre `diagonale(A)` prend en entrée une matrice A de taille $p \times n$ quelconque, et renvoie une matrice D de même taille telle que :

$$d_{ij} = \begin{cases} a_{ij} \text{ si } i = j \\ 0 \text{ si } i \neq j \end{cases}$$

[1]:
```
def diagonale(A) :
    n = A.nrows()
    p = A.ncols()
    M = []
    for i in range(n) :
        for j in range(p) :
            if i == j :
                M.append(A[i, j])
    return diagonal_matrix(M)

A = matrix([[2, -1, -1, -2], [4,  4,  2,  0], [0,  4, -2,  0], [-1, -1,  0, -3]])
A, diagonale(A)
```

[1]:

```
(
[ 2 -1 -1 -2]  [ 2  0  0  0]
[ 4  4  2  0]  [ 0  4  0  0]
[ 0  4 -2  0]  [ 0  0 -2  0]
[-1 -1  0 -3], [ 0  0  0 -3]
)
```

Exercice 1. Écrire une fonction `minimum` qui trouve la valeur minimale des coefficients d'une matrice A et sa position `(i_min,j_min)` dans la matrice A.

2 Opération élémentaire sur les lignes et les colonnes

Dans certains cas, on peut travailler directement sur toute une ligne ou toute une colonne d'une matrice.
Si M est une matrice

- la commande `M[i, :]` correspond à la **ligne** numéro `i` de la matrice ;
- la commande `M[:, j]` correspond à la **colonne** numéro `j` de la matrice.

[4]:
```
M = matrix(2, 3, [1,2,3,4,5,6])
M
```

[4]:
```
[1 2 3]
[4 5 6]
```

[5]:
```
M[0, :]
```

[5]:
```
[1,2,3]
```

[6]:
```
M[:, 1]
```

[6]:
```
[2]
[5]
```

Remarque importante

Si on pose `A = matrix([[1,2],[3,4]])`, puis `B = A`, puis que l'on modifie la matrice A par `A[0,0] = 7`. Que se passe-t-il ?

[7]:
```
A=matrix([[1,2],[3,4]])
B = A
A[0, 0] = 7
A, B
```

[7]:
```
(
[7 2]  [7 2]
[3 4], [3 4]
)
```

La matrice B est aussi modifiée !

Explications

- A ne contient pas les coefficients de la matrice, mais l'adresse où sont stockés ces coefficients (c'est plus léger de manipuler l'adresse d'une matrice que toute la matrice).
- Comme A et B pointent vers la même zone qui a été modifiée, les deux matrices A et B sont modifiées.
- Pour pouvoir définir une copie de A , avec une nouvelle adresse mais les mêmes coefficients, on écrit `B = copy(A)` . Les modifications sur A ne changeront plus B.

Exercice 2. La méthode du pivot de Gauss repose sur trois opérations élémentaires les lignes :
- opération 1 : $L_i \leftarrow cL_i$ avec $c \neq 0$: on multiplie une ligne par un réel ;
- opération 2 : $L_i \leftarrow L_i + cL_j$: on ajoute à la ligne L_i un multiple d'une autre ligne L_j ;
- opération 3 : $L_i \leftrightarrow L_j$: on échange deux lignes.

Écrire les fonctions `op1(A, i, c)`, `op2(A, i, j, c)` et `op3(A, i, j)` correspondant à ces trois opérations élémentaires.

Attention ! ! la matrice A ne devra pas être modifiée pendant l'exécution de cette fonction ! Pour cela, il faudra écrire `AA = copy(A)` ou `AA = A.copy()` et travailler sur cette copie.

3 Algorithme du pivot du Gauss

La méthode du pivot de Gauss est décrite par l'algorithme suivant.

Méthode d'élimination de Gauss-Jordan

Entrées : $C = (c_{ij})$ la matrice augmentée
1 **début**
2 **pour** k *allant de 1 à* n **faire**
3 choix du pivot $c_{k'k'} \neq 0$ avec $|c_{k'k'}|$ le plus grand possible ;
4 échanger les lignes L_k et $L_{k'}$;
5 $pivot \leftarrow c_{kk}$;
6 **pour** j *allant de* $(k+1)$ *à* n **faire**
7 $L_j \leftarrow L_j - \dfrac{c_{jk}}{pivot} L_k$;
8 **fin**
9 **fin**
10 **fin**

On retrouve lignes 4 et 7 nos fonctions `echanger(C, k, k')` et `transvection(C, j, k, mu)` avec

$$\mu = -\frac{c_{jk}}{pivot}$$

Par contre, la ligne 3 correspond au choix du *pivot*. Ce choix ne s'effectue pas de la même façon que lorsque l'on déroule l'algorithme à la main où on a juste besoin qu'il soit non nul.

3.1 Choix du pivot

Mathématiquement, tous les pivots non nuls se valent. Il n'en est pas de même du point de vue numérique :
diviser par un pivot dont la valeur absolue est trop faible par rapport aux autres coefficients du système conduit à des erreurs d'arrondi importantes.

Pour éviter ces erreurs d'arrondi, on utilise comme pivot le plus grand coefficient en valeur absolue de l'inconnue à éliminer.

Exercice 3. Écrire une fonction en *sagemath* appelée `Pivot(A, j0)` chargée de la recherche du pivot de valeur absolue maximale dans une matrice $A = (a_{i,j})$ sur une colonne (colonne j_0) à partir de la ligne j_0 et qui **retourne le numéro de la ligne qui le contient**.

Par exemple, si $A = \begin{pmatrix} 1 & 3 & 2 & 4 \\ 0 & 0 & 3 & 7 \\ 0 & 1 & 5 & 1 \\ 0 & -2 & 1 & 1 \end{pmatrix}$ et $j = 2$ (2^{e} colonne) alors `Pivot(A, 2)` renvoie 4.

Remarque. La commande `L.index(cond)` renvoie l'indice de de l'élément de la liste `L` correspondant à la condition `cond`.
Par exemple, pour déterminer l'index de l'élément maximal d'une liste `L`, on écrira `L.index(max(L))`.

3.2 Pivot de Gauss

On considère un système linéaire s'écrivant sous la forme :

$$AX = B$$

où $A = (a_{ij})$ est une matrice carrée d'ordre n inversible.

Lorsque l'on s'occupe de la colonne d'indice i, l'algorithme du pivot de Gauss se déroule de la façon suivante :

1. rechercher un pivot : on le trouve à la ligne i_0 ;
2. échanger les lignes i et i_0 ;
3. pour chaque valeur de k entre $i+1$ et n, réaliser l'opération :

$$L_k \leftarrow L_k - \frac{a_{ki}}{a_{ii}} L_i$$

Et on passe ensuite à la colonne suivante

Exercice 4. Écrire une fonction en *sagemath* appelée `Gauss(A, B)` qui retourne la matrice triangulaire A' obtenu par cette méthode et le vecteur B' correspondant.

Par exemple, si :

$$A = \begin{pmatrix} 1 & 3 & 2 & 4 \\ 0 & 0 & 3 & 6 \\ 0 & 1 & 5 & 3 \\ 1 & 2 & 1 & 2 \end{pmatrix} \quad \text{et} \quad B = \begin{pmatrix} 3 \\ 3 \\ 4 \\ 2 \end{pmatrix}$$

on doit obtenir :

$$A' = \begin{pmatrix} 1 & 3 & 2 & 4 \\ 0 & 1 & 5 & 3 \\ 0 & 0 & 4 & 1 \\ 0 & 0 & 0 & 5{,}25 \end{pmatrix} \quad \text{et} \quad B' = \begin{pmatrix} 3 \\ 4 \\ 3 \\ 0{,}75 \end{pmatrix}$$

Bonus : Écrire un script qui permmette de résoudre le système écéddent, ie, qui effectue la *remontée* de l'algorithme de Gauss.