

FEUILLE D'EXERCICES n° 6

Algèbre linéaire

1. DÉCLARATIONS

1) On peut définir un vecteur et une matrice de la manière suivante.

```
w = vector([1,1,-4])
```

```
A = matrix([[1,2,3],[3,2,1],[1,1,1]]); A
```

Remarquons d'abord que les indices commencent à 0. Si l'on tape `A[0,0]`, on obtient 1.

2) Exécuter les commandes

```
A.det()
```

```
A*w
```

```
w*A
```

```
parent(A)
```

```
parent(w)
```

3) Comme les coefficients de A sont des entiers, A est considérée par défaut comme une matrice de $M_n(\mathbb{Z})$. Pour entrer la même matrice en temps qu'élément de $M_n(\mathbb{Q})$, on peut utiliser la commande

```
B = matrix(QQ, [[1,2,3],[3,2,1],[1,1,1]])
```

ou bien, comme A est déjà définie

```
B = matrix(QQ,A)
```

```
v = vector(QQ,[1,1,-4])
```

Vérifier ensuite ce que donnent `parent(B)` et `parent(v)`.

4) On peut également indiquer la taille, puis les coefficients :

```
C = matrix(2,3,[1,2,3,4/3,5,6])
```

```
C
```

```
parent(C)
```

5) Il est parfois commode d'utiliser une formule sur les indices. Par exemple :

```
C = matrix(3, lambda i,j: i-j); C
```

ou bien, pour une matrice rectangulaire

```
C = matrix(3,4, lambda i,j: i-j); C
```

ou bien, pour indiquer l'anneau (ou corps) de base

```
C = matrix(QQ, 3,4, lambda i, j: i-j); C
```

Définir de cette façon la matrice de $M_{3,4}(\mathbb{Z})$ dont tous les coefficients sont égaux à 1.

6) Pour les vecteurs, la commande

```
vector(QQ,3, lambda i: i^2)
```

ne fonctionne pas. On peut par exemple utiliser

```
vector(QQ, [i^2 for i in [0..2]])
vector(QQ, [i^2 for i in range(3)])
```

7) On peut aussi commencer par définir l'espace des matrices ou des vecteurs

```
MatrixSpace(ZZ,3,2)
VectorSpace(QQ,3)
```

Cette commande désigne un espace vectoriel, donc défini sur un corps; si l'on écrit ZZ à la place de QQ, on obtient un message d'erreur. Il faut écrire "module libre" à la place de "espace vectoriel".

```
FreeModule(ZZ,3)
```

Par exemple, pour définir la matrice A ci-dessus, on peut utiliser les commandes suivantes

```
M3Z = MatrixSpace(ZZ,3); M3Z
A = M3Z([1,2,3,3,2,1,1,1,1]); A
```

Vérifier ce que donne alors `parent(A)`. De même, pour définir B , connaissant déjà A :

```
M3Q = MatrixSpace(QQ,3); M3Q
B = M3Q(A); B
parent(B)
```

8) Définir $M_3(\mathbb{F}_3)$, puis la matrice A_{bar} , réduction de la matrice A modulo 3. Quelle commande donne une matrice choisie au hasard dans $M_3(\mathbb{F}_3)$?

9) Expérimenter les commandes

```
M3Q(0)
M3Q(1)
M3Q(2)
```

Pour définir la matrice identité, on peut aussi utiliser `identity_matrix`. Regarder ce que donnent les commandes

```
identity_matrix(4)
identity_matrix(QQ,4)
identity_matrix(RR,4)
```

10) Attention aux identifications. Essayer les commandes suivantes.

```
L=identity_matrix(4);L
L[1,0]=1;L
M=L
M[2,0]=-1
M
L
```

Refaire ensuite toute cette séquence en remplaçant `M=L` par `M=copy(L)`.

2. ADDITION, MULTIPLICATION, DÉTERMINANT, INVERSE

Tout cela se fait de façon naturelle.

```
M = M3Z.random_element()
N = M3Z.random_element()
M + N
M * N
M.determinant()
M.det()
M^2
Minv = M^(-1)
M*Minv
```

3. EXTRACTION, CONCATÉNATION

Essayons sur la matrice

```
M = matrix(6,4,lambda i,j: i+j)
```

On se souvient que la numérotation des indices commence à 0 : $M[0,0]$ donne 0.

1) Essayer

```
M[[0,2,3],1]
parent(_)
```

(le caractère « `_` » désigne le résultat de la dernière commande exécutée).

2) Essayer

```
M[[0,2,3], [1..3]]
```

Plus généralement, si $l1$ et $l2$ sont des listes, $M[l1, l2]$ donne la sous matrice de M formée des lignes de la liste $l1$ et des colonnes de la liste $l2$. Remarquons qu'il est possible de répéter des lignes, ou des colonnes :

```
M[[1,1], [0..3]+[0]]
```

Essayer aussi les commandes `column`, `row`, `nrows` et `ncols`.

3) On peut concaténer à M une matrice de 6 lignes. Prenons une matrice N au hasard dans $M_{6,3}(\mathbb{Q})$. Définir pour cela cet espace de matrice, puis utiliser la commande `random_element`. Pour concaténer M et N , on utilise

```
M.augment(N)
```

Si les coefficients de N ne sont pas entiers, on obtient alors un message d'erreur parce-que M est définie sur \mathbb{Z} et N sur \mathbb{Q} . Il faut remplacer M dans $M_{6,4}(\mathbb{Q})$.

```
M64Q = MatrixSpace(QQ,6,4)
M64Q(M).augment(N)
```

4) Construction par blocs. Définir une matrice carrée A . En utilisant la commande `block_matrix`, définir la matrice par blocs

$$\begin{pmatrix} A & -A \\ 0 & 2A - I_2 \end{pmatrix}$$

4. NOYAU, IMAGE, ÉQUATIONS LINÉAIRES

1) Exécuter la commande

```
KA = A.kernel(); KA
```

Remarquons que pour sage, le noyau d'une matrice A est le noyau à gauche, c'est-à-dire le noyau de l'application qui à x associe xA . Pour le noyau à droite, on utilise

```
A.right_kernel()
```

Bien sûr, le résultat de ces commandes dépend de l'anneau de base, c'est-à-dire de l'espace où se situe A . Avec A , on a obtenu un \mathbb{Z} -module alors que si l'on fait $KB = B.kernel()$, on obtient un \mathbb{Q} -espace vectoriel. Pour obtenir une base :

```
KB.basis()
```

Exécuter les commandes suivantes.

```
NoyauB=KB.basis()
u=NoyauB[0]
B*u
u*B
KBdroite=B.right_kernel();KBdroite
KBd=B.right_kernel_matrix();KBd
B*(KBd.transpose())
```

2) Pour l'image, on peut utiliser la commande $B.image()$, qui donne l'image à gauche, c'est-à-dire l'espace vectoriel engendré par les lignes de B . On peut pour cela également utiliser $B.row_module()$. De même, l'image de B à droite est donnée par

```
B.column_module()
```

3) Voyons maintenant les résolutions d'équations.

```
Y = vector(QQ, [0, -4, -1])
```

On veut résoudre l'équation $BX = Y$:

```
X0 = B.solve_right(Y)
X0; B*X0
```

Cette commande `solve_right` ne donne qu'une solution $X0$. Comme on connaît aussi le noyau de B , on a l'ensemble de toutes les solutions de l'équation.

4) Reprenons l'équation différentielle de la feuille 2

$$(1) \quad (3t - 2)y'' + 2t^2y' - 2(2t - 1)y = 0$$

Nous allons utiliser le calcul matriciel pour calculer les solutions polynomiales de degré inférieur ou égal à 2 de cette équation différentielle, toujours en utilisant un polynôme à coefficients indéterminés. Comme dans la feuille 2, on définit :

```
PR = PolynomialRing(QQ, 3, 'a')
v = list(PR.gens())
PRt.<t> = PolynomialRing(PR)
p = v[2]*t^2+v[1]*t+v[0]
```

```

p1 = diff(p,t)
p2 = diff(p1,t)
q = (3*t-2)*p2+2*t^2*p1-2*(2*t-1)*p
l = q.coeffs()

```

puis on définit la matrice des coefficients de q :

```

M = matrix(QQ,3,3, lambda i,j: l[i].coefficient(v[j]))

```

Alors `M.right_kernel()` donne l'ensemble des polynômes de degré inférieur ou égal à 2 qui sont solutions de (1).

5) En utilisant `solve_right`, trouver les polynômes de degré inférieur ou égal à 2 qui sont solution de l'équation avec second membre

$$(3t - 2)y'' + 2t^2y' - 2(2t - 1)y = 2t^2 + 10t - 6.$$

6) Même question avec l'équation

$$(3t - 2)y'' + 2t^2y' - 2(2t - 1)y = -6t^2 + 10t + 2.$$

7) Écrire une fonction qui, étant donnés a, b, c, d dans $\mathbb{Q}[t]$ et un entier n utilise cette méthode pour donner les polynômes de degré inférieur ou égal à n , qui sont solutions de l'équation

$$ay'' + by' + cy = d.$$

5. RÉDUCTION

1) Soit $A = \begin{pmatrix} 1 & 2 & 3 \\ 3 & 2 & 1 \\ 1 & 1 & 1 \end{pmatrix}$. Appliquer à A les commandes

```

characteristic_polynomial
minimal_polynomial
eigenvalues
eigenvectors_right
eigenspaces_right
eigenmatrix_right

```

Cette dernière commande rend un couple de matrices $[B, U]$. Vérifier que $U^{-1}AU = B$.

2) Même exercice avec $B = \begin{pmatrix} 0 & 2 \\ 1 & 0 \end{pmatrix}$.

3) Avec cette matrice B , on utilise des approximation car ses valeurs propres sont $\pm\sqrt{2}$. On peut aussi manier formellement les valeurs exactes. Pour cela, définir $K = \mathbb{Q}(\sqrt{2})$, puis $M2K = M_2(K)$. Alors on indique que l'on veut considérer B comme élément de $M_2(K)$ en écrivant $B = M2K(B)$. Appliquer les fonctions précédentes à cette nouvelle matrice B .

4) Soit $M = \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 1 \\ 0 & 1 & -2 \end{pmatrix}$. Calculer son polynôme minimal m et vérifier qu'il est irréductible. Soit K un polynôme de rupture de m . Définir $M_3(K)$ et appliquer `eigenvalues` à M , considérée comme un élément de $M_3(K)$.

5) Même exercice avec la matrice $M = \begin{pmatrix} -3 & -1 & 1 \\ 5 & 2 & -1 \\ 1 & 6 & -1 \end{pmatrix}$. Commenter.

6) Reprenons la matrice $M = \begin{pmatrix} -3 & -1 & 1 \\ 5 & 2 & -1 \\ 1 & 6 & -1 \end{pmatrix}$. Définir $M3F7 = M_3(\mathbb{F}_7)$, et poser $M = M3F7(M)$. Essayer la fonction `eigenvalues` sur cette nouvelle matrice M . Définir l'espace de matrices dans lequel on pourra calculer les valeurs propres de M . Diagonaliser M dans cet espace.

7) Dans $M_5(\mathbb{Q})$, on considère les matrices suivantes.

$$A = \frac{1}{4} \begin{pmatrix} -1 & 1 & -4 & 4 & -3 \\ 1 & -1 & 4 & -4 & 3 \\ 10 & -2 & 12 & -8 & 6 \\ 7 & 1 & 4 & 0 & 3 \\ 6 & -6 & 8 & -8 & 10 \end{pmatrix}, \quad B = \frac{1}{4} \begin{pmatrix} 12 & -4 & 4 & -4 & 2 \\ -4 & 12 & -4 & 4 & -2 \\ 17 & -25 & 28 & -24 & 13 \\ 17 & -25 & 20 & -16 & 13 \\ -8 & 8 & -8 & 8 & 4 \end{pmatrix}.$$

Calculer $AB - BA$. Trouver une matrice $P \in GL_5(\mathbb{Q})$ telle que $P^{-1}AP$ et $P^{-1}BP$ soient diagonales.

6. MULTIPLICATION DE MATRICES : ALGORITHME DE STRASSEN

Soient A et B deux matrices $n \times n$. On cherche à calculer $C = AB$ de façon économique. On suppose pour commencer (questions 1 et 2) que n est une puissance de 2. On divise A , B et C en 4 matrices $n/2 \times n/2$.

$$A = \begin{pmatrix} A_1 & A_2 \\ A_3 & A_4 \end{pmatrix}, \quad B = \begin{pmatrix} B_1 & B_2 \\ B_3 & B_4 \end{pmatrix} \quad \text{et} \quad C = \begin{pmatrix} C_1 & C_2 \\ C_3 & C_4 \end{pmatrix}.$$

On pose

$$\begin{cases} P_1 = A_1(B_2 - B_4) \\ P_2 = (A_1 + A_2)B_4 \\ P_3 = (A_3 + A_4)B_1 \\ P_4 = A_4(B_3 - B_1) \\ P_5 = (A_1 + A_4)(B_1 + B_4) \\ P_6 = (A_2 - A_4)(B_3 + B_4) \\ P_7 = (A_1 - A_3)(B_1 + B_2) \end{cases}$$

1) Vérifier que $C_1 = P_4 + P_5 - P_2 + P_6$, $C_2 = P_1 + P_2$, $C_3 = P_3 + P_4$ et $C_4 = P_1 + P_5 - P_3 - P_7$.

2) En utilisant ces relations, écrire sur sage une fonction qui calcule le produit AB avec une complexité algébrique en $O(n^{\ln 7 / \ln 2})$.

3) Généraliser à n quelconque.

4) Comparer la complexité de l'algorithme obtenu avec celle de l'algorithme naïf issu de l'application directe de la formule

$$c_{ij} = \sum_{k=1}^n a_{ik} b_{kj}.$$