Université de Bordeaux
Mathématiques

Algèbre et calcul formel – Agrégation
2022–2023

FEUILLE D'EXERCICES n° 6

Algèbre linéaire

1. Déclaration

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

```
w = vector([1,1,-4])
A = matrix([[1,2,3],[4,5,6],[7,8,9]]); 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],[4,5,6],[7,8,9]])
```

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) On peut également utiliser une formule sur les indices, pour définir une matrice, 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 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 réellement 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 alors 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([1,2,3,4,5,6,7,8,9]); 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 `Abar`, 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)
```

2. Addition, multiplication, déterminant, inverse

Tout cela se fait de façon naturelle.

```
set_random_seed(42)
M = M3Z.random_element()
N = M3Z.random_element()
M + N
M * N
M.determinant()
M.det()
M^2
M.is_invertible()
```

```
Minv = M^(-1)
M*Minv
```

1) Expliquer pourquoi `M.is_invertible()` renvoie `False` alors que Sage arrive ensuite à calculer `M^(-1)`.

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]]
```

3) Les commandes

```
M.nrows()
M.ncols()
```

donnent les dimensions de M. 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)
```

on obtient alors un message d'erreur parce-que M est définie sur $\mathbb{Z}$ et N sur $\mathbb{Q}$. Il faut replacer M dans $M_{6,4}(\mathbb{Q})$.

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

4. Noyau, image, équations linéaires

1) Exécuter la commande

```
A = matrix(ZZ,3,3,[1,2,3,3,2,1,1,1,1])
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 `B = matrix(QQ,A); KB = B.kernel()`, on obtient un $\mathbb{Q}$-espace vectoriel. Pour obtenir une base :

```
KB.basis()
```

2) Pour l'image, on 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.

Si `Y` n'est pas dans l'image des colonnes de `B`, la fonction `solve_right` renvoie une erreur.

```
Y = vector(QQ,[1, -4, -1])
X0 = B.solve_right(Y)
```

5. Générateur pseudo-aléatoire

Fixons un entier premier q. Pour tout $a, b \in \mathbb{F}_q$, on définit la fonction

$$\begin{aligned} F_{a,b} : \mathbb{F}_q &\to \mathbb{F}_q \\ x &\mapsto a \cdot x + b \end{aligned}$$

On peut se servir d'une telle fonction pour générer du pseudo-aléa. Étant donnée une graine x_0, on définit une suite de nombres pseudo-aléatoires en faisant $x_{n+1} = F_{a,b}(x_n)$. Ces générateurs de pseudo-aléa (appelés générateur congruentiel linéaire[1]) ne sont pas très bons, et l'objectif de l'exercice est de montrer que même si a et b sont secrets, la connaissance de 3 termes consécutifs de la suite est suffisante pour les retrouver et également retrouver la graine x_0.
(Conséquence : il ne faut pas utiliser ce genre de générateurs pseudo-aléatoires en cryptographie).

1) Expliquer comment, étant donné x_n, x_{n+1} et x_{n+2}, on peut retrouver a et b en résolvant un système affine.

2) Expliquer comment, étant donné x_0, a et b, on peut calculer x_n en $O(\log n)$ opérations dans $\mathbb{F}_q$.
(Indice 1 : on définira v_n un vecteur de dimension 2 dépendant de x_n tel que $v_{n+1} = A \cdot v_n$, où A est une matrice 2×2.)

[1] `en.wikipedia.org/wiki/Linear_congruential_generator`

(Indice 2 : On pourra utiliser l'exponentiation rapide pour calculer v_n à partir de v_0 rapidement.)

3) Adapter la question précédente pour obtenir x_0 à partir de a, b et x_n en temps $O(\log n)$ (opérations dans $\mathbb{F}_q$).

4) On donne $q = 521$ et $x_{33} = 205, x_{34} = 473, x_{35} = 506$. Utiliser Sage pour retrouver a et b, puis la graine x_0.
(Solution : on doit trouver $x_0 = 42$.)

6. Réduction

1) Soit $A = \left(\begin{smallmatrix} 1 & 2 & 3 \\ 3 & 2 & 1 \\ 1 & 1 & 1 \end{smallmatrix}\right)$. 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 $B = \left(\begin{smallmatrix} 0 & 2 \\ 1 & 0 \end{smallmatrix}\right)$.

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 $M = \left(\begin{smallmatrix} 0 & 0 & 1 \\ 1 & 0 & 1 \\ 0 & 1 & -2 \end{smallmatrix}\right)$. Calculer son polynôme minimal m et vérifier qu'il est irréductible. Soit K un corps 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 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 $P \in \mathrm{GL}_5(\mathbb{Q})$ telle que $P^{-1}AP$ et $P^{-1}BP$ soient diagonales.

7. Algorithme de Strassen

Soient A et B deux matrices $n \times n$ sur un corps, où n est une puissance de 2. Nous voulons calculer le produit $C = AB$ le plus économiquement possible. On subdivise A, B et C en quatre blocs de taille $n/2 \times n/2$:

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

Soit

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

1) Écrire C_2 en fonction de P_1, P_2 ; C_3 en fonction de P_3, P_4 ; C_1 en fonction de P_4, P_5, P_2 et P_6 ; C_4 en fonction de P_1, P_5, P_3 et P_7.

2) Combien d'additions et multiplications de matrices $n/2 \times n/2$ sont utilisées ?

3) En déduire la complexité (en terme d'opérations dans K) de l'algorithme récursif associé, qui calcule $C = AB$. (C'est l'algorithme de Strassen).
Quelle est la complexité de l'algorithme naïf de multiplication de matrices ?

4) Implémenter l'algorithme de Strassen, et le tester sur des matrices aléatoires de dimension 16 dans $\mathbb{F}_{13}$.