Université de Bordeaux **Algorithmique matricielle**
Mathématiques 2024–2025

FEUILLE D'EXERCICES n° 4

Sage et Algèbre linéaire - Travail sur machine

1. Introduction

Sage est un logiciel libre de calcul, accessible sur http ://www.sagemath.org. Il utilise le langage python. Il existe au moins trois façons de l'utiliser.

- En utilisant une interface graphique (le notebook ou l'accès internet).
- Par la ligne de commande interactive, en appelant éventuellement des programmes écrits sur un éditeur de texte. Pour cela, il suffit de taper la commande `sage` sur une fenêtre terminal.
- En écrivant des scripts python indépendants qui font appel à la bibliothèque `sage`.

Vous pourrez utiliser au choix la ligne de commande interactive sur une fenêtre terminal ou bien le notebook.

2. Quelques commandes utiles

- Pour ouvrir `sage` : `sage` sur une fenêtre terminal.
- Pour ouvrir le notebook sur jupyter : sur une fenêtre terminal, taper `sage -n jupyter`. Une fenêtre `jupyter` s'ouvre alors sur votre navigateur web. Il est possible sur cette fenêtre de naviguer dans vos répertoire. Pour ouvrir une feuille de travail sage, cliquer en haut à droite sur `nouveau` puis `sagemath 9.5`.
- Pour quitter `sage` (sur terminal) : `ctrl d`
- Pour interrompre un calcul (sur terminal) : `ctrl c`
- Dernier résultat : `_`
- Complétion : `Tab`
- Commentaire : `#` pour une ligne, `"'` blablabla `"'` pour un bloc.
- Informations sur une fonction : `Nom_de_fonction?` (par exemple `factor?`). Pour quitter l'aide ainsi activée, taper `q`.

3. listes

Exécuter la commande

```
l=[2,-6,9,3];l
```

Alors `l` est une liste. Essayer les commandes

```
l[0]
sum(l)
max(l)
```

Essayer aussi

```
range(0,5)
[0..5]
```

Ce sont deux autres exemples de listes. On peut aussi définir des listes par des formules

```
m=[i^2 for i in range(0,5)];m
```

On peut même ajouter des conditions

```
m=[i^2 for i in range(0,5) if i!=2];m
```

Pour concaténer `l` et `m`, on peut écrire

```
l+m
```

Essayer aussi

```
[sqrt(a) for a in m]
```

4. Déclaration des matrices

Dans ce paragraphe, nous allons explorer différentes façons pour définir les matrices et vecteurs sur sage.

Après chaque ligne, valider en tapant `Entrée`.

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

```
w = vector([1,1,-4])
w
A = matrix([[-1,2,3],[3,2,1],[1,1,1]]); A
```

Remarquons que les indices commencent à 0. Si l'on tape `A[0,0]`, on obtient `-1`. On peut modifier les coefficients :

```
A[0,0]=1
A
```

Remarquons aussi que `A` et `w` sont définies en utilisant des listes.

2) Exécuter les commandes

```
A.det()
A*w
w*A
parent(A)
parent(w)
```

"Free module", ça veut dire "module libre". Nous ne travaillerons pas sur les modules. Ils sont définis comme les espaces vectoriels, sauf qu'un espace vectoriel est défini sur un corps et un module sur un anneau (ici : $\mathbb{Z}$).

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 tant 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])
parent(B)
```

```
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)
```

ou

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

5) On peut avoir besoin de définir une matrice par une formule sur les indices, par exemple :

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

ou bien, pour une matrice non carrée

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

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

6) Soit la fonction f définie sur $\mathbb{Z}^2$ comme suit : $f(i,j) = i^2$ si $i \geqslant j$ et $f(i,j) = j - i$ si $i < j$. On décrit ci-dessous comment on peut programmer f.

```
def f(i,j):
    if i>=j:
        return i^2
    else:
        return j-i
```

Attention : les indentations et les " : " sont importants.

Définir la matrice de $M_{5,5}(\mathbb{Q})$ dont le coefficient (i,j) est égal à $f(i,j)$.

7) Pour les vecteurs, la commande

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

ne fonctionne pas. Pour pallier à cela, on peut utiliser une liste.

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

(`range(3)`, c'est la même chose que `range(0,3)`)

8) Pour entrer des matrices, on peut aussi commencer par définir l'espace des matrices ou des vecteurs

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

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

```
M3Q = MatrixSpace(QQ,3); M3Q
B = M3Q([1,2,3,3,2,1,1,1,1]);B
parent(B)
```

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)
```

5. Éléments pris au hasard

Soit un ensemble noté `A` dans sage. On peut choisir un élément au hasard dans A par la commande `A.random_element()`. Essayer par exemple

```
ZZ.random_element()
QQ.random_element()
```

On peut aussi préciser des bornes pour le choix de ces éléments.

```
ZZ.random_element(100)
QQ.random_element(num_bound=100,den_bound=100)
M3Q.random_element(num_bound=100,den_bound=100)
```

6. Addition, multiplication, déterminant, inverse

Tout cela se fait de façon naturelle.

```
M = M3Q.random_element(num_bound=10,den_bound=10);M
N = M3Q.random_element(num_bound=10,den_bound=10);N
M + N
M * N
M.determinant()
M.det()
M^2
Minv = M^(-1)
M*Minv
```

7. Extraction, matrices par blocs

Essayons sur la matrice

```
M = matrix(QQ,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(_)
```

(la commande « `_` » 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()
M.dimensions()
```

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` en choisissant une borne pour la valeur absolue des numérateur et dénominateur des coefficients de N.

Pour définir des matrices par blocs, il existe une fonction

```
block_matrix
```

Construire la matrice L définie par blocs par $L = \begin{pmatrix} M & N \\ N & I_3 \end{pmatrix}$

4) Par la commande `transpose`, transposer M.

8. Réduction

1) Soit $A = \left(\begin{smallmatrix} 1 & 2 & 3 \\ 3 & 2 & 1 \\ 1 & 1 & 1 \end{smallmatrix}\right) \in M_3(\mathbb{Q})$. Appliquer à A les fonctions

```
characteristic_polynomial
minimal_polynomial
eigenvalues
eigenvectors_right
```

Vérifier que les vecteurs donnés par cette commande sont bien des vecteurs propres pour les valeurs propres correspondantes. Pour cela, on peut utiliser ce genre de commandes :

```
VectPropres=eigenvectors_right()
A*VectPropres[0][1][0]-VectPropres[0][0]*VectPropres[0][1][0]
```

En effet, `VectPropres` est une liste. `VectPropres[0]` correspond à l'une des valeurs propres `VectPropres[0][0]` est la valeur propre et `VectPropres[0][1]` est une base de l'espace propre correspondant. `VectPropres[0][2]` est sa dimension. Ici, les espaces propres sont de dimension 1.

```
eigenspaces_right
eigenmatrix_right
```

Cette dernière commande rend un couple de matrices $[D, P]$. Vérifier que $P^{-1}AP = D$.

2) Même exercice avec $B = \left(\begin{smallmatrix} 0 & 2 \\ 1 & 0 \end{smallmatrix}\right)$. Avec cette matrice B, on utilise des approximation car ses valeurs propres sont $\pm\sqrt{2}$.

On peut aussi manier formellement les valeurs exactes, mais nous ne le ferons pas ici.

3) Vérifier sur des matrices M et N prises au hasard dans $M_5(\mathbb{Q})$ que MN et NM ont même polynôme caractéristique.

9. Gershgorin-Hadamard

1) Écrire une fonction qui pour tout $M \in M_n(\mathbb{C})$ calcule $\rho(M)$.

2) Que calcule la fonction suivante ?

```
def majoration(M):
    n=M.nrows()
    l=[]
    for i in range(n):
        s=sum([abs(M[i,j]) for j in range(n)])
        l=l+[s]
    return max(l)
```

Vérifier sur des matrices prises au hasard dans $M_4(\mathbb{Q})$ que $\rho(M) \leqslant \max_i \sum_j |M_{i,j}|$.

3) On verra que cette expression $\max_i \sum_j |M_{i,j}|$ définit une norme sur $M_n(\mathbb{C})$, que l'on appelle $|||M|||_\infty$. Sur sage, on peut calculer cette norme en utilisant la commande `M.norm(infinity)` ou bien `M.norm(oo)`. Vérifier sur un exemple que cette norme correspond bien à la fonction `majoration`.

4) Pour les calculs d'approximation de valeurs complexes, on peut utiliser `CC` (le corps des complexes) ou bien `CDF` (le même en "double précision"). Soient

```
A=matrix(CDF,3,[10*I,3,1,-I,I-1,-3,0,2,8])
```

La liste des centres des cercles de Gershgorin de A est $[(0,10),(-1,1),(8,0)]$. La liste de leurs rayons est $[4,4,2]$.

a) Écrire une fonction `Gershgorin` qui étant donnée une matrice M calcule ces deux listes. Pour cela, on peu utiliser `real` (partie réelle) et `imag`.

b) Soit

```
VPA=A.eigenvalues()
```

On peut inscrire dans un graphe les points du plan complexe qui correspondent à ces valeurs propres :

```
point(VP,size=50,color='red')
```

On peut aussi donner un nom à ce graphe pour pouvoir l'inscrire ensuite dans un autre graphe.

```
Spectre=point(VP,size=50,color='red')
```

c) Le cercle de centre $(0,10)$ et de rayon 4 s'écrit

```
circle((0,10),4)
```

Définir la liste `lC` des cercles de Gershgorin de A. Alors on peut mettre dans le même graphe la liste de ces cercles et les valeurs propres de A par la commande

```
sum(lC)+Spectre
```

5) Écrire une fonction `ConfigurationGH` qui étant donnée une matrice A dessine les valeurs propres de A et ses cercles de Gershgorin.

6) Appliquer cette fonction aux matrices à coefficients dans $\mathbb{C}$ suivantes.

$$\begin{pmatrix} 1 & 0 & 3 \\ 2 & 3 & 1 \\ 1 & 0 & -2 \end{pmatrix} \qquad \begin{pmatrix} 1 & 2 & 3 \\ 3 & 2 & 1 \\ 1 & 1 & 1 \end{pmatrix} \qquad \begin{pmatrix} 2 & 1 & -1 \\ 0 & 1 & 1 \\ 2 & -1 & -3 \end{pmatrix}$$

$$\begin{pmatrix} 3+i & -1,5 & 0 & 1,5i \\ 0,5 & 4 & i & 0,5i \\ \sqrt{2} & \sqrt{2}i & 2+3i & 0 \\ i & 1 & i & 4i \end{pmatrix}$$

10. Décomposition LU

1) Soit

$$A = \begin{pmatrix} -2 & 3 & -2 & -5 \\ 1 & -2 & 1 & 3 \\ -4 & 7 & -3 & -8 \\ -3 & 8 & -1 & -5 \end{pmatrix} \in M_4(\mathbb{Q}).$$

Appliquer la commande `LU` à cette matrice. Vérifier que l'on obtient des matrices P, L et U telles que $A = PLU$.

2) Essayer la commande `A.LU(pivot='nonzero')`, et vérifier le résultat obtenu.

3) Essayer aussi l'option `format='compact'`.

4) Écrire une fonction qui à partir du format compact de `LU` donne les matrices P, L et U in extenso.

5) Définir L'espace de matrices $M_{10}(\mathbb{Q})$. Choisir au hasard une matrice M de $M_{10}(\mathbb{Q})$ dont les coefficients sont des entiers inférieurs à 100 en valeur absolue. Pour cela, on peut utiliser la commande `random(num_bound=100,den_bound=1)`

6) Appliquer à M la fonction `LU`, directement d'abord, puis en utilisant des approximations numériques (utiliser `RDF` pour $\mathbb{R}$).