MI 101 Maple
Algèbre linéaire 2
Feuille de TD numéro 9 - jeudi 13 décembre

Table des matières

1 Infos diverses

- Petit rappel : il y a un QCM (5 minutes, 5%) jeudi prochain, qui portera jusqu'à cette semaine.
- La publication des notes sur le web n'est toujours pas assurée.
- La séance de cette semaine est un peu légère, avec beaucoup de petits détails, l'occasion de bien comprendre l'algèbre linéaire sous Maple. Nous ferons les suites et les limites la semaine 10, puis les commandes graphiques (2D, 3D, animations, paramétrique) et l'étude de fonctions la semaine 11 (après les vacances) et il nous restera une séance libre pour réviser et expliquer tout ce qui peut être obscur, ou essayer de résoudre un problème concret, selon vos souhaits.

2 Rappels et compléments

2.1 Multiplication des matrices

Lorsque Maple rencontre le signe `*`, il considère qu'il s'agit d'une multiplication classique, et simplifie les calculs avant de considérer ce qu'il multiplie, ce qui pose problème lorsqu'il s'agit de matrices, dont le produit n'est pas commutatif. Par exemple, il remplacera `A*B*A` par le produit de B et de A^2, ce qui est correct pour des scalaires, mais faux pour les matrices. Il existe deux solutions :

- On peut utiliser `multiply()` pour efffectuer des produits matriciels. Par exemple, on tapera `M:=multiply(A,B,A,C+D,inverse(B))` pour calculer $M := A.B.A.(C+D).B^{-1}$.
- On peut utiliser l'opérateur `&*`, peu clair il est vrai, mais qui indique à Maple que le produit n'est pas commutatif. Le résultat détaillé du produit est ensuite obtenu par `evalm()`. L'exemple précédent donne alors `M:=A&*B&*A&*(C+D)&*inverse(B)`. De la même manière, on utilisera `&**` pour les puissances de matrices carrées.

2.2 Extraction des éléments des vecteurs et matrices

Lorsqu'un vecteur `V` et une matrice `M` sont définis, on peut ontenir un élément du vecteur ou de la matrice sans l'afficher entièrement, en indiquant la position de l'élément à afficher entre crochets par `V[3];` ou `M[5,2];`. On peut de la même manière modifier un seul élément du vecteur ou de la matrice, sans avoir besoin de tour redéfinir, avec par exemple `V[3]:=7;` ou `M[5,2]:=a;`.

2.3 Extraction les lignes et colonnes des matrices

Si `M` est une matrice, `row(N,n)` et `col(N,n)` retournent respectivement le `n`-ième vecteur ligne ou le `n`-ième vecteur colonne de `M`, et `row(N,n1..n2)` et `col(N,n1..2)` retournent les séquences des vecteurs lignes ou colonnes dont les indices vont de `n1` à `n2`.

2.4 Matrices diagonales

Une matrice remplie de 0 à l'exception de 1, 9 et 5 sur la diagonale est donnée par `diag(1,9,5);`. Ainsi, une matrice identité de taille 7 sera donnée par `diag(seq(1,k=1..7))`.

3 Espaces vectoriels, bases

- La commande `basis()` permet d'obtenir une base d'un espace vectoriel. `basis(`*`vects`*`)` retourne une liste ou un ensemble de vecteurs constituant une base de l'espace vectoriel engendré par *`vects`*, selon que *`vects`* est une liste ou un ensemble de vecteurs. On obtient ensuite la dimension de l'espace vectoriel en comptant (par `nops()`) le nombre de vecteurs de la base obtenue.

- On peut considérer qu'un espace vectoriel est défini par la donnée de l'une de ses bases. Si *base1*, *base2*,..., *basen*, sont des bases (ie des listes ou ensembles de vecteurs) de sous-espaces vectoriels, `intbasis(`*`base1,base2,...,basen`*`)` retourne une base de l'intersection de ces sous-espaces, et `sumbasis(`*`base1,base2,...,basen`*`)` une base de la somme des sous-espaces.

4 Noyau et image

- On obtient une base du noyau de la matrice `M` par `kernel(M)`.

- On obtient une base de l'espace vectoriel engendré par les lignes de la matrice `M` par `rowspace(M)`.

- On obtient une base de l'espace vectoriel engendré par les colonnes de la matrice `M` par `colspace(M)`.

  Remarque : On peut aussi utiliser `kernel(M,'d')` qui, en plus de retourner une base du noyau, affecte à `d` la dimension de celui-ci, tout comme `rowspace(M,d)` et `colspace(M,d)` (il n'y a pas ici de guillemets).

Ainsi, l'image d'une matrice étant l'espace vectoriel engendré par ses colonnes, celle-ci peut-être obtenue par `colspace()`. Sa dimension est le nombre maximal de vecteurs colonnes que 'contient' la matrice, c'est-à dire son rang, qu'on obtinient par `rank(M)`.

5 Exercices

Exo 1
Afin de vérifier et pratiquer l'usage de `&*`*, définissez deux matrices carrées de même taille* `A` *et* `B`*, puis calculez* `A*B-B*A` *et* `A&*B-B&*A`*. Essayez également avec* `multiply()`*.*

Exo 2
Définissez la matrice $A := \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 4 \end{bmatrix}$*, puis, à l'aide de row ou col, extrayez-en les lignes ou colonnes et permutez la première et la troisième, et la seconde et la quatrième.*

Exo 3
On considère les sous-espaces vectoriels $E1$ *et* $E2$ *de* R^4*, tels que* $E1$ *est engendré par* $\begin{bmatrix} 3 & 1 & -2 & 1 \end{bmatrix}$ *et* $\begin{bmatrix} -6 & 2 & 4 & -1 \end{bmatrix}$*, et* $E2$ *par* $\begin{bmatrix} 4 & 2 & -1 & -5 \end{bmatrix}$ *et* $\begin{bmatrix} -3 & -4 & 2 & 7 \end{bmatrix}$*. On note* ES *leur somme et* EI *leur intersection. Vérifiez que dim(*ES*)=dim(*$E1$*)+dim(*$E2$*)-dim(*EI*).*

Une famille de n vecteurs est libre ssi l'espace vectoriel qu'ils engendrent est de dimension n, ie ssi toute base de cet espace vectoriel contient n vecteurs. De même, une famille de n vecteurs est liée ssi l'espace vectoriel qu'ils engendrent est de dimension inférieure à n, ie ssi toute base de cet espace vectoriel contient moins de n vecteurs. Enfin, une famille de vecteurs de dimension m (vecteurs à m éléments) est génératrice ssi elle engendre un espace vectoriel de dimension m. Connaissant cela, dites si les familles proposées ci-après sont libres, liées, génératrices.

Exo 4
$\begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix}, \begin{bmatrix} 2 & 1 & 0 & 0 \end{bmatrix}, \begin{bmatrix} 3 & 2 & 1 & 0 \end{bmatrix}, \begin{bmatrix} 4 & 3 & 2 & 1 \end{bmatrix}$

Exo 5
$\begin{bmatrix} 3 & 4 & 1 \end{bmatrix}, \begin{bmatrix} -1 & 2 & 0 \end{bmatrix}, \begin{bmatrix} 5 & 0 & 1 \end{bmatrix}$

Exo 6
$\begin{bmatrix} 2 & 1 \end{bmatrix}, \begin{bmatrix} -7 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 0 \end{bmatrix}, \begin{bmatrix} 1 & 1 \end{bmatrix}$

On peut considérer une matrice carrée A de taille n comme une application de R^n dans R^n; en effet, tout élément X de R^n (vecteur réel de taille n), multiplié à gauche par A, donne $Y = A.X$, où Y est un autre élément de R^n. Y est ainsi en quelque sorte l'image de X par l'application f_A associée à A. On a alors le résultat suivant :

f_A est une bijection
ssi le noyau de A est $\{0\}$ (de dimension 0)
ssi l'image de A est R^n (de dimension n)
ssi A est de rang n
ssi le déterminant de A est non nul.

En utilisant ces quatre outils, dites si les matrices M ci-après sont associées à des bijections.

Exo 1
$M1 := \begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 4 & 8 & 16 \\ 3 & 9 & 27 & 81 \\ 4 & 16 & 64 & 264 \end{bmatrix}$

Exo 2
$M2 := \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix}$, $M3 := M2 + I_3$, $M2 := \begin{bmatrix} 0 & 0 & 1 \\ 0 & t & 0 \\ 1 & 0 & 0 \end{bmatrix}$ *(discutez selon la valeur de t)*

Exo 3
$M5 := \begin{bmatrix} cos(t) & sin(t) \\ -sin(t) & cos(t) \end{bmatrix}$, $M6 := M4 + 5.I_2$

6 Correction

Exo 4
Par exemple,

```
A:=matrix(2,2,[1,3,6,2]); A:=matrix(2,2,[3,7,2,4]);
A*B-B*A; evalm(A&*B-B&*A); evalm(multiply(A,B)-multiply(B,A));
```

Exo 5

```
A=diag(1,2,3,4);
L1:=col(A,1); L2:=col(A,2); L3:=col(A,3); L4:=col(A,4);
B:=matrix([L3,L4,L1,L2]);
```

Exo 6

```
base1:=[vector([3,1,-2,1]),vector(-6,2,4,-1)];
base2:=[vector([4,2,-1,-5]),vector( -3,-4,2,7)];
D1:=nops(basis(base1)); D2:=nops(basis(base2));
DS:=nops(sumbasis(base1,base2)); DI:=nops(intbasis(base1,base2));
```

Exo 7
Pour simplifier, on fait une procédure polyvalente :

```
info:=proc(F)
local
df:=nops(F);
de:=nops(basis(F));
dim:=vectdim(op(1,F));
if de=df then print(``libre''); else print(``liée''); fi;
if de=dim then print(``génératrice''); else print(``non génératrice''); fi;
end;
```

Puis on définit de manière classique les vecteurs `V1`, `V2`, `V3`, `V4`, *et on appelle*

```
info([V1,V2,V3,V4]);
```

Exo 8
idem.

Exo 9
idem.

Exo 7

```
M1:=...;
nops(kernel(M1)); nops(colspace(M1)); rank(M1); det(M1);
```

Exo 8

```
M2:=...;
M3:=...;
M4:=...;
P:=det(M4); solve(P,t);
```

Exo 9
idem.