Benjamin COLLAS
benjamin.collas@telaxo.com

Mi-101 **Mathematica** 2004-2005

Leçon 6 : Algèbre linéaire

Objectifs :

- se familiariser avec les notions de familles libres et génératrices
- étudier les propriétés de la $\mathbb{R}$-algèbre associative et unifère des matrices
- trouver une base et un système d'équation des noyau et image d'une application linéaire
- résoudre un système linéaire avec paramètres

Famille libre et génératrice

Afin de vérifier si une famille est libre ou génératrice d'un espace vectoriel, on utilise la commande *Solve* qui permet de résoudre des équations. Dans notre cas, les variables seront les coefficients scalaires utilisé pour les combinaisons linéaires.

■ Famille libre

Considérons les vecteurs u et v dans un espace vectoriel de dimension supérieure ou égale à 2.
Il suffit de vérifier que l'équation $a * u + b * v == 0$ d'inconnues a et b n'admette que la solution $(0,0)$.

```
u = {1, -1};
v = {-2, 3};

Solve[a * u + b * v == 0, {a, b}]
```

■ Exercice

Vérifier que les vecteurs $u=\{1,-1,-1\}$, $v=\{1,0,0\}$ et $w=\{0,1,1\}$ ne forment pas une famille libre.
Vérifier que les vecteurs $u=\{1,0,1,-1\}$, $v=\{1,0,1,0\}$ et $w=\{0,1,1,2\}$ forment une famille libre.

■ Famille génératrice

Considérons pour notre exemple les vecteurs u et v dans un espace vectoriel de dimension 2.
Vérifier que la famille $\{u,v\}$ est génératrice, c'est vérifier que quelque soit le vecteur $\{x,y\}$, l'équation d'inconnues a et b, a * u + b * v == {x,y} admette une solution dépendant de x et de y.


```
u = {1, 2};
v = {2, 3};

Solve[a * u + b * v == {x, y}, {a, b}]
```

■ Exercice

Vérifier que les vecteurs $u=\{1,-1,1\}$, $v=\{1,0,0\}$ et $w=\{0,1,1\}$ forment une famille génératrice de $\mathbb{R}^3$.
Vérifier que les vecteurs u et v de la section précédente forment une famille génératrice de $\mathbb{R}^2$.

Les matrices

Sous *Mathematica*, les matrices sont représentées sous la forme de liste de listes de longueur identique : chacune de ces sous-listes représente une ligne d'une matrice.
De manière générale, on peut accèder à un élément, à une ligne ou une colonne d'une matrice M=(m_{ij}) en utilisant le double crochet *[[]]*.
Ainsi *M[[2]]* fournit la deuxième ligne, *M[[2,1]]* fournit l'élément m_{21}, *ie* le premier élément de la seconde ligne. Pour obtenir une colonne, on passe par la récupération d'une ligne de la matrice transposée *via Transpose*.

Astuce : lors de calculs matriciels, il est vivement conseillé de faire suivre la ligne de calcul par un point virgule, puis par l'expression *%//MatrixForm* qui permet la lecture du résultat sous forme traditionnelle.

■ Définitions

```
M = {{1, 2, 3}, {0, 5, 3}, {3, 2, 1}};
% // MatrixForm
```

$$\begin{pmatrix} 1 & 2 & 3 \\ 0 & 5 & 3 \\ 3 & 2 & 1 \end{pmatrix}$$

■ Exercice :

Récupérer l'élément m_{23} de la matrice M ci-dessus.
Ecrire un vecteur ligne et un vecteur colonne (vérifiez votre syntaxe avec *MatrixForm*).
NB : dans la pratique, *Mathematica* accepte de considérer {x,y,z} comme un vecteur colonne, même si *MatrixForm* ne le représente pas comme tel.

■ Opérations

Nous avons déja vu que l'addition et la multiplication entre listes de même dimension effectuaient ces opérations élément par élément.

Ainsi, la somme entre deux matrices *M* et *N* sera bien tapée *M+N*, tandis que la multiplication emploie le nouveau symbole "."

Attention à l'emploi de *M^n* pour élever une matrice à la puissance *n*. Cette commande permet d'élever chacun des éléments de la matrice à la puissance *n* mais n'est en rien équivalent à *n* multiplications matricielles. Pour ce faire on emploiera la commande *MatrixPower*.

```
NN = {{3, 2, 1}, {5, 6, -2}, {1, -3, 0}};

M.NN // MatrixForm
```

$$\begin{pmatrix} 16 & 5 & -3 \\ 28 & 21 & -10 \\ 20 & 15 & -1 \end{pmatrix}$$

■ Exercice :

En utilisant les matrices *M* et *NN* définies ci-dessus, vérifiez la non-commutativité de la multiplication matricielle (on pourra utiliser l'opérateur booléen "==" ou "!=").

Vérifiez sur ces matrices la différence entre *M^2* et *MatrixPower[M,2]*

En calculant quelques puissances de la matrice suivante, vérifiez que l'algèbre des matrices n'est pas un corps:

$$\begin{pmatrix} 0 & 1 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix}$$

■ Inversion

Si une matrice représente une application linéaire bijective, on peut alors calculer la matrice inverse (ie l'application linéaire réciproque) :

```
Det[M]
```

```
-28
```

```
Inverse[M];
% // MatrixForm
```

$$\begin{pmatrix} \frac{1}{28} & -\frac{1}{7} & \frac{9}{28} \\ -\frac{9}{28} & \frac{2}{7} & \frac{3}{28} \\ \frac{15}{28} & -\frac{1}{7} & -\frac{5}{28} \end{pmatrix}$$

■ Exercice :

Comparez *Inverse* et *MatrixPower[. ,-1]*

Ecrire une fonction qui, lorsque c'est possible, à une matrice associe son inverse présentée avec *MatrixForm*.

■ Exercices sur les matrices :

■ Matrices symétriques et antisymétriques

Une matrice symétrique est une matrice égale à sa transposée, une matrice antisymétrique est une matrice égale à l'opposée de sa transposée.

Vérifiez que *M0* et *M1* sont respectivement des matrices symétrique et antisymétrique:

```
M0 = (Transpose[M] + M) / 2;
M1 = (M - Transpose[M]) / 2;
```

Vérifiez que M s'exprime comme la somme des ces deux matrices.
Faire le lien avec la décomposition en parties réelle et imaginaires des nombres complexes.

■ Création de matrice

En utilisant *Table,* créez les matrices $A=(a_{ij})$ de dimension 3x3 telles que $(a_{ij})=f(i,j)$, où f est une des fonctions définies ci-dessous:

```
f1[i_, j_] := i / j + j / i

f2[i_, j_] := i * j

f3[i_, j_] := If[i < j, 0, 1]
```

Déduire de la dernière fonction un moyen d'obtenir une matrice triangulaire supérieure, supérieure stricte, et diagonales.

■ Matrice de rotation d'angle t

Considérons la matrice de rotation suivante :

```
Mr = {{Cos[t], -Sin[t]}, {Sin[t], Cos[t]}};
% // MatrixForm
```

$$\begin{pmatrix} \text{Cos}[t] & -\text{Sin}[t] \\ \text{Sin}[t] & \text{Cos}[t] \end{pmatrix}$$

1) Explicitez l'image par la rotation d'angle t d'un vecteur *(x,y)*.
En déduire le vecteur image par la rotation d'angle $\frac{\pi}{4}$ du vecteur *(1,1)* (on pourra utiliser les règles de remplacement).
2) Calculez les images successives par la rotation d'angle $\frac{\pi}{6}$ du point de coordonnées *(1,2).*
Affichez ces points.

■ Matrice et endomorphisme

Soit $\mathbb{R}_3[x]$ l'espace vectoriel des polynômes à coefficients réels de degrés inférieurs ou égaux à *3.*
Soit l'endomorphisme $u: x \to 1+x$, de matrice *P.*

```
P = Transpose[Table[Binomial[n - 1, p - 1], {n, 1, 4}, {p, 1, 4}]];
% // MatrixForm
```

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

```
Inverse[P];
% // MatrixForm
```

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

Justifiez la forme de la matrice *P* associée à l'endomorphisme *u*.
Retrouvez l'inverse de la matrice en partant de la définition de l'endomorphisme.

Kernel et Image

La caractérisation d'une application linéaire passe par l'étude de son noyau et de son image.

Pour ce faire, on utilise dans la pratique une résolution de système linéaire par la méthode de Gauss.
La résolution du système fournit un second système d'équations (les équations du noyau ou de l'image), qui permet ensuite d'obtenir des vecteurs générateurs et indépendants (une base du noyau ou de l'image).

Sous *Mathematica*, chacune de ces deux types de représentation base/équation est associé à deux méthodes différentes : on peut utiliser des commandes propres à *Mathematica* ou des méthodes proches de celles de la résolution d'équations linéaires.

■ Déterminer une base du noyau

La commande *NullSpace* permet d'obtenir une base du Kernel d'une application linéaire exprimée sous forme de matrice :

```
MKer = {{1, 0, 2}, {-2, 0, -4}, {1, 5, -3}};
% // MatrixForm
```

$$\begin{pmatrix} 1 & 0 & 2 \\ -2 & 0 & -4 \\ 1 & 5 & -3 \end{pmatrix}$$

```
baseKerM = NullSpace[MKer];
% // MatrixForm
```

$$\begin{pmatrix} -2 & 1 & 1 \end{pmatrix}$$

Vérifions :
Attention, le résultat de *NullSpace* est une ligne et non pas un vecteur colonne. Afin de pouvoir effectuer la multiplication matricielle avec *M*, il faut transposer le vecteur fourni.


```
MKer.Transpose[NullSpace[MKer]];
% // MatrixForm
```

$$\begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

■ Exercice :

Calculez une base du noyau de la matrice suivante:

$$\begin{pmatrix} -10 & 4 & 2 \\ 20 & -8 & -4 \\ -5 & 2 & 1 \end{pmatrix}$$

En utilisant *Length*, écrire une fontion *dimKer* qui à une matrice associe la dimension du noyau.

■ Déterminer le rang

En dimension finie le théorème de la dimension nous permet de déterminer le rang d'une matrice (*ie* la dimension de l'image de l'application linéaire correspondante) connaissant la dimension du noyau :

> ***Théorème de la dimension :***
> Soient E, F deux espace vectoriels de dimension finie. Si l'on considère une application linéaire *u* de E dans F, alors :
> dim(E)=dim (Im u)+dim(Ker u)

On utilise ainsi la commande *Dimensions* qui renvoie le nombre de lignes et le nombre de colonnes d'une matrice :

```
Mrang = {{1, 2, 3}, {1, 2, 3}};
% // MatrixForm
```

$$\begin{pmatrix} 1 & 2 & 3 \\ 1 & 2 & 3 \end{pmatrix}$$

```
Dimensions[Mrang]
```

```
{2, 3}
```

■ Exercice:

Ecrire une fonction *rang* qui à une matrice associe son rang (on utilisera *NullSpace* et *Dimensions*).

■ Equations du noyau et de l'image

Trouver des équations du noyau et des bases, c'est trouver un certain nombre de relations linéaires entre les coordonnées d'un vecteur *(x,y,z)* qui appartient à l'ensemble que l'on veut caractériser.
On utilise pour cela deux commandes : *Reduce* qui permet de résoudre des équations, et *Eliminate* qui élimine des variables et permet par extension de trouver des relations entre les autres.

■ Le Noyau :

On utilise *Reduce* qui permet de résoudre un sytème d'équations.
Attention : il existe plusieurs commandes *Mathematica* qui permettent la résolution de sytèmes d'équations, mais *Reduce* permet d'obtenir toutes les solutions, et cela y compris dans le cas où les équations font intervenir des paramètres.

Reprenons la matrice *MKer* : si *{x,y,z}* est dans le noyau il vérifie l'équation *M.{x,y,z}==0*, *ie* :

```
Reduce[MKer.{x, y, z} == 0, {x, y, z}]
```

```
x == -2 z && y == z
```

Exercice :

Trouver un système d'équations du noyau de la matrice *Mexemple* suivante :

$$\begin{pmatrix} -24 & -9 & 6 \\ -8 & -3 & 2 \\ -16 & -6 & 4 \end{pmatrix}$$

■ Image :

On utilise *Eliminate* qui permet de trouver une relation entre les coordonnées du vecteur appartenant à la base.

```
Mimg = {{2, 4, -6}, {-4, -8, 12}, {-3, -6, 9}};
Mimg // MatrixForm
```

$$\begin{pmatrix} 2 & 4 & -6 \\ -4 & -8 & 12 \\ -3 & -6 & 9 \end{pmatrix}$$

Ainsi, si un élément *(x,y,z)* appartient à l'image de *Mimg*, c'est qu'il existe un vecteur *(a,b,c)* tel que *M.(a,b,c) =(x,y,z)*, *ie* en *Mathematica* :

```
Eliminate[Mimg.{a, b, c} == {x, y, z}, {a, b, c}]
```

```
3 x == -2 z && 3 y == 4 z
```

■ Lien entre les deux méthodes de représentation :

Lorsque l'on connait une base de l'espace considéré, on peut aussi s'appuyer sur la méthode manuelle qui permet de passer à un système d'équations correspondant.
Reprenons l'exemple de *MKer*, dont on connait une base et déduisons en un système d'équations :

```
NullSpace[MKer]
```

```
{{-2, 1, 1}}
```

Les éléments appartenant à cet espace vectoriel sont les éléments qui sont combinaisons linéaires des vecteurs de base :


```
Apply[Plus, NullSpace[MKer] * {a}]
```

```
{-2 a, a, a}
```

Grâce à *Eliminate*, on passe au système d'équations:

```
Eliminate[Apply[Plus, NullSpace[MKer] * {a}] == {x, y, z}, {a}]
```

```
x == -2 y && y == z
```

Exercice :

Refaire cet exercice en utilisant la matrice *Mexemple*.
On utilisera *Reduce* afin de déterminer si les deux résultats sont équivalents (quelle aurait été la réponse de *Reduce* si cela n'avait pas été le cas ?)

■ Résolution d'équations avec *Reduce*

■ Exercices préliminaires

Exercice :
Résoudre dans $\mathbb{R}$ l'équation suivante d'inconnue x : $a*x+b=0$

Exercice :
Résoudre dans $\mathbb{C}$ l'équation suivante d'inconnue x : $a*x^2+b*x+c=0$
Mathematica retourne-t-il toutes les solutions complexes ?

■ Retour aux systèmes linéaires

On considère la matrice suivante :

```
Mexo = {{a, 0, 0, 0, b}, {0, a, 0, b, 0}, {0, 1, 1, 1, 0}, {0, b, 0, a, 0}, {b, 0, 0, 0, a}};
% // MatrixForm
```

$$\begin{pmatrix} a & 0 & 0 & 0 & b \\ 0 & a & 0 & b & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & b & 0 & a & 0 \\ b & 0 & 0 & 0 & a \end{pmatrix}$$

Résoudre le système suivant en x, y, z, u, v (on discutera des solutions en fonction des conditions booléennes retournées :

$$\begin{pmatrix} b\,v+a\,x \\ b\,u+a\,y \\ u+y+z \\ a\,u+b\,y \\ a\,v+b\,x \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ a \\ 0 \\ 0 \end{pmatrix}$$