TP : L'examen de math3 avec Sagemath

Algèbre linéaire et matricielle

mars 2017

f-x. dehon (dehon@unice.fr) avec l'aide de Cocalc - Jupiter notebook

Ex 1. On pose $u = (1, 0, -2, -1)$, $v = (1, 6, 4, -1)$, $w = (-2, -3, 1, 2)$ et $F = \mathrm{vect}(u, v, w)$. Déterminer une base et une équation de F.

Solution.

1ère approche : Famille de vecteurs

On observe :

(1) une famille (u_i) de vecteurs est libre si pour chaque k le vecteur u_k n'est pas combinaison linéaire des vecteurs u_i, $i < k$ (avec la convention que 0 est combinaison linéaire de la famille vide de vecteur).

(2) une famille (u_i) de vecteurs est génératrice de F si chacun des vecteurs u, v, w est combinaison linéaire des u_i.

(3) si on dispose d'une famille génératrice (f_j) d'un espace E et d'une famille libre (e_i) de vecteurs de E (éventuellement vide), on peut compléter (e_i) en une base de E en ajoutant succéssivement les vecteurs f_j pourvu qu'ils ne soient pas combinaison linéaire de la famille jusque là formée.

(4) une fois choisie une base adaptée à F (base de F complétée en une base de $\mathbb{R}^4$), une équation de F est donnée par l'annulation des coordonnées relativement aux vecteurs de la base qui ne sont pas dans F.

Pour mettre en oeuvre cette approche dans Sagemath il faut disposer ou créer un test qui dit si un vecteur x donné est combinaison linéaire d'une famille (u_i) de vecteurs donnée . Si un tel test existe, il indiquera comment les vecteurs doivent être encodés (par une liste ? un objet de type vecteur ? une matrice ligne ou une matrice colonne ?). Si un tel test est à construire, il faudra décider d'une façon d'encoder les vecteurs.

Méthodes disponibles dans Sagemath : lire Sage Linear Algebra Quick Reference sur https://wiki.sagemath.org/quickref (https://wiki.sagemath.org/quickref), chercher des tutoriels Sagemath algèbre linéaire. Le type vecteur et espace vectoriel engendré par une liste de vecteurs existent mais les méthodes disponibles sont essentiellement matricielles ; voir par exemple https://ask.sagemath.org/question/39481/checking-if-a-subset-of-columns-of-a-matrix-span-a-given-vector/ (https://ask.sagemath.org/question/39481/checking-if-a-subset-of-columns-of-a-matrix-span-a-given-vector/)

Un vecteur peut être encodé comme liste [1,0,-2,-1] ou comme vecteur vector([1,0,-2,-1]) ou comme matrice ligne matrix([1,0,-2,-1]) ou comme matrice colonne matrix(4,1,[1,0,-2,-1]). L'avantage des trois derniers est qu'on peut comparer le vecteur à 0 :

In [1]:

```
print vector([1,0,-2,-1])==0, vector([0,0,0,0])==0, [0,0,0,0]==0
```

```
False True False
```

In [2]:

```
span([vector([1,0,-2,-1])],QQ) #QQ plutôt que RR pour une représentation exacte
 des nombres
```

Out[2]:

```
Vector space of degree 4 and dimension 1 over Rational Field
Basis matrix:
[ 1  0 -2 -1]
```

In [3]:

```
#test si un vecteur u est combinaison d'une liste de vecteurs l
#u et les éléments de l sont de type vecteur de nombres rationnels
def est_combi(u,l):
    if l==[]:return(u==0)
    else:return(span([u],QQ).is_subspace(span(l,QQ)))
```

In [4]:

```
u=vector([1,0,-2,-1]);v=vector([1,6,4,-1]);w=vector([-2,-3,1,2])
print est_combi(u,[]), est_combi(v,[u]), est_combi(w,[u,v])
```

```
False False True
```

On en déduit que (u,v) est une base de vect(u,v,w)

Rq. Méthode matricielle avec la fonction rank() et le fait qu'une matrice à même rang que sa transposée :

In [14]:

```
print matrix([u,v,w])#transposée de la matrice des coordonnées de u,v,w
print
print matrix([u]).rank(), matrix([u,v]).rank(), matrix([u,v,w]).rank()
```

```
[ 1  0 -2 -1]
[ 1  6  4 -1]
[-2 -3  1  2]

1 2 2
```

Pour obtenir une équation de F, on complète la famille (u,v) en une base (u,v,f,g) de $\mathbb{R}^4$; un vecteur $(x_1,\ldots,x_4)$ de $\mathbb{R}^4$ est dans F si et seulement si ses deux dernières coordonnées dans la base (u,v,f,g) sont nulles.

Coordonnées dans la nouvelle base : si P est la matrice des coordonnées (dans la base canonique de $\mathbb{R}^4$) des vecteurs u,v,f,g, X les coordonnées d'un vecteur x dans la base canonique et Y les coordonnées de x dans la base (u,v,f,g) alors $X=PY$ donc $Y=P^{-1}X$.

In [33]:

```
#Complétion de (u,v) en une base de Q^4
I=matrix.identity(4) #matrice des coordonnées de la base canonique
print vector(I[1,:])
```

```
(0, 1, 0, 0)
```

In [34]:

```
l=[u,v] #base de F qu'on va compléter en une base de R^4
for i in range(4):
    if not est_combi(vector(I[i,:]),l):l=l+[vector(I[i,:])]
print l
```

```
[(1, 0, -2, -1), (1, 6, 4, -1), (1, 0, 0, 0), (0, 1, 0, 0)]
```

In [8]:

```
P=matrix(l).transpose();print P #matrice des coordonnées de la nouvelle base
```

```
[ 1  1  1  0]
[ 0  6  0  1]
[-2  4  0  0]
[-1 -1  0  0]
```

In [9]:

```
var('x1 x2 x3 x4');X=vector([x1,x2,x3,x4]);print X
print
print 'P^(-1)*X = ',P^(-1)*X
print 'Equations :',[(P^(-1)*X)[i]==0 for i in [2,3]]
```

```
(x1, x2, x3, x4)

P^(-1)*X =  (-1/6*x3 - 2/3*x4, 1/6*x3 - 1/3*x4, x1 + x4, x2 - x3 +
2*x4)
Equations : [x1 + x4 == 0, x2 - x3 + 2*x4 == 0]
```

Vérification : base de solutions de l'équation trouvée :

In [17]:

```
solve([x1 + x4 == 0, x2 - x3 + 2*x4 == 0],[x1,x2,x3,x4])
```

Out[17]:

```
[[x1 == -r1, x2 == -2*r1 + r2, x3 == r2, x4 == r1]]
```

Solve rend un système d'équations équivalent qui s'interprète comme un paramétrage des solutions par une application linéaire bijective. Lorsque (r_1, r_2) décrit une base de $\mathbb{R}^2$, $(x1, x2, x3, x4)$ décrit une base de l'espace des solutions.

In [31]:

```
soleq=[x1 == -r1, x2 == -2*r1 + r2, x3 == r2, x4 == r1]
print X.subs(soleq)
print "base de solutions : ",X.subs(soleq)(r1=1,r2=0),X.subs(soleq)(r1=0,r2=1)
```

```
(-r1, -2*r1 + r2, r2, r1)
base de solutions :  (-1, -2, 0, 1) (0, 1, 1, 0)
```

In [32]:

```
sol=span([X.subs(soleq)(r1=1,r2=0),X.subs(soleq)(r1=0,r2=1)],QQ)# Espace des sol
utions
print "sol=F ?",sol==span([u,v,w],QQ)
```

```
sol=F ? True
```

2ème approche : Méthode matricielle, pivot de Gauss

pour trouver une équation de F : algorithme du pivot sur les lignes de la matrice des coordonnées des vecteurs générateurs de F

In [10]:

```
M=matrix([u,v,w]).transpose();print M
```

```
[ 1  1 -2]
[ 0  6 -3]
[-2  4  1]
[-1 -1  2]
```

In [11]:

```
MM=copy(M)
P=MM.echelonize(transformation=true) #matrice des opérations sur les lignes. MM
 est transformée par cette instruction
print P*M
```

```
[ 1  1 -2]
[ 0  6 -3]
[ 0  0  0]
[ 0  0  0]
```

In [12]:

```
Maug=block_matrix(1,2,[M,matrix(X).transpose()]);print Maug
print
show(P*Maug)
```

```
[ 1  1 -2|x1]
[ 0  6 -3|x2]
[-2  4  1|x3]
[-1 -1  2|x4]
```

Out[12]:

$$\left(\begin{array}{rrr|r} 1 & 1 & -2 & -x_4 \\ 0 & 6 & -3 & x_3 - 2\,x_4 \\ 0 & 0 & 0 & x_1 + x_4 \\ 0 & 0 & 0 & x_2 - x_3 + 2\,x_4 \end{array}\right)$$

Equation : rang(M)=rang(Maug) donc annulation des deux dernières lignes de P*Maug