# Alglin-corrige

December 12, 2019

## 1 Algorithme de réduction des matrices

### 1.1 Algorithmes de Gauss et de Smith

#### 1.1.1 1-Matrices en Sage

En Sage, on définit des matrices comme suit:

```
In [3]: A = matrix(QQ, 2,3, [1,2,3,4,5,6])
```

ou comme suit:

```
In [4]: B = matrix(ZZ, [[1,2,3],[4,5,6]])
```

Ou encore:

```
In [5]: A=matrix(QQ,3,[i/(2*i^2+6) for i in range(18) if i%3==0]); A

Out[5]: [    0   1/8]
        [ 1/13  3/56]
        [ 2/49 5/152]

In [6]: A = matrix(QQ, 2,3, [1,2,3,4,5,6])

In [0]:
```

On obtient la taille d'une matrice comme suit:

```
In [7]:  A.nrows(),A.ncols()

Out[7]: (2, 3)
```

Le test d'égalité marche...

```
In [8]: A==B

Out[8]: True
```

Définir quelques matrices spéciales:

```
In [9]: identity_matrix(5)

Out[9]: [1 0 0 0 0]
        [0 1 0 0 0]
        [0 0 1 0 0]
        [0 0 0 1 0]
        [0 0 0 0 1]

In [10]: matrix(QQ,4,3,0)

Out[10]: [0 0 0]
         [0 0 0]
         [0 0 0]
         [0 0 0]
```

Par blocs...

```
In [11]: E = block_matrix([[A,0],[0,B]]); E

Out[11]: [1 2 3|0 0 0]
         [4 5 6|0 0 0]
         [-----+-----]
         [0 0 0|1 2 3]
         [0 0 0|4 5 6]

In [12]: C=matrix(QQ,0,0,0)

In [13]: block_matrix([[A,0],[0,B]])

Out[13]: [1 2 3|0 0 0]
         [4 5 6|0 0 0]
         [-----+-----]
         [0 0 0|1 2 3]
         [0 0 0|4 5 6]
```

Pour extraire des sous-matrices:

```
In [14]: A.matrix_from_rows_and_columns([0,1],[2])

Out[14]: [3]
         [6]
```

### 1.1.2 2-Pivot de Gauss

Pour échelonner une matrice (en lignes), Sage a une fonction déjà codée:

```
In [15]: C=A.echelon_form()
```

Attention: le résultat que l'on obtient est une matrice immuable: on ne peut pas changer ses valeurs.

```
In [16]: C[1,1]=3


         ---------------------------------------------------------------------------

         ValueError                                Traceback (most recent call last)

         <ipython-input-16-1627321a9699> in <module>()
    ----> 1 C[Integer(1),Integer(1)]=Integer(3)


         /ext/sage/sage-8.4_1804/local/lib/python2.7/site-packages/sage/matrix/matrix0.pyx in sag
        1372         # If the matrix is immutable, check_mutability will raise an
        1373         # exception.
    -> 1374         self.check_mutability()
        1375
        1376         if type(key) is tuple:


         /ext/sage/sage-8.4_1804/local/lib/python2.7/site-packages/sage/matrix/matrix0.pyx in sag
         382         """
         383         if self._is_immutable:
    --> 384             raise ValueError("matrix is immutable; please change a copy instead (i.e
         385         else:
         386             self._cache = None


         ValueError: matrix is immutable; please change a copy instead (i.e., use copy(M) to chan
```

Si l'on veut l'éditer, il faut en créer une copie:

```
In [17]: C=copy(A.echelon_form())

In [18]: C[1,1]=3; C

Out[18]: [ 1  0 -1]
         [ 0  3  2]
```

On a déjà des fonctions codées pour les opérations sur les lignes et les colonnes:

```
In [19]: A.with_added_multiple_of_column(0,1,2), B.with_added_multiple_of_row(0,1,3)

Out[19]: (
         [ 5  2  3]  [13 17 21]
         [14  5  6], [ 4  5  6]
         )

In [20]: A.with_swapped_columns(1,2)
```

```
Out[20]: [1 3 2]
         [4 6 5]

In [21]: A.with_rescaled_col(2,2),B.with_rescaled_row(1,3)

Out[21]: (
         [ 1  2  6]  [ 1  2  3]
         [ 4  5 12], [12 15 18]
         )
```

La fonction *echelon_form* de Sage ne donne pas la matrice de passage...

Soit $A \in \mathrm{M}_n(\mathbf{Q})$ une matrice. On a vu en cours qu'il est très utile de disposer de matrices inversibles $P$ et $Q$ telles que $PAQ$ soit "diagonale". Il y a une fonction prédéfinie dans Sage qui réduit selon les lignes et selon les colonnes et renvoie les matrices de passage: c'est la fonction *smith_form()*.

```
In [22]: A.smith_form()

Out[22]: (
                                  [ 1 -2  1]
         [1 0 0]  [   1    0]  [ 0  1 -2]
         [0 1 0], [ 4/3 -1/3], [ 0  0  1]
         )
```

Dans un premier temps, vous pouvez utiliser cette fonction. Vous pourrez la coder vous-même à la fin de la séance.

```
In [0]:
```

**Exercice 1**: Soient $u, v, w, z$ les vecteurs de $\mathbf{Q}^{10}$ ci-dessous.

a)Donner un système d'équations pour le sous-e.v. de $\mathbf{Q}^{10}$ engendré par ces vecteurs

```
In [23]: u=vector(QQ, [1/i for i in primes_first_n(10)]);
         v=vector(QQ, [integer_floor(exp(2*i)) for i in range(10)])
         w=vector(QQ, [i for i in range(10)])
         z=vector(QQ, [1,0,4,12/5, 0,0,0,0,0,1/3])
```

Pour trouver un système d'équations vérifiées par ces 4 vecteurs, on utilise la méthode présentée en td: on les met dans une matrice, on calcule le forme de Gauss et on lit le résultat dans la matrice de passage...

```
In [24]: A=matrix (QQ,4,[u,v,w,z]); A=A.transpose(); A

Out[24]: [     1/2        1        0        1]
         [     1/3        7        1        0]
         [     1/5       54        2        4]
         [     1/7      403        3     12/5]
         [    1/11     2980        4        0]
         [    1/13    22026        5        0]
         [    1/17   162754        6        0]
         [    1/19  1202604        7        0]
         [    1/23  8886110        8        0]
         [    1/29 65659969        9      1/3]
```

Pour une matrice à coefficients rationnels, la fonction Smith_form() calcule la forme de Gauss avec les matrices de passage.

```
In [25]: D,P,Q=A.smith_form(); D

Out[25]: [1 0 0 0]
         [0 1 0 0]
         [0 0 1 0]
         [0 0 0 1]
         [0 0 0 0]
         [0 0 0 0]
         [0 0 0 0]
         [0 0 0 0]
         [0 0 0 0]
         [0 0 0 0]
```

On lit que $A$ est de rang 4: il faut donc trouver 10-4=6 équations. Les coefficients de ces équations sont dans les 6 dernières lignes de $P$.

```
In [30]: E=P.matrix_from_rows(range(4,10));E

Out[30]: [      -14540732/2489861           1853076/226351        16173065/2489861       -2089647
         [     -165476498/2942563       470788065/5885126         11716730/226351      -36982788
         [    -1663854324/3847967      2368317918/3847967      1490407875/3847967     -179074049
         [   -13829929270/4300669     39368345559/8601338     12333896350/4300669    -2958804677
         [  -123831141352/5206073    176245761021/5206073    110358778765/5206073   -13233498904
         [-3461625677987/19692537  3284549679853/13128358  2056450143305/13128358 -739756812168

In [32]: R = PolynomialRing(QQ,10,"x") # On définit un anneau où peut vivre l'équation; ici un a

In [42]: v=vector(R.gens())

In [43]: E*v

Out[43]: (-14540732/2489861*x0 + 1853076/226351*x1 + 16173065/2489861*x2 - 20896470/2489861*x3 +
```

On vérifie notre résultat:

```
In [31]: E*A

Out[31]: [0 0 0 0]
         [0 0 0 0]
         [0 0 0 0]
         [0 0 0 0]
         [0 0 0 0]
         [0 0 0 0]
```

b) Donner une base de l'intersection de l'ev précédent avec l'hyperplan $\sum_{i=0}^{10} x_i = 0$.

On rajoute cette équation aux précédentes (celles de E) dans une matrice $E_1$

```
In [47]: W=matrix(QQ,1,10,[1 for i in range(10)]); W

Out[47]: [1 1 1 1 1 1 1 1 1 1]

In [49]: E1=block_matrix([[E],[W]]); E1

Out[49]: [      -14540732/2489861           1853076/226351        16173065/2489861       -2089647
         [     -165476498/2942563        470788065/5885126          11716730/226351      -36982788
         [    -1663854324/3847967       2368317918/3847967       1490407875/3847967     -179074049
         [   -13829929270/4300669      39368345559/8601338      12333896350/4300669    -2958804677
         [  -123831141352/5206073     176245761021/5206073     110358778765/5206073   -13233498904
         [-3461625677987/19692537  3284549679853/13128358  2056450143305/13128358 -7397568121685
         [-----------------------------------------------------------------------------------------
         [                       1                         1                       1

In [53]: D1,P1,Q1=E1.smith_form(); D1

Out[53]: [1 0 0 0 0 0 0 0 0 0]
         [0 1 0 0 0 0 0 0 0 0]
         [0 0 1 0 0 0 0 0 0 0]
         [0 0 0 1 0 0 0 0 0 0]
         [0 0 0 0 1 0 0 0 0 0]
         [0 0 0 0 0 1 0 0 0 0]
         [0 0 0 0 0 0 1 0 0 0]
```

$E_1$ est de rang 7, donc l'équation rajoutée n'est pas combinaison linéaire des précédentes (ce n'est pas très surprenant: les vecteurs de départ ne vérifiaient pas cette équation!). On cherche une base du noyau de $E_1$ qui est donc de dim 3. D'après le td, on lit cette base dans les colonnes de $Q1$.

```
In [58]: B=Q1.matrix_from_columns(range(7,10)); B

Out[58]: [  1077499083169018460 2/201553337623442969  -1070968230502539105 5/201553337623442969
         [  1069128691373947113 2/201553337623442969  -2092612669549766701 5/403106675246885938
         [-8147694478816359675 4/100776668811721484 5  7753226043042503169 9/100776668811721484 -1
         [-4716532824059799856 6/100776668811721484 5  9017991342467974599 7/201553337623442690  -
         [   2078600530538451510/201553337623442969   -1924575236886685238/201553337623442969
         [   1396739956223422789/201553337623442969   -1226677828047530487/201553337623442969
         [    585283035937346062/201553337623442969    -4019980484679836 21/201553337623442969
         [                                        1                                          0
         [                                        0                                          1
         [                                        0                                          0
```

Ces 3 colonnes forment une base du noyau de $E_1$. Vérifions le:

```
In [59]: E1*B

Out[59]: [0 0 0]
         [0 0 0]
         [0 0 0]
```

```
[0 0 0]
[0 0 0]
[0 0 0]
[-----]
[0 0 0]
```

## 1.2 3-Réduction des matrices entières

**Exercice 2:** Soit $A$ la matrice entière suivante.

```
In [96]: A=matrix(ZZ,[[2,4,0],[0,3,6],[6,0,5]]); A

Out[96]: [2 4 0]
         [0 3 6]
         [6 0 5]
```

**a)** Appliquez lui pas à pas, en utilisant les fonctions prédéfinies, les suites d'opérations élémentaires comme dans l'algorithme présenté en cours.

```
In [97]: A.add_multiple_of_row(0,1,-1); A

Out[97]: [ 2  1 -6]
         [ 0  3  6]
         [ 6  0  5]

In [98]: A.swap_columns(0,1);A

Out[98]: [ 1  2 -6]
         [ 3  0  6]
         [ 0  6  5]

In [99]: A.add_multiple_of_column(1,0,-2); A

Out[99]: [ 1  0 -6]
         [ 3 -6  6]
         [ 0  6  5]

In [100]: A.add_multiple_of_column(2,0,6); A

Out[100]: [ 1  0  0]
          [ 3 -6 24]
          [ 0  6  5]

In [101]: A.add_multiple_of_row(1,0,-3); A

Out[101]: [ 1  0  0]
          [ 0 -6 24]
          [ 0  6  5]

In [102]: A.add_multiple_of_row(1,2,-5);A
```

```
Out[102]: [  1   0   0]
          [  0 -36  -1]
          [  0   6   5]

In [103]: A.swap_columns(2,1); A

Out[103]: [  1   0   0]
          [  0  -1 -36]
          [  0   5   6]

In [104]: A.add_multiple_of_column(2,1,-36); A

Out[104]: [   1    0    0]
          [   0   -1    0]
          [   0    5 -174]

In [105]: A.add_multiple_of_row(2,1,5); A

Out[105]: [   1    0    0]
          [   0   -1    0]
          [   0    0 -174]
```

C'est bon: A est sous la forme d'une matrice diagonale. On vérifie que Sage est d'accord avec nous...

```
In [116]: matrix(ZZ,[[2,4,0],[0,3,6],[6,0,5]]).smith_form()

Out[116]: (
          [  1   0   0]  [  0   0   1]  [  1   0   5]
          [  0   1   0]  [  1  -1 -10]  [  2   1 -46]
          [  0   0 174], [  3  -4 -30], [ -1   0  -6]
          )

In [0]:
```

**N.B.** En sage, l'algorithme présenté en cours est aussi codé dans la fonction *smith_form* lorsqu'on l'utilise sur une matrice à coefficients entiers.

**b)** A quelle conditions sur les entiers $a, b, c$ existe-t-il des entiers $x, y, z$ tels que l'on ait $A $\begin{bmatrix} x \\ y \\ z \end{bmatrix}$ = $\begin{bmatrix} a \\ b \\ c \end{bmatrix}$ $?

On cherche les "équations" de l'image. On les trouve toujours grâce à la matrice de passage "P" dans la forme de Smith. Plutôt que d'utiliser celle que me fournit Smith_form(), je récupère la "mienne" en faisant sur la matrice identité les mêmes opérations sur les lignes.

```
In [117]: P=identity_matrix(3); P.add_multiple_of_row(0,1,-1);P.add_multiple_of_row(1,0,-3);P.ad
```

```
Out[117]: [  1  -1   0]
          [ -3   4  -5]
          [-15  20 -24]
```

On trouve l'équation

$$-15x + 20y - 24z \equiv 0[174].$$

**N.B.:** En utilisant la matrice de Sage, on touverait plutôt l'équation

$$3x - 4y - 30z \equiv 0[174].$$

Qui a tort, qui a raison? Les deux, bien entendu: en multipliant cette de Sage par -5 (qui est inversible dans $ Z/174 Z$) on trouve la mienne ($5 \times 30 = 150 \equiv -24[174]$).

```
In [0]:
```

```
In [0]:
```

```
In [0]:
```

**Exercice 3** a)Exhiber une matrice $P$ inversible dans $\mathrm{M}_n(\mathbb{Z})$ vérifiant $\begin{pmatrix} 24 & 18 & 12 & 15 \end{pmatrix} P = \begin{pmatrix} d & 0 & 0 & 0 \end{pmatrix}$ pour un entier $d$ convenable.

```
In [122]: A=matrix(ZZ,1,[24,18,12,15]); D,Q,P=A.smith_form(); D,P
```

```
Out[122]: (
                     [-2 -5 -2 -3]
                     [ 2  5  2  4]
                     [ 0  0  1  0]
          [3 0 0 0], [ 1  2  0  0]
          )
```

Donc $d = 3$, et $P$ est la matrice ci-dessus.

```
In [0]:
```

b)En déduire une $\mathbf{Z}$-base du noyau de l'application $\mathbf{Z}^4 \to \mathbf{Z}$, $(x, y, z, t) \mapsto 24x + 18y + 12z + 15t$.

Elles se lit dans les colonnes de $P$. Ce sont les 3 dernières colonnes de cette matrice.

```
In [141]: P1=P.matrix_from_columns(range(1,4)); P1
```

```
Out[141]: [-5 -2 -3]
          [ 5  2  4]
          [ 0  1  0]
          [ 2  0  0]
```

c)Puis un repère affine des solutions entières de l'équation $24x + 18y + 12z + 15t = 9$.

Les solutions s'obtiennent en ajoutant une solution particulière à la solution générale de l'équation homogène trouvée à la question précédente. On trouve une solution particulière à partir d'une solution particulière de $3x + 0y + 0z + 0t = 9$, à savoir (3,0,0,0).

```
In [132]: a=vector([3,0,0,0])

In [137]: P.inverse()

Out[137]: [ 8  6  4  5]
          [-4 -3 -2 -2]
          [ 0  0  1  0]
          [ 1  1  0  0]

In [138]: a0=P*a; a0

Out[138]: (-6, 6, 0, 3)

In [139]: a0*vector([24,18,12,15])

Out[139]: 9
```

La solution générale est donc de la forme:

```
In [142]: R.<u,v,w>=PolynomialRing(QQ)

In [143]: vector(a0)+P1*vector([u,v,w])

Out[143]: (-5*u - 2*v - 3*w - 6, 5*u + 2*v + 4*w + 6, v, 2*u + 3)
```

**Exercice 4** (examen 2017) Donner une base du groupe abélien formé des éléments de $\mathbf{Z}^5$ qui sont combinaisons linéaires à *coefficients rationnels* des deux vecteurs $(1,2,3,4,5)$ et $(6,7,8,9,10)$.

Comme d'habitude, on résout cette question en se ramenant au cas d'une matrice diagonale en utilisant Smith.

```
In [3]: u=range(1,6); v=range(6,11); u,v

Out[3]: ([1, 2, 3, 4, 5], [6, 7, 8, 9, 10])

In [7]: A=matrix(ZZ,2,[u,v]); A=A.transpose(); A

Out[7]: [ 1  6]
        [ 2  7]
        [ 3  8]
        [ 4  9]
        [ 5 10]

In [0]:

In [8]: d,p,q=A.smith_form(); d,p,q

Out[8]: (
        [1 0]  [ 0  0  0  4 -3]
        [0 5]  [ 0  0  0 -5  4]
        [0 0]  [ 1  0  0 -4  3]
        [0 0]  [ 0  1  0 -3  2]  [ 1  6]
        [0 0], [ 0  0  1 -2  1], [ 0 -1]
        )
```

Pour la matrice d, une base de l'image entière est $e_1$ et $5e_2$) (où $e_i$ est le ième vecteur de la base canonique). Les cominaisons rationnelles qui restent entières sont les combinaisons linéaires entières de $e_1$ et $e_2$. Comme Im $A = P^{-1}$Im $d$, on en déduit qu'une base entière est formée des 2 premières colonnes de $P^{-1}$.

```
In [11]: (p.inverse()).matrix_from_columns([0,1])

Out[11]: [1 0]
         [2 1]
         [3 2]
         [4 3]
         [5 4]

In [0]:

In [93]: A.add_multiple_of_column(0,1,7); A

Out[93]: [29  2  3]
         [74  5  6]
```

### 1.3 Programmation

**Exercice** Programmez une fonction analogue à smith_form:
a)pour les matrices rationnelles
b)pour les matrices entières
Voir le corrigé dans le cours.