L'algorithme FGLM

L'examen. Le texte suivant présente l'algorithme FGLM sur lequel porte une partie du sujet d'examen (les deux sessions). Parmi les explications, on trouve des considérations sur les bases de Gröbner et des considérations d'algèbre linéaire. Les questions porteront sur le cours (les bases de Gröbner) et *a priori* pas sur les méthodes d'algèbre linéaire. Les questions peuvent consister à justifier certains points en citant des théorèmes du cours ou à programmer en MAPLE certaines des opérations menées interactivement ci–dessous.

Principe. L'algorithme FGLM (des initiales de ses inventeurs : Jean-Charles Faugère, Patricia Gianni, Daniel Lazard et Teo Mora) est un algorithme de changement d'ordre admissible pour bases de Gröbner. Il résout efficacement le problème suivant : étant donnés

1. une base de Gröbner $base_1$ d'un idéal $\mathscr{I}$ pour un ordre admissible $ordre_1$ et
2. un ordre admissible $ordre_2 \neq ordre_1$,

calculer la base de Gröbner réduite $base_2$ de $\mathscr{I}$ pour l'ordre admissible $ordre_2$. L'algorithme ne s'applique pas à toutes les bases de Gröbner : il ne s'applique que si l'idéal $\mathscr{I}$ admet un nombre fini de solutions complexes (le cas abordé en cours).

Illustration sur un exemple. La variable $base_1$ contient une base de Gröbner pour l'ordre admissible $ordre_1$.

```
with (Groebner):
with (LinearAlgebra):
p1 := 2*x*y^2  - 5*y^2 - y + 2;
p2 := x^2*y + x - y - 4;
ordre1 := plex (y, x);
base1 := gbasis ([p1, p2], ordre1);
                 2            4      3                                  3       2
    [-74 - 29 x  + 71 x + 2 x  + 3 x , 285 x + 315 y - 368 - 26 x  - 83 x ]
```

Voici les équations de la base, sous la forme de règles de réécriture, calculées avec la fonction *regle_reec* donnée au cours 11.

```
map (regle_reec, base1, ordre1);
     4               2                  3         19       368    26    3    83    2
   [x  = 37 + 29/2 x  - 71/2 x - 3/2 x , y = --- x + --- + --- x  + --- x ]
                                             21       315   315        315
```

D'un point de vue calculatoire, l'algorithme exécute une boucle qui énumère les termes par ordre croissant pour l'ordre admissible $ordre_2$:

$$1 = t_0 < t_1 < t_2 < t_3 < \cdots$$

À chaque itération i, la forme normale $\mathrm{NF}(t_i)$ du terme courant t_i est calculée, pour l'ordre admissible *ordre*$_1$ et l'algorithme cherche s'il existe des rationnels $\lambda_0, \ldots, \lambda_i$ tels que

$$\lambda_0\ \mathrm{NF}(t_0) + \lambda_1\ \mathrm{NF}(t_1) + \cdots + \lambda_i\ \mathrm{NF}(t_i) = 0.$$

Dès qu'une telle combinaison linéaire est trouvée, l'algorithme en déduit que le polynôme

$$\lambda_0\, t_0 + \lambda_1\, t_1 + \cdots + \lambda_i\, t_i \in \mathscr{I}$$

et il le rajoute à la base de Gröbner *base*$_2$ en construction.

Recherche du premier polynôme. Voici comment procéder concrètement. La variable *ordre*$_2$ reçoit l'ordre admissible cible *ordre*$_2$. Initialement, la base de Gröbner *base*$_2$ est vide.

```
ordre2 := tdeg (y, x);
base2 := [];
```

Affectons à T l'ensemble des termes irréductibles par *base*$_1$ et à *AEnumerer* la liste des (mettons) cinq premiers termes pour l'ordre *ordre*$_2$ (ils sont énumérés par ordre croissant).

```
T := [1, x, x^2, x^3];
AEnumerer := [1, x, y, x^2, x*y];
```

On voit que les formes normales des termes à énumérer ne dépendent que des termes présents dans T.

```
for i from 1 to nops (AEnumerer) do
    normalf (AEnumerer [i], base1, ordre1)
od;
                                 1

                                 x

                    19      368   26   3   83   2
                   --- x + --- + --- x   + --- x
                    21      315   315       315

                                 2
                                x

                   92   2   37      44   3   962
                   --- x  - -- x + --- x   + ---
                   315      21     315       315
```

Ces formes normales sont–elles linéairement dépendantes sur $\mathbb{Q}$? Pour le savoir, on les voit comme des vecteurs dans la base T, on forme une matrice $m \times n$ où m désigne le nombre de formes normales et n le nombre d'éléments de T.

```
m := nops (AEnumerer);
n := nops (T);
M := Matrix (m, n);
for lig from 1 to m do
    nf := normalf (AEnumerer [lig], base1, ordre1);
    koeffs := [ coeffs (nf, indets (nf), 'termes') ];
    termes := [ termes ];
    for j from 1 to nops (termes) do
        member (termes [j], T, 'col');
        M [lig, col] := koeffs [j];
    od;
od;
M;
                                    [ 1      0      0      0 ]
                                    [                        ]
                                    [ 0      1      0      0 ]
                                    [                        ]
                                    [368    -19    83     26 ]
                                    [---    ---    ---    ---]
                                    [315    21     315    315]
                                    [                        ]
                                    [ 0      0      1      0 ]
                                    [                        ]
                                    [962    -37    92     44 ]
                                    [---    ---    ---    ---]
                                    [315    21     315    315]
```

En appliquant un pivot de Gauss, on voit que la dernière ligne est identiquement nulle : les formes normales sont bien linéairement dépendantes[1].

```
GaussianElimination (M);
                                      [1     0      0      0 ]
                                      [                      ]
                                      [0     1      0      0 ]
                                      [                      ]
                                      [            83     26 ]
                                      [0     0     ---    ---]
                                      [            315    315]
                                      [                      ]
                                      [                   -26]
                                      [0     0      0     ---]
                                      [                   83 ]
                                      [                      ]
                                      [0     0      0      0 ]
```

On obtient les coefficients λ_i par une petite manipulation d'algèbre linéaire : on borde la matrice M par la matrice identité $m \times m$ et on applique à nouveau le pivot de Gauss. Les coefficients recherchés sont sur les m dernières colonnes de la dernière ligne de la matrice ainsi obtenue.

[1]Il n'est pas nécessaire de procéder à un pivot de Gauss puisque le nombre de lignes est supérieur au nombre de colonnes mais on ne tient pas compte de ce raisonnement qui est particulier à l'exemple.

```
M1 := < M | IdentityMatrix (m) >;
                 [ 1      0      0      0     1    0    0    0    0]
                 [                                                  ]
                 [ 0      1      0      0     0    1    0    0    0]
                 [                                                  ]
                 [368    -19    83     26                           ]
                 [---    ---    ---    ---    0    0    1    0    0]
           M1 := [315    21     315    315                          ]
                 [                                                  ]
                 [ 0      0      1      0     0    0    0    1    0]
                 [                                                  ]
                 [962    -37    92     44                           ]
                 [---    ---    ---    ---    0    0    0    0    1]
                 [315    21     315    315                          ]

M2 := GaussianElimination (M1);
             [1    0     0      0      1      0      0      0      0]
             [                                                     ]
             [0    1     0      0      0      1      0      0      0]
             [                                                     ]
             [           83     26     -368    19                   ]
             [0    0     ---    ---    ----    --     1      0      0]
             [           315    315    315     21                   ]
       M2 := [                                                     ]
             [                  -26    368    -285   -315           ]
             [0    0     0      ---    ---    ----   ----    1      0]
             [                  83     83      83     83            ]
             [                                                     ]
             [                         -14            -22           ]
             [0    0     0      0      ---    3/13   ---    2/13   1]
             [                         13             13            ]

lambdas := Row (SubMatrix (M2, 1..m, n+1..n+m), -1);
                              [-14          -22           ]
                   lambdas := [---, 3/13, ---, 2/13, 1]
                              [13           13            ]
```

Il ne reste plus qu'à multiplier les coefficients λ_i par les termes t_i pour obtenir un premier polynôme de la base de Gröbner recherchée.

```
q1 := lambdas . Vector (m, AEnumerer [1..m]);
                         14             22            2
                 q1 := - -- + 3/13 x - -- y + 2/13 x  + x y
                         13             13
base2 := [ op (base2), q1 ];
map (regle_reec, base2, ordre2);
                           14             22            2
                   [x y = -- - 3/13 x + -- y - 2/13 x ]
                           13             13
```

Recherche d'un deuxième polynôme. On cherche maintenant un deuxième polynôme de la base *base*$_2$. Pour cela, on commence par retirer le terme $x\,y$ de la liste *AEnumerer*. Les formes normales des termes restants sont linéairement indépendantes. Ensuite, on rajoute à cette liste, un à la fois, les termes supérieurs à $x\,y$ pour l'ordre *ordre*$_2$, en évitant les multiples du terme $x\,y$. On continue jusqu'à détecter une nouvelle dépendance linéaire. Sur l'exemple, on voit que la forme normale du premier terme considéré, y^2, dépend linéairement des autres formes normales.

```
AEnumerer := [1, x, y, x^2];
AEnumerer := [1, x, y, x^2, y^2];
m := nops (AEnumerer);
n := nops (T);
M := Matrix (m, n);
for lig from 1 to m do
    nf := normalf (AEnumerer [lig], base1, ordre1);
    koeffs := [ coeffs (nf, indets (nf), 'termes') ];
    termes := [ termes ];
    for j from 1 to nops (termes) do
        member (termes [j], T, 'col');
        M [lig, col] := koeffs [j];
    od;
od;
M;
                                 [ 1       0      0       0  ]
                                 [                           ]
                                 [ 0       1      0       0  ]
                                 [                           ]
                                 [368     -19    83      26  ]
                                 [---     ---    ---     --- ]
                                 [315     21     315     315 ]
                                 [                           ]
                                 [ 0       0      1       0  ]
                                 [                           ]
                                 [1426    97     361      22 ]
                                 [----    ---    ----    ----]
                                 [6615    441    6615    6615]

GaussianElimination (M);
                                    [1    0     0      0 ]
                                    [                    ]
                                    [0    1     0      0 ]
                                    [                    ]
                                    [          83     26 ]
                                    [0    0    ---    ---]
                                    [          315    315]
                                    [                    ]
                                    [                 -26]
                                    [0    0     0     ---]
                                    [                 83 ]
                                    [                    ]
                                    [0    0     0      0 ]
```

Par le même procédé que précédemment, on obtient un nouveau polynôme de la base *base*$_2$.

```
M1 := < M | IdentityMatrix (m) >;
M2 := GaussianElimination (M1);
lambdas := Row (SubMatrix (M2, 1..m, n+1..n+m), -1);

                              [-46  -10  -11  -4   ]
                   lambdas := [---, ---, ---, --, 1]
                              [273  39   273  91   ]

q2  := lambdas . Vector (m, AEnumerer [1..m]);
base2 := [ op (base2), q2 ];
map (regle_reec, base2, ordre2);

          14            22            2   2   46    10     11            2
   [x y = -- - 3/13 x + -- y - 2/13 x , y  = --- + -- x + --- y + 4/91 x ]
          13            13                   273   39     273
```

On recommence : on retire le terme y^2 de la liste *AEnumerer*. On rajoute le plus petit terme supérieur à y^2 en évitant les multiples de $x\,y$ et de y^2. Il s'agit de x^3. On trouve à nouveau une dépendance linéaire entre les formes normales et donc un nouveau polynôme de *base*$_2$.

```
AEnumerer := [1, x, y, x^2];
AEnumerer := [1, x, y, x^2, x^3];

m := nops (AEnumerer);
n := nops (T);
M := Matrix (m, n);
for lig from 1 to m do
    nf := normalf (AEnumerer [lig], base1, ordre1);
    koeffs := [ coeffs (nf, indets (nf), 'termes') ];
    termes := [ termes ];
    for j from 1 to nops (termes) do
        member (termes [j], T, 'col');
        M [lig, col] := koeffs [j];
    od;
od;
M;
                          [ 1      0      0      0 ]
                          [                        ]
                          [ 0      1      0      0 ]
                          [                        ]
                          [368    -19    83     26 ]
                          [---    ---    ---    ---]
                          [315    21     315    315]
                          [                        ]
                          [ 0      0      1      0 ]
                          [                        ]
                          [ 0      0      0      1 ]


GaussianElimination (M);
```

```
                        [1    0     0      0 ]
                        [                    ]
                        [0    1     0      0 ]
                        [                    ]
                        [          83     26 ]
                        [0    0    ---    ---]
                        [          315    315]
                        [                    ]
                        [                -26]
                        [0    0     0     ---]
                        [                 83 ]
                        [                    ]
                        [0    0     0      0 ]
```

```
M1 := < M | IdentityMatrix (m) >;
M2 := GaussianElimination (M1);
lambdas := Row (SubMatrix (M2, 1..m, n+1..n+m), -1);
```

```
                             [184  -285  -315  83   ]
                  lambdas := [---, ----, ----, --, 1]
                             [13    26    26   26   ]
```

```
q3 := lambdas . Vector (m, AEnumerer [1..m]);
base2 := [ op (base2), q3 ];
map (regle_reec, base2, ordre2);
map (regle_reec, base2, ordre2);
```

```
       14            22            2   2   46    10     11             2
[x y = -- - 3/13 x + -- y - 2/13 x , y  = --- + -- x + --- y + 4/91 x ,
       13            13                   273   39     273

      3     184   285     315     83  2
     x  = - --- + --- x + --- y - -- x ]
            13    26      26      26
```

Arrêt de l'algorithme. Il n'existe aucun terme supérieur à x^3 qui ne soit multiple d'aucun des termes de tête de la base *base*$_2$. L'algorithme FGLM s'arrête et retourne *base*$_2$. On peut vérifier par acquis de conscience qu'il s'agit bien de la même base que celle que l'algorithme de Buchberger aurait calculée.

```
basis2 := gbasis ([p1, p2], ordre2):
 map (regle_reec, basis2, ordre2);
       14            22            2   2   46    10     11             2
[x y = -- - 3/13 x + -- y - 2/13 x , y  = --- + -- x + --- y + 4/91 x ,
       13            13                   273   39     273

      3     184   285     315     83  2
     x  = - --- + --- x + --- y - -- x ]
            13    26      26      26
```