CHAPITRE
II

Prise en main du système de calcul Sage

Sommaire

§ 1 Prise en main de l'environnement de calcul SAGE

L'objectif de ce chapitre est de découvrir le système de calcul Sage, en particulier sa syntaxe, ses mécanismes d'affectation et d'évaluation. Dans ce cours, nous utilisons Sage pour le calcul polynômial, nous présentons ici les principales opérations de Sage pour la manipulation de polynômes.

II.1.1. Le système de calcul Sage.— Sage est un système de calcul formel et numérique dont le développement a débuté en 2005 à l'université de Washington [1]. Sage est construit au dessus de systèmes libres déjà existants tels que MAXIMA et SYMPY pour le calcul symbolique, GAP pour la théorie des groupes, PARI pour la théorie des nombres, SINGULAR pour l'algèbre commutative, SCYPY pour le calcul numérique, R pour les statistiques. Sage a pour objectif de fournir une alternative libre aux systèmes propriétaires tels que MAPLE, MAGMA, MATLAB.

Un point fort, outre la mise en commun des potentialités de tous ces systèmes, est l'utilisation, au lieu d'une multitude de langages spécifiques, d'un langage informatique universel Python [2] comme langage fédérateur. Ainsi les structures mathématiques sont

1. `http://www.sagemath.org/`
2. `http://www.python.org/`

implémentées dans un cadre catégorique et orienté objets avec des méthodes pour les objets structurés et des méthodes pour leurs éléments. Les classes ainsi définies sont regroupées dans des modules Python.

Il existe plusieurs façons d'utiliser Sage : en « ligne de commande » , en écrivant des programmes interprétés ou compilés en Sage, en écrivant des scrits Python qui font appels à la bibiothèque Sage. Nous utiliserons ici Sage par l'intermédiaire d'une « interface graphique » permettant d'éditer des « feuilles de travail » .

II.1.2. Connexion au serveur Sage.— La connection au serveur Sage passe par une identification via la page à l'adresse `http://sage-math.univ-lyon1.fr`. Saisir cette url dans votre navigateur web, puis saisir votre login référérencé par l'annuaire ldap de l'UCBL. Après cette première identification, vous êtes connecté à un serveur Sage qui demande une identification : saisir le même login et le mot de passe associé. Vous accédez ainsi à une interface de gestion de « feuilles de travail » .

II.1.3. Première feuille de travail.— Pour ouvir une nouvelle feuille de travail, cliquer sur le lien New Worksheet. La feuille présente différentes fonctionnalités, en particulier :

- une zone de menus : File, Action, Data, sage, permettant de gérer la feuille de travail, en particulier une entrée du menu File permet d'enregister la feuille de travail dans un fichier (le serveur Sage de l'UCBL étant en phase expérimentale, il est conseillé d'enregistrer ses feuilles de travail à l'issue de la séance),
- trois boutons Save, Save & quit, Discard & quit, permettant d'enregistrer la feuille de travail et de quitter la feuille de travail en l'enregistrant ou non.
- des cellules pour la saisie des commandes.

Dans un premier temps, nous n'utiliserons pas les autres fonctionnalités disponibles.

II.1.4. Les cellules.— Les cellules permettent de saisir les instructions. Pour évaluer l'ensemble des instructions saisies dans une cellule, on peut cliquer sur le lien evaluate au-dessous de la cellule ou bien utiliser le racourci clavier `<MAJ><ENTREE>`.

Exercice 1. — Saisir l'expression suivante puis l'évaluer.

```
sage: 2+2
```

Il est possible d'évaluer séquentiellement toutes les cellules de la feuille de travail avec l'entrée Evaluate All du menu Action.

Pour insérer une cellule (entre deux cellules), cliquer sur la ligne bleu au-dessus ou au-dessous de la cellule ou bien faire `<CTRL><ENTREE>` au clavier. Pour supprimer une cellule, vider la cellule de son contenu puis `<DELETE>`.

II.1.5. Aide en ligne et complétion dynamique.— Une aide en ligne est disponible pour les commandes et les méthodes. Cette aide en ligne permet aussi d'obtenir une description des classes. Pour obtenir de l'aide sur une méthode ou une classe on exécute l'instruction `nomMethode?` ou `nomClasse?`.

Exercice 2. — Tester les instructions suivantes :

```
sage: cos?
sage: RationalField?
sage: PolynomialRing?
```

Pour écrire le nom d'une méthode ou d'une classe, on peut utiliser la complétion dynamique en utilisant la touche <TAB>.

§ 2 Variables et expressions

II.2.1. L'affectation.— Pour affecter à une *variable* une valeur, on utilise le symbole =, par exemple

```
sage: x = 2 + 2
sage: x
          4
sage: x = cos(pi/12)
sage: x
          1/12*(sqrt(3) + 3)*sqrt(6)
```

L'affectation d'une variable n'affiche pas la valeur de la variable sur la sortie standard. Tester :

```
sage: x = 1 ; x
        1
sage: y = 1
...   print y
        1
```

Pour un affichage plus visuel on peut utiliser la méthode `show()`. Saisir l'instruction suivante :

```
sage: x.show()
```

Pour afficher la valeur de variable `x` avec une approximation avec 20 chiffres décimaux :

```
sage: x.n(digits=20)
```

Il est possible de saisir des instructions sur plusieurs lignes, pour passer à la ligne, utiliser la touche <ENTRÉE> :

```
sage : x=2
...    y=x
...    print y
...    z=3 ; t = z+y ; z = z+t
...    print z
2
8
```

Pour réinitialiser les variables, on peut utiliser la méthode `reset()` :

```
sage : x=2
...    print x
...    reset()
...    print x
2
x
```

II.2.2. Expressions symboliques.— Sage permet de manipuler des expressions symboliques. La commande `var` permet de déclarer des variables symboliques.

Exercice 3. — Saisir les instructions suivantes puis les évaluer.

```
sage : var('x,y')
...    z = cos(x)^2 + sin(y)^2
...    print z
...    z = z(x=y)
...    print z
...    z.simplify_trig()
```

La méthode `subs.expr()`, ou la syntaxe `expression(variable = valeur)`, permet de substituer des valeurs dans une expression symbolique :

```
sage : var('x,y')
...    z = cos(x)^2 + sin(y)^2
...    print z.subs_expr(x==pi/2)
...    print z(y=pi/2)
```

II.2.3. Fonctions.— On peut définir des fonctions de la façon suivante :

```
sage : reset()
...    var('x,y')
...    f(x,y) = x/sin(x) + sqrt(y)
...    f
```

Dans Sage, toute expression mathématique est un objet. Un objet possède des attributs définissant sa structure et des méthodes permettant de modifier sa structure ou de communiquer avec lui. En particulier, les fonctions sont des objets :

```
sage : f.parent()
```

qui possèdent des méthodes d'accès, par exemple :

```
sage : f.show()
...    f.limit(x=0).show()
...    f.diff(x).show()
```

II.2.4. Les fonctions.— Pour définir une fonction avec Sage, on peut aussi utiliser la commande `def`. Par exemple, pour la fonction

$$x \longmapsto x^2$$

on peut procéder de la façon suivante :

```
sage: def carre(x):
...       return x^2
sage: carre(3)
9
sage: A = Matrix([[1,1],[1,1]])
sage: carre(A)
[2 2]
[2 2]
```

On peut définir des fonctions de plusieurs variables de la même façon. Par exemple, pour la fonction

$$(x,y) \longmapsto \cos^2(x) + \sin^2(y)$$

```
sage: def fonc(x,y):
          return cos(x)^2 + sin(y)^2
sage: fonc(pi/3,pi/2)
5/4
```

§ 3 Les polynômes à une indéterminée

II.3.1. Anneaux de polynômes à une indéterminée.— Pour construire un anneau de polynômes, on utilise le constructeur `PolynomialRing`. L'anneau $R = A[x]$ des polynômes à une indéterminée x à coefficients dans un anneau ou corps A peut se spécifier par les expressions équivalentes suivantes :

`R.<x> = A[ ],  R.<x> = PolynomialRing(A),`
`R = A['x'],  R = PolynomialRing(A, 'x').`

Par exemple, on utilise `ZZ['x']`, `QQ['x']`, `RR['x']` et `Integers(n)['x']` pour construire les anneaux $\mathbb{Z}[x]$, $\mathbb{Q}[x]$, $\mathbb{R}[x]$ et $\mathbb{Z}/n\mathbb{Z}[x]$.

Pour accéder à l'anneau de base A, on utilise la méthode `R.base_ring()`. On peut obtenir l'indéterminée avec la méthode `R.gen()` ou `R.0`.

II.3.2. Exemple.— Pour construire $\mathbb{Q}[x]$:

```
sage: R = PolynomialRing(QQ, 'x')
```

L'argument 'x' de `PolynomialRing` est une chaîne de caractères qui représente le nom de l'indéterminée de l'anneau ou du corps. On peut écrire aussi

```
sage: R = QQ['x']
sage: R
        Univariate Polynomial Ring in x over Rational Field
```

Pour récupérer l'indéterminée (avec le même nom) :

```
sage: x = R.gen()
```

La variable Python x représente alors le polynôme x de $\mathbb{Q}[x]$:

```
sage: x.parent()
Univariate Polynomial Ring in x over Rational Field
```

Ainsi, le polynôme x de $\mathbb{Q}[x]$ est différent du polynôme x de $\mathbb{R}[x]$ et du polynôme t de $\mathbb{Q}[t]$.

II.3.3. Construction de polynômes.— Un polynôme de $A[x]$ se définit simplement comme une expression algébrique en l'indéterminée x :

```
sage: R = QQ['x']
sage: f = 2*x^2 - 3*x +1
sage: f.parent()
        Univariate Polynomial Ring in x over Rational Field
```

Pour changer d'anneau de base de $A[x]$ en $B[x]$, on utilise la méthode `f.change_ring(B)`. Pour changer le nom de l'indéterminée, on utilise la méthode `change_variable_name` :

```
sage: g = f.change_variable_name('y')
sage: g
        2*y^2 - 3*y + 1
sage: h = g.change_ring(RR)
sage: h.parent()
        Univariate Polynomial Ring in y over Real Field with 53 bits of
            precision
```

II.3.4. Opérations d'accès sur les polynômes d'une seule indéterminée.— Les principales opérations d'accès permettent d'obtenir

- l'indéterminée x : `f.variable_name()`,
- le coefficient de x^k : `f[k]`,
- le coefficient dominant : `f.leading_coefficient()`,
- le degré : `f.degree()`,
- la liste des coefficients : `f.coeffs()`,
- la liste des coefficients non nuls : `f.coefficients()`.

II.3.5. Opérations arithmétiques élémentaires.— Les opérations arithmétiques élémentaires s'écrivent naturellement : l'addition `f + g`, la soustraction `f - g`, la multiplication `f * g` et la puissance `f^k`.

Il existe plusieurs syntaxes pour la substitution dans un polynôme `f` de l'indéterminée par une valeur a. On peut utiliser `f(a)` ou la méthode `subs` : `f.subs(a)`. Par exemple :

```
sage: R = QQ['x']
sage: f = x^2 + 3*x +1
sage: f(sqrt(3))
        (sqrt(3) + 3)*sqrt(3) + 1
sage: A = Matrix([[0,1],[1,0]])
sage: f(A)
        [2 3]
        [3 2]
```

Pour calculer la dérivée d'un polynôme, on a la méthode `f.derivative()` ou `diff(f)`. La méthode `random_element()` génère un polynôme au hasard.

```
sage: f = R.random_element(degree=6)
sage: f
                2*x^6 + 11*x^5 + 1/4*x^3 + 1/3*x^2 + 2*x - 2
sage: f.subs(x^3)
        2*x^18 + 11*x^15 + 1/4*x^9 + 1/3*x^6 + 2*x^3 - 2
sage: f.subs(2)
        1456/3
sage: f.derivative()
        12*x^5 + 55*x^4 + 3/4*x^2 + 2/3*x + 2
```

II.3.7. Arithmétique euclidienne.— L'anneau $\mathbb{K}[x]$ étant euclidien, les méthodes de division sont simples :

- le test de divisibilité de f par g : `f.divides(g)`,
- pour obtenir la multiplicité d'un diviseur g de f : `k = f.valuation(g)`,
- pour calculer la division euclidienne dans $\mathbb{K}[x]$, où $\mathbb{K}$ est un corps, $f = qg + r$:

```
sage: q, r = f.quo_rem(g)
sage: f = f//g
sage: r = f%g
```

- pour calculer le plus grand commun diviseur (pgcd) de polynômes de $\mathbb{K}[x]$, où $\mathbb{K}$ est un corps, on utilise la méthode `gcd` : `f.gcd(g)`, `gcd([f1, f2, f3])` :

```
sage: R.<x> = QQ[]
sage: f = 15*(x^12 - 3*x^5)*(x^2 -1)
sage: f.gcd(f.derivative())
      x^4
```

- pour calculer le plus petit commun multiple : `p.lcm(q)`, `lcm([p1, p2, p3])` :

```
sage: R.<x> = QQ[]
sage: f = x^5 -1
sage: g = x + 1
sage: f.lcm(g)
      x^6 + x^5 - x - 1
```

- la méthode `xgcd` permet de calculer une relation de Bézout :

$$g = \mathrm{pgcd}(f_1, f_2) = u_1 f_1 + u_2 f_2, \quad \text{où} \quad g, u_1, u_2, f_1, f_2 \in \mathbb{K}[x].$$

La syntaxe est la suivante `g, a, b = p.xgcd(q)` ou `xgcd(p, q)`. Par exemple,

```
sage: R.<x> = QQ[]
sage: f = x^7 - 1
sage: g = x^3 - 1
sage: f.xgcd(g)
```

Exercice 4. — On considère les polynômes

$$f_1 = x^7 - 3x^5 + 2x^4 - x^2 + 1, \quad f_2 = x^8 + 2x^6 - 3x^3 - x + 5.$$

1. Déterminer le degré, le coefficient dominant et la liste des coefficients du produit $f_1 f_2$.

2. Calculer la division euclidienne de f_2 par f_1.
3. Calculer le pgcd de f_1 et f_2.
4. Calculer $f_1(1)$.

II.3.8. Racines d'un polynôme.— Il existe plusieurs méthodes pour calculer les racines d'un polynôme. La méthode `roots` d'un polynôme retourne les racines du polynôme dans son anneau de base, sous la forme d'une liste de couples (racine, multiplicité) :

```
sage: R.<x> = QQ[]
sage: f = (x-1)*(x^2-1)*(x^2+1)*(x^2 - 5)
sage: f.roots(QQ)
      [(-1, 1), (1, 2)]
```

Il est possible de spécifier le domaine, par exemple pour obtenir les racines réelles, puis les racines complexes :

```
sage: f.roots(RR)
      [(-2.23606797749979, 1), (-1.00000000000000, 1),
        (1.00000000000000, 2), (2.23606797749979, 1)]
sage: f.roots(CC)
      [(-2.23606797749979, 1), (-1.00000000000000, 1),
       (1.00000000000000, 2), (2.23606797749979, 1),
       (-5.44339946922833e-36 - 1.00000000000000*I, 1),
       (-5.44339946922833e-36 + 1.00000000000000*I, 1)]
```

II.3.9. Idéaux de $A[x]$.— Les idéaux d'anneaux de polynômes sont des objets construits à partir de la méthode `ideal` :

```
sage: I = R.ideal(x^2 + 1)
sage: I
        Principal ideal (x^2 + 1) of Univariate Polynomial Ring in x over
        Rational Field
```

ou bien avec la syntaxe suivante, pour construire l'idéal I de $\mathbb{Q}[x]$ engendré par les deux polynômes $f_1 = x^2 - 1$ et $f_2 = x^3 - x^2 + x - 1$:

```
sage: R.<x> = QQ[]
sage: I = (x^2 - 1, x^3 - x^2 + x - 1)*R
sage: I
        Principal ideal (x - 1) of Univariate Polynomial Ring in x
        over Rational Field
```

Pour réduire un polynôme modulo un idéal :

```
sage: I = R.ideal((x^2 - 1)*(x+1))
sage: g = I.reduce(x^4 + 1) ; g
        2*x^2
```

Exercice 5. — Soit I l'idéal de $\mathbb{Q}[x]$ engendré par les polynômes

$$f_1 = x^6 - 1, \quad f_2 = x^4 + 2x^3 + 2x^2 - 2x - 3.$$

1. Déterminer un générateur g de I tel que $I = \langle g \rangle$.
2. Montrer que le polynôme $f = x^4 + 2x^2 - 3$ est dans I.
3. Décomposer f en une combinaison algébrique de f_1 et f_2.

§ 4 Polynômes à plusieurs indéterminées

Sage permet de calculer avec les polynômes à plusieurs indéterminées. La différence fondamentale avec le cas à une seule indéterminée est que l'anneau $\mathbb{K}[x_1,\dots,x_n]$ n'est pas principal, cf. exercice 1 du chapitre IV. Pour l'arithmétique euclidienne dans ces anneaux, les bases de Gröbner sont alors un outils précieux.

II.4.1. Anneaux de polynômes à plusieurs indéterminées.— Pour construire l'anneau des polynômes à plusieurs indéterminées, on utilise le constructeur `PolynomialRing`. Pour construire l'anneau $A[x,y]$, on utilise `PolynomialRing(A, 'x,y')` ou `A['x,y']`. Par exemple, pour l'anneau $\mathbb{Q}[x,y,z]$:

```
sage: R = PolynomialRing(QQ, 'x,y,z')
sage: x, y, z = R.gens()
```

On peut aussi utiliser la syntaxe équivalente : `R.<x,y,z> = QQ[]`.

II.4.2. Construction d'anneaux de polynômes.— Il est possible de construire des indéterminées d'un même nom indicé par des entiers naturels, comme par exemple $x_0,x_1,\dots,x_n$ ou $y_0,y_1,\dots,y_k$. Pour construire l'anneau $A[x_0,\dots,x_{n-1}]$, on utilise le constructeur `PolynomialRing(A, 'x', n)`.

Par exemple pour construire l'anneau $\mathbb{Q}[x_0,x_1,x_2,x_3,x_4,x_5]$:

```
sage: R = PolynomialRing(QQ, 'x', 5)
sage: x = R.gens()
sage: sum(x[i] for i in xrange(5))
        x0 + x1 + x2 + x3 + x4
```

La méthode `R.gens()` retourne le n-uplet des indéterminées. On peut aussi obtenir le 1er, 2e, ... générateur avec `R.0`, `R.1`, ...

Pour l'évaluation, il faut donner une valeur à chacune des indéterminées ou préciser les indéterminées à substituer :

```
sage: R.<x,y,z> = QQ[]
sage: f = 2*x^2*y*z^2 + 3*x*y^2*z - 4*z^2
sage: f(1,0,2)
        -16
sage: g = f.subs(x=1, z=y^2-1) ; g
        2*y^5 - y^4 - 4*y^3 + 5*y^2 + 2*y - 4
sage: g.parent()
        Multivariate Polynomial Ring in x, y, z over Rational Field
```

Avec le constructeur `PolynomialRing`, on peut spécifier un ordre monomial sur un anneau de polynômes à plusieurs indéterminées :

```
sage: R.<x,y,z> = PolynomialRing(QQ, 'x,y,z', order='lex')
sage: x > y^2
        True
sage: x^2 *y* z > x * y
        True
sage: x^2 *y* z > x^3 * y
        False
```

Dans cet exemple, on a spécifié l'ordre lexicographique, `lex`, induit par $z < y < x$. On peut spécifier l'ordre lexicographique "inverse", `invlex`, induit par $x < y < z$.

```
sage: R.<x,y,z> = PolynomialRing(QQ, 'x,y,z', order='invlex')
sage: x * y > z
        False
sage: x^2 *y* z > x * y
        True
sage: x^2 *y* z > x^3 * y
        True
```

Il existe également l'ordre lexicographique gradué, `deglex` (voir définition à la partie IV.4.8).

```
sage: R.<x,y,z> = PolynomialRing(QQ, 'x,y,z', order='deglex')
sage: z^3 > x^2
        True
sage: x^2 *y* z > x * y * z^2
        True
sage: x^2 *y* z > x^3 * y
        False
```

L'ordre spécifié par défaut est l'ordre lexicographique inverse gradué, voir définition sur la page `http://www.sagemath.org/doc/reference/sage/rings/polynomial/term_order.html`.

II.4.3. Méthodes d'accès.— Les principales opérations d'accès permettent d'obtenir
- le support : `f.exponents()`,
- les coefficients non nuls `f.coefficients()`,
- le degré total : `f.degree()`,
- le degré en x : `f.degree(x)`.

```
sage: R.<x,y,z> = QQ[]
sage: f = 2*x^2*y*z^2 + 3*x*y^2*z - 4*z^2
sage: print f.degree() , f.degree(x)
        5 2
```

On peut utiliser l'opérateur crochet `[]` pour extraire des coefficients, il accepte comme paramètre un monôme ou son *exposant* :

```
sage: f[2,1,2] , f[1,2,1], f[0,0,2]
        (2, 3, -4)
sage: f[x^2*y*z^2]
        2
```

Exercice 6. — Soit $f = 5x^2y^3z^2 + 3xy^2z + 4z^2 - 2xy + zy + z$ un polynôme de $\mathbb{Q}[x,y,z]$.

1. Avec Sage, donner le terme dominant, le coefficient dominant et le monôme dominant de f pour l'ordre lexicographique et pour l'ordre lexicographique gradué.
2. Donner le degré total de f,
3. Écrire f comme un polynôme en x.

II.4.4. Opérations arithmétiques sur le polynômes.— Pour les polynômes à plusieurs indéterminées, les opérations arithmétiques +, -, * s'utilisent comme dans le cas d'une seule indéterminée. Comme $\mathbb{K}[x_1,\ldots,x_n]$ n'est pas principal, les méthodes basées sur la division euclidienne des polynômes ne fonctionnent pas dans ce cas.

Exercice 7.— Soient $f = x^3y^3 + 2y^2$, $f_1 = 2xy^2 + 3x + 4y^2$ et $f_2 = y^2 - 2y - 2$ des polynômes de $\mathbb{Q}[x,y]$. En utilisant l'ordre lexicographique induit par $y < x$, calculer le reste de la division de f par f_1 puis par f_2. Calculer ensuite le reste de la division de f par f_2, puis f_1.

Exercice 8.— Soit $f = x^2y^2 - w^2$, $f_1 = x - y^2w$, $f_2 = y - zw$, $f_3 = z - w^3$ et $f_4 = w^3 - w$ des polynômes de $\mathbb{Q}[x,y,z,w]$. En utilisant l'ordre lexicographique induit par $w < z < y < x$, calculer le reste de la division de f par f_1, f_2, f_3, f_4 dans cet ordre. Faire le même calcul avec l'ordre f_4, f_3, f_2, f_1. Faire les mêmes calculs avec $f = x^2y^2 + w^3$.

II.4.5. Calcul de bases de Gröbner.— La méthode `I.groebner_basis()` retourne une base de Gröbner réduite de l'idéal I. Une base est de Gröbner G d'un idéal I est dite *réduite* si

i) $\mathrm{lc}(g) = 1$, pour tout $g \in G$,

ii) pour tout $g \in G$, aucun monôme de p est dans l'idéal $\langle \mathrm{lt}(G - \{g\})\rangle$.

On peut montrer, un ordre monomial étant fixé, que tout idéal non nul possède une unique base de Gröbner réduite.

```
sage: R.<x,y,z> = PolynomialRing(RationalField(), 3, order='deglex')
sage: f1 = x*y-y^2
sage: f2 = x^2-z^2
sage: I = (f1, f2)*R
sage: G = I.groebner_basis()
sage: print G
        [y^3 - y*z^2, x^2 - z^2, x*y - y^2]
```

Avec l'argument `toy:buchberger`, la base de l'idéal est complétée en une base de Gröbner sans être réduite :

```
sage: R.<x,y,z> = PolynomialRing(RationalField(), 3, order='deglex')
sage: f1 = x^2 + y^2 - 1
sage: f2 = x^2 + z^2 - 1
sage: I = (f1, f2)*R
sage: G = I.groebner_basis()
sage: H = I.groebner_basis('toy:buchberger')
sage : print G
         [x^2 + z^2 - 1, y^2 - z^2]
sage: print H
         [x^2 + y^2 - 1, x^2 + z^2 - 1, y^2 - z^2]
```

Exercice 9.— Pour les idéaux suivants, construire une base de Gröbner en utilisant l'ordre lexicographique, puis l'ordre lexicographique gradué.

1. $I = \langle x^2y - 1, xy^2 - x\rangle$,
2. $I = \langle x^2 + y, x^4 + 2x^2y + y^2 + 3\rangle$,
3. $I = \langle x - z^4, y - z^5\rangle$.

Exercice 10. — Montrer que les polynômes f_1 et f_2 de l'exercice 7 ne forment pas une base de Gröbner de l'idéal qu'ils engendrent pour l'ordre lexicographique induit par $y < x$.

Exercice 11. — Montrer que les polynômes f_1, f_2, f_3, f_4 de l'exercice 8 ne forment pas une base de Gröbner de l'idéal qu'ils engendrent pour l'ordre lexicographique induit par $z < y < x < w$. Existe-t-il un ordre lexicographique pour lequel cette famille forme une base de Gröbner ?

Exercice 12. — Calculer les S-polynômes des couples (f_1, f_2) de polynômes de $\mathbb{Q}[x,y,z]$ suivants avec l'ordre lexicographique et l'ordre lexicographique gradué induits par $z < y < x$:

1. $f_1 = 3x^2yz - y^3z^3$, $f_2 = xy^2 + z^2$,
2. $f_1 = 3x^2yz - xy^3$, $f_2 = xy^2 + z^2$,
3. $f_1 = 3x^2y - yz$, $f_2 = xy^2 + z^4$.

II.4.6. Problème de l'appartenance à un idéal.—

Exercice 13. — Soit I l'idéal de $\mathbb{Q}[x,y,z]$ défini par

$$I = \left\langle y - x^3, x^2y - z \right\rangle.$$

1. Calculer une base de Gröbner de I pour l'ordre lexicographique induit par $z < y < x$.
2. Le polynôme $f = xy^3 - z^2 + y^5 - z^3$ appartient-il à I ?

Exercice 14. — Les polynôme $x^3 + 1$ et $x^3 - 1$ s'écrivent-ils comme combinaison algébrique des polynômes $x + y + z$, $xy + yz + zx$ et $xyz - 1$?

Exercice 15. — On considère dans $\mathbb{Q}[x,y,z]$ les idéaux

$$I = \left\langle x^2 + z, xy + y^2 + z, xz - y^3 - 2yz, y^4 + 3y^2z + z^2 \right\rangle,$$
$$J = \left\langle x^2 + z, xy + y^2 + z, x^3 - yz \right\rangle.$$

A-t-on $I \subset J$, $J \subset I$ ou $I = J$?

II.4.7. Résolution de systèmes d'équations polynomiales.—

Exercice 16. — Soit $I = \langle f_1, f_2 \rangle$ l'idéal de $\mathbb{R}[x,y]$ engendré par les polynômes

$$f_1 = x^2 + 2y^2 - 1,$$
$$f_2 = x^2 + xy + y^2 - 1.$$

1. Déterminer une famille de générateurs de l'idéal $I \cap \mathbb{R}[x]$ et de l'idéal $I \cap \mathbb{R}[y]$.
2. Déterminer les solutions du système d'équations suivant

$$\left| \begin{array}{ll} f_1(x,y) & = 0, \\ f_2(x,y) & = 0. \end{array} \right.$$

Exercice 17. — Pour $a = 1, 2, 3$, déterminer toutes les solutions dans $\mathbb{Q}$ du système

$$\left| \begin{array}{rl} x^2 + 2y^2 & = a, \\ x^2 + xy + y^2 & = a. \end{array} \right.$$

Exercice 18. — Déterminer une base des idéaux d'élimination I_1 et I_2 pour l'idéal I engendré par les équations

$$\left| \begin{array}{rl} x^2 + y^2 + z^2 & = 4, \\ x^2 + 2y^2 & = 5, \\ xz & = 1. \end{array} \right.$$

Déterminer les solutions dans $\mathbb{Q}^3$ de ce système.

Exercice 19. — On considère les équations

$$\left| \begin{array}{rl} t^2 + x^2 + y^2 + z^2 & = 0, \\ t^2 + 2x^2 - xy - z^2 & = 0, \\ t + y^3 - z^3 & = 0. \end{array} \right.$$

Soit I l'idéal engendré par ces équations.

1. Calculer une base de Gröbner de I pour l'ordre lexicographique induit par $z < y < x < t$.
2. Calculer une base de Gröbner de l'idéal $I \cap \mathbb{Q}[x, y, z]$.
3. Calculer une base de Gröbner de $I \cap \mathbb{Q}[x, y, z]$ avec l'ordre monomial `degrevlex`.

II.4.8. Recherche d'extrema et de points critiques.—

Exercice 20. — Déterminer les extréma de la fonction réelle

$$f = x^2 + y^2 + xy,$$

soumis à la contrainte $x^2 + 2y^2 = 1$.

Exercice 21. — Montrer que la fonction réelle

$$f = (x^2 + y^2)(x^2 + y^2 - 1)z + z^3 + x + y$$

ne possède pas de point critique.