ANNEXE
A

Prise en main du système de calcul Sage

Sommaire

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

L'objectif de cette annexe 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 polynomial, nous présentons ici les principales opérations de Sage pour la manipulation de polynômes.

A.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 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.

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

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 bibliothèque Sage. Nous utiliserons ici Sage par l'intermédiaire d'une « interface graphique » permettant d'éditer des « feuilles de travail » .

A.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 » .

A.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.

A.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 127. — 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>`.

A.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 128. — 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

A.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 = sqrt(2)
...   y = 3
...   y
      3

sage: y
...   print x
...   y
      sqrt(2)
      3
```

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

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

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
```

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

Exercice 129. — 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)
```

A.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()
```

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
```


A.2.4. Remarque.— Tous les types d'objets ne possèdent pas la méthode `show()`. Tester la méthode `show()` sur une matrice :

```
sage: A = Matrix([[1,2],[3,5]])
...   A.show()
```

Tester ensuite la fonction `show(-)` :

```
sage: show(A)
```

§ 3 Les polynômes à une indéterminée

A.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 l'expression suivante :

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

Pour construire les anneaux $\mathbb{Z}[x]$, $\mathbb{Q}[x]$, $\mathbb{R}[x]$ et $\mathbb{Z}/n\mathbb{Z}[x]$, on utilise respectivement les expressions `R.<x> = PolynomialRing(ZZ, 'x')`, `R.<x> = PolynomialRing(QQ, 'x')`, `R.<x> = PolynomialRing(RR, 'x')`, `R.<x> = PolynomialRing(Integers(n), 'x')`.

Par exemple, pour construire $\mathbb{Q}[x]$:

```
sage: R.<x> = PolynomialRing(QQ, 'x')
sage: R
        Univariate Polynomial Ring in x over Rational Field
```

L'argument `QQ` de `PolynomialRing` est ici le corps de base $\mathbb{Q}$. Pour accéder à l'anneau (ou au corps) de base de `R`, on peut utiliser la méthode `R.base_ring()`.

```
sage: R.base_ring()
        Rational Field
```

L'argument `'x'` de `PolynomialRing` est une chaîne de caractères qui représente le nom de l'indéterminée de l'anneau de polynômes.

L'expression x dans `R.<x>` est une variable Python ayant pour valeur l'indéterminée de l'anneau de polynômes `R`.

La variable x est donc égale au 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 $\mathbb{Q}[t]$.

A.3.2. Construction de polynômes.— Un polynôme de $A[x]$ se définit simplement comme une expression algébrique en l'indéterminée x :

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

On peut générer un polynôme au hasard avec la méthode `random_element()`.

```
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
```

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

```
sage: g = f.change_ring(RR)
sage: g.parent()
        Univariate Polynomial Ring in x over Real Field with 53 bits of
precision
```

A.3.3. Évalutation d'un polynôme et substitution dans un polynôme de l'indéterminée.— Pour évaluer le polynôme f en la valeur a on peut utiliser l'instruction `f(a)` :

```
sage: reset()
sage: R.<x>= PolynomialRing(QQ, 'x')
sage: f = x^2 + 3*x +1
sage: f(sqrt(3))
        (sqrt(3) + 3)*sqrt(3) + 1
sage: f(sqrt(3)).expand()
        3*sqrt(3)+4
```

On peut également substituer l'indéterminée par une expression :

```
sage: f(x^2+1)
        x^4 + 5*x^2 + 5
```

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

- le nom de l'indéterminée : `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()`.

A.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`.

```
sage: reset()
sage: R.<x>= PolynomialRing(QQ, 'x')
sage: f = x^3 + 3*x^2 +1
sage: g = x^2 +1
sage : f^2*g+ 3*f-g^4
        6*x^7 + 6*x^6 + 8*x^5 + 9*x^4 + 5*x^3 + 12*x^2 + 3
```

On peut également calculer la dérivée d'un polynôme avec la méthode `f.derivative()`.

```
sage : f.derivative()
        3*x^2 + 6*x
```


A.3.6. Arithmétique euclidienne.— L'anneau $\mathbb{K}[x]$ étant euclidien, les méthodes de division sont simples :

- le test de divisibilité de f par g : `g.divides(f)`,
- 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: q
        x + 3
sage: r
        -x - 2
sage: q = f//g
        x + 3
sage: r = f%g
        -x - 2
```

- 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: reset()
sage: R.<x>= PolynomialRing(QQ, 'x')
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: 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, u1, u2 = f1.xgcd(f2)` ou `xgcd(f1, f2)`. Par exemple,

```
sage: f1 = x^7 - 1
sage: f2 = x^3 - 1
sage: f1.xgcd(f2)
        (x - 1, 1, -x^4 - x)
sage: 1*f1+(-x^4-x)*f2
        x - 1
```

Exercice 130. — 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 $f_1(1)$.

A.3.7. 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: reset()
sage: R.<x>= PolynomialRing(QQ, 'x')
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)]
```

A.3.8. Idéaux de $A[x]$.— Les idéaux d'anneaux de polynômes sont des objets construits à partir de la méthode `ideal` :

```
sage: reset()
sage: R.<x>= PolynomialRing(QQ, 'x')
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: 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 h modulo un idéal I, on peut utiliser la méthode `I.reduce(h)` :

```
sage: I = R.ideal(x^5+x^4+x^3+x^2+x+1, x^4-x^3+x-1)
sage: I.reduce(x^4 + 1)
        -x + 1

sage: I
        Principal ideal (x^3 + 1) of Univariate Polynomial Ring in x over
        Rational Field
sage: f = x^3+1
sage: g = x^4+1
sage: g%f
        -x + 1
```

Exercice 131. — 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 $\mathbb{K}[x_1, \ldots, x_n]$ n'est pas principal, cf. exercice 59 du chapitre III. Pour l'arithmétique euclidienne dans ces anneaux, les bases de Gröbner sont alors un outils précieux.

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

```
sage: R.<x,y,z> = PolynomialRing(QQ, 'x,y,z')
sage: R
        Multivariate Polynomial Ring in x, y, z over Rational Field
```

Les variables Python `x,y,z` ont pour valeur les trois indéterminées et sont donc égales respectivement aux monômes x, y et z de l'anneau $\mathbb{Q}[x, y, z]$.

Il est également possible de construire l'anneau des polynômesdes à plusieurs indéterminées ayant un même nom indicé par des entiers naturels, comme par exemple $x_0, x_1, \ldots, x_n$ ou $y_0, y_1, \ldots, y_k$. Pour construire l'anneau $A[x_0, \ldots, x_{n-1}]$, on utilise le constructeur `PolynomialRing(A, 'x', n)`.

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

```
sage: reset()
sage: R = PolynomialRing(QQ, 'x', 5)
sage: R
        Multivariate Polynomial Ring in x0, x1, x2, x3, x4 over Rational Field
```

Attention, dans le code précédent on n'a pas encore introduit de variables Python pour les indéterminées :

```
sage: f = x0+x2*x3
        NameError: name 'x0' is not defined
```

Il est donc nécessaire de récupérer le n-uplet des indéterminées en utilisant la méthode `R.gens()` :

```
sage: reset()
sage: R = PolynomialRing(QQ, 'x', 5)
sage: x = R.gens()
```

Les variables Python `x[0], x[1], ..., x[4]` ont alors respectivement pour valeurs x_0, x_1, ..., x_4.

```
sage: f= x[0]+x[2]*x[3]
sage: f
        x2*x3 + x0
sage: g = sum(x[i]^i for i in xrange(5))
sage: g
        x4^4 + x3^3 + x2^2 + x1 + 1
```

A.4.2. Évalutation et substitution.— Pour l'évaluation d'un polynôme f, il faut donner une valeur à chacune des indéterminées ou préciser les indéterminées à substituer. Dans le second cas, on utilise la méthode `f.subs()`.

```
sage: reset()
sage: R.<x,y,z> = PolynomialRing(QQ, 'x,y,z')
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

sage: reset()
sage: R = PolynomialRing(QQ, 'x', 6)
sage: x = R.gens()
sage: f = sum((i+1)^2*x[i]*x[i+1]^i for i in xrange(5))
sage: f
        25*x4*x5^4 + 16*x3*x4^3 + 9*x2*x3^2 + 4*x1*x2 + x0
sage: g = f.subs({x[i] : -i for i in xrange(5)}) ; g
        -100*x5^4 + 2918
```

A.4.3. Ordre monomial sur l'anneau de polynômes.— 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 III.4.9).

```
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 `degrevlex` :

```
sage: R.<x,y,z> = PolynomialRing(QQ, 'x,y,z')
sage: R.term_order()
        Degree reverse lexicographic term order
```

La définition de cet ordre se trouve sur la page `http://www.sagemath.org/doc/reference/sage/rings/polynomial/term_order.html`.

A.4.4. 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)`,
- le terme dominant : `f.lt()`,
- le coefficient dominant : `f.lc()`,
- le monôme dominant : `f.lm()`.

```
sage: R.<x,y,z> = PolynomialRing(QQ, 'x,y,z')
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 132.— 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.

A.4.5. Opérations arithmétiques sur les 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. En particulier, il ne faut pas utiliser les commandes `f//g` et `f%g` qui ne tiennent pas compte de l'ordre monomial spécifié. On peut par contre utiliser la méthode `divides()` et la méthode quotient `//` sur des termes en plusieurs variables.

Pour réduire le terme dominant de f modulo g suivant l'ordre monomial spécifié, vous pouvez utiliser la méthode `f.mod(g)`. Cette méthode ne réduit successivement que le terme dominant et s'arrête avec un reste ayant un terme dominant non divisible par celui de g.

Par exemple :

```
sage: reset()
sage: R.<x,y,z> = PolynomialRing(QQ, 'x,y,z', order ='lex')
sage: f = x^2*y^2*z - x^2*y^3+ x*y^2*z + y^2*z^2 +y; f
        x^2*y^2*z - x^2*y^2 + x*y^2*z + y^2*z^2 + y
sage: g = y^2 -y*z; g
        y^2 - y*z
sage: p= f.mod(g); p
        x^2*y*z^2 - x^2*y*z + x*y^2*z + y^2*z^2 + y
```

Les termes xy^2z et y^2z^2 de p sont encore divisibles par le terme dominant de g mais ils ne sont pas dominants. Si l'on veut obtenir le reste de la division de f par g, il faudra alors mettre de coté successivement les termes dominants :

```
sage: r=p.lt()
sage: p = p - p.lt()
sage: p = p.mod(g); p
        -x^2*y*z + x*y^2*z + y^2*z^2 + y
```

```
sage: r=r+p.lt(); p = p - p.lt(); p = p.mod(g); p
        x*y*z^2 + y^2*z^2 + y
sage: r=r+p.lt(); p = p - p.lt(); p = p.mod(g); p
        y*z^3 + y
sage: r = r+p; r
        x^2*y*z^2 - x^2*y*z + x*y*z^2 + y*z^3 + y
```

Exercice 133. — 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, f_2 comme décrit dans l'algorithme de division III.5.3. Calculer ensuite le reste de la division de f par f_2, f_1.

Exercice 134. — 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 $f = x^2y^2 + w^3$.

Exercice 135. — Écrire une méthode qui prend en argument un polynôme f et des polynômes f_1, f_2, ..., f_s et qui retourne les polynômes u_1, ..., u_s et r de l'algorithme de division de f par f_1, ..., f_s (cf Algorithme III.5.3).

On pourra compléter le code suivant (F représente la liste des polynômes f_1, ..., f_s) :

```
sage: def algodivision(f,F):
          U = [0 for i in [0..(len(F)-1)]]
          r = 0; p = f
          ...
          return [U,r]
```

Vérifier à l'aide de cette méthode vos résultats dans les deux exercices précédents.

A.4.6. 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 n'est dans l'idéal $\langle \mathrm{lt}(G - \{g\})\rangle$.

Pour un ordre monomial fixé, tout idéal non nul possède une unique base de Gröbner réduite d'après la proposition V.7.

```
sage: R.<x,y,z> = PolynomialRing(QQ, 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(QQ, 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 136. — 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 137. — Montrer que les polynômes f_1 et f_2 de l'exercice 133 ne forment pas une base de Gröbner de l'idéal qu'ils engendrent pour l'ordre lexicographique induit par $y < x$.

Exercice 138. — Montrer que les polynômes f_1, f_2, f_3, f_4 de l'exercice 134 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 139. — 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. $f_1 = 3x^2y - yz$, $f_2 = xy^2 + z^4$.

Exercice 140. — Écrire une méthode qui prend en argument une liste F de polynômes et qui retourne une base de Gröbner de l'idéal engendré par F, à l'aide de l'algorithme de Buchberger (cf Algorithme V.3.3).

On pourra compléter le code suivant et utiliser la méthode `algodivision` de l'exercice 135 :

```
sage: def algoBuchberger(F):
        G = copy(F)
        B = [[F[i],F[j]] for i in [0..(len(G)-1)] for j in [(i+1)..(len(G)-1)]]
        while (len(B)>0):
            f,g = B.pop(0) #récupère et supprime le premier couple de B
            ....
    return G
```

Exercice 141. — Écrire une méthode qui prend en argument une base de Gröbner et retourne une base de Gröbner réduite. On pourra utiliser l'algorithme décrit dans l'exercice 100. (La méthode `sort()` permet de trier une liste.)

A.4.7. Problème de l'appartenance à un idéal.—

Exercice 142. — Soit I l'idéal de $\mathbb{Q}[x,y,z]$ défini par

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

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

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

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

A.4.8. Résolution de systèmes d'équations polynomiales.—

Exercice 145. — 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

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

Exercice 146. — 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 147. — 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 148. — 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`.

A.4.9. Recherche d'extrema et de points critiques.—

Exercice 149. — Déterminer les extrema de la fonction réelle

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

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

Exercice 150. — 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.

Bibliographie

[1] William W. Adams, Philippe Loustaunau, *An introduction to Gröbner bases*, American mathematical society, (1994)

[2] Bruno Buchberger, Franz Winker, *Ideals, Varieties and Algorithms*, Cambridge University Press (1998)

[3] David Cox, John Little, Donal O'Shea, *Ideals, Varieties and Algorithms*, Third Edition, Springer (2007)

[4] Daniel Perrin, *Géométrie algébrique. Une introduction*, InterEditions et CNRS Editions (1995), Chapitre I : Ensembles algébriques affines