Bases de Gröbner et algorithme de Buchberger

Marcelo Domingues & Samuel Gallay

Groupes de lecture

19 octobre 2022

Rappels utiles ?

Benjamin & Eliot : Ordres monomiaux

- Définitions, exemples...

Matthias & Victor : Algorithme de division

- $f = a_1 f_1 + ... + a_s f_s + r$
- Reste dépendant de l'ordre des diviseurs

Marine & Ludovic : Idéaux monomiaux

- $I = \langle x^\alpha : \alpha \in A \rangle$
- Lemme de Dickson : $I = \langle x^{\alpha(1)}, ..., x^{\alpha(s)} \rangle$

Farid & Ilan : Bases de Gröbner

- $\langle \mathrm{LT}(I) \rangle = \langle \mathrm{LT}(g_1), ..., \mathrm{LT}(g_s) \rangle$
- Théorème de la base de Hilbert : $I = \langle g_1, ..., g_s \rangle$
- Chaînes ascendantes d'idéaux

Alice & Clara : S-polynômes

- G est une base de Gröbner de $I = \langle G \rangle$ ssi $\forall i, j,\ \overline{S(g_i, g_j)}^{G} = 0$

Algorithme de Buchberger

Algorithme 1 : Algorithme de Buchberger

Entrées : $F = (f_1, ..., f_s)$

Sorties : Base de Gröbner $G = (g_1, ..., g_t)$ de $I = \langle f_1, ..., f_s \rangle$

1 $G := F$
2 $G' := \emptyset$ **tant que** $G \neq G'$ **faire**
3 $\quad G' := G$
4 $\quad$ **pour** $\{p, q\} \subset G'$ tq $p \neq q$ **faire**
5 $\quad\quad S :=$ reste de la division de $S(p, q)$ par G'
6 $\quad\quad$ **si** $S \neq 0$ **alors**
7 $\quad\quad\quad G := G \cup S$
8 $\quad\quad$ **fin**
9 $\quad$ **fin**
10 **fin**
11 **retourner** G

Preuves de terminaison et de correction

```
G := F

RÉPÉTER

  G' := G

  POUR  p ≠ q dans G' FAIRE
    S := reste de S(p, q) par G'
    SI S ≠ 0 ALORS G := G ∪ {S}

JUSQU'À CE QUE G = G'
```

Lemme 1

À chaque étape de l'algorithme, $G \subset I$.

Correction

Si l'algorithme termine, alors G est une base de Gröbner de I.

Lemme 2

Si $G \neq G'$, alors $\langle \text{LT}(G') \rangle \subsetneq \langle \text{LT}(G) \rangle$.

Terminaison

L'algorithme termine.

Lien avec le cas univarié

Dans $k[X]$, soit $I = \langle P_1, P_2 \rangle$.

Par le théorème de Bézout, $I = \langle \text{PGCD}(P_1, P_2) \rangle$. C'est une base de Gröbner.

Un lien entre l'algorithme d'Euclide et celui de Bucherger ?

Bases minimales et réduites

Lemme 3

Soit G une base de Gröbner de I et $p \in G$. Si $\mathrm{LT}(p) \in \langle \mathrm{LT}(G - \{p\}) \rangle$, alors $G - \{p\}$ est une base de Gröbner de I.

Bases minimales

Une base de Gröbner G est dite minimale si $\forall p \in G, \mathrm{LT}(p) \notin \langle \mathrm{LT}(G - \{p\}) \rangle$.

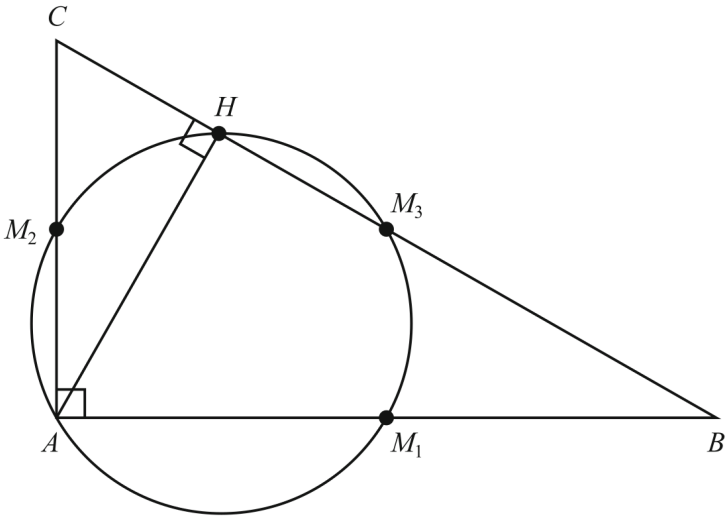
Bases réduites

Une base de Gröbner G est dite réduite si $\forall p \in G$, $\mathrm{LC}(p) = 1$, et aucun des monomes de p n'appartient à $\langle \mathrm{LT}(G - \{p\}) \rangle$.

Existence et unicité

Étant donné un idéal I, I possède une unique base réduite.

Un peu de géométrie



Un peu de géométrie

```
R.<x1,x2,x3,x4,x5,x6,x7,x8> = PolynomialRing(QQ, order='degrevlex')

H1 = 2*x1 - u1                                  # Milieux des côtés
H2 = 2*x2 - u2
H3 = 2*x3 - u1
H4 = 2*x4 - u2
H5 = u2*x5 + u1*x6 - u1*u2                      # H appartient à (BC)
H6 = u1*x5 - u2*x6                             # (AH) orthogonal à (BC)
H7 = (x1-x7)^2 + x8^2 - x7^2 - (x8-x2)^2        # OM_1 = OM_2
H8 = (x1-x7)^2 + x8^2 - (x3-x7)^2 - (x4-x8)^2 # OM_1 = OM_3

H = [H1, H2, H3, H4, H5, H6, H7, H8]
Q1 = (x5-x7)^2 + (x6-x8)^2 - (x1-x7)^2 - x8^2 # OH = OM_1
remainder(Q1, buchberger(H))
```

Algorithme de division

```
def divide(a, b):
  p = a; s = len(b); r = 0; q = [0] * s
  while (p != 0):
    isDivided = False
    for i in range(s):
      if R.monomial_divides(b[i], p.lt()) and not isDivided:
        q[i] += R.monomial_quotient(p.lt(), b[i], coeff=True)
        p -= R.monomial_quotient(p.lt(), b[i], coeff=True) * b[i]
        isDivided = True
    if not isDivided :
      r += p.lt()
      p -= p.lt()
  return (q, r)
```

S-polynômes

```
def remainder(a, b):
    return divide(a, b)[1]
def S_polynomial(f1, f2):
    lcm = R.monomial_lcm(f1.lt(), f2.lt())
    return R.monomial_quotient(lcm, f1.lt(), coeff=True)*f1 -
     ↪ R.monomial_quotient(lcm, f2.lt(), coeff=True)*f2
def is_gröbner_basis(G):
    for i in range(len(G)):
        for j in range(i+1, len(G)):
            if remainder (S_polynomial(G[i], G[j]), G) != 0 :
                return False
    return True
```

Algorithme de Buchberger

```
def buchberger(F):
    G = F.copy()
    is_gröbner = False
    while not is_gröbner:
        is_gröbner = True
        old_G = G.copy()
        for i in range(len(old_G)):
            for j in range(i+1, len(old_G)):
               S = remainder(S_polynomial(old_G[i], old_G[j]), old_G)
                if S != 0:
                    is_gröbner = False
                    G.append(S)
    assert is_gröbner_basis(G)
    return G
```