# Remarks about complexity and implementation of bivariate Gröbner bases

Xavier Dahan[0001-6042-6132]

dahan.xavier.a2@tohoku.ac.jp
Tohoku University
980-8576, Sendai, Japan

**Abstract.** Let $a, b \in \mathbb{K}[x, y]$ be two polynomials and $T = P^k \in \mathbb{K}[x]$ the power of an irreducible one. We analyze the complexity of a simplified version of the algorithm in [2], which computes a minimal not reduced lexicographic Gröbner basis of the ideal $\langle a, b, P^k \rangle$. The cost is similar to that of the quadratic algorithm for the gcd of $a$ and $b$ (we need the whole remainder sequence), in $O^{\sim}(k \deg_y(a) \deg_y(b) \deg_x(P))$. The reduction of this minimal Gröbner basis takes far more time than its computation; We report on an implementation in Magma of Schost & StPierre's fast normal form [10].

**Keywords:** Gröbner basis, lexicographic order, bivariate

## 1 Introduction

**Background.** Lexicographic Gröbner bases (lexGb for short) play a fundamental role when manipulating polynomial systems due to the elimination property that they are endowed with. But the lexicographic order often does not behave well with standard Gröbner bases algorithms [7], whereas the degree reverse lexicographic order has been often observed to behave the best among monomial orders when computing a Gröbner basis. This is grounded in strong theoretical evidences [1, 9]. As a result, a standard strategy to compute a zero-dimensional lexGb consists first in computing a Gröbner basis for the degree reverse lexicographic order with efficient modern algorithms like F4, F5 [4, 5] and then proceed to a change of order algorithm [6] to compute the reduced lexGb.

In the case of two variables only, the situation is different: Schost & StPierre [11] have shown how a Hermite normal form (HNF) computation of a generalized Sylvester matrix of the input polynomials can reveal the reduced lexGb. While benefiting of the excellent complexity of recent HNF algorithms, the latter are challenging to implement. On the other hand, an approach based on Euclidean division was proposed in [2]. Although limited to input of type $a$, $b$, $T$ with $a$, $b \in \mathbb{K}[x, y]$ and $T(x) \in \mathbb{K}[x]$, the implementation shows very good performance for computing a minimal, but not reduced, lexGb of $\langle a, b, T \rangle$. It turns out that reducing the minimal lexGb ends up being by far the most time consuming part (see Section 4).

**Results.** In this paper, we present a simplified version of [2] when $T = P^k \in \mathbb{K}[x]$ is the power of an irreducible polynomial, and analyze its complexity (the paper [2] does not prove any complexity bound but mentions it as a note). While [2] studies the relationship between the subresultant pseudo-remainder sequence of $a$ and $b$ modulo $P^k$ and the lexGb of $\langle a, b, P^k \rangle$, here we use simpler Euclidean divisions. Without surprise, the cost is equivalent to that of the Euclidean algorithm. More precisely: The cost of computing a minimal lexGb of the ideal $\langle a, b, P^k \rangle$ is within $O^{\sim}(k d_a d_b d_P)$ where $d_a := \deg_y(a)$, $d_b := \deg_y(b)$ and $d_P := \deg_x(P)$. The tilde refers to logarithmic terms. Note that these logarithmic terms are of small degree, and the constant can be taken small as they come from the quadratic Euclidean algorithm (see Theorem 1 for a precise estimate).

To cope with the reduction of the minimal lexGb to the reduced one, we implemented the "fast" version proposed by Schost & StPierre in [10, Section 4] (no implementation has been reported so far). The timings are virtually always better than the estimable command `Reduce`

of Magma. We discuss the situation in more details in Section 4. Given a minimal lexGb $\mathcal{H}$ contining $s+1$ polynomials and having maximal $y$-degree $n_0$, then reducing $\mathcal{H}$ amounts up to $O^{\sim}(s^2 n_0 k d_P)$ operations in $\mathbb{K}$ (see [10, Prop. 4.4 & Prop. 4.6]).

Combining the complexity of the "fast" normal form, and that of Theorem 1, we obtain a total complexity for computing the reduced lexGb of $\langle a,\, b,\, P^k\rangle$ of

$$O^{\sim}(s^2 n_0 k d_P + d_a d_b k d_P). \tag{1}$$

Note that by Lazard's theorem (Section 2.3) the number of polynomials $s$ is always smaller than the maximal $x$-degree $k$, or $y$-degree $n_0$; And that $n_0 \leq d_a$ by Lemma 1. To make a fair comparison, let us mention the Howell normal form approach in [11]. This variant of the Hermite normal form allows coefficients in $\mathbb{K}[x]/x^k$, whereas they are in $\mathbb{K}[x]$ in the Hermite case. It is shown in [11] that after some minor tweakings, the reduced lexGb of $\langle a, b, x^k\rangle$ can be read off the Howell normal form of a generalized Sylvester matrix of the input polynomials. The complexity is $O^{\sim}(d_a^{\omega} k d_P)$.

Thus our approach is better or comparable as soon as $s^2 n_0 = O(d_a^{\omega})$. In the case $s \approx d_a \approx k d_p$, we obtain $O^{\sim}(d_a^4)$ which is worse than $O^{\sim}(d_a^{\omega+1})$. But in many cases, the asymptotic complexity is better.

Again, designing efficient algorithms for the Hermite (or Howell) normal form that would reflect the impressive theoretical progresses made the past 10-15 years reveals challenging. Current algorithms implemented in Magma, Sage or Maple are less sophisticated and not as efficient. Thus our algorithmically simple approach comparable to the efficient Euclidean algorithm, not only showing good performance, but also often theoretically faster, is worth of further investigation.

**Generalizations.** Although currently limited, going from an input $P^k \in \mathbb{K}[x]$ to a general polynomial $T \in \mathbb{K}[x]$ can be done as in [2] through the dynamic evaluation principle. Although this requires a careful complexity analysis, classical dynamic evaluation has shown to increase only moderately the computational cost [3, 12], and it is reasonable to expect the same moderate increase for the one developed in [2].

A more general version of this algorithm would take several polynomials $f_1, \ldots, f_t \in \mathbb{K}[x, y]$ and a univariate $T \in \mathbb{K}[x]$ as input. One can expect a recursive approach where the base case would correspond to $t = 2$ input polynomials treated in the present article. The core algorithm would also be similar to the Euclidean algorithm, and thus the complexity can also be expected to be roughly quadratic as in this work.

## 2 Preliminaries

All along this article given polynomials $f, g$ and a set of polynomials $F$, $f \equiv g[F]$ means that $f - g \in \langle F\rangle$, the ideal generated by $F$.

### 2.1 Monic form

Let $f$ be a bivariate polynomial modulo $P^k$ (that is seen as univariate in $y$ with coefficients in $\mathbb{K}[x]/\langle P^k\rangle$). Write $C_f := P^{v_P(c(f))}$ the largest power of $P$ that divides the content of $f$. It follows that $\tilde{f} := f/C_f$ has a coefficient of largest degree $\lambda$ which is invertible modulo $P$. A version of Weierstrass preparation theorem asserts that there is a monic polynomial $f^1$ of degree $\lambda$ and coefficients in $\mathbb{K}[x]/\langle P^k\rangle$ such that $\tilde{f} \equiv u f^1 [P^k]$, where $u$ is an invertible power series in the local complete ring $\mathbb{K}[[x]]/\langle P^k\rangle \simeq \mathbb{K}[x]/\langle P^k\rangle$. We write:

$$C_f,\ f^1 := \textsc{MonicForm}(f, P^k) \quad \Rightarrow \quad \langle f,\ P^k\rangle = \langle C_f f^1,\, P^k\rangle,$$

which means that $f \equiv u C_f f^1 [P^k]$. For example:

$$x,\ y^2 + 5y + 10x + 6 \leftarrow \textsc{MonicForm}((9x^2 + 10x)y^2 + (x^2 + 6x)y + 5x, x^3).$$

Which means:

$$\langle x(y^2+5y+10x+6),\ x^3\rangle = \langle (9x^2+10x)y^2+(x^2+6x)y+5x,\ x^3\rangle.$$

Such kind monic forms have appeared in various areas across Symbolic Computation and refer to [2, Section 2.3] for more details. It is key here to pursue the Euclidean algorithm with monic polynomials, when a remainder is not monic. It is convenient to implement the Weierstrass preparation formula through Hensel lifting. The complexity is then in $K\mathsf{M}(\deg_y(f))$ operations in $\mathbb{K}[x]/P^k$ for a constant $K$, totaling in $K\mathsf{M}(\deg_y(f))\mathsf{M}(kd_P)$ (see Algorithm 15.10 and Theorem 15.11 in [13] for the details). As usual, $\mathsf{M}(d)$ refers to the cost of multiplying two polynomials of degree at most $d$. It is standard that $\mathsf{M}(d) \in O^\sim(d)$.

### 2.2 Notations

n1. Capital letters $C, G, H$ etc. refer to univariate polynomials in $\mathbb{K}[x]$ with $P$ an irreducible one of degree $d_P$; calligraphic letters $\mathcal{G}, \mathcal{H}$ etc. to minimal lexGb's, and usually small letters to bivariate polynomials.

n2. Given $a \in \mathbb{K}[x,y]$, we denote $\deg_y(a)$ (resp. $\deg_x(a)$) the maximal degree in $y$ (resp. in $x$).

n3. If $\mathcal{G}$ is a minimal lexGb, we denote it $\mathcal{G} = [g_0, \ldots, g_s]$ where $g_s = P^k$ is a power of $P$, and $\text{LM}(g_s) \prec \text{LM}(g_{s-1}) \prec \cdots \prec \text{LM}(g_0)$. The letters $n_i$ and $m_i$ always refer to exponents of $g_i$: $\text{LM}(g_i) = x^{m_i}y^{n_i}$.

n4. According to Lazard's theorem recalled in § 2.3, we can write $g_i = M_i g_i^1$ with $g_i^1$ monic in $y$ and $M_i \in \mathbb{K}[x]$ is the content. We have $M_0 \| M_1 \| \cdots \| M_s$.
Here $A|B$ means that polynomial $A$ divides polynomial $B$, and $A\|B$ means $A|B$ and $A \neq B$. Under our setting, all $M_i$'s are powers of the irreducible polynomial $P \in \mathbb{K}[x]$. Thus the degree $m_i$ of $M_i$ is a multiple of $d_P = \deg_x(P)$.

n5. $\deg_y(\mathcal{G})$ refers to $\deg_y(g_0) = n_0$, the largest $y$-degree of its polynomials.

n6. We define $c(\mathcal{G}) = M_0$, in some sense the "content" of $\mathcal{G}$, so that $\frac{1}{M_0}\mathcal{G}$ is a 0-dimensional lexGb.

### 2.3 Well-known facts about bivariate lexGb

In 1985 Lazard [8] completely characterized the structure of bivariate lexGb's:

***Theorem (1985, Lazard).*** *A family $\mathcal{G} = [g_0, \ldots, g_s]$ of non-constant monic polynomials of $\mathbb{K}[x,y]$, ordered so that $\text{LM}(g_{i-1}) \prec \text{LM}(g_i)$, is a lexGb if and only if: (a) seen as polynomial in $(\mathbb{K}[x])[y]$ the leading coefficient $M_k \in \mathbb{K}[x]$ of $g_k$ divides $g_k$, so that we can write $g_k = M_k g_k^1$ with $g_k^1$ monic in $y$.*

*(b) for all $k \leq s-2$, $g_k^1 \in \frac{1}{M_k}\langle \mathcal{G}_{\geq k+1}\rangle$ (in particular $M_k$ divides $M_j$ for $j \geq k+1$).*

*It is minimal if moreover $(\deg_y(g_k^1))_k$ (resp. $(\deg_x(M_k))_k$) strictly decreases (resp. increases).*

The following elementary result states that the degree of the largest variable for the lex order does not increase when adding a polynomial. We refer to notations n5. and n6. for $\deg_y(\mathcal{G})$ and $c(\mathcal{G})$.

**Lemma 1.** *Let $\mathcal{L}$ be a minimal lexGb of the system $[C\,f^1,\ C\,g^1,\ P^k]$ where $C \in \mathbb{K}[x]$ is a power of $P$, and where $f^1,\ g^1 \in \mathbb{K}[x,y]$ are monic (in $y$) polynomials satisfying $\deg_y(f^1) \geq \deg_y(g^1) > 0$. Then $c(\mathcal{L}) = C$ and $\deg_y(\mathcal{L}) \leq \deg_y(g^1)$.*

*Proof.* The ideal $I = \frac{1}{C}\langle \mathcal{L}\rangle = \langle f^1,\ g^1,\ P^k/C\rangle$ is an ideal of $\mathbb{K}[x,y]$. It contains a monic polynomial, hence is not the zero ideal, and is zero-dimensional. Since $C \cdot I = \langle \mathcal{L}\rangle$, we get $C = c(\mathcal{L})$.

Therefore, we can divide by $C$ and assume that $f = f^1$, $g = g^1$. Run the Euclidean algorithm with input $f$ and $g$ until a remainder $r$ verifies $r \equiv 0[P]$. Write $r \equiv C_r r^1 [P^k]$, so that $P|C_r$. The previous remainder $\rho$ is then the monic gcd of $f$ and $g$ modulo $P$. The polynomials $\rho$ and $r$ are successive remainders in the Euclidean algorithm initiated with $f$ and $g$ modulo $P^k$, therefore $\langle f,\ g,\ P^k\rangle = \langle \rho,\ r,\ P^k\rangle$. Now since $\deg_y(\rho) \leq \deg_y(g)$, the conclusion of the lemma follows.

## 3 Algorithm

The algorithm is essentially an Euclidean algorithm, with some case distinctions and the monic form. It is not surprising that it also naturally has the same complexity as the quadratic gcd algorithm with coefficient in $\mathbb{K}[x]/\langle P^k\rangle$.

**Example.** Consider the input $a, b$ modulo $T = x^3$.

$$a := y^3 + (2x^2 + 3x + 6)y^2 + (10x^2 + x)y + 6, \quad b := y^3 + (4x^2 + 4x + 6)y^2 + (9x^2 + 6x)y + 6x + 6,$$

and $C_a = C_b = 1$. We have (Line 9): $r = C_a a \mod C_b b = (9x^2 + 10x)y^2 + (x^2 + 6x)y + 5x$. And then (Line 15): $C_r,\ r^1 = x,\ y^2 + 5y + 10x + 6 = \textsc{MonicForm}(r, x^3)$.
Recursive call with $C_r b,\ C_r r^1$ (Line 18):

$$\mathcal{H} = \left\{ \begin{array}{l} x(y^2 + 5y + 10x + 6) \\ x^2(y + 3) \\ x^3 \end{array} \right\} \leftarrow \text{GBTwo}(xb,\ xr^1, x^2)$$

Finally the ouput is $[b]\ \mathsf{cat}\ \mathcal{H} = \left\{ \begin{array}{l} y^3 + (4x^2 + 4x + 6)y^2 + (9x^2 + 6x)y + 6x + 6 \\ xy^2 + 5xy + 10x^2 + 6x \\ x^2y + 3x^2 \\ x^3 \end{array} \right\} \mathcal{H}$

---

**Algorithm 1:** GBTwo$(a,\ b,\ P^k)$

---

**Input:** $C_a\, a^1,\ C_b\, b^1 \in \mathbb{K}[x, y]$, $P^k$ with $P$ irreducible.
**Assume:** $C_a,\ C_b \in \mathbb{K}[x]$ are powers of $P$.
$a^1,\ b^1 \in \mathbb{K}[x, y]$ are monic (in $y$) with $\deg_y(a^1) \geq \deg_y(b^1)$
**Output:** minimal lexGb of $\langle C_a\, a^1,\ C_b\, b^1,\ P^k\rangle$

```
1  if deg_y(a^1) == 0 then
2      return [gcd(C_a, C_b)]
3  if deg_y(b^1) == 0 then
4      if C_b|C_a then
5          return [C_b]
6      else (here C_a||C_b )
7          return [C_a a^1, C_b]
                — end base cases. If deg_y(a) = deg_y(b) and C_a||C_b, then switch a and b —
8  if C_b|C_a then
9      r ← C_a a^1 mod C_b b^1
10 else        (here C_a||C_b, hence deg_y(a) > deg_y(b))
11     G ← [C_a a^1]
12     r ← C_b a^1 mod C_b b^1
13 if r ≡ 0[P^k] then
14     return G cat [C_b b^1, P^k]
15 C_r, r^1 ← MonicForm(r, P^k)
16 if C_b||C_r then
17     G ← G cat [C_b b^1]
18 return G cat GBTwo(C_r b^1, C_r r^1, P^k)
```

---

**Theorem 1.** *Algorithm 1 correctly computes a minimal lexGb of* $\langle a,\ b,\ P^k\rangle$.

*Write* $r_0 := a,\ r_1 := b$ *and* $r_2 := r$ *and similarly* $r_3, \ldots, r_\ell, r_{\ell+1}$ *the subsequent remainders computed at Line 9 or 12 in the successive recursive calls made at Line 18. Here* $r_{\ell+1} \equiv 0[P^k]$ *(terminated at Line 14) or* $\deg_y(r_{\ell+1}) = 0$ *(terminated at Line 5 or 7).*

*Let $\rho_i := \deg_y(r_i)$ (so that $\rho_{\ell+1} = 0$ or 1). Algorithm 1 requires*

$$\sum_{i=1}^{\ell}(2\rho_i+1)(\rho_{i-1}-\rho_i+1) + K\mathsf{M}(\rho_{i+1}) \le 2\rho_0\rho_1 + \frac{K}{2}\mathsf{M}(\rho_1^2) + O(\rho_1^2) \subset O^{\sim}(\rho_0\rho_1).$$

*operations in $\mathbb{K}[x]/\langle P^k\rangle$.*

*Proof.* **Step 1: Correctness:** We start with base cases:

At Line 2, both $a^1$ and $b^1$ are equal to 1. Therefore, a minimal lexGb of the system $[C_a,\ C_b,\ P^k]$ is indeed $[\gcd(C_a,\ C_b)]$ as both $C_a$ and $C_b$ are powers of $P$.

At Line 5, we have $b^1 = 1$ and $C_b | C_a$ thus $\langle C_a a^1,\ C_b,\ P^k\rangle = \langle C_b\rangle$.

At Line 7, since $C_a \| C_b$, it holds $\langle C_a a^1,\ C_b,\ P^k\rangle = \langle C_a a^1,\ C_b\rangle$ which is a minimal lexGb.

Now we treat the general case where $\deg_y(a^1) \ge \deg_y(b^1) > 0$.

Termination: recursive call Line 18 involves input with a $y$-degree smaller than that of the input of the main call.

Correctness: Let us treat the case $C_b | C_a$ first, where we reach Line 9. The Euclidean division $C_a a^1 \equiv C_b b^1 q + r[P^k]$ was performed. It gives:

$$\langle C_a a^1,\ C_b b^1,\ P^k\rangle = \langle C_b b^1,\ r,\ P^k\rangle. \tag{2}$$

If $r \equiv 0[P^k]$ then the latter ideal is simply $\langle C_b b^1,\ P^k\rangle$. This is a minimal lexGb returned at Line 14.

Assume now that $r \equiv C_r r^1[P^k]$ with $r^1 \neq 0$ (Line 15). If $C_r = C_b$ then the algorithm outputs a minimal lexGb of $\mathcal{L} := [C_b b^1,\ C_r r^1,\ P^k]$ at Line 18. The ideal it generates is equal to $\langle C_a a^1,\ C_b b^1,\ P^k\rangle$ according to Eq. (2).

Suppose now that $C_b \| C_r$. The returned output (Line 18) is then $\mathcal{G} := [C_b b^1]\ \mathsf{cat}\ \mathcal{L}$ (where $\mathcal{L}$ is a minimal lexGb of the ideal $\langle C_r b^1,\ C_r r^1,\ P^k\rangle$). Since here $C_r b^1 \in \langle C_b b^1\rangle$, Eq. (2) implies:

$$\langle C_b b^1,\ C_r b^1,\ C_r r^1,\ P^k\rangle = \langle C_a a^1,\ C_b b^1,\ P^k\rangle,$$

proving that the output $\mathcal{G}$ satisfies $\langle\mathcal{G}\rangle = \langle C_a a^1,\ C_b b^1,\ P^k\rangle$, as is expected. We use the Lazard's criterion to verify that $\mathcal{G}$ is a minimal lexGb. Regarding that $\mathcal{L}$ is a minimal lexGb, it suffices to verify $b^1 \in \frac{1}{c(\mathcal{L})}\langle\mathcal{L}\rangle$. By Lemma 1 $C_r = c(\mathcal{L})$. By the definition of $\mathcal{L}$ it is clear that $b^1 \in \frac{1}{C_r}\langle\mathcal{L}\rangle$. It is also minimal: $C_r \| C_b$ and by Lemma 1 $\deg_y(b^1) > \deg_y(\mathcal{L})$.

Let us focus next to the case $C_a \| C_b$. Up to switching $a$ and $b$ when $\deg_y(a) = \deg_y(b)$ we can assume that $\deg_y(a) > \deg_y(b)$. The Euclidean division $C_b a^1 \equiv q C_b b^1 + r[P^k]$ Line 12 implies:

$$\langle C_a a^1,\ C_b b^1, P^k\rangle = \langle C_a a^1\rangle + \langle C_b a^1,\ C_b b^1,\ P^k\rangle = \langle C_a a^1\rangle + \langle C_b b^1,\ r,\ P^k\rangle. \tag{3}$$

If $r \equiv 0[P^k]$ then the algorithm output Line 14 the system $[C_a a^1,\ C_b b^1,\ P^k]$. Let us see why it is a minimal lexGb. First, $\deg_y(a) > \deg_y(b)$, and $C_a \| C_b \| P^k$. Second we check Lazard's criterion which boils down to verify $a^1 \in \langle b^1,\ \frac{P^k}{C_b}\rangle$. The latter follows from the fact that $C_b a^1 \equiv q C_b b^1 + r[P^k]$ with $r = 0$ here.

Assume now that $r \equiv C_r r^1[P^k]$ with $r^1 \neq 0[P^k]$. We treat first the case $C_r = C_b$. The output Line (18) is then $[C_a a^1]\ \mathsf{cat}\ \mathcal{L}$ where $\mathcal{L}$ is a minimal lexGb of $\langle C_b b^1,\ C_r r^1,\ P^k\rangle$. According to Eq. (3), it generates an ideal equal to $\langle C_a a^1,\ C_b b^1,\ P^k\rangle$. Lemma 1 gives $c(\mathcal{L}) = C_r = C_b$ so that $C_a \| C_r$, and also $\deg_y(\mathcal{L}) < \deg_y(a)$. Therefore, it suffices to verify Lazard's criterion, namely $a^1 \in \frac{1}{C_r}\langle\mathcal{L}\rangle$, which is apparent in Eq. (3).

Last, assume that $C_b \| C_r$. The output is then $[C_a a^1, C_b b^1]\ \mathsf{cat}\ \mathcal{L}$. We have $\deg_y(a) > \deg_y(b) > \deg_y(r)$ and by Lemma 1 $\deg_y(r) \ge \deg_y(\mathcal{L})$, as well as $C_a \| C_b \| C_r = c(\mathcal{L})$, therefore it suffices to verify Lazard's criterion:

$$a^1 \in \langle b^1\rangle + \frac{1}{C_b}\langle\mathcal{L}\rangle, \quad \text{and} \quad b^1 \in \frac{1}{C_r}\langle\mathcal{L}\rangle.$$

The first inclusion can be read off from Eq. (3). The second inclusion is implied by the very definition of $\mathcal{L}$ which generates $\langle C_r b^1,\ C_r r^1,\ P^k\rangle$.

**Step 2: Complexity:** Base cases are obvious. The output Line 14 when $r \equiv 0[P^k]$ was computed with only one Euclidean division, whose cost is:

$$(2\deg_y(b)+1)(\deg_y(a)-\deg_y(b)+1) = (2\rho_1+1)(\rho_0-\rho_1+1)$$

operations in $\mathbb{K}[x]/\langle P^k\rangle$. The monic form computations Line 15 needs $K\mathsf{M}(\rho_2)$ operations. If the algorithm requires a recursive call Line 18, induction hypothesis implies the cost $\sum_{i=2}^{\ell}(2\rho_i+1)(\rho_{i-1}-\rho_i+1) + K\mathsf{M}(\rho_{i+1})$, which concludes the proof.

Note that inside a recursive call $C_a = C_b$ and thus does not require computing $C_b a^1$ Line 12.

## 4 Schost-StPierre's normal form

Here, the ideal generated by $\mathcal{G} = [g_0, \ldots, g_s]$ is of dimension zero, implying $M_0 = 1$; as before $\deg_x(g_i) = m_i$ and $\deg_y(g_i) = n_i$, so that $n_0 > \cdots > n_s = 0$ and $m_s > \cdots > m_0 = 0$. The purpose is to reduce a polynomial $f$ modulo $\mathcal{G}$. Recall that $g_s = M_s \in \mathbb{K}[x]$ and let $D_i := M_i/M_{i-1} \in \mathbb{K}[x]$. Up to computing the remainder $f_1 := f \bmod g_s$ in time $O^\sim(\deg_y(f)\deg_x(f))$ and then $f_2 := f \bmod g_0$ over the ring $\mathbb{K}[x]/\langle g_s\rangle$ in time $O^\sim(\deg_y(f)m_s)$, we can assume that it is already reduced modulo $\langle g_0, g_s\rangle$; Its monomials $x^a y^b$ satisfy $0 \le a < m_s$ and $0 \le b < n_0$.

### 4.1 Overview

Naturally it goes by reducing successively $f^{(1)} := f \bmod \langle g_0, g_1, g_s\rangle$, $f^{(2)} := f^{(1)} \bmod \langle g_0, g_1, g_2, g_s\rangle$, $\ldots$, $f^{(s-1)} := f^{(s-2)} \bmod \langle g_0, \ldots, g_{s-1}, g_s\rangle$; It also successively outputs quotients of a *certain* division equality: $f = Q_1 g_1 + Q_2 g_2 + \cdots + Q_{s-1} g_{s-1} + Q_s g_s + f^{(s-1)}$. Note that the monomials occurring in the quotients $Q_i$ are not unique as they depend on the paving of the set of monomials in $\mathsf{S} \subset \mathrm{LM}(\langle\mathcal{G}\rangle)$ that is contained in the rectangle $\{0, \ldots, m_s - 1\} \times \{0, \ldots, n_0 - 1\}$. (see Figures 5-6 in [10]).

The two ideas of [10] to perform these reductions fast are:

1. A particular choice of paving (Definition 4.1 in [10]): for $i = 1, \ldots, s-1$, define then $h_i := 2^{\mathrm{val}_2(i)}$ (so that $i/h_i$ is odd), and $\ell_i := \min(h_i, s - i)$. It turns out that the rectangles $\mathsf{R}_i := \{m_i, \ldots, m_{\min(i+h_i,s)} - 1\} \times \{n_i, \ldots, n_{i-h_i} - 1\}$ pave $\mathsf{S}$. Then, the monomials occurring in the (shifted) quotient $Q_i \mathrm{LM}(g_i)$ have exponents belonging to $\mathsf{R}_i$.
2. To realize the division $f^{(i-1)} := f^{(i)} \bmod \langle g_1, \ldots, g_i, g_s\rangle$ using *mixed-radix* representations. (Algo. 4.4, Lines 6 and 9 in [10]). In this representation the cost of reducing modulo products of $D_i$ are free.

According to [10, Lemma 4.5], the cost of computing $f^{(i-1)}$ and $Q_i$ from $f^{(i)}$ and $G_i$ and the $D_i$'s is

$$O^\sim(n_0(m_{i+\ell_i} - m_i) + m_s(n_{i-h_i} - n_i) + \delta),$$

where $\delta$ is the degree of the zero-dimensional ideal $\langle\mathcal{G}\rangle$. Adding these terms form $i = 1$ to $s - 1$, we obtain that the cost of reducing $f$ is $O^\sim(n_0 m_s + s\delta)$ (this is [10, Proposition 4.4]).

### 4.2 Implementation

We have implemented all algorithms 4.1 to 4.5 in Section 4 of [10] in Magma v.2.28-3. See `http://xdahan.sakura.ne.jp/lexgb2.html`. We report timings in comparison with the internal function `Reduce` of Magma. Depending on the shape of the lex segment ideal (also sometimes called "Gröbner" staircase), the "fast" normal form can be up to around 9 times faster.

**Setting.** The field of coefficient $\mathbb{K}$ is a 64bits finite field. All minimal lexGb's to be reduced have a regular lex-segment (its "staircase" is made of the same stair of a given width and height)

**Experiment 1.** (Figure 1) We constructed minimal lexGb's modulo $P^k$ and recorded the timings for reducing it with Sc&St's normal form (bullets) or with Magma's `Reduce` command (squares).

The irreducible polynomial $P \in \mathbb{K}[x]$ is randomly generated and has degree from 1 to 4: one plot for each degree. It corresponds to the width of the stairs making the Gröbner staircase (labelled above each plot). Another randomly generated polynomial in $\mathbb{K}[x, y]$ defines the height of the stairs. Its degree (in $y$) can be from 1 to 7, and is reported in the legend "height" of each plot.

- Plot "width" 1 to 4: width of a stair=degree of the irreducible polynomial $P \in \mathbb{K}[x]$ randomly generated.
- $x$-axis: Number of stairs (from 45 to 90, depending on each plot).
- $y$-axis: Timing in sec (bullets: Sc&St's normal form. Squares: Magma's `Reduce`).
- legend "Height": height of the stairs.
- legend "Ratio": speed-up ratio Magma/Sc&St.

As we can observe, the speed-up ratio tends to increase with the height of each stair, and with their width (for height 4, the speed-up ratio is repectively 2, 5.1, 6.3, 7.6 for width 1,2,3,4). For width and height 1, Magma's `Reduce` is faster. This seems to be due to the large number of conversions between *mixed-radix* and standard representations required to run Sc&St's normal form.

**Experiment 2.** (Figures 2-3). We report timings concerning a family of examples with input $a, b, P^k$ which produces a minimal lexGb made of $k$ polynomials, with a staircase made of $k$ stairs of height 1

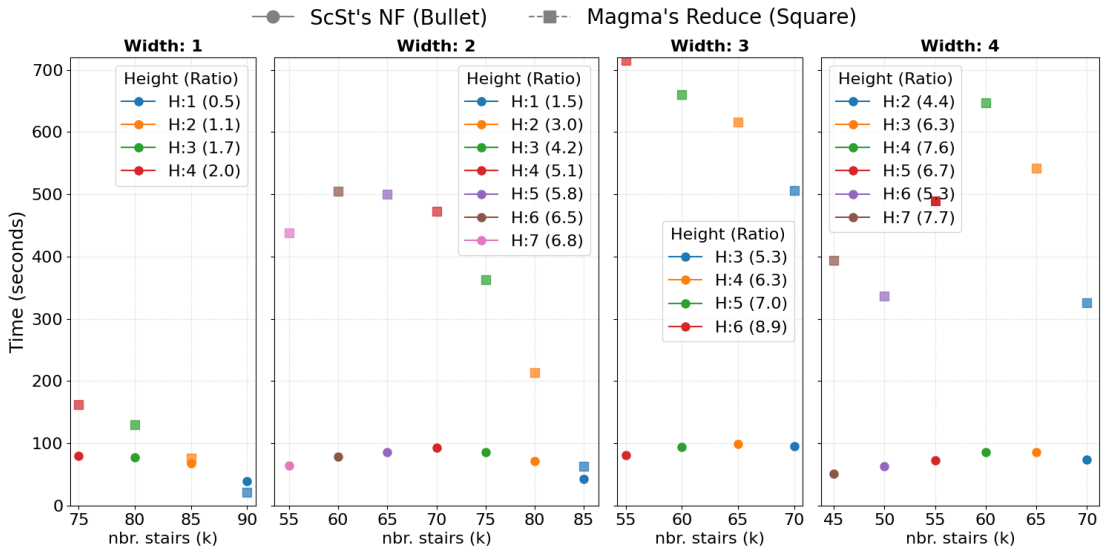


**Fig. 1.** Comparison between native Magma's `Reduce` and Sc&St's normal form.
Size of a stair (from left to right): width 1 to 4 and various heights (associated with the number of stairs) reported in the legends.

and width 1 to 4. In Fig. 2 (left), $P = x$ and (middle, right) $P$ is a random irreducible polynomial of degree 2 (middle) or 3 (right). In Fig. 3 $P$ is a random irreducible polynomial of degree 4,5,6 (from left to right).

First, we observe that the timings of Algorithm 1 (in blue) are negligible. Second, when the width is 1 (Fig. 2 left) the Sc&St's normal form is slower (we should point out straight that we have not optimized the code for input modulo $x^k$, which would make most conversions to and from mixed-radix unnecessary). And when the width is $\geq 2$ (Fig. 2 middle, right, and Fig. 3) it becomes faster. This is in line with the observations mentioned in Experiment 1.

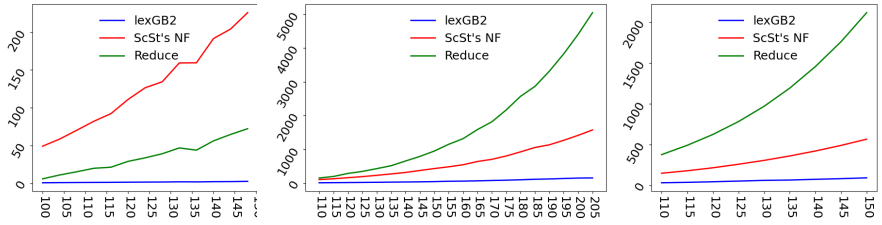


**Fig. 2.** Stairs of size 1x1 (left), modulo $x^k$. Of size 2x1 (3x1) in the middle (resp. right), modulo $P^k$, random $P$ irreducible of degree 2 (resp. degree 3). $y$-axis: time in sec. $x$-axis: number $k$ of polynomials in the lexGb

## References


1. David Bayer and Michael Stillman. A criterion for detecting m-regularity. *Inventiones mathematicae*, 87(1):1–11, 1987.
2. Xavier Dahan. Lexicographic Gröbner bases of bivariate polynomials modulo a univariate one. *Journal of Symbolic Computation*, 110:24–65, 2022.

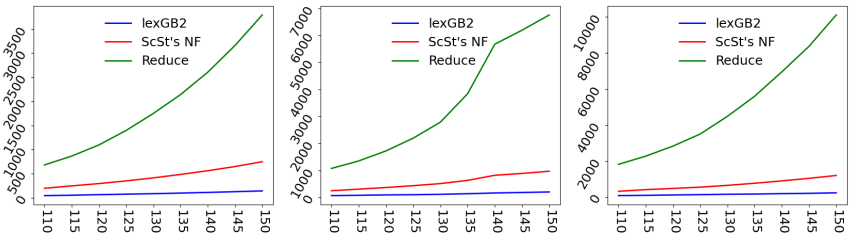


**Fig. 3.** Stairs of size 4x1, 5x1 and 6x1 ($y$-axis: time in sec. $x$-axis: number $k$ of polynomials in the lexGb). Modulo $P^k$, $P$ irreducible of degree 4,5,6


3. Xavier Dahan, Éric Schost, M Moreno Maza, Wenyuan Wu, and Yuzhen Xie. On the complexity of the D5 principle. *ACM SIGSAM Bulletin*, 39(3):97–98, 2005.
4. Jean-Charles Faugère. A new efficient algorithm for computing Gröbner bases (F4). *Journal of pure and applied algebra*, 139(1-3):61–88, 1999.
5. Jean-Charles Faugère. A new efficient algorithm for computing Gröbner bases without reduction to zero (F5). In *Proceedings of the 2002 international symposium on Symbolic and algebraic computation*, pages 75–83, NY, USA, 2002. ACM.
6. Jean-Charles Faugère, Patrizia Gianni, Daniel Lazard, and Teo Mora. Efficient computation of zero-dimensional Gröbner bases by change of ordering. *Journal of Symbolic Computation*, 16(4):329–344, 1993.
7. K. Kalorkoti. Counting and Gröbner bases. *J. Symbolic Computation*, 31:307–313, 2001.
8. Daniel Lazard. Ideal bases and primary decomposition: case of two variables. *Journal Symbolic Computation*, 1(3):261–270, 1985.
9. HyunBin Loh. The converse of a theorem by Bayer and Stillman. *Advances in Applied Mathematics*, 80:62–69, 2016.
10. Éric Schost and Catherine St-Pierre. Newton iteration for lexicographic Gröbner bases in two variables. *Journal of Algebra*, 653:325–377, 2024.
11. Éric Schost and Catherine St-Pierre. $p$-adic algorithms for bivariate Gröbner bases. *Journal of Symbolic Computations*, 2025.
12. Joris van der Hoeven and Grégoire Lecerf. Directed evaluation. *Journal of Complexity*, 60:101498, 2020.
13. Joachim von zur Gathen and Jürgen Gerhard. *Modern computer algebra*. Cambridge University Press, New York, NY, USA, 2003. Second Edition.



**Acknowledgments.** This work was supported by JSPS KAKENHI Grant Number 26K06921.