Fast Gröbner basis computation and polynomial reduction for generic bivariate ideals

JORIS VAN DER HOEVEN[a], ROBIN LARRIEU[b]

Laboratoire d'informatique de l'École polytechnique
LIX, UMR 7161 CNRS
Campus de l'École polytechnique
1, rue Honoré d'Estienne d'Orves
Bâtiment Alan Turing, CS35003
91120 Palaiseau, France

[a]. *Email:* `vdhoeven@lix.polytechnique.fr`
[b]. *Email:* `larrieu@lix.polytechnique.fr`

Final version of December 2019

Let $A, B \in \mathbb{K}[X, Y]$ be two bivariate polynomials over an effective field $\mathbb{K}$, and let G be the reduced Gröbner basis of the ideal $I := \langle A, B\rangle$ generated by A and B with respect to the usual degree lexicographic order. Assuming A and B sufficiently generic, we design a quasi-optimal algorithm for the reduction of $P \in \mathbb{K}[X, Y]$ modulo G, where "quasi-optimal" is meant in terms of the size of the input A, B, P. Immediate applications are an ideal membership test and a multiplication algorithm for the quotient algebra $\mathbb{A} := \mathbb{K}[X, Y]/\langle A, B\rangle$, both in quasi-linear time. Moreover, we show that G itself can be computed in quasi-linear time with respect to the output size.

KEYWORDS: polynomial reduction; Gröbner basis; complexity; algorithm

A.M.S. SUBJECT CLASSIFICATION: 13P10

1. INTRODUCTION

Gröbner bases, also known as standard bases, are a powerful tool for solving systems of polynomial equations, or to compute modulo polynomial ideals. The research area dedicated to their computation is very active, and there is an abundant literature on efficient algorithms for this task. See for example [5, 6, 8] and references therein. Although this problem requires exponential space in the worst case [21], it is in fact tractable for many practical instances. For example, computer algebra systems often implement Faugère's F5 algorithm [6] that is very efficient if the system has sufficient regularity. In this case, a polynomial complexity bound (counting the number of field operations in terms of the expected output size) was established in [1].

The F5 algorithm and all other currently known fast algorithms for Gröbner basis computations rely on linear algebra, and it may seem surprising that fast FFT-based polynomial arithmetic is not used in this area. This can be seen as an illustration of how difficult it is to compute standard bases, but there is another explanation: traditionnally, Gröbner basis algorithms consider a large number of variables and the degree of the generating polynomials is kept small; but fast polynomial arithmetic works best in the opposite regime (a fixed number of variables and large degrees). Even in this setting, it is not clear how to use FFT techniques for Gröbner basis computation. As a first step, one may consider related problems, such as the reduction of multivariate polynomials. It was shown in [16] that reduction can be done in quasi-linear time with respect to the size of the equation $P = Q_0 G_0 + \cdots + Q_n G_n + R$. However, this equation is in general much larger than the intrinsic complexity of the problem, given by the size of P and the degree D of the ideal (which is linked to the size of the generating polynomials). Recent work in the bivariate setting [17] gave an asymptotically optimal reduction algorithm for a particular class of Gröbner bases. This algorithm relies on a terse representation of $G := (G_0, \ldots, G_n)$ in $\tilde{O}(D)$ space, where $\tilde{O}$ stands for the "soft Oh" notation (that hides poly-logarithmic factors) [11]. Assuming that this representation has been precomputed, the extended reduction can be performed in time $\tilde{O}(|P| + D)$, instead of the previous $\tilde{O}(|P| + |G|)$ where $|G| = \Theta(n D)$.

Instead of making regularity assumptions on the Gröbner basis itself, one may focus on the generating polynomials. If the ideal is defined by generic polynomials given in total degree, then the Gröbner basis presents a particular structure, as studied for example in [10, 22]. This situation is often used as a benchmark for polynomial system solving: see the PoSSo problem [7]. In this paper, we restrict ourselves to the bivariate case, as studied for example in [20]. In what follows, $A, B \in \mathbb{K}[X, Y]$ are generic polynomials of degree n, m respectively. We denote by $\langle A, B \rangle$ the ideal they generate, and we consider its Gröbner basis with respect to the graded lexicographic order. The computation of such a basis is classical, but the hypotheses from [17] are not satisfied. However, we show in this paper that a similar terse representation does exist in this case. Therefore, reduction in quasi-linear time remains possible, and it even turns out that the suitable representation can be computed in time $\tilde{O}(D)$ from the input A, B. Combining these two algorithms, we obtain an ideal membership test $P \overset{?}{\in} \langle A, B \rangle$ in quasi-linear time $\tilde{O}(|P| + D)$, without any precomputation. Similarly, there is a quasi-linear multiplication algorithm for $\mathbb{A} := \mathbb{K}[X, Y] / \langle A, B \rangle$ (again without precomputation). Finally, we show that the reduced Gröbner basis can be computed in quasi-linear time with respect to the output size.

Notations and terminology. We assume that the reader is familiar with the theory of Gröbner basis and refer to [11, 2] for basic expositions. We denote the set of *monomials* in r variables by $\mathcal{M} := X_1^{\mathbb{N}} \cdots X_r^{\mathbb{N}} = \{X_1^{i_1} \cdots X_r^{i_r} : i_1, \ldots, i_r \in \mathbb{N}\}$. A *monomial ordering* $\prec$ on $\mathcal{M}$ is a total ordering that is compatible with multiplication. Given a polynomial in r variables $P = \sum_{M \in \mathcal{M}} P_M M \in \mathbb{K}[X_1, \ldots, X_r]$, its *support* $\operatorname{supp} P$ is the set of monomials $M \in \mathcal{M}$ with $P_M \neq 0$. If $P \neq 0$, then $\operatorname{supp} P$ admits a maximal element for $\prec$ that is called its *leading monomial* and that we denote by $\operatorname{lm}(P)$. If $M \in \operatorname{supp} P$, then we say that $P_M M$ is a *term* in P. Given a tuple $A = (A_0, \ldots, A_n)$ of polynomials in $\mathbb{K}[X_1, \ldots, X_r]$, we say that P is *reduced* with respect to A if $\operatorname{supp} P$ contains no monomial that is a multiple of the leading monomial of one of the A_i.

Unless stated otherwise, we will always work in the bivariate setting when $r = 2$, and use X and Y as our main indeterminates instead of X_1 and X_2. In particular, $\mathcal{M} := X^{\mathbb{N}} Y^{\mathbb{N}} = \{X^a Y^b : a, b \in \mathbb{N}\}$. Moreover, we only consider the usual degree lexicographic order with $X \prec Y$, that is

$$X^a Y^b \prec X^u Y^v \Leftrightarrow a + b < u + v \text{ or } (a + b = u + v \text{ and } b < v).$$

Acknowledgements. We thank Vincent Neiger for his remark that simplified Algorithm 6.

2. PRESENTATION OF THE SETTING

We consider an ideal $I \subset \mathbb{K}[X, Y]$ generated by two generic polynomials A, B of total degree n, m and we assume $n \leqslant m$. Here the adjective "generic" should be understood as "no accidental cancellation occurs during the computation". This is typically the case if A, B are chosen at random: assuming $\mathbb{K}$ has sufficiently many elements, the probability of an accidental cancellation is small. In this generic bivariate setting, it is well known that the reduced Gröbner basis $G^{\text{red}} = (G_0^{\text{red}}, \ldots, G_n^{\text{red}})$ of I with respect to $\prec$ can essentially be computed as follows:

- $G_0^{\text{red}} := A$, $G_1^{\text{red}} := B \operatorname{rem} A$.
- $G_i^{\text{red}} := \operatorname{Spol}(G_{i-2}^{\text{red}}, G_{i-1}^{\text{red}}) \operatorname{rem} (G_0^{\text{red}}, \ldots, G_{i-1}^{\text{red}})$ for all $i \in [2, \ldots, n]$.

Remark 1. For simplicity we neglected technical details to actually have the reduced Gröbner basis. First, one should set $G_1^{\text{red}} := B \operatorname{rem} A$, $G_0^{\text{red}} := A \operatorname{rem} G_1^{\text{red}}$ to ensure that no term in G_0^{red} is divisible by the leading term of G_1^{red} (which may happen if $n = m$). Moreover, elements of the reduced basis must be monic; this means a division by the leading coefficient is missing in the given formulas.

We will show next how to construct another (non-reduced) Gröbner basis $G = (G_0, \ldots, G_n)$ with even simpler recurrence relations. Based on these recurrence relations, we will design a variant of the algorithm from [17] that computes the reduction modulo G in quasi-linear time.

DEFINITION 2. *For a polynomial $P = \sum P_{i,j} X^i Y^j \in \mathbb{K}[X, Y]$ of total degree d, we define its dominant diagonal* $\operatorname{Diag}(P) \in \mathbb{K}[Z]$ *by* $\operatorname{Diag}(P) = \sum_{j \leqslant d} P_{d-j,j} Z^j$.

We have the trivial properties that $\operatorname{Diag}(XP) = \operatorname{Diag}(P)$ and $\operatorname{Diag}(YP) = Z \operatorname{Diag}(P)$. For generic A and B, the diagonals $\operatorname{Diag}(A)$ and $\operatorname{Diag}(B)$ are also generic. Then the remainders during the Euclidean algorithm follow a normal sequence (that is the degrees decrease by exactly 1 at each step).

Remark 3. When we say "A, B are generic", we actually make two assumptions. The first one is that the remainders follow a normal sequence as we just mentioned. The second one is simply that the coefficient of Y^n in A is nonzero, so that Y^n is the leading monomial of A.

Let us consider the sequence $G_0, \ldots, G_n$ defined as:

- $G_0 := A$, $G_1 := B \operatorname{rem} A$,
- $G_i := X^{d_i} G_{i-2} - (u_i Y + w_i X) G_{i-1}$ for all $i \in [2, \ldots, n]$ where (with the notation $D_i := \operatorname{Diag}(G_i)$)

$$u_i Z + w_i := D_{i-2} \operatorname{quo} D_{i-1}, \text{ and } d_i := \begin{cases} m - n + 1 & \text{if } i = 2 \\ 2 & \text{if } i > 2 \end{cases}.$$

Let us first notice that the term $X^{d_i} G_{i-2} - u_i Y G_{i-1}$ corresponds to the S-polynomial of G_{i-2} and G_{i-1}, as in the classical Buchberger algorithm [3]. We observe next that $D_1 = \operatorname{Diag}(B) \operatorname{rem} \operatorname{Diag}(A)$, and for $i \geqslant 2$, $D_i = D_{i-2} \operatorname{rem} D_{i-1}$, so that the diagonals are the successive remainders in the Euclidean algorithm (hence the quotients of degree 1). From these two facts, we deduce that the G_i have the same leading monomials as the G_i^{red}, so $G := (G_0, \ldots, G_n)$ is a Gröbner basis of $\langle A, B \rangle$ with respect to $\prec$. Of course, any $P \in \mathbb{K}[X, Y]$ has the same normal form with respect to G and G^{red}.

With this construction, it is easy to deduce higher order recurrence relations $G_{i+k} = U_{i,k} G_i + V_{i,k} G_{i+1}$ for each i, k (with $i + k \leqslant n$). As we will see in section 4, the polynomials $U_{i,k}$ and $V_{i,k}$ have a size that can be controlled as a function of i, k. More precisely, they are homogeneous polynomials of degree roughly k.

3. ALGORITHMIC PREREQUISITES

In this section, we quickly review some basic complexities for fundamental operations on polynomials over a field $\mathbb{K}$. Notice that results presented in this section are not specific to the bivariate case. Running times will always be measured in terms of the required number of field operations in $\mathbb{K}$.

3.1. Polynomial multiplication

We denote by $\mathsf{M}(d)$ the cost of multiplying two dense univariate polynomials of degree d in $\mathbb{K}[X]$. Over general fields, one may take [24, 23, 4]

$$\mathsf{M}(d) = O(d \log d \log \log d).$$

In the case of fields of positive characteristic, one may even take $\mathsf{M}(d) = O(d \log d \, 4^{\log^* d})$, where $\log^* d$ denotes the iterated logarithm [13, 14]. We make the customary assumptions that $\mathsf{M}(d)/d$ is increasing and that $\mathsf{M}(2d) = O(\mathsf{M}(d))$, with the usual implications, such as $\mathsf{M}(d) + \mathsf{M}(e) \leqslant \mathsf{M}(d+e)$.

For multivariate polynomials, the cost of multiplication depends on the geometry of the support. The classical example involves dense "block" polynomials in $\mathbb{K}[X_1, \ldots, X_r]$ of degree $<d_i$ in each variable X_i. This case can be reduced to multiplication of univariate polynomials of degree $<2^{r-1} d_1 \cdots d_r$ using the well known technique of Kronecker substitution [11]. More generally, for polynomials such that the support of the product is included in an initial segment with d elements, it is possible to compute the product in time $O(\mathsf{M}(d))$ [18]. Here an initial segment of $\mathcal{M}$ is a (finite) subset $\mathcal{S}$ such that for any monomial $M \in \mathcal{S}$, all its divisors are again in $\mathcal{S}$.

For the purpose of this paper, we need to consider dense polynomials P in $\mathbb{K}[X, Y]$ whose supports are contained in sets of the form $S_{l,h} := \{M \in \mathcal{M} : l \leqslant \deg M < h\}$. Modulo the change of variables $X^a Y^b \rightarrow T^{a+b} U^b$, such a polynomial can be rewritten as $P(X, Y) = T^l \tilde{P}(T, U)$, where the support of $\tilde{P}$ is an initial segment with the same size as $S_{l,h}$. For a product of two polynomials of this type with a support of size d, this means that the product can again be computed in time $O(\mathsf{M}(d))$.

3.2. Relaxed multiplication

For the above polynomial multiplication algorithms, we assume that the input polynomials are entirely given from the outset. In specific settings, the input polynomials may be only partially known at some point, and it can be interesting to anticipate the computation of the partial output. This is particularly true when working with (truncated) formal power series $f = f_0 + f_1 z + \cdots \in \mathbb{K}[[z]]$ instead of polynomials, where it is common that the coefficients are given as a stream.

In this so-called "relaxed (or online) computation model", the coefficient $(fg)_d$ of a product of two series $f, g \in \mathbb{K}[[z]]$ must be output as soon as $f_0, \ldots, f_d$ and $g_0, \ldots, g_d$ are known. This model has the advantage that subsequent coefficients $f_{d+1}, f_{d+2}, \ldots$ and $g_{d+1}, g_{d+2}, \ldots$ are allowed to depend on the result $(fg)_d$. This often allows us to solve equations involving power series f by rewriting them into *recursive equations* of the form $f = \Phi(f)$, with the property that the coefficient $\Phi(f)_{d+1}$ only depends on earlier coefficients $f_0, \ldots, f_d$ for all d. For instance, in order to invert a power series of the form $1 + zg$ with $g \in \mathbb{K}[[z]]$, we may take $\Phi(f) = 1 - zfg$. Similarly, if $\mathbb{K}$ has characteristic zero, then the exponential of a power series $g \in \mathbb{K}[[z]]$ with $g_0 = 0$ can be computed by taking $\Phi(f) = 1 + \int f g'$.

From a complexity point of view, let $\mathsf{R}(d)$ denote the cost of the relaxed multiplication of two polynomials of degree $<d$. The relaxed model prevents us from directly using fast "zealous" multiplication algorithms from the previous section that are typically based on FFT-multiplication. Fortunately, it was shown in [15, 9] that

$$\mathsf{R}(d) = O(\mathsf{M}(d) \log d). \tag{1}$$

This relaxed multiplication algorithm admits the advantage that it may use any zealous multiplication as a black box. Through the direct use of FFT-based techniques, the following bound has also been established in [19]:

$$\mathsf{R}(d) = d \log d \, \mathrm{e}^{O(\sqrt{\log \log d})}.$$

In the sequel, we will only use a suitable multivariate generalization of the algorithm from [15, 9], so we will always assume that

$$\mathsf{R}(d) \asymp \mathsf{M}(d) \log d.$$

In particular, we have $\mathsf{R}(d) + \mathsf{R}(e) \leqslant \mathsf{R}(d+e)$.

3.3. Polynomial reduction

Let us now consider a Gröbner basis of an ideal in $\mathbb{K}[X_1, \ldots, X_r]$, or, more generally, an auto-reduced tuple $A = (A_0, \ldots, A_n)$ of polynomials in $\mathbb{K}[X_1, \ldots, X_r]$. Then for any $P \in \mathbb{K}[X_1, \ldots, X_r]$, we may compute a relation

$$P = Q_0 A_0 + \cdots + Q_n A_n + R$$

such that R is reduced with respect to A. We call $(Q_0, \ldots, Q_n, R)$ an *extended reduction* of P with respect to A.

The computation of such an extended reduction is a good example of a problem that can be solved efficiently using relaxed multiplication and recursive equations. For a multivariate polynomial T with dense support of any of the types discussed in section 3.1, let $|T|$ denote a bound for the size of its support. With $\mathsf{R}(d)$ as in (1), it has been shown[1] in [16] that the *quotients* $Q_0, \ldots, Q_n$ and the *remainder* R can be computed in time

$$\mathsf{R}(|Q_0 A_0|) + \cdots + \mathsf{R}(|Q_n A_n|) + O(|R|). \tag{2}$$

This implies in particular that the extended reduction can be computed in quasi-linear time in the size of the equation $P = Q_0 A_0 + \cdots + Q_n A_n + R$. However, as pointed out in the introduction, this equation is in general much larger than the input polynomial P.

Extended reductions $(Q_0, \ldots, Q_n, R)$ are far from being unique (only R is unique, and only if A is a Gröbner basis). The algorithm from [16] for the computation of an extended reduction relies on a *selection strategy* that uniquely determines the quotients. More precisely, for every monomial $M \in \mathcal{M}$, we define the set $\mathcal{I}_M := \{i \in \{0, \ldots, n\} : \operatorname{lm}(A_i) \mid M\}$; then we need a rule to chose a particular index $i_M \in \mathcal{I}_M$ (assuming $\mathcal{I}_M$ is non-empty). The initial formulation [16] used the simplest such strategy by taking $i_M = \min \mathcal{I}_M$, but the complexity bound (2) holds for any selection strategy. Now the total size of all quotients $Q_0, \ldots, Q_n$ may be much larger than the size of P for a general selection strategy. One of the key ingredients of the fast reduction algorithm in this paper is the careful design of a "dichotomic selection strategy" that enables us to control the degrees of the quotients.

Remark 4. The notion of selection strategy is somewhat similar to the concept of *involutive division* introduced for the theory of involutive bases [12], although our definition is more permissive.

4. CONCISE REPRESENTATION FOR GRÖBNER BASES

The ideal $\langle A, B \rangle$ has a degree $D := n\,m$, and the reduced Gröbner basis G^r takes space $\Theta(n\,D)$. This overhead compared to D can be reduced: it was shown in [17] that special bases called *vanilla Gröbner bases* admit a terse representation in space $\tilde{O}(D)$ that allow for efficient reduction. Obtaining such a terse representation can be expensive, even if the Gröbner basis is known, but this can be seen as a precomputation. However, for the graded lexicographic order, neither G^r nor G are vanilla, so the results from [17] do not apply. Nevertheless, we show in this section that G does admit a *concise representation* which is analogous to the concept of terse representation. This representation is compatible with a new reduction algorithm detailed in section 5, and it turns out that it can even be computed in time $\tilde{O}(D)$ from the input A, B.

1. The results from [16] actually apply for more general types of supports, but this will not be needed in this paper.

4.1. Definition

As detailled in section 2, the basis G is constructed such that there are recurrence relations of the form $G_{i+k} = U_{i,k} G_i + V_{i,k} G_{i+1}$ for each i, k (with $i + k \leqslant n$). Equivalently, it is more convenient to write this in matrix notation

$$\begin{pmatrix} G_{i+k} \\ G_{i+k+1} \end{pmatrix} = M_{i,k} \begin{pmatrix} G_i \\ G_{i+1} \end{pmatrix} \tag{3}$$

(where we set $G_{n+1} := 0$ to avoid case distinction). This matrix notation has the advantage that the $M_{i,k}$ can be computed from one another using $M_{i,k+\ell} = M_{i+k,\ell} M_{i,k}$. Moreover, the size of the coefficients of the $M_{i,k}$ can be controlled as a function of i, k.

DEFINITION 5. *For each $i \in \{2, \ldots, n\}$, let $u_i Z + v_i := D_{i-2} \operatorname{quo} D_{i-1}$ be the successive quotients in the euclidean algorithm for the dominant diagonals $D_0 := \operatorname{Diag}(G_0)$ and $D_1 := \operatorname{Diag}(G_1)$ (as in section 2). For each i, k with $i + k \leqslant n$, define the matrix $M_{i,k}$ by*

$$\begin{aligned} M_{0,1} &:= \begin{pmatrix} 0 & 1 \\ X^{m-n+1} & -u_2 Y - v_2 X \end{pmatrix}, \\ M_{i,1} &:= \begin{pmatrix} 0 & 1 \\ X^2 & -u_{i+2} Y - v_{i+2} X \end{pmatrix}, \text{ for } 0 < i < n-1, \\ M_{n-1,1} &:= \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}, \\ M_{i,k+1} &:= M_{i+k,1} M_{i,k}. \end{aligned}$$

PROPOSITION 6. *Let the matrices $M_{i,k}$ be as in Definition 5. For all i, k with $i + k \leqslant n$, we then have*

$$\begin{pmatrix} G_{i+k} \\ G_{i+k+1} \end{pmatrix} = M_{i,k} \begin{pmatrix} G_i \\ G_{i+1} \end{pmatrix}.$$

Also, $M_{i,k+\ell} = M_{i+k,\ell} M_{i,k}$ for all i, k, ℓ with $i + k + \ell \leqslant n$.

Now consider the polynomials $U_{i,k}, V_{i,k}, \tilde{U}_{i,k}, \tilde{V}_{i,k}$ such that

$$M_{i,k} = \begin{pmatrix} U_{i,k} & V_{i,k} \\ \tilde{U}_{i,k} & \tilde{V}_{i,k} \end{pmatrix}.$$

With the convention that the zero polynomial is homogeneous of any degree, we have

- *For all i, $V_{i,k}$ is homogeneous of degree $k-1$ and $\tilde{V}_{i,k}$ is homogeneous of degree k.*
- *$U_{0,k}$ is homogeneous of degree $m-n-1+k$ and $\tilde{U}_{0,k}$ is homogeneous of degree $m-n+k$.*
- *For all $i \geqslant 1$, $U_{i,k}$ is homogeneous of degree k and $\tilde{U}_{i,k}$ is homogeneous of degree $k+1$.*

Proof. This is an immediate induction on k using $M_{i,k+1} = M_{i+k,1} M_{i,k}$. □

As in [17], we use a dichotomic selection strategy to control the degrees of the quotients in the extended reduction, together with rewriting rules to evaluate the combination $Q_0 G_0 + \cdots + Q_n G_n$. Then it is sufficient to know enough head terms of each G_i to compute the reduction. Finally, we observe that if G_i and G_{i+1} are known with sufficient precision, then G_{i+k} and G_{i+k+1} can be computed with the same precision using the relation (3) above. More formally, we define the precision as follows:

DEFINITION 7. *Given a polynomial $P \in \mathbb{K}[X,Y]$, we define its* ***upper truncation with precision p*** *as the polynomial $P^{\#} = \pi_p^{\#}(P)$ such that*

- $P^{\#} \in \langle X,Y \rangle^{\deg P - p}$*;*
- $\deg(P - P^{\#}) < \deg P - p$.

In other words, P and $P^{\#}$ have the same terms of degree at least $\deg P - p$*, but all terms of $P^{\#}$ of degree less than* $\deg P - p$ *are zero.*

With the dichotomic selection strategy that we plan to use, we have deg $Q_i < 3 \times 2^{\mathrm{val}_2(i)}$ for $0 < i < n$, so it is *a priori* sufficient to compute G_i with this precision. However, the rewriting rules between the G_i involve two consecutive elements, so that G_{2j} and G_{2j+1} need to be known with the same precision.

DEFINITION 8. *The* ***concise representation*** *of $G = (G_0, \ldots, G_n)$ consists of the following data:*

- *The sequence of truncated elements $G_0^{\#}, \ldots, G_n^{\#}$, where*
  - *for $i \in \{0, 1, n\}$, $G_i^{\#} := G_i$;*
  - *for all other i, $G_i^{\#} := \pi_{3 \times 2^{v(i)}}^{\#}(G_i)$, with the notation $v(i) := \max(\mathrm{val}_2 i, \mathrm{val}_2 (i-1))$;*
- *The collection of rewriting matrices $\mathcal{M}_\lambda$ for each $\lambda \in \{0, \ldots, \lceil \log_2 n \rceil\}$ where the notation $\mathcal{M}_\lambda$ is defined as follows: with the matrices $M_{i,k}$ as in Definition 5, we set*

$$\mathcal{M}_\lambda := (M_{0,2^\lambda}, M_{2^\lambda,2^\lambda}, \ldots, M_{2^\lambda q, r}),$$

  with $q := (n-1)$ quo 2^λ and $r := (n-1)$ rem $2^\lambda + 1$.

As expected, the concise representation requires quasi-linear space with respect to the degree of the ideal.

PROPOSITION 9. *The concise representation requires space $O(n\,m \log n)$.*

Proof. It is easy to see that G_i has degree at most $n + i \leqslant 2n$ in the variable Y, and at most $m + i \leqslant 2m$ in the variable X and in total degree. Then for $i \in \{0, 1, n\}$, each non-truncated element $G_i^{\#} := G_i$ takes $O(n\,m)$ space. Similarly, for $1 < i < n$, each truncated element $G_i^{\#}$ requires $O(m\,2^{v(i)})$ space. For each $\lambda \in \{1, \ldots, \lceil \log_2 n \rceil\}$, there are roughly $2\,n / 2^\lambda$ indices i such that $v(i) = \lambda$, so all elements together require $O(m\,n \log n)$ space.

There are $\lceil n / 2^\lambda \rceil$ elements in $\mathcal{M}_\lambda$, each consisting of four homogeneous polynomials of degree roughly 2^λ (by Proposition 6), except for $M_{0,2^\lambda}$ that has two polynomial entries of degree roughly 2^λ and two entries of degree roughly $m - n + 2^\lambda$. Hence $\mathcal{M}_\lambda$ requires $O(m)$ space and the collection of all $\mathcal{M}_\lambda$ takes $O(m \log n)$ space. □

4.2. Computation

The definition of the concise representation as above is constructive, but the order of the computation must be carefully chosen to achieve the expected quasi-linear complexity. First, by exploiting the recurrence relations $M_{i,k+\ell} = M_{i+k,\ell} M_{i,k}$, it is easy to compute $\mathcal{M}_{\lambda+1}$ from $\mathcal{M}_\lambda$ using the following auxilliary function:

Algorithm 1
Input: $\mathcal{M}_\lambda$ as in Definition 8.
Output: $\mathcal{M}_{\lambda+1}$ as in Definition 8.

```
Set $L := \#\mathcal{M}_\lambda$.
For each $j < L$ quo 2:
    Set $(\mathcal{M}_{\lambda+1})_j := (\mathcal{M}_\lambda)_{2j+1} (\mathcal{M}_\lambda)_{2j}$.
If $L$ rem $2 = 1$:
    Set $(\mathcal{M}_{\lambda+1})_{L \operatorname{quo} 2} := (\mathcal{M}_\lambda)_{L-1}$.
Return $\mathcal{M}_{\lambda+1}$.
```

LEMMA 10. *Algorithm 1 is correct and takes time $O(\mathsf{M}(m))$.*

Proof. Recall that $M_{i,k+\ell} = M_{i+k,\ell} M_{i,k}$ for all i, k, ℓ with $i + k + \ell \leqslant n$. In particular, taking $k = \ell = 2^\lambda$ shows that $(\mathcal{M}_{\lambda+1})_j = (\mathcal{M}_\lambda)_{2j+1} (\mathcal{M}_\lambda)_{2j}$ if $2j + 1 < L - 1$. Concerning the last element of $\mathcal{M}_{\lambda+1}$ (that is, $j = L$ quo $2 - 1$ if L is even, $j = L$ quo 2 otherwise), define

$$\begin{aligned} q &:= (n-1) \operatorname{quo} 2^\lambda, \\ r &:= (n-1) \operatorname{rem} 2^\lambda + 1, \\ q' &:= (n-1) \operatorname{quo} 2^{\lambda+1}, \\ r' &:= (n-1) \operatorname{rem} 2^{\lambda+1} + 1. \end{aligned}$$

If L rem $2 = 0$, then q is odd, that is $q' = (q-1)/2$ and $r' = r + 2^\lambda$, so that $M_{2^{\lambda+1} q', r'} = M_{2^\lambda q, r} M_{2^\lambda (q-1), 2^\lambda}$ is indeed the product of the last two elements of $\mathcal{M}_\lambda$. Conversely if L rem $2 = 1$, then $q' = q/2$ and $r' = r$ so that $\mathcal{M}_\lambda$ and $\mathcal{M}_{\lambda+1}$ have the same last element.

The complexity bound is obtained with the same argument as for Proposition 9. □

The algorithm to compute the concise representation can be decomposed in three steps. First a euclidean algorithm gives recurrence relations of order 1. A second elementary step is to deduce higher order relations by the above algorithm. Finally, one has to compute the truncated basis elements $G_i^\#$. Starting with those of highest precision avoids computing unnecessary terms, so that quasi-linear complexity can be achieved.

Algorithm 2
Input: (A, B), two generic bivariate polynomials of total degrees n and m with $n \leqslant m$.
Output: $(G^\#, \mathcal{M})$, the concise representation of a Gröbner basis of $I := \langle A, B \rangle$ with respect to $\prec$.

```
Set $G_0^\# := A$ and $G_1^\# := B$ rem $A$.
Set $D_0 := \operatorname{Diag}(G_0^\#)$ and $D_1 := \operatorname{Diag}(G_1^\#)$.
For each $i \in \{2, \ldots, n\}$:
    Set $D_i := D_{i-2}$ rem $D_{i-1}$ and $u_i Z + v_i := D_{i-2}$ quo $D_{i-1}$.      // Fail if the quotient has degree >1
    If $i = 2$, then set $d_i := m - n + 1$, otherwise set $d_i := 2$.
    Set $M_{i-2,1} := \begin{pmatrix} 0 & 1 \\ X^{d_i} & -u_i Y - v_i X \end{pmatrix}$.
Set $M_{n-1,1} := \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}$ and $\mathcal{M}_0 := (M_{0,1}, \ldots, M_{n-1,1})$.
For each $\lambda \in \{0, \ldots, \lceil \log_2 n \rceil - 1\}$:
    Compute $\mathcal{M}_{\lambda+1}$ from $\mathcal{M}_\lambda$ using Algorithm 1.
Compute $\begin{pmatrix} G_n^\# \\ 0 \end{pmatrix} := M_{0,n} \begin{pmatrix} G_0^\# \\ G_1^\# \end{pmatrix}$.      // use $\mathcal{M}_{\lceil \log_2 n \rceil} = (M_{0,n})$
```

```
    For each $\lambda \in \{\lceil \log_2 n \rceil - 1, \ldots, 1\}$:
        For each $i \in \{2, \ldots, n-1\}$ with $i \operatorname{rem} 2^{\lambda+1} = 2^\lambda$, set        // Use $\mathcal{M}_\lambda = (M_{0,2^\lambda}, M_{2^\lambda,2^\lambda}, \ldots,)$
            $\begin{pmatrix} G_i^\# \\ G_{i+1}^\# \end{pmatrix} := \pi_{3\times 2^\lambda}^\# \left( M_{i-2^\lambda,2^\lambda} \begin{pmatrix} G_{i-2^\lambda}^\# \\ G_{i-2^\lambda+1}^\# \end{pmatrix} \right).$
    Return $G^\# := (G_0^\#, \ldots, G_n^\#)$ and $(\mathcal{M}_0, \ldots, \mathcal{M}_{\lceil \log_2 n \rceil})$.
```

THEOREM 11. *Algorithm 2 is correct and takes time $O(\mathsf{R}(m^2) + \mathsf{M}(n\,m) \log(n))$*

Proof. The reduction $G_1^\# := B \operatorname{rem} A$ can be done in a relaxed way at a cost of $O(\mathsf{R}(m^2))$. The first loop is clearly correct and each step requires $O(\mathsf{M}(n))$ operations. Alternatively, all the successive quotients can be computed with a fast Euclidean algorithm (see for example [11]) at a cost of $O(\mathsf{M}(n) \log n)$ operations.

In the last loop on λ, since the precision decreases at each step and since there is no accidental cancellation, the invariant

$$\begin{pmatrix} G_i^\# \\ G_{i+1}^\# \end{pmatrix} = \pi_{3\times 2^\lambda}^\# \begin{pmatrix} G_i \\ G_{i+1} \end{pmatrix} \text{ for all } i \text{ with } i \operatorname{rem} 2^{\lambda+1} = 2^\lambda$$

holds. Indeed, regarding upper truncations, it is clear that $\pi_u^\#(P\,Q) = \pi_u^\#(\pi_v^\#(P)\,Q)$ as soon as $u \leqslant v$. Then we have (handle the case $i = 2^\lambda$ separately)

$$\pi_{3\times 2^\lambda}^\# \left(M_{i-2^\lambda,2^\lambda} \begin{pmatrix} G_{i-2^\lambda}^\# \\ G_{i-2^\lambda+1}^\# \end{pmatrix} \right) = \pi_{3\times 2^\lambda}^\# \left(M_{i-2^\lambda,2^\lambda} \begin{pmatrix} G_{i-2^\lambda} \\ G_{i-2^\lambda+1} \end{pmatrix} \right),$$

which proves the correctness. Let us now evaluate the complexity of this loop. For each index i such that $i \operatorname{rem} 2^{\lambda+1} = 2^\lambda$, we have to estimate the support of $G_j^\#$ and $G_{j+1}^\#$ (where $j := i - 2^\lambda$). For $i = 2^\lambda$, $G_j^\#$ and $G_{j+1}^\#$ are completely known, with a support of size $O(m\,n)$. In all other cases $G_j^\#$ and $G_{j+1}^\#$ are upper truncations, with a support of size $O(m\,2^\lambda)$. Consequently, each iteration of the loop requires $O(\mathsf{M}(n\,m))$ operations. □

5. FAST REDUCTION ALGORITHM

In this section, we present a variant of the algorithm from [17] that performs the reduction in quasi-linear time using the concise representation of G. The general idea is the same: reduce modulo the truncated basis to compute the quotients faster, and to compute the remainder, use the recurrence relations to maintain sufficient precision. However, crucial differences with the setting of vanilla Gröbner bases from [17] make the new reduction algorithm more cumbersome.

5.1. General idea

To understand the difficulties we have to overcome in this section, let us recall how reduction algorithms work. Say that P is to be reduced modulo G; then the most common way is to successively reduce the terms of P in decreasing order with respect to the relevant monomial ordering. This is in particular how the relaxed algorithm from section 3.3 works (see [16] for the details). If we decompose the set of all monomials into slices of constant degree, we see that the different slices are handled one after the other (starting with those of highest degrees).

In the setting of vanilla Gröbner bases from [17], these slices are roughly parallel to the Gröbner stairs. This implies that the quotients with respect to the truncated basis $G^{\#}$ are also valid with respect to the full basis G (under the assumption that the degrees of the quotients are controlled, and that each $G_i^{\#}$ is known with the appropriate precision). Then the reduction algorithm decomposes in two phases:

1. Compute the quotients $Q_0, \ldots, Q_n$ by reducing modulo $G^{\#}$. Here the speedup comes from the fact that the equation

$$P = Q_0 G_0^{\#} + \cdots + Q_n G_n^{\#} + R^{\#}$$

is much smaller (in terms of the number of coefficients) than the equation

$$P = Q_0 G_0 + \cdots + Q_n G_n + R. \tag{4}$$

2. Rewrite equation (4) to compute the remainder R. First, find new quotients S_0, S_1, S_n such that

$$S_0 G_0 + S_1 G_1 + S_n G_n = Q_0 G_0 + \cdots + Q_n G_n,$$

then compute R as

$$R := P - S_0 G_0 - S_1 G_1 - S_n G_n. \tag{5}$$

As we said earlier, Gröbner bases in the setting of this paper are *not* vanilla. In particular, the slices of constant degree are not parallel to the Gröbner stairs: the former have a slope of -1, while the latter have a slope of $-1/2$. For this reason, even with the bound on the degree of the Q_i, there are terms of $Q_i (G_i - G_i^{\#})$ that are not in normal form with respect to G. This implies that the quotients with respect to $G^{\#}$ are not valid with respect to G. To maintain sufficient precision to compute the remainder, the two phases must be merged, and equation (4) is rewritten "on the fly" during the reduction algorithm. More precisely, as soon as the quotient Q_j is known, the product $Q_j G_j$ is replaced by some $S_i G_i + S_{i+1} G_{i+1}$, where G_i, G_{i+1} are known with precision larger than G_j.

5.2. Dichotomic selection strategy

To control the degrees of the quotients, we use a dichotomic selection strategy as presented in Figure 1. The idea is to reduce each monomial preferably against one end of the Gröbner basis (G_0 or G_n), or the G_i where i has the highest valuation. To describe the reduction algorithm, it is convenient to introduce the function Φ_G (depending on the leading terms of the G_i) defined as follows:

$$\Phi_G(t) := \begin{cases} (0, t/\operatorname{lt}(G_0)) & \text{If } \operatorname{lt}(G_0) \text{ divides } t \\ (n, t/\operatorname{lt}(G_n)) & \text{If } \operatorname{lt}(G_n) \text{ divides } t \text{ (and } \operatorname{lt}(G_0) \text{ does not)} \\ (i, t/\operatorname{lt}(G_i)) & \text{If } \operatorname{lt}(G_i) \text{ divides } t \text{ with } \operatorname{val}_2(i) \text{ maximal} \\ (-1, t) & \text{If no } \operatorname{lt}(G_i) \text{ divides } t \end{cases}$$

Notice that this definition is non-ambiguous: there is only one index i with $\operatorname{val}_2(i)$ maximal such that $\operatorname{lt}(G_i)$ divides t. Indeed, if $i < j$ have the same valuation λ, then there is some k with $i < k < j$ and $\operatorname{val}_2 k > \lambda$. Moreover, if $\operatorname{lt}(G_i)$ and $\operatorname{lt}(G_j)$ both divide t, then so does $\operatorname{lt}(G_k)$.

Intuitively, if $\Phi_G(t) = (i, t')$, then the term t is reduced against G_i and leads to the term t' in Q_i. The case $i = -1$ corresponds to monomials that are already in normal form with respect to G.

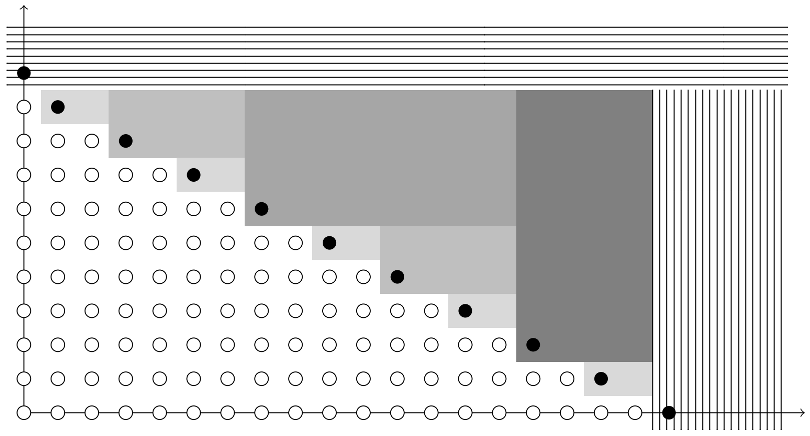

Figure 1. The dichotomic selection strategy ($n = m = 10$): monomials falling in each area are reduced against the corresponding basis element.

LEMMA 12. *Let* $(i, t') := \Phi_G(t)$. *If* $0 < i < n$, *then* $\deg(t') < 3 \times 2^{\operatorname{val}_2(i)} - 1$.

Proof. Let $X^a Y^b := t'$, and denote $\ell := 2^{\operatorname{val}_2 i}$. Then we observe that $b < \ell$: if not, then $\operatorname{lt}(G_{i-\ell})$ would divide t, whereas $\operatorname{val}_2 (i - \ell) > \operatorname{val}_2 i$. A similar reasoning with $G_{i+\ell}$ (or G_n, whenever $i + \ell > n$) shows that $a < 2\ell$. □

5.3. Reduction algorithm

Recall that Definition 5 gives recurrence relations between the G_i: for any indices i, k with $i + k \leqslant n$, we have

$$\begin{pmatrix} G_{i+k} \\ G_{i+k+1} \end{pmatrix} = M_{i,k} \begin{pmatrix} G_i \\ G_{i+1} \end{pmatrix}. \tag{6}$$

This allows for substitutions in the expression $P = Q_0 G_0 + \cdots + Q_n G_n + R$ that are analogous to those used in the reduction algorithm from [17]. In our setting, the substitutions used in the reduction algorithm are based on the following result:

LEMMA 13. *Let* $G_0, \ldots, G_n$ *be a Gröbner basis with recurrence relations defined by the matrices* $M_{i,k}$ *as in Definition 5 (for all indices* i, k *such that* $i + k \leqslant n$*). Given quotients* Q_{i+k}, Q_{i+k+1} *define*

$$(S_i, S_{i+1}) := (Q_{i+k}, Q_{i+k+1}) M_{i,k}.$$

Then we have

$$S_i G_i + S_{i+1} G_{i+1} = Q_{i+k} G_{i+k} + Q_{i+k+1} G_{i+k+1}. \tag{7}$$

Assume now that λ *is such that* Q_{i+k}, Q_{i+k+1} *have degree less than* $3 \times 2^{\lambda-1} - 1$ *and* $k < 2^\lambda$. *Then:*

- *If* $i > 0$, *then* S_i, S_{i+1} *have degree less than* $3 \times 2^\lambda - 1$ *and can be computed in time* $O(\mathsf{M}(4^\lambda))$.
- *If* $i = 0$, *then* $\deg(S_i) < m - n + 3 \times 2^\lambda$, $\deg(S_{i+1}) < 3 \times 2^\lambda - 1$ *and they can be computed in time* $O(\mathsf{M}((m-n) 2^\lambda + 4^\lambda))$.

Proof. Equation (7) is an immediate consequence of the recurrence relations (6) between the G_i. Similarly, the bounds on $\deg S_i$ are consequences of the degree bounds from Proposition 6. □

We can now adapt the extended reduction algorithm from [16] to perform these replacements during the computation.

Algorithm 3

Input: $(P, G^{\#}, \mathcal{M})$, where P is a bivariate polynomial of degree d, and $(G^{\#}, \mathcal{M})$ is the concise representation of a Gröbner basis G (as in Definition 8).

Output: $(Q_0, \ldots, Q_n, R)$, the extended reduction of P with respect to G.

```
Set $(Q_0, \ldots, Q_n) := (0, \ldots, 0)$
Set $(S_0, \ldots, S_n) := (0, \ldots, 0)$                    // new quotients after substitutions as in Lemma 13
Set $P^{\mathrm{subs}} := P$ and $\mathcal{I} := \{i \leqslant n : \deg(G_i) \leqslant d\} \cup \{n\}$     // $\mathcal{I}$: active indices for relaxed multiplications
For $i \in \{0, \ldots, n\}$, set $T_i^{\#} := G_i^{\#} - \operatorname{lt}(G_i^{\#})$.
For $d' \in \{d, \ldots, 0\}$:
    $\mathcal{J} := \{i \in \mathcal{I} : (i = 0) \vee (i = n) \vee (d' < \deg(G_i) + 3 \times 2^{\operatorname{val}_2 i})\}$     // keep only the indices with $Q_i \neq 0$
    For $a \in \{0, \ldots, d'\}$:
        Let $t$ be the term of $X^a Y^{d'-a}$ in $P^{\mathrm{subs}} - \sum_{i \in \mathcal{J}} Q_i T_i^{\#}$, computed using relaxed multiplications.
        Let $(i, t') := \Phi_G(t)$.
        If $i < 0$, then update $R += t'$, else update $Q_i := Q_i + t'$.
    For all $j$ such that $d' = \deg(G_j)$:                                   // See Remark 14
        If $j < n$, then update $S_j += Q_j$ and $\mathcal{I} := \mathcal{I} \setminus \{j\}$ and $P^{\mathrm{subs}} -= Q_j G_j^{\#}$
        If $1 < j < n$ and $\lambda := \operatorname{val}_2(j) > 0$, then:
            Set $k := j - 2^{\lambda}$.
            Set $(\Delta_k, \Delta_{k+1}) := (S_j, S_{j+1}) M_{k, 2^{\lambda}}$.
            Update $(S_k, S_{k+1}) += (\Delta_k, \Delta_{k+1})$.
            Update $P^{\mathrm{subs}} -= \Delta_k G_k^{\#} + \Delta_{k+1} G_{k+1}^{\#} - S_j G_j^{\#} - S_{j+1} G_{j+1}^{\#}$
            Set $(S_j, S_{j+1}) := (0, 0)$.
Return $(Q_0, \ldots, Q_n, R)$
```

Remark 14. Recall that G_0 has degree n and G_i has degree $m + i - 1$ for $i \geqslant 1$. Therefore, the loop on j such that $d' = \deg(G_j)$ is trivial: the set of such j contains at most 1 element, except when $d' = n = m$, in which case there are 2 elements 0 and 1.

THEOREM 15. *Algorithm 3 is correct and runs in time*

$$O(\mathsf{R}(d^2) + \mathsf{R}(n m) \log n).$$

Proof. Let us first explain why the relaxed strategy can indeed be used. We regard the quotients Q_i as streams of coefficients. These coefficients are produced by the updates $Q_i := Q_i + t'$ and consumed in the relaxed evaluation of the products $Q_i T_i^{\#}$. Since our reduction process is essentially based on the same recursive functional equation as in [16], the production of coefficients always occurs before their consumption.

We will show that Algorithm 3 computes the same result as the traditional relaxed reduction algorithm from [16], because the term t that is considered at each step is the same in both cases. More precisely, we must show that for each d', at the start of each iteration of the loop on a, we have

$$\left(P^{\mathrm{subs}} - \sum_{i \in \mathcal{J}} Q_i T_i^{\#}\right)_{a, d'-a} = \left(P - \sum_{i \leqslant n} Q_i T_i\right)_{a, d'-a} \tag{8}$$

where $T_i := G_i - \operatorname{lt}(G_i)$ is the non-truncated analogous of $T_i^{\#}$.

Let us start with the description of a few invariants at the start of the main loop on d'. To avoid case distinctions, let v be the function on $\{0,\ldots,n\}$ defined by

$$v(i) := \begin{cases} \lceil \log_2 n \rceil & \text{if } i \leqslant 1 \\ \max(\operatorname{val}_2 i, \operatorname{val}_2 (i-1)) & \text{otherwise.} \end{cases}$$

Then the following invariants hold:

1. $P^{\text{subs}} = P - \sum_{i \leqslant n} S_i G_i^{\#}$.
2. $\sum_{i \leqslant n} S_i G_i = \sum_{i \notin \mathcal{I}} Q_i G_i$.
3. $\deg(S_i)$ and $\deg(S_{i+1})$ are at most $3 \times 2^{\operatorname{val}_2(i)} - 1$ for all even $i \in \{1,\ldots,n-1\}$.
4. For some $i_0 \leqslant n+1$, we have $\mathcal{I} = \{0,\ldots,i_0-1\} \cup \{n\}$.
5. With i_0 as above, if $j \geqslant n$ or $i > i_0$ or ($i \geqslant i_0$ with i_0 even), then $S_i = 0$.

Invariant 1 is immediate. Invariants 2 and 3 follow from Lemma 13, using $\deg(Q_i) < 3 \times 2^{\operatorname{val}_2(i)} - 1$ by Lemma 12. For invariant 4, we recall that $\deg(G_k) \leqslant \deg(G_{k+1})$ for all $k < n$. Finally, invariant 5 is an immediate consequence of the definition of the index j.

Let us now prove the main claim (8). Notice first that if $0 < i < n$ and $i \in \mathcal{I}$, then $\deg(G_i) \leqslant d'$. Recall also that $\deg(Q_i) < 3 \times 2^{\operatorname{val}_2(i)} - 1$ for $0 < i < n$. This means $\deg(Q_i G_i) < d'$ for $i \in \mathcal{I} \setminus \mathcal{J}$. Since by definition

$$G_0^{\#} = G_0, G_n^{\#} = G_n, \text{and } G_i^{\#} = \pi^{\#}_{3 \times 2^{\operatorname{val}_2(i)}}(G_i) \text{ for } 0 < i < n,$$

we deduce that

$$\left(\sum_{i \in \mathcal{J}} Q_i T_i^{\#} \right)_{a,d'-a} = \left(\sum_{i \in \mathcal{I}} Q_i T_i^{\#} \right)_{a,d'-a} = \left(\sum_{i \in \mathcal{I}} Q_i T_i \right)_{a,d'-a}.$$

To complete the proof (by invariant 1), we show that

$$\left(\sum_{i \leqslant n} S_i G_i^{\#} \right)_{a,d'-a} = \left(\sum_{i \leqslant n} S_i G_i \right)_{a,d'-a} = \left(\sum_{i \notin \mathcal{I}} Q_i G_i \right)_{a,d'-a} = \left(\sum_{i \notin \mathcal{I}} Q_i T_i \right)_{a,d'-a}.$$

For the first identity, we contend that $S_i G_i$ and $S_i G_i^{\#}$ have the same terms of degree d' because of invariants 3 and 5. This is clear if $S_i = 0$, or if $i \leqslant 1$ since $G_0 = G_0^{\#}$ and $G_1 = G_1^{\#}$. Assume therefore that $i > 1$ and $S_i \neq 0$. By invariant 5, the index $i_0 := \min\{i \in \mathbb{N}, i \notin \mathcal{I}\}$ verifies $i \leqslant i_0 < n$, hence i_0 was removed from $\mathcal{I}$ during a previous iteration of the loop on d'. Since $\deg(G_{k+1}) = \deg(G_k) + 1$ for all $0 < k < n$, this actually happened during the previous iteration (with $d' + 1$ instead of d'). It follows that $\deg(G_i) \leqslant d' + 1 = \deg(G_{i_0})$. By definition of $G_i^{\#}$ and invariant 3, the polynomials $S_i G_i$ and $S_i G_i^{\#}$ have the same terms of degree at least $\deg(G_i) - 3 \times 2^{\operatorname{val}_2(i)} + \deg(S_i) \leqslant d'$.

The second identity follows immediately from invariant 2. The last identity follows from the implication $i \notin \mathcal{I} \Rightarrow d' < \deg(G_i)$, so that $Q_i G_i$ and $Q_i T_i$ have the same terms of degree d'.

For the complexity, relaxed multiplications are used to compute the coefficients of the $Q_i T_i^{\#}$, whose support is a subset of the support of $Q_i G_i^{\#}$. Then the relaxed multiplications take time

$$\mathsf{R}(|Q_0 G_0^{\#}|) + \cdots + \mathsf{R}(|Q_n G_n^{\#}|) = O(\mathsf{R}(d^2) + \mathsf{R}(n m) \log n).$$

It remains to evaluate the cost of the zealous multiplications during the rewriting steps. For each index $k \in \{0,\ldots,n-1\}$ with k even, and for each $\lambda < v(k)$, there is an update of S_k, S_{k+1} at a cost of $O(\mathsf{M}(4^{\lambda}))$, followed by an evaluation of the products $S_k G_k^{\#}$ and $S_{k+1} G_{k+1}^{\#}$ at a cost of $O(\mathsf{M}(m 2^{v(k)}))$ operations. This totals for $O(\mathsf{M}(m 2^{v(k)}) \log_2 n)$ operations. Summing for all k, we get a total cost of $O(\mathsf{M}(n m) \log^2 n)$ for all rewriting steps. This fits in the announced complexity bound, by our assumption that $\mathsf{R}(d) \asymp \mathsf{M}(d) \log d$. □

Remark 16. The reduction algorithm can be optimized by delaying the substitutions $(S_k, S_{k+1}) +=(S_j, S_{j+1}) M_{k,2^\lambda}$ until more quotients are known. This allows to decrease by a logarithmic factor the number of products $S_k G_k^\#$, but leads to various technical complications. *A priori*, this does not change the asymptotic complexity; however, with a bound for relaxed multiplication better than $\mathsf{R}(d) \asymp \mathsf{M}(d) \log d$, these zealous products would become predominant without this optimization.

6. APPLICATIONS

Under some regularity assumptions, we provided a quasi-linear algorithm for polynomial reduction, but unlike in [17], it does not rely on expensive precomputations. This leads to significant improvements in the asymptotic complexity for various problems. To illustrate the gain, let us assume to simplify that $n = m$, and neglect logarithmic factors. Then, ideal membership test and modular multiplication are essentially quadratic in n. Also, computing the reduced Gröbner basis has cubic complexity. In all these examples, the bound is intrinsically optimal, and corresponds to a speed-up by a factor n compared to the best previously known algorithms.

6.1. Ideal membership

From any fast algorithms for Gröbner basis computation and (multivariate) polynomial reduction, it is immediate to construct an ideal membership test:

Algorithm 4
Input: (A, B, P), bivariate polynomials of degrees n, m, and d with $n \leqslant m$ and A, B generic.
Output: **true** if $P \in \langle A, B \rangle$, **false** otherwise.

```
Let $(G^\#, \mathcal{M})$ be the concise representation of the Gröbner basis $G$ of $\langle A, B \rangle$ with respect to $\prec$, computed using Algorithm 2.
Let $(Q_0, \ldots, Q_n, R)$ be an extended reduction of $P$ modulo $G$, computed using Algorithm 3.
Return **true** if $R = 0$, **false** otherwise.
```

THEOREM 17. *Algorithm 4 is correct and takes time* $O(\mathsf{R}(m^2 + d^2) + \mathsf{R}(m n) \log n)$.

6.2. Multiplication in the quotient algebra

We designed a practical representation of the quotient algebra $\mathbb{A} := \mathbb{K}[X, Y] / \langle A, B \rangle$ that does not need more space (up to logarithmic factors) than the algebra itself, while still allowing for efficient computation. The main difference with the terse representation from [17] is that said representation is easy to compute, so that multiplication in $\mathbb{A}$ can be done in quasi-linear time, including the cost for the precomputation:

Algorithm 5
Input: (A, B, P, Q), bivariate polynomials, with A, B generic of degrees $n \leqslant m$ and $P, Q \in \mathbb{A}$ of degree at most $m + n$ (typically in normal form).
Output: $P Q \in \mathbb{A}$ in normal form.

```
Let $(G^\#, \mathcal{M})$ be the concise representation of the Gröbner basis $G$ of $\langle A, B \rangle$ with respect to $\prec$, computed using Algorithm 2.
Compute $P Q$ using any (zealous) multiplication algorithm.
Let $(Q_0, \ldots, Q_n, R)$ be an extended reduction of $P Q$ modulo $G$, computed using Algorithm 3.
Return $R$.
```

THEOREM 18. *Algorithm 5 is correct and takes time $O(\mathsf{R}(m^2)+\mathsf{R}(m\,n)\log n)$.*

Remark 19. If we assume $m=n$ or simply $m=O(n)$, then the algebra $\mathbb{A}$ has dimension $D=O(n^2)$ and the above bound can be rewritten $O(\mathsf{R}(D)\log D)$. In the general case, we have $D=m\,n$, and one might wish to discard the term $\mathsf{R}(m^2)$ to achieve complexity $O(\mathsf{R}(D)\log D)=\tilde{O}(D)$. This term includes the computation of $B \operatorname{rem} A$ during the call to Algorithm 2 (which can be seen as precomputation), the multiplication $P\,Q$, and the term $\mathsf{R}(d^2)$ in the complexity of the reduction (since here $d=2\,(m+n)$). If P and Q are given in normal form with respect to G, then $P\,Q$ has in fact degree $2\,(m+n)$ in the variable X and only $2n$ in Y. In this case, $P\,Q$ can be computed in time $O(\mathsf{M}(D))$, and a refined analysis of the reduction algorithm should reduce the term $\mathsf{R}(d^2)$ to $\mathsf{R}(D)$ by observing that the degree in Y remains $O(n)$.

6.3. Reduced Gröbner basis

Since we can reduce polynomials in quasi-linear time, we deduce a new method to compute the reduced Gröbner basis: first compute the non-reduced basis, together with additional information to allow the efficient reduction (which can be done fast); then reduce each element with respect to the others.

Algorithm 6
Input: (A,B), generic bivariate polynomials of total degrees n and m with $n\leqslant m$.
Output: $G^{\mathrm{red}}:=(G_0^{\mathrm{red}},\ldots,G_n^{\mathrm{red}})$ the reduced Gröbner basis of $\langle A,B\rangle$ with respect to $\prec$.

```
Let $(G^{\#},\mathcal{M})$ be the concise representation of $G$, computed using Algorithm 2.
For all $i\in\{0,\ldots,n\}$:
    Set $t_i:=X^{\max(0,m-n-1+2i)}Y^{n-i}=\operatorname{lt}(G_i)$.
    Let $(Q_{0i},\ldots,Q_{ni},R_i)$ be an extended reduction of $t_i$ modulo $G$, computed using Algorithm 3.
    Set $G_i^{\mathrm{red}}:=t_i-R_i$.
Return $(G_0^{\mathrm{red}},\ldots,G_n^{\mathrm{red}})$.
```

THEOREM 20. *Algorithm 6 is correct and takes time $O(\mathsf{R}(m^2)\,n\log n)$.*

Proof. Clearly G_i^{red} is in the ideal and has the same leading monomial as G_i. Moreover, G_i^{red} is monic and none of its terms is divisible by the leading term of any G_j, $j\neq i$. This proves G^{red} is indeed the reduced Gröbner basis of $\langle A,B\rangle$ with respect to $\prec$.

For the complexity, the call to Algorithm 2 takes time $O(\mathsf{R}(m^2)+\mathsf{M}(n\,m)\log n)$. Then, for each i, the reduction requires $O(\mathsf{R}(m^2)+\mathsf{R}(n\,m)\log n)$ operations. The assumption $n\leqslant m$ now yields the desired bound. $\square$

Remark 21. The input A,B has size $\Theta(n^2+m^2)$, and the output G^{red} needs $\Theta(n^2m)$, while Algorithm 6 runs in time $\tilde{O}(n\,m^2)$. Our complexity bound is therefore quasi-optimal only if $m\asymp n$. We expect that a truly quasi-linear bound can still be obtained in general through a refined analysis (as for Remark 19).

6.4. Perspectives

As we noted in different remarks, the announced bounds could be slightly improved at the expense of some technicalities. For example, the optimization mentioned in Remark 16 would lead to a tighter bound when considering even faster relaxed multiplication (as in [19]). Also, as noted in Remarks 19 and 21, a refined analysis is required when $n\ll m$ to give a truly quasi-linear complexity. Apart from this, several more theoretically challenging extensions can be considered.

First we notice that Algorithm 2 fails if the Euclidean algorithm raises a quotient with degree larger than 1. Generically, this should not happen, but this restriction can be limiting in practice, especially in the case of small finite fields. It is then natural to ask how to handle the case of non-normal degree sequences. We conjecture that our algorithm extends (and remains quasi-linear) to the case where the quotients have a bounded degree, with only a logarithmic number of them being larger than 1.

It would also be interesting to extend our ideas to the case of $r > 2$ variables. Whshful thinking suggests that this might possible for fixed r, but with a dependency polynomial in $r!$ (as the ratio between the volumes of a pyramid and a parallelepiped based on the same vectors). Extending the dichotomic selection strategy to higher dimension is rather straightforward, but the question of the successive substitutions is more subtle.

BIBLIOGRAPHY

[1] Magali Bardet, Jean-Charles Faugère, and Bruno Salvy. On the complexity of the F5 Gröbner basis algorithm. *Journal of Symbolic Computation*, pages 1–24, sep 2014.

[2] Thomas Becker and Volker Weispfenning. *Gröbner bases: a computational approach to commutative algebra*, volume 141 of *Graduate Texts in Mathematics*. Springer-Verlag, New York, 1993.

[3] Bruno Buchberger. *Ein Algorithmus zum Auffinden der Basiselemente des Restklassenrings nach einem nulldimensionalen Polynomideal*. PhD thesis, Universitat Innsbruck, Austria, 1965.

[4] David G Cantor and Erich Kaltofen. On fast multiplication of polynomials over arbitrary algebras. *Acta Informatica*, 28(7):693–701, 1991.

[5] Jean-Charles Faugère. A new efficient algorithm for computing Gröbner bases (F4). *Journal of Pure and Applied Algebra*, 139(1–3):61–88, 1999.

[6] 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*, ISSAC '02, pages 75–83. New York, NY, USA, 2002. ACM.

[7] Jean-Charles Faugère, Pierrick Gaudry, Louise Huot, and Guénaël Renault. Polynomial systems solving by fast linear algebra. *ArXiv preprint arXiv:1304.6039*, 2013.

[8] 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.

[9] M. J. Fischer and L. J. Stockmeyer. Fast on-line integer multiplication. *Proc. 5th ACM Symposium on Theory of Computing*, 9:67–72, 1974.

[10] Ralf Fröberg and Joachim Hollman. Hilbert series for ideals generated by generic forms. *Journal of Symbolic Computation*, 17(2):149 – 157, 1994.

[11] J. von zur Gathen and J. Gerhard. *Modern Computer Algebra*. Cambridge University Press, 3rd edition, 2013.

[12] Vladimir P. Gerdt and Yuri A. Blinkov. Involutive bases of polynomial ideals. *Mathematics and Computers in Simulation*, 45(5):519–541, 1998.

[13] D. Harvey and J. van der Hoeven. Faster integer and polynomial multiplication using cyclotomic coefficient rings. Technical Report, ArXiv, 2017. http://arxiv.org/abs/1712.03693.

[14] David Harvey, Joris van der Hoeven, and Grégoire Lecerf. Faster polynomial multiplication over finite fields. Technical Report, ArXiv, 2014. http://arxiv.org/abs/1407.3361.

[15] J. van der Hoeven. Relax, but don't be too lazy. *JSC*, 34:479–542, 2002.

[16] J. van der Hoeven. On the complexity of polynomial reduction. In I. Kotsireas and E. Martínez-Moro, editors, *Proc. Applications of Computer Algebra 2015*, volume 198 of *Springer Proceedings in Mathematics and Statistics*, pages 447–458. Cham, 2015. Springer.

[17] J. van der Hoeven and R. Larrieu. Fast reduction of bivariate polynomials with respect to sufficiently regular Gröbner bases. Technical Report, HAL, 2018. http://hal.archives-ouvertes.fr/hal-01702547.

[18] J. van der Hoeven and É. Schost. Multi-point evaluation in higher dimensions. *AAECC*, 24(1):37–52, 2013.

[19] Joris van der Hoeven. Faster relaxed multiplication. In *Proc. ISSAC '14*, pages 405–412. Kobe, Japan, Jul 2014.

[20] Romain Lebreton, Eric Schost, and Esmaeil Mehrabi. On the complexity of solving bivariate systems: the case of non-singular solutions. In *ISSAC: International Symposium on Symbolic and Algebraic Computation*, pages 251–258. Boston, United States, Jun 2013.

[21] Ernst Mayr. Membership in polynomial ideals over q is exponential space complete. *STACS 89*, pages 400–406, 1989.

[22] Guillermo Moreno-Socías. Degrevlex gröbner bases of generic complete intersections. *Journal of Pure and Applied Algebra*, 180(3):263 – 283, 2003.

[23] A. Schönhage. Schnelle Multiplikation von Polynomen über Körpern der Charakteristik 2. *Acta Infor.*, 7:395–398, 1977.

[24] A. Schönhage and V. Strassen. Schnelle Multiplikation großer Zahlen. *Computing*, 7:281–292, 1971.