GCD Computations Modulo Regular Chains

Xin Li [a] Marc Moreno Maza [a] Wei Pan [a]

[a] *University of Western Ontario*
Department of Computer Science
London, Ontario, Canada N6A 5B7

Abstract

The computation of triangular decompositions involves two fundamental operations: polynomial GCDs modulo regular chains and regularity test modulo saturated ideals. We propose new algorithms for these core operations based on modular methods and fast polynomial arithmetic. We rely on new results connecting polynomial subresultants and GCDs modulo regular chains. We report on extensive experimentation, comparing our code to pre-existing MAPLE implementations, as well as more optimized MAGMA functions. In most cases, our new code outperforms the other packages by several orders of magnitude.

Key words: Fast polynomial arithmetic, subresultants, regular chain, regular GCD, triangular decomposition, polynomial systems

1 Introduction

Triangular decomposition of polynomial systems are based on a recursively univariate vision of multivariate polynomials. Most of the methods computing these decompositions manipulate polynomial remainder sequences (PRS). Moreover, these methods are usually "factorization free", which explains why two different irreducible components may be represented by the same regular chain. An essential routine is then to check whether a hypersurface $f = 0$ contains one of the irreducible components encoded by a regular chain T. This is achieved by testing whether the polynomial f is a zero-divisor modulo the saturated ideal of T. This univariate vision on regular chains allows to perform

⋆ This research was partly supported by NSERC, Maplesoft and MITACS of Canada

Email addresses: lixin@ca.ibm.com (Xin Li), moreno@csd.uwo.ca (Marc Moreno Maza), wpan9@csd.uwo.ca (Wei Pan).

regularity test by means of GCD computations. However, since the saturated ideal of T may not be prime, the concept of a GCD used here is not standard.

The first formal definition of this type of GCDs was given by Kalkbrener in [14]. But in fact, GCDs over non-integral domains were already used in several papers [9,16,12] since the introduction of the celebrated *D5 Principle* [7] by Della Dora, Dicrescenzo and Duval. Indeed, this brilliant and simple observation allows one to carry out over direct product of fields computations that are usually conducted over fields. For instance, computing univariate polynomial GCDs by means of the Euclidean Algorithm.

To define a polynomial GCD of two (or more) polynomials modulo a regular chain T, Kalkbrener refers to the irreducible components that T represents. In order to improve the practical efficiency of those GCD computations by means of subresultant techniques, Rioboo and the second author proposed a more abstract definition in [24]. Their GCD algorithm is, however, limited to regular chains with zero-dimensional saturated ideals.

While Kalkbrener's definition cover the positive dimensional case, his approach cannot support triangular decomposition methods solving polynomial systems incrementally, that is, by solving one equation after another. This is a serious limitation since incremental solving is a powerful way to develop efficient sub-algorithms, by means of geometrical consideration. The first incremental triangular decomposition method was proposed by Lazard in [15], without proof nor a GCD definition. Another such method was established by the second author in [23] together with a formal notion of GCD adapted to the needs of incremental solving. This concept, called *regular GCD*, is reviewed in Section 2 in the context of regular chains. A more abstract definition follows.

Let $\mathbb{B}$ be a commutative ring with unity. Let P, Q and G be non-zero univariate polynomials in $\mathbb{B}[y]$. We say that G is a *regular GCD* of P, Q if the following three conditions hold:

(1) the leading coefficient of G in y is a regular element of $\mathbb{B}$,
(2) G lies in the ideal generated by P and Q in $\mathbb{B}[y]$, and
(3) if G has positive degree w.r.t. y, then G pseudo-divides both of P and Q, that is, the pseudo-remainders $\mathrm{prem}(P, G)$ and $\mathrm{prem}(Q, G)$ are null.

In the context of regular chains, the ring $\mathbb{B}$ is the residue class ring of a polynomial ring $\mathbb{A} := \mathbf{k}[x_1, \ldots, x_n]$ over a field $\mathbf{k}$ by the saturated ideal $\mathrm{sat}(T)$ of a regular chain T. Even if the leading coefficients of P, Q are regular and $\mathrm{sat}(T)$ is radical, the polynomials P, Q may not necessarily admit a regular GCD (unless $\mathrm{sat}(T)$ is prime). However, by splitting T into several regular chains $T_1, \ldots, T_e$ (in a sense specified in Section 2) one can compute a regular GCD of P, Q over each of the ring $\mathbb{A}/\mathrm{sat}(T_i)$, as shown in Section 4.

In this paper, we propose a new algorithm for this task, together with a theoretical study and implementation report, providing dramatic improvements w.r.t. previous work [14,23]. Section 3 exhibits sufficient conditions for a subresultant of P, Q (regarded as univariate polynomials in y) to be a regular GCD of P, Q w.r.t. T. Some of these properties could be known, but we could not find a reference for them, in particular when $\mathrm{sat}(T)$ is not radical.

These results reduce the computation of regular GCDs to that of subresultant chains. More precisely and reusing the above notations, Theorem 1 in Section 4 states that the regular chain T can be split into regular chains $T_1, \ldots, T_e$ such that for each $i = 1 \cdots e$, one of the subresultants of P and Q is a regular GCD of P, Q over $\mathbb{A}/\mathrm{sat}(T_i)$.

In Section 5 we describe our implementation for subresultant chain computation. We observe that, during the computation of triangular decomposition, whenever a regular GCD of P and Q w.r.t. T is needed, the resultant of P and Q w.r.t. y is likely to be computed too. This suggests to organize calculations in a way that the subresultant chain of P and Q is computed only once. To this end, we evaluate the subresultant chain of P and Q at sufficiently many values of $(x_1, \ldots, x_n)$ such that any coefficient of any subresultant P and Q can be interpolated when needed. In our implementation, this evaluation-interpolation scheme is based on FFT techniques. It is available in MAPLE in the module `FastArithmeticTools` of the `RegularChains` library.

The use of fast arithmetic for computing regular GCDs was proposed in [6] for regular chains with zero-dimensional radical saturated ideals. Algorithm 1 in Section 4, however, does not suffer from any such restrictions: the saturated ideal of T may be non-radical or of positive dimension.

Algorithm 1 relies on a procedure for testing whether a polynomial is regular w.r.t the saturated ideal of a regular chain. In Section 6, we propose a new algorithm for this task in dimension zero, see Algorithm 2. Then, under genericity assumptions, we establish running time estimates for both Algorithms 1 and 2, see Theorem 2 and Corollary 3. We explain in Section 6 why these results suggest that Algorithms 1 and 2 are probably more suitable for implementation than the algorithms of [6].

The experimental results of Section 7 illustrate the efficiency of our algorithms. We obtain speed-up factors of several orders of magnitude w.r.t. the algorithms of [23] for regular GCD computations and regularity test. Our code compares and often outperforms packages with similar specifications in MAPLE and MAGMA.

With respect to the ISSAC 2009 article [21], the present paper contains a formal presentation of Algorithm 1 together with a complete proof. In addition, the complexity results of Section 6 are new.

2 Preliminaries

Let $\mathbf{k}$ be a field and let $\mathbf{k}[\mathbf{x}] = \mathbf{k}[x_1, \ldots, x_n]$ be the ring of polynomials with coefficients in $\mathbf{k}$, with ordered variables $x_1 \prec \cdots \prec x_n$. Let $\overline{\mathbf{k}}$ be the algebraic closure of $\mathbf{k}$. If $\mathbf{u}$ is a subset of $\mathbf{x}$ then $\mathbf{k}(\mathbf{u})$ denotes the fraction field of $\mathbf{k}[\mathbf{u}]$. For $F \subset \mathbf{k}[\mathbf{x}]$, we denote by $\langle F \rangle$ the ideal it generates in $\mathbf{k}[\mathbf{x}]$ and by $\sqrt{\langle F \rangle}$ the radical of $\langle F \rangle$. For $H \in \mathbf{k}[\mathbf{x}]$, the *saturated ideal* of $\langle F \rangle$ w.r.t. H, denoted by $\langle F \rangle : H^\infty$, is the ideal

$$\{Q \in \mathbf{k}[\mathbf{x}] \mid \exists m \in \mathbb{N} \text{ s.t. } H^m Q \in \langle F \rangle\}.$$

A polynomial $P \in \mathbf{k}[\mathbf{x}]$ is a *zero-divisor* modulo $\langle F \rangle$ if there exists a polynomial Q such that $PQ \in \langle F \rangle$, and neither P nor Q belongs to $\langle F \rangle$. The polynomial P is *regular* modulo $\langle F \rangle$ if it is neither zero, nor a zero-divisor modulo $\langle F \rangle$. We denote by $V(F)$ the *zero set* (or algebraic variety) of F in $\overline{\mathbf{k}}^n$. For a subset $W \subset \overline{\mathbf{k}}^n$, we denote by $\overline{W}$ its closure in the Zariski topology.

2.1 Regular chains and related notions

Polynomial. If $P \in \mathbf{k}[\mathbf{x}]$ is a non-constant polynomial, the largest variable appearing in P is called the *main variable* of P and is denoted by $\mathrm{mvar}(P)$. We regard P as a univariate polynomial in its main variable. The degree and the leading coefficient of P as a univariate polynomial in $\mathrm{mvar}(P)$ are called *main degree* and *initial* of P; they are denoted by $\mathrm{mdeg}(P)$ and $\mathrm{init}(P)$ respectively.

Triangular Set. A subset T of non-constant polynomials of $\mathbf{k}[\mathbf{x}]$ is a *triangular set* if the polynomials in T have pairwise distinct main variables. Denote by $\mathrm{mvar}(T)$ the set of all $\mathrm{mvar}(P)$ for $P \in T$. A variable $v \in \mathbf{x}$ is *algebraic* w.r.t. T if $v \in \mathrm{mvar}(T)$; otherwise it is *free*. For a variable $v \in \mathbf{x}$ we denote by $T_{<v}$ (resp. $T_{>v}$) the subsets of T consisting of the polynomials with main variable less than (resp. greater than) v. If $v \in \mathrm{mvar}(T)$, we denote by T_v the polynomial $P \in T$ with main variable v. For T not empty, $T_{\max}$ denotes the polynomial of T with largest main variable.

Quasi-component and saturated ideal. Given a triangular set T in $\mathbf{k}[\mathbf{x}]$, denote by h_T the product of the $\mathrm{init}(P)$ for all $P \in T$. The *quasi-component* $W(T)$ of T is $V(T) \setminus V(h_T)$, that is, the set of the points of $V(T)$ which do not cancel any of the initials of T. We denote by $\mathrm{sat}(T)$ the *saturated ideal* of T, defined as follows: if T is empty then $\mathrm{sat}(T)$ is the trivial ideal $\langle 0 \rangle$; otherwise it is the ideal $\langle T \rangle : h_T^\infty$.

Regular chain. A triangular set T is a *regular chain* if either T is empty, or $T \setminus \{T_{max}\}$ is a regular chain and the initial of T_{max} is regular with respect to $\mathrm{sat}(T \setminus \{T_{max}\})$. In this latter case, $\mathrm{sat}(T)$ is a proper ideal of $\mathbf{k}[\mathbf{x}]$. From now on $T \subset \mathbf{k}[\mathbf{x}]$ is a regular chain; moreover we write $m = |T|$, $\mathbf{s} = \mathrm{mvar}(T)$ and $\mathbf{u} = \mathbf{x} \setminus \mathbf{s}$. The ideal $\mathrm{sat}(T)$ enjoys several properties. First, its zero-set equals $\overline{W(T)}$. Second, the ideal $\mathrm{sat}(T)$ is unmixed with dimension $n - m$. Moreover, any prime ideal $\mathfrak{p}$ associated to $\mathrm{sat}(T)$ satisfies $\mathfrak{p} \cap \mathbf{k}[\mathbf{u}] = \langle 0 \rangle$.

Given $P \in \mathbf{k}[\mathbf{x}]$ the *pseudo-remainder* (resp. *iterated resultant*) of P w.r.t. T, denoted by $\mathrm{prem}(P, T)$ (resp. $\mathsf{res}(P, T)$) is defined as follows. If $P \in \mathbf{k}$ or no variables of P is algebraic w.r.t. T, then $\mathrm{prem}(P, T) = P$ (resp. $\mathsf{res}(P, T) = P$). Otherwise, we set $\mathrm{prem}(P, T) = \mathrm{prem}(R, T_{<v})$ (resp. $\mathsf{res}(P, T) = \mathsf{res}(R, T_{<v})$) where v is the largest variable of P which is algebraic w.r.t. T and R is the pseudo-remainder (resp. resultant) of P and T_v w.r.t. v. We have: P is null (resp. regular) w.r.t. $\mathrm{sat}(T)$ if and only if $\mathrm{prem}(P, T) = 0$ (resp. $\mathsf{res}(P, T) \neq 0$).

Regular GCD. Let I be the ideal generated by $\sqrt{\mathrm{sat}(T)}$ in $\mathbf{k}(\mathbf{u})[\mathbf{s}]$. Then $\mathbb{L}(T) := \mathbf{k}(\mathbf{u})[\mathbf{s}]/I$ is a direct product of fields. It follows that every pair of univariate polynomials $P, Q \in \mathbb{L}(T)[y]$ possesses a GCD in the sense of [24]. The following GCD notion [23] is more convenient since it avoids considering radical ideals. Let $T \subset \mathbf{k}[x_1, \ldots, x_n]$ be a regular chain and let $P, Q \in \mathbf{k}[\mathbf{x}, y]$ be non-constant polynomials both with main variable y. Assume that the initials of P and Q are regular modulo $\mathrm{sat}(T)$. A non-zero polynomial $G \in \mathbf{k}[\mathbf{x}, y]$ is a *regular GCD* of P, Q *w.r.t.* T if these conditions hold:

(1) $\mathrm{lc}(G, y)$ is regular with respect to $\mathrm{sat}(T)$;
(2) there exist $u, v \in \mathbf{k}[\mathbf{x}, y]$ such that $g - up - vt \in \mathrm{sat}(T)$;
(3) if $\deg(G, y) > 0$ holds, then $\langle P, Q \rangle \subseteq \mathrm{sat}(T \cup G)$.

In this case, the polynomial G has several properties. First, it is regular with respect to $\mathrm{sat}(T)$. Moreover, if $\mathrm{sat}(T)$ is radical and $\deg(G, y) > 0$ holds, then the ideals $\langle P, Q \rangle$ and $\langle G \rangle$ of $\mathbb{L}(T)[y]$ are equal, so that G is a GCD of (P, Q) w.r.t. T in the sense of [24]. The notion of a regular GCD can be used to compute intersections of algebraic varieties. As an example we will use Formula (1) which follows from Theorem 32 in [23]. Assume that the regular chain T is simply $\{R\}$ where $R = \mathsf{res}(P, Q, y)$, for $R \notin \mathbf{k}$, and let H be the product of the initials of P and Q. Then, we have:

$$V(P, Q) = \overline{W(R, G)} \ \cup \ V(H, P, Q). \tag{1}$$

Splitting. Two polynomials P, Q may not necessarily admit a regular GCD w.r.t. a regular chain T, unless $\mathrm{sat}(T)$ is prime, see Example 1 in Section 3. However, if T "splits" into several regular chains, then P, Q may admit a

regular GCD w.r.t. each of them. This requires a notation. For non-empty regular chains $T, T_1, \ldots, T_e \subset \mathbf{k}[\mathbf{x}]$ we write $T \longrightarrow (T_1, \ldots, T_e)$ whenever

$$\sqrt{\text{sat}(T)} = \sqrt{\text{sat}(T_1)} \cap \cdots \cap \sqrt{\text{sat}(T_e)},$$

$\text{mvar}(T) = \text{mvar}(T_i)$ and $\text{sat}(T) \subseteq \text{sat}(T_i)$ hold for all $1 \leq i \leq e$. Observe that during splitting any polynomial H regular w.r.t $\text{sat}(T)$ is also regular w.r.t. $\text{sat}(T_i)$ for all $1 \leq i \leq e$.

2.2 Fundamental operations on regular chains

We list below the specifications of the fundamental operations on regular chains used in this paper. The names and specifications of these operations are the same as in the `RegularChains` library [18] in MAPLE.

Regularize. For a regular chain $T \subset \mathbf{k}[\mathbf{x}]$ and P in $\mathbf{k}[\mathbf{x}]$, the operation $\mathsf{Regularize}(P, T)$ returns regular chains $T_1, \ldots, T_e$ of $\mathbf{k}[\mathbf{x}]$ such that, for each $1 \leq i \leq e$, P is either zero or regular modulo $\text{sat}(T_i)$ and we have $T \longrightarrow (T_1, \ldots, T_e)$.

RegularGcd. Let T be a regular chain and let $P, Q \in \mathbf{k}[\mathbf{x}, y]$ be non-constant with $\text{mvar}(P) = \text{mvar}(Q) = y \notin \text{mvar}(T)$ and such that both $\text{init}(P)$ and $\text{init}(Q)$ are regular w.r.t. $\text{sat}(T)$. Then, the operation $\mathsf{RegularGcd}(P, Q, T)$ returns a sequence $(G_1, T_1), \ldots, (G_e, T_e)$, called a *regular GCD sequence*, where $G_1, \ldots, G_e$ are polynomials and $T_1, \ldots, T_e$ are regular chains of $\mathbf{k}[\mathbf{x}]$, such that $T \longrightarrow (T_1, \ldots, T_e)$ holds and G_i is a regular GCD of P, Q w.r.t. T_i for all $1 \leq i \leq e$.

NormalForm. Let T be a zero-dimensional normalized regular chain, that is, a regular chain whose saturated ideal is zero-dimensional and whose initials are all in the base field $\mathbf{k}$. Observe that T is a lexicographic Gröbner basis. Then, for $P \in \mathbf{k}[\mathbf{x}]$, the operation $\mathsf{NormalForm}(P, T)$ returns the *normal form* of P w.r.t. T in the sense of Gröbner bases.

2.3 Subresultants

We follow the presentation of [8], [26] and [10].

Determinantal polynomial. Let $\mathbb{B}$ be a commutative ring with identity and let $m \leq n$ be positive integers. Let M be a $m \times n$ matrix with coefficients in $\mathbb{B}$. Let M_i be the square submatrix of M consisting of the first $m-1$ columns of M and the i-th column of M, for $i = m \cdots n$; let $\det M_i$ be the determinant of M_i. We denote by $\mathrm{dpol}(M)$ the element of $\mathbb{B}[y]$, called the *determinantal polynomial* of M, given by

$$\mathrm{dpol}(M) = \det M_m y^{n-m} + \det M_{m+1} y^{n-m-1} + \cdots + \det M_n.$$

Note that if $\mathrm{dpol}(M)$ is not zero then its degree is at most $n-m$. Let $P_1, \ldots, P_m$ be polynomials of $\mathbb{B}[y]$ of degree less than n. We denote by $\mathrm{mat}(P_1, \ldots, P_m)$ the $m \times n$ matrix whose i-th row contains the coefficients of P_i, sorting in order of decreasing degree, and such that P_i is treated as a polynomial of degree $n-1$. We denote by $\mathrm{dpol}(P_1, \ldots, P_m)$ the determinantal polynomial of $\mathrm{mat}(P_1, \ldots, P_m)$.

Subresultant. Let $P, Q \in \mathbb{B}[y]$ be non-constant polynomials of respective degrees p, q with $q \leq p$. Let d be an integer with $0 \leq d < q$. Then the d-th *subresultant* of P and Q, denoted by $S_d(P, Q)$, is

$$S_d(P, Q) = \mathrm{dpol}(y^{q-d-1}P, y^{q-d-2}P, \ldots, P, y^{p-d-1}Q, \ldots, Q).$$

This is a polynomial which belongs to the ideal generated by P and Q in $\mathbb{B}[y]$. In particular, $S_0(P, Q)$ is $\mathrm{res}(P, Q)$, the resultant of P and Q. Observe that if $S_d(P, Q)$ is not zero then its degree is at most d. When $S_d(P, Q)$ has degree d, it is said *non-defective* or *regular*; when $S_d(P, Q) \neq 0$ and $\deg(S_d(P, Q)) < d$, $S_d(P, Q)$ is said *defective*. We denote by s_d the coefficient of $S_d(P, Q)$ in y^d. For convenience, we extend the definition to the q-th subresultant as follows:

$$S_q(P, Q) = \begin{cases} \gamma(Q)Q, & \text{if } p > q \text{ or } \mathrm{lc}(Q) \in \mathbb{B} \text{ is regular} \\ \text{undefined, } \textit{otherwise} \end{cases}$$

where $\gamma(Q) = \mathrm{lc}(Q)^{p-q-1}$. Note that when p equals q and $\mathrm{lc}(Q)$ is a regular element in $\mathbb{B}$, $S_q(P, Q) = \mathrm{lc}(Q)^{-1}Q$ is in fact a polynomial over the total fraction ring of $\mathbb{B}$. We call *specialization property of subresultants* the following statement. Let $\mathbb{D}$ be another commutative ring with identity and Ψ a ring homomorphism from $\mathbb{B}$ to $\mathbb{D}$ such that we have $\Psi(\mathrm{lc}(P)) \neq 0$ and $\Psi(\mathrm{lc}(Q)) \neq 0$. Then we have

$$S_d(\Psi(P), \Psi(Q)) = \Psi(S_d(P, Q)).$$

Divisibility relations of subresultants. The subresultants $S_{q-1}(P, Q)$, $S_{q-2}(P, Q)$, ..., $S_0(P, Q)$ satisfy relations which induce an Euclidean-like algorithm for computing them. Following [8] we first assume that $\mathbb{B}$ is an integral

domain. For convenience, we simply write S_d instead of $S_d(P, Q)$ for each d. We write $A \sim B$ for $A, B \in \mathbb{B}[y]$ whenever they are associated over $\mathrm{fr}(\mathbb{B})$, the field of fractions of $\mathbb{B}$. Then for $d = q-1, \ldots, 1$, we have:

(r_{q-1}) $S_{q-1} = \mathrm{prem}(P, -Q)$, the pseudo-remainder of P by $-Q$,
$(r_{<q-1})$ if $S_{q-1} \neq 0$, with $e = \deg(S_{q-1})$, then the following holds:

$$\mathrm{prem}(Q, -S_{q-1}) = \mathrm{lc}(Q)^{(p-q)(q-e)+1} S_{e-1},$$

(r_e) if $S_{d-1} \neq 0$, with $e = \deg(S_{d-1}) < d-1$, thus S_{d-1} is defective, and we have
(1) $\deg(S_d) = d$, thus S_d is non-defective,
(2) $S_{d-1} \sim S_e$ and $\mathrm{lc}(S_{d-1})^{d-e-1} S_{d-1} = {s_d}^{d-e-1} S_e$, thus S_e is non-defective,
(3) $S_{d-2} = S_{d-3} = \cdots = S_{e+1} = 0$,
(r_{e-1}) if both S_d and S_{d-1} are nonzero, with respective degrees d and e then we have $\mathrm{prem}(S_d, -S_{d-1}) = \mathrm{lc}(S_d)^{d-e+1} S_{e-1}$.

We consider now the case where $\mathbb{B}$ is an arbitrary commutative ring, following Theorem 4.3 in [10]. If S_d, S_{d-1} are nonzero, with respective degrees d and e and if s_d is regular in $\mathbb{B}$ then we have

$$\mathrm{lc}(S_{d-1})^{d-e-1} S_{d-1} = {s_d}^{d-e-1} S_e.$$

Moreover, there exists $C_d \in \mathbb{B}[y]$ such that we have:

$$(-1)^{d-1} \mathrm{lc}(S_{d-1}) s_e S_d + C_d S_{d-1} = \mathrm{lc}(S_d)^2 S_{e-1}.$$

In addition $S_{d-2} = S_{d-3} = \cdots = S_{e+1} = 0$ also holds.

3 Regular GCDs

Throughout this section, we assume $n \geq 1$ and we consider $P, Q \in \mathbf{k}[x_1, \ldots, x_{n+1}]$ non-constant polynomials with the same main variable $y := x_{n+1}$ and such that $p := \deg(P, y) \geq q := \deg(Q, y)$ holds. We denote by R the resultant of P and Q w.r.t. y. Let $T \subset \mathbf{k}[x_1, \ldots, x_n]$ be a non-empty regular chain such that $R \in \mathrm{sat}(T)$ and the initials of P, Q are regular w.r.t. $\mathrm{sat}(T)$. We denote by $\mathbb{A}$ and $\mathbb{B}$ the rings $\mathbf{k}[x_1, \ldots, x_n]$ and $\mathbf{k}[x_1, \ldots, x_n]/\mathrm{sat}(T)$, respectively. Let Ψ be both the canonical ring homomorphism from $\mathbb{A}$ to $\mathbb{B}$ and the ring homomorphism it induces from $\mathbb{A}[y]$ to $\mathbb{B}[y]$. For $0 \leq j \leq q$, we denote by S_j the j-th subresultant of P, Q in $\mathbb{A}[y]$.

Let d be an index in the range $1 \cdots q$ such that $S_j \in \mathrm{sat}(T)$ for all $0 \leq j < d$. Lemma 3 and Lemma 4 exhibit conditions under which S_d is a regular GCD of P and Q w.r.t. T. Lemma 1 and Lemma 2 investigate the properties of S_d when $\mathrm{lc}(S_d, y)$ is regular modulo $\mathrm{sat}(T)$ and $\mathrm{lc}(S_d, y) \in \mathrm{sat}(T)$ respectively.

Lemma 1 *If* $\mathrm{lc}(S_d, y)$ *is regular modulo* $\mathrm{sat}(T)$*, then the polynomial* S_d *is a non-defective subresultant of* P *and* Q *over* $\mathbb{A}$*. Consequently,* $\Psi(S_d)$ *is a non-defective subresultant of* $\Psi(P)$ *and* $\Psi(Q)$ *in* $\mathbb{B}[y]$*.*

Proof. When $d = q$ holds, we are done. Assume $d < q$. Suppose S_d is defective, that is, $\deg(S_d, y) = e < d$. According to item (r_e) in the divisibility relations of subresultants, there exists a non-defective subresultant S_{d+1} such that

$$\mathrm{lc}(S_d, y)^{d-e} S_d = s_{d+1}^{d-e} S_e,$$

where s_{d+1} is the leading coefficient of S_{d+1} in y. By our assumptions, S_e belongs to $\mathrm{sat}(T)$, thus $\mathrm{lc}(S_d, y)^{d-e} S_d \in \mathrm{sat}(T)$ holds. It follows from the fact $\mathrm{lc}(S_d, y)$ is regular modulo $\mathrm{sat}(T)$ that S_d is also in $\mathrm{sat}(T)$. However the fact that $\mathrm{lc}(S_d, y) = \mathrm{init}(S_d)$ is regular modulo $\mathrm{sat}(T)$ also implies that S_d is regular modulo $\mathrm{sat}(T)$. A contradiction. □

Lemma 2 *If* $\mathrm{lc}(S_d, y)$ *is contained in* $\mathrm{sat}(T)$*, then all the coefficients of* S_d *regarded as a univariate polynomial in* y *are nilpotent modulo* $\mathrm{sat}(T)$*.*

Proof. If the leading coefficient $\mathrm{lc}(S_d, y)$ is in $\mathrm{sat}(T)$, then $\mathrm{lc}(S_d, y) \in \mathfrak{p}$ holds for all the associated primes $\mathfrak{p}$ of $\mathrm{sat}(T)$. By the Block Structure Theorem of subresultants (Theorem 7.9.1 of [22]) over an integral domain $\mathbf{k}[x_1, \ldots, x_{n-1}]/\mathfrak{p}$, S_d must belong to $\mathfrak{p}$. Hence we have $S_d \in \sqrt{\mathrm{sat}(T)}$. Indeed, in a commutative ring, the radical of an ideal equals the intersection of all its associated primes. Thus S_d is nilpotent modulo $\mathrm{sat}(T)$. It follows from Exercise 2 of [1] that all the coefficients of S_d in y are also nilpotent modulo $\mathrm{sat}(T)$. □

Lemma 2 implies that, whenever $\mathrm{lc}(S_d, y) \in \mathrm{sat}(T)$ holds, the polynomial S_d will vanish on all the components of $\mathrm{sat}(T)$ after splitting T sufficiently. This is the key reason why Lemma 1 and Lemma 2 can be applied for computing regular GCDs. Indeed, up to splitting via the operation Regularize, one can always assume that either $\mathrm{lc}(S_d, y)$ is regular modulo $\mathrm{sat}(T)$ or $\mathrm{lc}(S_d, y)$ belongs to $\mathrm{sat}(T)$. Hence, up to splitting, one can assume that either $\mathrm{lc}(S_d, y)$ is regular modulo $\mathrm{sat}(T)$ or S_d belongs to $\mathrm{sat}(T)$. Therefore, if $S_d \notin \mathrm{sat}(T)$, we consider the subresultant S_d as a *candidate regular GCD* of P and Q modulo $\mathrm{sat}(T)$.

Example 1 *If* $\mathrm{lc}(S_d, y)$ *is not regular modulo* $\mathrm{sat}(T)$ *then* S_d *may be defective. Consider for instance the polynomials* $P = x_3^2 x_2^2 - x_1^4$ *and* $Q = x_1^2 x_3^2 - x_2^4$ *in* $\mathbb{Q}[x_1, x_2, x_3]$*. We have* $\mathrm{prem}(P, -Q) = (x_1^6 - x_2^6)$ *and* $R = (x_1^6 - x_2^6)^2$*. Let* $T = \{R\}$*. The last subresultant of* P, Q *modulo* $\mathrm{sat}(T)$ *is* $\mathrm{prem}(P, -Q)$*, which has degree 0 w.r.t* x_3*, although its index is* 1*. Note that* $\mathrm{prem}(P, -Q)$ *is nilpotent modulo* $\mathrm{sat}(T)$*.*

In what follows, we give sufficient conditions for the subresultant S_d to be a regular GCD of P and Q w.r.t. T. When $\mathrm{sat}(T)$ is a radical ideal, Lemma 4 states that the assumptions of Lemma 1 are sufficient. This lemma validates

the search for a regular GCD of P and Q w.r.t. T in a bottom-up style, from S_0 up to S_ℓ for some ℓ. Lemma 3 covers the case where $\mathrm{sat}(T)$ is not radical and states that S_d is a regular GCD of P and Q modulo T, provided that S_d satisfies the conditions of Lemma 1 and provided that, for all $d < k \leq q$, the coefficient $\mathrm{lc}(S_k, y^k)$ is either null or regular modulo $\mathrm{sat}(T)$.

Lemma 3 *We reuse the notations and assumptions of Lemma 1. Then S_d is a regular GCD of P and Q modulo $\mathrm{sat}(T)$, if for all $d < k \leq q$, the coefficient s_k of y^k in S_k is either null or regular modulo $\mathrm{sat}(T)$.*

Proof. There are three conditions to satisfy for S_d to be a regular gcd of P and Q modulo $\mathrm{sat}(T)$:

(1) $\mathrm{lc}(S_d)$ is regular modulo $\mathrm{sat}(T)$;
(2) there exists polynomials u and v such that $S_d - uP - vQ \in \mathrm{sat}(T)$; and
(3) both P and Q are in $\mathcal{I} := \mathrm{sat}(T \cup \{S_d\})$.

We write $\Psi(r)$ as $\bar{r}$ for brevity [1], and will prove the lemma in three steps.

Claim 1: If Q and S_{q-1} are in $\mathcal{I}$, then S_d is a regular gcd of P, Q modulo $\mathrm{sat}(T)$.

Indeed, the properties of S_d imply Conditions (1), (2) and we only need to show that the Condition (3) also holds. If $d = q$ holds, then $S_{q-1} \in \mathrm{sat}(T)$ and we are done. Otherwise, $S_{q-1} = \mathrm{prem}(P, -Q)$ is not null modulo $\mathrm{sat}(T)$, because $\bar{S}_{q-1} = 0$ implies that all subresultants of $\bar{P}$ and $\bar{Q}$ with index less than q vanish over $\mathbb{B}$. By assumption, both Q and $S_{q-1} = \mathrm{prem}(P, -Q)$ are in $\mathcal{I}$, P is also in $\mathcal{I}$, since $\mathrm{lc}(Q)$ is regular modulo $\mathrm{sat}(T)$ and is also regular modulo $\mathcal{I}$. This completes the proof of Claim 1.

In order to prove that Q and S_{q-1} are in $\mathrm{sat}(T)$, we define the following set of indices

$$\mathcal{J} = \{j \mid d < j < q, \mathrm{coeff}(S_j, y^j) \notin \mathrm{sat}(T)\}.$$

By assumption, $\mathrm{coeff}(S_j, y^j)$ is regular modulo $\mathrm{sat}(T)$ for each $j \in \mathcal{J}$. Our arguments rely on the Block Structure Theorem (BST) over an arbitrary ring [10] and Ducos' subresultant algorithm [8,23] along with the specialization property of subresultants.

Claim 2: If $\mathcal{J} = \emptyset$, then $S_i \in \mathcal{I}$ holds for all $d < i \leq q$.

Indeed, the BST over $\mathbb{B}$ implies that there exists *at most* one subresultant S_j such that $d < j < q$ and $S_j \notin \mathrm{sat}(T)$. Therefore all but S_{q-1} are in $\mathrm{sat}(T)$,

[1] The degree of $\bar{S}_k$ may be less than the degree of S_k, since its leading coefficient could be in $\mathrm{sat}(T)$. Hence, $\overline{\mathrm{lc}(S_k)}$ may differ from $\mathrm{lc}(\bar{S}_k)$. We carefully distinguish them when the leading coefficient of a subresultant is not regular in $\mathbb{B}$.

and thus $\bar{S}_{q-1}$ is defective of degree d. More precisely, the BST over $\mathbb{B}$ implies

$$\mathrm{lc}(\bar{S}_{q-1})^e S_{q-1} \equiv \mathrm{lc}(S_q)^e S_d \mod \mathrm{sat}(T) \tag{2}$$

for some integer $e \geq 0$. According to Relation (2), $\mathrm{lc}(\bar{S}_{q-1})$ is regular in $\mathbb{B}$. Hence, we have $S_{q-1} \in \mathcal{I}$. By the definition of S_d, we have $\mathrm{prem}(\bar{S}_q, -\bar{S}_{q-1}, y) \in \mathrm{sat}(T)$ which implies $S_q \in \mathcal{I}$. This completes the proof of Claim 2.

Now we consider the case $\mathcal{J} \neq \emptyset$. Write $\mathcal{J}$ explicitly as $\mathcal{J} = \{j_0, j_1, \ldots, j_{\ell-1}\}$, with $\ell = |\mathcal{J}|$ and we assume $j_0 < j_1 < \cdots < j_{\ell-1}$. For convenience, we write $j_\ell := q$. For each integer k satisfying $0 \leq k \leq \ell$ we denote by $\mathcal{P}_k$ the following property:

$$S_i \in \mathcal{I}, \quad \text{for all} \quad d < i \leq j_k.$$

Claim 3: The property $\mathcal{P}_k$ holds for all $0 \leq k \leq \ell$.

We proceed by induction on $0 \leq k \leq \ell$. The base case is $k = 0$. We need to show $S_i \in \mathcal{I}$ for all $d < i \leq j_0$. By the definition of j_0, $\bar{S}_{j_0}$ is a non-defective subresultant of $\bar{P}$ and $\bar{Q}$, and $\mathrm{coeff}(S_i, y^i)$ is in $\mathrm{sat}(T)$ for all $d < i < j_0$. By the BST over $\mathbb{B}$, there is *at most* one $d < i < j_0$ such that $S_i \notin \mathrm{sat}(T)$. If no such a subresultant exists, then we know that $\mathrm{prem}(\bar{S}_{j_0}, -\bar{S}_d)$ is in $\mathrm{sat}(T)$. Consequently, $S_{j_0} \in \mathcal{I}$ holds, which implies $S_j \in \mathcal{I}$ for all $d < i \leq j_0$. On the other hand, if S_{i_0} is not in $\mathrm{sat}(T)$ for some $d < i_0 < j_0$, then $\bar{S}_{i_0}$ is similar to $\bar{S}_d$ over $\mathbb{B}$. To be more precise, we have

$$\mathrm{lc}(\bar{S}_{i_0})^e \bar{S}_{i_0} \equiv \mathrm{lc}(\bar{S}_{j_0})^e \bar{S}_d \mod \mathrm{sat}(T) \tag{3}$$

for some integer $e \geq 0$. With the same reasoning as in the case $\mathcal{J} = \emptyset$, we know that $\mathrm{lc}(\bar{S}_{i_0})$ is regular modulo $\mathrm{sat}(T)$ and we deduce that $S_{i_0} \in \mathcal{I}$ holds. Also, we have $\mathrm{prem}(\bar{S}_{j_0}, -\bar{S}_{i_0}) \in \mathrm{sat}(T)$, by definition of S_d. This implies $S_{j_0} \in \mathcal{I}$ from the fact that $\mathrm{lc}(\bar{S}_{i_0})$ is regular modulo $\mathrm{sat}(T)$ (and thus regular modulo $\mathcal{I}$). Hence, we have $S_i \in \mathcal{I}$ for all $d < i \leq j_0$, as desired. Therefore the property $\mathcal{P}_k$ holds for $k = 0$.

Now we assume that the property $\mathcal{P}_{k-1}$ holds for some $1 \leq k \leq \ell$. We prove that $\mathcal{P}_k$ also holds. According to the BST over $\mathbb{B}$, there exists *at most* one subresultant between $\bar{S}_{j_{k-1}}$ and $\bar{S}_{j_k}$, both of which are non-defective subresultants of $\bar{P}$ and $\bar{Q}$. If $S_i \in \mathrm{sat}(T)$ holds for all $j_{k-1} < i < j_k$, then we have

$$\mathrm{prem}(\bar{S}_{j_k}, -\bar{S}_{j_{k-1}}) \equiv \mathrm{lc}(\bar{S}_{j_k})^e \bar{S}_u \mod \mathrm{sat}(T)$$

for some $d \leq u < j_{k-1}$ and some integer $e \geq 0$. Thus, $\mathrm{prem}(\bar{S}_{j_k}, -\bar{S}_{j_{k-1}}) \in \mathcal{I}$ by our induction hypothesis, and consequently, $S_{j_k} \in \mathcal{I}$ holds. On the other hand, if all subresultants S_i (for $j_{k-1} < i < j_k$) but S_{i_k} (for some index i_k such that $j_{k-1} < i_k < j_k$) are in $\mathrm{sat}(T)$, then $\bar{S}_{i_k}$ is similar to $\bar{S}_{j_{k-1}}$ over $\mathbb{B}$, that is,

$$\mathrm{lc}(\bar{S}_{i_k})^e \bar{S}_{i_k} \equiv \mathrm{lc}(\bar{S}_{j_k})^e \bar{S}_{j_{k-1}} \mod \mathrm{sat}(T) \tag{4}$$

for some integer $e \geq 0$. By Relation (4), $\mathrm{lc}(\bar{S}_{i_k})$ is regular modulo $\mathrm{sat}(T)$, and thus is regular modulo $\mathcal{I}$. Using Relation (4) again, we have $S_{i_k} \in \mathcal{I}$, since $S_{j_{k-1}}$ is in $\mathcal{I}$. Meanwhile, we have

$$\mathrm{prem}(\bar{S}_{j_k}, -\bar{S}_{i_k}) \equiv \mathrm{lc}(\bar{S}_{j_k})^e \bar{S}_u \mod \mathrm{sat}(T)$$

for some $d \leq u < j_{k-1}$ and some integer $e \geq 0$. By the induction hypothesis, we deduce $S_u \in \mathcal{I}$, which implies $S_{j_k} \in \mathcal{I}$ together with the fact that $\mathrm{lc}(\bar{S}_{i_k})$ is regular modulo $\mathcal{I}$. This shows that $S_i \in \mathcal{I}$ holds for all $d < i \leq j_k$. Therefore, property $\mathcal{P}_k$ holds.

Finally, we apply Claim 3 with $k = \ell$, leading to $S_i \in \mathcal{I}$ for all $d < i \leq j_\ell = q$, which completes the proof of our lemma. □

The consequence of the above corollary is that we ensure that S_d is a regular gcd after checking that the leading coefficients of all non-defective subresultants above S_d, are either null or regular modulo $\mathrm{sat}(T)$. Therefore, one may be able to conclude that S_d is a regular GCD simply after checking the coefficients "along the diagonal" of the pictorial representation of the subresultants of P and Q, see Figure 1.

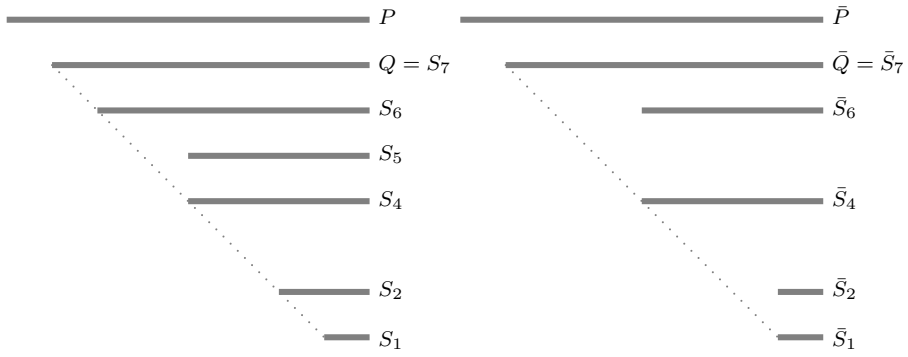


Fig. 1. A possible configuration of the subresultant chain of P and Q. On the left, P and Q have five nonzero subresultants over $\mathbf{k}[\mathbf{x}]$, four of which are non-defective and one of which is defective. Let T be a regular chain in $\mathbf{k}[\mathbf{x}]$ such that $\mathrm{lc}(P)$ and $\mathrm{lc}(Q)$ are regular modulo $\mathrm{sat}(T)$. Further, we assume that $\mathrm{lc}(S_1)$ and $\mathrm{lc}(S_4)$ are regular modulo $\mathrm{sat}(T)$, however, $\mathrm{lc}(S_6)$ is in $\mathrm{sat}(T)$. The right hand side is a possible configuration of the subresultant chain of $\bar{P}$ and $\bar{Q}$. In the proof of Claim 3, the set $\mathcal{J}$ is $\{j_0 = 4\}$ and $j_1 = 7$, whereas $i_0 = 2$ and $i_1 = 6$ are the indices of defective subresultants over $\mathbf{k}[\mathbf{x}]/\mathrm{sat}(T)$. In this case, S_1 is a regular gcd of P and Q modulo $\mathrm{sat}(T)$.

Lemma 4 *With the assumptions of Lemma 1, assume* $\mathrm{sat}(T)$ *radical. Then,* S_d *is a regular GCD of* P, Q *w.r.t.* T.

Proof. As for Lemma 3, it suffices to check that P, Q belong to $\mathrm{sat}(T \cup \{S_d\})$. Let $\mathfrak{p}$ be any prime ideal associated with $\mathrm{sat}(T)$. Define $\mathbb{D} = \mathbf{k}[x_1, \ldots, y]/\mathfrak{p}$

and let $\mathbb{L}$ be the fraction field of the integral domain $\mathbb{D}$. Clearly S_d is the last subresultant of $P,\ Q$ in $\mathbb{D}[y]$ and thus in $\mathbb{L}[y]$. Hence S_d is a GCD of $P,\ Q$ in $\mathbb{L}[y]$. Thus S_d divides $P,\ Q$ in $\mathbb{L}[y]$ and pseudo-divides $P,\ Q$ in $\mathbb{D}[y]$. Therefore $\mathrm{prem}(P, S_d)$ and $\mathrm{prem}(Q, S_d)$ belong to $\mathfrak{p}$. Finally $\mathrm{prem}(P, S_d)$ and $\mathrm{prem}(Q, S_d)$ belong to $\mathrm{sat}(T)$. Indeed, $\mathrm{sat}(T)$ being radical, it is the intersection of its associated primes. □

4 A regular GCD algorithm

Following the notations and assumptions of Section 3, we propose an algorithm to compute a regular GCD sequence of P, Q w.r.t. T. as specified in Section 2.2. Then, we explain how to relax the assumption $R \in \mathrm{sat}(T)$. First, the subresultants of P, Q in $\mathbb{A}[y]$ are assumed to be known. We explain in Section 5 how we compute them in our implementation. Secondly, we rely on the **Regularize** operation specified in Section 2.2. Lastly, we inspect the subresultant chain of P, Q in $\mathbb{A}[y]$ in a bottom-up manner. Therefore, we view $S_1, S_2, \ldots$ as successive candidates and apply either Lemma 4, (if $\mathrm{sat}(T)$ is known to be radical) or Lemma 3.

By virtue of Lemma 1 and Lemma 2 there exists regular chains $T_1, \ldots, T_e \subset \mathbf{k}[\mathbf{x}]$ such that $T \longrightarrow (T_1, \ldots, T_e)$ holds and for each $1 \leq i \leq e$ there exists an index $1 \leq d_i \leq q$ such that the leading coefficient $\mathrm{lc}(S_{d_i}, y)$ of the subresultant S_{d_i} is regular modulo $\mathrm{sat}(T_i)$ and $S_j \in \mathrm{sat}(T_i)$ for all $0 \leq j < d_i$. Such regular chains can be computed using the operation Regularize. If each $\mathrm{sat}(T_i)$ is radical then it follows from Lemma 4 that $(S_{d_1}, T_1), \ldots, (S_{d_e}, T_e)$ is a regular GCD sequence of P, Q w.r.t. T. In practice, when $\mathrm{sat}(T)$ is radical then so are all $\mathrm{sat}(T_i)$, see [2]. If some $\mathrm{sat}(T_i)$ is not known to be radical, then one can compute regular chains $T_{i,1}, \ldots, T_{i,e_i} \subset \mathbf{k}[\mathbf{x}]$ such that $T_i \longrightarrow (T_{i,1}, \ldots, T_{i,e_i})$ holds and for each $1 \leq \ell_i \leq e_i$ there exists an index $1 \leq d_{\ell_i} \leq q$ such that Lemma 3 applies and shows that the subresultant $S_{d_{\ell_i}}$ is regular GCD of P, Q w.r.t. T_{i,ℓ_i}. Such computation relies again on Regularize. The complete procedure of computing regular GCD sequence is given by the algorithm RGSZR.

Theorem 1 *The algorithm* RGSZR *terminates and computes a regular GCD sequence of P and Q with respect to T.*

Proof. We need to show the termination of two while-loops in the algorithm. The first one is the while-loop from Line **4** to Line **15**. Observe that if an item $[i, C]$ out of $Tasks$ satisfies $i = \mathrm{mdeg}(Q)$, then both S_i and c_i are regular modulo $\mathrm{sat}(C)$. Then no true splitting will happen (i.e. $C = D$ at Line **11**) while calling Regularize(S_i, C). Moreover, item $[i, D]$ will only be inserted into $Candidates$ (Line **15**). In other words, no item $[i, D]$ with $i > \mathrm{mdeg}(Q)$ will appear during the computation. Since only items $[i+1, D]$, replacing the item

RGSZR(P, Q, T);

Input : P and Q are polynomials $\in \mathbf{k}[\mathbf{x}][y]$ such that $\mathrm{lc}(P, y)$, $\mathrm{lc}(Q, y)$ are regular modulo $\mathrm{sat}(T)$, $\mathsf{res}(P, Q, y) \in \mathrm{sat}(T)$, and $\deg(P, y) \geq \deg(Q, y) > 0$

Output : a regular GCD sequence of P, Q w.r.t T

```
1  Compute subresultants S_i of P and Q in y for 1 ≤ i ≤ mdeg(Q)
   // Compute regular GCD candidates
2  Find the smallest index i such that S_i ∉ sat(T)
3  Candidates ← ∅, Tasks ← {[i, T]}
4  while Tasks ≠ ∅ do
5      Take and remove an item [i, C] out of Tasks
6      c_i ← lc(S_i, y)
7      if c_i ∈ sat(C) then
8          for D ∈ Regularize(S_i, C) do
9              Tasks ← Tasks ∪ {[i + 1, D]}
10     else
11         for D ∈ Regularize(S_i, C) do
12             if c_i ∈ sat(D) then
13                 Tasks ← Tasks ∪ {[i + 1, D]}
14             else
15                 Candidates ← Candidates ∪ {[i, D]}
   // Check all regular GCD candidates
16 if sat(T) is known to be radical then
17     for [i, C] ∈ Candidates do
18         Results ← Results ∪ {[S_i, C]}
19 else
20     for [i, C] ∈ Candidates do
21         Tasks ← {[i, C]}, Split ← ∅
22         while Tasks ≠ ∅ do
23             Take and remove an item [j, D] out of Tasks
24             if j = mdeg(Q) then
25                 Split ← Split ∪ {D}
26             else
27                 Find the smallest k > j, s.t. s_k = coeff(S_k, y^k) ∉ sat(D)
28                 for E ∈ Regularize(s_k, D) do
29                     Tasks := Tasks ∪ {[j + 1, E]}
30         for E ∈ Split do
31             Results ← Results ∪ {[S_i, E]}
32 return Results
```

Algorithm 1: RGSZR(P, Q, T)

$[i, C]$ from $Tasks$, can be inserted back at Line **9** and Line **13**, this while-loop terminates eventually. The second while-loop is from Line **22** to Line **29**. According to Line **24** and **25**, no item $[j, D]$ with $j > \mathrm{mdeg}(Q)$ will appear during the computation. Hence the latter while-loop terminates as well. This completes the proof of the termination of the algorithm.

Now we prove the correctness of the algorithm. During each iteration of the while-loop from Line **4** to Line **15**, from the specification of Regularize, regular chains in $\mathsf{Regularize}(S_i, C)$ form a splitting of C. This implies a loop invariant: regular chains in $Tasks$ and $Candidates$ form a splitting of T. What we need to show is: each item $[i, D]$ of $Candidates$ satisfies that S_i is a candidate regular GCD of P, Q w.r.t D and $\mathrm{lc}(S_i, y)$ is regular modulo $\mathrm{sat}(D)$.

We first prove a loop invariant for items in $Tasks$: *During each iteration of the first while-loop, each item* $[i, C]$ *in* $Tasks$ *satisfies* $S_k \in \mathrm{sat}(C)$ *for all* $k < i$.

Firstly, for each item $[i + 1, D]$ inserted back into $Tasks$ at Line **9**, we need to show $S_i \in \mathrm{sat}(D)$ for each $D \in \mathsf{Regularize}(S_i, C)$. Since $c_i \in \mathrm{sat}(C)$, by Lemma 2, S_i is a nilpotent modulo $\mathrm{sat}(C)$, and thus S_i cannot be regular modulo $\mathrm{sat}(D)$. By the specification of Regularize, $S_i \in \mathrm{sat}(D)$ for each D. Secondly, for each item $[i + 1, D]$ inserted back into $Tasks$ at Line **13**, we know $c_i \notin \mathrm{sat}(C)$ but $c_i \in \mathrm{sat}(D)$, and need to show $S_i \in \mathrm{sat}(D)$. Lemma 2 still applies, which implies that S_i is a nilpotent modulo $\mathrm{sat}(D)$. Since D is a regular chain in $\mathsf{Regularize}(S_i, D)$, S_i must be null modulo $\mathrm{sat}(D)$. This proves the loop invariant.

Now for each item $[i, D]$ inserted into $Candidates$ at Line **15**, $c_i = \mathrm{lc}(S_i, y)$ is regular modulo $\mathrm{sat}(D)$, since we have $c_i \notin \mathrm{sat}(D)$ and $D \in \mathsf{Regularize}(S_i, C)$. It follows from the fact that $[i, C]$ is taken from $Tasks$, $S_k \in \mathrm{sat}(D)$ holds for all $k < i$. Therefore, for each $[i, D]$ in $Candidates$, S_i is a candidate regular GCD of P, Q w.r.t D, as desired.

If $\mathrm{sat}(T)$ is known to be radical, then we are done according to Lemma 4.

To finalize the proof, we show the correctness of the procedure checking each candidate regular GCD (Lines **16** to **31**). With a similar reasoning as above, during each iteration of the for-loop from Line **20** to Line **31**, all regular chains appearing in $Candidates$ and $Results$ still form a splitting of T. We only need to show that each item $[S_i, E]$ inserted into $Results$ satisfies that S_i is a regular GCD of P, Q w.r.t E.

Indeed, a key invariant of the while-loop from Line **22** to Line **29** is: each item $[j, D]$ from $Tasks$ satisfies that $\mathrm{coeff}(S_k, y^k)$ is null or regular modulo $\mathrm{sat}(D)$ for each $i < k \leq j$. This invariant is clearly maintained by Lines **27**, **28** and **29**. After finishing this while-loop, the set $Split$ consists of regular chains E such that $\mathrm{coeff}(S_k, y^k)$ is null or regular modulo $\mathrm{sat}(E)$ for each $i < k \leq \mathrm{mdeg}(Q)$.

According to Lemma 3, S_i is a regular GCD of P, Q w.r.t E, which will be added into *Results* at Line **31**. □

Recall the definition of a candidate regular GCD. Given a regular chain T in $\mathbf{k}[\mathbf{x}]$, and polynomials P, Q in $\mathbf{k}[\mathbf{x}][x_{n+1}]$ such that $\mathrm{mvar}(P) = \mathrm{mvar}(Q) = x_{n+1}$, $\mathrm{mdeg}(P) \geq \mathrm{mdeg}(Q)$, and initials of P, Q are regular modulo $\mathrm{sat}(T)$. Assume that there exists an index d in the range $1 \cdots q = \mathrm{mdeg}(Q)$ such that $S_d \notin \mathrm{sat}(T)$ and $S_j \in \mathrm{sat}(T)$ for all $0 \leq j < d$. The subresultant S_d is called a *candidate regular GCD* of P and Q w.r.t T. The following corollary is a direct consequence of Theorem 1.

Corollary 1 (Existence of regular GCDs) *The subresultant S_d may not be a regular GCD of P, Q w.r.t T. However, there exists a splitting $T \rightarrow (T_1, \ldots, T_m)$ and a sequence $d_1, \ldots, d_m$ such that for each i in $1 \cdots m$, $d \leq d_i \leq q$ and S_{d_i} is a regular GCD of P, Q w.r.t. T_i.*

According to the definition of a regular GCD, S_{d_i} is a polynomial of positive degree d_i in x_{n+1} and its leading coefficient $s_{d_i} = \mathrm{lc}(S_{d_i}, x_{n+1})$ is regular modulo $\mathrm{sat}(T_i)$, which implies that S_{d_i} is of positive degree d_i modulo $\mathrm{sat}(T_i)$. In other words, regular GCD of positive degree exists in each branch of T.

We explain how to relax the assumption $R \in \mathrm{sat}(T)$ and thus obtain a general algorithm for the operation RegularGcd. The principle is straightforward. Let $R = \mathsf{res}(P, Q, y)$. We call $\mathsf{Regularize}(R, T)$ obtaining regular chains $T_1, \ldots, T_e$ such that $T \longrightarrow (T_1, \ldots, T_e)$. For each $1 \leq i \leq e$ we compute a regular GCD sequence of P and Q w.r.t. T_i as follows: If $R \in \mathrm{sat}(T_i)$ holds then we proceed as described above; otherwise $R \notin \mathrm{sat}(T_i)$ holds and the resultant R is actually a regular GCD of P and Q w.r.t. T_i by definition. Observe that when $R \in \mathrm{sat}(T_i)$ holds the subresultant chain of P and Q in y is used to compute their regular GCD w.r.t. T_i. This is one of the motivations for the implementation techniques described in Sections 5 and 6.

5 Subresultant chain computation

This section and the next one address implementation techniques and complexity issues. We describe our encoding of the subresultant chain of P, Q in $\mathbf{k}[x_1, \ldots, x_n][y]$. This representation is used in both our implementation and complexity results. For simplicity our analysis is restricted to the case where $\mathbf{k}$ is a finite field whose "characteristic is large enough". The case where $\mathbf{k}$ is the field $\mathbb{Q}$ of rational numbers could be handled in a similar fashion, with the necessary adjustments. We follow the notations introduced in Section 3. However we do not assume that $R = \mathsf{res}(P, Q, y)$ necessarily belongs to the saturated ideal of the regular chain T.

One motivation for the design of the techniques presented in this paper is the solving of systems of two equations, say $P = Q = 0$. Indeed, this can be seen as a fundamental operation in incremental methods for solving systems of polynomial equations, such as the one of [23]. We make two simple observations. Formula 1 p. 5 shows that solving this system reduces "essentially" to computing R and a regular GCD sequence of $P,\ Q$ modulo $\{R\}$, when R is not constant. This is particularly true when $n = 1$ since in this case the variety $V(H, P, Q)$ is likely to be empty for "generic" polynomials P, Q. The second observation is that, under the same genericity assumptions, a regular GCD G of P, Q w.r.t. $\{R\}$ is likely to exist and have degree one w.r.t. y. Therefore, once the subresultant chain of P, Q w.r.t. y is calculated, one can obtain G "essentially" at no additional cost. At the end of this section, we shall return to these observation and deduce complexity results from them.

The subresultant chain of P and Q are represented by homomorphic images: following [5], we evaluate $(x_1, \ldots, x_n)$ at sufficiently may points such that the subresultants of P and Q (regarded as univariate polynomials in $y = x_{n+1}$) can be computed by interpolation. To be more precise, we need some notations. Let d_i be the maximum of the degrees of P and Q in x_i, for all $i = 1, \ldots, n + 1$. Observe that $b_i := 2d_i d_{n+1}$ is an upper bound for the degree of R (or any subresultant of P and Q) in x_i, for all i. Let B be the product $(b_1+1) \cdots (b_n+1)$.

Specialization grid (SCube). We proceed by evaluation/interpolation; our sample points are chosen on an n-dimensional rectangular grid. We call *specialization grid* or simply *Scube* the data consisting of this grid and the values that the subresultant chain of P, Q takes at each point of on this grid. This is precisely how the subresultants of P, Q are encoded in our implementation. Of course, the validity of this approach requires that our evaluation points cancel no initials of P or Q. Even though finding such points deterministically is a difficult problem, this creates no issue in practice. Whenever possible (typically, over suitable finite fields), we choose roots of unity as sample points, so that we can use FFT (or van der Hoeven's Truncated Fourier Transform [13]); otherwise, standard fast evaluation/interpolation algorithms are used.

In order to reconstruct all subresultants of P and Q, from their SCube, one needs to perform $O(d_{n+1})$ evaluations and $O(d_{n+1}^2)$ interpolations. Since our sample points lie on a grid, the total cost (including the computation of the images of the subresultants on the grid) becomes

$$O\left(Bd_{n+1}^2 \sum_{i=1}^{n} \log(b_i)\right) \quad \text{or} \quad O\left(Bd_{n+1}^2 \sum_{i=1}^{n} \frac{\mathsf{M}(b_i)\log(b_i)}{b_i}\right),$$

depending on the choice of the sample points (see e.g. [25] for similar estimates).

Here, as usual, $\mathsf{M}(b)$ stands for the cost of multiplying univariate polynomials of degree less than b, see [11, Chap. 8]. Using the estimate $\mathsf{M}(b) \in O(b\log(b)\log\log(b))$ from [3], this respectively gives the bounds

$$O(d_{n+1}^2 B\log(B)) \quad \text{and} \quad O(d_{n+1}^2 B\log^2(B)\log\log(B)).$$

These estimates are far from optimal. A first important improvement consists in interpolating in the first place only the *leading coefficients* of the subresultants, and recover all other coefficients when needed. This is sufficient for the algorithm of Section 4. This idea brings the following result.

Lemma 5 *Constructing the SCube can be done within*

$$O(d_{n+1}^2 B + d_{n+1} B\log^2(B)\mathrm{loglog}(B))$$

operations in $\mathbf{k}$. *If multi-dimensional FFT can be used then this estimate becomes* $O(d_{n+1}^2 B + d_{n+1} B\log(B))$ *operations in* $\mathbf{k}$.

Another desirable improvement would consist in using fast arithmetic based on *Half-GCD* techniques [11], with the goal of reducing the total cost to $O^{\tilde{}}(d_{n+1}B)$, which is the best known bound for computing the resultant, or a given subresultant. However, as of now, we do not have such a result, due to the possible splittings.

We return now to the question of solving two equations. Our goal is to estimate the cost of computing the polynomials R and G in the context of Formula 1 p. 5. We propose an approach where the computation of G essentially comes for free, once R has been computed. This is a substantial improvement compared to traditional methods, such as [14,23], which compute G without recycling the intermediate calculations of R. With the above assumptions and notations, we saw that the resultant R can be computed in at most $O(d_{n+1}B\log(B)+d_{n+1}^2B)$ operations in $\mathbf{k}$. In many cases (typically, with random systems), G has degree one in $y = x_{n+1}$. Then, the GCD G can be computed within the same bound as the resultant. Besides, in this case, one can use the Half-GCD approach instead of computing all subresultants of P and Q. This leads to the following result in the bivariate case; we omit its proof here.

Corollary 2 *With* $n = 1$, *assuming that* $V(H, P, Q)$ *is empty, and assuming* $\deg(G, y) = 1$, *solving the input system* $P = Q = 0$ *can be done in* $O^{\sim}(d_2^2 d_1)$ *operations in* $\mathbf{k}$.

6 Regularity test in dimension zero

The operation Regularize specified in Section 2.1 is a core routine in methods computing triangular decompositions. It has been used in the algorithms presented in Section 4. Algorithms for this operation appear in [14,23].

The purpose of this section is to show how to realize efficiently this operation. For simplicity, we restrict ourselves to regular chains with zero-dimensional saturated ideals, in which case the separate operation of [14] and the regularize operation [23] are similar. We also restrict ourselves to reduced and normalized regular chains, which implies that these regular chains are reduced lexicographical Gröbner bases.

For such a regular chain T in $\mathbf{k}[\mathbf{x}]$ and a polynomial $p \in \mathbf{k}[\mathbf{x}]$ we denote by RegularizeDim0(p, T) the function call Regularize(p, T). In broad terms, it "separates" the points of $V(T)$ that cancel p from those which do not. The output is a set of regular chains $\{T_1, \ldots, T_e\}$ such that the points of $V(T)$ which cancel p are given by the T_i's modulo which p is null.

Algorithm 2 differs from those with similar specification in [14,23] by the fact that it creates opportunities for using modular methods and fast polynomial arithmetic. Our first trick is based on the following result (Theorem 1 in [4]): the polynomial p is invertible modulo T if and only if the iterated resultant of p with respect to T is non-zero. The correctness of Algorithm 2 follows from this result, the specification of the operation RegularGcd and an inductive process. Similar proofs appear in [14,23]. A complexity analysis of Algorithm 2, under some genericity assumptions, is reported at the end of this section.

The main novelty of Algorithm 2 is to employ the fast evaluation/interpolation strategy described in Section 5. In our implementation of Algorithm 2, at Line **6**, we compute the "Scube" representing the subresultant chain of q and C_v. This allows us to compute the resultant r and then to compute the regular GCDs (g, E) at Line **14** from the same "Scube". In this way, intermediate computations are recycled. Moreover, fast polynomial arithmetic is involved through the manipulation of the "Scube".

In Algorithm 2, a routine RegularizeInitialDim0 is called, whose specification is given below. See [23] for an algorithm. Briefly speaking, this routine splits a regular chain T into regular chains $T_1, \ldots, T_e$ according to a polynomial p such that for each $i = 1 \cdots e$ the polynomial p reduces modulo $\mathrm{sat}(T_i)$ to a constant polynomial or to a polynomial with a regular initial.

We shall now estimate the running time of Algorithm 2 under the following two genericity assumptions.

RegularizeDim0(p, T)

Input : T normalized reduced zero-dimensional regular chain and p polynomial, both in $\mathbf{k}[x_1, \ldots, x_n]$, with p reduced w.r.t. T.
Output : see the specification in Section 2.2.

```
1  Results ← ∅;
2  for (q, C) ∈ RegularizeInitDim0(P, T) do
3      if q ∈ k then Results ← {C} ∪ Results;
4      else
5          v ← mvar(q);
6          r ← res(q, C_v, v) ;
7          r ← NormalForm(r, C_<v);
8          for D ∈ RegularizeDim0(r, C_<v) do
9              s ← NormalForm(r, D);
10             if s ≠ 0 then
11                 U ← {D ∪ {C_v} ∪ C_>v} ;
12                 Results ← {C} ∪ Results;
13             else
14                 for (g, E) ∈ RegularGcd(q, C_v, D) do
15                     g ← NormalForm(g, E);
16                     U ← {E ∪ {g} ∪ D_>v};
17                     Results ← {C} ∪ Results;
18                     c ← NormalForm(quo(C_v, g), E);
19                     if deg(c, v) > 0 then
20                         Results ←
                           RegularizeDim0(q, E ∪ c ∪ C_>v) ∪ Results
21 return Results;
```

Algorithm 2: Regularize a polynomial

RegularizeInitDim0(p, T)

Input : T a normalized zero-dimensional regular chain and p a polynomial, both in $\mathbf{k}[x_1, \ldots, x_n]$
Output : A set of pairs $\{(p_i, T_i) \mid i = 1 \cdots e\}$, in which p_i is a polynomial and T_i is a regular chain, such that either p_i is a constant or its initial is regular modulo $\text{sat}(T_i)$, $p \equiv p_i \bmod \text{sat}(T_i)$ holds, and we have $T \longrightarrow (T_1, \ldots, T_e)$.

($\mathbf{H_1}$) T generates a radical ideal,
($\mathbf{H_2}$) none of the calls to RegularizeDim0 splits its second argument into several regular chains.

Ensuring that Hypothesis ($\mathbf{H_1}$) holds is standard. This is done by adapting to coefficients in products of fields squarefree part computation of univariate polynomial with coefficients in a field, see [23]. Hypothesis ($\mathbf{H_1}$) holds if T generates a maximal ideal. It is also likely to hold on a random dense input, as observed in our experimentation. Analyzing the running time of Algorithm 2 without Hypothesis ($\mathbf{H_2}$) leads to additional difficulties which can be handled using the techniques of [6], but this would be another paper.

In order to proceed with our analysis, we need some notations. We define $\mathrm{logp}(x) = \log_2(\max(2, x))$ and $\mathrm{llogp}(x) = \mathrm{logp}(\mathrm{logp}(x))$ for any real value x. Observe that for all a, b we have $\mathrm{logp}(ab) \leq \mathrm{logp}(a)\mathrm{logp}(b)$. Let d_i be the degree in x_i of the polynomial T_{x_i}. Let $s_1, \ldots, s_n$ be positive integers and let $s \in \mathbf{k}[x_1, \ldots, x_n]$ be a polynomial satisfying $\deg(s, x_i) < s_i$ for all $i = 1 \cdots n$. We denote by $\mathsf{NF}(s_1, \ldots, s_n, d_1, \ldots, d_n)$ an upper bound for the number of operations in $\mathbf{k}$ performed when computing the normal form of s w.r.t. T. If no confusion is possible, we simply write $\mathsf{NF}(s_1, \ldots, s_n)$ instead of $\mathsf{NF}(s_1, \ldots, s_n, d_1, \ldots, d_n)$. Next, we denote by $\mathsf{RZ}(d_1, \ldots, d_n)$ (resp. $\mathsf{M}_T(d_1, \ldots, d_n)$) an upper bound for the number of operations in $\mathbf{k}$ performed when computing $\mathsf{RegularizeDim0}(p, T)$ where p is reduced w.r.t. T (resp. when multiplying modulo $\langle T \rangle$ two polynomials reduced w.r.t. T). In [20] it is shown that there exists a constant $\mathsf{K} > 1$ such that

$$\mathsf{M}_T(d_1, \ldots, d_n) \leq 4^n \, \mathsf{K} \, D_n \, \mathrm{logp}(D_n) \mathrm{llogp}(D_n)$$

holds where $D_k = d_1 \cdots d_k$. By convention $D_0 = 1$.

Lemma 6 *With the above notations, we have:*

$$\mathsf{NF}(s_1, \ldots, s_n) \leq 5 \, \mathsf{K} \, \mathrm{logp}(\sigma) \, \mathrm{llogp}(\sigma) \, \mathrm{logp}(D_{n-1}) \, \mathrm{llogp}(D_{n-1}) \sum_{i=1}^{n} 4^{i-1} S_i D_{i-1},$$

where we define $\sigma = \max(s_1, \ldots, s_n)$ *and* $S_i = s_i \cdots s_n$, *for all* $i = 1 \cdots n$.

Proof. Let $c_0, \ldots, c_t$ be the coefficients of s w.r.t. x_n such that s writes $\sum_{i=0}^{t} c_i x_n^i$. To compute $\mathsf{NormalForm}(s, T)$ we start by computing s' which is $\sum_{i=0}^{t} c'_i x_n^i$ where c'_i is $\mathsf{NormalForm}(c_i, T_{<x_n})$. Since $t < s_n$, this first step costs at most $s_n \mathsf{NF}(s_1, \ldots, s_{n-1})$ operations in $\mathbf{k}$. Then, we compute the remainder in the Euclidean division of s' by T_{x_n} modulo $\langle T_{<x_n} \rangle$. Using the results of Chapter 9 in [11], this latter step amounts to at most $5\mathsf{M}(s_n)\mathsf{M}_T(d_1, \ldots, d_{n-1})$. This leads to the following inequality

$$\mathsf{NF}(s_1, \ldots, s_n) \leq s_n \mathsf{NF}(s_1, \ldots, s_{n-1}) + 5\mathsf{M}(s_n)\mathsf{M}_T(d_1, \ldots, d_{n-1}).$$

Unrolling this relation yields

$$\mathsf{NF}(s_1, \ldots, s_n) \leq 5 s_n \cdots s_2 \mathsf{M}(s_1) + \sum_{i=2}^{n} 5 s_n \cdots s_{i+1} \mathsf{M}(s_i) \mathsf{M}_T(d_1, \ldots, d_{i-1}).$$

Therefore, we have

$$\mathsf{NF}(s_1, \ldots, s_n) \leq 5\mathsf{K}\mathrm{logp}(\sigma)\,\mathrm{llogp}(\sigma)\,\mathrm{logp}(D_{n-1})\,\mathrm{llogp}(D_{n-1}) \sum_{i=1}^{n} 4^{i-1} S_i D_{i-1}.$$

□

Lemma 7 *With notations of Lemma 6, assuming $d_{n+1} \geq 2$ and $s_i = 2d_i d_{n+1}$ for all $i = 1 \cdots n$, we have*

$$\mathsf{NF}(s_1, \ldots, s_n) \leq 80\,\mathsf{K}\,n\,2^n\,d_{n+1}^n\,D_n\,\mathrm{logp}^2(D_n)\,\mathrm{llogp}^2(D_n).$$

Proof. We apply Lemma 6. First, we observe that $\mathrm{logp}(\sigma) \leq 2\,\mathrm{logp}(D_n)$ holds if $n > 1$. However, to cover $n = 1$, we use the estimate $\mathrm{logp}(\sigma) \leq 4\,\mathrm{logp}(D_n)$. Since $\mathrm{logp}(D_n) \geq 1$ holds, we deduce $\mathrm{llogp}(\sigma) \leq 4\,\mathrm{llogp}(D_n)$. Next, we observe that $S_i = (2d_{n+1})^{n-i+1} d_i \cdots d_n$ holds which brings

$$\sum_{i=1}^{n} 4^{i-1} S_i D_{i-1} = 2^n d_{n+1}^n D_n \sum_{i=1}^{n} (2/d_{n+1})^{i-1} \leq n\,2^n\,d_{n+1}^n\,D_n$$

and the conclusion follows. □

As in Section 5 we consider two polynomials $P, Q \in [x_1, \ldots, x_n, x_{n+1}]$ with positive degree in $y = x_{n+1}$ such that we have $0 < \deg(Q, x_{n+1}) \leq \deg(P, x_{n+1}) =: d_{n+1}$. We assume that the initials of P, Q are regular w.r.t. $\mathrm{sat}(T)$, that the resultant of P, Q w.r.t. x_{n+1} belongs to $\mathrm{sat}(T)$ and that all coefficients of P and Q w.r.t. x_{n+1} are reduced w.r.t. T. Let us denote by $\mathsf{SRC}(d_1, \ldots, d_n, d_{n+1})$ an upper bound for the number of operations in $\mathbf{k}$ necessary to construct the SCube of P, Q. It follows from Lemma 5 that there exists a constant $\mathsf{C} > 0$ such that

$$\mathsf{SRC}(d_1, \ldots, d_n, d_{n+1}) \leq \mathsf{C}\left(d_{n+1}^2 B_n + d_{n+1} B_n \mathrm{logp}^2(B_n)\mathrm{llogp}(B_n)\right) \quad (5)$$

where $B_n = 2^n d_{n+1}^n D_n$. Moreover, one can choose C such that each coefficient w.r.t. x_{n+1} of a subresultant of P and Q w.r.t. x_{n+1} can be interpolated within $\mathsf{C}\,B_n \mathrm{logp}^2(B_n)\mathrm{llogp}(B_n)$ operations in $\mathbf{k}$.

We denote by $\mathsf{GCD}(d_1, \ldots, d_n, d_{n+1})$ an upper bound for the number of operations in $\mathbf{k}$ performed when computing a regular GCD sequence of P, Q modulo $\mathrm{sat}(T)$. We have the following result.

Lemma 8 *Under Hypotheses* $(\mathbf{H_1})$ *and* $(\mathbf{H_2})$*, we have:*

$$\mathsf{GCD}(d_1, \ldots, d_{n+1}) \leq \mathsf{SRC}(d_1, \ldots, d_n, d_{n+1}) + \mathsf{RZ}(d_1, \ldots, d_n) + \frac{d_{n+1}(d_{n+1}+1)}{2}\left(\mathsf{C}\,B_n \mathrm{logp}^2(B_n)\mathrm{llogp}(B_n) + \mathsf{NF}(s_1, \ldots, s_n)\right)$$

where $s_i = 2d_id_{n+1}$ *for all* $i = 1 \cdots n$.

Proof. Recall that Hypothesis ($\mathbf{H_2}$) means that computations do not split. This implies that when Algorithm 1 calls $\mathsf{RegularizeDim0}(c_i, C)$ with a (reduced, normalized, zero-dimensional) regular chain C and with a polynomial c_i reduced w.r.t. C, then c_i is either null or invertible modulo $\langle C \rangle$. Consider now the *candidate search phase* (Lines **4** to **15**) in Algorithm 1. With the notations of this algorithm, consider an item $[i, C]$ at Line **5**. If c_i belongs to $\mathrm{sat}(C)$ then the whole subresultant S_i belongs to $\mathrm{sat}(C)$. This follows from Lemma 2 and the fact that $\mathrm{sat}(C)$ is radical (Hypothesis ($\mathbf{H_2}$)). If c_i does not belong to $\mathrm{sat}(C)$ then c_i is invertible modulo $\mathrm{sat}(C)$ and thus S_i is a candidate. This implies that, in the worst case, the *candidate search phase* is accomplished by:

- interpolating all subresultant of P and Q w.r.t. x_{n+1} from the SCube,
- computing the normal form of all these coefficients w.r.t. T,
- performing one regularity test of a polynomial which is not in $\langle T \rangle$.

Finally, Hypothesis ($\mathbf{H_1}$) together with Lemma 4 implies that the candidate is actually a regular GCD of P, Q modulo $\mathrm{sat}(T)$. Hence the *candidate check phase* of Algorithm 1 comes at no cost. The conclusion follows. □

Lemma 9 *Under Hypotheses* ($\mathbf{H_1}$) *and* ($\mathbf{H_2}$) *and assuming* $d_{n+1} \geq 2$*, we have*

$$\mathsf{GCD}(d_1, \ldots, d_{n+1}) \leq O(n^2\, 2^n)\, d_{n+1}^{n+2}\, D_n\, L_n + \mathsf{RZ}(d_1, \ldots, d_n),$$

where $L_n = \mathrm{logp}(n)\, \mathrm{logp}^2(d_{n+1})\, \mathrm{llogp}(d_{n+1})\, \mathrm{logp}^2(D_n)\, \mathrm{llogp}^2(D_n)$.

Proof. From Lemma 8 and Equation (5) we have

$$\begin{aligned} \mathsf{GCD}(d_1, \ldots, d_n, d_{n+1}) \leq\ & \mathsf{C}^{\frac{3(d_{n+1}+1)d_{n+1}}{2}} B_n \mathrm{logp}^2(B_n) \mathrm{llogp}(B_n) \\ & \mathsf{NF}(s_1, \ldots, s_n)^{\frac{(d_{n+1}+1)d_{n+1}}{2}} + \mathsf{RZ}(d_1, \ldots, d_n) \end{aligned} \tag{6}$$

We shall simplify the above inequality. Since $B_n = 2^n d_{n+1}^n D_n$ and $n \geq 1$, we deduce

$$\mathrm{logp}(B_n) \ \leq\ n + n\log_2(d_{n+1}) + \log_2(D_n) \ \leq\ 3\, n\, \mathrm{logp}(d_{n+1})\, \mathrm{logp}(D_n),$$

and

$$\mathrm{llogp}(B_n) \leq 2\, \mathrm{logp}(n)\, \mathrm{llogp}(d_{n+1})\, \mathrm{llogp}(D_n),$$

which leads to

$$B_n\, \mathrm{logp}^2(B_n)\, \mathrm{llogp}(B_n) \leq 18\, n^2\, 2^n\, d_{n+1}^n\, D_n\, L_n. \tag{7}$$

Next, we deduce from Lemma 7 that

$$\mathsf{NF}(s_1, \ldots, s_n) \leq 80\, \mathsf{K}\, n\, 2^n\, d_{n+1}^n\, D_n\, L_n \tag{8}$$

Using $\frac{1}{2}(d_{n+1}+1)d_{n+1} \leq d_{n+1}^2$ together with (6), (7) and (8) we obtain

$$\mathsf{GCD}(d_1, \ldots, d_n, d_{n+1}) \leq (54\,\mathsf{C}\,n + 80\,\mathsf{K})\; n\, 2^n\, d_{n+1}^{n+2}\, D_n\, L_n + \mathsf{RZ}(d_1, \ldots, d_n).$$

This completes the proof. □

Theorem 2 *Under Hypotheses* $(\mathbf{H_1})$ *and* $(\mathbf{H_2})$ *and assuming* $d_i \geq 2$ *for all* $i = 1 \cdots n$ *we have, for* $n \geq 2$

$$\mathsf{RZ}(d_1, \ldots, d_n) \leq O(n^2\, 2^{n-1})\, d_n^{n+1}\, D_{n-1}\, L_{n-1} + 2\,\mathsf{RZ}(d_1, \ldots, d_{n-1}) \tag{9}$$

which implies

$$\mathsf{RZ}(d_1, \ldots, d_n) \in O^{\sim}(2^n) \sum_{i=2}^{n} \left(i^2\, d_i^i\, D_i\right). \tag{10}$$

Proof. We follow Algorithm 2, which computes $\mathsf{RegularizeDim0}(p, T)$. Recall that the input polynomial p is reduced w.r.t T. Since we are looking for an upper bound for $\mathsf{RZ}(d_1, \ldots, d_n)$ we can assume that the main variable of p is x_n. Hypothesis $(\mathbf{H_2})$ implies that $\mathrm{init}(p)$ is invertible modulo $\langle T \rangle$ and thus that executing Line **2** amounts at most to $\mathsf{RZ}(d_1, \ldots, d_{n-1})$. Observe that at Line **5** we have $q = p$ and $C_v = T_{x_n}$. The next cost is at Lines **6** and **7** with the computation of the SCube of p and T_{x_n} w.r.t. x_n, the interpolation of their resultant r and the computation of $\mathsf{NormalForm}(r, T_{<x_n})$. We observe that this cost is included in (resp. dominated by) the estimate of $\mathsf{GCD}(d_1, \ldots, d_{n-1}, d_n)$ given by Lemma 8, if $\mathsf{NormalForm}(r, T_{<x_n}) = 0$ (resp. r is invertible modulo $\langle T_{<x_n} \rangle$). The next cost is at Line **8** with the call $\mathsf{RegularizeDim0}(r, T_{<x_n})$, amounting at most to $\mathsf{RZ}(d_1, \ldots, d_{n-1})$. Hypothesis $(\mathbf{H_2})$ implies that Line **9** comes at no cost. At this point either $r \notin \mathrm{sat}(T_{<x_n})$ holds and the algorithm terminates, or the next expense is at Line **14** which fits within $\mathsf{GCD}(d_1, \ldots, d_{n-1}, d_n)$. In this latter case, $(\mathbf{H_2})$ implies $\deg(g, x_n) = \deg(q, x_n)$ and no other computations take place. Finally, we obtain Relation (9) by virtue of Lemma 9. □

Corollary 3 *Under Hypotheses* $(\mathbf{H_1})$ *and* $(\mathbf{H_2})$ *and assuming* $d_i \geq 2$ *for all* $i = 1 \cdots n$ *we have, for* $n \geq 2$

$$\mathsf{GCD}(d_1, \ldots, d_n, d_{n+1}) \in O^{\sim}(n^2\, 2^n) d_{n+1}^{n+2} D_n + O^{\sim}(2^n) \sum_{i=2}^{n} \left(i^2\, d_i^i\, D_i\right). \tag{11}$$

Proof. The claim follows from Theorem 2 and Lemma 9. □

Essentially, Relation (11) depends "quadratically" on the product of the degrees $d_1, \ldots, d_n, d_{n+1}$. This is clear when $d_1 = \cdots = d_n = d_{n+1}$ holds. Moreover the "exponential factor" is only 2^n. In [6], the Authors provide an algorithm with the same specification as Algorithm 2. Under the same hypotheses, they

achieve a running estimate which depends "linearly" (up to logarithmic factors) on the product of the degrees $d_1, \ldots, d_n, d_{n+1}$. However, their "exponential factor" is of the form c^n where $\mathsf{c} \geq 700$. Since practical values for $d_1, \ldots, d_n, d_{n+1}$ are often below the hundreds, in particular for d_{n+1} with n large, this suggests that the algorithms presented in this paper are probably more suitable for implementation than those of [6].

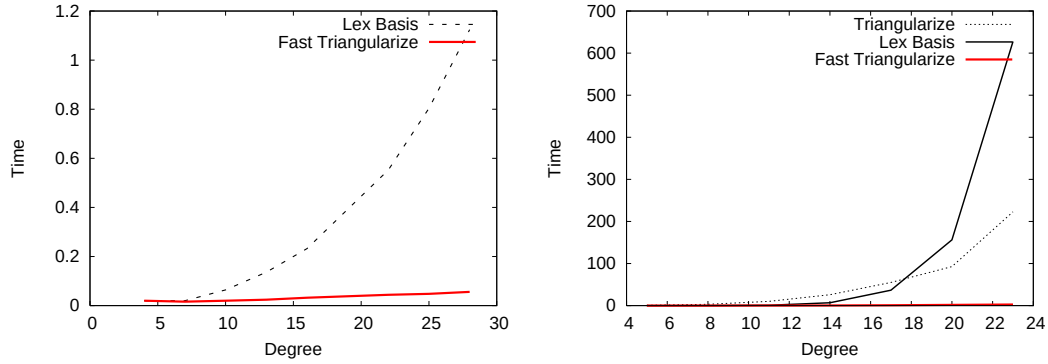
7 Experimentation

We have implemented in C language all the algorithms presented in the previous sections. The corresponding functions rely on the asymptotically fast arithmetic operations from our `modpn` library [19]. For this new code, we have also realized a MAPLE interface, called FastArithmeticTools, which is a new module of the `RegularChains` library [18].

In this section, we compare the performance of our `FastArithmeticTools` commands with MAPLE's and MAGMA's existing counterparts. For MAPLE, we use its latest release, namely version 13; For MAGMA we use Version `V2.15-4`, which is the latest one at the time of writing this paper. However, for this release, the MAGMA commands `TriangularDecomposition` and `Saturation` appear to be some time much slower than in Version `V2.14-8`. When this happens, we provide timings for both versions.

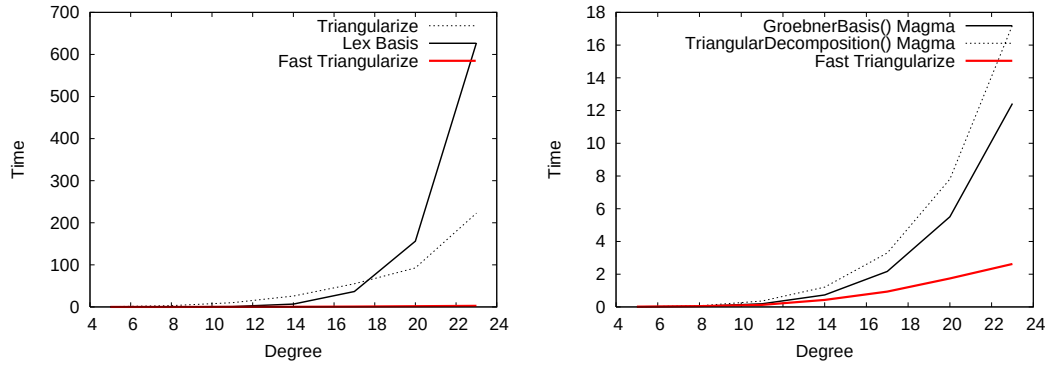
We have three test cases dealing respectively with the solving of bivariate systems, the solving of two-equation systems and the regularity testing of a polynomial w.r.t. a zero-dimensional regular chain. In our experimentation all polynomial coefficients are in a prime field whose characteristic is a 30-bit prime number. For each of our figure or table the degree is the total degree of any polynomial in the input system. All the benchmarks were conducted on a 64-bit Intel Pentium VI Quad CPU 2.40 GHZ machine with 4 MB L2 cache and 3 GB main memory.

For the solving of bivariate systems we compare the command `Triangularize` to the command `BivariateModularTriangularize` of the module `FastArithmeticTools`. Indeed both commands have the same specification for such input systems. Note that `Triangularize` is a high-level generic code which applies to any type of input system and which does not rely on fast polynomial arithmetic or modular methods. On the contrary, `BivariateModularTriangularize` is specialized to bivariate systems (see Corollary 2 in Section 5) is mainly implemented in C and is supported by the `modpn` library. `BivariateModularTriangularize` is an instance of a more general fast algorithm called `FastTriangularize`; we use this second name in our figures.



(a) bivariate systems (b) trivariate systems

Fig. 2. Timing for generic dense systems



(a) bivariate systems (b) bivariate systems

Fig. 3. Timing for highly non-equiprojectable systems

Since a triangular decomposition can be regarded as a "factored" lexicographic Gröbner basis we also benchmark the computation of such bases in MAPLE and MAGMA. Figure 2a compares `FastTriangularize` and (lexicographic) `Groebner:-Basis` in MAPLE on generic dense input systems. On the largest input example the former solver is about 20 times faster than the latter. For the solving of systems with two equations, we compare `FastTriangularize` with `GroebnerBasis` in MAGMA. On Figure 2b these two solvers are simply referred as MAGMA and MAPLE. For this benchmark the input are generic dense trivariate systems.

Figure 3a compares `FastTriangularize` and (lexicographic) `Groebner:-Basis` on highly non-equiprojectable dense input systems; for these systems the number of equiprojectable components is about half the degree of the variety. At the total degree 23 our solver is approximately 100 times faster than `Groebner:-Basis`. Figure 3b compares `FastTriangularize`, `GroebnerBasis` in MAGMA and `TriangularDecomposition` in MAGMA on the same set of highly non-equiprojectable dense input systems. Once again our solver outperforms its competitors.

d_1	d_2	Reg.	Fast Reg.	Magma	d_1	d_2	Reg.	Fast Reg.	Magma
2	2	0.052	0.016	0.000	20	38	69.876	0.776	3.660
4	6	0.236	0.016	0.010	22	42	107.154	0.656	6.600
6	10	0.760	0.016	0.010	24	46	156.373	1.036	10.460
8	14	1.968	0.020	0.050	26	50	220.653	2.172	17.110
10	18	4.420	0.052	0.090	28	54	309.271	1.640	25.900
12	22	8.784	0.072	0.220	30	58	434.343	2.008	42.600
14	26	15.989	0.144	0.500	32	62	574.923	4.156	57.000
16	30	27.497	0.208	0.990	34	66	746.818	6.456	104.780
18	34	44.594	0.368	1.890					

Fig. 4. bivariate random dense

d_1	d_2	d_3	Reg.	Fast Reg.	Magma	d_1	d_2	d_3	Reg.	Fast Reg.	Magma
2	2	3	0.240	0.008	0.000	8	14	21	168.910	2.204	8.250
3	4	6	1.196	0.020	0.020	9	16	24	332.036	14.764	23.160
4	6	9	4.424	0.032	0.030	10	18	27	>1000	21.853	61.560
5	8	12	12.956	0.148	0.200	11	20	30	>1000	57.203	132.240
6	10	15	33.614	0.360	0.710	12	22	33	>1000	102.830	284.420
7	12	18	82.393	1.108	2.920						

Fig. 5. trivariate random dense

Figures 4, 5 and 6 compare our fast regularity test algorithm (Algorithm 2) with the `RegularChains` library `Regularize` and its MAGMA counterpart. More precisely, in MAGMA, we first saturate the ideal generated by the input zero-dimensional regular chain T with the input polynomial P using the `Saturation` command. Then the `TriangularDecomposition` command decomposes the output produced by the first step. The total degree of the input i-th polynomial in T is d_i. For Figure 4 and Figure 5, the input T and P are randomly generated such that the intermediate computations do not split. In this non-splitting cases, our fast `Regularize` algorithm is significantly faster than the other commands. For Figure 6, the input T and P are constructed such that many intermediate computations need to split. In this case, our fast `Regularize` algorithm is slightly slower than its MAGMA counterpart, but still much faster than the generic (non-modular and non-supported by `modpn`) `Regularize` command of the `RegularChains` library. The slow down w.r.t. the MAGMA code is due to the (large) overheads of the C - MAPLE interface, see [19] for details.

8 Conclusion

The concept of a regular GCD extends the usual notion of polynomial GCD from polynomial rings over fields to polynomial rings modulo saturated ideals of regular chains. Regular GCDs play a central role in triangular decomposi-

d_1	d_2	d_3	Reg.	Fast Reg.	V2.15-4	V2.14-8
2	2	3	0.184	0.028	0.000	0.000
3	4	6	0.972	0.060	0.000	0.010
4	6	9	3.212	0.092	>1000	0.030
5	8	12	8.228	0.208	>1000	0.150
6	10	15	21.461	0.888	807.850	0.370
7	12	18	51.751	3.836	>1000	1.790
8	14	21	106.722	9.604	>1000	2.890
9	16	24	207.752	39.590	>1000	10.950
10	18	27	388.356	72.548	>1000	19.180
11	20	30	703.123	138.924	>1000	56.850
12	22	33	>1000	295.374	>1000	76.340

Fig. 6. trivariate dense with many splittings

tion methods. Traditionally, regular GCDs are computed in a top-down manner, by adapting standard PRS techniques (Euclidean Algorithm, subresultant algorithms, ...).

In this paper, we have examined the properties of regular GCDs of two polynomials w.r.t a regular chain. The theoretical results in Section 3 show that one can proceed in a bottom-up manner, leading to Algorithm 1 in Section 4. This has three benefits described in Sections 5 and 6. First, this algorithm is well-suited to employ modular methods and fast polynomial arithmetic. Secondly, we avoid the repetition of (potentially expensive) intermediate computations. Lastly, we avoid, as much as possible, computing modulo regular chains and use polynomial computations over the base field instead, while controlling expression swell. The running time estimates of Section 6. and experimental results of Section 7 illustrate the high efficiency of our algorithms.

9 Acknowledgement

The authors would like to thank our friend Changbo Chen, who pointed out that Lemma 3 in an earlier version of this paper was incorrect.

References

[1] M.F. Atiyah and L. G. Macdonald. *Introduction to Commutative Algebra*. Addison-Wesley, 1969.

[2] F. Boulier, F. Lemaire, and M. Moreno Maza. Well known theorems on triangular systems and the D5 principle. In *Proc. of* Transgressive Computing 2006, Granada, Spain, 2006.

[3] D.G. Cantor and E. Kaltofen. On fast multiplication of polynomials over arbitrary algebras. *Acta Informatica*, 28:693–701, 1991.

[4] C. Chen, O. Golubitsky, F. Lemaire, M. Moreno Maza, and W. Pan. *Comprehensive Triangular Decomposition*, volume 4770 of *LNCS*, pages 73–101. Springer Verlag, 2007.

[5] G.E. Collins. The calculation of multivariate polynomial resultants. *Journal of the ACM*, 18(4):515–532, 1971.

[6] X. Dahan, M. Moreno Maza, É. Schost, and Y. Xie. On the complexity of the D5 principle. In *Proc. of* Transgressive Computing 2006, Granada, Spain, 2006.

[7] J. Della Dora, C. Dicrescenzo, and D. Duval. About a new method for computing in algebraic number fields. In *Proc. EUROCAL 85 Vol. 2*, Springer-Verlag, 1985.

[8] L. Ducos. *Effectivité en théorie de Galois. Sous-résultants.* PhD thesis, Université de Poitiers, 1997.

[9] D. Duval. *Questions Relatives au Calcul Formel avec des Nombres Algébriques.* Université de Grenoble, 1987. Thèse d'État.

[10] M'hammed El Kahoui. An elementary approach to subresultants theory. *J. Symb. Comp.*, 35:281–292, 2003.

[11] J. von zur Gathen and J. Gerhard. *Modern Computer Algebra.* Cambridge University Press, 1999.

[12] T. Gómez Díaz. *Quelques applications de l'évaluation dynamique.* PhD thesis, Université de Limoges, 1994.

[13] J. van der Hoeven. The Truncated Fourier Transform and applications. In *ISSAC'04*, pages 290–296. ACM, 2004.

[14] M. Kalkbrener. A generalized euclidean algorithm for computing triangular representations of algebraic varieties. *J. Symb. Comp.*, 15:143–167, 1993.

[15] D. Lazard. A new method for solving algebraic systems of positive dimension. *Discr. App. Math*, 33:147–160, 1991.

[16] D. Lazard. Solving zero-dimensional algebraic systems. *J. Symb. Comp.*, 15:117–132, 1992.

[17] F. Lemaire, M. Moreno Maza, W. Pan, and Y. Xie. When does (T) equal Sat(T)? In *Proc. ISSAC'20008*, pages 207–214. ACM Press, 2008.

[18] F. Lemaire, M. Moreno Maza, and Y. Xie. The `RegularChains` library. In Ilias S. Kotsireas, editor, Maple Conference 2005, pages 355–368, 2005.

[19] X. Li, M. Moreno Maza, R. Rasheed, and É. Schost. The modpn library: Bringing fast polynomial arithmetic into maple. In *MICA'08*, 2008.

[20] X. Li, M. Moreno Maza, and É. Schost. Fast arithmetic for triangular sets: From theory to practice. In *ISSAC'07*, pages 269–276. ACM Press, 2007.

[21] X. Li, M. Moreno Maza, and W. Pan. Computations modulo regular chains. In *ISSAC'09*, pages 239–246. ACM Press, 2009.

[22] B. Mishra. *Algorithmic Algebra*. Springer, New York, 1993.

[23] M. Moreno Maza. On triangular decompositions of algebraic varieties. Technical Report TR 4/99, NAG Ltd, Oxford, UK. Presented at the MEGA-2000 Conference, Bath, England.

[24] M. Moreno Maza and R. Rioboo. Polynomial gcd computations over towers of algebraic extensions. In *Proc. AAECC-11*, pages 365–382. Springer, 1995.

[25] V. Y. Pan. Simple multivariate polynomial multiplication. *J. Symb. Comp.*, 18(3):183–186, 1994.

[26] C. K. Yap. *Fundamental Problems in Algorithmic Algebra*. Princeton University Press, 1993.