




# A Modular Algorithm for Computing the Intersection of a One-Dimensional Quasi-Component and a Hypersurface

Alexander Brandt<sup>(✉)</sup> , Juan Pablo González Trochez, Marc Moreno Maza, and Haoze Yuan

Department of Computer Science, The University of Western Ontario,  
London, Canada

{abrandt5, jgonza55, hyuan46}@uwo.ca, moreno@csd.uwo.ca

**Abstract.** Computing triangular decompositions of polynomial systems can be performed incrementally with a procedure named *Intersect*. This procedure computes the common zeros (encoded as regular chains) of a quasi-component and a hypersurface. As a result, decomposing a polynomial system into regular chains can be achieved by repeated calls to the *Intersect* procedure. Expression swell in *Intersect* has long been observed in the literature. When the regular chain input to *Intersect* is of positive dimension, intermediate expression swell is likely to happen due to spurious factors in the computation of resultants and subresultants.

In this paper, we show how to eliminate this issue. We report on its implementation in the polynomial system solver of the BPAS (Basic Polynomial Algebra Subprogram) library. Our experimental results illustrate the practical benefits. The new solver can process various systems which were previously unsolved by existing implementations of regular chains. Those implementations were either limited by time, memory consumption, or both. The modular method brings orders of magnitude speedup.

**Keywords:** Polynomial system solving · Triangular decomposition · Modular method · Regular chains · Intersection · Quasi-component

## 1 Introduction

Since the early works of Ritt [35], Wu [42], and Yang and Zhang [45], the Characteristic Set Method has been extended and improved by many researchers. This effort has produced more powerful decomposition algorithms, and now applies to different types of polynomial systems or decompositions: parametric algebraic systems [18, 22, 44], differential systems [8, 19, 26], difference systems [24], unmixed decompositions and primary decomposition [38] of polynomial ideals, intersection multiplicities [31], cylindrical algebraic decomposition [16, 28], quantifier elimination [17], parametric [44] and non-parametric [14] semi-algebraic systems. Today, triangular decomposition algorithms are available in several software packages [4, 13, 40, 41, 43]. Moreover, they provide

back-engines for computer algebra system front-end solvers, such as MAPLE's `solve` command.

Despite of their successful application in various areas (automatic theorem proving, dynamical systems, program verification, to name a few), solvers based on triangular decompositions are sometimes put to challenge with input polynomial systems that appear to be easy to solve by other methods, based on Gröbner bases. Of course, one should keep in mind that different solvers may have different specifications, not always easy to compare. Nevertheless, for certain classes of systems, say zero-dimensional systems, one can expect that a triangular decomposition on one hand, and the computation of a lexicographical Gröbner basis (followed by the application of Lazard's `Lextriangular` algorithm [29]) on the other, produce essentially the same thing.

While the development of modular methods for computing Gröbner bases took off in the 1980's thanks to Traverso [39] and Faugère [23], with follow-up works by Arnold [1] and others, the development of such methods for triangular decompositions started only in 2005 with the paper [21] by Dahan, Moreno Maza, Schost, Wu and Xie. This latter method computes a triangular decomposition  $\Delta$  of a zero-dimensional polynomial system  $V(F)$  over the rational numbers by

1. first computing a triangular decomposition, say  $\Delta_p$ , of that system modulo a sufficiently large prime number  $p$ ;
2. transforming  $\Delta_p$  into a canonical triangular decomposition of  $V(F \bmod p)$ , called the equiprojectable decomposition,  $E_p$  of  $V(F \bmod p)$ ; and
3. finally, lifting  $E_p$  (using the techniques of Schost [37]) into the equiprojectable decomposition of  $V(F)$ .

Hence, this method helps to control the effect of expression swell at the level of the numerical coefficients, which resulted in a significant efficiency improvement on a number of famous test systems. However, this modular method has no benefits on expression swell when expression swell manifests as an (unnecessary) inflation on the number of terms. This phenomenon is generally caused by the so-called extraneous or spurious factors in resultants, which have been studied in the case of Dixon resultants [27]. Most algorithms for computing triangular decompositions compute *iterated resultants*, either explicitly or implicitly.

In broad terms, the iterated resultant  $\text{res}(f, T)$  between  $f$  and a regular chain<sup>1</sup>  $T \subseteq \mathbf{k}[X_1 < \dots < X_n]$  encodes conditions for the hypersurface  $V(f)$  and the quasi-component  $W(T)$  to have a non-empty intersection.

To be precise, we recall some of the results in Sect. 6 of [15]. Assume that  $T$  is a zero-dimensional regular chain. We denote by  $V_M(T)$  the multiset of the zeros of  $T$ , where each zero of  $T$  appears a number of times equal to its local multiplicity as defined in Chap. 4 of [20]. If  $T$  is normalized, that is, the initial of every polynomial in  $T$  is a constant, then we have:

$$\text{res}(f, T) = \prod_{\alpha \in V_M(T)} f(\alpha).$$

---

<sup>1</sup> See Sect. 2 for a review of regular chain theory, including definitions of the terms quasi-component, initial, etc.

This *Poisson Formula* tells us that, if  $T$  is normalized, then  $\text{res}(f, T)$  is “fully meaningful”. In other words, it does not contain extraneous factors. Now, let us relax the fact that  $T$  is normalized. For  $i = 1, \dots, n$ , we denote respectively by  $t_i, h_i, r_i$ : (1) the polynomial of  $T$  whose main variable is  $X_i$ , (2) the initial of  $t_i$ , (3) the iterated resultant  $\text{res}(\{t_1, \dots, t_{i-1}\}, h_i)$ . In particular, we have  $r_1 = h_1$ . We also define: (1)  $e_n = \deg(f, X_n)$ , (2)  $f_i = \text{res}(\{t_{i+1}, \dots, t_n\}, f)$ , for  $0 \leq i \leq n - 1$ , (3)  $e_i = \deg(f_i, x_i)$ , for  $1 \leq i \leq n - 1$ . Then,  $\text{res}(T, f)$  is given by:

$$h_1^{e_1} \left( \prod_{\beta_1 \in V_M(t_1)} h_2(\beta_1) \right)^{e_2} \cdots \left( \prod_{\beta_{n-1} \in V_M(t_1, \dots, t_{n-1})} h_n(\beta_{n-1}) \right)^{e_n} \left( \prod_{\alpha \in V_M(T)} f(\alpha) \right)$$

From that second Poisson formula, we can see that all factors but the rightmost one (that is, the one from the first Poisson formula) are extraneous. Indeed, in the intersection  $V(f) \cap W(T)$  there are no points cancelling the initials  $h_2, \dots, h_n$ .

These observations generalize to regular chains of positive dimension (just seeing the field  $\mathbf{k}$  as a field of rational functions) and can explain how the calculation of iterated resultants can cause expression swells in triangular decomposition algorithms. To deal with that problem, the authors of [15] study a few trivariate systems consisting of a polynomial  $f(X_1, X_2, X_3)$  and a regular chain  $T = \{t_2(X_1, X_2), t_3(X_1, X_2, X_3)\}$ . They compute  $\text{res}(T, f)$  by

1. specializing  $X_1$  at sufficiently many well-chosen values  $a$ ,
2. computing  $R(a) := \text{res}(N(a), f(a))$  where  $f(a) = f(a, X_2, X_3)$  and  $N(a)$  is the normalized regular chain generating the ideal  $\langle t_2(a, X_2), t_3(a, X_2, X_3) \rangle$  in  $\mathbf{k}[X_2, X_3]$ , and
3. combining the  $R(a)$ 's and applying rational function reconstruction.

The numerator of the reconstructed fraction is essentially the desired non-extraneous factor of  $\text{res}(T, f)$ .

In this paper, we extend the ideas of [15] so that one can actually compute  $V(f) \cap W(T)$  and not just obtain conditions on the existence of those common solutions for  $f$  and  $T$ . Computing such intersections is the core routine of the incremental triangular decomposition method initiated by Lazard in [28] and further developed by Chen and Moreno Maza [15, 33]. Consequently, we have implemented the proposed techniques and measured the benefits that they bring to the solver presented in [4].

We stress the fact that our objective is to optimize the **Intersect** algorithm [15] for computing intersections of the form  $V(f) \cap W(T)$ . Moreover, one of the main applications of our work in this area is to support algorithms in differential algebra, as in the articles [10, 11]. With the challenges of that application<sup>2</sup> in mind and noting the success obtained in applying regular chain theory to differential algebra, our approach to optimize the **Intersect** algorithm must remain free of (explicit) Gröbner basis computations.

---

<sup>2</sup> The differential ideal generated by finitely many differential polynomials is generally not finitely generated, when regarded as an algebraic ideal.

We observe that if Gröbner basis computations are to be used to support triangular decompositions, efficient algorithms exist since the 1990's. As shown in [34], applying Lazard's `Lextriangular` to the lexicographical Gröbner basis  $G(F)$  of a zero-dimensional polynomial ideal  $\langle F \rangle$  produces a triangular decomposition of the algebraic variety  $V(F)$  in a time which is negligible comparing to that of computing  $G(F)$ . This efficiency follows from the structure of a lexicographical Gröbner basis as stated by the Gianni-Kalkbrener theorem [29].

The presentation of our modular method for computing  $V(f) \cap W(T)$  is dedicated to the case where  $T$  is one-dimensional. The cases where  $T$  is of dimension higher than one are work in progress but not reported here. Our approach to the design of such a modular method is as follows.

In Sect. 3, we identify hypotheses under which  $V(f) \cap W(T)$  is given by a single zero-dimensional regular chain  $C$ , such that  $V(f) \cap W(T) = W(C)$  holds. We call those hypotheses *genericity assumptions* because  $C$  is *shape lemma* in the sense of [7]. In Sect. 4, we develop a modular method which computes  $C$ , if the genericity assumptions hold, and detects which assumption does not hold otherwise. One intention of that algorithm is that, whenever a genericity assumption fails, one should be able to recycle the computations performed by the modular method, in order to finish the computations, see Sect. 5 for details. Section 6 gathers some notes about a preliminary implementation of the modular algorithm presented in Sect. 4. The experimentation, reported in Sect. 7, contains very promising results. Indeed, our solver based on this modular method can process various systems which were previously unsolved by our solver (without the modular method) and unsolved by the `RegularChains` library of MAPLE.

## 2 Preliminaries

This section is a short review of concepts from the theory of regular chains and triangular decompositions of polynomial systems. Details can be found in [15]. This paper also relies on the theory of subresultants and we refer the unfamiliar reader to the concise preliminaries section of [5].

**Polynomials.** Throughout this paper, let  $\mathbf{k}$  be a perfect field,  $\mathbf{K}$  be its algebraic closure, and  $\mathbf{k}[X]$  be the polynomial ring over  $\mathbf{k}$  with  $n$  ordered variables  $X = X_1 < \dots < X_n$ . Let  $p \in \mathbf{k}[X] \setminus \mathbf{k}$ . Denote by  $\text{mvar}(p)$ ,  $\text{init}(p)$ , and  $\text{mdeg}(p)$ , respectively, the greatest variable appearing in  $p$  (called the *main variable* of  $p$ ), the leading coefficient of  $p$  w.r.t.  $\text{mvar}(p)$  (called the *initial* of  $p$ ), and the degree of  $p$  w.r.t.  $\text{mvar}(p)$  (called the *main degree* of  $p$ ). For  $F \subseteq \mathbf{k}[X]$ , we denote by  $\langle F \rangle$  and  $V(F)$  the ideal generated by  $F$  in  $\mathbf{k}[X]$  and the algebraic set of  $\mathbf{K}^n$  consisting of the common roots of the polynomials of  $F$ , respectively.

**Triangular Sets.** Let  $T \subseteq \mathbf{k}[X]$  be a *triangular set*, that is, a set of non-constant polynomials with pairwise distinct main variables. Denote by  $\text{mvar}(T)$  the set of main variables of the polynomials in  $T$ . A variable  $v \in X$  is called *algebraic* w.r.t.  $T$  if  $v \in \text{mvar}(T)$ , otherwise it is said *free* w.r.t.  $T$ . For  $v \in \text{mvar}(T)$ , we denote by  $T_v$  and  $T_v^-$  (resp.  $T_v^+$ ) the polynomial  $f \in T$  with  $\text{mvar}(f) = v$  and the

polynomials  $f \in T$  with  $\text{mvar}(f) < v$  (resp.  $\text{mvar}(f) > v$ ). Let  $h_T$  be the product of the initials of the polynomials of  $T$ . We denote by  $\text{sat}(T)$  the *saturated ideal* of  $T$ : if  $T = \emptyset$  holds, then  $\text{sat}(T)$  is defined as the trivial ideal  $\langle 0 \rangle$ , otherwise it is the ideal  $\langle T \rangle : h_T^\infty$ . The *quasi-component*  $W(T)$  of  $T$  is defined as  $V(T) \setminus V(h_T)$ . For  $f \in \mathbf{k}[X]$ , we define  $Z(f, T) := V(f) \cap W(T)$ . The Zariski closure of  $W(T)$  in  $\mathbf{K}^n$ , denoted by  $\overline{W}(T)$ , is the intersection of all algebraic sets  $V \subseteq \mathbf{K}^n$  such that  $W(T) \subseteq V$  holds; moreover we have  $\overline{W}(T) = V(\text{sat}(T))$ . For  $f \in \mathbf{k}[X]$ , we denote by  $\text{res}(f, T)$  the *iterated resultant* of  $f$  w.r.t.  $T$ , that is: if  $f \in \mathbf{k}$  or  $T = \emptyset$  then  $f$  itself, else  $\text{res}(f, T_v, v, T_v^-)$  if  $v \in \text{mvar}(T)$  and  $v = \text{mvar}(f)$  hold, or  $\text{res}(f, T_v^-)$  otherwise.

**Regular Chains, Triangular Decomposition.** A triangular set  $T \subseteq \mathbf{k}[X]$  is a *regular chain* if either  $T$  is empty, or letting  $v$  be the largest variable occurring in  $T$ , the set  $T_v^-$  is a regular chain, and the initial of  $T_v$  is regular (that is, neither zero nor a zero divisor) modulo  $\text{sat}(T_v^-)$ . Let  $H \subseteq \mathbf{k}[X]$ . The pair  $[T, H]$  is a *regular system* if each polynomial in  $H$  is regular modulo  $\text{sat}(T)$ . The *dimension* of  $T$ , denoted by  $\text{dim}(T)$ , is by definition, the dimension of its saturated ideal and, as a property, equals  $n - |T|$ , where  $|T|$  is the number of elements of  $T$ . If  $T$  has dimension zero, then  $T$  generates  $\text{sat}(T)$  and we have  $V(T) = W(T)$ .

The saturated ideal  $\text{sat}(T)$  enjoys important properties, in particular the following, proved in [9]. Let  $U_1, \dots, U_d$  be all the free variables of  $T$ . Then  $\text{sat}(T)$  is unmixed of dimension  $d$ . Moreover, we have  $\text{sat}(T) \cap \mathbf{k}[U_1, \dots, U_d] = \langle 0 \rangle$ . Another property is the fact that a polynomial  $p$  belongs to  $\text{sat}(T)$  if and only if  $p$  reduces to 0 by pseudo-division w.r.t.  $T$ , see [6]. Last but not least, a polynomial  $p$  is regular modulo  $\text{sat}(T)$  if and only if we have  $\text{res}(p, T) \neq 0$ .

**Specialization and Border Polynomial.** Let  $[T, H]$  be a regular system of  $\mathbf{k}[X]$ . Let  $U = U_1, \dots, U_d$  be the free variables of  $T$ . Let  $a = (a_1, \dots, a_d) \in \mathbf{K}^d$ . We say that  $[T, H]$  *specializes well* at  $a$  if:

- (i) for each  $t \in T$  the polynomial  $\text{init}(t)$  is not zero modulo the ideal  $\langle U_1 - a_1, \dots, U_d - a_d \rangle$ ; and
- (ii) the image of  $[T, H]$  modulo  $\langle U_1 - a_1, \dots, U_d - a_d \rangle$  is a regular system.

Let  $B_{T,H}$  be the primitive and square-free part of the product of all  $\text{res}(h, T)$  for  $h \in H \cup \{h_T\}$ . We call  $B_{T,H}$  the *border polynomial* of  $[T, H]$ . From the specialization property of sub-resultants, one derives the following [32]: The system  $[T, H]$  specializes well at  $a \in \mathbf{K}^d$  if and only if  $B_{T,H}(a) \neq 0$  holds.

**Normalized Regular Chain.** The regular chain  $T \subseteq \mathbf{k}[X]$  is said to be *normalized* if, for every  $v \in \text{mvar}(T)$ , none of the variables occurring in  $\text{init}(T_v)$  is algebraic w.r.t.  $T_v^-$ . Let  $d = \text{dim}(T)$ ,  $Y = \text{mvar}(T)$ , and  $U = U_1, \dots, U_d$  be  $X \setminus Y$ . Then,  $T$  normalized means that for every  $t \in T$  we have  $\text{init}(t) \in \mathbf{k}[U]$ . It follows that if  $T$  is normalized, then  $T$  is a lexicographical Gröbner basis of the ideal that  $T$  generates in  $\mathbf{k}(U)[Y]$  (that is, over the field  $\mathbf{k}(U)$  of rational functions), and we denote by  $\text{nf}(p, T)$  the normal form of a polynomial  $p \in \mathbf{k}(U)[Y]$  w.r.t.  $T$  as a Gröbner basis. Importantly, if  $T$  is normalized and has dimension zero, then  $\text{init}(t) \in \mathbf{k}$  for every  $t \in T$ .

**Regular GCD.** Let  $T \subseteq \mathbf{k}[X]$  be a regular chain. Let  $i$  be an integer with  $1 \leq i \leq n$ . Let  $p, t \in \mathbf{k}[X] \setminus \mathbf{k}$  be polynomials with the same main variable  $X_i$ , and  $g \in \mathbf{k}$  or  $g \in \mathbf{k}[X]$  with  $\text{mvar}(g) \leq X_i$ . Assume that:

1.  $X_i > X_j$  holds for all  $X_j \in \text{mvar}(T)$ ; and
2. both  $\text{init}(p)$  and  $\text{init}(t)$  are regular w.r.t.  $\text{sat}(T)$ .

For the residue class ring  $\mathbf{k}[X_1, \dots, X_{i-1}]/\sqrt{\text{sat}(T)}$ , denote its total ring of fractions as  $\mathcal{A}$ . Note that  $\mathcal{A}$  is isomorphic to a direct product of fields. We say that  $g$  is a *regular GCD* of  $p, t$  w.r.t.  $T$  whenever the following conditions hold:

- (G<sub>1</sub>) the leading coefficient of  $g$  in  $X_i$  is invertible in  $\mathcal{A}$ ;
- (G<sub>2</sub>)  $g$  belongs to the ideal generated by  $p$  and  $t$  in  $\mathcal{A}[X_i]$ ; and
- (G<sub>3</sub>) if  $\deg(g, X_i) > 0$ , then  $g$  divides both  $p$  and  $t$  in  $\mathcal{A}[X_i]$ , that is, both  $\text{prem}(p, g)$  and  $\text{prem}(t, g)$  belong to  $\sqrt{\text{sat}(T)}$ .

When Conditions (G<sub>1</sub>), (G<sub>2</sub>), (G<sub>3</sub>) and  $\deg(g, X_i) > 0$  hold:

- (G<sub>4</sub>) if  $\text{mdeg}(g) = \text{mdeg}(t)$ , then  $\sqrt{\text{sat}(T \cup t)} = \sqrt{\text{sat}(T \cup g)}$  and  $W(T \cup t) \subseteq Z(h_g, T \cup t) \cup W(T \cup g) \subseteq \overline{W(T \cup t)}$ ;
- (G<sub>5</sub>) if  $\text{mdeg}(g) < \text{mdeg}(t)$ , let  $q = \text{pquo}(t, g)$ , then  $T \cup q$  is a regular chain and we have
  - (a)  $\sqrt{\text{sat}(T \cup t)} = \sqrt{\text{sat}(T \cup g)} \cap \sqrt{\text{sat}(T \cup q)}$  and
  - (b)  $W(T \cup t) \subseteq Z(h_g, T \cup t) \cup W(T \cup g) \cup W(T \cup q) \subseteq \overline{W(T \cup t)}$ ;
- (G<sub>6</sub>)  $W(T \cup g) \subseteq V(p)$ ; and
- (G<sub>7</sub>)  $V(p) \cap W(T \cup t) \subseteq W(T \cup g) \cup V(p, h_g) \cap W(T \cup t) \subseteq V(p) \cap \overline{W(T \cup t)}$ .

**Intersect and Regularize.** Let  $p \in \mathbf{k}[X]$  and let  $T \subseteq \mathbf{k}[X]$  be a regular chain. The function `Intersect`( $p, T$ ) computes regular chains  $T_1, \dots, T_e$  such that:  $V(p) \cap W(T) \subseteq W(T_1) \cup \dots \cup W(T_e) \subseteq V(p) \cap \overline{W(T)}$ . The function call `Regularize`( $p, T$ ) computes regular chains  $T_1, \dots, T_e$  such that: (1) for each  $i = 1, \dots, e$ , either  $p \in \text{sat}(T_i)$  holds or  $p$  is regular w.r.t.  $\text{sat}(T_i)$ ; and (2) we have  $\overline{W(T)} = \overline{W(T_1)} \cup \dots \cup \overline{W(T_e)}$ , and  $\text{mvar}(T) = \text{mvar}(T_i)$  holds for  $i = 1, \dots, e$ .

**Triangular Decomposition.** Let  $F \subseteq \mathbf{k}[X]$ . The regular chains  $T_1, \dots, T_e$  of  $\mathbf{k}[X]$  form a *triangular decomposition* of  $V(F)$  in the sense of Kalkbrener (resp. Wu and Lazard) whenever we have  $V(F) = \bigcup_{i=1}^e \overline{W(T_i)}$  (resp.  $V(F) = \bigcup_{i=1}^e W(T_i)$ ). Hence, a triangular decomposition of  $V(F)$  in the sense of Wu and Lazard is necessarily a triangular decomposition of  $V(F)$  in the sense of Kalkbrener, while the converse is not true. Note that a triangular decomposition can thus be computed from repeated calls to `Intersect`; see [15].

### 3 Genericity Assumptions

Let  $\mathbf{k}$  be a field of characteristic zero or a prime field of *sufficiently large* characteristic, where that latter condition will be specified later. Let  $f, t_2, \dots, t_n \in \mathbf{k}[X]$  be non-constant polynomials in the ordered variables  $X = X_1 < \dots < X_n$ .

Assume that  $T := \{t_2, \dots, t_n\}$  is a regular chain with  $\text{mvar}(t_i) = X_i$  for  $2 \leq i \leq n$ . Assume also  $\text{mvar}(f) = X_n$ . Our goal is to compute the intersection  $V(f) \cap W(T)$  in the sense of the function call `Intersect(f, T)`, as specified in Sect. 2. We shall show that, under some assumptions, one can compute a regular chain  $C \subseteq \mathbf{k}[X]$  so that  $C$  is zero-dimensional and we have:  $V(f) \cap W(T) = W(C)$ .

For convenience, we define  $r_n := f$ . Regarding  $t_n$  and  $r_n$  as polynomials in  $(\mathbf{k}[X_1, \dots, X_{n-1}])[X_n]$ , let  $S(t_n, r_n, X_n)$  be the subresultant chain of  $t_n$  and  $r_n$ , if  $\text{mdeg}(t_n) \geq \text{mdeg}(r_n)$ , or the subresultant chain of  $r_n$  and  $t_n$  otherwise. Let  $S_0(t_n, r_n, X_n)$  and  $S_1(t_n, r_n, X_n)$  be the subresultants of index 0 and 1 from  $S(t_n, r_n, X_n)$ . We let

$$r_{n-1} := S_0(t_n, r_n, X_n) \quad \text{and} \quad g_n := S_1(t_n, r_n, X_n).$$

Continuing in this manner, for  $2 \leq i \leq n - 1$ , let  $S(t_i, r_i, X_i)$  be the subresultant chain of  $t_i$  and  $r_i$  (resp.  $r_i$  and  $t_i$ ) regarded as polynomials in  $(\mathbf{k}[X_1, \dots, X_{i-1}])[X_i]$  if  $\text{mdeg}(t_i) \geq \text{mdeg}(r_i)$  (resp.  $\text{mdeg}(t_i) < \text{mdeg}(r_i)$ ) holds. Let  $S_0(t_i, r_i, X_i)$  and  $S_1(t_i, r_i, X_i)$  be the subresultants of index 0 and 1 from  $S(t_i, r_i, X_i)$ . We let

$$r_{i-1} := S_0(t_i, r_i, X_i) \quad \text{and} \quad g_i := S_1(t_i, r_i, X_i).$$

To make the problem generic, we assume the following:

- Hypothesis 1:**  $r_i \notin \mathbf{k}$  and  $\text{mvar}(r_i) = X_i$ , for  $1 \leq i \leq n - 1$  (1)
- Hypothesis 2:**  $g_i \notin \mathbf{k}$ , for  $2 \leq i \leq n$ , (2)
- Hypothesis 3:**  $C := \{\bar{s}, g_2, \dots, g_n\}$  is a regular chain, (3)
- Hypothesis 4:**  $(\forall i \in \{2, \dots, n\}) \text{res}(\text{init}(t_i), \{\bar{s}, g_2, \dots, g_{i-1}\}) \neq 0$ , (4)

where  $\bar{s}$  is the squarefree part of  $s := r_1$ , that is,  $s/\text{gcd}(s, \text{der}(s))$ . Hypothesis 3 has a number of consequences which, essentially, rephrase the fact that  $C$  is a regular chain. Proposition 1 gathers those consequences. Building on that, Proposition 2 yields Eq. (5) which plays a key role in our method for computing `Intersect(f, T)`.

**Proposition 1.** *The polynomials  $\bar{s}, g_2, \dots, g_n$  are non-constant and have main variables  $X_1, X_2, \dots, X_n$ , respectively. Moreover, the initial of  $g_i$  is invertible modulo the ideal  $\langle \bar{s}, g_2, \dots, g_{i-1} \rangle$  generated by  $\bar{s}, g_2, \dots, g_{i-1}$  in  $\mathbf{k}[X_1, \dots, X_{i-1}]$ .*

Hypothesis 4 expresses the fact that the initial of the polynomial  $t_i$  is invertible modulo the ideal  $\langle \bar{s}, g_2, \dots, g_{i-1} \rangle$ , for  $i = 2 \cdots n$ . We note that from Hypothesis 4, the set  $\{\bar{s}, g_2, \dots, g_{i-1}, t_i\}$  is also a regular chain, for  $2 \leq i \leq n$ .

**Proposition 2.** *Fix an integer  $i$  such that  $2 \leq i \leq n$  holds. Then, the polynomial  $g_i$  is a regular GCD of  $r_i$  and  $t_i$  modulo the regular chain  $\{\bar{s}, g_2, \dots, g_{i-1}\}$ . Moreover, we have:*

$$V(\bar{s}, g_2, \dots, g_{i-1}, r_i, t_i) = V(\bar{s}, g_2, \dots, g_{i-1}, g_i). \tag{5}$$

*Proof.* We first prove that  $g_i$  is a regular GCD of  $r_i$  and  $t_i$  modulo the regular chain  $\{\bar{s}, g_2, \dots, g_{i-1}\}$ . Since  $\{\bar{s}, g_2, \dots, g_{i-1}, g_i\}$  is a regular chain, Property  $(G_1)$  of a regular GCD clearly holds. We prove  $(G_2)$ . Subresultant theory tells us that there exist polynomials  $u_i, v_i \in \mathbf{k}[X_1, \dots, X_i]$  so that we have:  $u_i r_i + v_i t_i = g_i$ . Let  $\mathcal{A}_i$  be the total ring of fractions of  $\mathbf{k}[X_1, \dots, X_i]/\langle \bar{s}, g_2, \dots, g_i \rangle$ . Since  $\bar{s}$  is squarefree and since  $\text{mdeg}(g_2) = \dots = \text{mdeg}(g_{i-1}) = 1$ , the ring  $\mathcal{A}_{i-1}$  is actually a direct product of fields which tells us that  $g_i$  is the GCD (in the sense of a Euclidean domain) of  $r_i$  and  $t_i$  over each of those fields. Therefore, Property  $(G_2)$  holds. In particular, both  $r_i$  and  $t_i$  belong to the ideal generated by  $g_i$  in  $\mathcal{A}_{i-1}[X_i]$ . Thus, there exist polynomials  $q_{r_i}, q_{t_i} \in \mathcal{A}_{i-1}[X_i]$  so that the following hold in  $\mathcal{A}_{i-1}[X_i]$ :  $r_i = q_{r_i} g_i$  and  $t_i = q_{t_i} g_i$ . Every polynomial  $p \in \mathcal{A}_{i-1}[X_i]$  can be written as the fraction of a polynomial  $n \in \mathbf{k}[X_1, \dots, X_i]$  over a polynomial  $d \in \mathbf{k}[X_1, \dots, X_{i-1}]$  so that  $d$  is invertible modulo  $\langle \bar{s}, g_2, \dots, g_{i-1} \rangle$ . Therefore, there exist polynomials in  $\mathcal{A}_{i-1}[X_i]$ , that we denote again  $q_{r_i}$  and  $q_{t_i}$  for convenience, so that the following hold in  $\mathbf{k}[X_1, \dots, X_i]$ :  $r_i \equiv q_{r_i} g_i \pmod{\langle \bar{s}, g_2, \dots, g_{i-1} \rangle}$  and  $t_i \equiv q_{t_i} g_i \pmod{\langle \bar{s}, g_2, \dots, g_{i-1} \rangle}$ . From the above, it is clear that  $g_i$  pseudo-divides (actually divides) both  $r_i$  and  $t_i$  modulo  $\langle \bar{s}, g_2, \dots, g_{i-1} \rangle$ . Therefore, Property  $(G_3)$  holds and we have proved that  $g_i$  is a regular GCD of  $r_i$  and  $t_i$  modulo the regular chain  $\{\bar{s}, g_2, \dots, g_{i-1}\}$ . The second claim of this proposition follows from the first one and Lemma 1.

**Lemma 1.** *Fix an integer  $i$  such that  $2 \leq i \leq n$  holds. Let  $\hat{g}_i \in \mathbf{k}[X_1, \dots, X_i]$  be a non-constant polynomial with  $\text{mvar}(\hat{g}_i) = X_i$ . Assume that  $\hat{g}_i$  is a regular GCD of  $r_i$  and  $t_i$  modulo the regular chain  $\{\bar{s}, g_2, \dots, g_{i-1}\}$ . Then, we have:*

$$V(\bar{s}, g_2, \dots, g_{i-1}, r_i, t_i) = V(\bar{s}, g_2, \dots, g_{i-1}, \hat{g}_i). \tag{6}$$

*Proof.* We denote by  $T_i$  the regular chain  $\{\bar{s}, g_2, \dots, g_{i-1}, t_i\}$ . It follows from Property  $(G_7)$  of a regular GCD that:  $V(r_i) \cap W(T_i) \subseteq W(\{\bar{s}, g_2, \dots, g_{i-1}, \hat{g}_i\}) \cup V(r_i, h_{\hat{g}_i}) \cap W(T_i) \subseteq V(r_i) \cap \overline{W(T_i)}$ . Since  $h_{\hat{g}_i}$ , the initial of  $\hat{g}_i$ , is invertible modulo  $\langle \bar{s}, g_2, \dots, g_{i-1} \rangle$ , we have:  $V(r_i, h_{\hat{g}_i}) \cap W(T_i) = \emptyset$ . Since  $V(T_i)$  and  $V(\{\bar{s}, g_2, \dots, g_{i-1}, \hat{g}_i\})$  are both zero-dimensional, we have:  $V(\bar{s}, g_2, \dots, g_{i-1}, t_i) = W(T_i) = \overline{W(T_i)}$  and  $V(\bar{s}, g_2, \dots, g_{i-1}, \hat{g}_i) = W(\{\bar{s}, g_2, \dots, g_{i-1}, \hat{g}_i\})$ . Therefore, we have:  $V(r_i, \bar{s}, g_2, \dots, g_{i-1}, t_i) = V(\bar{s}, g_2, \dots, g_{i-1}, \hat{g}_i)$ .

Theorem 1 tells us that, under our genericity assumptions, the result of  $\text{Intersect}(f, T)$  is given by the regular chain  $C = \{\bar{s}, g_2, \dots, g_n\}$ .

**Theorem 1.** *With our four Hypotheses 1, 2, 3 and 4, we have:*

$$V(f, t_2, \dots, t_n) = V(\bar{s}, g_2, \dots, g_n). \tag{7}$$

*Proof.* This follows immediately from Proposition 2 and Lemma 2.

**Lemma 2.** *For each integer  $i$ , such that  $2 \leq i \leq n$  holds, let  $\hat{g}_i \in \mathbf{k}[X_1, \dots, X_i]$  be a non-constant polynomial with  $\text{mvar}(\hat{g}_i) = X_i$  so that  $\hat{g}_i$  is a regular GCD of  $r_i$  and  $t_i$  w.r.t. the regular chain  $\{\bar{s}, \hat{g}_2, \dots, \hat{g}_{i-1}\}$ . Then, we have:*

$$V(f, t_2, \dots, t_n) = V(\bar{s}, \hat{g}_1, \dots, \hat{g}_n). \tag{8}$$



**Algorithm 1.** GenericIntersectDimOne

---

**Require:**  $(f, T)$  as in Theorem 1. Recall:  $f \notin \mathbf{k}$  and  $\text{mvar}(f) = X_n$ .

**Ensure:**  $C$  as in Theorem 1.

```

1:  $r_n := f$ 
2: for  $i := n \dots 1$  do
3:    $r_{i-1} := S_0(t_i, r_i, X_i)$ 
4:    $g_i := S_1(t_i, r_i, X_i)$ 
5:   if  $r_{i-1} \in \mathbf{k}$  or  $\text{mvar}(r_{i-1}) \neq X_{i-1}$  then
6:     throw HYPOTHESIS 1 NOT MET
7:   if  $g_i \in \mathbf{k}$  then
8:     throw HYPOTHESIS 2 NOT MET
9:    $\bar{s} := \text{squareFreePart}(r_1)$ 
10:   $C := \{\bar{s}, g_2, \dots, g_n\}$ 
11:  if  $C$  is not a regular chain then
12:    throw HYPOTHESIS 3 NOT MET
13:  for  $i := 2 \dots n$  do
14:    if  $h_i$  is not regular w.r.t.  $C$  then
15:      throw HYPOTHESIS 4 NOT MET
16:  return  $C$ 

```

---

*Proof.* Since  $r_n = f$  and since  $r_{i-1}$  belongs to the ideal generated by  $r_i$  and  $t_i$ , we have:  $V(f, t_2, \dots, t_n) = V(r_1, t_2, r_2, \dots, t_n, r_n)$ . Since  $\bar{s}$  is the squarefree part of  $s = r_1$ , we also have:  $V(f, t_2, \dots, t_n) = V(\bar{s}, t_2, r_2, \dots, t_n, r_n)$ . With repeated application of Lemma 1, we deduce:  $V(f, t_2, \dots, t_n) = V(\bar{s}, \hat{g}_1, \dots, \hat{g}_n)$ .

Algorithm 1 summarizes the results of this section. Note that Algorithm 1 computes  $\text{Intersect}(f, T)$  only if Hypotheses 1, 2, 3, 4 hold, and throws an exception otherwise. The general task of computing  $\text{Intersect}(f, T)$  can be achieved by the algorithms presented in [15]. In fact, these exceptions can be caught by a wrapper algorithm, which can then call the general  $\text{Intersect}$  procedure. Moreover, one can attach to these exceptions the data already computed by Algorithm 1 so that the wrapper algorithm can avoid unnecessary computations. We will return to the handling of the exceptions of Algorithm 1 in Sect. 5.

## 4 The Modular Method

We use the same notations as in Sect. 3. The objective of this section is to turn Algorithm 1 into a modular algorithm where:

1. we evaluate  $f$  and  $T$  at sufficiently many values of  $X_1$  so that:
  - (a)  $T$  specializes well at  $X_1 = a$  to a zero-dimensional regular chain  $T(a)$ ,
  - (b)  $T(a)$  is replaced with a normalized regular chain  $N_a$  generating the same ideal,
  - (c) the images of  $g_n, \dots, g_2, r_1$  at  $X_1 = a$  are computed efficiently; and
2. the polynomials  $g_n, \dots, g_2, r_1$  are reconstructed from their images by means of interpolation and rational function reconstruction.

Let  $r_i(a)$  be the polynomial  $r_i$  evaluated at  $X_1 = a$  and  $t_i(a)$  be the polynomial from  $N_a$  with main variable  $X_i$ . The benefit of this modular algorithm is that the computation of the subresultants  $S_0(t_i(a), r_i(a), X_i)$  and  $S_1(t_i(a), r_i(a), X_i)$  avoid the expression swell described in Sect. 1. Indeed, the regular chain  $N_a$  is normalized. This modular algorithm leads to the *usual* questions:

1. Can all computed modular images be combined in order to retrieve the desired result, or are there some specializations that must be discarded?
2. If so, how do we detect those specializations that must be discarded?
3. How many modular images do we need in order to obtain the desired result?

We detail the answers to these three questions in the following three subsections, respectively. Luckily, there are only finitely many *bad specializations* which must be discarded.

#### 4.1 The Fumber of Bad Specializations is Finite

Let  $a \in \mathbf{k}$  and let  $\Phi_a$  be the evaluation homomorphism from  $\mathbf{k}[X_1, \dots, X_n]$  to  $\mathbf{k}[X_2, \dots, X_n]$  which evaluates  $X_1$  at  $a$ . Recall that  $C$  stands for  $\{\bar{s}, g_2, \dots, g_n\}$ . Assume that  $a$  is not a root of the border polynomial  $B_C \in \mathbf{k}[X_1]$  of  $C$ . Therefore, for  $2 \leq i \leq n$ , the polynomial  $\Phi_a(t_i)$  is not constant and has main variable  $X_i$ . Moreover, the set  $\{\Phi_a(t_2), \dots, \Phi_a(t_n)\}$  is a zero-dimensional regular chain in  $\mathbf{k}[X_2, \dots, X_n]$ . Let  $S(\Phi_a(t_i), \Phi_a(r_i), X_i)$  be the subresultant chain of  $\Phi_a(t_i)$  and  $\Phi_a(r_i)$  regarded as polynomials in  $(\mathbf{k}[X_2, \dots, X_{i-1}])[X_i]$ . From this subresultant chain, let  $S_0(\Phi_a(t_i), \Phi_a(r_i), X_i)$  and  $S_1(\Phi_a(t_i), \Phi_a(r_i), X_i)$  be the subresultants of index 0 and 1.

**Proposition 3.** *With Hypothesis 1, there exists a finite subset  $D(f, T) \subseteq \mathbf{k}$  such that, for all  $a \notin D(f, T)$ , for all  $2 \leq i \leq n$ , we have:*

$$\Phi_a(g_i) = S_1(\Phi_a(t_i), \Phi_a(r_i), X_i), \text{ and } \Phi_a(r_1) = S_0(\Phi_a(t_2), \Phi_a(r_2), X_2).$$

*Proof.* Fix  $i \in \mathbb{N}$  such that  $2 \leq i \leq n$ . From Hypothesis 1, we have  $r_i \notin \mathbf{k}$  and  $\text{mvar}(r_i) = X_i$ . Using the lexicographical term order induced by  $X_2 < \dots < X_i$ , let  $c_{i-1}$  be the leading coefficient of  $r_i$  regarded as a multivariate polynomial in  $\mathbf{k}[X_1][X_2, \dots, X_i]$ , such that  $c_{i-1} \in \mathbf{k}[X_1]$ . If  $a$  is not a root of  $c_{i-1}$  then  $\Phi_a(r_i)$  and  $r_i$  have the same degree in  $X_i$ . Since  $B_T(a) \neq 0$ , the polynomials  $\Phi_a(t_i)$  and  $t_i$  have the same degree in  $X_i$  too. It follows from the specialization property of subresultants that  $\Phi_a(g_i)$  and  $S_1(\Phi_a(t_i), \Phi_a(r_i), X_i)$  are equal. Therefore, the desired set is:  $D(f, T) = \{a \in \mathbf{k} \mid (B_T \cdot c_1 \cdots c_{n-1})(a) = 0\}$ , which is finite.

#### 4.2 Number of Bad Specializations and Other Degree Estimates

We start by giving an estimate of the cardinality of  $D(f, T)$  based on considerations directly derived from subresultant theory. This estimate is pessimistic and, in a second phase, we will revisit it to derive a modular algorithm computing the regular chain  $C = \{\bar{s}, g_2, \dots, g_n\}$ , as stated in Theorem 1.

Let  $d_i^{(n)}$  be the maximum of the degrees of  $f, t_n, \dots, t_2$  w.r.t.  $X_i$ , for  $1 \leq i \leq n$ . Using the determinantal formulation of subresultants, it follows that the degree of any subresultant of  $S(t_n, r_n, X_n)$  w.r.t.  $X_i$ , for  $1 \leq i \leq n-1$ , is bounded over by  $\deg(t_n, X_n) \deg(r_n, X_i) + \deg(r_n, X_n) \deg(t_n, X_i) \leq 2d_n^{(n)} d_i^{(n)} =: d_i^{(n-1)}$ . Using again the determinantal formulation, it follows that the degree of any subresultant of  $S(t_{n-1}, r_{n-1}, X_{n-1})$  w.r.t.  $X_i$  for  $1 \leq i \leq n-2$ , is bounded over by  $\deg(t_{n-1}, X_{n-1}) \deg(r_{n-1}, X_i) + \deg(r_{n-1}, X_{n-1}) \deg(t_{n-1}, X_i)$ , yielding

$$d_i^{(n-2)} := d_{n-1}^{(n)} d_i^{(n-1)} + d_{n-1}^{(n-1)} d_i^{(n)} = 2d_n^{(n)} d_i^{(n)} (d_i^{(n)} + d_{n-1}^{(n)}).$$

Continuing, the degree of any subresultant of  $S(t_{n-j}, r_{n-j}, X_{n-j})$  w.r.t.  $X_i$  for  $1 \leq i \leq n-j$ , is bounded above by  $d_i^{(n-j-1)} = d_{n-j}^{(n)} d_i^{(n-j)} + d_{n-j}^{(n-j)} d_i^{(n)}$ . To obtain a concise result, let  $d$  be the maximum of  $d_1^{(n)}, \dots, d_n^{(n)}$ . Then, we have  $d_i^{(n-1)} = 2d^2$ ,  $d_i^{(n-2)} = 4d^3$  and  $d_i^{(n-j-1)} = 2^{j+1} d^{j+2}$ , for  $0 \leq j \leq n-2$ .

Returning to the polynomial  $B_T \cdot c_1 \cdots c_{n-1}$ , we are now ready to estimate its degree. First, we note that  $\deg(c_{n-j-1}) \leq d_i^{(n-j-1)}$ , thus we have  $\deg(c_{n-j-1}) \leq 2^{j+1} d^{j+2}$ . Second, let  $h_i$  be the initial of  $t_i$ , for  $2 \leq i \leq n$ . The border polynomial  $B_T$  of the regular chain  $T$  is the product of the iterated resultants  $\text{res}(h_i, T)$ , for  $2 \leq i \leq n$ . Observe that, in the above discussion, the degree estimates  $d_i^{(n-j-1)}$  remain valid when we replace  $f$  by each of  $h_2, \dots, h_n$ . Therefore, we have:  $\deg(B_T) = \deg(\text{res}(h_2, T)) + \dots + \deg(\text{res}(h_n, T)) \leq (n-1)d_1^{(1)} = (n-1)2^{n-1}d^n$ . Finally, we deduce:  $\deg(B_T c_1 \cdots c_{n-1}) \leq (n-1)2^{n-1}d^n + 2^{n-1}d^n + \dots + 2d^2 \leq n2^n d^{n+1}$ .

**Proposition 4.** *With the hypotheses and notations of Proposition 3, the cardinality of  $D(f, T)$  is at most  $n2^n d^{n+1}$ , where  $d$  is the maximum partial degree of  $f, t_2, \dots, t_n$  in any variable  $X_1, \dots, X_n$ .*

Of course, this estimate is not sharp, particularly if the product of the partial degrees  $d_1^{(n)}, \dots, d_n^{(n)}$  exceeds the total degree of either  $f$  or  $t_n$ . Therefore, in order to design a modular method for computing the regular chain  $C = \{\bar{s}, g_2, \dots, g_n\}$  of Theorem 1, by means of an evaluation and interpolation strategy, we take advantage of the Bézout inequality (see Theorem 3 in [36]). Since  $V(f, t_2, \dots, t_n)$  is a zero-dimensional affine variety, the number of its elements is bounded over by the product of the total degrees of the polynomials  $f, t_2, \dots, t_n$ , that we denote by  $B(f, t_2, \dots, t_n)$ . Thus, the degree of the univariate polynomial  $\bar{s} \in \mathbf{k}[X_1]$  cannot exceed  $B(f, t_2, \dots, t_n)$ .

Furthermore, assume that the call  $\text{Intersect}(f, T)$  (with  $T = \{t_2, \dots, t_n\}$ ) was made as part of the triangular decomposition of a zero-dimensional system, say  $\{f_1, \dots, f_m\}$ . Then, one can use the Bézout bound  $B(f_1, \dots, f_m)$  instead of  $B(f, t_2, \dots, t_n)$ , since the former is likely to be (much) smaller than the latter.

In fact, any bound  $B$  on the number of points of  $V(f_1, \dots, f_m)$  can be used as an upper bound for  $\deg(\bar{s})$ . Moreover, our experimentation suggests that the degrees of the univariate polynomials  $c_1, \dots, c_{n-1}$  are not likely to exceed the degree of  $r_1$ . Hence, the number of specializations  $X_1 = a$  which do not cancel

the border polynomial  $B_C$  but cancel one of  $c_1, \dots, c_{n-1}$  are likely to be bounded over by  $(n-1)B$ . Therefore, using  $nB+1$  specializations  $X_1 = a$  is likely to be sufficient for computing  $\bar{s}$ , assuming that we have a practically efficient criterion for avoiding the specialization cancelling  $B_C$ . This latter observation leads us to the algorithm of Sect. 4.3. In fact, we shall see that, in practice, the quantity  $nB+1$  can often be reduced to  $2B+1$  or  $3B+1$ , even when  $n > 3$  holds.

### 4.3 A Modular Algorithm

In addition to the strategy presented in Sect. 4, the other key ingredients of our modular algorithm are the following ones: (1) Monagan's probabilistic strategy for computing resultants via evaluation and interpolation [30], (2) the *small prime* modular algorithm for computing the GCD of two univariate polynomials over  $\mathbb{Z}$ , see Chap. 6 in [25], and (3) rational function reconstruction, see Sect. 5.7 in [25].

Algorithm 2 takes as input the same arguments  $f$  and  $T$  as Algorithm 1. In addition, Algorithm 2 takes three other arguments  $B, s, D$  which are positive integers with the following respective roles:

1.  $B$  is an estimate of the degree of  $r_1$ .
2.  $e$  controls the behavior of Monagan's probabilistic strategy: once  $2B+e+1$  images (of the polynomials  $g_n, \dots, g_2, r_1$ ) are computed then the recombination of the first  $2B+1$  images is compared to the recombination of the first  $2B+e+1$ ; if they are equal, then rational function reconstruction is attempted. If rational function reconstruction fails, then  $e$  more images are collected and the next comparison uses the first  $2B+e+1$  and the first  $2B+2e+1$  images, and so on.
3.  $D$  is an estimate for the number of bad specializations defined in Sect. 4.1.

As we shall see, if  $B$  is an upper bound for the degree of  $r_1$ , and if  $D$  is an upper bound for the number of bad specializations, then the algorithm is deterministic, otherwise it is probabilistic.

In practice, a smaller  $B$  and a small  $e$  makes the algorithm check for termination (in the sense Monagan's probabilistic strategy) more frequently, which may have an impact on performance, positive or negative. In practice, if  $B$  is believed to be a sharp estimate for  $\deg(r_1)$ , then  $e$  can be small, even a small percentage of  $B$ , without negative performance impact. Similarly, a smaller  $D$  makes the algorithm check earlier whether  $C$  has the required properties, that is, whether Hypotheses 2, 3, 4 hold or not. This may also have an impact on performance, positive or negative. In practice, if  $B$  is believed to be a sharp estimate for  $\deg(r_1)$ , then  $D$  can be small, say a percentage of  $B$ .

Algorithm 2 uses two simple sub-procedures specified below:

- `InitializeImageCollection` initializes  $\mathcal{A}$  and  $\mathcal{G}$  to the empty list, and  $d$  to a list of  $n$  zeros.  $\mathcal{A}$  will store the evaluation points and  $\mathcal{G}$  the corresponding images of  $g_n, \dots, g_2, r_1$ .

---

**Algorithm 2.** ModularGenericIntersectDimOne

---

**Require:**  $(f, T, B, e, D)$ , where  $f, T$  are as in Theorem 1 with  $f \notin \mathbf{k}$  and  $\text{mvar}(f) = X_n$ ,  $B$  is a positive integer which estimates  $\deg(r_1)$ ,  $e$  is a positive integer, and  $D$  estimates the number of bad specializations.

**Ensure:**  $C$  as in Theorem 1, provided Hypotheses 1, 2, 3 and 4 are met, otherwise an exception is raised.

```

1:  $a := \text{Random}()$ ;  $\mathcal{P} := \{a\}$  ▷ a random element of  $\mathbf{k}$  used as a seed
2:  $M := 2B + 1$  ▷ Twice the bound is necessary for rational function reconstruction
3:  $c := 0$  ▷ counts the number of specializations used so far
4:  $b := 0$  ▷ counts the number of bad specializations met so far
5:  $(\mathcal{A}, \mathcal{G}, d) := \text{InitializeImageCollection}(f, T)$ 
6:  $C_M := \{\}$ ;  $C_{M+e} := \{\}$ 
7: while  $c < M + e + D$  do
8:    $(a, T(a), f(a), \mathcal{P}) := \text{FindCandidateSpecialization}(f, T, \mathcal{P})$ 
9:    $c := c + 1$ 
10:   $i := n$ 
11:   $r_i(a) := f(a)$ 
12:   $N_a := \text{Normalize}(T(a))$  ▷ normalize the regular chain
13:   $t_i(a) := \text{Polynomial}(X_i, N_a)$  ▷ The poly. of  $N_a$  with main var.  $X_i$ 
14:  while  $i > 1$  do
15:     $r_{i-1}(a) := S_0(t_i(a), r_i(a), X_i)$ 
16:    if  $r_{i-1} \in \mathbf{k}$  or  $\text{mvar}(r_{i-1}) < X_{i-1}$  then ▷ Bad specialization or Hypothesis 1 not met
17:       $b := b + 1$ ; Goto LINE 8
18:    if  $\#\mathcal{A} > 0$  and  $\deg(r_{i-1}(a), X_{i-1}) > d[i - 1]$  then ▷ Every specialization in  $\mathcal{A}$  is bad
19:       $b := b + \#\mathcal{A}$ ; Goto LINE 5
20:    if  $\#\mathcal{A} > 0$  and  $\deg(r_{i-1}(a), X_{i-1}) < d[i - 1]$  then ▷ The specialization  $X_1 = a$  is bad
21:       $b := b + 1$ ; Goto LINE 8
22:     $d[i - 1] = \deg(r_{i-1}(a), X_{i-1})$ 
23:     $g_i(a) := S_1(t_i(a), r_i(a), X_i)$ 
24:     $i := i - 1$ 
25:   $\mathcal{G} := \text{Append}(\mathcal{G}, [g_n(a), \dots, g_2(a), r_1(a)])$ 
26:   $\mathcal{A} := \text{Append}(\mathcal{A}, a)$ 
27:  if  $\#\mathcal{A} = M$  and  $C_M = \{\}$  then
28:     $C_M := \text{Interpolate}(\mathcal{A}, \mathcal{G}, X_1)$  ▷ Recover  $X_1$  in  $g_n, \dots, g_2, r_2$ 
29:  if  $\#\mathcal{A} = M + e$  and  $C_{M+e} = \{\}$  then
30:     $C_{M+e} := \text{Interpolate}(\mathcal{A}, \mathcal{G}, X_1)$ 
31:  if  $C_M = C_{M+e} \neq \{\}$  and  $c > D$  then ▷ If  $M$  and  $M + e$  images produce the same recombination and those are expected to have the correct degrees
32:     $C := \text{RationalFunctionReconstruction}(C_M, \mathcal{A}, X_1)$ 
33:    if  $C \neq \text{Failure}$  then
34:      if one of  $g_n, \dots, g_2$  is constant then
35:        throw HYPOTHESIS 2 NOT MET
36:      if  $C$  is not a regular chain then
37:        throw HYPOTHESIS 3 NOT MET
38:      if one of  $h_2, \dots, h_n$  is not regular w.r.t.  $C$  then
39:        throw HYPOTHESIS 4 NOT MET
40:      return  $(C)$ 
41:     $M := M + e$ ;  $C_M := C_{M+e}$ ;  $C_{M+e} := \{\}$ 
42: throw HYPOTHESIS 1 NOT MET

```

---

- **FindCandidateSpecialization**( $f, T, \mathcal{P}$ ): (1) randomly chooses  $a \in \mathbf{k}$  such that  $a \notin \mathcal{P}$ ,  $a$  does not cancel  $B_T$  and  $\text{init}(f)$ , and (2) returns  $f$  and  $T$  specialized at  $X_1 = a$ . Finally,  $\mathcal{P}$  is replaced with  $\mathcal{P} \cup \{a\}$ .

To avoid the use of a couple more sub-procedures (which would have many arguments and complicated specifications), the pseudo-code of Algorithm 2 uses **Goto** statements in three places:

- At Line 19, the **Goto** statement forces the algorithm to resume from Line 5, thus discarding all images that have been computed up to that point.
- At Lines 17 and 21, the **Goto** statement forces the algorithm to resume from Line 8, thus discarding the image that is currently being computed.

A few more observations about the pseudo-code of Algorithm 2:

- Between Lines 14 and 24, the while-loop is used to compute and collect the images of  $g_n, \dots, g_2, r_1$  for  $X_1 = a$ .
- Between Lines 7 and 41, the main loop is located. Each iteration of that loop starts with the selection of a new specialization point. If the images of  $g_n, \dots, g_2, r_1$  at that specification are successfully collected, then the algorithm checks whether the desired result has been reached. When this is not the case, more images may be computed. Note that this while-loop runs until  $c \geq M + s + D$  holds, or until an exception is raised, or until the result is returned. The quantity  $M$  is replaced by  $M + s$  during the loop. However, as we shall see in Theorem 2 the algorithm always terminates.

Finally, note that pseudo-code uses two counters  $c$  and  $b$ . They, respectively, count the total number of specializations used and the number of bad specializations hit during the execution of the algorithm. The counter  $b$  is not used by the algorithm, but it is an interesting information that the algorithm can return.

**Theorem 2.** *Algorithm 2 always terminates. This is a probabilistic algorithm for computing the regular chain  $C$  as defined in Theorem 1, if Hypotheses 1, 2, 3, and 4 all hold, or detecting which Hypothesis does not hold, otherwise. If the input arguments  $B$  and  $D$  are upper bounds for  $\deg(r_1)$  and the number of bad specializations, respectively, then the algorithm is deterministic.*

*Proof.* We first prove termination. Suppose that Hypothesis 1 does not hold. Then, the while-loop between Lines 14 and 24 will never succeed in reaching  $i = 0$ . Indeed, each time this while-loop is entered the **Goto** statement at Line 17 will force the algorithm to exit this while-loop and resume at Line 8. As a result, the counter  $c$  will reach the bound  $M + s + D$  of the outer while-loop (between Lines 7 and 41) and the algorithm will terminate by throwing the exception **HYPOTHESIS 1 NOT MET**.

Suppose now that Hypothesis 1 holds. Then, the while-loop between Lines 14 and 24 will exit before reaching  $i = 0$  if and only if bad specializations are discovered:

1. at Line 19, because all previous specializations were bad,
2. or at Line 21, because the current specialization is bad.

Thus, when the while-loop between Lines 14 and 24 reaches  $i = 0$ , a new image of  $(g_n, \dots, g_2, r_1)$  is added to  $\mathcal{G}$ . Once the total number of images of  $(g_n, \dots, g_2, r_1)$  is greater than or equal to  $M + e$  and  $D$ , the algorithm:

1. tests at Line 31 whether the recombination of those images has stabilized, and, if so,
2. attempts rational function reconstruction at Line 32, and, if successful,
3. checks whether Hypotheses 2, 3 and 4 all hold

When the condition  $c > D$  holds, the current recombination of the images of  $g_n, \dots, g_2, r_1$  are believed to have the correct degrees. And, in fact, they do have the correct degrees whenever  $D$  is an upper bound for the number of bad specializations. Now, if  $c \leq D$  holds or if rational function reconstruction fails, the value of  $M$  is replaced by  $M + e$ , and thus the while-loop bound  $M + e + D$  is increased. Nevertheless, after combining sufficiently images of  $g_n, \dots, g_2, r_1$  (not using bad specializations) both conditions  $C_M = C_{M+e}$  and  $c > D$  will be true together, and, moreover, rational function reconstruction will succeed. Consequently, the section of code between Lines 34 and 40 will be entered and, therefore, the algorithm will terminate. Clearly, if the input arguments  $B$  and  $D$  are upper bounds for  $\deg(r_1)$  and the number of bad specifications, respectively, then the Algorithm 2 satisfies its specifications in a deterministic way.

## 5 Relaxing the Hypotheses

The previously described modular method works well to avoid expression swell and makes certain problems tractable, see Sect. 7. However, when one of the Hypotheses 1, 2, 3 or 4 does not hold, the algorithm will fail to produce a result. We take this section to sketch how a *wrapper algorithm* handles the cases where Algorithm 2 throws an exception.

**When Hypothesis 1 Fails.** If  $r_i \in \mathbf{k}$  or  $\text{mvar}(r_i) \neq X_i$ , for some  $i$ , three cases must be considered. First, if  $r_i = 0$  then the polynomials  $r_{i+1}$  and  $t_{i+1}$  have a GCD with a positive degree in  $X_i$ . Let us call this GCD  $d$ . The computations thus split into two cases:  $d \neq 0$  and  $d = 0$ . This leads, in principle, to two recursive calls to the `Intersect` algorithm; see [15]: one to compute the intersection of  $f$  and  $\{t_2, \dots, t_i, t_{i+1}/d, t_{i+2}, \dots, t_n\}$  and one to compute the intersection of  $f$  and  $\{t_2, \dots, t_i, d, t_{i+2}, \dots, t_n\}$ . We note that the first one may be attempted by our modular algorithm. Meanwhile in the second one, we have  $r_{i+1}$  null modulo  $\text{sat}(T)$ , thus the computations performed in the original call can be recycled in order to complete `Intersect(f, T)`. Second, if  $r_i \in \mathbf{k} \setminus \{0\}$  then  $V(f) \cap W(T) = \emptyset$  and the empty set should be returned. Third, If  $\text{mvar}(r_i) \neq X_i$ , say  $\text{mvar}(r_i) = X_j$  for  $j < i$ . Then, one simply needs to “skip” computing the subresultant chain of  $r_i$  and  $t_i$  and instead compute the subresultant chain between  $r_i$  and  $t_j$  with respect to  $X_j$ . Then, the corresponding  $g_{i-1}, \dots, g_{j+1}$  are set to be  $t_{i-1}, \dots, t_{j+1}$ , respectively.

**When Hypothesis 2 Fails.** If one of the  $g_2, \dots, g_n$  is constant, say  $g_i$ , then a regular GCD for  $r_i$  and  $t_i$  can be found using a subresultant of index higher than

1 from  $S(r_i, t_i, X_i)$ . Since Algorithm 2 has computed  $S(r_i, t_i, X_i)$  (by computing modular images of it), one can recycle the computations performed by that algorithm in order to obtain a regular GCD for  $r_i$  and  $t_i$ .

**When Hypothesis 3 Fails.** When this happens, the set  $C := \{\bar{s}, g_2, \dots, g_n\}$  is not a regular chain. As in the previous case, one of the polynomials  $g_2, \dots, g_n$ , say  $g_i$ , fails to be a regular GCD of  $r_i$  and  $t_i$  modulo  $\langle \bar{s}, g_2, \dots, g_{i-1} \rangle$ . Here again, one can recycle the modular images  $S(r_i, t_i, X_i)$  to obtain a correct regular GCD.

Recovering from the failure of Hypotheses 2 or 3 can be accomplished by means of a task-pool scheme where each task consists of an integer  $i$  and a proposed regular chain  $C'$ . The general idea is to process the regular chain “bottom-up”, replacing any offending  $g_i$  with a new regular GCD, and splitting computations as necessary. For  $g_i$  to be a regular GCD of  $r_i$  and  $t_i$  modulo  $\langle \bar{s}, g_2, \dots, g_{i-1} \rangle$ ,  $\text{init}(g_i)$  must be regular modulo  $\langle \bar{s}, g_2, \dots, g_{i-1} \rangle$ ; this can easily be checked with a call to the function `Regularize`.

As soon as we hit a  $g_i$  such that its initial is not regular modulo  $C_i := \langle \bar{s}, g_2, \dots, g_{i-1} \rangle$ , the regular chain  $C_i$  is split in two (or more) regular chains  $C_{i,1}$  and  $C_{i,2}$ . For one of these regular chains, say  $C_{i,1}$ , we have that the initial of  $g_i$  is regular modulo  $C_{i,1}$ . This implies that in this particular branch of the computations,  $g_i$  is a regular GCD of  $r_i$  and  $t_i$ . For the second branch,  $g_i$  is zero modulo  $C_{i,2}$  and thus  $g_i$  is not a regular GCD of  $r_i$  and  $t_i$ . Hence, we need to replace  $g_i$  with the next non-zero polynomial in the subresultant chain between  $r_i$  and  $t_i$ , say  $g'_i$ . We replace the previous task with two new ones: one in which we want to check the regularity of the initial of  $g_{i+1}, \dots, g_n$  modulo  $C_{i,1}$ , and another one in which we want to check the regularity of  $g'_i, g_{i+1}, \dots, g_n$  modulo  $C_{i,2}$ . A task is considered complete once  $g_n$  is found to be regular. We repeat this process until the task pool is empty.

**When Hypothesis 4 Fails.** Lastly, consider Hypothesis 4. This hypothesis says that the resulting regular chain  $C := \{\bar{s}, g_2, \dots, g_n\}$  must maintain the *inequalities* defined by the initials of the polynomials  $t_i$  in the regular chain  $T$ , that is, none of those initials must vanish on  $V(f) \cap W(T)$ . Hypothesis 4 fails if and only if (at least) one of the  $\text{init}(t_i)$ 's is not invertible modulo the ideal  $\langle \bar{s}, g_2, \dots, g_n \rangle$ . Rectifying this issue is handled easily by a call to `Regularize`. Let  $t_i$  be a polynomial whose initial is not regular modulo  $\langle C \rangle$ . Since  $C$ , after passing Hypothesis 3, is a regular chain, one can call `Regularize`( $\text{init}(t_i), C$ ) to compute regular chains  $C_1, \dots, C_e$  such that  $\text{init}(t_i)$  is either regular or zero modulo  $C_j$ . Then, one simply discards any  $C_j$  for which  $\text{init}(t_i)$  is zero. A similar “discarding process” is applied by the `CleanChain` procedure in the non-modular case [15].

## 6 Implementation

In the preceding sections we have discussed a modular algorithm based on evaluation-interpolation. In fact, we employ two separate modular methods. In practice, triangular decompositions are often performed over the rational numbers. Thus,  $\mathbf{k}$  should be  $\mathbb{Q}$  in all of the previous algorithms.



Algorithm 2 is actually implemented and executed over a finite field. Our implementation is written in the C programming language as part of the BPAS Library [2] and follows [3] for its implementation of sparse multivariate polynomials over the rationals and finite fields. Moreover, our implementation actually implements a *wrapper function*, as detailed in Sect. 5. This function is able to catch the exceptions of Hypotheses 2, 3, or 4, recover from them, and produce a correct output. The implementation does not yet handle when Hypothesis 1 fails, instead falling back to the non-modular implementation of `Intersect` in BPAS [4, 12]. This is left to future work.

The implementation of Algorithm 2 is broken into three main phases: computing subresultants, interpolation and reconstruction of the regular chain  $C$  (see Sect. 3 for notations), and *lifting the coefficients* from a finite field to  $\mathbb{Q}$ . Interpolation and rational function reconstruction are standard algorithms. Thus, we describe the other two main parts.

Subresultants are computed in three different ways depending of the degrees of the input polynomials. All three methods are detailed in [5]. First, an optimized version of Ducos' subresultant chain algorithm handles the general case. Second, when degrees are high, one can compute each subresultant itself using evaluation-interpolation. We can evaluate the variables  $X_2, \dots, X_{i-1}$ , compute strictly univariate subresultants, and then recover the true subresultants through interpolation. We implement this multivariate evaluation-interpolation using a multi-dimensional truncated Fourier transform (TFT). Third, when computations are univariate (either when computing  $\bar{s}$  or as univariate images in the evaluation-interpolation scheme), one can use an algorithm based on Half-GCD to compute only the subresultants of index 0 and 1 rather than the entire subresultant chain.

Recovering the rational number coefficients is an implementation of the technique based on Hensel-lifting described in [21]. With an implementation of this Hensel lifting for triangular sets, notice that a modular algorithm for a zero-dimensional intersect is immediate. Algorithm 2 can be transformed to compute the intersection between  $f$  and a zero-dimensional regular chain  $T$  as follows. Working modulo a prime  $p$ , do not specialize any variables and directly normalize  $T$ . Compute the iterated subresultants of  $f$  and  $T$ , do not interpolate any variables, and directly construct  $C$ . Then, perform Hensel lifting to reconstruct the coefficients of  $C$  over  $\mathbb{Q}$ . This method is very effective in practice to reduce expression swell in the coefficients, as we describe next.

## 7 Experimentation and Discussion

Our experimentation was collected on a desktop running 20.04.1-Ubuntu with an Intel Core i7-7700K processor at 4.20GHz, and 16GB DDR4 memory at 2.4 GHz. We first show that the modular method is effective in practice to significantly reduce the computational time of computing a triangle decomposition, and even solves some polynomial systems which were infeasible for previous solvers.

Table 1 summarizes these results by describing the structure of these well-known systems, as well as the execution time to solve the system using the

**Table 1.** Running times of MAPLE vs. BPAS(non-modular) vs. BPAS(modular)

System	Number of Variables	Number of Equations	Bézout Bound	Number of Solutions	MAPLE Time (s)	BPAS(non-modular) Time (s)	BPAS(modular) Time (s)
noon5	5	5	243	233	1.46	0.61	0.42
eco8	8	8	1458	64	N/A	N/A	60.63
Cassou-Nogues	4	4	1344	16	1.43	5.60	0.42
childDraw-2	10	10	256	42	12.70	2.83	2.48
Issac97	4	4	16	16	156.33	101.16	1.92
Themos-net-2	6	6	32	24	55.10	57.60	1.37
Uteshev-Bikker	4	4	36	36	N/A	N/A	362.97
Theomes-net-3	5	5	32	24	54.93	57.01	1.36
Noonburg-5	5	5	243	233	2011.24	314.72	6.43
cohn2	4	4	900	Positive Dimension	145.39	1322.98	1315.34
rabno	9	9	36000	16	3.77	2.97	2.97
tangents0	6	6	64	24	3.69	2.40	0.39
Cassou-Nogues-2	4	4	450	8	N/A	N/A	2145.28

**Table 2.** Runtime analysis of the subroutines of the modular method

System	Call Number	Number of Evaluations	Bézout Bound	Time (s) for Collect Images	Time (s) for Subresultants	Time (s) for Interpolation	Time (s) for Modular Intersect	Time (s) for Hensel lifting
noon5	1st	161	243	0.01	0.01	0.02	0.00	0.02
noon5	2nd	161	243	0.00	0.01	0.02	0.00	0.00
noon5	3rd	161	243	0.00	0.01	0.02	0.00	0.01
eco8	1st	289	1458	0.07	0.06	0.31	0.12	2.56
Cassou-Nogues	1st	161	1344	0.02	0.04	0.02	0.01	0.08
Issac97	1st	161	16	0.01	0.01	0.02	0.00	0.33
Themos-net-2	1st	161	32	0.02	0.02	0.03	0.01	0.74
Uteshev-Bikker	1st	193	36	0.02	0.03	0.04	0.01	58.91
Themos-net-3	1st	161	32	0.02	0.02	0.03	0.01	0.75
Noonburg-5	1st	257	243	0.02	0.64	1.22	0.01	2.93
tangents0	1st	161	64	0.02	0.02	0.02	0.00	0.13
Cassou-Nogues-2	1st	161	450	0.01	0.03	0.02	0.00	0.04

modular method and not using the modular method, if the latter is possible. The non-modular implementation is the (serial) version described in [4, 12]. As a point of comparison, we also present the time to compute a triangular decomposition using the `RegularChains` library of *Maple 2022*. In particular, the systems *eco8*, *Uteshev-Bikker*, and *Cassou-Nogues-2*, could not be solved within two hours of computation time. However, the modular method allows the first two to be solved on the order of minutes, and *Cassou-Nogues-2* on the order of 10s of minutes.

In Table 2, we describe a detailed analysis of the modular intersect in dimension one. Observe that the running time for each main task is provided. Additionally, it is important to mention that in some cases, the number of collected images is far below the Bézout bound of the input systems. Therefore, this shows the importance of stabilization techniques in the implementation.

Among our works-in-progress is, of course, is the adaptation and implementation of this modular method for `Intersect(f, T)` for *T* in dimension higher than 1. This is necessary in order to tackle even harder polynomial systems. Moreover, recovering from cases where Hypothesis 1 fails must also be implemented.

## References

1. Arnold, E.A.: Modular algorithms for computing Gröbner bases. *J. Symb. Comput.* **35**(4), 403–419 (2003)
2. Asadi, M., et al.: Basic Polynomial Algebra Subprograms (BPAS) (2023). <https://www.bpaslib.org>
3. Asadi, M., Brandt, A., Moir, R.H.C., Moreno Maza, M.: Algorithms and data structures for sparse polynomial arithmetic. *Mathematics* **7**(5), 441 (2019)
4. Asadi, M., Brandt, A., Moir, R.H.C., Moreno Maza, M., Xie, Y.: Parallelization of triangular decompositions: techniques and implementation. *J. Symb. Comput.* **115**, 371–406 (2023)
5. Asadi, M., Brandt, A., Moreno Maza, M.: Computational schemes for subresultant chains. In: Boulier, F., England, M., Sadykov, T.M., Vorozhtsov, E.V. (eds.) *CASC 2021. LNCS*, vol. 12865, pp. 21–41. Springer, Cham (2021). [https://doi.org/10.1007/978-3-030-85165-1\\_3](https://doi.org/10.1007/978-3-030-85165-1_3)
6. Aubry, P., Lazard, D., Moreno Maza, M.: On the theories of triangular sets. *J. Symb. Comput.* **28**(1–2), 105–124 (1999)
7. Becker, E., Mora, T., Marinari, M.G., Traverso, C.: The shape of the shape lemma. In: MacCallum, M.A.H. (ed.) *Proceedings of ISSAC 1994*, pp. 129–133. ACM (1994)
8. Boulier, F., Lazard, D., Ollivier, F., Petitot, M.: Representation for the radical of a finitely generated differential ideal. In: *Proceedings of the International Symposium on Symbolic and Algebraic Computation*, pp. 158–166. ACM (1995)
9. Boulier, F., Lemaire, F., Moreno Maza, M.: Well known theorems on triangular systems and the D5 principle. In: *Proceedings of the Transgressive Computing* (2006)
10. Boulier, F., Lazard, D., Ollivier, F., Petitot, M.: Computing representations for radicals of finitely generated differential ideals. *Appl. Algebra Eng. Commun. Comput.* **20**(1), 73–121 (2009)
11. Boulier, F., Lemaire, F., Moreno Maza, M.: Computing differential characteristic sets by change of ordering. *J. Symb. Comput.* **45**(1), 124–149 (2010)
12. Brandt, A.: The design and implementation of a high-performance polynomial system solver. Ph.D. thesis, University of Western Ontario (2022)
13. Chen, C., et al.: Computing the real solutions of polynomial systems with the regularchains library in maple. *ACM Commun. Comput. Algebra* **45**(3/4) (2011)
14. Chen, C., Davenport, J.H., May, J.P., Moreno Maza, M., Xia, B., Xiao, R.: Triangular decomposition of semi-algebraic systems. *J. Symb. Comput.* **49**, 3–26 (2013)
15. Chen, C., Moreno Maza, M.: Algorithms for computing triangular decomposition of polynomial systems. *J. Symb. Comput.* **47**(6), 610–642 (2012)
16. Chen, C., Moreno Maza, M.: An incremental algorithm for computing cylindrical algebraic decompositions. In: Feng, R., Lee, W., Sato, Y. (eds.) *Computer Mathematics*, pp. 199–221. Springer, Heidelberg (2014). [https://doi.org/10.1007/978-3-662-43799-5\\_17](https://doi.org/10.1007/978-3-662-43799-5_17)
17. Chen, C., Moreno Maza, M.: Quantifier elimination by cylindrical algebraic decomposition based on regular chains. *J. Symb. Comput.* **75**, 74–93 (2016)
18. Chou, S., Gao, X.: Computations with parametric equations. In: *Proceedings of the ISSAC 1991*, pp. 122–127 (1991)
19. Chou, S., Gao, X.: A zero structure theorem for differential parametric systems. *J. Symb. Comput.* **16**(6), 585–596 (1993)

20. Cox, D.A., Little, J., O’Shea, D.: *Ideals, Varieties, and Algorithms: An Introduction to Computational Algebraic Geometry and Commutative Algebra*. UTM, Springer, Cham (2015). <https://doi.org/10.1007/978-3-319-16721-3>
21. Dahan, X., Moreno Maza, M., Schost, É., Wu, W., Xie, Y.: Lifting techniques for triangular decompositions. In: *Proceedings of the International Symposium on Symbolic and Algebraic Computation*, pp. 108–115 (2005)
22. Dong, R., Lu, D., Mou, C., Wang, D.: Comprehensive characteristic decomposition of parametric polynomial systems. In: *Proceedings of the International Symposium on Symbolic and Algebraic Computation*, pp. 123–130. ACM (2021)
23. Faugère, J.C.: *Résolution des systèmes d’équations algébriques*. Ph.D. thesis, Université Paris 6 (1994)
24. Gao, X., van der Hoeven, J., Yuan, C., Zhang, G.: Characteristic set method for differential-difference polynomial systems. *J. Symb. Comput.* **44**(9) (2009)
25. von zur Gathen, J., Gerhard, J.: *Modern Computer Algebra*, 2nd edn. Cambridge University Press, Cambridge (2003)
26. Hu, Y., Gao, X.S.: Ritt-Wu characteristic set method for Laurent partial differential polynomial systems. *J. Syst. Sci. Complex.* **32**(1), 62–77 (2019)
27. Kapur, D., Saxena, T.: Extraneous factors in the Dixon resultant formulation. In: *Proceedings of the ISSAC 1997*, pp. 141–148. ACM (1997)
28. Lazard, D.: A new method for solving algebraic systems of positive dimension. *Discret. Appl. Math.* **33**(1–3), 147–160 (1991)
29. Lazard, D.: Solving zero-dimensional algebraic systems. *J. Symb. Comput.* **13**(2), 117–132 (1992)
30. Monagan, M.B.: Probabilistic algorithms for computing resultants. In: *Proceedings of the ISSAC*, pp. 245–252. ACM (2005)
31. Moreno Maza, M., Sandford, R.: Towards extending Fulton’s algorithm for computing intersection multiplicities beyond the bivariate case. In: Boulier, F., England, M., Sadykov, T.M., Vorozhtsov, E.V. (eds.) *CASC 2021*. LNCS, vol. 12865, pp. 232–251. Springer, Cham (2021). [https://doi.org/10.1007/978-3-030-85165-1\\_14](https://doi.org/10.1007/978-3-030-85165-1_14)
32. Moreno Maza, M., Xia, B., Xiao, R.: On solving parametric polynomial systems. *Math. Comput. Sci.* **6**(4), 457–473 (2012)
33. Moreno Maza, M.: On triangular decompositions of algebraic varieties. Technical report. TR 4/99, NAG Ltd, Oxford, UK (1999). Presented at the MEGA-2000 Conference
34. Maza, M.M., Rioboo, R.: Polynomial GCD computations over towers of algebraic extensions. In: Cohen, G., Giusti, M., Mora, T. (eds.) *AAECC 1995*. LNCS, vol. 948, pp. 365–382. Springer, Heidelberg (1995). [https://doi.org/10.1007/3-540-60114-7\\_28](https://doi.org/10.1007/3-540-60114-7_28)
35. Ritt, J.F.: *Differential Algebra*. Dover Publications Inc., New York (1966)
36. Schmid, J.: On the affine Bezout inequality. *Manuscr. Math.* **88**(1), 225–232 (1995)
37. Schost, É.: Degree bounds and lifting techniques for triangular sets. In: *Proceedings of the ISSAC 2002*, pp. 238–245. ACM (2002)
38. Shimoyama, T., Yokoyama, K.: Localization and primary decomposition of polynomial ideals. *J. Symb. Comput.* **22**(3), 247–277 (1996)
39. Traverso, C.: Gröbner trace algorithms. In: Gianni, P. (ed.) *ISSAC 1988*. LNCS, vol. 358, pp. 125–138. Springer, Heidelberg (1989). [https://doi.org/10.1007/3-540-51084-2\\_12](https://doi.org/10.1007/3-540-51084-2_12)
40. Wang, D.K.: The Wsolve package. [www.mmrc.iss.ac.cn/~dwang/wsolve.html](http://www.mmrc.iss.ac.cn/~dwang/wsolve.html)
41. Wang, D.M.: Epsilon 0.618. <http://wang.cc4cm.org/epsilon/index.html>
42. Wu, W.T.: A zero structure theorem for polynomial equations solving. *MM Res. Preprints* **1**, 2–12 (1987)

43. Xia, B.: DISCOVERER: a tool for solving semi-algebraic systems. *ACM Commun. Comput. Algebra* **41**(3), 102–103 (2007)
44. Yang, L., Hou, X., Xia, B.: A complete algorithm for automated discovering of a class of inequality-type theorems. *Sci. China Ser. F Inf. Sci.* **44**(1), 33–49 (2001)
45. Yang, L., Zhang, J.: Searching dependency between algebraic equations: an algorithm applied to automated reasoning. In: *Artificial Intelligence in Mathematics*, pp. 147–156. Oxford University Press (1994)