On the complexity of computing with zero-dimensional triangular sets

Adrien Poteaux[⋆] Éric Schost[†]
adrien.poteaux@lifl.fr eschost@uwo.ca
[†]: Computer Science Department, The University of Western Ontario, London, ON, Canada
[⋆]: LIFL, UMR Lille1-CNRS 8022 France

July 16, 2012

Abstract

We study the complexity of some fundamental operations for triangular sets in dimension zero. Using Las-Vegas algorithms, we prove that one can perform such operations as change of order, equiprojectable decomposition, or quasi-inverse computation with a cost that is essentially that of *modular composition*. Over an abstract field, this leads to a subquadratic cost (with respect to the degree of the underlying algebraic set). Over a finite field, in a boolean RAM model, we obtain a quasi-linear running time using Kedlaya and Umans' algorithm for modular composition.

Conversely, we also show how to reduce the problem of modular composition to change of order for triangular sets, so that all these problems are essentially equivalent.

Our algorithms are implemented in Maple; we present some experimental results.

1 Introduction

Triangular sets (in dimension zero, in this paper) are families of polynomials with a simple triangular structure, which turns out to be well adapted to solve many problems for systems of polynomial equations. As a result, there is now a vast literature dedicated to algorithms with triangular sets, their generalization to *regular chains*, and applications: without being exhaustive, we refer the reader to [24, 3, 33, 23, 38, 39].

However, from the algorithmic point of view, many questions remain. Despite a growing amount of work [31, 28, 8], the complexity of many basic operations with triangular sets (such as set-theoretic operations on their zero-sets, change of variable order, or arithmetic operations modulo a triangular set) remains imperfectly understood.

The aim of this paper is to answer some of these questions, by describing fast algorithms for several operations with triangular sets, extending our previous results from [35]. In particular, we will focus on the relationship between these problems and some classical

operations on univariate and bivariate polynomials, called *modular composition* and *power projection*. To describe these issues with more details, we need a few definitions.

1.1 Basic definitions

Triangular sets. Let $\mathbb{K}$ be our base field, and let $\mathbf{X} = X_1, \ldots, X_n$ be indeterminates over $\mathbb{K}$; we order them as $X_1 < \cdots < X_n$. A (monic) triangular set $\mathbf{T} = (T_1, \ldots, T_n)$, for this variable order, is a family of polynomials in $\mathbb{K}[\mathbf{X}]$ with the following triangular structure

$$\mathbf{T} \left| \begin{array}{l} T_n(X_1, \ldots, X_n) \\ \quad \vdots \\ T_1(X_1), \end{array} \right.$$

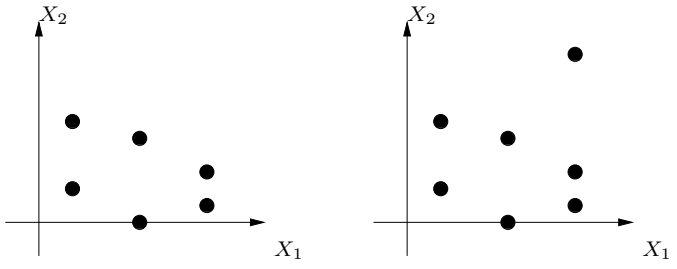
and such that for all i, T_i is monic in X_i and reduced modulo $\langle T_1, \ldots, T_{i-1} \rangle$, in the sense that $\deg(T_i, X_j) < \deg(T_j, X_j)$ for $j < i$; in particular, $\mathbf{T}$ is a zero-dimensional Gröbner basis for the lexicographic order induced by $X_1 < \cdots < X_n$. In all that follows, we will impose the condition that $\mathbb{K}$ is a perfect field; often, we will also require that $\langle \mathbf{T} \rangle$ is a radical ideal.

We write $d_i = \deg(T_i, X_i)$; $\mathbf{d} = (d_1, \ldots, d_n)$ will be called the *multidegree* of $\mathbf{T}$. Define further $R_{\mathbf{T}} = \mathbb{K}[\mathbf{X}]/\langle \mathbf{T} \rangle$. Then, $\delta_{\mathbf{T}} = d_1 \cdots d_n$ is the natural complexity measure associated to computations modulo $\langle \mathbf{T} \rangle$, as it represents the dimension of the residue class ring $R_{\mathbf{T}}$ over $\mathbb{K}$. This integer will be called the *degree* of $\mathbf{T}$.

In all our algorithms, elements of $R_{\mathbf{T}}$ are represented on the monomial basis $B_{\mathbf{T}} = \{X_1^{a_1} \cdots X_n^{a_n} \mid 0 \leq a_i < d_i \text{ for all } i\}$. Dually, all $\mathbb{K}$-linear forms $R_{\mathbf{T}} \to \mathbb{K}$ are represented by their values on the basis $B_{\mathbf{T}}$.

Equiprojectable sets. Not every zero-dimensional radical ideal I in $\mathbb{K}[\mathbf{X}]$ admits a triangular set of generators: this is the case only when the zero-set $V = V(I) \subset \overline{\mathbb{K}}^n$ possesses a geometric property called *equiprojectability* [4]. For the moment, we will simply give an idea of the definition; proper definitions are in Section 4.

Roughly speaking, V is equiprojectable if all fibers of the projection $V \to \overline{\mathbb{K}}^{n-1}$ have the same cardinality, and similarly for the further projections to $\overline{\mathbb{K}}^{n-2}, \ldots, \overline{\mathbb{K}}$. For instance, of the following pictures, the left-hand one describes an equiprojectable set, whereas the right-hand one does not (since the rightmost fiber has a larger cardinality than the others).



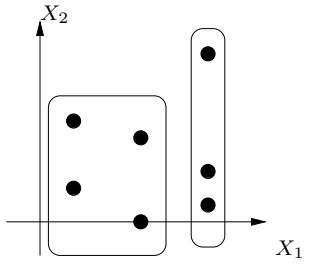
The relationship with triangular representations is described in [4]: V is equiprojectable if and only if its defining ideal I is generated by a triangular set (for this equivalence, it is required that the base field be perfect).

Equiprojectable decomposition. Any finite set can be decomposed, in general not uniquely, into a finite union of pairwise disjoint equiprojectable sets. At the level of ideals, this amounts to write a zero-dimensional radical ideal I as $I = \langle \mathbf{T}^{(1)} \rangle \cap \cdots \cap \langle \mathbf{T}^{(s)} \rangle$, with all $\mathbf{T}^{(j)}$ being triangular sets and all ideals $\langle \mathbf{T}^{(j)} \rangle$ being pairwise coprime. Of course, starting from I in $\mathbb{K}[\mathbf{X}]$, we want all $\mathbf{T}^{(j)}$ to have coefficients in $\mathbb{K}$ as well.

To solve the non-uniqueness issue, the decomposition of I into an intersection of *maximal* ideals may appear as a good candidate; however, it suffers from significant drawbacks. For instance, computing it requires us to factor polynomials over $\mathbb{K}$, or extensions of it: even if we strengthen our model by requiring that $\mathbb{K}$ and its finite extensions support this operation, it is usually prohibitively costly.

There exists another canonical way to find such a decomposition, called the *equiprojectable decomposition* [15]. For instance, among its useful properties is the fact that it behaves well under specialization: if $\mathbb{K}$ is the fraction field of a ring $\mathbb{A}$ such as $\mathbb{A} = k[Z_1, \ldots, Z_r]$ or $\mathbb{A} = \mathbb{Z}$ and $\mathfrak{m}$ is a maximal ideal of $\mathbb{A}$, the equiprojectable decomposition of $(I \bmod \mathfrak{m})$ coincides with the equiprojectable decomposition of I, reduced modulo $\mathfrak{m}$, for "most" maximal ideals $\mathfrak{m}$. We refer to [15] for more precise statements; here, we simply point out that this property makes it for instance possible to apply modular methods, such as Hensel lifting techniques [38, 39], to recover the equiprojectable decomposition of I starting from that of $(I \bmod \mathfrak{m})$; the decomposition of I into maximal ideals does not have this useful specialization property.

While the definition of the equiprojectable decomposition is technical, the idea is simple. We will proceed geometrically: to obtain the equiprojectable decomposition of a finite set $V \subset \overline{\mathbb{K}}^n$, we first split it using the cardinality of the fibers of the projection $\overline{\mathbb{K}}^n \to \overline{\mathbb{K}}^{n-1}$. Then we apply the same process to all the components we obtained, using the projection to $\overline{\mathbb{K}}^{n-2}$, and so on (again, we refer the reader to Section 4 for precise definitions). The following picture (from [15]) shows the equiprojectable decomposition of the non-equiprojectable set V of the former example.



Each component of the equiprojectable decomposition is an equiprojectable set. As a result, this construction allows us to represent an arbitrary finite set V, defined over $\mathbb{K}$, by means of a canonical family of triangular sets with coefficients in $\mathbb{K}$, that depends only on

the order $<$ we have chosen on the variables. The collection of these triangular sets will thus be denoted by $\mathscr{D}(V, <)$.

1.2 Our contribution

Our purpose is to give algorithms for various operations involving a triangular set, or a family thereof. We will make these questions more precise below; for the moment, one should have in mind problems such as modular arithmetic, computation of the equiprojectable decomposition, or change of order on the variables.

Two central problems. The following two problems, called modular composition and power projection, will be at the heart of our algorithms. Given a triangular set $\mathbf{T}$ in $\mathbb{K}[X_1, \ldots, X_n]$, the general forms of these questions are the following.

- *modular composition:* given F in $\mathbb{K}[Y_1, \ldots, Y_m]$, with $\deg(F, Y_i) < f_i$ for all i, and $(G_1, \ldots, G_m)$ in $R_{\mathbf{T}}^m$, compute $F(G_1, \ldots, G_m) \in R_{\mathbf{T}}$
- *power projection:* given a linear form $\ell : R_{\mathbf{T}} \to \mathbb{K}$, $(G_1, \ldots, G_m)$ in $R_{\mathbf{T}}^m$ and bounds $f_1, \ldots, f_m$, compute $\ell(G_1^{c_1} \cdots G_m^{c_m})$, for all $c_1 < f_1, \ldots, c_m < f_m$.

In both cases, we will write $\mathbf{f} = (f_1, \ldots, f_m)$ and $\delta_{\mathbf{f}} = f_1 \cdots f_m$, so that the size of the problem is characterized by $\delta_{\mathbf{f}}$ and $\delta_{\mathbf{T}}$. We will call (m, n) the *parameters* for these questions, and $\max(\delta_{\mathbf{f}}, \delta_{\mathbf{T}})$ the *size*. When $\mathbf{T}$ and $G_1, \ldots, G_m$ are fixed, the two problems become linear in respectively F and ℓ; as it turns out, they are dual problems, as was observed by Shoup for $m = n = 1$ [40].

The only cases we will need actually have parameters (m, n) in $\{1, 2\}$. Besides, we will always suppose that $\delta_{\mathbf{f}} \leq \delta_{\mathbf{T}}$, so that all costs can be measured in terms of $\delta_{\mathbf{T}}$ only. However, even in this simple situation, these questions have resisted many attempts.

As of now, no quasi-linear time algorithm is known in an algebraic complexity model (say using an algebraic RAM, counting field operations at unit cost). Among the best results known to us is that both operations can be done in time $O(\delta_{\mathbf{T}}^{(\omega+1)/2})$, where ω is such that matrices over $\mathbb{K}$ of size n can be multiplied in time $O(n^\omega)$; we assume $\omega > 2$, otherwise logarithmic terms may appear. Using the exponent $\omega \leq 2.38$ from [13], this gives the subquadratic estimate $O(\delta_{\mathbf{T}}^{1.69})$.

For $(m, n) = (1, 1)$, this claim follows from respectively Brent and Kung's modular composition algorithm [10] and Shoup's power projection algorithm [40], which is actually the transpose of Brent and Kung's. For power projection, extensions to parameters $(m, n) = (1, 2)$ are in [41, 25, 5], and the case $(m, n) = (2, 2)$ is partially dealt with in [34]. For completeness, in Section 2.1, we will give straightforward extensions of the Brent-Kung and Shoup algorithms to all cases $(m, n) \in \{1, 2\}$, establishing the bound $O(\delta_{\mathbf{T}}^{(\omega+1)/2})$ claimed above.

We will thus write $\mathsf{C} : \mathbb{N} \to \mathbb{N}$ to denote a function such that over any field, one can do both modular composition and power projection in $\mathsf{C}(\delta_{\mathbf{T}})$ base field operations, under the assumptions that the parameters (m, n) are in $\{1, 2\}$ and $\delta_{\mathbf{f}} \leq \delta_{\mathbf{T}}$. We take C super-linear,

in the sense that we require that $\mathsf{C}(d_1 + d_2) \geq \mathsf{C}(d_1) + \mathsf{C}(d_2)$ holds for all d_1, d_2. Then, the former discussion shows that we can take $\mathsf{C}(d) \in O(d^{(\omega+1)/2}) \subset O(d^{1.69})$.

Some further restrictions are imposed on the function C. As is now customary, we let $\mathsf{M} : \mathbb{N} \to \mathbb{N}$ be such that over any ring, polynomials of degree less than d can be multiplied in $\mathsf{M}(d)$ base ring operations; we make the standard superlinearity assumptions of [18, Chapter 8]. Using Cantor and Kaltofen's algorithm [12], we can take $\mathsf{M}(d)$ in $O(d \log(d) \log\log(d))$. Then, to simplify several estimates, we also make the reasonable assumption that $\mathsf{M}(d)\log(d)$ is in $O(\mathsf{C}(d))$; this is the case for $\mathsf{M}(d)$ quasi-linear and $\mathsf{C}(d) = d^{(\omega+1)/2}$.

The Kedlaya-Umans algorithm and its applications. In a boolean model (using a boolean RAM, with logarithmic cost for data access), and for $\mathbb{K} = \mathbb{F}_q$, it turns out that one can do much better than in the algebraic model for modular composition and power projection.

The best known result comes from Kedlaya and Umans' work [26]: for $n = 1$, they show how to solve both problems in $\delta_{\mathbf{T}}^{1+\varepsilon} \log(q)^{1+o(1)}$ bit operations, for all $\varepsilon > 0$. Their algorithm uses modular techniques (transferring the problem over $\mathbb{F}_q$ to a problem over $\mathbb{Z}$, and vice versa), and the idea does not seem to extend easily to an arbitrary base field. In [35], we described an extension of this result to any parameters $(m, n) \in \{1, 2\}$, with a running time of $\delta_{\mathbf{T}}^{1+\varepsilon} O^{\sim}(\log(q))$ bit operations for any $\varepsilon > 0$; the $O^{\sim}$ notation indicates the omission of polylogarithmic factors of the form $\log\log(q)^{O(1)}$.

In this paper, we will be interested in both models, algebraic and boolean. Now, for a given algorithm, the cost analysis in the boolean model differs from the analysis in the algebraic model (where we only count base field operations) by a few aspects. A minor issue is that we should count the cost of fetching data (which grows like $\log(a)$, to access the contents at address a). Another difference is that in the boolean model, we need to take into account the boolean cost of operations in $\mathbb{F}_q$: disregarding the cost of fetching data, any arithmetic operations in $\mathbb{F}_q$ can be done in $O^{\sim}(\log(q))$ bit operations, say $\log(q)\log\log(q)^k$ for some fixed $k \geq 0$.

As a result, in what follows, in all rigor, we should prove most statements twice, once in the algebraic complexity model and once in the boolean one. To avoid making the paper excessively heavy, we will indeed state our main results twice, but *all intermediate results and proofs will be given for the algebraic model*. There would actually be no major difference in the boolean model, only some extra bookkeeping, on the basis of the remarks in the previous paragraph.

Similarly to the algebraic case, $\mathsf{C}_{\mathrm{bool}}$ will thus denote a function such that one can do both modular composition and power projection over $\mathbb{F}_q$ using $\mathsf{C}_{\mathrm{bool}}(\delta_{\mathbf{T}}, q)$ bit operations, assuming that the parameters (m, n) are in $\{1, 2\}$ and that $\delta_{\mathbf{f}} \leq \delta_{\mathbf{T}}$. As before, we require that $\mathsf{C}_{\mathrm{bool}}(d_1 + d_2, q) \geq \mathsf{C}_{\mathrm{bool}}(d_1, q) + \mathsf{C}_{\mathrm{bool}}(d_2, q)$ holds for all d_1, d_2, q. As in the algebraic case, we will also assume that the cost of polynomial multiplication and related operations can be absorbed into $\mathsf{C}_{\mathrm{bool}}$: explicitly, we require that for any function $f(d) \in O^{\sim}(d)$, the function $f(d)\log(q)\log\log(q)^k$ is in $O(\mathsf{C}_{\mathrm{bool}}(d, q))$, where k is the constant introduced above. The results of [35] imply that we can take $\mathsf{C}_{\mathrm{bool}}(d, q)$ in $d^{1+\varepsilon} O^{\sim}(\log(q))$ for any $\varepsilon > 0$.

Main results. The questions we will consider are the following set-theoretic operations. In all the following items, *all triangular sets are supposed to generate zero-dimensional radical ideals.*

P$_1$. Given triangular sets $\mathbf{T}^{(1)}, \ldots, \mathbf{T}^{(\ell)}$ and $\mathbf{S}^{(1)}, \ldots, \mathbf{S}^{(r)}$ in $\mathbb{K}[X_1, \ldots, X_n]$, for a variable order $<$, and given a target variable order $<'$, compute the equiprojectable decomposition

$$\mathscr{D}\big(V(\mathbf{T}^{(1)}) \cup \cdots \cup V(\mathbf{T}^{(\ell)}) - V(\mathbf{S}^{(1)}) - \cdots - V(\mathbf{S}^{(r)}), <'\big).$$

We let δ_1 be the sum of the degrees of $\mathbf{T}^{(1)}, \ldots, \mathbf{T}^{(\ell)}$ and $\mathbf{S}^{(1)}, \ldots, \mathbf{S}^{(r)}$.

P$_2$. Given a triangular set $\mathbf{T}$ in $\mathbb{K}[X_1, \ldots, X_n]$, for a variable order $<$, as well as F in $R_{\mathbf{T}}$ and a target variable order $<'$, compute the equiprojectable decompositions

$$\mathscr{D}(V(\mathbf{T}) \cap V(F), <') \quad \text{and} \quad \mathscr{D}(V(\mathbf{T}) - V(F), <');$$

for every $\mathbf{T}'$ in $\mathscr{D}(V(\mathbf{T}) - V(F), <')$, compute also the inverse of F in $R_{\mathbf{T}'}$. (Note that even if F is only defined modulo $\langle \mathbf{T} \rangle$, the two sets above are actually defined unambiguously.) In this case, we let δ_2 be the degree of $\mathbf{T}$.

These questions are general enough to allow us to solve a variety of classical problems for triangular sets. When the initial and target orders are the same, and when $r = 0$, the first question amounts to compute the equiprojectable decomposition of a family of triangular sets, which is a key subroutine in the algorithms of [15]. When the initial and target orders are different, taking only a single triangular set $\mathbf{T}$ as input, the first question allows us to perform a change of order on $\mathbf{T}$, and to output a canonical family of triangular sets for the target order. Taking the same order for input and output, the second operation allows us to compute the *quasi-inverse* of a polynomial F modulo $\langle \mathbf{T} \rangle$, which amounts to split $V(\mathbf{T})$ into its components where F vanishes, resp. is invertible. This is an important subroutine for triangular decomposition algorithms [28].

With that being said, our first main results are the following. Our results are of the probabilistic Las Vegas kind: the outputs are always correct, but the running time is given on average (we indicate this by giving *expected* running times, as customary).

Theorem 1. *In an algebraic RAM complexity model, the following holds over any field* $\mathbb{K}$ *of characteristic* p*:*

- *if* $p = 0$ *or* p *is greater than* δ_1^2*, one can answer question* **P**$_1$ *using an expected* $O(n\mathsf{C}(\delta_1)(n + \log(\delta_1)))$ *base field operations;*
- *if* $p = 0$ *or* p *is greater than* δ_2^2*, one can answer question* **P**$_2$ *using an expected* $O(n\mathsf{C}(\delta_2)(n + \log(\delta_2)))$ *base field operations.*

In a boolean RAM complexity model, the following holds over any finite field $\mathbb{F}_q$ *of characteristic* p*:*

- *if p is greater than δ_1^2, one can answer question $\mathbf{P}_1$ using an expected $O(n\mathsf{C}_{\mathrm{bool}}(\delta_1, q)(n+\log(\delta_1)))$ bit operations;*
- *if p is greater than δ_2^2, one can answer question $\mathbf{P}_2$ using an expected $O(n\mathsf{C}_{\mathrm{bool}}(\delta_2, q)(n+\log(\delta_2)))$ bit operations.*

Using the estimates of the previous paragraphs, the former costs are $O^{\tilde{}}(n^2\delta_1^{(\omega+1)/2})$ and $O^{\tilde{}}(n^2\delta_2^{(\omega+1)/2})$, and the latter are $n^2\delta_1^{1+\varepsilon}O^{\tilde{}}(\log(q))$ and $n^2\delta_2^{1+\varepsilon}O^{\tilde{}}(\log(q))$, for any $\varepsilon > 0$. Since the input sizes are roughly proportional to δ_1 (resp. δ_2) field elements, this means that with respect to δ_1 (resp. δ_2), we obtain a subquadratic running time in the algebraic model, and a quasi-linear running time in the boolean model.

Before discussing further questions, we briefly comment on the assumption on the characteristic of $\mathbb{K}$. We do need $2, \ldots, \delta_1$ (resp. $2, \ldots, \delta_2$) to be invertible in $\mathbb{K}$; otherwise, the algorithm will not work. The stronger requirement that $2, \ldots, \delta_1^2$ (resp. $2, \ldots, \delta_2^2$) are units allows us to find random elements in $\mathbb{K}$ that are "lucky" with large probability; if this assumption does not hold, the algorithm may still succeed, but we lose the control on the expected running time (see Lemma 1).

The basic idea of our algorithms is from [35]: we reduce everything to computations with univariate polynomials, since most operations above will be easy to deal with in the univariate case. To this end, we perform a change of representation between our input and a univariate representation, by using repeatedly modular composition and power projection.

This raises the question of whether better algorithms may be possible, bypassing modular composition and power projection. The following theorem essentially proves that this is not the case, and that computing the equiprojectable decomposition is essentially equivalent to modular composition or power projection, at least for the choice of parameter $m = 1$.

In what follows, let $\mathsf{E} : \mathbb{N}^2 \to \mathbb{N}$ be such that one can solve problem $\mathbf{P}_1$ above in $\mathsf{E}(n, \delta_1)$ base field operations (in an algebraic model), for triangular sets in n variables. Then, our second main result is the following.

Theorem 2. *Let $\mathbf{T}$ be a triangular set in n variables, with $n \in \{1, 2\}$, that generates a radical ideal. Then, we can compute modular compositions and power projections modulo $\langle \mathbf{T} \rangle$ with parameters $(1, n)$ and size $\delta_{\mathbf{f}} \leq \delta_{\mathbf{T}}$ in time $2\mathsf{E}(4, \delta_{\mathbf{T}}) + O^{\tilde{}}(\delta_{\mathbf{T}})$.*

In other words, if we are able to compute four-variate equiprojectable decompositions efficiently, we can compute modular compositions and power projections efficiently for some small values of the parameters (which cover in particular the most useful case $m = n = 1$, that is, computing $F(G) \bmod T$, for univariate polynomials F, G, T). Note that an entirely similar result holds for the boolean model as well.

Organization of the paper. Section 2 introduces most basic algorithms used in the paper: a reminder on modular composition and power projection for triangular sets in one or two variables, and conversions between univariate and triangular representations. Section 3 gives an algorithm to compute the so-called ϕ-decomposition of a zero-dimensional algebraic set V, that is, a decomposition according to the cardinalities of the fibers of a mapping $\phi : V \to \overline{\mathbb{K}}^m$.

We use this in Section 4 to prove Theorem 1; in that section, we also present experimental results obtained with a Maple implementation. Finally, Section 5 proves Theorem 2.

Previous work. Let us first review previous work for the questions we consider in the algebraic complexity model.

For a triangular set $\mathbf{T}$, some previous algorithms have costs of the form $O^{\tilde{}}(4^n\delta_{\mathbf{T}})$ for multiplication in $R_{\mathbf{T}}$ [31] or $O^{\tilde{}}(K^n\delta_{\mathbf{T}})$ for computing quasi-inverses in $R_{\mathbf{T}}$ [16], for K a large constant. For multiplication, some particular cases with a better cost are discussed in [8]. An algorithm for regularization, a similar question to quasi-inverse, is given in [28, 29]; under a non-degeneracy assumption, its cost grows like $\sum_{2\leq i\leq n} 2^i d_1\cdots d_{i-1}d_i^{i+1}$, up to polylogarithmic factors. In particular, all these algorithms involve an extra factor of the form K^n.

For change of order, previous work includes [9] (which covers more general questions, e.g. in positive dimension), for which we are not aware of a complexity analysis. A close reference to our work is [34]: the results in that paper are restricted to the bivariate case, but use similar techniques; our algorithms are actually a generalization of those in [34].

It is worth discussing in some detail a natural approach to change of order, based on resultant computations. In the simplest case of bivariate systems, changing the order in a triangular set $(T_1(X_1), T_2(X_1, X_2))$ can be done by first computing the resultant $\mathrm{res}(T_1, T_2, X_1)$, so as to eliminate X_1 — this would of course be only the first step of the algorithm, since we would also have to deal with X_2. Still, already this first step may be costly, since the best algorithm we are aware of takes time $O^{\tilde{}}(d_1^2 d_2)$, which can be as large as $O^{\tilde{}}(\delta_{\mathbf{T}}^2)$. An extension to triangular sets in more variables could be done along the lines of [28, 29]; roughly speaking, it may induce costs similar to the one seen above for regularization.

For the problem of computing the equiprojectable decomposition (or more generally, for our question $\mathbf{P}_1$), we are not aware of previous complexity results.

In the boolean model, relying on the results by Kedlaya and Umans mentioned above, we showed in [35] that it is possible to answer some of our questions in $n^2\delta_{\mathbf{T}}^{1+\varepsilon}O^{\tilde{}}(\log(q))$ bit operations, for any fixed $\varepsilon > 0$ (note that exponential terms of the form K^n have disappeared). Those results addressed multiplication in $R_{\mathbf{T}}$ and some restricted forms of inversion and change of order, but did not consider any issues related to equiprojectable decomposition.

Extension. To conclude this introduction, let us mention the following question, which is a natural extension of the discussion in this paper (we owe this to one of the referees): given a triangular set $\mathbf{T}$ that describes an algebraic set $V \subset \overline{\mathbb{K}}^n$, and a polynomial mapping $\varphi = (\varphi_1, \ldots, \varphi_m) : \overline{\mathbb{K}}^n \to \overline{\mathbb{K}}^m$, compute the equiprojectable decomposition of the zero-dimensional algebraic set $\varphi(V) \subset \mathbb{K}^m$.

For instance, as pointed out by the referee, with $m = n = 2$, taking $\varphi(x_1, x_2) = (x_2, x_1)$ amounts to bivariate change of order; taking $m = 1$, so that $\varphi(\mathbf{x}) = (\varphi_1(\mathbf{x}))$ for some polynomial φ_1, allows us to compute the (square-free decomposition of the) characteristic polynomial of φ_1 modulo $\langle\mathbf{T}\rangle$.

This operation can be done by introducing new dummy variables Y_i, adding new equations $Y_i - \varphi_i$ to $\mathbf{T}$ (with $Y_i > X_j$ for all i, j) and applying the above change the order algorithm. When all polynomials $(\varphi_1, \ldots, \varphi_m)$ defining φ are reduced modulo $\langle \mathbf{T} \rangle$, the complexity analysis follows from Theorem 1; when they are not, we should take into account the time needed to first reduce them modulo $\langle R_{\mathbf{T}} \rangle$.

2 Notations and known results

In this section, we first recall a few results from the literature, and describe algorithms for bivariate modular composition and power projection (thereby proving the claim made in the introduction regarding the cost of these operations in an algebraic model). In a second subsection, we discuss the representation of zero-dimensional algebraic sets by means of *univariate representations*, and give some basic algorithms for this data structure.

2.1 Basic algorithms

In this subsection, we let $\mathbb{A}$ denote either $\mathbb{K}[X_1]$ or $\mathbb{K}[X_1, X_2]$ and we consider a triangular set $\mathbf{T}$ in $\mathbb{A}$; we write as usual $R_{\mathbf{T}} = \mathbb{A}/\langle \mathbf{T} \rangle$ and we let V be the zero-set of $\mathbf{T}$, in either $\overline{\mathbb{K}}$ or $\overline{\mathbb{K}}^2$. We will describe a few useful algorithms for computing in $R_{\mathbf{T}}$; most of them actually extend to $\mathbb{A} = \mathbb{K}[X_1, \ldots, X_n]$, but the costs would then involve an extra factor of the form K^n, for some constant K.

In all this subsection, we will assume that the characteristic of $\mathbb{K}$ is equal to 0 or greater than $\delta_{\mathbf{T}}$.

Multiplication and transposed multiplication. Using univariate multiplication, we can do the following in $O(\mathsf{M}(\delta_{\mathbf{T}}))$ operations in $\mathbb{K}$:

- *modular multiplication:* given $A, B \in R_{\mathbf{T}}$, compute $AB \in R_{\mathbf{T}}$
- *transposed multiplication:* given a linear form $\ell : R_{\mathbf{T}} \to \mathbb{K}$ and $A \in R_{\mathbf{T}}$, compute the linear form $A \cdot \ell : R_{\mathbf{T}} \to \mathbb{K}$ defined by $(A \cdot \ell)(B) = \ell(AB)$.

See for instance [19] and [34] for a proof.

Modular composition. In this paragraph, we discuss modular composition with parameters (m, n), with $m = 2$: given $F \in \mathbb{K}[Y_1, Y_2]$, with $\deg(F, Y_1) < f_1$ and $\deg(F, Y_2) < f_2$, and given G_1, G_2 in $R_{\mathbf{T}}$, this amounts to compute $F(G_1, G_2) \in R_{\mathbf{T}}$. For $(m, n) = (1, 1)$, that is, with F univariate and $\mathbf{T} = (T_1) \in \mathbb{K}[X_1]$, the best-known algorithm is due to Brent and Kung [10]. We present here a straightforward generalization, under the simplifying assumption that $f_1 f_2 \leq \delta_{\mathbf{T}}$. Note that solving this problem for $m = 2$ actually also solves it for $m = 1$, by taking $f_2 = 1$.

We let $\varepsilon_1, \varepsilon_1'$ and $\varepsilon_2, \varepsilon_2'$ be positive integers such that $\varepsilon_1\varepsilon_1' \geq f_1$ and $\varepsilon_2\varepsilon_2' \geq f_2$ (to be specified below), and we decompose F into "rectangular slices" of the form

$$F = \sum_{i_1<\varepsilon_1, i_2<\varepsilon_2} F_{i_1,i_2}(Y_1, Y_2) Y_1^{\varepsilon_1' i_1} Y_2^{\varepsilon_2' i_2},$$

with each F_{i_1,i_2} in $\mathbb{K}[Y_1, Y_2]$ and satisfying $\deg(F_{i_1,i_2}, Y_1) < \varepsilon_1'$ and $\deg(F_{i_1,i_2}, Y_2) < \varepsilon_2'$. Then, we have

$$F(G_1, G_2) = \sum_{i_1<\varepsilon_1, i_2<\varepsilon_2} \varphi_{i_1,i_2} \gamma_1^{i_1} \gamma_2^{i_2},$$

with $\varphi_{i_1,i_2} = F_{i_1,i_2}(G_1, G_2)$, $\gamma_1 = G_1^{\varepsilon_1'}$ and $\gamma_2 = G_2^{\varepsilon_2'}$, all equalities being modulo $\langle \mathbf{T} \rangle$. This gives the following algorithm:

1. Compute all powers $G_1^{j_1} G_2^{j_2}$ mod $\langle \mathbf{T} \rangle$, for $j_1 < \varepsilon_1'$, $j_2 < \varepsilon_2'$, γ_1, as well as γ_2. This costs a total of $\varepsilon_1'\varepsilon_2'$ multiplications in $R_{\mathbf{T}}$ (one per monomial).

2. We deduce all φ_{i_1,i_2} by linear algebra: given (i_1, i_2), $\varphi_{i_1,i_2} = F_{i_1,i_2}(G_1, G_2)$ mod $\langle \mathbf{T} \rangle$ is obtained by doing the matrix-vector product $M_G V_{i_1,i_2}$, where M_G is the matrix of size $(\delta_{\mathbf{T}} \times \varepsilon_1'\varepsilon_2')$ that contains the coefficients of all $G_1^{j_1} G_2^{j_2}$ mod $\langle \mathbf{T} \rangle$ (in columns) and V_{i_1,i_2} is the column-vector of coefficients of F_{i_1,i_2}; to do it for all (i_1, i_2), we end up doing one matrix product of size $(\delta_{\mathbf{T}} \times \varepsilon_1'\varepsilon_2') \times (\varepsilon_1'\varepsilon_2' \times \varepsilon_1\varepsilon_2)$.

3. We eventually get $F(G_1, G_2)$ mod $\langle \mathbf{T} \rangle$ by using Horner's scheme twice: first, to compute

$$\varphi_{i_1} = \sum_{i_2<\varepsilon_2} \varphi_{i_1,i_2} \gamma_2^{i_2} \text{ mod } \langle \mathbf{T} \rangle, i_1 < \varepsilon_1;$$

   this is done with $\varepsilon_2 - 1$ multiplications modulo $\langle \mathbf{T} \rangle$. Then to compute

$$F(G_1, G_2) \text{ mod } \langle \mathbf{T} \rangle = \sum_{i_1<\varepsilon_1} \varphi_{i_1} \gamma_1^{i_1}.$$

   The total is $\varepsilon_1\varepsilon_2 - 1$ multiplications modulo $\langle \mathbf{T} \rangle$.

In total, we do at most $\varepsilon_1\varepsilon_2 + \varepsilon_1'\varepsilon_2'$ multiplications modulo $\langle \mathbf{T} \rangle$ and a matrix product of size $(\delta_{\mathbf{T}} \times \varepsilon_1'\varepsilon_2') \times (\varepsilon_1'\varepsilon_2' \times \varepsilon_1\varepsilon_2)$. We take $\varepsilon_1 \simeq \varepsilon_1' \simeq {f_1}^{1/2}$ and $\varepsilon_2 \simeq \varepsilon_2' \simeq {f_2}^{1/2}$, and we write $\varphi = f_1 f_2$. Then, we end up with $O(\varphi^{1/2})$ multiplications modulo $\langle \mathbf{T} \rangle$ and a matrix product of size $(\delta_{\mathbf{T}} \times \varphi^{1/2}) \times (\varphi^{1/2} \times \varphi^{1/2})$. Since by assumption $\varphi \leq \delta_{\mathbf{T}}$, the cost is $O(\mathsf{M}(\delta_{\mathbf{T}})\delta_{\mathbf{T}}^{1/2} + \delta_{\mathbf{T}}^{(\omega+1)/2})$, which is $O(\delta_{\mathbf{T}}^{(\omega+1)/2})$.

Power projection. Next, we present an algorithm to solve the power projection problem for parameters (m, n), with $m = 2$. Recall that power projection takes as input a linear form $\ell \in R_{\mathbf{T}}^*$, G_1 and G_2 in $R_{\mathbf{T}}$, some bounds (f_1, f_2), and outputs the sequence $(\ell(G_1^{i_1} G_2^{i_2} \text{ mod } \langle \mathbf{T} \rangle))_{i_1<f_1, i_2<f_2}$.

For parameters $(m, n) = (1, 1)$, the algorithm is due to Shoup [41] and an extension to $n = 2$ is due to Kaltofen [25]; these algorithms are dual to Brent-Kung's algorithm. As for modular composition, we present a straightforward generalization to $m = 2$, with the assumption $f_1 f_2 \leq \delta_{\mathbf{T}}$. The algorithm is obtained by simply transposing steps 2 and 3 of the modular composition algorithm (step 1 is kept as a preprocessing phase), so the cost estimate is therefore the same.

Let $\varepsilon_1, \varepsilon_1', \varepsilon_2, \varepsilon_2'$ be as above, and let again $\gamma_1 = G_1^{\varepsilon_1'} \bmod \langle \mathbf{T} \rangle$ and $\gamma_2 = G_2^{\varepsilon_2'} \bmod \langle \mathbf{T} \rangle$. For $i_1 < \varepsilon_1$ and $i_2 < \varepsilon_2$, let

$$\ell_{i_1,i_2} = (\gamma_1^{i_1} \gamma_2^{i_2}) \cdot \ell,$$

where the "dot" denotes transposed multiplication. It follows that for $j_1 < \varepsilon_1'$ and $j_2 < \varepsilon_2'$, we have

$$\begin{aligned} \ell_{i_1,i_2}(G_1^{j_1} G_2^{j_2} \bmod \langle \mathbf{T} \rangle) &= \ell(\gamma_1^{i_1} \gamma_2^{i_2} G_1^{j_1} G_2^{j_2} \bmod \langle \mathbf{T} \rangle) \\ &= \ell(G_1^{\varepsilon_1' i_1 + j_1} G_2^{\varepsilon_2' i_2 + j_2} \bmod \langle \mathbf{T} \rangle). \end{aligned}$$

Thus, we compute all $\ell_{i_1,i_2}(G_1^{j_1} G_2^{j_2} \bmod \langle \mathbf{T} \rangle)$, for $i_1 < \varepsilon_1$, $i_2 < \varepsilon_2$, $j_1 < \varepsilon_1'$ and $j_2 < \varepsilon_2'$, as this gives us the values we need.

1. First, we compute all powers $G_1^{j_1} G_2^{j_2} \bmod \langle \mathbf{T} \rangle$, with $j_1 < \varepsilon_1'$ and $j_2 < \varepsilon_2'$. This costs $\varepsilon_1' \varepsilon_2' - 1$ multiplications modulo $\langle \mathbf{T} \rangle$. We need as well γ_1 and γ_2, for two extra multiplications.

2. Then, we compute the linear forms ℓ_{i_1,i_2} incrementally by $\ell_{i_1+1,i_2} = \gamma_1 \cdot \ell_{i_1,i_2}$ and $\ell_{i_1,i_2+1} = \gamma_2 \cdot \ell_{i_1,i_2}$; each of them takes one transposed multiplication.

3. We finally compute all $\ell_{i_1,i_2}(G_1^{j_1} G_2^{j_2} \bmod \langle \mathbf{T} \rangle)$ by computing the matrix product $M_L M_G$, where M_G is the same $(\delta_{\mathbf{T}} \times \varepsilon_1' \varepsilon_2')$ matrix as in the modular composition case, and M_L is the $(\varepsilon_1 \varepsilon_2 \times \delta_{\mathbf{T}})$ matrix giving the coefficients of the ℓ_{i_1,i_2}.

In total, we do $\varepsilon_1 \varepsilon_2 + \varepsilon_1' \varepsilon_2'$ (transposed) multiplications modulo $\langle \mathbf{T} \rangle$ and a matrix product of size $(\varepsilon_1 \varepsilon_2 \times \delta_{\mathbf{T}}) \times (\delta_{\mathbf{T}} \times \varepsilon_1' \varepsilon_2')$. Let $\varphi = f_1 f_2$. With $\varepsilon_1 \simeq \varepsilon_1' \simeq {f_1}^{1/2}$ and $\varepsilon_2 \simeq \varepsilon_2' \simeq {f_2}^{1/2}$, we end up with $2\varphi^{1/2}$ (transposed) multiplications modulo $\langle \mathbf{T} \rangle$ and a matrix product of size $(\varphi^{1/2} \times \delta_{\mathbf{T}}) \times (\delta_{\mathbf{T}} \times \varphi^{1/2})$. Since we assumed $\varphi \leq \delta_{\mathbf{T}}$, the cost is $O(\mathsf{M}(\delta_{\mathbf{T}}) \delta_{\mathbf{T}}^{1/2} + \delta_{\mathbf{T}}^{(\omega+1)/2})$, which is $O(\delta_{\mathbf{T}}^{(\omega+1)/2})$.

Together with the former algorithm for modular composition, this shows indeed that we can take $\mathsf{C}(d)$ in $O(d^{(\omega+1)/2})$, as claimed in the introduction.

Trace and characteristic polynomial. For $A \in R_{\mathbf{T}}$, we let $\tau(A) \in \mathbb{K}$ and $\chi_A \in \mathbb{K}[X]$ be respectively the trace and characteristic polynomial of the multiplication-by-A endomorphism of $R_{\mathbf{T}}$. We discuss briefly how to compute these objects.

The trace $\tau : R_{\mathbf{T}} \to \mathbb{K}$ is actually a $\mathbb{K}$-linear form. Using fast multiplication, it is possible to determine its values on the monomial basis $B_{\mathbf{T}}$ of $R_{\mathbf{T}}$ using $O(\mathsf{M}(\delta_{\mathbf{T}}))$ operations [34].

Since $R_{\mathbf{T}}$ is a reduced algebra, by [14, Prop. 4.2.7] (sometimes called Stickelberger's Theorem), we have

$$\chi_A = \prod_{\mathbf{x} \in V} (X - A(\mathbf{x})). \tag{1}$$

We can compute χ_A using power projection (this is well-known, see e.g. [36] for a presentation of this algorithm in a more general context). We start by computing the values of the trace τ on the monomial basis $B_{\mathbf{T}}$. By power projection, we can then compute the traces $\tau(A^i)$, for $i = 0, \ldots, \delta_{\mathbf{T}} - 1$, which are the power sums of χ_A. By our assumption on the characteristic of $\mathbb{K}$, we can then use Newton iteration (for the exponential of a power series) to deduce the characteristic polynomial χ_A of A in time $O(\mathsf{M}(\delta_{\mathbf{T}}))$, see [10, 37]. By our assumption that $\mathsf{M}(d)\log(d) = O(\mathsf{C}(d))$, we deduce that the power projection is the dominant part of this algorithm, so the total cost is $O(\mathsf{C}(\delta_{\mathbf{T}}))$.

Inverse modular composition. A second use of trace formulas is an inverse modular composition. Given A and B in $R_{\mathbf{T}}$, we want to compute a polynomial $U \in \mathbb{K}[X]$, if it exists, such that $B = U(A)$ in $R_{\mathbf{T}}$. In [35], following ideas from [40, 36], we recall an algorithm that computes a polynomial U in time $O(\mathsf{C}(\delta_{\mathbf{T}}))$, such that if B can indeed be written as a polynomial in A, then $B = U(A)$; note that the analysis uses the assumption that $\mathsf{M}(d)\log(d)$ is in $O(\mathsf{C}(d))$, and our assumption on the characteristic of $\mathbb{K}$. Verifying whether $B = U(A)$ can be done for another modular composition, so the total time is $O(\mathsf{C}(\delta_{\mathbf{T}}))$.

2.2 Univariate representations

We next turn to questions related to the representation of zero-dimensional algebraic sets. We have already introduced triangular representations; in this subsection, we will discuss *univariate representations*, which rely on the introduction of a linear combination of all variables, and for which most of our questions are easy to solve.

In all that follows, the *degree* $\deg(V)$ of a zero-dimensional algebraic set V simply denotes its cardinality.

Definition. Let $V \subset \overline{\mathbb{K}}^n$ be a zero-dimensional algebraic set of degree δ, defined over $\mathbb{K}$, and let I be its defining ideal.

A *univariate representation* $\mathscr{U} = (P, \mathbf{U}, \mu)$ of V consists of a polynomial $P \in \mathbb{K}[X]$, a sequence of polynomials $\mathbf{U} = (U_1, \ldots, U_n) \in \mathbb{K}[X]$, with $\deg(U_i) < \deg(P)$ for all i, as well as a linear form $\mu = \mu_1 X_1 + \cdots + \mu_n X_n$ with coefficients in $\mathbb{K}$, such that

$$\begin{array}{rccc} \Psi_{\mathscr{U}} : & \mathbb{K}[\mathbf{X}]/I & \to & \mathbb{K}[X]/\langle P\rangle \\ & X_1, \ldots, X_n & \mapsto & U_1, \ldots, U_n \\ & \mu_1 X_1 + \cdots + \mu_n X_n & \leftarrow\!\shortmid & X \end{array} \tag{2}$$

is an isomorphism: this allows one to transfer most algebraic operations to the ring $\mathbb{K}[X]/\langle P\rangle$, where arithmetic is easy. In particular, the definition implies that P is squarefree, and that it is the characteristic polynomial of μ in $\mathbb{K}[\mathbf{X}]/I$. Thus, we have

$$P = \prod_{\mathbf{x} \in V} (X - \mu(\mathbf{x}))$$

and $x_i = U_i(\mu(\mathbf{x}))$ for all $\mathbf{x} = (x_1, \ldots, x_n)$ in V and $i \leq n$.

This kind of representation is familiar: up to a few differences, it is used for instance in [20, 2, 36, 21, 22].

We will call a linear form $\mu = \mu_1 X_1 + \cdots + \mu_n X_n$ a *separating element* for V if for all distinct $\mathbf{x}, \mathbf{x}'$ in V, $\mu(\mathbf{x}) \neq \mu(\mathbf{x}')$. One easily sees that μ is separating if and only if V admits a univariate representation of the form $\mathscr{U} = (P, \mathbf{U}, \mu)$, if and only if the characteristic polynomial P of μ in $\mathbb{K}[\mathbf{X}]/I$ is squarefree. This characterization implies the following well-known lemma.

Lemma 1. *If the characteristic of $\mathbb{K}$ is at least δ^2, and if $\mu_1, \ldots, \mu_n$ are chosen uniformly at random in $\mathfrak{S} = \{0, \ldots, \delta^2 - 1\}$, the probability that $\mu = \mu_1 X_1 + \cdots + \mu_n X_n$ be a separating element for V is at least $1/2$. The same remains true if μ_n is set to 1 and $\mu_1, \ldots, \mu_{n-1}$ are chosen uniformly at random in $\mathfrak{S}$.*

Proof. The above characterization implies that μ is separating if and only if $(\mu_1, \ldots, \mu_n)$ does not cancel the polynomial Δ of degree $\delta(\delta-1)/2$ defined by

$$\Delta(M_1, \ldots, M_n) = \prod_{\mathbf{x}, \mathbf{x}' \in V,\ \mathbf{x} \neq \mathbf{x}'} \left(M_1(x_1 - x_1') + \cdots + M_n(x_n - x_n')\right).$$

The Zippel-Schwartz lemma implies that there are at most $\delta^{2n}/2$ roots of Δ in $\mathfrak{S}^n$, and the first statement follows. To get the second one, observe that Δ is homogeneous, so we can set $M_n = 1$ without loss of generality; the second statement follows. □

Useful algorithms. We conclude this section with a few algorithms for univariate representations. Most of what is here is standard, or at least folklore, although the complexity statements themselves may be new (e.g., one finds in [22] an equivalent of Lemma 2 below, but with a quadratic running time).

Lemma 2. *Given a univariate representation $\mathscr{U} = (P, \mathbf{U}, \mu)$ of an algebraic set $V \subset \overline{\mathbb{K}}^n$ defined over $\mathbb{K}$, and a linear form $\nu = \nu_1 X_1 + \cdots + \nu_n X_n$ with coefficients in $\mathbb{K}$, one can decide whether ν is a separating element for V, and if so compute the corresponding univariate representation $\mathscr{V} = (Q, \mathbf{V}, \nu)$, in time $O(n\mathsf{C}(\delta))$, with $\delta = \deg(V)$, provided that the characteristic of $\mathbb{K}$ is equal to 0 or greater than δ.*

Proof. Let $\Psi_{\mathscr{U}}$ be as in Equation (2). We first compute $N = \Psi_{\mathscr{U}}(\nu) = \nu_1 U_1 + \cdots + \nu_n U_n$; this takes only $O(n\delta)$ operations.

Next, we compute the characteristic polynomial Q of N in $\mathbb{K}[X]/\langle P \rangle$; as mentioned before, ν is a separating element for V if and only if Q is squarefree. We have seen that computing Q takes time $O(\mathsf{C}(\delta))$; testing squarefreeness takes time $O(\mathsf{M}(\delta)\log(\delta))$, which is by assumption $O(\mathsf{C}(\delta))$.

When μ is separating, we can use the algorithm for inverse modular composition, to find polynomials $V_1, \ldots, V_n$ such that $U_i = V_i(N) \bmod Q$ holds for all i; then, we have found $\mathscr{V} = (Q, (V_1, \ldots, V_n), \nu)$. In view of the results recalled in Subsection 2.1 on inverse modular composition, the total time is $O(n\mathsf{C}(\delta))$. □

Lemma 3. *Given univariate representations $\mathscr{U} = (P, \mathbf{U}, \mu)$ and $\mathscr{V} = (Q, \mathbf{V}, \nu)$ of two algebraic sets $V \subset \overline{\mathbb{K}}^n$ and $W \subset \overline{\mathbb{K}}^n$ defined over $\mathbb{K}$, one can compute univariate representations of either $V \cup W$ or $V - W$ in expected time $O(n\mathsf{C}(\delta))$, with $\delta = \deg(V) + \deg(W)$, provided that the characteristic of $\mathbb{K}$ is equal to 0 or greater than δ^2.*

Proof. The following process is repeated until success. We pick a random linear form $\lambda = \lambda_1 X_1 + \cdots + \lambda_n X_n$ with coefficients in $\mathfrak{S} = \{0, \ldots, \delta^2 - 1\}$, and apply the algorithm of Lemma 2 to $(\mathscr{U}, \lambda)$ and $(\mathscr{V}, \lambda)$. The cost of this step is $O(n\mathsf{C}(\delta))$. In case of success, we let $\mathscr{U}' = (P', \mathbf{U}', \lambda)$ and $\mathscr{V}' = (Q', \mathbf{V}', \lambda)$ be the resulting univariate representations of V and W; if either subroutine fails, we pick another λ.

At this stage, λ is separating for both V and W. Now, we compute the polynomial $S = \gcd(P', Q')$, as well as $P'' = P'/S$ and $Q'' = Q'/S$. We also compute

$$U_i'' = U_i' \bmod P'', \quad T_i = U_i' \bmod S, \quad W_i = V_i' \bmod S, \quad V_i'' = V_i' \bmod Q''$$

for all i. Using fast GCD and fast Euclidean division, this can be done in time $O(\mathsf{M}(\delta)\log(\delta) + n\mathsf{M}(\delta))$, which is negligible compared to the cost of the first step.

These polynomials will allow us to determine whether λ is a separating element for $V \cup W$. This is the case if and only if for any common root α of P' and Q', the equalities $U_i'(\alpha) = V_i'(\alpha)$ hold for all $i \leq n$, that is, if $T_i = W_i$ holds for all i. Doing this test takes time $O(n\delta)$; if not all equalities hold, we pick another λ. Note that if λ is separating for $V \cup W$, it is separating for $V - W$.

Assuming λ is a separating element for $V \cup W$, we obtain a univariate representation for $V \cup W$ by computing $(P''SQ'', (E_1, \ldots, E_n), \lambda)$, where E_i is obtained by applying the Chinese Remainder Theorem to (U_i'', T_i, V_i'') and moduli (P'', S, Q''), for all i. Computing these polynomials takes time $O(n\mathsf{M}(\delta)\log(\delta))$, which is again $O(n\mathsf{C}(\delta))$. Similarly, we obtain a univariate representation for $V - W$ as $(P'', (U_1'', \ldots, U_n''), \lambda)$.

By Lemma 1, we expect to test $O(1)$ choices of λ on average (precisely, at most two, on average) before finding a suitable one. As a consequence, the expected running time is $O(n\mathsf{C}(\delta))$. □

To conclude this section, we mention the following result about conversions between univariate and triangular representations.

As a preliminary, remember that if $\mathscr{U} = (P, \mathbf{U}, \mu)$ is a univariate representation of an algebraic set V, there exists an isomorphism $\Psi_{\mathscr{U}} : \mathbb{K}[\mathbf{X}]/I(V) \to \mathbb{K}[X]/\langle P \rangle$. If furthermore the defining ideal of V admits a triangular set of generators $\mathbf{T}$ for some variable order $<$, we also have $\mathbb{K}[\mathbf{X}]/I(V) \simeq R_{\mathbf{T}}$. As a result, there exists change-of-basis isomorphisms

$$\Phi_{\mathbf{T},\mathscr{U}} : \mathbb{K}[X]/\langle P \rangle \to R_{\mathbf{T}} \quad \text{and} \quad \Psi_{\mathbf{T},\mathscr{U}} : R_{\mathbf{T}} \to \mathbb{K}[X]/\langle P \rangle,$$

which will be useful in the sequel.

Lemma 4. *Let $V \subset \overline{\mathbb{K}}^n$ be an algebraic set of degree δ, defined over $\mathbb{K}$, and let $I \subset \mathbb{K}[X_1, \ldots, X_n]$ be its defining ideal; suppose that the characteristic of $\mathbb{K}$ is equal to 0 or greater than δ^2. Let finally $<$ be an order on the variables $X_1, \ldots, X_n$ and suppose that I is generated by a triangular set $\mathbf{T}$ for the variable order $<$. Then the following holds:*

- *Given a univariate representation $\mathscr{U} = (P, \mathbf{U}, \mu)$ of V, one can compute the triangular set $\mathbf{T}$ in expected time $O(n^2\mathsf{C}(\delta))$. Given A in $\mathbb{K}[X]/\langle P\rangle$, one can then compute $\Phi_{\mathbf{T},\mathscr{U}}(A) \in R_{\mathbf{T}}$ in time $O(n\mathsf{C}(\delta))$.*

- *Given $\mathbf{T}$, one can compute a univariate representation $\mathscr{U} = (P, \mathbf{U}, \mu)$ of V in expected time $O(n^2\mathsf{C}(\delta))$. Given A in $R_{\mathbf{T}}$, one can then compute $\Psi_{\mathbf{T},\mathscr{U}}(A) \in \mathbb{K}[X]/\langle P\rangle$ in time $O(n\mathsf{C}(\delta))$.*

Proof. We will merely describe the main ideas, so as to highlight the roles of modular composition and power projection. Details are given in [35, Section 5.3 and 6.3], together with worked-out examples (the complexity analysis there is given in the boolean model, but carries over to the algebraic model without difficulty). In both directions, we proceed one variable at a time.

- In the first direction, we change (if needed) the linear form μ, so as to ensure that the coefficient of X_n in μ is equal to 1; this is done in expected time $O(n\mathsf{C}(\delta))$ by means of Lemmas 1 and 2. This mild condition is needed to apply the algorithm of [35]; we still write the input $\mathscr{U} = (P, \mathbf{U}, \mu)$.

  Then, we let $\mu' = \mu'_1X_1 + \cdots + \mu'_{n-2}X_{n-2} + X_{n-1}$ be a random combination of $X_1, \ldots, X_{n-1}$, with coefficients in $\{0, \ldots, \delta^2 - 1\}$, whose coefficient in X_{n-1} is 1. We can then replace the single polynomial $P_n(X) = P(X)$ by a bivariate triangular set

  $$\left|\begin{array}{l} T_{n-1,n}(X, X_n) \\ P_{n-1}(X), \end{array}\right.$$

  where P_{n-1} is the squarefree part of the characteristic polynomial of $\mu'_1U_1 + \cdots + \mu'_{n-2}U_{n-2} + U_{n-1}$ modulo P_n. As we go, we also compute expressions of $U_1, \ldots, U_{n-1}$ as polynomials in μ', to allow the process to continue. In the second step, we introduce a triangular set

  $$\left|\begin{array}{l} T_{n-2,n}(X, X_{n-1}, X_n) \\ T_{n-2,n-1}(X, X_{n-1}) \\ P_{n-2}(X) \end{array}\right.$$

  in three variables X, X_{n-1}, X_n, and so on until we obtain $\mathbf{T}$.

  Using formulas from [34, 35], going from (P_n) to $(P_{n-1}, T_{n-1,n})$ is done by means of power projections with parameters $(1, 1)$ and $(2, 1)$ and size $\delta = \deg(P_n)$, as well as inverse modular compositions, all computed modulo $\langle P_n\rangle$; the total time is $O(n\mathsf{C}(\delta))$. The computation of $(P_{n-1}, T_{n-1,n})$ proceeds as follows:

  - As mentioned above, we obtain P_{n-1} by taking the squarefree part of the characteristic polynomial of $\mu'_1U_1 + \cdots + \mu'_{n-2}U_{n-2} + U_{n-1}$ modulo P_n. We described how to compute this characteristic polynomial in the previous subsection, by means of a power projection with parameters $(1, 1)$; explicitly, we need the traces of elements of the form $(\mu'_1U_1 + \cdots + \mu'_{n-2}U_{n-2} + U_{n-1})^j$ modulo P_n.

– Computing $T_{n-1,n}$ follows a similar but slightly more complicated strategy. Knowing P_{n-1}, we can first determine the degree $\delta' = \deg(T_{n-1,n}, X_n)$ from the equality $\delta' = \deg(P_n, X)/\deg(P_{n-1}, X)$.

We are going to compute the reversal of $T_{n-1,n}$ with respect to X_n, that is, $U_{n-1,n} = X_n^{\delta'} T_{n-1,n}(X, 1/X_n)$. From this polynomial, we can then deduce $T_{n-1,n}$ immediately, by reading the coefficients of $U_{n-1,n}$ backward.

Let $\mathbb{A}$ be the residue class ring $\mathbb{K}[X]/\langle P_n\rangle$ and let us write the power series expansion of the logarithmic derivative of $U_{n-1,n}$ as

$$\frac{\partial U_{n-1,n}/\partial X_n}{U_{n-1,n}} = \sum_{i\geq 0} S_i X_n^i,$$

for some coefficients $S_0, \ldots$; note that this is an equality in $\mathbb{A}[[X_n]]$. Our strategy is to compute $S_0, \ldots, S_{\delta'}$; from them, we obtain $U_{n-1,n}$ by means of a power series exponentiation in $\mathbb{A}[[X_n]]$ (similarly to what we do to compute characteristic polynomials). A crucial fact, proved in [34, Prop. 11], is that the coefficients $S_0, \ldots, S_{\delta'}$ can easily be computed by means of a power projection with parameters $(2, 1)$: explicitly, we need the traces modulo P_n of elements of the form $X^i(\mu'_1 U_1 + \cdots + \mu'_{n-2} U_{n-2} + U_{n-1})^j$, for $0 \leq i \leq \delta'$ and $0 \leq j < \deg(P_{n-1}, X)$. This is the dominant cost, since the power series exponentiation can be done in quasi-linear time.

The change of basis $\mathbb{K}[X]/\langle P_n\rangle \to \mathbb{K}[X, X_n]/\langle P_{n-1}, T_{n-1,n}\rangle$ is done by means of a modular composition with parameters $(1, 2)$ and size $\delta = \deg(P_n)$, computed modulo $\langle P_{n-1}, T_{n-1,n}\rangle$; it takes time $O(\mathsf{C}(\delta))$.

The further steps are done in the same manner. For instance, going from $(P_{n-1}, T_{n-1,n})$ to $(P_{n-2}, T_{n-2,n-1}, T_{n-2,n})$ requires first to compute $(P_{n-2}, T_{n-2,n-1})$, similarly to what we did in the first step. Then, we obtain $T_{n-2,n}$ by applying the change of basis $\mathbb{K}[X]/\langle P_{n-1}\rangle \to \mathbb{K}[X, X_n]/\langle P_{n-2}, T_{n-2,n-1}\rangle$ to all coefficients of $T_{n-1,n}$.

There are n such steps before we reach $\mathbf{T}$; each takes an expected $O(n\mathsf{C}(\delta))$, so the total time is an expected $O(n^2\mathsf{C}(\delta))$.

Staring from A in $\mathbb{K}[X]/\langle P\rangle$, we obtain its image in $R_{\mathbf{T}}$ by computing its representations in $\mathbb{K}[X, X_n]/\langle P_{n-1}, T_{n-1,n}\rangle$, and so on. Each conversion is done as above by means of modular compositions with parameters $(1, 2)$ and takes time $O(\mathsf{C}(\delta))$; the total number of operations is thus $O(n\mathsf{C}(\delta))$.

- To compute a univariate representation starting from a triangular set $\mathbf{T} = (T_1, \ldots, T_n)$, we follow the same process backward. Starting from $(T_{n,1}, \ldots, T_{n,n}) = (T_1, \ldots, T_n)$, we first work with $(T_{n,1}, T_{n,2})$, and find a univariate representation for these two polynomials; this gives us the triangular set in $n-1$ variables $(P_{n-1}, T_{n-1,3}, \ldots, T_{n-1,n})$. We continue until we reach a single polynomial P_n, which we will simply write P.

The polynomial $P_{n-1}(X)$ is the characteristic polynomial of a random combination of X_1, X_2 with coefficients in $\{0, \ldots, \delta^2 - 1\}$, computed modulo $\langle T_{n,1}, T_{n,2}\rangle$; all other polynomials $T_{n-1,j}$ are obtained by applying the change-of-basis $\mathbb{K}[X_1, X_2]/\langle T_{n,1}, T_{n,2}\rangle \to \mathbb{K}[X]/\langle P_{n-1}\rangle$.

This first step requires a power projection with parameters $(1, 2)$, as well as modular compositions with parameters $(2, 1)$, and the cost is an expected $O(n\mathsf{C}(\delta))$. Since there are n such steps, the total cost is then an expected $O(n^2\mathsf{C}(\delta))$. The change-of-basis $R_{\mathbf{T}} \to \mathbb{K}[X]/\langle P\rangle$ is obtained similarly by means of modular compositions, and takes time $O(n\mathsf{C}(\delta))$.

□

The following example, taken from [35], illustrates the previous lemma. Our base field is $\mathbb{F}_{101}$, with variables $X_1 < X_2 < X_3$. We consider the algebraic set V defined by the triangular set $\mathbf{T} = (T_1, T_2, T_3)$ given by

$$\mathbf{T} \left| \begin{array}{l} T_3 = X_3^2 + 100X_1 \\ T_2 = X_2^2 + X_1 \\ T_1 = X_1^2 + 1. \end{array} \right.$$

This algebraic set admits the univariate representation $\mathscr{U} = (P, \mathbf{U}, \mu)$ with $\mu = X_1 + X_2 + X_3$ and

$$\begin{aligned} P &= X^8 + 4X^6 + 99X^4 + 52X^2 + 9 \\ U_1 &= X^7 + 64X^5 + 96X^3 + 5X \\ U_2 &= 50X^7 + 30X^6 + 69X^5 + 30X^4 + 53X^3 + 14X^2 + 99X + 78 \\ U_3 &= 50X^7 + 71X^6 + 69X^5 + 71X^4 + 53X^3 + 87X^2 + 99X + 23. \end{aligned}$$

Let us write $P_3 = P$. Starting from $\mathscr{U} = (P, \mathbf{U}, \mu)$, we first show how to compute $\mathbf{T}$. We take $\mu' = X_1 + X_2$. Then, the polynomial P_2 is the minimal polynomial of $U_1 + U_2$ modulo P_3, which is

$$P_2 = X^4 + 2X^2 + 97X + 2;$$

the expressions of X_1 and X_2 in terms of μ' are respectively $68X^3 + 34X^2 + 2X + 32$ and $33X^3 + 67X^2 + 100X + 69$. The polynomial $T_{2,3}$ gives the degree-2 minimal polynomial of X_3 above P_2; it turns out to be $X_3^2 + 33X^3 + 67X^2 + 99X + 69$, so that we have obtained

$$\left| \begin{array}{l} T_{2,3} = X_3^2 + 33X^3 + 67X^2 + 99X + 69 \\ P_2 = X^4 + 2X^2 + 97X + 2. \end{array} \right.$$

The next step would start by computing the minimal polynomial of X_1, that is, of $68X^3 + 34X^2 + 2X + 32$ modulo P_2, and so on.

In the converse direction, we start from $\mathbf{T} = (T_1, T_2, T_3)$, which we rewrite as $(T_{3,1}, T_{3,2}, T_{3,3})$. First, we compute a univariate representation of the bivariate triangular set $(T_{3,1}, T_{3,2})$: taking the same linear form $X_1 + X_2$ as above, it thus consists of the polynomial $P_2 = X^4 + 2X^2 +$

$97X + 2$, and the parametrizations $68X^3 + 34X^2 + 2X + 32$ and $33X^3 + 67X^2 + 100X + 69$ of respectively X_1 and X_2 we saw above. To deduce the triangular set

$$\left| \begin{array}{l} T_{2,3} = X_3^2 + 33X^3 + 67X^2 + 99X + 69 \\ P_2 = X^4 + 2X^2 + 97X + 2, \end{array} \right.$$

we take $T_{3,3} = X_3^2 + 100X_1$ and evaluate X_1 at $68X^3 + 34X^2 + 2X + 32$ to obtain $T_{2,3}$. To complete the algorithm, we would then compute a univariate representation of the bivariate triangular set $(P_2, T_{2,3})$.

3 The ϕ-decomposition

In this section, we define the notions of *ϕ-equiprojectable* sets and *ϕ-decomposition* of a zero-dimensional algebraic set $V \subset \overline{\mathbb{K}}^n$, where ϕ is a mapping $\overline{\mathbb{K}}^n \to \overline{\mathbb{K}}^m$. We then give an algorithm to compute the ϕ-decomposition of V, by reducing again this problem to (mainly) modular composition and power projection.

In what follows, we suppose that V is a zero-dimensional algebraic subset of $\overline{\mathbb{K}}^n$ of cardinality δ, defined over $\mathbb{K}$, and we let $I \subset \mathbb{K}[\mathbf{X}] = \mathbb{K}[X_1, \ldots, X_n]$ be its defining ideal. We make the assumption that the characteristic of $\mathbb{K}$ is equal to 0 or greater than δ^2.

We start with the definition of some counting functions. Let ϕ be a mapping $\overline{\mathbb{K}}^n \to \overline{\mathbb{K}}^m$, given by polynomials with coefficients in $\mathbb{K}$. For $\mathbf{x}$ in V, we let $c(V, \mathbf{x}, \phi)$ be the cardinality of the set $\{\mathbf{x}' \in V, \phi(\mathbf{x}') = \phi(\mathbf{x})\}$: this is the number of points $\mathbf{x}'$ in V such that $\phi(\mathbf{x}') = \phi(\mathbf{x})$. Then, we say that V is *ϕ-equiprojectable* if there exists a positive integer d such that for all $\mathbf{x}$ in V, $c(V, \mathbf{x}, \phi) = d$.

In general, we should not expect V to be ϕ-equiprojectable. Then, we define

$$\mathscr{C}(V, \phi, r) = \{\mathbf{x} \in V,\ c(V, \mathbf{x}, \phi) = r\};$$

this is the set of all $\mathbf{x} \in V$ with r points in their ϕ-fiber. Since V is finite, $\mathbf{x} \mapsto c(V, \mathbf{x}, \phi)$ takes only finitely many values on V, say $r_1 < \cdots < r_s$. As a consequence, the sets

$$V_{r_1} = \mathscr{C}(V, \phi, r_1), \quad \ldots, \quad V_{r_s} = \mathscr{C}(V, \phi, r_s) \tag{3}$$

form a partition of V; by construction, all these sets are ϕ-equiprojectable. We will write

$$\mathsf{Dec}(V, \phi) = \{V_{r_1}, \ldots, V_{r_s}\},$$

and we will call this decomposition the *ϕ-decomposition* of V. Although it may not be clear from our definition, all V_{r_i} are in fact defined over $\mathbb{K}$.

Lemma 5. *With notation as in* (3)*, $V_{r_1}, \ldots, V_{r_s}$ are defined over $\mathbb{K}$.*

Proof. We are going to prove that for any $r \geq 1$,

$$\mathscr{C}'(V, \phi, r) = \{\mathbf{x} \in V,\ c(V, \mathbf{x}, \phi) \geq r\}$$

is defined over $\mathbb{K}$. Since $\mathscr{C}(V,\phi,r) = \mathscr{C}'(V,\phi,r) - \mathscr{C}'(V,\phi,r+1)$, and since the set-theoretic difference of two zero-dimensional algebraic sets defined over $\mathbb{K}$ is still defined over $\mathbb{K}$, this will be sufficient to establish our claim.

Fix $r \geq 1$, and let $V^{(r)}$ be the r-fold product $V \times \cdots \times V \subset \overline{\mathbb{K}}^{nr}$; obviously, $V^{(r)}$ is defined over $\mathbb{K}$. Let $(\mathbf{x}_1, \ldots, \mathbf{x}_r)$ be the coordinates on $\overline{\mathbb{K}}^{nr}$, where each $\mathbf{x}_i$ has length n, and let

$$W^{(r)} = V^{(r)} - \cup_{1 \leq i < j \leq r} \Delta_{i,j},$$

where $\Delta_{i,j}$ is defined by $\mathbf{x}_i = \mathbf{x}_j$. Again, $W^{(r)}$ is defined over $\mathbb{K}$, and $(\mathbf{x}_1, \ldots, \mathbf{x}_r)$ is in $W^{(r)}$ if and only if all $\mathbf{x}_i$ are in V and pairwise distinct. Finally, we define

$$Z^{(r)} = W^{(r)} \cap_{1 \leq i < j \leq r} V(\phi(\mathbf{x}_i) - \phi(\mathbf{x}_j));$$

then, $\mathscr{C}'(V,\phi,r)$ is the projection of $Z^{(r)}$ on the first factor $\overline{\mathbb{K}}^n$, so it is indeed defined over $\mathbb{K}$, as claimed. □

Before discussing an algorithm that computes $\mathsf{Dec}(V,\phi)$, we prove a simple lemma that will be used in the next section.

Lemma 6. *Consider two mappings $\phi : \overline{\mathbb{K}}^n \to \overline{\mathbb{K}}^m$ and $\psi : \overline{\mathbb{K}}^n \to \overline{\mathbb{K}}^p$, such that $\psi = f \circ \phi$, for some mapping $f : \overline{\mathbb{K}}^m \to \overline{\mathbb{K}}^p$, and suppose that V is ϕ-equiprojectable. Then any V' in $\mathsf{Dec}(V,\psi)$ is both ϕ-equiprojectable and ψ-equiprojectable.*

Proof. Let d be the common cardinality of the fibers of the restriction of ϕ to V. Let further V' be in $\mathsf{Dec}(V,\psi)$, and let $\mathbf{x}$ be in V'. We will show that $c(V',\mathbf{x},\phi) = d$, thereby establishing that V' is ϕ-equiprojectable (V' is ψ-equiprojectable by construction).

Remember that $c(V',\mathbf{x},\phi)$ is the cardinality of the fiber $F' = \{\mathbf{x}' \in V', \phi(\mathbf{x}') = \phi(\mathbf{x})\}$. We claim that we actually have $F' = F$, with $F = \{\mathbf{x}' \in V, \phi(\mathbf{x}') = \phi(\mathbf{x})\}$. Since by assumption $|F| = d$, proving $F = F'$ is sufficient to prove that $c(V',\mathbf{x},\phi) = d$.

Of course, F' is a subset of F. Conversely, let $\mathbf{x}'$ be in F. Then, $\phi(\mathbf{x}) = \phi(\mathbf{x}')$ and our assumption on ϕ and ψ implies that $\psi(\mathbf{x}) = \psi(\mathbf{x}')$. This implies that $\mathbf{x}'$ is in V', as claimed. □

We now explain how to compute $\mathsf{Dec}(V,\phi)$. For simplicity, we will assume that $m \leq n$, and that ϕ is a simple linear map (the algorithm would not be substantially different in general, but a few extra terms could appear in the cost analysis).

Proposition 1. *Consider an algebraic set $V \subset \overline{\mathbb{K}}^n$ defined over $\mathbb{K}$ and of degree δ, and a univariate representation $\mathscr{U} = (P, \mathbf{U}, \mu)$ of V, and let $\mathsf{Dec}(V,\phi) = \{V_{r_1}, \ldots, V_{r_s}\}$. Suppose that the following conditions are satisfied:*

- *the characteristic of $\mathbb{K}$ is equal to 0 or greater than δ^2,*
- *ϕ is a linear map $\overline{\mathbb{K}}^n \to \overline{\mathbb{K}}^m$, of the form $\phi(x_1, \ldots, x_n) = (x_1, \ldots, x_m)$.*

Then we can compute univariate representations $(P_k, \mathbf{U}_k, \mu)_{1 \leq k \leq s}$ of $V_{r_1}, \ldots, V_{r_s}$ in expected time $O(\mathsf{C}(\delta)(n + \log(\delta)))$.

The rest of this section is devoted to prove this proposition. In what follows, we write $W = \phi(V)$ and, for all $k \leq s$, $W_{r_k} = \phi(V_{r_k})$. We also write $\mathbf{U} = (U_1, \ldots, U_n)$, with all U_i in $\mathbb{K}[X]$. Since for all $\mathbf{x} = (x_1, \ldots, x_n)$ in V we have $x_i = U_i(\mu(\mathbf{x}))$, we deduce that $\phi(\mathbf{x}) = (U_1(\mu(\mathbf{x})), \ldots, U_m(\mu(\mathbf{x})))$ for $\mathbf{x}$ in V.

Step 1. Choose a random linear form $\nu = \nu_1 Y_1 + \cdots + \nu_m Y_m$ with coefficients in $\{0, \ldots, \delta^2 - 1\}$, compute $N = \nu_1 U_1 + \cdots + \nu_m U_m$, and compute the characteristic polynomial χ_N of N in $\mathbb{K}[X]/\langle P \rangle$. Computing N takes time $O(n\delta)$ and computing its characteristic polynomial takes time $O(\mathsf{C}(\delta))$, see Subsection 2.1.

The linear form ν must be a separating element for W. To verify if this is the case, we check whether $U_1, \ldots, U_m$ can be written as polynomials in N modulo P. This is done using the algorithm for inverse modular composition, and takes time $O(m\mathsf{C}(\delta))$, which is $O(n\mathsf{C}(\delta))$. Due to our assumption on the characteristic of $\mathbb{K}$, we need to test an expected $O(1)$ choices of ν before finding a separating element, see Lemma 1.

Remark that for $\mathbf{x}$ in V, $N(\mu(\mathbf{x})) = \nu_1 U_1(\mu(\mathbf{x})) + \cdots + \nu_m U_m(\mu(\mathbf{x})) = \nu(\phi(\mathbf{x}))$.

Step 2. Compute the squarefree decomposition of χ_N; this takes times $O(\mathsf{M}(\delta)\log(\delta))$, see [18, Chapter 14]. Using the previous notation, we claim this decomposition has the form

$$\chi_N = C_1^{r_1} \cdots C_s^{r_s}, \quad \text{with} \quad C_k = \prod_{\mathbf{y} \in W_{r_k}} (X - \nu(\mathbf{y})).$$

Indeed, by Stickelberger's Theorem, we have the factorization

$$\begin{aligned} \chi_N &= \textstyle\prod_{\mathbf{x} \in V} (X - N(\mu(\mathbf{x}))) \\ &= \textstyle\prod_{\mathbf{x} \in V} (X - \nu(\phi(\mathbf{x}))). \end{aligned}$$

For $\mathbf{y} \in W$, let $r(\mathbf{y})$ be the cardinality of the fiber $\phi^{-1}(\mathbf{y}) \cap V$. Then we obtain the factorization

$$\begin{aligned} \chi_N &= \textstyle\prod_{\mathbf{y} \in W} (X - \nu(\mathbf{y}))^{r(\mathbf{y})} \\ &= \textstyle\prod_{k \leq s} \prod_{\mathbf{y} \in W_{r_k}} (X - \nu(\mathbf{y}))^{r_k}, \end{aligned}$$

since by construction the projections W_{r_k} are pairwise disjoint. As ν is separating for W, the linear factors $X - \nu(\mathbf{y})$ are pairwise distinct, which proves our claim.

For future use, note that $\sum_{i \leq s} \deg(C_i) \leq \delta$, since $\chi_N = C_1^{r_1} \cdots C_s^{r_s}$ has degree δ.

Step 3. For $k \leq s$, compute $P_k = \gcd(C_k(N), P)$. We will prove at the end of the section that this can be done in time $O(\mathsf{C}(\delta)\log(\delta))$. That proof will be somewhat lengthy; for the moment, we will only prove that for $k \leq s$, we have

$$P_k = \prod_{\mathbf{x} \in V_{r_k}} (X - \mu(\mathbf{x})). \tag{4}$$

Both sides are squarefree (since they divide P), so to prove our claim it is enough to prove that the roots of P_k are exactly the values $\mu(\mathbf{x})$ for $\mathbf{x}$ in V_{r_k}. As a preliminary remark, recall that for all $\mathbf{x}$ in V, we have $\nu(\phi(\mathbf{x})) = N(\mu(\mathbf{x}))$.

- For $\mathbf{x} = (x_1, \ldots, x_n)$ in V_{r_k}, $\phi(\mathbf{x})$ is in W_{r_k} so $\nu(\phi(\mathbf{x}))$ is a root of C_k. By the remark above, this shows that $\mu(\mathbf{x})$ is a root of $C_k(N)$. But of course $\mu(\mathbf{x})$ is also a root of P, so $\mu(\mathbf{x})$ is a root of P_k.

- Conversely, consider a root α of P_k. Since any root of P_k is a root of P, α is of the form $\mu(\mathbf{x})$ for some $\mathbf{x}$ in V. But by assumption $\alpha = \mu(\mathbf{x})$ is also a root of $C_k(N)$, which means that $\nu(\phi(\mathbf{x}))$ is a root of C_k. In particular, $\nu(\phi(\mathbf{x}))$ is a root of no other $C_{k'}$, because these polynomials are pairwise coprime. This implies that $\phi(\mathbf{x})$ belongs to no other $W_{r_{k'}}$, so it must belong to W_{r_k}; thus, $\mathbf{x}$ is in V_{r_k}.

Note also that we have $P = P_1 \cdots P_s$, all P_k being pairwise coprime.

Step 4. For $k \leq s$ and $j \leq n$, compute $U_{k,j} = U_j \bmod P_k$. This can be done in time $O(n\mathsf{M}(\delta)\log(\delta))$ using fast multiple reduction [18, Chapter 10], which is $O(n\mathsf{C}(\delta))$. Writing $\mathbf{U}_k = (U_{k,1}, \ldots, U_{k,n})$, Eq. (4) shows that for $k \leq s$, $(P_k, \mathbf{U}_k, \mu)$ is a univariate representation of V_{r_k}, so we are done.

Analysis of Step 3. Summing all the costs mentioned above gives the cost estimate claimed in Proposition 1. All that is missing is to prove that, as announced, the cost of computing the polynomials P_k of Step 3 is $O(\mathsf{C}(\delta)\log(\delta))$.

Recall that for all $k \leq s$, $P_k = \gcd(C_k(N), P)$. We cannot compute the polynomials $C_k(N)$, or even $C_k(N) \bmod P$, as there are too many of them: one easily sees that s could be as large as $\sqrt{\delta}$; each polynomial $C_k(N) \bmod P$ requires to store δ field elements, so computing all of them would take time at least $\delta^{1.5}$.

Therefore, we compute the P_k directly, using divide-and-conquer techniques. Given polynomials $A, Q \in \mathbb{K}[X]$, we will write

$$\begin{aligned} \Gamma(A, Q) &= \gcd(A(N),\, Q) && (5)\\ &= \gcd(A(N \bmod Q) \bmod Q,\, Q), && (6) \end{aligned}$$

so that the polynomials we want to compute are $P_1 = \Gamma(C_1, P), \ldots, P_s = \Gamma(C_s, P)$.

Assuming we know $N \bmod Q$, Definition (6) shows that we can compute $\Gamma(A, Q)$ by computing first $A(N \bmod Q) \bmod Q$, then taking its GCD with Q. Since by assumption $\mathsf{M}(d)\log(d)$ is $O(\mathsf{C}(d))$, we can thus obtain $\Gamma(A, Q)$ in time $O(\mathsf{C}(d))$ by modular composition and fast GCD, with $d = \max(\deg(A), \deg(Q))$; we will call this the *plain algorithm*. In particular, we could compute any P_k in time $O(\mathsf{C}(\delta))$. However, as we mentioned above, computing all P_k directly in this manner incurs a cost of the form $s\mathsf{C}(\delta)$, which is too much for our purposes.

The key equality we will use is the following: for any polynomials A, B, we have

$$\Gamma(A, Q) = \Gamma(A, \Gamma(AB, Q)). \tag{7}$$

Indeed, using Definition (5), the left-hand side reads

$$\Gamma(A, Q) = \gcd(A(N),\, Q),$$

whereas the right-hand side is

$$\Gamma(A, \Gamma(AB, Q)) = \gcd(A(N),\, \gcd((AB)(N),\, Q));$$

equality (7) follows from the fact that $\gcd(F_1, G) = \gcd(F_1, \gcd(F_1F_2, G))$ holds for all polynomials F_1, F_2, G.

We are now ready to explain how to complete Step 3. To simplify our presentation, we will assume that s is a power of two, of the form $s = 2^w$; when this is not the case, we can complete $C_1, \ldots, C_s$ by dummy polynomials $C_k = 1$, so as to replace s by the next power of two, without affecting the asymptotic running time.

Step 3.1. We compute the *subproduct tree* (see details below) associated to $C_1, \ldots, C_s$. From [18, Chapter 10], this can be done in time $O(\mathsf{M}(\delta)\log(\delta))$, since we have seen that $\sum_{i\le s} \deg(C_i) \le \delta$. Using our assumption on M and C, this is in $O(\mathsf{C}(\delta))$.

At the top level of the subproduct tree, the root is labelled by $K_{0,1} = C_1 \cdots C_s$; its two children are labelled by $K_{1,1} = C_1 \cdots C_v$ and $K_{1,2} = C_{v+1} \cdots C_s$ with $v = s/2$, and so on. For $j = 0, \ldots, w$, the polynomials the jth level are written $K_{j,i}$, with $i = 1, \ldots, 2^j$, so that $K_{j,i} = K_{j+1,2i-1}K_{j+1,2i}$. At the leaves, for $j = w$, we have $K_{w,i} = C_i$.

In what follows, we are going to compute all polynomials $\Gamma(K_{j,i}, P)$, for $j = 0, \ldots, w$ and $i = 1, \ldots, 2^j$, in a top-down manner. At the leaves, for $j = w$, we will obtain the polynomials $\Gamma(K_{w,i}, P) = \Gamma(C_i, P) = P_i$ we are looking for.

Step 3.2. We compute

$$\gamma_{0,1} = \Gamma(K_{0,1}, P)$$

using the plain algorithm, in time $O(\mathsf{C}(\delta))$, as well as $N_{0,1} = N \bmod \gamma_{0,1}$ in time $O(\mathsf{M}(\delta))$, by fast Euclidean division. The latter cost is negligible.

Step 3.3. For $j = 0, \ldots, w-1$ and $i = 1, \ldots, 2^j$, assuming we know $\gamma_{j,i}$ and $N_{j,i} = N \bmod \gamma_{j,i}$, we compute

$$\gamma_{j+1,2i-1} = \Gamma(K_{j+1,2i-1}, \gamma_{j,i}) \quad \text{and} \quad \gamma_{j+1,2i} = \Gamma(K_{j+1,2i}, \gamma_{j,i})$$

followed by

$$N_{j+1,2i-1} = N_{j,i} \bmod \gamma_{j+1,2i-1} \quad \text{and} \quad N_{j+1,2i} = N_{j,i} \bmod \gamma_{j+1,2i}.$$

Our claim is twofold: first, we will prove that $\gamma_{j,i} = \Gamma(K_{j,i}, P)$ for all j, i; second, we will establish that the total running time is $O(\mathsf{C}(\delta)\log(\delta))$. Note that this is enough to finish the proof of Proposition 1, since we have seen that for $j = w$, we have $\Gamma(K_{w,i}, P) = P_i$.

The proof that $\gamma_{j,i} = \Gamma(K_{j,i}, P)$ is done by induction on j. By definition, this is true for $\gamma_{0,1}$; for $j > 1$, this follows from Equation (7), first taking $A = K_{j+1,2i-1}$, $B = K_{j+1,2i}$ and $Q = P$, then $A = K_{j+1,2i}$, $B = K_{j+1,2i-1}$ and $Q = P$. Since $\gamma_{j+1,2i-1}$ and $\gamma_{j+1,2i}$ divide $\gamma_{j,i}$, we can also prove by induction that $N_{j,i} = N \bmod \gamma_{j,i}$ holds for all j, i.

It remains to do the cost analysis. Since $N_{j,i} = N \bmod \gamma_{j,i}$ is known, we can indeed compute $\gamma_{j+1,2i-1}$ and $\gamma_{j+1,2i}$ from $K_{j+1,2i-1}$, $K_{j+1,2i}$ and $\gamma_{j,i}$ by the plain algorithm in time $O(\mathsf{C}(d_{j,i}))$, where we write

$$d_{j,i} = \max(\deg(K_{j+1,2i-1}), \deg(K_{j+1,2i}), \deg(\gamma_{j,i})) \leq \max(\deg(K_{j,i}), \deg(\gamma_{j,i})).$$

The computation of $N_{j+1,2i-1}$ and $N_{j+1,2i}$ can be done in time $O(\mathsf{M}(\deg(\gamma_{j,i})))$, which is negligible by assumption. Hence, the total cost is, up to a constant factor,

$$\sum_{j=0,\ldots,w-1} \sum_{i=1,\ldots,2^j} \mathsf{C}(\max(\deg(K_{j,i}), \deg(\gamma_{j,i}))).$$

This admits the obvious upper bound

$$\sum_{j=0,\ldots,w-1} \sum_{i=1,\ldots,2^j} \mathsf{C}(\deg(K_{j,i})) + \sum_{j=0,\ldots,w-1} \sum_{i=1,\ldots,2^j} \mathsf{C}(\deg(\gamma_{j,i})).$$

Using the super-linearity of C, we obtain the upper bound

$$\sum_{j=0,\ldots,w-1} \mathsf{C}\left(\sum_{i=1,\ldots,2^j} \deg(K_{j,i})\right) + \sum_{j=0,\ldots,w-1} \mathsf{C}\left(\sum_{i=1,\ldots,2^j} \deg(\gamma_{j,i})\right).$$

To conclude the cost analysis, we will prove the inequalities

$$\sum_{i\leq 2^j} \deg(K_{j,i}) \leq \delta \quad \text{and} \quad \sum_{i\leq 2^j} \deg(\gamma_{j,i}) \leq \delta.$$

These inequalities imply a cost upper bound of the form $\sum_{j=0,\ldots,w-1} \mathsf{C}(\delta)$, up to a constant factor. The claim on the total cost follows, since w is in $O(\log(\delta))$.

- The first inequality $\sum_{i\leq 2^j} \deg(K_{j,i}) \leq \delta$ is a straightforward consequence of the equality $\sum_{i\leq 2^j} \deg(K_{j,i}) = \sum_{i\leq s} \deg(C_i)$, which itself follows from the definition of the subproduct tree, and the fact that $\sum_{i\leq s} \deg(C_i) \leq \delta$.

- To obtain the second inequality $\sum_{i\leq 2^j} \deg(\gamma_{j,i}) \leq \delta$, we start by proving that for fixed j, and for $i \neq i'$, $\gamma_{j,i}$ and $\gamma_{j,i'}$ are coprime. Indeed, we have seen that

  $$\gamma_{j,i} = \gcd(K_{j,i}(N), P),$$

  where $K_{j,i}$ has the form $K_{j,i} = \prod_{\ell\in\kappa_{j,i}} C_\ell$. Here, $\kappa_{j,i}$ is a set of indices which we will not need to make explicit; however, for further use, we note that for $i \neq i'$, $\kappa_{j,i}$ and $\kappa_{j,i'}$ are disjoint. The factorization of $K_{j,i}$ implies that

  $$\gamma_{j,i} = \gcd(\prod_{\ell\in\kappa_{j,i}} C_\ell(N), P).$$

Recall now that the polynomials $P_\ell = \gcd(C_\ell(N), P)$ are pairwise coprime; as a result, the former equality gives

$$\gamma_{j,i} = \prod_{\ell \in \kappa_{j,i}} \gcd(C_\ell(N), P) = \prod_{\ell \in \kappa_{j,i}} P_\ell.$$

Since for fixed j the sets $\kappa_{j,i}$ are pairwise disjoint, and since the polynomials P_ℓ are pairwise coprime, we deduce that for fixed j, the polynomials $\gamma_{j,i}$ themselves are pairwise coprime, as claimed.

Since by construction all $\gamma_{j,i}$ divide P, the product $\prod_{i \leq 2^j} \gamma_{j,i}$ must divide P as well, and the inequality $\sum_{i \leq 2^j} \deg(\gamma_{j,i}) \leq \delta$ follows.

4 Proof of Theorem 1

In this section, we prove Theorem 1. We start by defining equiprojectable sets and the equiprojectable decomposition. The algorithms underlying Theorem 1 are then straightforward applications of the results of the previous section.

4.1 The equiprojectable decomposition

Let $V \subset \overline{\mathbb{K}}^n$ be a zero-dimensional algebraic set defined over $\mathbb{K}$. We suppose that we are given an order $<$ on the variables; up to renaming them, we can suppose that the order is simply $X_1 < \cdots < X_n$. For $1 \leq i \leq n$, we define the projection

$$\begin{array}{rccc} \pi_i : & \overline{\mathbb{K}}^n & \to & \overline{\mathbb{K}}^i \\ & \mathbf{x} = (x_1, \ldots, x_n) & \mapsto & (x_1, \ldots, x_i). \end{array}$$

Then, we say that V is equiprojectable if it is π_i-equiprojectable for $i = 1, \ldots, n$; in other words, V is equiprojectable if all fibers of π_1 on V have a common cardinality δ_1, all fibers of π_2 on V have a common cardinality δ_2, etc.

In general, we should not expect V to be equiprojectable. There are potentially many ways to decompose V into equiprojectable sets; the equiprojectable decomposition will be a canonical partition of V into pairwise disjoint equiprojectable sets, that will all be defined over $\mathbb{K}$.

We will actually define a sequence $\mathsf{Dec}(V, i, <)$, for $i = n, \ldots, 1$, which will all be partitions of V, refining one another. At index n, we write $\mathsf{Dec}(V, n, <) = \{V\}$. Then, for $i < n$, assuming that we have defined

$$\mathsf{Dec}(V, i+1, <) = \{V_{i+1,1}, \ldots, V_{i+1,s_{i+1}}\},$$

we obtain $\mathsf{Dec}(V, i, <)$ by computing the π_i-decomposition of every element in $\mathsf{Dec}(V, i+1, <)$:

$$\mathsf{Dec}(V, i, <) = \cup_{k \leq s_{i+1}} \mathsf{Dec}(V_{i+1,k}, \pi_i),$$

which we rewrite as

$$\mathsf{Dec}(V, i, <) = \{V_{i,1}, \ldots, V_{i,s_i}\}.$$

An easy decreasing induction proves that for $i = 1, \ldots, n$ and $k \leq s_i$, every $V_{i,k}$ is π_j-equiprojectable for $j = i, \ldots, n$:

- For $i = n$, $\mathsf{Dec}(V, n, <)$ is simply $\{V\}$, which is π_n-equiprojectable (since π_n is the identity).
- For $i < n$, assuming that the claim holds for $\mathsf{Dec}(V, i+1, <)$, we prove it for $\mathsf{Dec}(V, i, <)$. To do so, it is enough to take $V_{i+1,k}$ in $\mathsf{Dec}(V, i+1, <)$ and prove that every V' in $\mathsf{Dec}(V_{i+1,k}, \pi_i)$ is π_j-equiprojectable, for $j = i, \ldots, n$.

  Obviously, V' is π_i-equiprojectable. Besides, since by the induction assumption $V_{i+1,k}$ is π_j-equiprojectable for $j = i+1, \ldots, n$, Lemma 6 implies that V' is also π_j-equiprojectable for $j = i+1, \ldots, n$.

Taking $i = 1$, $\mathsf{Dec}(V, 1, <)$ is the *equiprojectable decomposition* of V; we will actually denote it by $\mathsf{Dec}(V, <)$. Dropping the subscript $_1$, we will write

$$\mathsf{Dec}(V, <) = \{V_1, \ldots, V_s\}.$$

This is thus a decomposition of V into pairwise disjoint equiprojectable sets V_j.

Aubry and Valibouze proved in [4] that an algebraic set is equiprojectable if and only if its defining ideal is generated by a triangular set. Besides, by Lemma 5, each V_j is defined over $\mathbb{K}$; thus, its defining ideal is generated by a triangular set $\mathbf{T}^{(j)}$ in $\mathbb{K}[\mathbf{X}]$. As said in the introduction, we will write $\mathscr{D}(V, <)$ to denote the collection of the triangular sets $\{\mathbf{T}^{(1)}, \ldots, \mathbf{T}^{(s)}\}$. In ideal-theoretic terms, the ideals $\langle \mathbf{T}^{(j)} \rangle$ are thus pairwise coprime, and their intersection is the defining ideal I of V, so that $\mathbb{K}[\mathbf{X}]/I \simeq R_{\mathbf{T}^{(1)}} \times \cdots \times R_{\mathbf{T}^{(s)}}$.

The following proposition gives a cost estimate on the computation of the equiprojectable decomposition, using a univariate representation as input.

Proposition 2. *Let $V \subset \overline{\mathbb{K}}^n$ be a zero-dimensional algebraic set defined over $\mathbb{K}$, of degree δ. If the characteristic of $\mathbb{K}$ is equal to 0 or greater than δ^2, given a univariate representation $\mathscr{U}$ of V, we can compute $\mathscr{D}(V, <) = \{\mathbf{T}^{(1)}, \ldots, \mathbf{T}^{(s)}\}$ in expected time $O(n\mathsf{C}(\delta)(n + \log(\delta)))$. Besides, the following change of bases can be done in time $O(n\mathsf{C}(\delta))$:*

- *given A in $\mathbb{K}[X]/\langle P \rangle$, compute its images $(A_1, \ldots, A_s)$ in $R_{\mathbf{T}^{(1)}} \times \cdots \times R_{\mathbf{T}^{(s)}}$;*
- *given $(A_1, \ldots, A_s)$ in $R_{\mathbf{T}^{(1)}} \times \cdots \times R_{\mathbf{T}^{(s)}}$, compute their preimage A in $\mathbb{K}[X]/\langle P \rangle$.*

Proof. Let us write as before $\mathsf{Dec}(V, <) = \{V_1, \ldots, V_s\}$. The algorithm to compute $\mathscr{D}(V, <)$ proceeds in two steps: first, we compute univariate representations of all V_j; secondly, we convert them into triangular sets. As we go, we also explain how to perform the change of basis from A to $(A_1, \ldots, A_s)$, and back.

Step 1. Recall the definition of the sequence $\mathsf{Dec}(V, i, <)$: we have $\mathsf{Dec}(V, n, <) = \{V\}$ and starting from

$$\mathsf{Dec}(V, i+1, <) = \{V_{i+1,1}, \ldots, V_{i+1,s_{i+1}}\},$$

we set

$$\mathsf{Dec}(V, i, <) = \cup_{k \leq s_{i+1}} \mathsf{Dec}(V_{i+1,k}, \pi_i).$$

The first step of the algorithm follows the same loop, and computes univariate representations of all $V_{i,k}$. We set $\mathscr{U}_{n,1} = \mathscr{U}$, and for $i = n-1, \ldots, 1$, we let $\mathscr{U}_{i,1}, \ldots, \mathscr{U}_{i,s_i}$ be the univariate representations obtained by applying the algorithm of Proposition 1 to $\mathscr{U}_{i+1,1}, \ldots, \mathscr{U}_{i,s_{i+1}}$ and π_i. If $\delta_{i+1,k}$ denotes the degree of $V_{i+1,k}$, applying the algorithm of Proposition 1 to $\mathscr{U}_{i+1,k}$ and π_i takes an expected time

$$O(\mathsf{C}(\delta_{i+1,k})(n + \log(\delta_{i+1,k}))).$$

Using the super-linearity of C, and the fact that $\delta_{i+1,1} + \cdots + \delta_{i+1,s_{i+1}} = \delta$, the time spent at index i is seen to be an expected $O(\mathsf{C}(\delta)(n + \log(\delta)))$. Summing over all i, the total time is an expected

$$O(n\mathsf{C}(\delta)(n + \log(\delta))).$$

Let P be the characteristic polynomial of $\mathscr{U}$, and let $P_1, \ldots, P_{s_1}$ be those of $\mathscr{U}_{1,1}, \ldots, \mathscr{U}_{1,s_1}$. Since the separating elements of $\mathscr{U}$ and $\mathscr{U}_{1,1}, \ldots, \mathscr{U}_{1,s_1}$ are the same, we have $P = P_1 \ldots P_{s_1}$. The change of basis $\mathbb{K}[X]/\langle P\rangle \to \mathbb{K}[X]/\langle P_1\rangle \times \cdots \times \mathbb{K}[X]/\langle P_{s_1}\rangle$ is done by multiple reduction, and the inverse conversion is done using the Chinese Remainder Theorem. Using the results of [18, Chapter 10], both conversions take time $O(\mathsf{M}(\delta)\log(\delta))$, which is $O(\mathsf{C}(\delta))$.

Step 2. Starting from $\mathscr{U}_{1,1}, \ldots, \mathscr{U}_{1,s_1}$, we now compute the corresponding triangular sets $\mathbf{T}^{(1)}, \ldots, \mathbf{T}^{(s)}$. This is done by applying Lemma 4, which shows that we can compute each triangular set $\mathbf{T}^{(j)}$ in expected time $O(n^2\mathsf{C}(\delta_j))$, where δ_j is the degree of V_j. Summing over all j and using the super-linearity of the function C gives a total expected time of $O(n^2\mathsf{C}(\delta))$.

Using the notation of Subsection 2.2, the conversion

$$\mathbb{K}[X]/\langle P_1\rangle \times \cdots \times \mathbb{K}[X]/\langle P_{s_1}\rangle \to R_{\mathbf{T}^{(1)}} \times \cdots \times R_{\mathbf{T}^{(s)}}$$

and its inverse are done by applying

$$(\Phi_{\mathbf{T}^{(1)},\mathscr{U}_{1,1}}, \ldots, \Phi_{\mathbf{T}^{(s_1)},\mathscr{U}_{1,s_1}}) \quad \text{and} \quad (\Psi_{\mathbf{T}^{(1)},\mathscr{U}_{1,1}}, \ldots, \Psi_{\mathbf{T}^{(s_1)},\mathscr{U}_{1,s_1}}).$$

By Lemma 4, and using the super-linearity of C, each conversion takes time $O(n\mathsf{C}(\delta))$. □

4.2 Solving question $\mathbf{P}_1$

We can now show how to solve question $\mathbf{P}_1$ stated in the introduction. Given triangular sets $\mathbf{T}^{(1)}, \ldots, \mathbf{T}^{(\ell)}$ and $\mathbf{S}^{(1)}, \ldots, \mathbf{S}^{(r)}$ for an order $<$, and a target order $<'$, we want to compute $\mathscr{D}(V, <')$, with

$$V = V(\mathbf{T}^{(1)}) \cup \cdots \cup V(\mathbf{T}^{(\ell)}) - V(\mathbf{S}^{(1)}) - \cdots - V(\mathbf{S}^{(r)}).$$

We let δ be the sum of the degrees of $\mathbf{T}^{(1)}, \ldots, \mathbf{T}^{(\ell)}$ and $\mathbf{S}^{(1)}, \ldots, \mathbf{S}^{(r)}$ and we make the assumption that the characteristic of $\mathbb{K}$ is equal to 0 or greater than δ^2.

Our strategy is to reduce to univariate representations, perform the set theoretic operations on univariate polynomials, and finally compute the equiprojectable decomposition for the new order.

Step 1. We compute univariate representations $\mathscr{U}_1, \ldots, \mathscr{U}_\ell$ and $\mathscr{V}_1, \ldots, \mathscr{V}_r$ of respectively $V(\mathbf{T}^{(1)}), \ldots, V(\mathbf{T}^{(\ell)})$ and $V(\mathbf{S}^{(1)}), \ldots, V(\mathbf{S}^{(r)})$. By Lemma 4, this can be done in expected time

$$O\big(n^2(\mathsf{C}(\delta_1) + \cdots + \mathsf{C}(\delta_\ell) + \mathsf{C}(\delta'_1) + \cdots + \mathsf{C}(\delta'_r))\big),$$

where δ_i is the degree of $\mathbf{T}^{(i)}$ and δ'_i is the degree of $\mathbf{S}^{(i)}$. Using the super-linearity of C, this is seen to be an expected $O(n^2\mathsf{C}(\delta))$.

Step 2. We compute univariate representations $\mathscr{U}$ of $V(\mathbf{T}^{(1)}) \cup \cdots \cup V(\mathbf{T}^{(\ell)})$ and $\mathscr{V}$ of $V(\mathbf{S}^{(1)}) \cup \cdots \cup V(\mathbf{S}^{(r)})$. The following divide-and-conquer process takes an expected time $O(n\mathsf{C}(\delta)\log(\delta))$ to achieve this task.

We apply repeatedly the union algorithm of Lemma 3 to $\mathscr{U}_1, \ldots, \mathscr{U}_\ell$, respectively $\mathscr{V}_1, \ldots, \mathscr{V}_r$. To compute say $\mathscr{U}$, we let $\ell' = \lceil \ell/2 \rceil$, and we compute recursively univariate representations of

$$V(\mathbf{T}^{(1)}) \cup \cdots \cup V(\mathbf{T}^{(\ell')}) \quad \text{and} \quad V(\mathbf{T}^{(\ell'+1)}) \cup \cdots \cup V(\mathbf{T}^{(\ell)});$$

then, these two univariate representations are merged by means of Lemma 3. The running time analysis is the same as in the proof of Proposition 1: the divide-and-conquer structure of the algorithm induces the loss of a logarithmic factor, as is the case for other algorithms with the same structure [18, Chapter 10].

Step 3. By another application of Lemma 3 to $\mathscr{U}$ and $\mathscr{V}$, this time for computing a set-theoretic difference, we finally obtain a univariate representation $\mathscr{W}$ of V. This takes an expected time $O(n\mathsf{C}(\delta))$.

Step 4. Starting from $\mathscr{W}$, we compute $\mathscr{D}(V, <')$ using the algorithm of Proposition 2. This takes an expected time $O(n\mathsf{C}(\delta)(n + \log(\delta)))$.

The total cost of this algorithm is an expected $O(n\mathsf{C}(\delta)(n + \log(\delta)))$, as claimed in Theorem 1.

4.3 Solving question $\mathbf{P}_2$

Next, we show how to solve question $\mathbf{P}_2$ stated in the introduction. Given a triangular set $\mathbf{T}$ in $\mathbb{K}[X_1, \ldots, X_n]$, for a variable order $<$, as well as F in $R_{\mathbf{T}}$ and a target variable order $<'$, we are to compute the equiprojectable decompositions

$$\mathscr{D}(V(\mathbf{T}) \cap V(F), <') \quad \text{and} \quad \mathscr{D}(V(\mathbf{T}) - V(F), <'),$$

as well as the inverse of F modulo each $\mathbf{T}'$ in $\mathscr{D}(V(\mathbf{T})-V(F),<')$. We let δ be the degrees of $\mathbf{T}$ and we make the assumption that the characteristic of $\mathbb{K}$ is equal to 0 or greater than δ^2.

Our strategy is similar to the one of the previous subsection: we convert to a univariate representation, operate with univariate polynomials, and convert back to triangular representations.

Step 1. We compute a univariate representation $\mathscr{U} = (P, \mathbf{U}, \mu)$ of $V(\mathbf{T})$ and $F^\star = \Psi_{\mathbf{T},\mathscr{U}}(F)$. By Lemma 4, this can be done in expected time $O(n^2\mathsf{C}(\delta))$.

Step 2. We compute $P' = \gcd(P, F^\star)$ and $P'' = P/P'$, as well as the inverse $G^\star$ of $F^\star$ modulo P'' (this inverse exists, since P is squarefree). This takes time $O(\mathsf{M}(\delta)\log(\delta))$, which is $O(\mathsf{C}(\delta))$.

The roots of P' describe the points of $V(\mathbf{T})$ where F vanishes; the roots of P'' describe those where F is nonzero.

Step 3. Writing $\mathbf{U} = (U_1, \ldots, U_n)$, we compute $U_i' = U_i \bmod P'$ and $U_i'' = U_i \bmod P''$ for all i, and we define $\mathscr{U}' = (P', (U_1', \ldots, U_n'), \mu)$ and $\mathscr{U}'' = (P'', (U_1'', \ldots, U_n''), \mu)$. This takes time $O(n\mathsf{M}(\delta))$, which is negligible compared to the cost of Step 1.

Note that $\mathscr{U}'$ is a univariate representation of $V(\mathbf{T}) \cap V(F)$, and that $\mathscr{U}''$ is a univariate representation of $V(\mathbf{T}) - V(F)$.

Step 4. Starting from $\mathscr{U}'$ and $\mathscr{U}''$ we compute the equiprojectable decompositions $\mathscr{D}(V(\mathbf{T})\cap V(F),<')$ and $\mathscr{D}(V(\mathbf{T})-V(F),<')$ using the algorithm of Proposition 2. This takes an expected time $O(n\mathsf{C}(\delta)(n+\log(\delta)))$. Besides, using the second part of Proposition 2, we can compute the image of $G^\star$ in each $R_{\mathbf{T}'}$, for $\mathbf{T}'$ in $\mathscr{D}(V(\mathbf{T})-V(F),<')$. This image is the inverse of F in $R_{\mathbf{T}'}$.

As for question $\mathbf{P}_2$, the total cost of this algorithm is an expected $O(n\mathsf{C}(\delta)(n+\log(\delta)))$, as claimed in Theorem 1.

4.4 Experimental results

This section reports on experimental results obtained with a Maple implementation of the algorithms of Subsection 4.2 and 4.3.

Our implementation supports inputs with coefficients in finite fields of the form $\mathbb{F}_p$, p prime. This is the most natural choice, since over base fields such as $\mathbb{Q}$ or rational function fields, the cost of arithmetic operations in the base field cannot be assumed to be constant. For inputs defined over e.g. $\mathbb{Q}$, the natural approach would be to use modular methods, using for instance lifting techniques (for which the equiprojectable decomposition is particularly well suited, as we pointed out in the introduction).

Over base fields such as $\mathbb{F}_p$, we have two choices for modular composition and power projection: algorithms following Brent and Kung's idea, as described in Section 2.1, or the extension of the Kedlaya-Umans algorithm given in [35]. Unfortunately, even though the

Table 1: Timings for equiprojectable decomposition

n	d	δ	us	Maple
3	2	4	0.03	0.03
3	3	10	0.07	0.12
3	4	20	0.12	0.52
3	5	35	0.22	1.6
3	6	56	0.44	4.2

n	d	δ	us	Maple
4	2	5	0.06	0.05
4	3	15	0.2	0.4
4	4	35	0.3	2.1
4	5	70	0.8	8.4
4	6	126	1.9	40

n	d	δ	us	Maple
5	2	6	0.09	0.08
5	3	21	0.37	0.96
5	4	56	0.81	6.5
5	5	126	2.4	45
5	6	252	9.5	512

n	d	δ	us	Maple
6	2	7	0.15	0.13
6	3	28	0.5	2.1
6	4	84	1.8	19
6	5	210	8.2	300
6	6	462	49	5885

latter is asymptotically better, the large constants hidden in the $O^{\sim}$ notation make it inferior for the range of degrees we consider. Thus, our implementation relies on the Brent-Kung approach.

Other than modular composition and power projection, our algorithms use only univariate and bivariate polynomial arithmetic. As a result, they were implemented using the `modp1` functions [32, Section 4]. `modp1` is a set of Maple functions that provides efficient univariate polynomial arithmetic (multiplication, gcd, ...); these functions are partly written in C and are dedicated to base fields of the form $\mathbb{F}_p$, for p a word-size prime.

The following timings are obtained using Maple 15 on an 2.8 GHz AMD Athlon II X2 240e processor. The base field is $\mathbb{F}_p$, with $p = 962592769$. All timings are in seconds, and all computations were interrupted whenever they used 2Gb of RAM or more.

Our first experiments concern the particular case of question $\mathbf{P}_1$, where the input and the target order are the same, and $r = 0$. In other words, we take as input some triangular sets $\mathbf{T}^{(1)}, \ldots, \mathbf{T}^{(\ell)}$ for an order $<$, and we compute the equiprojectable decomposition of

$$V(\mathbf{T}^{(1)}) \cup \cdots \cup V(\mathbf{T}^{(\ell)}),$$

for the same order. In Table 1, we show comparisons with the function `EquiprojectableDecomposition` of the `RegularChains` library [27], which has similar specifications (we are not aware of other implementations of such an algorithm).

In each sub-table, the number n of variables is fixed; we show timings for the equiprojectable decompositions of sets of points of cardinality δ; the column d gives an upper bound on all d_i that appear as main degrees in the triangular sets in the output. In almost all cases, our implementation does better than the built-in function; the fact that we are relying on the `modp1` functions is certainly a key factor for this.

Table 2: Timings for inversion in $R_{\mathbf{T}}$

n	d	δ	us	`Inverse`	`modpn`
3	2	8	0.04	0.3	0.01
3	3	27	0.06	1.4	0.01
3	4	64	0.14	5.2	0.02
3	5	125	0.24	6.1	0.05
3	6	216	0.75	21	0.06

n	d	δ	us	`Inverse`	`modpn`
4	2	16	0.07	1.1	0.01
4	3	81	0.2	4.8	0.06
4	4	256	1	600	0.1
4	5	625	5.3	10536	0.8
4	6	1296	23	>2 Gb	1.2

n	d	δ	us	`Inverse`	`modpn`
5	2	32	0.14	210	0.03
5	3	243	1	1576	0.42
5	4	1024	1.5	>2 Gb	1.2
5	5	3125	151	>2 Gb	24
5	6	7776	1007	>2 Gb	37

n	d	δ	us	`Inverse`	`modpn`
6	2	64	0.3	>2 Gb	0.1
6	3	729	8.8	>2 Gb	4.6
6	4	4096	273	>2 Gb	18
6	5	15625	5099	>2 Gb	661
6	6	46656	67339	>2 Gb	1135

Our second experiments address inverse computation modulo a triangular set, which is a particular case of question $\mathbf{P}_2$: the input and the target order are the same, and (by construction of our examples), no splitting occurred. In other words, we take as input a triangular set $\mathbf{T}$ and $F \in R_{\mathbf{T}}$, invertible in $R_{\mathbf{T}}$; we output the inverse of F in $R_{\mathbf{T}}$.

In Table 2, we give examples for various situations: n denotes the number of variables and d is such that the input triangular set has multidegree $(d, \ldots, d)$, of length n; thus, its degree δ is d^n.

We show comparisons with the function `Inverse` of the `RegularChains` library. This function may induce splittings; if we wanted the same output as in our implementation, we would also have to perform a recombination after the call to `Inverse` (we did not include this step in the timings). As in the previous example, our code usually does better.

We also include timings obtained by using the C `modpn` library [30], which can be called from a Maple session. Obviously, we expect this compiled library to be much faster than our interpreted code; however, timings are sometimes within a factor of 10 or less, which we see as a sign that our implementation performs well. Note that `modpn` relies on FFT techniques, as a result, only those finite fields $\mathbb{F}_p$ with suitable roots of unity are supported (the field $\mathbb{F}_p$ in our examples is one of them).

5 The converse reduction

This section is mostly independent from the other ones. In the previous sections, we used modular composition and power projection as our basic subroutines, and reduced other questions to these two operations. In this section, we will do the opposite, by reducing modular composition and power projection to equiprojectable decomposition.

As mentioned in the introduction, modular composition and power projection are dual problems. An algorithmic theorem called the *transposition principle* shows that an algorithm for the former can be transformed into an algorithm for the latter, and conversely [11, 7]: this result could in principle allow us to deal only with e.g. modular composition. However, it applies only in a restricted computational model (using *linear programs*), which is not suited to questions such as decompositions of triangular sets (which are inherently non-linear). As a result, we give explicit reductions for both modular composition and power projection.

In the introduction, we defined $\mathsf{E} : \mathbb{N}^2 \to \mathbb{N}$ as a function such that one can solve problem $\mathbf{P}_1$ (computing the equiprojectable decomposition of a family of triangular sets in n variables, with sum of degrees δ) using $\mathsf{E}(n, \delta)$ base field operations.

Recall then the statement of Theorem 2: we take $(m, n) = (1, 1)$ or $(m, n) = (1, 2)$, and we let $\mathbf{T}$ be a triangular set in n variables that generates a radical ideal. Then, we can compute modular compositions and power projections modulo $\langle \mathbf{T} \rangle$ with parameters (m, n) and size $\delta_{\mathbf{f}} \leq \delta_{\mathbf{T}}$ in time $2\mathsf{E}(4, \delta_{\mathbf{T}}) + O\tilde{}(\delta_{\mathbf{T}})$.

The two subsections address respectively modular composition and power projection. In both cases, we can assume that $n = 2$, since any triangular set in one variable (that is, any polynomial $T_1(X_1)$) can be seen as a triangular set in two variables, by adding a dummy polynomial $T_2(X_1, X_2) = X_2$. Note that the proofs would generalize to computations in more than two variables, and would involve terms of the form $\mathsf{E}(n + 2, \delta_{\mathbf{T}})$.

5.1 Modular composition

Following the previous discussion let thus $\mathbf{T} = (T_1, T_2)$ be a triangular set in $\mathbb{K}[X_1, X_2]$, G in $R_{\mathbf{T}}$, and F in $\mathbb{K}[Y]$, of degree $\deg(F) \leq \delta_{\mathbf{T}}$. We show here how to compute $K = F(G) \in R_{\mathbf{T}}$, using change of order as our main subroutine.

Consider the triangular set (for the order $X_1 < X_2 < Y$)

$$\mathbf{T}' \left| \begin{array}{l} Y - G(X_1, X_2) \\ T_2(X_1, X_2) \\ T_1(X_1); \end{array} \right.$$

let $V \subset \overline{\mathbb{K}}^3$ be its zero-set, and let us compute $\mathscr{D}(V, <')$, where $<'$ is the order $Y <' X_1 <' X_2$. We obtain a family of triangular sets $\mathbf{U}^{(1)}, \ldots, \mathbf{U}^{(N)}$ of the form

$$\mathbf{U}^{(i)} \left| \begin{array}{l} U_{i,2}(Y, X_1, X_2) \\ U_{i,1}(Y, X_1) \\ R_i(Y). \end{array} \right.$$

Let now I be the ideal generated by the polynomials (which do not form a triangular set, since the first polynomial is not reduced)

$$\left| \begin{array}{l} Z - F(Y) \\ Y - G(X_1, X_2) \\ T_2(X_1, X_2) \\ T_1(X_1). \end{array} \right.$$

After reduction, we see that I is generated by the triangular set (for the order $X_1 < X_2 < Y < Z$)

$$\mathbf{T}'' \left| \begin{array}{l} Z - K(X_1, X_2) \\ Y - G(X_1, X_2) \\ T_2(X_1, X_2) \\ T_1(X_1), \end{array} \right.$$

where K is the polynomial we want to compute. On the other hand, the construction of the triangular sets $\mathbf{U}^{(i)}$ shows that I is the intersection of the ideals generated by the triangular sets $\mathbf{V}^{(i)}$ (for the order $Y <' X_1 <' X_2 <' Z$) given by

$$\mathbf{V}^{(i)} \left| \begin{array}{l} Z - F_i(Y) \\ U_{i,2}(Y, X_1, X_2) \\ U_{i,1}(Y, X_1) \\ R_i(Y), \end{array} \right.$$

with $F_i = F \bmod R_i$. The algorithm is then the following:

- First, we compute all triangular sets $\mathbf{U}^{(i)}$. Since $\mathbf{T}'$ generates a radical ideal, this can be done in $\mathsf{E}(3, \delta_{\mathbf{T}}) \leq \mathsf{E}(4, \delta_{\mathbf{T}})$ base field operations (obviously, $\mathsf{E}(n, \delta) \leq \mathsf{E}(n', \delta)$ holds for all $n \leq n'$, as can be seen by using $n' - n$ dummy polynomials to obtain a triangular set in n' variables).

- Next, we compute all triangular sets $\mathbf{V}^{(i)}$. This requires us to compute all F_i. Since $\deg(F) \leq \delta_{\mathbf{T}}$, and since the sum of the degrees of the R_i is at most $\delta_{\mathbf{T}}$ as well, all F_i can be computed in time $O(\mathsf{M}(\delta_{\mathbf{T}}) \log(\delta_{\mathbf{T}}))$ using fast multiple reduction [18, Chapter 10].

- Finally, we compute $\mathbf{T}''$, and thus K, by computing the equiprojectable decomposition of $V(\mathbf{V}^{(1)}) \cup \cdots \cup V(\mathbf{V}^{(N)})$, for the order $X_1 < X_2 < Y < Z$. Again, this takes time $\mathsf{E}(4, \delta_{\mathbf{T}})$.

The total time is at most $2\mathsf{E}(4, \delta_{\mathbf{T}}) + O(\mathsf{M}(\delta_{\mathbf{T}}) \log(\delta_{\mathbf{T}}))$, which fits into the claimed bound.

5.2 Power projection

We will now prove the second part of Theorem 2, dealing with power projection. Let thus $\mathbf{T} = (T_1, T_2)$ be a triangular set in $\mathbb{K}[X_1, X_2]$ that generates a radical ideal, let G be in $R_{\mathbf{T}}$, and let $\ell : R_{\mathbf{T}} \to \mathbb{K}$ be a $\mathbb{K}$-linear form. Given an integer $f \leq \delta_{\mathbf{T}}$, we show here how to compute the values $\ell(G^c)$, for $0 \leq c < f$. We start with a folklore lemma involving univariate computations only.

Univariate computations. Let $\mathbb{A}$ be a ring, F a monic polynomial of degree d in $\mathbb{A}[X]$, and R the free $\mathbb{A}$-module $\mathbb{A}[X]/\langle F \rangle$, with the (classes of) $1, X, \ldots, X^{d-1}$ as a basis. In this context, the *trace* $\tau : R \to \mathbb{A}$ is still well-defined, with $\tau(A)$ being the trace of the multiplication map by A in R. For $A \in R$ and ℓ an $\mathbb{A}$-linear form $R \to \mathbb{A}$, the $\mathbb{A}$-linear form $A \cdot \ell$ is defined as before, by $(A \cdot \ell)(B) = \ell(AB)$.

Lemma 7. *Suppose that the derivative $\partial F/\partial X$ of F is invertible in R, with inverse G. Given G, and given an $\mathbb{A}$-linear form $\ell : R \to \mathbb{A}$, we can compute A in R such that $\ell = A \cdot \tau$, using $O(\mathsf{M}(d))$ operations in $\mathbb{A}$.*

Proof. Let us define another useful $\mathbb{A}$-linear form, the *residue* $\rho : R \to \mathbb{A}$, by $\rho(X^i) = 0$ for $i < d-1$ and $\rho(X^{d-1}) = 1$. Given ℓ as above, it is known that there exists B such that $\ell = B \cdot \rho$. Indeed, a straightforward computation shows that the values $(B \cdot \rho)(X^i)$, for $i = 0, \ldots, d-1$, are the coefficients of $\mathrm{rev}(B, d-1)/\mathrm{rev}(F, d) \bmod X^d$, where for any polynomial $P \in \mathbb{A}[X]$ and any $d \geq \deg(P)$, we write $\mathrm{rev}(P, d) = X^d P(1/X)$. This implies that given ℓ, we can find the requested B by means of a power series multiplication modulo X^d, which can be done in $\mathsf{M}(d)$ operations in $\mathbb{A}$.

Furthermore, the *Euler formula* [17, Proposition 2.4] shows that $\tau = \partial F/\partial X \cdot \rho$, so that $\rho = G \cdot \tau$. With ℓ and B as above, this implies that we have $\ell = A \cdot \tau$, with $A = BG \bmod F$. Computing A thus takes another $O(\mathsf{M}(d))$ operations in $\mathbb{A}$, proving the lemma. □

Bivariate computations. We will now apply the results of the former paragraph in a bivariate context. The notation is the one introduced at the beginning of this subsection; furthermore, we let $\mathrm{tr} : R_{\mathbf{T}} \to \mathbb{K}$ be the trace linear form. We also write $d_1 = \deg(T_1, X_1)$ and $d_2 = \deg(T_2, X_2)$, so that $\delta_{\mathbf{T}} = d_1 d_2$.

Lemma 8. *Given a $\mathbb{K}$-linear form $\ell : R_{\mathbf{T}} \to \mathbb{K}$, one can compute an element $A \in R_{\mathbf{T}}$ such that $\ell = A \cdot \mathrm{tr}$ in time $O(\mathsf{M}(d_1)\mathsf{M}(d_2)\log(d_1)\log(d_2))$.*

Proof. Let us define $S_{\mathbf{T}} = \mathbb{K}[X_1]/\langle T_1 \rangle$, so that we have $R_{\mathbf{T}} = S_{\mathbf{T}}[X_2]/\langle T_2 \rangle$. Let further $\tau_1 : S_{\mathbf{T}} \to \mathbb{K}$ and $\tau_2 : R_{\mathbf{T}} \to S_{\mathbf{T}}$ be the trace forms; thus, τ_1 is $\mathbb{K}$-linear, τ_2 is $S_{\mathbf{T}}$-linear, and we have $\mathrm{tr} = \tau_1 \circ \tau_2$.

First, we are going to factor $\ell : R_{\mathbf{T}} \to \mathbb{K}$ as $\ell = \tau_1 \circ L$, where $L : R_{\mathbf{T}} \to S_{\mathbf{T}}$ is a suitable $S_{\mathbf{T}}$-linear form. Computing L amounts to compute $\lambda_{i_2} = L(X_2^{i_2})$, for $i_2 = 0, \ldots, d_2 - 1$; the condition defining L is equivalent to $\ell(X_1^{i_1} X_2^{i_2}) = \tau_1(L(X_1^{i_1} X_2^{i_2}))$, for $i_1 = 0, \ldots, d_1 - 1$ and $i_2 = 0, \ldots, d_2 - 1$. This can be rewritten as $\ell(X_1^{i_1} X_2^{i_2}) = \tau_1(X_1^{i_1} \lambda_{i_2})$, by $S_{\mathbf{T}}$-linearity of L. For a fixed $i_2 < d_2$, let ℓ_{i_2} be the $\mathbb{K}$-linear form $S_{\mathbf{T}} \to \mathbb{K}$ defined by $\ell_{i_2}(A) = \ell(A X_2^{i_2})$. Then, the previous condition says that $\ell_{i_2} = \lambda_{i_2} \cdot \tau_1$.

Computing the linear forms ℓ_{i_2} is free (since their values on the canonical basis of $S_{\mathbf{T}}$ are simply values of ℓ); then, finding λ_{i_2} is done by first inverting T_1' modulo T_1, and applying Lemma 7 for the extension $S_{\mathbf{T}} \to \mathbb{K}$. The total time to computing all λ_{i_2} is thus $O((\log(d_1) + d_2)\mathsf{M}(d_1))$.

Now that we have written $\ell = \tau_1 \circ L$, we will apply Lemma 7 to L, for the extension $R_{\mathbf{T}} \to S_{\mathbf{T}}$. This requires us to invert $\partial T_2/\partial X_2$ in $R_{\mathbf{T}}$; a quasi-linear time algorithm is given in [1], with a cost $O(\mathsf{M}(d_1)\mathsf{M}(d_2)\log(d_1)\log(d_2))$. Once this is done, Lemma 7 gives us an element $A \in R_{\mathbf{T}}$ such that $L = A \cdot \tau_2$ in time $O(\mathsf{M}(d_1)\mathsf{M}(d_2))$.

To summarize, we have written $\ell = \tau_1 \circ L$ and $L = A \cdot \tau_2$, so that $\ell(B) = \tau_1(\tau_2(AB))$ holds for all $B \in R_{\mathbf{T}}$. Since $\tau_1 \circ \tau_2 = \mathrm{tr}$, this implies that $\ell = A \cdot \mathrm{tr}$. □

Transposed multiple reduction. Our next ingredient is an algorithm for the following operation. Consider some pairwise coprime monic polynomials $R_1, \ldots, R_N$ in $\mathbb{K}[X]$, and let $R = R_1 \cdots R_N$.

We have already mentioned the multiple reduction map $\mathbb{K}[X]/\langle R\rangle \to \mathbb{K}[X]/\langle R_1\rangle \times \cdots \times \mathbb{K}[X]/\langle R_N\rangle$; writing $d = \deg(R)$, this operation can be done in time $O(\mathsf{M}(d)\log(d))$. In this paragraph, we will discuss the dual map. On input linear forms $\ell_i : \mathbb{K}[X]/\langle R_i\rangle \to \mathbb{K}$, this dual map computes the linear form $\ell : \mathbb{K}[X]/\langle R\rangle \to \mathbb{K}$ defined by

$$A \mapsto \sum_{i\leq N} \ell_i(A \bmod R_i),$$

where all ℓ_i and ℓ are given by means of their values on the monomials bases of the respective $\mathbb{K}[X]/\langle R_i\rangle$ and $\mathbb{K}[X]/\langle R\rangle$. In other words, it computes the values

$$\sum_{i\leq N} \ell_i(X^j \bmod R_i),$$

for $j = 0, \ldots, d-1$. In [6], an algorithm called $\mathsf{TSimulMod}$ is given that solves this problem in time $O(\mathsf{M}(d)\log(d))$. Computing the above values up to index e, for some $e > d$, can then be done in time $O(\mathsf{M}(e))$, see for instance [7].

Conclusion. Let us return to the proof of Theorem 2. On input $\mathbf{T} = (T_1, T_2)$, $G \in R_{\mathbf{T}}$ and $\ell : R_{\mathbf{T}} \to \mathbb{K}$, we will show how to compute the values $\ell(G^c)$, for $0 \leq c < \delta_{\mathbf{T}}$. Using the algorithm of Lemma 8, we can compute $A \in R_{\mathbf{T}}$ such that the values we want are of the form $\mathrm{tr}(AG^c)$, for $0 \leq c < \delta_{\mathbf{T}}$.

Let us introduce the triangular set (for the order $X_1 < X_2 < Y < Z$)

$$\mathbf{T}' \left| \begin{array}{l} Z - G(X_1, X_2) \\ Y - A(X_1, X_2) \\ T_2(X_1, X_2) \\ T_1(X_1), \end{array} \right.$$

and let its equiprojectable decomposition for the order $Z <' Y <' X_1 <' X_2$ be given by triangular sets

$$\mathbf{U}^{(i)} \left| \begin{array}{ll} U_{i,2}(Z, Y, X_1, X_2) & \\ U_{i,1}(Z, Y, X_1) & \\ S_i(Z, Y) & \qquad 1 \leq i \leq N. \\ R_i(Z), & \end{array} \right.$$

For $i \leq N$, let $\tau_i : R_{\mathbf{U}^{(i)}} \to \mathbb{K}$ be the trace modulo $\mathbf{U}^{(i)}$. Since $R_{\mathbf{T}}$ and $R_{\mathbf{T}'}$ are isomorphic $\mathbb{K}$-algebras, the traces in $R_{\mathbf{T}}$ and $R_{\mathbf{T}'}$ coincide. Since $\langle \mathbf{T}'\rangle$ is the intersection of the pairwise coprime ideals $\langle \mathbf{U}^{(i)}\rangle$, it follows (for instance from Stickelberger's Theorem) that for any index c, we have

$$\mathrm{tr}(A\, G^c) = \sum_{i\leq N} \tau_i(Y Z^c).$$

For $i \leq N$, let ℓ_i be the linear form $\mathbb{K}[Z]/\langle R_i \rangle \to \mathbb{K}$ defined by $\ell_i(B) = \tau_i(YB)$. Then, one sees that $\tau_i(YZ^c) = \ell_i(Z^c)$, so that we have

$$\mathrm{tr}(A\,G^c) = \sum_{i \leq N} \ell_i(Z^c). \tag{8}$$

Using this remark, we can now give the whole algorithm and its running time.

- First, we compute $A \in R_{\mathbf{T}}$ such that $\ell = A \cdot \mathrm{tr}$. By Lemma 8, this can be done in time $O(\mathsf{M}(d_1)\mathsf{M}(d_2)\log(d_1)\log(d_2))$.
- Next, we compute the triangular sets $\mathbf{U}^{(i)}$, $i = 1, \ldots, N$. This takes time $\mathsf{E}(4, \delta_{\mathbf{T}})$.
- The following step consists of computing the linear forms τ_i (by means of their values on the canonical bases of the residue class rings $R_{\mathbf{U}^{(i)}}$). We have seen in Subsection 2.1 that we can compute each of those in time $O(\mathsf{M}(\delta_{\mathbf{U}^{(i)}}))$, so the total time is $O(\mathsf{M}(\delta_{\mathbf{T}}))$ by the super-linearity of M.
- Knowing the linear forms τ_i, we can deduce ℓ_i by first computing all $Y \cdot \tau_i$ (for a total time of $O(\mathsf{M}(\delta_{\mathbf{T}}))$ again), from which the values of ℓ_i on the basis of $\mathbb{K}[Z]/\langle R_i \rangle$ can be read off.
- Finally, we obtain $\mathrm{tr}(A\,G^c)$, for $c = 0, \ldots, \delta_{\mathbf{T}} - 1$, using Eq. (8) and the algorithm for transposed multiple reduction; this takes time $O(\mathsf{M}(\delta_{\mathbf{T}})\log(\delta_{\mathbf{T}}))$.

Taking a quasi-linear M, and summing all previous costs, the claim in Theorem 2 follows.

Acknowledgments

Adrien Poteaux was supported by the EXACTA grant of the National Science Foundation of China (NSFC 60911130369) and the French National Research Agency (ANR-09-BLAN-0371-01). Éric Schost is supported by NSERC and the Canada Research Chair program. We wish to thank Marc Moreno Maza and Yuzhen Xie for interesting discussions during the preparation of this article, and the referees for their helpful suggestions.

References

[1] C. J. Accettella, G. M. Del Corso, and G. Manzini. Inversion of two level circulant matrices over $\mathbb{Z}_p$. *Linear Algebra and its Applications*, 366:5 – 23, 2003.

[2] M. E. Alonso, E. Becker, M.-F. Roy, and T. Wörmann. Zeros, multiplicities and idempotents for zerodimensional systems. In *MEGA 94*, volume 142 of *Progress in Mathematics*, pages 1–15. Birkhäuser, 1996.

[3] P. Aubry, D. Lazard, and M. Moreno Maza. On the theories of triangular sets. *Journal of Symbolic Computation*, 28(1, 2):45–124, 1999.

[4] P. Aubry and A. Valibouze. Using Galois ideals for computing relative resolvents. *Journal of Symbolic Computation*, 30(6):635–651, 2000.

[5] A. Bostan, P. Flajolet, B. Salvy, and É. Schost. Fast computation of special resultants. *Journal of Symbolic Computation*, 41(1):1–29, 2006.

[6] A. Bostan, G. Lecerf, B. Salvy, É. Schost, and B. Wiebelt. Complexity issues in bivariate polynomial factorization. In *ISSAC'04*, pages 42–49. ACM, 2004.

[7] A. Bostan, G. Lecerf, and É. Schost. Tellegen's principle into practice. In *ISSAC'03*, pages 37–44. ACM, 2003.

[8] A. Bostan, M. F. I. Chowdhury, J. van der Hoeven, and É. Schost. Homotopy methods for multiplication modulo triangular sets. *Journal of Symbolic Computation*. To appear.

[9] F. Boulier, F. Lemaire, and M. Moreno Maza. PARDI! In *ISSAC'01*, pages 38–47. ACM, 2001.

[10] R. P. Brent and H. T. Kung. Fast algorithms for manipulating formal power series. *Journal of the ACM*, 25(4):581–595, 1978.

[11] P. Bürgisser, M. Clausen, and M. A. Shokrollahi. *Algebraic Complexity Theory*. Springer, 1997.

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

[13] D. Coppersmith and S. Winograd. Matrix multiplication via arithmetic progressions. *Journal of Symbolic Computation*, 9(3):251–280, 1990.

[14] D. Cox, J. Little, and D. O'Shea. *Using Algebraic Geometry*. Springer, New York, 1998.

[15] X. Dahan, M. Moreno Maza, É. Schost, W. Wu, and Y. Xie. Lifting techniques for triangular decompositions. In *ISSAC'05*, pages 108–115. ACM Press, 2005.

[16] X. Dahan, M. Moreno Maza, É. Schost, and Y. Xie. On the complexity of the D5 principle. In *Transgressive Computing*, pages 149–168, 2006.

[17] M. Demazure. Charles Hermite déjà .... Note informelle du calcul formel, École polytechnique, 1987.

[18] J. von zur Gathen and J. Gerhard. *Modern Computer Algebra*. Cambridge University Press, 2003.

[19] J. von zur Gathen and V. Shoup. Computing Frobenius maps and factoring polynomials. *Computational Complexity*, 2(3):187–224, 1992.

[20] P. Gianni and T. Mora. Algebraic solution of systems of polynomial equations using Gröbner bases. In *AAECC-5*, volume 356 of *Lecture Notes in Computer Science*, pages 247–257. Springer, 1989.

[21] M. Giusti, J. Heintz, J.-E. Morais, and L.-M. Pardo. When polynomial equation systems can be solved fast? In *AAECC-11*, volume 948 of *LNCS*, pages 205–231. Springer, 1995.

[22] M. Giusti, G. Lecerf, and B. Salvy. A Gröbner-free alternative for polynomial system solving. *Journal of Complexity*, 17(1):154–211, 2001.

[23] É. Hubert. Notes on triangular sets and triangulation-decomposition algorithms. I. Polynomial systems. In *Symbolic and numerical scientific computation*, volume 2630 of *LNCS*, pages 1–39. Springer, 2003.

[24] M. Kalkbrener. A generalized Euclidean algorithm for computing triangular representations of algebraic varieties. *Journal of Symbolic Computation*, 15(2):143–167, 1993.

[25] E. Kaltofen. Challenges of symbolic computation: my favorite open problems. *Journal of Symbolic Computation*, 29(6):891–919, 2000.

[26] K. Kedlaya and C. Umans. Fast polynomial factorization and modular composition. *SIAM Journal on Computing*, 40(6):1767–1802, 2011.

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

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

[29] X. Li, M. Moreno Maza, and W. Pan. Computations modulo regular chains, 2010. Extended version of [28].

[30] X. Li, M. Moreno Maza, R. Rasheed, and É. Schost. The `Modpn` library: Bringing fast polynomial arithmetic into Maple. *Journal of Symbolic Computation*, 46(7):841–858, 2011.

[31] X. Li, M. Moreno Maza, and É. Schost. Fast arithmetic for triangular sets: from theory to practice. *Journal of Symbolic Computation*, 44(7):891–907, 2009.

[32] M. Monagan. In-place arithmetic for polynomials over $\mathbb{Z}_n$. In *DISCO'92*, volume 721 of *Lecture Notes in Computer Science*, pages 22–34. Springer, 1992.

[33] M. Moreno Maza. On triangular decompositions of algebraic varieties. Technical Report TR 4/99, NAG Ltd, Oxford, UK, 1999. `http://www.csd.uwo.ca/~moreno/`.

[34] C. Pascal and É. Schost. Change of order for bivariate triangular sets. In *ISSAC'06*, pages 277–284. ACM, 2006.

[35] A. Poteaux and É. Schost. Modular composition modulo triangular sets and applications. Submitted, 2010.

[36] F. Rouillier. Solving zero-dimensional systems through the Rational Univariate Representation. *Applicable Algebra in Engineering, Communication and Computing*, 9(5):433–461, 1999.

[37] A. Schönhage. The fundamental theorem of algebra in terms of computational complexity. Technical report, Univ. Tübingen, 1982. 73 pages.

[38] É. Schost. Complexity results for triangular sets. *Journal of Symbolic Computation*, 36(3–4):555–594, 2003.

[39] É. Schost. Computing parametric geometric resolutions. *Applicable Algebra in Engineering, Communication and Computing*, 13(5):349–393, 2003.

[40] V. Shoup. Fast construction of irreducible polynomials over finite fields. *Journal of Symbolic Computation*, 17(5):371–391, 1994.

[41] V. Shoup. Efficient computation of minimal polynomials in algebraic extensions of finite fields. In *ISSAC'99*, pages 53–58. ACM, 1999.