Homotopy techniques for multiplication modulo triangular sets

Alin Bostan, Muhammad F. I. Chowdhury, Joris Van Der Hoeven, Éric Schost

Homotopy techniques for multiplication modulo triangular sets

Alin Bostan

Algorithms Project, INRIA Rocquencourt, 78153 Le Chesnay Cedex, France

Muhammad Chowdhury

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

Joris van der Hoeven

CNRS, Département de Mathématiques, Université Paris-Sud, 91405 Orsay Cedex, France

Éric Schost

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

Abstract

We study the cost of multiplication modulo triangular families of polynomials. Following previous work by Li, Moreno Maza and Schost, we propose an algorithm that relies on homotopy and fast evaluation-interpolation techniques. We obtain a quasi-linear time complexity for substantial families of examples, for which no such result was known before. Applications are given to notably addition of algebraic numbers in small characteristic.

Key words: Triangular sets, multiplication, complexity

* This research was supported by ANR Gecko, the joint Inria-Microsoft Research Lab, MITACS, NSERC and the Canada Research Chair program.

Email addresses: alin.bostan@inria.fr (Alin Bostan), mchowdh3@csd.uwo.ca (Muhammad Chowdhury), Joris.Vanderhoeven@math.u-psud.fr (Joris van der Hoeven), eschost@uwo.ca (Éric Schost).

1. Introduction

Triangular families of polynomials are a versatile data structure, well adapted to encode geometric problems with some form of symmetry [3, 19, 13, 16]. However, in spite of this, many complexity questions are still not answered in a satisfying manner.

A high-level question is to provide sharp estimates on the cost of solving polynomial systems by means of triangular representations. This problem has a geometric nature; it itself relies on several difficult lower-level questions, such as the cost of basic operations with triangular sets. In this paper, we address one such question: the arithmetic cost of multiplication of polynomials modulo a triangular set. This justifiably stands as a central question, since many higher-level routines are built on top of it, such as inversion [20, 23, 11, 22], lifting techniques [10] for modular algorithms, solving systems of equations [21], etc.

Problem statement, overview of our results. We work in $\mathsf{R}[X_1, \ldots, X_n]$, where R is a ring, and we are given a set of relations of the form

$$\mathbf{T} \left| \begin{array}{l} T_n(X_1, \ldots, X_n) \\ \quad \vdots \\ T_2(X_1, X_2) \\ T_1(X_1). \end{array} \right.$$

The polynomials $\mathbf{T}$ form a *triangular set*: for all i, T_i is in $\mathsf{R}[X_1, \ldots, X_i]$, 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 $j < i$. As an aside, note that the case where T_i is not monic but with a leading coefficient invertible modulo $\langle T_1, \ldots, T_{i-1} \rangle$ reduces in principle to the monic case; however, inversion modulo $\langle T_1, \ldots, T_{i-1} \rangle$ remains a difficult question [11], of cost higher than that of multiplication.

As input, we consider two polynomials A, B reduced modulo $\langle \mathbf{T} \rangle$. The direct approach to multiply them modulo $\langle \mathbf{T} \rangle$ is to perform a polynomial multiplication, followed by the reduction modulo $\langle \mathbf{T} \rangle$, by a generalization of Euclidean division. As far as complexity is concerned, when the number of variables grows, this kind of approach cannot give linear time algorithms. Consider for instance the case where all T_i have degree 2 in their main variables X_i. Then, A and B both have 2^n monomials, but their product before reduction has 3^n monomials; after reduction, the number of monomials is 2^n again. If we let $\delta = 2^n$ be a measure of the input and output size, the cost of such an algorithm is at least $3^n = \delta^{\log_2(3)}$.

In this paper, we show that a different approach can lead to a quasi-linear time algorithm, in cases where the monomial support of $\mathbf{T}$ is sparse, or when the polynomials in $\mathbf{T}$ have a low total degree. This will for example be the case for systems of the form

$$\left| \begin{array}{l} X_n^2 - 2X_{n-1} \\ \quad \vdots \\ X_2^2 - 2X_1 \\ X_1^2 \end{array} \right. \quad \text{or} \quad \left| \begin{array}{l} X_n^2 - X_{n-1} \\ \quad \vdots \\ X_2^2 - X_1 \\ X_1^2, \end{array} \right. \tag{1}$$

whose applications are described later on. Our result also applies to the following construction: start from $F \in \mathsf{R}[X]$, say $F = X^3 - X^2 + X - 3$, and define the so-called "Cauchy modules" [27] F_1, F_2, F_3, which are used in effective Galois theory [27, 1, 26]:

$$\left|\begin{array}{lll} F_1(X_1) & = F(X_1) & = X_1^3 - X_1^2 + X_1 - 3 \\ F_2(X_1, X_2) & = \frac{F_1(X_1)-F_1(X_2)}{X_1-X_2} & = X_2^2 + X_2X_1 - X_2 + X_1^2 - X_1 + 1 \\ F_3(X_1, X_2, X_3) & = \frac{F_2(X_1,X_2)-F_2(X_1,X_3)}{X_2-X_3} & = X_3 + X_2 + X_1 - 1. \end{array}\right. \quad (2)$$

For examples (1) and (2), our algorithms give the following results:

- for $\mathbf{T}$ as in (1), multiplication modulo $\langle \mathbf{T} \rangle$ can be performed in quasi-linear time $O^{\tilde{}}(\delta)$, where $\delta = 2^n$ is the input and output size, and where $O^{\tilde{}}(\delta)$ stands for $O(\delta(\log(\delta))^{O(1)})$.
- for $\mathbf{T}$ as in (2), with $n = \deg(F)$, multiplication modulo $\langle \mathbf{T} \rangle$ can be performed in quasi-linear time $O^{\tilde{}}(\delta)$, where $\delta = n!$ is the input and output size.

No previous algorithm was known featuring such complexity estimates.

Our approach. To obtain this quasi-linear cost, we have to avoid multiplying A and B as polynomials. Our solution is to use evaluation and interpolation techniques, just as FFT multiplication of univariate polynomials is multiplication modulo $X^n - 1$.

Fast evaluation and interpolation may not be possible directly, if $\mathbf{T}$ does not have roots in R (as in the previous examples). However, they become possible using deformation techniques: we construct a new triangular set $\mathbf{U}$ with all roots in R, and multiply A and B modulo $\mathbf{S} = \eta\mathbf{T} + (1-\eta)\mathbf{U}$, where η is a new variable. The triangular set $\mathbf{S}$ has roots in $\mathsf{R}[[\eta]]$, by Hensel's lemma, so one can use evaluation-interpolation techniques over $\mathsf{R}[[\eta]]$.

This idea was introduced in [22], but was limited to the case where all polynomials in $\mathbf{T}$ are univariate: T_i was restricted to depend on X_i only, so this did not apply to the examples above. Here, we extend this idea to cover such examples; our main technical contribution is a study of precision-related issues involved in the power series computations, and how they relate to the monomial support of $\mathbf{T}$.

Previous work. It is only recently that fast algorithms for triangular representations have been throughly investigated; thus, previous results on efficient multiplication algorithms are scarce. All natural approaches introduce in their cost estimate an overhead of the form k^n, for some constant k.

The main challenge (still open) is to get rid of this exponential factor unconditionally: we want algorithms of cost $O^{\tilde{}}(\delta)$, where $\delta = d_1 \cdots d_n$ is the number of monomials in A, B and their product modulo $\langle \mathbf{T} \rangle$. For instance, with $T_i = X_i^{d_i}$, the first complexity result of the form $O(\delta^{1+\varepsilon})$, for any $\varepsilon > 0$, was in [29].

The previous work [22] gives a general algorithm of cost $O^{\tilde{}}(4^n\delta)$. That algorithm uses fast Euclidean division; for polynomials of low degree (e.g., for $d_i = 2$), the naive approach for Euclidean division can actually give better results. Previous mentions of such complexity estimates (with a constant higher than 4) are in [20].

As said above, in [22], one also finds the precursor of the algorithm presented here; the algorithm of [22] applies to families of polynomials having $T_i \in \mathsf{R}[X_i]$, and achieves the cost $O(\delta^{1+\varepsilon})$ for any $\varepsilon > 0$. In that case, the analysis of the precision in power series computation was immediate. Our main contribution here is to perform this study in the general case, and to show that we can still achieve similar costs for much larger families of examples.

Basic notation. Let $\mathbf{d} = (d_1, \ldots, d_n)$ be a vector of positive integers. In what follows, these will represent the main degrees of the polynomials in our triangular sets; without loss of generality, we will thus always suppose $d_i \geq 2$ for all i.

Recall that R is our base ring and that $X_1, \ldots, X_n$ are indeterminates over R. We let $M_{\mathbf{d}}$ be the set of monomials

$$M_{\mathbf{d}} = \left\{ X_1^{e_1} \cdots X_n^{e_n} \mid 0 \leq e_i < d_i \text{ for all } i \right\}.$$

We denote by $\mathsf{Span}(M_{\mathbf{d}})$ the free R-submodule generated by $M_{\mathbf{d}}$ in $\mathsf{R}[X_1, \ldots, X_n]$:

$$\mathsf{Span}(M_{\mathbf{d}}) = \left\{ A = \sum_{m \in M_{\mathbf{d}}} a_m m \mid a_m \in \mathsf{R} \text{ for all } m \in M_{\mathbf{d}} \right\}.$$

This is thus the set of polynomials A in $\mathsf{R}[X_1, \ldots, X_n]$, such that $\deg(A_i, X_i) < d_i$ holds for all i. Finally, we let $\delta_{\mathbf{d}}$ be the product $\delta_{\mathbf{d}} = d_1 \cdots d_n$; this is the cardinality of $M_{\mathbf{d}}$. Remark that since all d_i are at least 2, we have the bounds

$$2^n \leq \delta_{\mathbf{d}} \quad \text{and} \quad \sum_{i \leq n} d_1 \cdots d_i \leq 2\delta_{\mathbf{d}}.$$

The former plainly follows from the inequality $2 \leq d_i$; the latter comes from observing that $d_1 \cdots d_i 2^{n-i} \leq d_1 \cdots d_n = \delta_{\mathbf{d}}$; this yields $d_1 \cdots d_i \leq \delta_{\mathbf{d}}/2^{n-i}$, from which the claim follows by summation.

The *multi-degree* of a triangular set $\mathbf{T} = (T_1, \ldots, T_n)$ is the n-tuple $\mathbf{d} = (d_1, \ldots, d_n)$, with $d_i = \deg(T_i, X_i)_{1 \leq i \leq n}$. In this case, $\mathsf{R}[X_1, \ldots, X_n]/\langle \mathbf{T} \rangle$ is a free R-module isomorphic to $\mathsf{Span}(M_{\mathbf{d}})$. We say that a polynomial $A \in \mathsf{R}[X_1, \ldots, X_n]$ is *reduced* with respect to $\mathbf{T}$ if $\deg(A, X_i) < d_i$ holds for all i. For any $A \in \mathsf{R}[X_1, \ldots, X_n]$, there exists a unique $A' \in \mathsf{R}[X_1, \ldots, X_n]$, reduced with respect to $\mathbf{T}$, and such that $A - A'$ is in the ideal $\langle \mathbf{T} \rangle$. We call it the *normal form* of A and write $A' = A \bmod \langle \mathbf{T} \rangle$.

Outlook of the paper. In Section 2, we introduce some basic complexity notation. The next section presents basic evaluation-interpolation algorithms for so-called *equiprojectable* sets, which are extensions of algorithms known for univariate polynomials. We deduce our multiplication algorithm in Section 4; examples, applications and experimental results are in Sections 5 and 6.

2. Preliminaries

Big-O notation is delicate to use in our situation, since our estimates may depend on several (possibly an unbounded number of) parameters (typically, the multi-degree of our triangular sets). Hence, whenever we use a big-O inequality such as $f \in O(g)$, it is implied that there exists a universal constant λ such that $f(v_1, \ldots, v_s) \leq \lambda g(v_1, \ldots, v_s)$ holds for *all* possible values of the arguments. When needed, we use explicit inequalities. Finally, the notation $f \in O\tilde{}(g)$ means that there exists a constant α such that $f \in O(g \log(g)^\alpha)$, where the big-O is to be understood as above.

Our complexity estimates count additions, multiplications, and inversions, when they are possible. We denote by $\mathsf{M} : \mathbb{N} \to \mathbb{N}$ a function such that over any ring, polynomials of degree less than d can be multiplied in $\mathsf{M}(d)$ operations, and which satisfies the superlinearity conditions of [15, Chapter 8]. Using the algorithm of Cantor-Kaltofen [9], one can take $\mathsf{M}(d) \in O(d \lg(d) \lg\lg(d))$, with $\lg(d) = \log_2(\max(d, 2))$.

We next let $\mathsf{C}_0 : \mathbb{N} \to \mathbb{N}$ be a function such that $d \mapsto \mathsf{C}_0(d)/d$ is non-decreasing, $\mathsf{C}_0(d) \geq d$ holds for all d and such that we have, over any ring R:

(1) for any $x_1, \ldots, x_d$ in R and any polynomial $A \in \mathsf{R}[X]$ of degree less than d, one can compute all values $A(x_i)$ in $\mathsf{C}_0(d)$ additions and multiplications in R;

(2) for any $x_1, \ldots, x_d$ in R, one can compute the coefficients of the polynomial $(X - x_1) \cdots (X - x_d)$ in $\mathsf{C}_0(d)$ additions and multiplications in R;

(3) for any $x_1, \ldots, x_d$ in R, with $x_i - x_j$ a unit for $i \neq j$, and any values $v_1, \ldots, v_d$ in R, one can compute the unique polynomial $A \in \mathsf{R}[X]$ of degree less than d such that $A(x_i) = v_i$ holds for all i in $\mathsf{C}_0(d)$ operations in R.

By the results of [15, Chapter 10], one can take $\mathsf{C}_0(d) \in O(\mathsf{M}(d)\log(d))$. We continue with the well-known fact that the function C_0 also enables us to estimate the cost of lifting power series roots of a bivariate polynomial by Newton iteration. In the following lemma, η is a new variable over R.

Lemma 1. *For any polynomial T in $\mathsf{R}[\eta, X]$, monic in X and with $\deg(T, X) = d$, if the roots $a_1, \ldots, a_d$ of $T(0, X)$ are known and mutually distinct, one can compute the roots of $T(\eta, X)$ in $\mathsf{R}[\eta]/\langle\eta^\ell\rangle$ in $O(\mathsf{C}_0(d)\mathsf{M}(\ell))$ operations in R.*

Proof. The algorithm consists in lifting all roots of T in parallel using Newton iteration, using fast evaluation to compute the needed values of T and $\partial T/\partial X$; it is given in Figure 1, where we use a subroutine called EvalUnivariate to do the evaluation. Each pass through the loop at line 2 takes two evaluations in degree d and d inversions, with coefficients that are power series of precision ℓ'. Using [15, Chapter 9], the cost is thus $2\mathsf{C}_0(d)\mathsf{M}(\ell') + \lambda d\mathsf{M}(\ell')$, for some constant λ. Using the super-linearity of the function M, the conclusion follows. □

Remark. When performing all multiplications in $\mathsf{R}[\eta, X]/\langle\eta^\ell\rangle$ using Kronecker's method [15, Chapter 8.4], a more precise cost analysis yields the bound $O(\mathsf{M}(d\ell)\log(d))$ instead of

$$O(\mathsf{C}_0(d)\mathsf{M}(\ell)) = O(\mathsf{M}(d)\mathsf{M}(\ell)\log(d)).$$

If $\ell = O(d)$, then we usually have $\mathsf{M}(d\ell) = O(\mathsf{M}(d)\ell)$, which makes the new bound slightly better. However, this improvement only has a minor impact on what follows, so it will be more convenient to use the technically simpler bound from the lemma.

```
LiftRoots(T, a_1, ..., a_d, ℓ)
1     ℓ' ← 2
2     while ℓ' < ℓ do
2.1       v_1, ..., v_d ← EvalUnivariate(T, a_1, ..., a_d) mod η^ℓ'
2.2       w_1, ..., w_d ← EvalUnivariate(∂T/∂X, a_1, ..., a_d) mod η^ℓ'
2.3       for i = 1, ..., d do
2.3.1        a_i ← a_i − v_i/w_i mod η^ℓ'
2.4       ℓ' ← 2ℓ'
3     return [a_i mod X^ℓ | 1 ≤ i ≤ d]
```

Fig. 1. Lifting all roots of a bivariate polynomial.

To obtain simpler estimates, we let $\mathsf{C}(d) = \Lambda\mathsf{C}_0(d)$, where $\Lambda \geq 1$ is the constant implied in the big-O estimate in the former lemma. Hence, problems (1), (2) and (3) above can

be dealt with in $\mathsf{C}(d)$ operations, and the lifting problem of the previous lemma can be solved in $\mathsf{C}(d)\mathsf{M}(\ell)$ operations. Finally, we introduce another short-hand notation: for a multi-degree $\mathbf{d} = (d_1, \ldots, d_n)$, we write

$$\mathsf{L}(\mathbf{d}) = \sum_{i \leq n} \frac{\mathsf{C}(d_i)}{d_i} \leq n \frac{\mathsf{C}(d)}{d}, \tag{3}$$

with $d = \max_{i \leq n} d_i$. In view of the estimates $\mathsf{C}(d) \in O(\mathsf{M}(d)\log(d))$ and $\mathsf{M}(d) \in O(d\lg(d)\lg\lg(d))$, we also have the upper bound $\mathsf{L}(\mathbf{d}) \in O(\lg(\delta_{\mathbf{d}})^3)$, which shows that $\mathsf{L}(\mathbf{d})$ is of polylogarithmic growth in $\delta_{\mathbf{d}}$.

3. Evaluation and interpolation at equiprojectable sets

In this section, we recall from [3] the definition of *equiprojectable sets*. We prove that one can perform evaluation and interpolation at, and construct the vanishing ideal of, equiprojectable sets in linear time, up to logarithmic factors. We deduce an algorithm for multiplication modulo the vanishing ideal of such sets with a similar complexity. These results extend those given in [24, 22], which dealt with the case of points on a regular grid. The extension to our more general context is rather straightforward, but to our knowledge, it has not appeared in print before.

In all this section, R is a ring; we study subsets of R^n and their successive projections on the subspaces R^i, for $i \leq n$. For definiteness, we let R^0 be a one-point set. Then, for $1 \leq j \leq i \leq n$, we let $\pi_{i,j}$ be the projection

$$\begin{aligned} \pi_{i,j} : \quad \mathsf{R}^i \quad &\to \quad \mathsf{R}^j \\ (x_1, \ldots, x_i) &\mapsto (x_1, \ldots, x_j); \end{aligned}$$

if $j = 0$, we adapt this definition by letting $\pi_{i,0}$ be the constant map $\mathsf{R}^i \to \mathsf{R}^0$. Finally, since this is the projection we use most, we simply write $\pi = \pi_{n,n-1}$ for the projection $\mathsf{R}^n \to \mathsf{R}^{n-1}$.

If V is a subset of R^n, for β in $\pi(V)$, we let V_β be the fiber $V \cap \pi^{-1}(\beta)$. Hence, if β has coordinates $(\beta_1, \ldots, \beta_{n-1})$, the points in V_β have the form $(\beta_1, \ldots, \beta_{n-1}, a)$, for some values a in R. In all that follows, a finite set is by convention non-empty.

3.1. Equiprojectable sets

Let V be a finite set in R^n. *Equiprojectability* is a property of V that describes a combinatorial regularity in the successive projections of V. For $n = 0$, we say that the unique non-empty subset of R^0 is equiprojectable. Then, for $n > 0$, $V \subset \mathsf{R}^n$ is equiprojectable if the following holds:

- the projection $\pi(V)$ is equiprojectable in R^{n-1}, and
- there exists an integer d_n such that for all β in $\pi(V)$, the fiber V_β has cardinality d_n.

The vector $\mathbf{d} = (d_1, \ldots, d_n)$ is called the *multi-degree* of V. Remark that for $n = 1$, any finite $V \subset \mathsf{R}$ is equiprojectable. One easily sees that if V is equiprojectable, its cardinality equals $\delta_{\mathbf{d}} = d_1 \cdots d_n$; more generally, $\pi_{n,i}(V) \subset \mathsf{R}^i$ is equiprojectable of cardinality $d_1 \cdots d_i$. When R is a perfect field, it is proved in [3] that equiprojectable sets

are exactly the zero-sets of triangular sets that generate radical ideals in $\mathsf{R}[X_1, \ldots, X_n]$; we will discuss this in more detail in Subsection 3.4.

We give first a slightly more precise notation for the fibers V_β: if V is equiprojectable, then for all $\beta = (\beta_1, \ldots, \beta_{n-1}) \in \pi(V)$, there exist exactly d_n pairwise distinct values $v_\beta = [a_{\beta,1}, \ldots, a_{\beta,d_n}]$ in R such that

$$V_\beta = [(\beta_1, \ldots, \beta_{n-1}, a_{\beta,i}) \mid a_{\beta,i} \in v_\beta]$$

and thus

$$V \;=\; [(\beta_1, \ldots, \beta_{n-1}, a_{\beta,i}) \mid \beta = (\beta_1, \ldots, \beta_{n-1}) \in \pi(V),\ a_{\beta,i} \in v_\beta,\ 1 \le i \le d_n].$$

For instance, n-dimensional grids are special cases of equiprojectable sets, where v_β is independent of β. Remark also that for some special choices of v_β, improvements in the algorithms below are possible (e.g., if v_β is a translation of a set of points in geometric progression, using the algorithm of [2]).

3.2. Evaluation

Let V be an equiprojectable set of multi-degree $\mathbf{d} = (d_1, \ldots, d_n)$, and let $M_\mathbf{d}$, $\delta_\mathbf{d}$ be as in Section 1. We denote by Eval_V the evaluation map

$$\begin{aligned} \mathsf{Eval}_V : \mathsf{Span}(M_\mathbf{d}) &\to \quad \mathsf{R}^{\delta_\mathbf{d}} \\ F \quad &\mapsto [F(\alpha) \mid \alpha \in V]. \end{aligned}$$

We let $\mathsf{C}_{\mathsf{Eval}}$ be a function such that for any V equiprojectable of multi-degree $\mathbf{d}$, the map Eval_V can be evaluated in $\mathsf{C}_{\mathsf{Eval}}(\mathbf{d})$ operations. In one variable, with $n = 1$ and $\mathbf{d} = (d_1)$, $\mathsf{C}_{\mathsf{Eval}}(\mathbf{d})$ simply describes the cost of evaluating a polynomial of degree less than d_1 at d_1 points of R, so we can take $\mathsf{C}_{\mathsf{Eval}}(\mathbf{d}) = \mathsf{C}(d_1)$. More generally, we have the following quasi-linear time estimate.

Proposition 1. *One can take* $\mathsf{C}_{\mathsf{Eval}}(\mathbf{d}) \le \delta_\mathbf{d} \mathsf{L}(\mathbf{d})$.

Proof. We will use a straightforward recursion over the variables $X_n, \ldots, X_1$. Let $W = \pi(V)$, let $\mathbf{e} = (d_1, \ldots, d_{n-1})$ be the multi-degree of W and let $A(X_1, \ldots, X_n) \in \mathsf{Span}(M_\mathbf{d})$ be the polynomial to evaluate. We write

$$A = \sum_{i<d_n} A_i(X_1, \ldots, X_{n-1}) X_n^i,$$

with A_i in $\mathsf{Span}(M_\mathbf{e})$, and, for β in R^{n-1}, we define

$$A_\beta = \sum_{i<d_n} A_i(\beta) X_n^i \;\in\; \mathsf{R}[X_n].$$

Hence, for $\beta = (\beta_1, \ldots, \beta_{n-1})$ in R^{n-1} and x in R, $A(\beta_1, \ldots, \beta_{n-1}, x) = A_\beta(x)$. As a consequence, to evaluate A at V, we start by evaluating all A_i at all points $\beta \in W$. This gives all polynomials A_β, which we evaluate at the fibers v_β. The algorithm is given in Figure 2. From this, we deduce that we can take $\mathsf{C}_{\mathsf{Eval}}$ satisfying the recurrence

$$\mathsf{C}_{\mathsf{Eval}}(d_1, \ldots, d_n) \le \mathsf{C}_{\mathsf{Eval}}(d_1, \ldots, d_{n-1})\, d_n + d_1 \cdots d_{n-1} \mathsf{C}(d_n).$$

$\underline{\mathsf{Eval}(A, V)}$

1 if $n = 0$ return $[A]$
2 $W \leftarrow \pi(V)$
3 for $i = 0, \ldots, d_n - 1$ do
3.1 $\quad A_i \leftarrow \mathsf{coeff}(A, X_n, i)$
3.2 $\quad \mathrm{val}[i] \leftarrow \mathsf{Eval}(A_i, W)$
$\quad /*$ val$[i]$ has the form $[A_i(\beta) \mid \beta \in W]$ $*/$
4 for β in W do
4.1 $\quad A_\beta \leftarrow \sum_{i<d_n} A_i(\beta) X_n^i$
5 return $[\mathsf{EvalUnivariate}(A_\beta, v_\beta) \mid \beta \in W]$

Fig. 2. Evaluation algorithm.

This implies

$$\mathsf{C}_{\mathsf{Eval}}(d_1, \ldots, d_n) \;\leq\; \sum_{i \leq n} \delta_{\mathbf{d}} \frac{\mathsf{C}(d_i)}{d_i},$$

which proves the proposition. □

3.3. Interpolation

Using the same notation as above, the inverse of the evaluation map is interpolation at V:

$$\begin{aligned} \mathsf{Interp}_V : \quad & \mathsf{R}^{\delta_{\mathbf{d}}} \quad \to \mathsf{Span}(M_{\mathbf{d}}) \\ & [F(\alpha) \mid \alpha \in V] \mapsto \quad F. \end{aligned}$$

For this map to be well-defined, we impose a natural condition on the points of V. Let $W = \pi(V) \in \mathsf{R}^{n-1}$. We say that V *supports interpolation* if

- if $n > 1$, W supports interpolation, and
- for all β in W and all x, x' in v_β, $x - x'$ is a unit;

if the base ring is a field, this condition is vacuous. We will see in the following proposition that if V supports interpolation, then the map Interp_V is well-defined. Moreover, we let $\mathsf{C}_{\mathsf{Interp}}$ be such that, for V equiprojectable of multi-degree $\mathbf{d}$, if V supports interpolation, then the map Interp_V can be evaluated in $\mathsf{C}_{\mathsf{Interp}}(\mathbf{d})$ operations (including inversions).

Proposition 2. *If V supports interpolation, the map Interp_V is well-defined. Besides, one can take $\mathsf{C}_{\mathsf{Interp}}(\mathbf{d}) \leq \delta_{\mathbf{d}} \mathsf{L}(\mathbf{d})$.*

Proof. If $n = 0$, we do nothing; otherwise, we let $W = \pi(V)$. The set of values to interpolate at V has the shape $[f_\alpha \mid \alpha \in V] \in \mathsf{R}^{\delta_{\mathbf{d}}}$; we can thus rewrite it as $[f_\beta \mid \beta \in W]$, where each f_β is in R^{d_n}.

Since V supports interpolation, for β in W, there exists a unique polynomial $A_\beta \in \mathsf{R}[X_n]$ of degree less than d_n, such that $\mathsf{Eval}(A_\beta, v_\beta) = f_\beta$. Applying the algorithm recursively on the coefficients of the polynomials A_β, we can find a polynomial A such that $A(\beta, X_n) = A_\beta(X_n)$ holds for all $\beta \in W$. Then, the polynomial A satisfies our constraints. This provides a right-inverse, and thus a two-sided inverse for the map Eval. The

```
Interp(f, V)
1    if n = 0 return [f]
2    W ← π(V)
3    for β in W do
3.1      A_β ← InterpUnivariate(f_β, v_β)
4    for i = 0, ..., d_n − 1 do
4.1      c_i ← [coeff(A_β, X_n, i) | β ∈ W]
4.2      A_i ← Interp(c_i, W)
5    return Σ_{i<d_n} A_i X_n^i
```

Fig. 3. Interpolation algorithm.

algorithm is given in Figure 3; we use a subroutine called $\mathsf{InterpUnivariate}$ for univariate interpolation. As for evaluation, we deduce that we can take $\mathsf{C}_{\mathsf{Interp}}$ satisfying

$$\mathsf{C}_{\mathsf{Interp}}(d_1,\dots,d_n) \le \mathsf{C}_{\mathsf{Interp}}(d_1,\dots,d_{n-1})\,d_n + d_1\cdots d_{n-1}\mathsf{C}(d_n),$$

which gives our claim, as in the case of evaluation. □

3.4. Associated triangular set

Next, we associate to an equiprojectable set $V \subset \mathsf{R}^n$ of multi-degree $\mathbf{d} = (d_1,\dots,d_n)$ a triangular set $\mathbf{T} = (T_1,\dots,T_n)$ of the same multi-degree, which vanishes on V. As soon as V supports interpolation, the existence of $\mathbf{T}$ is guaranteed (and is established in the proof of the next proposition). Uniqueness holds as well: if $(T_1,\dots,T_n)$ and $(T'_1,\dots,T'_n)$ both vanish on V and have multi-degree $\mathbf{d}$, then for all i, $T_i - T'_i$ vanishes at V as well and is in $\mathsf{Span}(M_{\mathbf{d}})$; hence, it is zero. We call $\mathbf{T}$ the *associated triangular set*; if R is a field, $\mathbf{T}$ is a lexicographic Gröbner basis of the vanishing ideal of V.

Proposition 3. *Given an equiprojectable set V of multi-degree $\mathbf{d}$ that supports interpolation, one can construct the associated triangular set $\mathbf{T}$ in time $O(\delta_{\mathbf{d}}\mathsf{L}(\mathbf{d}))$.*

Proof. We proceed inductively, and suppose that we already have computed $T_1,\dots,T_{n-1}$ as the associated triangular set of $W = \pi(V)$. We will write $\mathbf{d} = (d_1,\dots,d_n)$ and $\mathbf{e} = (d_1,\dots,d_{n-1})$.

For β in W, let T_β be the polynomial $\prod_{a\in v_\beta}(X_n - a) \in \mathsf{R}[X_n]$. For $j < d_n$, let further $T_{j,n}$ be the polynomial in $\mathsf{Span}(M_{\mathbf{e}})$ that interpolates the jth coefficient of the polynomials T_β at W; for $j = d_n$, we take $T_{d_n,n} = 1$. We then write $T_n = \sum_{j\le d_n} T_{j,n}X_n^j$: this polynomial is in $\mathsf{R}[X_1,\dots,X_n]$, monic of degree d_n in X_n, has degree less than d_i in X_i, for $i < n$, and vanishes on V. Thus, the polynomials $\mathbf{T} = (T_1,\dots,T_n)$ form the triangular set we are looking for. The algorithm is in Figure 4; we use a function $\mathsf{PolyFromRoots}$ to compute the polynomials T_β.

For a given β in W, the function $\mathsf{PolyFromRoots}$ computes T_β in $\mathsf{C}(d_n)$ base ring operations; this implies that given $T_1,\dots,T_{n-1}$, one can construct T_n using $d_1\cdots d_{n-1}\mathsf{C}(d_n) + \mathsf{C}_{\mathsf{Interp}}(d_1,\cdots,d_{n-1})d_n$ operations. The total cost for constructing all T_i is thus at most

$$\sum_{i\le n} d_1\cdots d_{i-1}\mathsf{C}(d_i) + \sum_{i\le n}\mathsf{C}_{\mathsf{Interp}}(d_1,\cdots,d_{i-1})d_i.$$

```
AssociatedTriangularSet(V, n)
1   if n = 0 return []
2   W ← π(V)
3   (T_1, ..., T_{n-1}) ← AssociatedTriangularSet(W, n)
4   for β in W do
4.1     T_β ← PolyFromRoots(v_β)
5   for j = 0, ..., d_n − 1 do
5.1     T_{j,n} ← Interp([coeff(T_β, X_n, j) | β ∈ W], W)
6   return Σ_{j<d_n} T_{j,n} X_n^j + X_n^{d_n}
```

Fig. 4. Associated triangular set of V.

Using the trivial bound $d_1 \cdots d_i \leq \delta_{\mathbf{d}}$ for the left-hand term, and the bound given in Proposition 2 for the right-hand one, we get the upper bounds

$$\delta_{\mathbf{d}} \sum_{i \leq n} \frac{\mathsf{C}(d_i)}{d_i} + \sum_{i \leq n} d_1 \cdots d_i \sum_{j \leq i-1} \frac{\mathsf{C}(d_j)}{d_j} \;\leq\; \delta_{\mathbf{d}} \sum_{i \leq n} \frac{\mathsf{C}(d_i)}{d_i} + \sum_{i \leq n} d_1 \cdots d_i \sum_{j \leq n} \frac{\mathsf{C}(d_j)}{d_j}.$$

Using the upper bound $\sum_{i \leq n} d_1 \cdots d_i \leq 2\delta_{\mathbf{d}}$, we finally obtain the estimate $3\delta_{\mathbf{d}}\mathsf{L}(\mathbf{d})$. □

3.5. Multiplication

Using our evaluation and interpolation algorithms, it becomes immediate to perform multiplication modulo a triangular set $\mathbf{T}$ associated to an equiprojectable set.

Proposition 4. *Let $V \subset \mathsf{R}^n$ be an equiprojectable set of multi-degree $\mathbf{d} = (d_1, \ldots, d_n)$ that supports interpolation, and let $\mathbf{T}$ be the associated triangular set. Then one can perform multiplication modulo $\langle \mathbf{T} \rangle$ in time $O(\delta_{\mathbf{d}}\mathsf{L}(\mathbf{d}))$.*

Proof. The algorithm is the same as in [22, Section 2.2], except that we now use the more general evaluation and interpolation algorithms presented here. Let A and B be reduced modulo $\langle \mathbf{T} \rangle$, and let $C = AB \bmod \langle \mathbf{T} \rangle$. Then for all α in V, $C(\alpha) = A(\alpha)B(\alpha)$. Since C is reduced modulo $\langle \mathbf{T} \rangle$, it suffices to interpolate the values $A(\alpha)B(\alpha)$ to obtain C. The cost is thus that of two evaluations, one interpolation, and of all pairwise products; the bounds of Propositions 1 and 2 conclude the proof. □

```
Mul(A, B, V)
1   Val_A ← Eval(A, V)
2   Val_B ← Eval(B, V)
3   Val_C ← [ Val_A(α)Val_B(α) | α ∈ V ]
4   return Interp(Val_C, V)
```

Fig. 5. Multiplication algorithm.

4. Homotopy techniques for multiplication

Let $\mathbf{T}$ be a triangular set in $\mathsf{R}[X_1, \ldots, X_n]$. We saw in the previous section that if $\mathbf{T}$ has all its roots in R, and if $V(\mathbf{T})$ supports interpolation, then multiplication modulo $\langle \mathbf{T} \rangle$ can be done in quasi-linear time. In this section, we extend this approach to an arbitrary $\mathbf{T}$ by setting up an homotopy between $\mathbf{T}$ and a new, more convenient, triangular set $\mathbf{U}$. This extends the approach of [22, Section 2.2], which dealt with the case where T_i is in $\mathsf{R}[X_i]$ for all i.

Let $\mathbf{d}$ be the multi-degree of $\mathbf{T}$ and assume that there exists an equiprojectable set V in R^n which supports interpolation and has multi-degree $\mathbf{d}$. Let $\mathbf{U}$ be the triangular set associated to V and let η be a new variable. We then define the set $\mathbf{S}$ in $\mathsf{R}[[\eta]][X_1, \ldots, X_n]$ by

$$S_i = \eta T_i + (1 - \eta) U_i, \quad 1 \leq i \leq n.$$

Since $\mathbf{U}$ and $\mathbf{T}$ have the same multi-degree $\mathbf{d}$, this set $\mathbf{S}$ is triangular, with multi-degree $\mathbf{d}$.

In Subsection 4.1, we prove that $\mathbf{S}$ has all its roots in $\mathsf{R}[[\eta]]$. Thus, we can use evaluation-interpolation techniques to do multiplication modulo $\langle \mathbf{S} \rangle$; this will in turn be used to perform multiplication modulo $\langle \mathbf{T} \rangle$.

The algorithm involves computing with power series; the quantity that will determine the cost of the algorithm will be the required precision in η. For $\mathbf{e} = (e_1, \ldots, e_n)$ in $\mathbb{N}^n$, we define

$$H_0(e_1, \ldots, e_n) = \deg(X_1^{e_1} \cdots X_n^{e_n} \bmod \langle \mathbf{S} \rangle, \eta)$$

and

$$H(e_1, \ldots, e_n) = \max_{e'_1 \leq e_1, \ \ldots, \ e'_n \leq e_n} H_0(e'_1, \ldots, e'_n).$$

Let us then define $r = H(2d_1 - 2, \ldots, 2d_n - 2)$. Subsection 4.2 shows that multiplication modulo $\langle \mathbf{T} \rangle$ can be performed in time $O^{\sim}(\delta_{\mathbf{d}} r)$. Finally, in Subsection 4.3, we give upper bounds on r that are determined by the monomial support of $\mathbf{S}$; this is the technical core of this article.

4.1. Computing the roots of $\mathbf{S}$

We show here that $\mathbf{S}$ has all its roots in $\mathsf{R}[[\eta]]$, by a straightforward application of Hensel's lemma.

First, we need some notation. Given positive integers k, ℓ and a subset $A \subset \mathsf{R}[[\eta]]^k$, $A \bmod \eta^\ell$ denotes the set $[a \bmod \eta^\ell \mid a \in A]$. Besides, we usually denote objects over $\mathsf{R}[[\eta]]$ with a * superscript, to distinguish them from their counterparts over R. Finally, we extend the notation π to denote the following projection

$$\begin{aligned} \pi : \quad \mathsf{R}[[\eta]]^n \quad &\to \quad \mathsf{R}[[\eta]]^{n-1} \\ (\alpha_1^\star, \ldots, \alpha_n^\star) &\mapsto (\alpha_1^\star, \ldots, \alpha_{n-1}^\star). \end{aligned}$$

Recall in what follows that $V \subset \mathsf{R}^n$ is equiprojectable of multi-degree $\mathbf{d}$, that its associated triangular set is $\mathbf{U}$, and that $\mathbf{S} = \eta \mathbf{T} + (1 - \eta)\mathbf{U}$.

Proposition 5. *There exists a unique set $V^\star$ in $\mathsf{R}[[\eta]]^n$ such that the following holds:*

- $V = V^\star \bmod \eta$*;*
- $V^\star$ *is equiprojectable of multi-degree* $\mathbf{d}$*;*

- $V^\star$ *supports interpolation;*
- $\mathbf{S}$ *is the triangular set associated to* $V^\star$.

Proof. We first claim that for $i \leq n$ and $\alpha = (\alpha_1, \ldots, \alpha_n)$ in V, the partial derivative

$$\frac{\partial S_i}{\partial X_i}(\eta = 0, \alpha_1, \ldots, \alpha_n) = \frac{\partial U_i}{\partial X_i}(\alpha_1, \ldots, \alpha_n) = \frac{\partial U_i}{\partial X_i}(\alpha_1, \ldots, \alpha_i)$$

is non zero. Let indeed $\alpha' = (\alpha_1, \ldots, \alpha_{i-1}) \in \mathsf{R}^{i-1}$. Then, we have by construction

$$U_i(\alpha_1, \ldots, \alpha_{i-1}, X_i) = \prod_{a \in v_{\alpha'}} (X_i - a),$$

so that the previous partial derivative equals

$$\frac{\partial U_i}{\partial X_i}(\alpha_1, \ldots, \alpha_{i-1}, \alpha_i) = \prod_{a \in v_{\alpha'}, a \neq \alpha_i} (\alpha_i - a).$$

Since V supports interpolation, this quantity is a product of units, so it is a unit as well, establishing our claim.

Since the system $\mathbf{S}$ is triangular, its Jacobian determinant is the product of the partial derivatives $\partial S_i / \partial X_i$. By the previous remark, all these derivatives are units at α, so the Jacobian itself is a unit at α. As a consequence, by Hensel's lemma, for all α in V, there exists a unique $\alpha^\star$ in $\mathsf{R}[[\eta]]^n$ such that $\alpha = \alpha^\star \bmod \eta$ and $\mathbf{S}(\alpha^\star) = 0$. We thus let $V^\star \subset \mathsf{R}^n$ be the set of all such $\alpha^\star$; hence $V = V^\star \bmod \eta$ and $\mathbf{S}$ vanishes at $V^\star$.

Next, we prove that $V^\star$ is equiprojectable of multi-degree $\mathbf{d}$. By induction, we can assume that we have proved that $\pi(V^\star)$ is equiprojectable of multi-degree $(d_1, \ldots, d_{n-1})$; it suffices to prove that for each $\beta^\star$ in $\pi(V^\star)$, the fiber $V^\star_{\beta^\star}$ has cardinality d_n.

Let thus $\alpha^\star$ be in $V^\star_{\beta^\star}$. We prove that for all $\gamma^\star$ in $V^\star$, $\pi(\alpha^\star) = \pi(\gamma^\star)$ if and only if $\pi(\alpha) = \pi(\gamma)$, with $\alpha = \alpha^\star \bmod \eta$ and $\gamma = \gamma^\star \bmod \eta$. To prove our claim, remark first that if $\pi(\alpha^\star) = \pi(\gamma^\star)$ then $\pi(\alpha) = \pi(\gamma)$, by reduction modulo η. Conversely, suppose that $\pi(\alpha) = \pi(\gamma)$. Since the system $\mathbf{S}$ is triangular, and since $\alpha^\star$ and $\gamma^\star$ are obtained by lifting α and γ using this system, we deduce that $\pi(\alpha^\star) = \pi(\gamma^\star)$, as requested. Thus, $V^\star$ is equiprojectable of multi-degree $\mathbf{d}$.

Finally, we prove that $V^\star$ supports interpolation. This is again done by induction: assume that the projection $\pi(V^\star)$ supports interpolation, let $\beta^\star$ be in $\pi(V^\star)$, and let $a^\star$ and $a'^\star$ be in $v^\star_\beta$. By assumption on V, $a - a' \bmod \eta$ is a unit in R; thus, by Hensel's lemma, $a^\star - a'^\star$ is a unit in $\mathsf{R}[[\eta]]$, as requested.

This proves the existence of $V^\star$ with the requested properties. Uniqueness follows in a straightforward manner from the uniqueness property of Hensel's lemma. □

We continue with complexity estimates: we prove that the roots of $\mathbf{S}$ can be computed in quasi-linear time.

Proposition 6. *Given* $\mathbf{T}$, V *and* $\ell > 0$, *one can compute* $V^\star \bmod \eta^\ell$ *in time* $O(\delta_{\mathbf{d}} \mathsf{L}(\mathbf{d}) \mathsf{M}(\ell))$.

Proof. As before, we proceed inductively: we suppose that the projection $W^\star = \pi(V^\star)$ is known modulo η^ℓ, and show how to deduce $V^\star \bmod \eta^\ell$. To do so, we evaluate all coefficients of S_n at all points of $W^\star$ modulo η^ℓ. Then, for each $\beta^\star$ in $W^\star$, it suffices to use Hensel's lemma to lift the roots of $S_n(\beta^\star, X_n)$ at precision ℓ. The pseudo-code is in Figure 6; for simplicity, we write there $W^\star$ instead of $W^\star \bmod \eta^\ell$.

```
LiftRootsMultivariate(V, S, ℓ)
1   n = |S|
2   if n = 0 return []
3   W* ← LiftRootsMultivariate(π(V), (S_1, ..., S_{n−1}), ℓ)
4   for i = 0, ..., d_n − 1 do
4.1     val_i ← Eval(coeff(S_n, X_n, i), W*)
        /* all computations are done modulo η^ℓ */
5   for β* in W* do
5.1     S_{β*} ← Σ_{i<d_n} val_{i,β*} X_n^i + X_n^{d_n}
5.2     v*_{β*} ← LiftRoots(S_{β*}, v_β, ℓ)
6   return [v*_{β*} | β* ∈ W*]
```

Fig. 6. Lifting the roots of $\mathbf{S}$.

Lemma 1 shows that we can lift the power series roots of a bivariate polynomial of degree d at precision ℓ in time $\mathsf{C}(d)\mathsf{M}(\ell)$. As a consequence, the overall cost $\mathsf{C}_{\mathsf{Lift}}$ of the lifting process satisfies

$$\begin{aligned}\mathsf{C}_{\mathsf{Lift}}(d_1,\ldots,d_n,\ell) \quad \leq \quad & \mathsf{C}_{\mathsf{Lift}}(d_1,\ldots,d_{n-1},\ell) + \mathsf{C}_{\mathsf{Eval}}(d_1,\ldots,d_{n-1})d_n\mathsf{M}(\ell)\\ & + d_1\cdots d_{n-1}\mathsf{C}(d_n)\mathsf{M}(\ell);\end{aligned}$$

the middle term gives the cost of evaluating the coefficients of S_n at $W^\star \bmod \eta^\ell$ (so we apply our evaluation algorithm with power series coefficients); and the right-hand term gives the cost of lifting the roots of S_n. This gives

$$\mathsf{C}_{\mathsf{Lift}}(d_1,\ldots,d_n,\ell) \leq \sum_{i\leq n} \mathsf{C}_{\mathsf{Eval}}(d_1,\ldots,d_{i-1})d_i\mathsf{M}(\ell) + \sum_{i\leq n} d_1\cdots d_{i-1}\mathsf{C}(d_i)\mathsf{M}(\ell).$$

As in the proof of Proposition 3, one deduces that the overall sum is bounded by

$$3\delta_{\mathbf{d}} \sum_{i\leq n} \frac{\mathsf{C}(d_i)}{d_i}\mathsf{M}(\ell) = 3\delta_{\mathbf{d}}\mathsf{L}(\mathbf{d})\mathsf{M}(\ell),$$

which concludes the proof. □

4.2. Multiplication by homotopy

We continue with the same notation as before. To multiply two polynomials $A, B \in \mathsf{Span}(M_{\mathbf{d}})$ modulo $\langle \mathbf{T}\rangle$, we may multiply them modulo $\langle \mathbf{S}\rangle$ over $\mathsf{R}[\eta]$ and let $\eta = 1$ in the result. The results of the multiplication modulo $\langle \mathbf{S}\rangle$ over $\mathsf{R}[\eta]$ and over $\mathsf{R}[[\eta]]$ are the same; when working over $\mathsf{R}[[\eta]]$, we may use the evaluation-interpolation techniques from Subsection 3.5. Indeed, by Proposition 5, $\mathbf{S}$ is associated to a subset $V^\star$ of $\mathsf{R}[[\eta]]^n$ that supports interpolation.

Of course, when multiplying A and B modulo $\langle \mathbf{S}\rangle$ over $\mathsf{R}[[\eta]]$, we cannot compute with (infinite) power series, but rather with their truncations at a suitable order. On the one hand, this order should be larger than the largest degree of a coefficient of the multiplication of A and B modulo $\langle \mathbf{S}\rangle$ over $\mathsf{R}[\eta]$. On the other hand, this order will

determine the cost of the multiplication algorithm, so it should be kept to a minimum. For $\mathbf{e} = (e_1, \ldots, e_n)$ in $\mathbb{N}^n$, recall that we defined

$$H_0(e_1, \ldots, e_n) = \deg(X_1^{e_1} \cdots X_n^{e_n} \bmod \langle \mathbf{S} \rangle, \eta)$$

and

$$H(e_1, \ldots, e_n) = \max_{e'_1 \le e_1, \ \ldots, \ e'_n \le e_n} H_0(e'_1, \ldots, e'_n).$$

The following proposition relates the cost of our algorithm to the function H; the behavior of this function is studied in the next subsection.

Proposition 7. *Given $A, B, \mathbf{T}$ and V, one can compute $AB \bmod \langle \mathbf{T} \rangle$ in time $O(\delta_{\mathbf{d}} \mathsf{L}(\mathbf{d}) \mathsf{M}(r)) \subset O^{\tilde{}}(\delta_{\mathbf{d}} r)$, with $r = H(2d_1 - 2, \ldots, 2d_n - 2)$.*

Proof. The algorithm is simple: we compute $\mathbf{U}$ and use it to obtain $V^\star$ at a high enough precision. In $\mathsf{R}[X_1, \ldots, X_n]$, the product AB satisfies $\deg(AB, X_i) \le 2d_i - 2$ for all $i \le n$; since the multiplication algorithm does not perform any division by η, it suffices to apply it with coefficients in $\mathsf{R}[\eta]/\langle \eta^{r+1} \rangle$, with $r = H(2d_1 - 2, \ldots, 2d_n - 2)$. The resulting algorithm is given in Figure 7; as before, we write $V^\star$ for simplicity, whereas we should write $V^\star \bmod \eta^{r+1}$.

```
Mul(A, B, T, V)
0.  U ← AssociatedTriangularSet(V, n)
1.  S ← ηT + (1 − η)U
2.  V* ← LiftRootsMultivariate(V, S, r + 1)
3.  C_η ← Mul(A, B, V*)
4.  return C_η(1, X_1, ..., X_n)
    /* C_η is seen in R[η][X_1, ..., X_n] */
```

Fig. 7. Multiplication algorithm.

The computation of $\mathbf{U}$ takes time $O(\delta_{\mathbf{d}} \mathsf{L}(\mathbf{d}))$ by Proposition 3; that of $\mathbf{S}$ takes time $O(\delta_{\mathbf{d}})$. Computing $V^\star \bmod \eta^{r+1}$ takes time $O(\delta_{\mathbf{d}} \mathsf{L}(\mathbf{d}) \mathsf{M}(r))$ by Proposition 5. The modular multiplication takes time $O(\delta_{\mathbf{d}} \mathsf{L}(\mathbf{d}) \mathsf{M}(r))$ by Proposition 4; remark that this algorithm is run with coefficients in $\mathsf{R}[[\eta]]/\langle \eta^{r+1} \rangle$, where all arithmetic operations take time $O(\mathsf{M}(r))$. Finally, specializing η at 1 takes time $O(\delta_{\mathbf{d}} r)$. Summing all these costs gives our result. □

4.3. Precision analysis

In this section, we study the functions H_0 and H introduced in the previous section, and show how they are related to the monomial support of the polynomials in the triangular set $\mathbf{T}$. Before giving a general bound for these functions, we discuss an example, to motivate our general result.

4.3.1. A worked example

For this example, we assume that $n = 2$ and that the triangular set $\mathbf{S} = (S_1, S_2)$ is as follows:

$$S_1 = X_1^3 - \eta X_1^2 - \eta X_1 - \eta$$
$$S_2 = X_2^3 - \eta X_2^2 - \eta X_1^2.$$

Here, the multi-degree of $\mathbf{S}$ is $\mathbf{d} = (3, 3)$. Our question of estimating the functions H_0 and H then essentially boils down to the following: given a monomial m in X_1, X_2, what will be the degree in η of the normal form of m modulo $\langle \mathbf{S} \rangle$? Taking $m = X_1^4 X_2^6$, we are going to follow the steps of this reduction to show the growth of the degrees.

Reducing $m = X_1^4 X_2^6$ with respect to S_2 once, we get $\eta X_1^4 X_2^5 + \eta X_1^6 X_2^3$. If we reduce $m_1 = \eta X_1^4 X_2^5$ with respect to S_2, we get $\eta^4 X_1^4 X_2^2 + \cdots + \eta^4 X_1^6$; reducing $\eta^4 X_1^6$ with respect to S_1 increases the degree in η by 4, reaching a degree 8 (this is maximum we obtain). If we reduce $m_2 = \eta X_1^6 X_2^3$ with respect to S_2, we obtain $\eta^2 X_1^6 X_2^2 + \eta^2 X_1^8$; the reduction of X_1^8 with respect to S_1 increases the degree in η by 6, and the resulting degree will be 8 as well.

The reductions with respect to S_2 amount to replace X_2^3 by $\eta X_2^2 + \eta X_1^2$. They can be described graphically using a tree: the initial monomial m is the root, and the children of a node correspond to the two monomials that appear after one reduction step. This enables us to dispense with writing the η factors: the degree in η of any monomial is the length of the path from the root to the corresponding node. Figure 8 shows this tree for the initial monomial $X_1^4 X_2^6$.

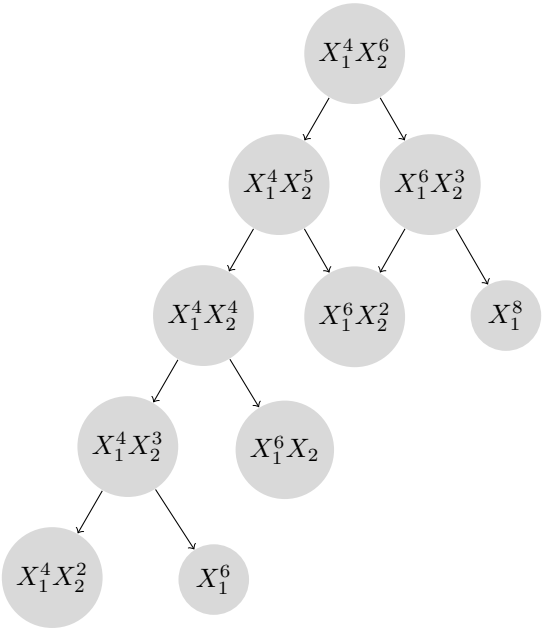

Fig. 8. Tree structure of the reduction by S_2.

In this tree, one step to the left reduces the degree in X_2 by 1; one step to the right reduces the degree in X_2 by 3 and increases the degree in X_1 by 2. We perform reductions as long as the degree in X_2 remains at least 3. When we reach a leaf, with degree r in X_1, we increase the degree in η by $r-2$ through the reduction by S_1.

With these rules, our goal is to find an upper bound on the degree in η in the result. We introduce two variables f_1 and f_2, that respectively represent the number of steps to the left and to the right. After (f_1, f_2) steps, the degree in X_1 is $4+2f_2$ and the degree in X_2 is $6-f_1-3f_2$, so that the end-points of our walk satisfy $0 \leq 6-f_1-3f_2 \leq 2$. Figure 9 depicts this situation in the plane of coordinates (f_1, f_2); the lower oblique line has equation $6-f_1-3f_2=2$ and the higher one has equation $6-f_1-3f_2=0$. Also, at each point with integer coordinates (f_1, f_2), we indicate the degrees in (X_1, X_2); the origin $(f_1, f_2) = (0, 0)$ is in the lower corner on the left. The points in **bold face** can be reached by our reduction process.

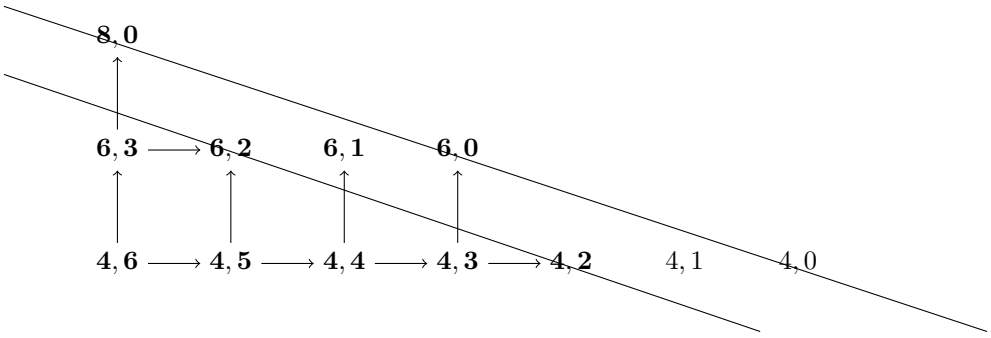

Fig. 9. A view of the reduction tree in the coordinates (f_1, f_2).

This walk only describes reduction with respect to S_2; we still need to reduce with respect to S_1. This is done by remembering that the reduction of a monomial $X_1^{r_1}X_2^{r_2}$ with respect to S_1 increases the degree in η by r_1-2.

Thus, starting from $X_1^4X_2^6$, the degree in η we obtain after reduction is bounded by the maximal value of the sum $(f_1+f_2)+(2+2f_2) = 2+f_1+3f_2$ over all (f_1, f_2) that lie in between the two oblique lines: the term (f_1+f_2) gives the degree increase due to the number of steps (reduction with respect to S_2), and the term $2+2f_2 = 4+2f_2-2$ gives the degree increase due to the reduction with respect to S_1 (remember that the degree in X_1 is $4+2f_2$).

To simplify the maximization, we can forget the lower constraint $6-f_1-3f_2 \leq 2$, and maximize in the larger area defined by $f_1 \geq 0,\ f_2 \geq 0,\ 6-f_1-3f_2 \geq 0$; we can also remove the condition that f_1 and f_2 are integers. Indeed, both simplifications increase the size of the search space, so any upper-bound obtained after simplifications will be an upper bound for the initial problem as well.

After these simplifications, we are left to maximize a linear function over a simplex; the maximum will thus be obtained at one of the vertices. The vertex $(f_1, f_2) = (0, 0)$ is not a maximum (it is the minimum); the other vertices are $(f_1, f_2) = (0, 2)$ and $(f_1, f_2) = (6, 0)$, and the maximum of $2+f_1+3f_2$ is 8, obtained at both vertices (in this case, we obtain the *exact* maximum for our original problem; in most cases, this won't be guaranteed).

4.3.2. General analysis

We will now formalize the former considerations, showing how the monomial structure of the polynomials in $\mathbf{S}$ affects the cost of the algorithm. For $i \leq n$ and $\nu = (\nu_1, \ldots, \nu_i)$ in $\mathbb{N}^i$, we will use the notation $\mathbf{X}_i^\nu = X_1^{\nu_1} \cdots X_i^{\nu_i}$ and we write the monomial expansion of S_i as

$$S_i = X_i^{d_i} + \sum_{\nu \in E_i} s_\nu \mathbf{X}_i^\nu, \tag{4}$$

where E_i is the set of exponents that appear in S_i, the exponents ν are in $\mathbb{N}^i$, and s_ν is linear in η. Let us further introduce the coefficients h_i defined by $h_0 = 0$ and for $i \geq 1$,

$$h_i = \max_{\nu \in E_i} \frac{h_1\nu_1 + \cdots + h_{i-1}\nu_{i-1} + 1}{d_i - \nu_i}. \tag{5}$$

One easily checks that all h_i are positive. The following proposition shows that through the coefficients h_i, the support E_i determines the cost of our algorithm.

Proposition 8. *The inequality*

$$H(e_1, \ldots, e_n) \leq h_1 e_1 + \cdots + h_n e_n$$

holds for all $(e_1, \ldots, e_n) \in \mathbb{N}^n$.

Using Proposition 7, this proposition gives as an easy corollary the following statement, where we take $e_i = 2d_i - 2 \leq 2d_i$; we still use the previous notation $\mathbf{T}$ and V.

Corollary 1. *Given* $A, B, \mathbf{T}$ *and* V*, one can compute* $AB \bmod \langle \mathbf{T} \rangle$ *in time* $O(\delta_{\mathbf{d}} \mathsf{L}(\mathbf{d}) \mathsf{M}(r)) \subset O^{\sim}(\delta_{\mathbf{d}} r)$*, with* $r \leq 2(h_1 d_1 + \cdots + h_n d_n)$.

Hence, the lower the h_i the better. However, without putting extra assumptions on the monomial supports E_i, Corollary 1 only yields estimates of little interest. Even in sparse cases, it remains difficult to simplify the recurrence giving the coefficients h_i. Still, several examples in the next section will show that for some useful families of monomial supports, significantly sharper bounds can be derived.

The rest of this section is devoted to prove Proposition 8. In all that follows, the multi-degree $\mathbf{d} = (d_1, \ldots, d_n)$ and the supports E_i are fixed. We also let E_i' be the set of modified exponents

$$E_i' = \{\nu - (0, \ldots, 0, d_i) \in \mathbb{Z}^i \mid \nu \in E_i\},$$

so that for all $\nu = (\nu_1, \ldots, \nu_i)$ in E_i', $\nu_j \geq 0$ for $j < i$ and $\nu_i < 0$. Hence, Equation (5) takes the (slightly more handy) form

$$h_i = \max_{\nu \in E_i'} \frac{h_1\nu_1 + \cdots + h_{i-1}\nu_{i-1} + 1}{-\nu_i}. \tag{6}$$

Since the proof is rather technical, we will first give a roadmap of it.

Recall that the function H_0 was defined with domain $\mathbb{N}^n$; in what follows, we also see it as a function over $\mathbb{N}^i$, for $1 \leq i \leq n$, by defining $H_0(e_1, \ldots, e_i) = H_0(e_1, \ldots, e_i, 0, \ldots, 0)$, where the right-hand expression contains $n - i$ zeros; for completeness, we write $H_0() = 0$ for $i = 0$. Then, for $\mathbf{e} = (e_1, \ldots, e_i)$ in $\mathbb{N}^i$, $\mathbf{e}' = (e_1, \ldots, e_{i-1})$ in $\mathbb{N}^{i-1}$ and $i \geq 1$, Lemma 2 proves that

$$H_0(\mathbf{e}) = H_0(\mathbf{e}') \quad \text{if} \quad e_i < d_i, \qquad H_0(\mathbf{e}) \leq 1 + \max_{\nu \in E_i'} H_0(\mathbf{e} + \nu) \quad \text{otherwise;}$$

this translates the fact that if $\mathbf{X}^{\mathbf{e}}$ is reduced with respect to S_i, we can drop the last variable X_i; otherwise, we do one reduction step by S_i. Iterating this process, Lemma 3 proves that

$$H_0(\mathbf{e}) \leq \max_{\substack{(f_\nu)_{\nu\in E_i'} \text{ non-negative integers} \\ \text{such that } 0 \leq e_i + \sum_{\nu\in E_i'} f_\nu \nu_i \leq d_i - 1}} H_0(\mathbf{e} + \sum_{\nu\in E_i'} f_\nu \nu) + \sum_{\nu\in E_i'} f_\nu.$$

This expression generalizes the example seen before, introducing integer-valued variables that count the reduction steps we do; the linear constraints we obtain are of the same nature as the ones seen in the example. Finally, Lemma 4 will "solve" this optimization problem, by observing (as in the example) that we can find a reasonable bound on the maximum by inspecting the vertices of a simplex.

We will now prove the inequalities showed before. The following recurrence relation enables us to control the growth of H_0.

Lemma 2. *For $i \geq 1$, let $\mathbf{e} = (e_1, \ldots, e_i)$ be in $\mathbb{N}^i$ and let $\mathbf{e}' = (e_1, \ldots, e_{i-1})$ in $\mathbb{N}^{i-1}$. Then the following (in)equalities hold:*

$$H_0(\mathbf{e}) = H_0(\mathbf{e}') \quad \text{if} \quad e_i < d_i, \qquad H_0(\mathbf{e}) \leq 1 + \max_{\nu\in E_i'} H_0(\mathbf{e}+\nu) \quad \text{otherwise.}$$

Proof. Let us first suppose $e_i < d_i$; then,

$$X_1^{e_1}\cdots X_i^{e_i} \bmod \langle \mathbf{S}\rangle = (X_1^{e_1}\cdots X_{i-1}^{e_{i-1}} \bmod \langle \mathbf{S}\rangle) X_i^{e_i},$$

since the latter product is reduced modulo $\langle \mathbf{S}\rangle$. Both sides have thus the same degree in η, and our first claim follows.

We can now focus on the case $e_i \geq d_i$, for which we write $f_i = e_i - d_i$, so that $f_i \geq 0$. From Equation (4), we deduce

$$X_i^{f_i} S_i = X_i^{f_i+d_i} + \sum_{\nu\in E_i} s_\nu X_i^{f_i} \mathbf{X}_i^\nu,$$

and thus we get

$$X_i^{f_i} S_i = X_i^{e_i} + \sum_{\nu\in E_i'} s_\nu X_i^{e_i} \mathbf{X}_i^\nu,$$

by the definition of E_i'. In our notation, we have $X_1^{e_1}\cdots X_i^{e_i} = \mathbf{X}_i^{\mathbf{e}}$. Thus, after multiplication by $X_1^{e_1}\cdots X_{i-1}^{e_{i-1}}$ and term reorganization, the former equality implies that

$$\mathbf{X}_i^{\mathbf{e}} - X_1^{e_1}\cdots X_{i-1}^{e_{i-1}} X_i^{f_i} S_i = -\sum_{\nu\in E_i'} s_\nu \mathbf{X}_i^{\mathbf{e}+\nu}.$$

As a consequence, we deduce that

$$\deg(\mathbf{X}_i^{\mathbf{e}} \bmod \langle \mathbf{S}\rangle, \eta) \;\leq\; \max_{\nu\in E_i'} \deg(s_\nu \mathbf{X}_i^{\mathbf{e}+\nu} \bmod \langle \mathbf{S}\rangle, \eta).$$

Since for ν in E_i', we have

$$\deg(s_\nu \mathbf{X}_i^{\mathbf{e}+\nu} \bmod \langle \mathbf{S}\rangle, \eta) = \deg(s_\nu, \eta) + \deg(\mathbf{X}_i^{\mathbf{e}+\nu} \bmod \langle \mathbf{S}\rangle, \eta) = 1 + H_0(\mathbf{e}+\nu),$$

the conclusion follows. □

Iterating the process of the previous lemma, we obtain the following bound. In the next lemma, $(f_\nu)_{\nu\in E'_i}$ is a family of integer valued variables.

Lemma 3. *Let $\mathbf{e}=(e_1,\dots,e_i)$ be in $\mathbb{N}^i$. Then the following inequality holds:*

$$H_0(\mathbf{e}) \leq \max_{\substack{(f_\nu)_{\nu\in E'_i} \text{ non-negative integers} \\ \text{such that } 0\leq e_i+\sum_{\nu\in E'_i} f_\nu\nu_i\leq d_i-1}} H_0(\mathbf{e}+\sum_{\nu\in E'_i} f_\nu\nu)+\sum_{\nu\in E'_i} f_\nu.$$

Proof. We prove the claim by induction on e_i. For $e_i\leq d_i-1$, the family $(f_\nu=0)_{\nu\in E'_i}$ satisfies the constraint $0\leq e_i+\sum_{\nu\in E'_i} f_\nu\nu_i\leq d_i-1$; for this choice, the value of the function we maximize is precisely $H_0(\mathbf{e})$, so our claim holds. Suppose now that $e_i\geq d_i$. Then, the previous lemma gives

$$H_0(\mathbf{e})\leq 1+\max_{\nu\in E'_i} H_0(\mathbf{e}+\nu). \tag{7}$$

Let us fix ν in E'_i; then $\mathbf{e}+\nu$ has non-negative integer coordinates, and its ith coordinate is less than e_i. Thus, we can apply the induction assumption, obtaining

$$H_0(\mathbf{e}+\nu)\leq \max_{\substack{(f_{\nu'})_{\nu'\in E'_i} \text{ non-negative integers} \\ \text{such that } 0\leq e_i+\nu_i+\sum_{\nu'\in E'_i} f_{\nu'}\nu'_i\leq d_i-1}} H_0(\mathbf{e}+\nu+\sum_{\nu'\in E'_i} f_{\nu'}\nu')+\sum_{\nu'\in E'_i} f_{\nu'}.$$

To any set of non-negative integers $(f_{\nu'})_{\nu'\in E'_i}$ with

$$0\leq e_i+\nu_i+\sum_{\nu'\in E'_i} f_{\nu'}\nu'_i\leq d_i-1$$

appearing in the previous maximum, we associate the non-negative integers $(f'_{\nu'})_{\nu'\in E'_i}$, with $f'_\nu=f_\nu+1$ and $f'_{\nu'}=f_{\nu'}$ otherwise. These new integers satisfy

$$0\leq e_i+\sum_{\nu'\in E'_i} f'_{\nu'}\nu'_i\leq d_i-1$$

and

$$H_0(\mathbf{e}+\nu+\sum_{\nu'\in E'_i} f_{\nu'}\nu')+\sum_{\nu'\in E'_i} f_{\nu'}=H_0(\mathbf{e}+\sum_{\nu'\in E'_i} f'_{\nu'}\nu')+\sum_{\nu'\in E'_i} f'_{\nu'}-1.$$

Taking maxima, we deduce from the previous inequality

$$H_0(\mathbf{e}+\nu)\leq \max_{\substack{(f'_{\nu'})_{\nu'\in E'_i} \text{ non-negative integers} \\ \text{such that } 0\leq e_i+\sum_{\nu'\in E'_i} f'_{\nu'}\nu'_i\leq d_i-1}} H_0(\mathbf{e}+\sum_{\nu'\in E'_i} f'_{\nu'}\nu')+\sum_{\nu'\in E'_i} f'_{\nu'}-1.$$

Substituting in Equation (7) and taking the maximum over ν in E'_i concludes the proof. □

For $i\leq n$, let L_i be the linear form $(e_1,\dots,e_i)\mapsto h_1e_1+\cdots+h_ie_i$, where the h_i are as in Equation (5). The following lemma concludes the proof of Proposition 8; as we did for H_0, for $i\leq n$, we extend H to $\mathbb{N}^i$, by writing $H(e_1,\dots,e_i)=H(e_1,\dots,e_i,0,\dots,0)$.

Lemma 4. *For $i \leq n$ and $\mathbf{e} = (e_1, \ldots, e_i)$ in $\mathbb{N}^i$, the inequality $H(\mathbf{e}) \leq L_i(\mathbf{e})$ holds.*

Proof. It is sufficient to prove that $H_0(\mathbf{e}) \leq L_i(\mathbf{e})$ holds; since all coefficients of L_i are non-negative, L_i is non-decreasing with respect to all of its variables, which implies the thesis.

We prove our inequalities by induction on $i \geq 0$. For $i = 0$, we have $H_0() = L_0() = 0$; hence, our claim vacuously holds at this index. For $i \geq 1$, we now prove that if our inequality holds at index $i-1$, it will also hold at index i. Lemma 3 shows that for any $\mathbf{e} \in \mathbb{N}^i$, we have the inequality

$$H_0(\mathbf{e}) \leq \max_{\substack{(f_\nu)_{\nu \in E'_i} \text{ non-negative integers} \\ \text{such that } 0 \leq e_i + \sum_{\nu \in E'_i} f_\nu \nu_i \leq d_i - 1}} H_0(\mathbf{e} + \sum_{\nu \in E'_i} f_\nu \nu) + \sum_{\nu \in E'_i} f_\nu.$$

Let φ be the natural projection $\mathbb{N}^i \to \mathbb{N}^{i-1}$, let $(f_\nu)_{\nu \in E'_i}$ be non-negative integers that satisfy the conditions in the previous inequality. Since $\mathbf{e} + \sum_{\nu \in E'_i} f_\nu \nu$ has degree in X_i less than d_i, the first point of Lemma 2 shows that

$$H_0(\mathbf{e} + \sum_{\nu \in E'_i} f_\nu \nu) \;=\; H_0(\varphi(\mathbf{e} + \sum_{\nu \in E'_i} f_\nu \nu));$$

the induction assumption implies that this quantity is bounded from above by

$$L_{i-1}(\varphi(\mathbf{e} + \sum_{\nu \in E'_i} f_\nu \nu)).$$

As a consequence, $H_0(\mathbf{e})$ admits the upper bound

$$\max_{\substack{(f_\nu)_{\nu \in E'_i} \text{ non-negative integers} \\ \text{such that } 0 \leq e_i + \sum_{\nu \in E'_i} f_\nu \nu_i \leq d_i - 1}} L_{i-1}(\varphi(\mathbf{e} + \sum_{\nu \in E'_i} f_\nu \nu)) + \sum_{\nu \in E'_i} f_\nu.$$

This quantity itself is upper-bounded by a similar expression, where we allow the f_ν to be non-negative reals numbers; this gives

$$H_0(\mathbf{e}) \;\leq \max_{\substack{(f_\nu)_{\nu \in E'_i} \text{ non-negative real numbers} \\ \text{such that } 0 \leq e_i + \sum_{\nu \in E'_i} f_\nu \nu_i \leq d_i - 1}} L_{i-1}(\varphi(\mathbf{e} + \sum_{\nu \in E'_i} f_\nu \nu)) + \sum_{\nu \in E'_i} f_\nu.$$

Since all h_i and all $\nu_1, \ldots, \nu_{i-1}$ are non-negative, the function of $(f_\nu)_{\nu \in E'_i}$ we want to maximize is affine with non-negative coefficients. The domain where we maximize it is defined by the conditions

$$f_\nu \geq 0 \text{ for all } \nu \in E'_i, \quad 0 \leq e_i + \sum_{\nu \in E'_i} f_\nu \nu_i \leq d_i - 1,$$

and it is contained in the domain D defined by the conditions

$$f_\nu \geq 0 \text{ for all } \nu \in E'_i, \quad 0 \leq e_i + \sum_{\nu \in E'_i} f_\nu \nu_i.$$

Since all unknowns f_ν are non-negative, while the coefficients ν_i are negative, the domain D is convex and bounded. Hence, the maximal value we look for is upper-bounded by the maximal value at the end-vertices of D, distinct from the origin; these vertices are

$$E_\nu = \{f_{\nu'} = 0 \ \text{ for } \ \nu' \neq \nu, \qquad f_\nu = -\frac{e_i}{\nu_i}\}, \quad \text{for} \quad \nu \in E_i'.$$

At the point E_ν, the objective function takes the value

$$L_{i-1}(\varphi(\mathbf{e} - \frac{e_i}{\nu_i}\nu)) - \frac{e_i}{\nu_i}.$$

By the linearity of L_{i-1} and φ, this can be rewritten as

$$L_{i-1}(\varphi(\mathbf{e})) - L_{i-1}(\varphi(\frac{e_i}{\nu_i}\nu)) - \frac{e_i}{\nu_i} \ = \ L_{i-1}(\varphi(\mathbf{e})) - \frac{L_{i-1}(\varphi(\nu)) + 1}{\nu_i} e_i.$$

As a consequence, we obtain the upper bound

$$H_0(\mathbf{e}) \ \leq L_{i-1}(\varphi(\mathbf{e})) + \max_{\nu \in E_i'} \frac{L_{i-1}(\varphi(\nu)) + 1}{-\nu_i} e_i.$$

To simplify this further, note that the term $L_{i-1}(\varphi(\mathbf{e}))$ rewrites as $h_1 e_1 + \cdots + h_{i-1} e_{i-1}$. Similarly, $L_{i-1}(\varphi(\nu)) + 1$ equals $h_1\nu_1 + \cdots + h_{i-1}\nu_{i-1}$. We deduce the inequality

$$H_0(\mathbf{e}) \ \leq \ h_1 e_1 + \cdots + h_{i-1} e_{i-1} \ + \ \max_{\nu \in E_i'} \frac{h_1\nu_1 + \cdots + h_{i-1}\nu_{i-1} + 1}{-\nu_i} e_i,$$

which we can finally rewrite as

$$H_0(\mathbf{e}) \ \leq \ h_1 e_1 + \cdots + h_{i-1} e_{i-1} + h_i e_i,$$

as requested. □

5. Examples

5.1. Main family of examples

We give explicit estimates for the coefficients h_i of the previous section on the following family of examples. We consider triangular sets $\mathbf{T} = (T_1, \ldots, T_n)$ such that T_i has the form

$$T_i = X_i^{d_i} + \sum_{\nu \in D_i} t_\nu \mathbf{X}_i^\nu, \quad t_\nu \in \mathsf{R}, \tag{8}$$

where all $\mathbf{X}_i^\nu$ are monomials in $X_1, \ldots, X_i$ of total degree at most λ_i, for some $\lambda_i \in \mathbb{N}$. We let $d = \max_{i \leq n} d_i$, and we suppose that R contains at least d pairwise distinct values $x_1, \ldots, x_d$, with $x_i - x_j$ a unit for $i \neq j$.

The following proposition illustrates three different situations. The first two cases display a cost quasi-linear in $d\delta_{\mathbf{d}}$, which is satisfying, especially for small d; the last one shows that small changes in the assumptions can induce large overheads. We will see in the next subsection cases where d_i is constant equal to d, or $d_i = n + 1 - i$; in such cases, d is logarithmic in $\delta_{\mathbf{d}}$ and the cost $O^{\tilde{}}(d\delta_{\mathbf{d}})$ is thus $O^{\tilde{}}(\delta_{\mathbf{d}})$, which is what we were aiming at.

Proposition 9. *With assumptions as above, multiplication modulo $\langle \mathbf{T} \rangle$ can be performed with the following complexities:*

$$\begin{aligned} O\big(\, n\,\delta_{\mathbf{d}}\,\tfrac{\mathsf{C}(d)}{d}\,\mathsf{M}(nd)\,\big) &\subset O^{\tilde{}}(d\delta_{\mathbf{d}}) && \text{if } \lambda_i = d_i - 1 \text{ for all } i,\\ O\big(\, n\,\delta_{\mathbf{d}}\,\tfrac{\mathsf{C}(d)}{d}\,\mathsf{M}(n^2d)\,\big) &\subset O^{\tilde{}}(d\delta_{\mathbf{d}}) && \text{if } \lambda_i = d_i \text{ for all } i,\\ O\big(\, n\,\delta_{\mathbf{d}}\,\tfrac{\mathsf{C}(d)}{d}\,\mathsf{M}(2^nd)\,\big) &\subset O^{\tilde{}}(2^nd\delta_{\mathbf{d}}) && \text{if } \lambda_i = d_i + 1 \text{ for all } i. \end{aligned}$$

Proof. First, we construct V: we simply choose the grid

$$V = [x_1, \ldots, x_{d_1}] \times \cdots \times [x_1, \ldots, x_{d_n}]. \tag{9}$$

Thus, we have $U_i = (X_i - x_1)\cdots(X_i - x_{d_i})$; as before we let $\mathbf{S} = \eta\mathbf{T} + (1-\eta)\mathbf{U}$. Thus, the monomial support E_i associated with S_i is contained in

$$D_i' = D_i \cup \{(0, \ldots, 0, \nu_i) \mid 0 \le \nu_i < d_i\}.$$

Since each monomial in D_i has an exponent of the form $(\nu_1, \ldots, \nu_i)$, with $\nu_1 + \cdots + \nu_i \le \lambda_i$ and $\nu_i < d_i$, we deduce from Equation (5) that

$$h_i \;\le\; \max_{\nu \in D_i'} \frac{h_1\nu_1 + \cdots + h_{i-1}\nu_{i-1} + 1}{d_i - \nu_i} \;\le\; \max\Big(\max_{\nu \in D_i} \frac{h_1\nu_1 + \cdots + h_{i-1}\nu_{i-1} + 1}{d_i - \nu_i},\; 1\Big).$$

Let $h_i' = \max(h_1, \ldots, h_i)$, so that

$$h_i \;\le\; \max\Big(\max_{\nu \in D_i} \frac{h_{i-1}'(\nu_1 + \cdots + \nu_{i-1}) + 1}{d_i - \nu_i}, 1\Big) \;\le\; \max\Big(\max_{\nu \in D_i} \frac{h_{i-1}'(\lambda_i - \nu_i) + 1}{d_i - \nu_i}, 1\Big). \tag{10}$$

Knowing the distribution of the d_i and λ_i, the former relation makes it possible to analyze the growth of the coefficients h_i, and thus of $2(d_1h_1 + \cdots + d_nh_n)$.

Case 1. Suppose first that $\lambda_i = d_i - 1$. Then, the former inequality implies $h_i \le 1$ for all i, so that $2(d_1h_1 + \cdots + d_nh_n) \le 2nd$.

Case 2. If $\lambda_i = d_i$, then (10) becomes $h_i \le h_{i-1}' + 1$, so that $h_i \le i$ for all i, and thus $2(d_1h_1 + \cdots + d_nh_n) \le n(n+1)d$.

Case 3. If finally $\lambda_i = d_i + 1$, then (10) becomes $h_i \le 2h_{i-1}' + 1$, so that $h_i' \le 2^i - 1$. In this case, we get $2(d_1h_1 + \cdots + d_nh_n) \le 2^{n+2}d$.

To conclude the proof, we simply plug the previous estimates in the cost estimate $O(\delta_{\mathbf{d}}\mathsf{L}(\mathbf{d})\mathsf{M}(r))$ of Corollary 1, with $r \le 2(d_1h_1 + \cdots + d_nh_n)$, and we use the upper bound $\mathsf{L}(\mathbf{d}) \le n\mathsf{C}(d)/d$ of Equation (3). □

5.2. Cauchy modules

Cauchy modules [27] are a basic construction in Galois theory and invariant theory [32, 27, 1, 26]. Starting from a monic polynomial $F \in \mathsf{R}[X]$ of degree d, we define a triangular set $F_1, \ldots, F_d$ by letting $F_1(X_1) = F(X_1)$ and taking iterated divided differences:

$$F_{i+1}(X_1, \ldots, X_{i+1}) = \frac{F_i(X_1, \ldots, X_{i-1}, X_i) - F_i(X_1, \ldots, X_{i-1}, X_{i+1})}{X_i - X_{i+1}} \qquad 1 \le i < d.$$

The polynomials $F_1, \ldots, F_d$ form a triangular set of multi-degree $\mathbf{d} = (d, d-1, \ldots, 1)$, so that $\delta_{\mathbf{d}} = d!$; their interest stems from the fact that they form a system of generators of

the ideal $(\sigma_i - (-1)^i f_{d-i})_{1\le i\le d}$, where σ_i is the ith elementary symmetric polynomial in $X_1, \ldots, X_d$ and f_i is the coefficient of X^i in F.

One easily checks that F_i has total degree at most $d+1-i$. Hence, assuming that $0, \ldots, d-1$ are units in R, we are under the assumptions of Subsection 5.1, with $\lambda_i = d_i = d+1-i$ for all i and $(x_1, \ldots, x_d) = (0, \ldots, d-1)$. As a consequence, Proposition 9 shows that multiplication modulo $\langle F_1, \ldots, F_d\rangle$ can be done using $O\big(d!\,\mathsf{C}(d)\,\mathsf{M}(d^3)\big)$ operations in R, that is, in quasi-linear time $O\tilde{}(d!)$. This improves for instance the results given in [17] on the evaluation properties of symmetric polynomials.

5.3. Polynomial multiplication

We show now how to derive quasi-linear time algorithms for *univariate* multiplication in $\mathsf{R}[X]$ from our previous multivariate construction. Unfortunately, our algorithm does not improve on the complexity of Cantor-Kaltofen's algorithm [9]; however, we believe it is worth mentioning. Precisely, given $n \ge 1$, we give here an algorithm to perform truncated multiplication in $\mathsf{R}[X]/\langle X^{2^n}\rangle$. We introduce variables $X_1, \ldots, X_n$; computing in $\mathsf{A} = \mathsf{R}[X]/\langle X^{2^n}\rangle$ is equivalent to computing in $\mathsf{B} = \mathsf{R}[X_1, \ldots, X_n]/\langle V_1, \ldots, V_n\rangle$, with $\mathbf{V} = (V_1, \ldots, V_n)$ given by

$$\left|\begin{array}{l} X_1 - X_n^{2^{n-1}} \\ \quad\vdots \\ X_{n-1} - X_n^2 \\ X_n^{2^n}, \end{array}\right.$$

since the dummy variables $X_1, \ldots, X_{n-1}$ play no role in this representation. However, changing the order of the variables, we see that the ideal $\langle V_1, \ldots, V_n\rangle$ is also equal to the ideal $\langle T_1, \ldots, T_n\rangle$ given by

$$\left|\begin{array}{l} X_n^2 - X_{n-1} \\ \quad\vdots \\ X_2^2 - X_1 \\ X_1^2. \end{array}\right.$$

The R-basis of B corresponding to $\mathbf{V}$ is $(X_n^i)_{i<2^n}$; the basis corresponding to $\mathbf{T}$ is $M_{\mathbf{d}}$ (notation defined in the introduction), with $\mathbf{d} = (2, \ldots, 2)$. Besides, the change of basis does not use any arithmetic operation, since it amounts to rewrite the exponents i in base 2, and conversely.

Hence, we can apply our multivariate multiplication algorithm modulo $\langle \mathbf{T}\rangle$. Remark that the triangular set $\mathbf{T}$ satisfies the assumptions of Subsection 5.1 (for any R), with $d_1 = \cdots = d_n = d = 2$, $\delta_{\mathbf{d}} = 2^n$, $\lambda_1 = \cdots = \lambda_n = 1$ and $(x_1, x_2) = (0, 1)$. By Proposition 9, we deduce that the cost of a multiplication in B, and thus in A, is $O(2^n n\mathsf{M}(n))$. Since one can multiply univariate polynomials of degree 2^n using two multiplications in A, this gives the recurrence

$$\mathsf{M}(2^n) \;\le\; \mathsf{k}2^n n\mathsf{M}(n) \quad \text{and thus} \quad \mathsf{M}(d) \;\le\; \mathsf{k}' d\log(d)\mathsf{M}(\log(d))$$

for some constants k, k'. Unrolling the recursion 1, 2, ..., times, and taking $\mathsf{M}(n) \in O(n^2)$ to end the recursion, we obtain quasi-linear estimates of the form

$$\mathsf{M}(d) \in O(d\log(d)^3) \quad \text{or} \quad \mathsf{M}(d) \in O(d\log(d)^2\log(\log(d))^3), \quad \ldots$$

The main noteworthy feature of this multiplication algorithm is that no root of unity is present, though our multivariate evaluation-interpolation routine is somewhat similar to a multivariate Fourier Transform. In particular, the case when 2 is a zero-divisor in R requires no special treatment, contrary to [9].

5.4. Exponential generating series multiplication

We continue with a question somehow similar to the one in the previous subsection. Given two sequences $a_0, \ldots, a_d$ and $b_0, \ldots, b_d$ in R, we want to compute the sequence $c_0, \ldots, c_d$ such that

$$c_k = \sum_{i+j=k} \binom{k}{i} a_i b_j, \tag{11}$$

where the binomial coefficients are the coefficients of the expansion of $(1+X)^i$ in $\mathsf{R}[X]$. We discuss an application of this question in the next section.

The naive algorithm has cost $O(d^2)$. If $1, \ldots, d$ are units in R, the former equation takes the form

$$\sum_{i\le d} \frac{c_i}{i!} X^i \;=\; \sum_{i\le d} \frac{a_i}{i!} X^i \sum_{i\le d} \frac{b_i}{i!} X^i \mod X^{d+1}, \tag{12}$$

so we can achieve a cost $O(\mathsf{M}(d))$. Under some much milder assumptions on R, we are going to see how to achieve a similar cost through multivariate computations.

We will suppose that there exists a prime p such that for $a \in \mathbb{N}$, if $\gcd(a,p) = 1$, then a is a unit in R (this is the case e.g. for $\mathsf{R} = \mathbb{Z}/p^k\mathbb{Z}$). Let n be such that $d+1 \le p^n$, and introduce the triangular set $\mathbf{T} = (T_1, \ldots, T_n)$ defined by

$$\left|\begin{array}{l} X_n^p - pX_{n-1} \\ \quad \vdots \\ X_2^p - pX_1 \\ X_1^p. \end{array}\right.$$

In what follows, for $i \ge 0$, $(i_0, i_1, \ldots)$ denotes the sequence of its coefficients in base p; thus, for $i \le d$, only $i_0, \ldots, i_{n-1}$ can be non-zero. Besides, we let $f : \mathbb{N} \to \mathbb{N}$ be defined by $f(i) = i!/p^{v(i!)}$, where $v(i!)$ is the p-adic valuation of $i!$. In particular, $f(i)$ is a unit in R.

Proposition 10. *Let*

$$A = \sum_{i\le d} \frac{a_i}{f(i)} X_n^{i_0} \cdots X_1^{i_{n-1}}, \quad B = \sum_{i\le d} \frac{b_i}{f(i)} X_n^{i_0} \cdots X_1^{i_{n-1}}, \quad C = \sum_{i\le d} \frac{c_i}{f(i)} X_n^{i_0} \cdots X_1^{i_{n-1}}.$$

Then $C = AB \bmod \langle \mathbf{T} \rangle$.

Proof. Let $i, j \le d$, with $k = i + j \le d$. We start by the obvious remark that

$$\binom{k}{i} = \frac{f(k)}{f(i)f(j)}\, p^{v(\binom{k}{i})} \tag{13}$$

holds in R. Besides, the normal form of the product $\frac{a_i}{f(i)} X_n^{i_0} \cdots X_1^{i_{n-1}}$ by $\frac{b_j}{f(j)} X_n^{j_0} \cdots X_1^{j_{n-1}}$ modulo $\langle \mathbf{T} \rangle$ is

$$\frac{a_i b_j}{f(i)f(j)} p^{c(i,j)} X_n^{k_0} \cdots X_1^{k_{n-1}},$$

where $c_{i,j}$ is the number of carries held in the addition of i and j in base p. From [31, Eq. (1.6)], $c_{i,j}$ is exactly the valuation of the binomial coefficient $\binom{k}{i}$. Thus, by (13), the former product equals

$$\binom{k}{i} \frac{a_i b_j}{f(k)} X_n^{k_0} \cdots X_1^{k_{n-1}}.$$

Summing over all i, j gives our claim. □

As in the previous subsection, we can apply our multivariate multiplication algorithm modulo $\langle \mathbf{T} \rangle$. Remark that the triangular set $\mathbf{T}$ satisfies the assumptions of Subsection 5.1, with $d_1 = \cdots = d_n = d = p$, $\delta_{\mathbf{d}} = p^n$, $\lambda_1 = \cdots = \lambda_n = 1$ and $(x_1, \ldots, x_p) = (0, \ldots, p-1)$. Note as well that we can take $n \in O(\log_p(d))$, and that $\delta_{\mathbf{d}} = p^n \leq pd$.

By Proposition 9, we deduce that the cost of computing C, and thus all $c_0, \ldots, c_d$, is $O(d \log(d)\, \mathsf{M}(p)\, \mathsf{M}(p \log_p(d)))$. If p is fixed, we obtain the estimate $O(d \log(d) \mathsf{M}(\log(d)))$. This is not as good as the estimate $O(\mathsf{M}(d)) = O(d \log(d) \log\log(d))$ we obtained in characteristic zero, but quite close.

6. Application: computing with algebraic numbers

We finally present an application of the previous constructions to computation with algebraic numbers, and give timings of our implementation.

6.1. Presentation of the problem

Let k be a field and let f and g be monic polynomials in $k[T]$, of degrees m and n respectively. We are interested in computing their *composed sum* $h = f \oplus g$. This is the polynomial of degree $d = mn$ defined by

$$f \oplus g = \prod_{\alpha,\beta} (T - \alpha - \beta),$$

the product running over all the roots α of f and β of g, counted with multiplicities, in an algebraic closure $\overline{k}$ of k.

A natural approach consists in computing $h(T)$ as the resultant of $f(T-U)$ and $g(U)$ in U. However, the fastest algorithm for resultants [25] has a complexity of order $O^{\tilde{}}(d^{1.5})$ for $m = n$. To do better, Dvornicich and Traverso [12] suggested to compute the power sums

$$a_i = \sum_{f(\alpha)=0} \alpha^i, \quad b_i = \sum_{g(\beta)=0} \beta^i$$

of respectively f and g, and deduce the power sums c_i of h, by means of Equation (11). In [5], this approach is showed to take time $O(\mathsf{M}(d))$, over fields of characteristic zero or larger than d. Indeed, computing $(a_i)_{i \leq d}$ and $(b_i)_{i \leq d}$ can be done in $O(\mathsf{M}(d))$ operations, over any field, using Newton iteration for power series division [28]. Then, by our assumption on the characteristic, one can compute $(c_i)_{i \leq d}$ in quasi-linear time using Equation (12), for another $\mathsf{M}(d) + O(d)$ operations. Finally, knowing $(c_i)_{i \leq d}$, one can

then recover h in time $O(\mathsf{M}(d))$ as well, using fast exponential computation [8, 28, 34, 7]; this step relies as well on the assumption on the characteristic.

If k has positive characteristic less than d, two issues arise: Equation (12) makes no sense anymore and $(c_i)_{i\le d}$ are actually not enough to recover h. To our knowledge, no general solution better than the resultant method was known up to now (partial answers are in [5, 29] under restrictive conditions). We propose here a solution that works over finite fields, following an idea introduced in [18].

For simplicity, we consider only $k = \mathbb{F}_p$. Since our algorithm actually does computations over rings of the form $\mathbb{Z}/p^\alpha\mathbb{Z}$, measuring its complexity in $\mathbb{F}_p$-operations as we did up to now is not appropriate. Instead, we count bit operations. Thus, we let $\mathsf{M}_\mathbb{Z}$ be such that integers of bit-length ℓ can be multiplied using $\mathsf{M}_\mathbb{Z}(\ell)$ bit operations; quasi-linear estimates are known as well for $\mathsf{M}_\mathbb{Z}$, the best to date being Fürer's $\ell\log(\ell)2^{O(\log^*(\ell))}$ [14].

Proposition 11. *Given f and g, one can compute h using*

$$O\big((\mathsf{M}(d) \;+\; d\,\log(d)\,\mathsf{M}(p)\,\mathsf{M}(p\log_d(p)))\;\mathsf{N}(p,d)\big)$$

bit operations, with $\mathsf{N}(p,d) = O(\mathsf{M}_\mathbb{Z}(\log(p))\log(\log(p)) + \mathsf{M}_\mathbb{Z}(\log(d)))$.

After simplification, this cost is seen to be $O^{\tilde{}}(dp^2)$ bit operations. Also, if we consider p fixed, the cost becomes

$$O\big(\,(\mathsf{M}(d) \;+\; d\log(d)\mathsf{M}(\log(d)))\,\mathsf{M}_\mathbb{Z}(\log(d))\,\big),$$

that is, quasi-linear.

Proof. Let $\mathbb{Z}_p$ be the ring of p-adic integers and let F and G be monic lifts of f and g in $\mathbb{Z}_p[T]$, of degrees m and n. Defining $H = F \oplus G \in \mathbb{Z}_p[T]$, we have that $h = H \bmod p$. Let further $(A_i)_{i\ge0}$, $(B_i)_{i\ge0}$ and $(C_i)_{i\ge0}$ be the power sums of respectively F, G and H. For any $\alpha \ge 0$, the reductions $A_i \bmod p^\alpha$, $B_i \bmod p^\alpha$, and $C_i \bmod p^\alpha$ satisfy Equation (11), so we can apply the results of Subsection 5.4 to deduce $(C_i \bmod p^\alpha)_{i\le d}$ from $(A_i \bmod p^\alpha)_{i\le d}$ and $(B_i \bmod p^\alpha)_{i\le d}$.

Besides, taking $\alpha = \lfloor \log_p(d) \rfloor + 1$, it is proved in [6] that given $(C_i \bmod p^\alpha)_{i\le d}$, one can compute h in quasi-linear time $O(\mathsf{M}(d)\mathsf{M}_\mathbb{Z}(\log_p(d)))$ bit operations. Remark that this step is non trivial: recovering a polynomial of degree d from its Newton sums requires divisions by $1, \ldots, d$, and not all these numbers are units in small characteristic.

In the algorithm, the function Lift simply lifts its argument from $\mathbb{F}_p[T] = \mathbb{Z}/p\mathbb{Z}[T]$ to $\mathbb{Z}/p^\alpha\mathbb{Z}[T]$; the function PowerSums computes the first d power sums of its arguments by the algorithm of [28]. Step 7 applies the algorithm of Subsection 5.4, and the last step uses the algorithm presented in [6] to recover h.

Our choice of α implies that $\log(p^\alpha) = O(\log(d))$. Thus, operations $(+, \times)$ modulo p^α take $O(\mathsf{M}_\mathbb{Z}(\log(d)))$ bit operations [15, Chapter 9]. Using Newton iteration, inversions modulo p^α take $\mathsf{N}(p,d) = O(\mathsf{M}_\mathbb{Z}(\log(p))\log(\log(p)) + \mathsf{M}_\mathbb{Z}(\log(d)))$ bit operations, where the first term stands for the cost computing the inverse modulo p, and the second one for lifting it modulo p^α.

The cost of computing $(A_i)_{i\le d}$ and $(B_i)_{i\le d}$ is $O(\mathsf{M}(d))$ operations modulo p^α; this dominates the cost of recovering h. The remaining cost is that of computing $(C_i)_{i\le d}$, which is reported in Subsection 5.4 in terms of numbers of operations modulo p^α. The previous estimate on $\mathsf{N}(p,d)$ concludes the proof. □

ComposedSum(f, g)

1. $d \leftarrow \deg(f)\deg(g)$
2. $\alpha \leftarrow \lfloor \log_p(d) \rfloor + 1$
3. $F \leftarrow \mathsf{Lift}(f, \alpha)$
4. $(A_i)_{i \le d} \leftarrow \mathsf{PowerSums}(F, d)$
5. $G \leftarrow \mathsf{Lift}(g, \alpha)$
6. $(B_i)_{i \le d} \leftarrow \mathsf{PowerSums}(G, d)$
7. $(C_i)_{i \le d} \leftarrow \mathsf{ExponentialGeneratingSeriesMultiplication}(A, B)$
8. return $\mathsf{PowerSumsToPolynomial}(C)$

Fig. 10. Composed sum in small characteristic.

Remark that a more general question is to compute

$$\prod_{\alpha,\beta}(T - q(\alpha, \beta)),$$

where q is any polynomial in $k[X, Y]$, and where the product is taken as before over all roots α of f and β of g. Here, we dealt with $q = X + Y$; the case $q = XY$ is actually simpler, and is discussed in [12, 5, 6]. However, in the general case, even in characteristic zero, no quasi-linear time algorithm is known as of now.

6.2. Experimental results

We implemented the composed sum algorithm over $\mathbb{F}_2$ (*i.e.*, $p = 2$ here). We used the NTL C++ package as a basis [30]. Since NTL does not implement bivariate resultants, we also used Magma [4] for comparison with the resultant method. All timings are obtained on an AMD Athlon 64 with 5GB of RAM.

Figure 11 gives detailed timings for our algorithm; each colored area gives the time of one of the main tasks. The less costly step is the first, the conversion from the original polynomials to their Newton sums. Then, we give the time needed to compute all the power series roots needed for our multiplication algorithm, followed by the evaluation-interpolation process itself; finally, we give the time necessary to recover h from its power sums. Altogether, the practical behavior of our algorithm matches the quasi-linear complexity estimates. The steps we observe correspond to the increase in the number of variables in our multivariate polynomials, and are the analogues of the steps observed in classical FFT.

Figure 12 gives timings obtained in Magma, using the built-in resultant function, on the same set of problems as above. As predicted by the complexity analysis, the results are significantly slower (about two orders of magnitude for the larger problems).

7. Conclusion

Several questions remain open after this work. First, it should be stressed that for triangular sets in few variables (say up to 4), the approach discussed here is not expected to be competitive with the general purpose algorithms: our advantages appear in structured situations. Indeed, the most challenging open problem remains how to unconditionally get rid of all exponential factors in multiplication algorithms for triangular sets.

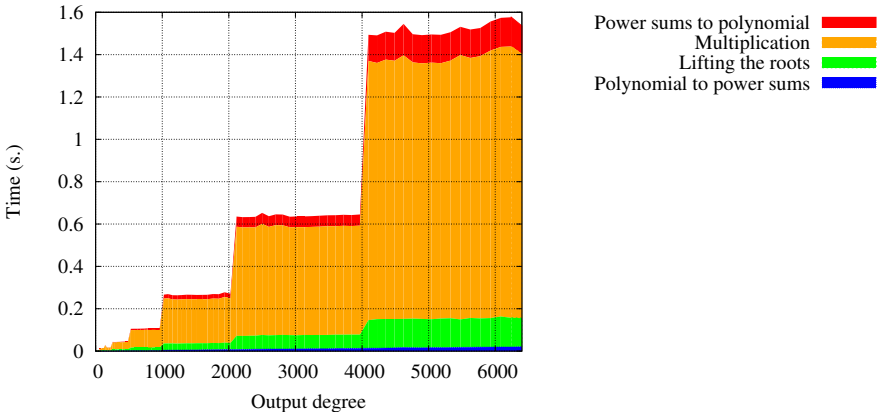

Fig. 11. Detailed timings for our algorithm.

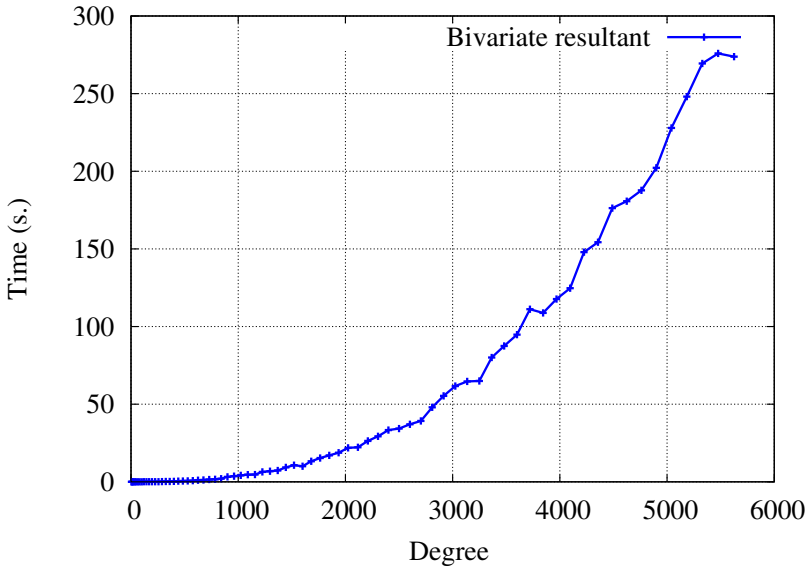

Fig. 12. Timings in magma.

More immediate questions may be the following: at the fine tuning level, adapting the idea of the Truncated Fourier Transform [33] should enable us to reduce the step effect in the timings of the previous section. Besides, it will be worthwhile to investigate what other applications can be dealt with using the "homotopy multiplication" model, such as the product of matrices with entries defined modulo a triangular set, or further tasks such as modular inversion or modular composition.

References

[1] I. Abdeljaouad, S. Orange, G. Renault, and A. Valibouze. Computation of the decomposition group of a triangular ideal. *Applicable Algebra in Engineering Com-*

munication and Computing, 15(3-4):279–294, 2004.

[2] A. V. Aho, K. Steiglitz, and J. D. Ullman. Evaluating polynomials at fixed sets of points. *SIAM J. Comp.*, 4(4):533–539, 1975.

[3] P. Aubry and A. Valibouze. Using Galois ideals for computing relative resolvents. *J. Symb. Comp.*, 30(6):635–651, 2000.

[4] W. Bosma, J. Cannon, and C. Playoust. The Magma algebra system. I. The user language. *J. Symb. Comp.*, 24(3-4):235–265, 1997.

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

[6] A. Bostan, L. González-Vega, H. Perdry, and É. Schost. From Newton sums to coefficients: complexity issues in characteristic p. In *MEGA'05*, 2005.

[7] A. Bostan and É. Schost. A simple and fast algorithm for computing exponentials of power series. Available at `http://algo.inria.fr/bostan/`, 2008.

[8] R. P. Brent. Multiple-precision zero-finding methods and the complexity of elementary function evaluation. In *Analytic computational complexity*, pages 151–176. Academic Press, 1976.

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

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

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

[12] R. Dvornicich and C. Traverso. Newton symmetric functions and the arithmetic of algebraically closed fields. In *AAECC-5*, volume 356 of *LNCS*, pages 216–224. Springer, 1989.

[13] M. Foursov and M. Moreno Maza. On computer-assisted classification of coupled integrable equations. *J. Symb. Comp.*, 33:647–660, 2002.

[14] M. Fürer. Faster integer multiplication. In *39th Annual ACM Symp. Theory Comp.*, pages 57–66. ACM, 2007.

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

[16] P. Gaudry and É. Schost. Construction of secure random curves of genus 2 over prime fields. In *Eurocrypt'04*, pages 239–256. Springer, 2004.

[17] P. Gaudry, É. Schost, and N. Thiéry. Evaluation properties of symmetric polynomials. *International Journal of Algebra and Computation*, 16(3):505–523, 2006.

[18] L. González-Vega and H. Perdry. Computing with Newton sums in small characteristic. In *EACA'04*, 2004.

[19] I. A. Kogan and M. Moreno Maza. Computation of canonical forms for ternary cubics. In *ISSAC'02*, pages 151–160. ACM, 2002.

[20] L. Langemyr. Algorithms for a multiple algebraic extension. In *Effective methods in algebraic geometry)*, volume 94 of *Progr. Math.*, pages 235–248. Birkhäuser, 1991.

[21] X. Li, M. Moreno Maza, R. Rasheed, and É Schost. High-performance symbolic computation in a hybrid compiled-interpreted programming environment. In *ICCSA'08*, pages 331–341. IEEE, 2008.

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

[23] M. van Hoeij and M. Monagan. A modular GCD algorithm over number fields presented with multiple extensions. In *ISSAC'02*, pages 109–116. ACM, 2002.
[24] V. Y. Pan. Simple multivariate polynomial multiplication. *J. Symb. Comp.*, 18(3):183–186, 1994.
[25] D. Reischert. Asymptotically fast computation of subresultants. In *ISSAC'97*, pages 233–240. ACM, 1997.
[26] G. Renault and K. Yokoyama. A modular algorithm for computing the splitting field of a polynomial. In *Algorithmic Number Theory, ANTS VII*, number 4076 in LNCS, pages 124–140. Springer, 2006.
[27] N. Rennert and A. Valibouze. Calcul de résolvantes avec les modules de Cauchy. *Experimental Mathematics*, 8(4):351–366, 1999.
[28] A. Schönhage. The fundamental theorem of algebra in terms of computational complexity. Technical report, Univ. Tübingen, 1982.
[29] É. Schost. Multivariate power series multiplication. In *ISSAC'05*, pages 293–300. ACM, 2005.
[30] V. Shoup. NTL: A library for doing number theory. `http://www.shoup.net`.
[31] A. Straub, T. Amdeberhan, and V. H. Moll. The p-adic valuation of k-central binomial coefficient, 2008.
[32] B. Sturmfels. *Algorithms in invariant theory.* Texts and Monographs in Symbolic Computation. Springer-Verlag, 1993.
[33] J. van der Hoeven. The Truncated Fourier Transform and applications. In *ISSAC'04*, pages 290–296. ACM, 2004.
[34] J. van der Hoeven. Newton's method and FFT trading. Technical Report 2006-17, Univ. Paris-Sud, 2006. Submitted to J. Symb. Comp.