Multivariate Power Series Multiplication

Éric Schost, LIX, École polytechnique
91128 Palaiseau, France
Eric.Schost@polytechnique.fr

ABSTRACT

We study the multiplication of multivariate power series. We show that over large enough fields, the bilinear complexity of the product modulo a monomial ideal M is bounded by the product of the regularity of M by the degree of M. In some special cases, such as partial degree truncation, this estimate carries over to total complexity. This leads to complexity improvements for some basic algorithms with algebraic numbers, and some polynomial system solving algorithms.

Categories and Subject Descriptors

I.1.2 [**Computing Methodologies**]: Symbolic and Algebraic Manipulation—*Algebraic Algorithms*

General Terms

Algorithms, Experimentation, Theory

Keywords

Power series, multiplication

1. INTRODUCTION

We investigate the complexity of the multiplication of multivariate power series. We will work modulo a 0-dimensional monomial ideal M in $k[X_1, \ldots, X_n]$, where k is a field. This is no loss of generality, since the k-algebras $k[X_1, \ldots, X_n]/M$ and $k[[X_1, \ldots, X_n]]/M$ are isomorphic.

Complexity conventions. Once k is fixed, M is specified by the number n of variables and the degrees of some generators. Thus, our complexity estimates are functions defined on some subsets of $\mathbb{N}^{\mathbb{N}}$, which may vary, depending on M being arbitrary, or restricted to some special patterns, such as total or partial degree truncation (unless specified otherwise, the number of variables is not fixed). These estimates will be expressed directly in terms of the degrees of the generators of M, or in terms of some quantities attached to the algebra $k[X_1, \ldots, X_n]/M$: its dimension as a k-vector space, also called the *degree* $\deg_M$ of M, and possibly the *regularity* reg_M of M, which is defined later on.

If V is a set, and f and g are maps $V \to \mathbb{R}_{>0}$, we say that $f \in O(g)$ if there exists $C \in \mathbb{R}$ such that $f(v) \leq Cg(v)$ holds for all v in V. To make expressions involving nested logarithms well-defined in all cases, we write $\lg(x) = \max(1, \log_2(\max(x, 1)))$. The notation $f \in O_{\lg}(g)$ indicates the omission of logarithmic factors, that is, that there exists a constant α such that f is in $O(g \lg^{\alpha}(g))$.

Previous work. In all that follows, we will distinguish the *bilinear complexity*, which estimates the number of algebra multiplications, and is also called *rank*, and the *total complexity*, which counts linear operations as well (precise definitions are given in the next section). We start by reviewing the previous results in one variable, *i.e.* when $n = 1$.

• The rank of $k[X_1]/(X_1^d)$ is at least $2d - 1$ [33, 16], and if k has cardinality at least $2d - 1$, this is also an upper bound. Hence, in this case, the rank of power series multiplication equals that of polynomial multiplication.

• For some specific multiplication algorithms (with complexity more than linear, such as Karatsuba or Toom-Cook algorithms), truncated power series multiplication is faster than polynomial multiplication by a constant factor, regarding total and bilinear complexity [26, 18].

When $n \geq 2$, the situation is more complex, and results might depend on the shape of the quotient algebra. We will in the first place focus on the "general" case, for which no assumption is made on the ideal M. We will also mention two important special cases, *partial* and *total* degree truncation.

• Truncating in partial degrees $(d_1, \ldots, d_n)$ amounts to compute modulo the ideal $(X_1^{d_1}, \ldots, X_n^{d_n})$ of degree $d_1 \cdots d_n$; the support of such power series is a rectangular parallelotope. Section 4 gives applications of this truncation pattern.

• Truncating in total degree d amounts to compute modulo the ideal generated by all monomials of total degree d, which has degree $\binom{d+n-1}{n}$; the support of such power series is a simplex. This truncation pattern is used in various versions of Newton-Hensel lifting algorithms, see [23] for references.

In the case $n = 2$, the following results are known for these two special truncation patterns. For truncation in partial degrees (d, d), Bläser [5] gave a lower bound for the rank of $\frac{7}{3}d^2 - O(d)$, and an upper bound of $3d^2 + O(d)$ (for base fields of characteristic 0, containing all roots of unity), improving results of Schönhage [32]. For truncation in total degree d, Bläser [5] gave a lower bound for the rank of $\frac{5}{4}d^2 - O(d)$, and an upper bound of $\frac{3}{2}d^2 + O(d)$ (with the same restrictions on k), again improving results of [32].

Let now the number of variables be arbitrary. From now on, we will denote by T the monomial basis of $k[X_1, \ldots, X_n]/M$. The direct approach to power series multiplication consists in expanding products. Then, obtaining the coefficient of a single monomial $X_1^{t_1} \cdots X_n^{t_n} \in \mathsf{T}$ in the output requires to perform $2(t_1 + 1) \cdots (t_n + 1)$ operations. For partial degree truncation, the total cost is then approximately $\frac{1}{2^n}(\deg_M)^2$; for total degree truncation, the total cost is about $\frac{1}{n!}(\deg_M)^2$. In any case, the cost is at most quadratic in $\deg_M$.

A second approach is to perform the product as that of polynomials, and discard unwanted coefficients: the penalty induced by computing more terms than necessary can be compensated by the use of fast multiplication algorithms. To give complexity estimates for this approach, let V_M denote the cardinality of the set of terms $\mathsf{T} \times \mathsf{T}$: this is the number of monomials that appear when power series are multiplied as polynomials. Then over fields of characteristic 0, using Emiris and Pan's sparse multiplication algorithm [15] yields a total complexity in $O_{\lg}(nV_M)$.

At worst, V_M grows quadratically with $\deg_M$. However, better estimates hold in many cases. For instance, for partial degree truncation, $V_M \le 2^n \deg_M$; thus in this case, the product modulo M can be done with a total complexity in $O_{\lg}(2^n \deg_M)$. Note however that this particular result on partial degree truncation actually holds more generally (it suffices that k is a ring), using Kronecker's substitution [14].

Finally, fast algorithms are known for total degree truncation. Let $M \subset k[X_1, \ldots, X_n]$ be the ideal generated by all monomials of total degree d. In [23], with Lecerf, we gave sharp estimates for the product modulo M in characteristic 0: the total complexity is in $O(\deg_M \lg^3(\deg_M) \lg\lg(\deg_M))$, and the bilinear complexity is at most $\deg_M \lg(\deg_M)$. An algorithm of Griewank's using similar ideas was previously given in [4]; it has the same bilinear complexity, but the total complexity relies on results for evaluation / interpolation that are not detailed there.

In [19], van der Hoeven extended the result of [23] to weighted total degree; he gave better estimates in [20], but the algorithm of [20, Section 5] does not seem to work as claimed, leaving the correctness of these results pending. Finally, Bläser gave in [5] the lower bound $\deg_M(3 - \varepsilon(\deg_M))$ for the rank, for fixed d and $n \to \infty$, with $\varepsilon(\deg_M) \to 0$.

Our contribution. We obtain two kinds of results: on the rank of the multiplication modulo an arbitrary 0-dimensional monomial ideal, and on the total complexity in some specific cases. The core of these results is to prove that power series modulo M can be *approximately* multiplied with a number of multiplications that equals the degree of M, using the idea of approximate algorithm introduced in [2, 1, 3].

In one variable, this approach is well-known, and can be easily illustrated for multiplication modulo X_1^2 [3, 30]. If A and $B \in k[X_1]$ have degree at most 1, then computing AB modulo $X_1^2 - \varepsilon^2$ (ε is a new variable) and letting $\varepsilon = 0$ in the result yields AB modulo X_1^2. Now, the product AB modulo $X_1^2 - \varepsilon^2$ can be obtained by *(i)* evaluating A and B at $\pm\varepsilon$, *(ii)* multiplying the values, *(iii)* interpolating the result at $\pm\varepsilon$. Thus, this algorithm requires two bilinear multiplications of degree 1 polynomials in ε, plus some linear operations. Generalizing this to an arbitrary ideal M will show that the rank of multiplication modulo M is essentially bounded by the product of the regularity of M by the degree of M.

One would like a similar result to hold for total complexity; to obtain that extension, the missing elements are fast (linear-time) algorithms for some multivariate evaluation / interpolation questions (which generalize evaluation / interpolation at $\pm\varepsilon$ used above). Even though no such result is known for general M, we will however obtain suitable algorithms in the specific case of partial degree truncation.

Acknowledgments. I thank Alin Bostan, Marc Giusti, Andreas Griewank, Joris van der Hoeven, Grégoire Lecerf and Bruno Salvy for fruitful discussions.

2. MAIN RESULTS

Basic definitions. Let k be a field. Given $n \ge 1$, we write $k[X_1, \ldots, X_n]$ for the ring of polynomials in n variables over k. A *term* is a product $X_1^{\alpha_1} \cdots X_n^{\alpha_n}$, with all $\alpha_i \ge 0$; if S is a finite set of terms, $\mathrm{Span}(\mathsf{S}) \subset k[X_1, \ldots, X_n]$ is the k-vector space of all sums $\sum_{t \in \mathsf{S}} c_t t$, with all c_t in k.

Let M be a 0-dimensional monomial ideal in $k[X_1, \ldots, X_n]$, so that $Q = k[X_1, \ldots, X_n]/M$ is a finite-dimensional k-vector space. Let T be the set of terms not in M: T is the monomial basis of Q, so, as a k-vector space, Q is isomorphic to $\mathrm{Span}(\mathsf{T})$. The *degree* of M, written $\deg_M$, is the dimension of $k[X_1, \ldots, X_n]/M$ over k, that is, $\deg_M = |\mathsf{T}|$. The *regularity* of M, written reg_M, is the first degree d for which the degree d component of the graded algebra Q is zero. Any term in T has a total degree less than reg_M.

For $1 \le i \le n$, let d_i be the least integer such that $X_i^{d_i}$ belongs to M (d_i is well-defined, since M has dimension 0). We will work under the following assumption.

For all i, d_i is less than or equal to the cardinality of k.

For $1 \le i \le n$ and $0 \le j < d_i$, let $a_{i,j}$ be in k, with the requirement that $a_{i,j} \ne a_{i,j'}$ for all $j \ne j'$. We will write

$$\mathsf{A}_1 = [a_{1,0}, \ldots, a_{1,d_1-1}], \ldots, \mathsf{A}_n = [a_{n,0}, \ldots, a_{n,d_n-1}].$$

Then, to the set T we associate $\mathsf{A}_\mathsf{T} \subset \mathsf{A}_1 \times \cdots \times \mathsf{A}_n$ given by

$$\mathsf{A}_\mathsf{T} = [(a_{1,c_1}, \ldots, a_{n,c_n}) \mid X_1^{c_1} \cdots X_n^{c_n} \in \mathsf{T}];$$

for definiteness, we may suppose that T is ordered in some way, so that A_T inherits its order. The set of points A_T is a parameter for our algorithms: in our computational model, any value depending on these points only will be available for free (the cost of computing the relevant such values could be estimated as well, using a more involved model).

To illustrate this last definition, take for instance $a_{i,j} = j$, if the characteristic of k allows this. Then, the set A_T equals

$$[(c_1, \ldots, c_n) \mid X_1^{c_1} \cdots X_n^{c_n} \in \mathsf{T}],$$

so it equals the set of exponents of the terms in T. In any case, we have the equality $|\mathsf{A}_\mathsf{T}| = |\mathsf{T}| = \deg_M$, and the set A_T has the same "shape" as the above set of exponents.

Model of computation. We will estimate both the rank (which counts bilinear multiplications) and the total complexity of the multiplication in Q. The former quantity is defined in the *bilinear algorithm* computational model, and the later using *linear straight-line programs.*

Let R be a ring, and S be an R-algebra, which is also a free R-module. The inputs and outputs of our algorithms are given by means of their coordinates on an R-basis of S (all cases considered below come with a natural choice of a basis, which we will not mention explicitly). A *bilinear algorithm* of length ℓ for the multiplication in S is the data of 2ℓ R-linear forms $f_1, \ldots, f_\ell$ and $h_1, \ldots, h_\ell$ over S, and ℓ elements $w_1, \ldots, w_\ell$ in S, such that the equality

$$AB = \sum_{i=1}^{\ell} f_i(A) h_i(B) w_i$$

holds for all A and B in S. The *rank* of the multiplication map in S is the smallest length of such a bilinear algorithm. We will further say that a bilinear algorithm has *total complexity* L if $\ell \le L$, and if the linear map

$$A \mapsto [f_1(A), \ldots, f_\ell(A)], \quad B \mapsto [h_1(B), \ldots, h_\ell(B)],$$
$$[\lambda_1, \ldots, \lambda_\ell] \mapsto \textstyle\sum_i \lambda_i w_i$$

can be computed by a R-linear straight-line program of size at most L (see [11, Chapter 13] for the definition).

Multiplication, evaluation and interpolation. Our algorithms rely on polynomial multiplication, evaluation and interpolation. We now give the corresponding notation.

We let $\mathsf{M}_{\mathrm{Bil}}$ and M be two maps $\mathbb{N} \to \mathbb{N}$ such that for any ring R and any integer s, the product of two polynomials of degree at most s in $R[T]$ can be computed by a bilinear algorithm of length $\mathsf{M}_{\mathrm{Bil}}(s)$ and total complexity $\mathsf{M}(s)$. We also impose some standard growth conditions on these maps, see for instance [17, Chapter 8]. It follows from [12] that there exists a constant K such that one can take

$$\mathsf{M}_{\mathrm{Bil}}(s) \le Ks \lg(s), \quad \mathsf{M}(s) \le Ks \lg(s) \lg\lg(s). \qquad (1)$$

Next, we consider some questions of multivariate evaluation and interpolation, for the family of points A_T defined above. We write $\mathsf{Eval}(\mathsf{A}_\mathsf{T})$ for the minimal size of a linear straight-line program that computes the evaluation map

$$\begin{array}{rcl} \mathrm{Span}(\mathsf{T}) & \to & k^{\mathsf{A}_\mathsf{T}} \\ P & \mapsto & [P(a) \mid a \in \mathsf{A}_\mathsf{T}], \end{array}$$

and $\mathsf{Interp}(\mathsf{A}_\mathsf{T})$ for the minimal size of a linear straight-line program that computes the inverse map (Section 3 gives the proof of invertibility). In our computational model, naive estimates for $\mathsf{Eval}(\mathsf{A}_\mathsf{T})$ and $\mathsf{Interp}(\mathsf{A}_\mathsf{T})$ are *quadratic* in $\deg_M$.

Main results: the general case. We can now give our main result (proved in Section 3) on the rank and the total complexity of multiplication modulo an arbitrary monomial ideal. The algorithm is parametrized by the choice of points A_T: while the rank estimate does not depend on it, the total complexity does, since evaluation and interpolation on A_T appear as "linear subroutines" of the underlying algorithm.

THEOREM 1. *The rank of the multiplication in Q admits the upper bound $\mathsf{M}_{\mathrm{Bil}}(\mathrm{reg}_M) \deg_M$. The total complexity is in $O\left((\mathsf{Eval}(\mathsf{A}_\mathsf{T}) + \mathsf{Interp}(\mathsf{A}_\mathsf{T}))\mathrm{reg}_M + \mathsf{M}(\mathrm{reg}_M) \deg_M\right)$.*

The best results one could obtain with these techniques would be total complexity estimates in $O_{\lg}(\mathrm{reg}_M \deg_M)$. We already know that $\mathsf{M}(\mathrm{reg}_M)$ is in $O_{\lg}(\mathrm{reg}_M)$. Thus, the main missing elements are linear-type estimates for evaluation and interpolation, that would hold for any choice of M. For the moment, I am not aware of such results.

It should also be stressed that Theorem 1 does not improve the previous results in all situations. Its worst case occurs for $n = 1$: even with fast polynomial arithmetic, the algorithm of Theorem 1 then requires quadratic time, versus an almost linear complexity for the naive algorithm. Our algorithm is efficient when the number of variables is large; we will see that in a sense, it complements the previous approaches.

Partial degree truncation. More complete results are obtained for multiplication truncated in *partial degree*, since in this case, sharp estimates on $\mathsf{Eval}(\mathsf{A}_\mathsf{T})$ and $\mathsf{Interp}(\mathsf{A}_\mathsf{T})$ are known in terms of *univariate* evaluation / interpolation. We will prove the following corollaries in Section 4 (following the convention given before, the constants in the big-O notation are independent of the number of variables).

COROLLARY 1. *Let M be the ideal $(X_1^{d_1}, \ldots, X_n^{d_n})$. Then the total complexity of the multiplication modulo M is in*

$$O\left(\mathrm{reg}_M \deg_M \sum_i \frac{\mathsf{M}(d_i) \lg(d_i)}{d_i} + \mathsf{M}(\mathrm{reg}_M) \deg_M\right).$$

Using $\mathrm{reg}_M \le \sum_i d_i$ and $\deg_M = \prod_i d_i$, and Equations (1), the total complexity is thus in the complexity class

$$O_{\lg}(\mathrm{reg}_M \deg_M) \subset O_{\lg}\left(\Big(\sum_i d_i\Big)\Big(\prod_i d_i\Big)\right).$$

As mentioned above, this algorithm is useful when the number of variables is large with respect to the degrees. Combining it with the algorithm using Kronecker's substitution, we will get the following corollary. It shows that multiplication truncated in partial degree has an almost linear complexity in $\deg_M$.

COROLLARY 2. *For any $\varepsilon > 0$, the total complexity of the product modulo $(X_1^{d_1}, \ldots, X_n^{d_n})$ is in $O((\deg_M)^{1+\varepsilon})$.*

Of course, estimates in $O_{\lg}(\deg_M)$ would be desirable, but our methods do not seem to give such results.

Applications. We will conclude this paper by two applications. First, we show how to speed up algorithms for the addition of algebraic numbers: over fields of small characteristic, the algorithm of [7] uses multivariate power series multiplication truncated in partial degree. The second application is the resolution of polynomial systems: Lecerf's deflation algorithm for multiple roots [21] requires multiplications modulo suitable "gradients of ideals", which are here similar to multiplications truncated in partial degree. Our algorithms improve the known results for both questions.

3. PROOF OF THEOREM 1

We now prove Theorem 1, still using the notation of Section 2. We first recall a deformation result of Macaulay's [24]. Let $g_1, \dots, g_R$ be monomial generators of M such that g_i does not divide g_j, for $i \neq j$. For $1 \le r \le R$, we write

$$g_r = X_1^{\delta_{1,r}} \cdots X_n^{\delta_{n,r}}.$$

Let $\mathsf{A}_1, \dots, \mathsf{A}_n$ and A_T be the sets of points introduced in the previous section, and let ε be a new indeterminate over k. Then, for $1 \le r \le R$, we define the polynomial

$$G_r = \prod_{i=1}^{n} \prod_{j=0}^{\delta_{i,r}-1} (X_i - \varepsilon a_{i,j}),$$

so that G_r is in $k[\varepsilon, X_1, \dots, X_n] \subset k(\varepsilon)[X_1, \dots, X_n]$, and $g_r = G_r(0, X_1, \dots, X_n)$. Up to the use of the parameter ε, this definition is due to [24]. Together with generalizations, it is studied in more details in [25], who gives references to other occurrences in the literature.

Let us fix a monomial order $<$ that refines the total degree, in both $k[X_1, \dots, X_n]$ and $k(\varepsilon)[X_1, \dots, X_n]$: in particular, for $1 \le r \le R$, the leading term of G_r is g_r. Then the following results are proved in [25] (see also [29]).

PROPOSITION 1. *The polynomials $G_r(1, X_1, \dots, X_n)$, $1 \le r \le R$, form the Gröbner basis in $k[X_1, \dots, X_n]$ of the ideal of the set of points $\mathsf{A}_\mathsf{T} \subset \mathbb{A}^n(k)$. The polynomials $G_1, \dots, G_R$ form the Gröbner basis in $k(\varepsilon)[X_1, \dots, X_n]$ of the ideal of the set of points $[\varepsilon a \mid a \in \mathsf{A}_\mathsf{T}] \subset \mathbb{A}^n(k(\varepsilon))$.*

The first part of the proposition shows that the map

$$\begin{array}{rcl} k^{\mathsf{A}_\mathsf{T}} & \to & \mathrm{Span}(\mathsf{T}) \\ [P(a) \mid a \in \mathsf{A}_\mathsf{T}] & \mapsto & P \end{array}$$

is well-defined, as claimed previously. The next intermediate result is to relate reductions modulo $g_1, \dots, g_R$ and $G_1, \dots, G_R$. For A in $k(\varepsilon)[X_1, \dots, X_n]$, we write $R_g(A)$ and $R_G(A)$ for its normal forms modulo respectively $g_1, \dots, g_R$ and $G_1, \dots, G_R$. For A in $k[\varepsilon, X_1, \dots, X_n]$, we write $\deg A$ for its total degree in $\varepsilon, X_1, \dots, X_n$.

LEMMA 1. *Let $A \in k[X_1, \dots, X_n] \subset k(\varepsilon)[X_1, \dots, X_n]$. Then $R_G(A)$ is in $k[\varepsilon, X_1, \dots, X_n] \subset k(\varepsilon)[X_1, \dots, X_n]$, its specialization at $\varepsilon = 0$ is $R_g(A)$ and $\deg R_G(A) \le \deg A$.*

PROOF. It suffices to prove the result for terms; we do it by induction for the order $<$. The property holds for all terms in T; let thus $t \notin \mathsf{T}$ be a term, such that the property holds for all terms less than t for the order $<$.

The remainder $R_g(t)$ is 0; let δ be a term and $1 \le r \le R$ be such that $t = \delta g_r$, and let $P = t - \delta G_r$. Then, P is in $k[\varepsilon, X_1, \dots, X_n]$, and is less than t for the order $<$. Besides, $R_G(P) = R_G(t)$ and $\deg P = \deg t$ (since G_r is homogeneous in $\varepsilon, X_1, \dots, X_n$).

Write $P = \sum_{t' < t} p_{t'} t'$ with all $p_{t'}$ in $k[\varepsilon]$; then $R_G(P) = \sum_{t' < t} p_{t'} R_G(t')$. By the induction assumption, $R_G(P)$ is in $k[\varepsilon, X_1, \dots, X_n]$ and $\deg R_G(P) \le \max_{t'}\{\deg p_{t'} + \deg R_G(t')\}$ is upper bounded by $\max_{t'}\{\deg p_{t'} + \deg t'\} = \deg P$. So the first and last properties hold for t.

To conclude, note that $g_r - G_r$ is in $\varepsilon k[X_1, \dots, X_n]$. Thus, $P = \delta(g_r - G_r)$ is in $\varepsilon k[X_1, \dots, X_n]$ as well, and it is then also the case for $R_G(P)$. Letting $\varepsilon = 0$ in $R_G(P) = R_G(t)$ yields $0 = R_g(t)$, concluding the proof. $\square$

We can then describe our multiplication algorithm. Given A and B in $\mathrm{Span}(\mathsf{T}) \subset k[X_1, \dots, X_n]$, it outputs $C = R_g(AB)$. Let $C_\varepsilon = R_G(AB)$. By Lemma 1, C_ε can be written

$$C_\varepsilon = \sum_{i \le \mu} C_{i,0} + C_{i,1}\varepsilon + \cdots + C_{i,e_i}\varepsilon^{e_i},$$

for some μ and $e_i \ge 0$, with $C_{i,j}$ homogeneous of degree i in $k[X_1, \dots, X_n]$. The same lemma shows that $C = \sum_{i \le \mu} C_{i,0}$ and that $i + e_i \le \deg AB < 2\,\mathrm{reg}_M$ for all i, so $\mu < 2\,\mathrm{reg}_M$.

Let us evaluate A and B on all points εa, for $a \in \mathsf{A}_\mathsf{T}$, and multiply these values. The last part of Proposition 1 shows the equality $C_\varepsilon(\varepsilon a) = A(\varepsilon a) B(\varepsilon a)$ for all $a \in \mathsf{A}_\mathsf{T}$.

For $i < 2\,\mathrm{reg}_M$, let $\Gamma_i = \sum_{j+\ell=i} C_{j,\ell} \in k[X_1, \dots, X_n]$; then $C_{i,0}$ is the homogeneous component of degree i of Γ_i. Since $C_\varepsilon(\varepsilon a) = \sum_i \Gamma_i(a)\varepsilon^i$ is known for all $a \in \mathsf{A}_\mathsf{T}$, Γ_i is obtained by interpolation on A_T. This yields the following algorithm.

Multiplication modulo $g_1, \dots, g_R$.

Input: A and B in $\mathrm{Span}(\mathsf{T})$.

Output: The product AB modulo $g_1, \dots, g_R$.

1. Let $v_A = [A(\varepsilon a) \mid a \in \mathsf{A}_\mathsf{T}]$ and $v_B = [B(\varepsilon a) \mid a \in \mathsf{A}_\mathsf{T}]$.
2. Compute the pairwise product v_C of v_A and v_B.
3. For $i < 2\,\mathrm{reg}_M$, compute Γ_i by interpolating on A_T the degree-i coefficients of the entries of v_C. Let $C_{i,0}$ be the homogeneous component of degree i of Γ_i. Return $\sum_{i < 2\,\mathrm{reg}_M} C_{i,0}$.

Analysis. Evaluating the polynomial $A \in k[X_1, \dots, X_n]$ at $[\varepsilon a \mid a \in \mathsf{A}_\mathsf{T}]$ is done as follows. Let $A = A_0 + \cdots + A_\delta$, with A_i homogeneous of degree i; then $A(\varepsilon a) = \sum_{i \le \delta} A_i(a)\varepsilon^i$. Thus, v_A is obtained by evaluating all polynomials A_i on A_T. Since A is in $\mathrm{Span}(\mathsf{T})$, δ is less than reg_M, and all A_i are in $\mathrm{Span}(\mathsf{T})$ as well; thus, v_A is obtained by less than reg_M evaluations on A_T. The same holds for v_B. The entries of v_A and v_B have degree in ε less than reg_M, so all entries of v_C are obtained for $\deg_M$ further multiplications in $k[\varepsilon]$ in degree less than reg_M. Finally, C is recovered by performing at most $2\,\mathrm{reg}_M$ interpolations on A_T.

Evaluations and interpolations do not contribute to the bilinear complexity, which is thus $\deg_M$ times the bilinear cost of univariate polynomial multiplication in degree less than reg_M. This gives the first part of Theorem 1. To get the total complexity, one adds the costs of evaluation and interpolation, to the total cost of the polynomial multiplications. This concludes the proof.

4. SPECIAL CASES AND APPLICATIONS

Giving precise total complexity estimates requires results on the functions Eval and Interp, that is, on multivariate evaluation and interpolation. I do not know sharp results for this in general; however, it is easy to give estimates in the particular case of *partial degree truncation*. This section is devoted to present a standard algorithm for this task, and some applications and extensions. We first review previous results on *univariate* evaluation and interpolation.

Let B be d pairwise distinct points in k. We will write $\mathsf{EvalUni}(\mathsf{B})$ for the minimal size of a linear straight-line program that computes the linear map $P \mapsto [P(b) \mid b \in \mathsf{B}]$, where $P \in k[T]$ has degree less than d; similarly, we write $\mathsf{InterpUni}(\mathsf{B})$ for the minimal size of a linear straight-line program that computes the inverse univariate interpolation map. The coefficients of these linear straight-line programs are functions of B: in our computational model, we do not take the cost of computing these coefficients into account.

For arbitrary points, $\mathsf{EvalUni}(\mathsf{B})$ and $\mathsf{InterpUni}(\mathsf{B})$ belong to $O(\mathsf{M}(d)\lg(d))$, see [17] and references therein, as well as [8] for more recent algorithms. Better estimates are known in special cases, if the points B form a geometric progression [28, 6, 10] or are suitable powers of a root of unity [20].

4.1 Truncation in partial degree

We consider here truncation in partial degrees $(d_1, \dots, d_n)$; we prove Corollaries 1 and 2 and present an application to computation with algebraic numbers. The notation of Section 2 is still in use.

Evaluation and interpolation on a rectangular grid. When truncating in partial degree, the basis T is the set of all monomials $X_1^{t_1} \cdots X_n^{t_n}$, with $0 \le t_i < d_i$ for $1 \le i \le n$. The multivariate evaluation and interpolation problems we have to solve then become the following: Evaluation requires to evaluate a polynomial in $\mathrm{Span}(\mathsf{T})$ on the rectangular grid $\mathsf{A}_\mathsf{T} = \mathsf{A}_1 \times \cdots \times \mathsf{A}_n$, with

$$\mathsf{A}_1 = [a_{1,0}, \dots, a_{1,d_1-1}], \dots, \mathsf{A}_n = [a_{n,0}, \dots, a_{n,d_n-1}],$$

and interpolation is the inverse operation. Fast solutions to these problems are well known. Let $P \in \mathrm{Span}(\mathsf{T}) \subset k[X_1, \dots, X_n]$. To perform evaluation, we proceed with one variable at a time. We first compute the d_1 polynomials

$$P(a_{1,0}, X_2, \dots, X_n), \dots, P(a_{1,d_1-1}, X_2, \dots, X_n)$$

in $k[X_2, \dots, X_n]$. Proceeding separately with each monomial in $X_2, \dots, X_n$, we see that the cost of this operation is $\mathsf{EvalUni}(\mathsf{A}_1)\, d_2 \cdots d_n$ operations in k. Then, we evaluate these d_1 polynomials on A_2, etc ... Summing up, the whole cost of these operations is

$$d_1 \cdots d_n \sum_i \frac{\mathsf{EvalUni}(\mathsf{A}_i)}{d_i} = \deg_M \sum_i \frac{\mathsf{EvalUni}(\mathsf{A}_i)}{d_i}.$$

Interpolation is performed in the same manner, interpolating one variable after the other; similarly, the total cost is

$$d_1 \cdots d_n \sum_i \frac{\mathsf{InterpUni}(\mathsf{A}_i)}{d_i} = \deg_M \sum_i \frac{\mathsf{InterpUni}(\mathsf{A}_i)}{d_i}.$$

See for instance [34, Chapter 14.1] for the details of this algorithm. Using the most general estimates on $\mathsf{EvalUni}$ and $\mathsf{InterpUni}$, we deduce the following estimates:

$$\mathsf{Eval}(\mathsf{A}_\mathsf{T}),\ \mathsf{Interp}(\mathsf{A}_\mathsf{T}) \in O\left(\deg_M \sum_i \frac{\mathsf{M}(d_i)\lg(d_i)}{d_i}\right).$$

(Of course, these estimates can be improved for the special choices of $\mathsf{A}_1, \dots, \mathsf{A}_n$ mentioned above.) This suffices to prove Corollary 1. Indeed, plugging these estimates into those of Theorem 1, we deduce that the total complexity of multiplication truncated in partial degrees $(d_1, \dots, d_n)$ is in the requested class

$$O\left(\mathrm{reg}_M\ \deg_M \sum_i \frac{\mathsf{M}(d_i)\lg(d_i)}{d_i} + \mathsf{M}(\mathrm{reg}_M)\deg_M\right).$$

Combining algorithms. To conclude our discussion, we prove Corollary 2: *For any $\varepsilon > 0$, the total complexity of the product modulo $M = (X_1^{d_1}, \dots, X_n^{d_n})$ is in $O((\deg_M)^{1+\varepsilon})$.*

The idea behind the proof is the following. The algorithm of Corollary 1 is efficient when the number of variables is large compared to the truncation degrees, whereas the naive algorithm using Kronecker's substitution is efficient when the degree is large compared to the number of variables. Combining these two strategies leads to the proof of Corollary 2.

First we give some preliminary formulas, which explicit the above statements. Corollary 1 shows that there exist L and α in $\mathbb{R}$ such that for all ℓ and all $d_1, \dots, d_\ell$, the total complexity of the product in the k-algebra

$$k[X_1, \dots, X_\ell]/(X_1^{d_1}, \dots, X_\ell^{d_\ell})$$

is at most $\mathfrak{C}(d_1, \dots, d_\ell) := LSP\lg^{\alpha}(SP)$, with $S = d_1 + \cdots + d_\ell$ and $P = d_1 \cdots d_\ell$. On the other hand, we can use Kronecker's substitution to multiply series as polynomials, and discard unwanted coefficients. Using [14, Section 3.5], we obtain that for any ring R, any ℓ and n, and any $d_{\ell+1}, \dots, d_n$, the total complexity of the product in the R-algebra

$$R[X_{\ell+1}, \dots, X_n]/(X_{\ell+1}^{d_{\ell+1}}, \dots, X_n^{d_n})$$

is at most $\mathfrak{D}(d_{\ell+1}, \dots, d_n) := KP'\lg(P')\lg\lg(P')$, where K is the constant introduced in Equation (1) of Section 2, and P' equals $2^{n-\ell}\, d_{\ell+1} \cdots d_n$.

We can then prove the corollary itself. Let $\varepsilon > 0$, let n and $d_1, \dots, d_n$ be integers; let $M = (X_1^{d_1}, \dots, X_n^{d_n})$ and

$$Q = k[X_1, \dots, X_n]/(X_1^{d_1}, \dots, X_n^{d_n}).$$

Due to the k-algebra isomorphisms $k[X]/(X) \simeq k$ and $k[X,Y]/(X^d, Y^e) \simeq k[X,Y]/(X^e, Y^d)$, we can without loss of generality suppose that $2 \le d_1 \le \cdots \le d_n$; note then that $\deg_M = d_1 \cdots d_n \ge 2^n$. We define $\omega = \frac{2}{\varepsilon}$, and ℓ by the condition that $d_\ell \le 2^{\omega} < d_{\ell+1}$ (we let $d_0 = 0$ and $d_{n+1} = \infty$ to make ℓ well-defined).

LEMMA 2. *The total complexity of the product in Q is at most $6\mathfrak{C}\mathfrak{D}$, with $\mathfrak{C} = \mathfrak{C}(d_1, \ldots, d_\ell)$ and $\mathfrak{D} = \mathfrak{D}(d_{\ell+1}, \ldots, d_n)$.*

PROOF. Let $R = k[X_1, \ldots, X_\ell]/(X_1^{d_1}, \ldots, X_\ell^{d_\ell})$, so that $Q = R[X_{\ell+1}, \ldots, X_n]/(X_{\ell+1}^{d_{\ell+1}}, \ldots, X_n^{d_n})$. We apply our algorithm for the product in R, and Kronecker's substitution for the product in Q seen as an R-algebra.

Going back to the definition of our computational model, we deduce that there exist $\mathfrak{c} \le \mathfrak{C}$, $\mathfrak{d} \le \mathfrak{D}$, R-linear forms $\lambda_i, \mu_i : Q \to R$ and $\rho_i \in Q$, with $1 \le i \le \mathfrak{d}$, k-linear forms $f_j, g_j : R \to k$ and $w_j \in R$, with $1 \le j \le \mathfrak{c}$, such that for all A, B in Q one has

$$AB = \sum_{i \le \mathfrak{d}, j \le \mathfrak{c}} \big((f_j \circ \lambda_i)(A)\big)\ \big((g_j \circ \mu_i)(B)\big)\ w_j \rho_i;$$

furthermore, the R-linear map

$$\begin{array}{c} A \mapsto [\lambda_1(A), \ldots, \lambda_{\mathfrak{d}}(A)], \quad B \mapsto [\mu_1(B), \ldots, \mu_{\mathfrak{d}}(B)], \\ [\eta_1, \ldots, \eta_{\mathfrak{d}}] \mapsto \sum_i \eta_i \rho_i \end{array} \qquad (2)$$

can be evaluated by a R-linear straight-line program Γ of size $\mathfrak{D}$, and the k-linear map

$$\begin{array}{c} a \mapsto [f_1(a), \ldots, f_{\mathfrak{c}}(a)], \quad b \mapsto [g_1(b), \ldots, g_{\mathfrak{c}}(b)], \\ [\eta_1, \ldots, \eta_{\mathfrak{c}}] \mapsto \sum_j \eta_j w_j \end{array} \qquad (3)$$

can be evaluated by a k-linear straight-line program of size $\mathfrak{C}$. Since $\mathfrak{c}\mathfrak{d} \le 6\mathfrak{C}\mathfrak{D}$, it remains to estimate the cost of evaluating the map $A \mapsto (f_j \circ \lambda_i)(A)$, $B \mapsto (g_j \circ \mu_i)(B)$, $[\eta_{1,1}, \ldots, \eta_{\mathfrak{c},\mathfrak{d}}] \mapsto \sum \eta_{i,j} w_j \rho_i$. In view of the multiplication algorithm of [12], Γ can be taken with constants in the image of $\mathbb{Z}$ in k; then, the above linear map can be read off the tensor product of the maps (2) and (3). In view of the sizes of the matrices involved, Lemma 13.7.5 in [11] shows that this tensor product can be computed by a linear straight-line program of size at most $3\mathfrak{d}\mathfrak{C} + 3\mathfrak{c}\mathfrak{D} \le 6\mathfrak{C}\mathfrak{D}$. $\square$

Using the inequalities $2^n \le \deg_M$ and $d_1 + \cdots + d_\ell \le \deg_M$, this bound becomes $6KL(d_1 + \cdots + d_\ell)\Delta 2^{n-\ell} \deg_M$, with $\Delta = \lg^{\alpha+1}((\deg_M)^2) \lg\lg((\deg_M)^2)$.

Next, we write the inequalities $d_1 + \cdots + d_\ell \le n d_\ell \le n 2^{\omega} \le 2^{\omega} \lg(\deg_M)$ and $(2^{\omega})^{n-\ell} \le d_{\ell+1} \cdots d_n \le \deg_M$, so that $2^{n-\ell} \le (\deg_M)^{\frac{\varepsilon}{2}}$. In particular, $(d_1 + \cdots + d_\ell)\Delta$ is polylogarithmic in $\deg_M$, so it admits the upper bound $N_\varepsilon (\deg_M)^{\frac{\varepsilon}{2}}$, for some constant N_ε. Putting all these estimates together finishes the proof, since it shows that the total complexity of the product in Q is upper bounded by $6KLN_\varepsilon(\deg_M)^{1+\varepsilon}$.

Application: computing with algebraic numbers. Let f and g be monic polynomials in $k[T]$, of degrees m and n. We are interested in computing their *composed sum* $f \oplus g$, which is the polynomial of degree $D = mn$ given by

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

the product running over all the roots α of f and β of g, counted with multiplicities. See [7] for a review of some applications of this basic operation on algebraic numbers.

In characteristic 0, the fastest known approach for computing $f \oplus g$ is due to Dvornicich and Traverso [13]. The key idea is that the exponential generating series of the *power sums* of $f \oplus g$ is the product of the exponential generating series of the power sums of f and g. Using fast conversions between coefficients and power sums, computing $f \oplus g$ then has complexity $O(\mathsf{M}(D))$, which is almost linear in D.

Suppose now that k has characteristic $p < D$. Then the previous approach does not apply anymore. The first difficulty is that a polynomial is not determined by its power sums. The second difficulty is that the relation between the power sums of f, g and $f \oplus g$ is no more valid: the above generating series cannot be defined, since they require division by p.

We first discuss how to handle the first problem: if the multiplicities of all roots of $f \oplus g$ are less than p, given its first $2D$ power sums, one can recover $f \oplus g$ in time

$$O\left(\mathsf{M}(D) + p\,\mathsf{M}\left(\frac{D}{p}\right) \lg\left(\frac{D}{p}\right)\right),$$

in the computation tree model, using an elaboration by Pan [27] of an idea of Schönhage's [31]. On the other hand, the first $2D$ power sums of f and g can still be computed in $O(\mathsf{M}(D))$ operations in k.

The article [7] gives a solution for the second problem. Let $\mathbf{T} = (T_i)_{i \ge 0}$ be an infinite set of indeterminates; let Λ be the set of finite sequences (of arbitrary length) $\mathbf{i} = (i_0, \ldots, i_s)$, with all i_ℓ in $\{0, \ldots, p-1\}$. For $\mathbf{i} \in \Lambda$, we write $\mathbf{i}! = i_0! \cdots i_s!$, $\mathbf{T}^{\mathbf{i}} = T_0^{i_0} \cdots T_s^{i_s}$ and $\lambda(\mathbf{i}) = i_0 + i_1 p + \cdots + i_s p^s$. For a univariate polynomial $f \in k[T]$, let $\mathfrak{n}_i(f)$ be its ith power sum. Writing

$$\mathsf{N}_p(f) \ = \ \sum_{\mathbf{i} \in \Lambda} \frac{\mathfrak{n}_{\lambda(\mathbf{i})}(f)}{\mathbf{i}!} \mathbf{T}^{\mathbf{i}},$$

then the following identity holds:

$$\mathsf{N}_p(f \oplus g) = \ \mathsf{N}_p(f)\mathsf{N}_p(g) \mod (\mathbf{T}^p),$$

where $(\mathbf{T}^p)$ denotes the ideal generated by all monomials of the form T_i^p in $k[[\mathbf{T}]]$. Since $2D$ power sums of $f \oplus g$ are requested, we only need the coefficients of the monomials in $T_0, \ldots, T_s$, where $s = \lfloor \lg(2D)/\lg(p) \rfloor$. Given the first $2D$ power sums of f and g, this can be done by multiplying two multivariate power series involving at most $\lg(2D)/\lg(p)$ variables and of degree less than p in each variable.

Suppose for simplicity that p is constant; based on this discussion, the algorithm of [7] has complexity $O(D^{1+1/p})$ in this case. Corollary 1 yields the following improved bound.

COROLLARY 3. *Let f, g be as above, and suppose that all roots of $f \oplus g$ have multiplicity less than p. Then $f \oplus g$ can be computed in $O(D \lg^2(D) \lg^2 \lg(D))$ operations in k.*

This algorithm was implemented using Shoup's C++ NTL library, available at `www.shoup.net`; timings were obtained on an AMD 64 3500+, running in 64 bit mode. Figure 1 illustrates this algorithm taking $\mathbb{F}_3$ for base field, with $\deg f = \deg g$ in abscissae and time (in seconds) in ordinates; we distinguish the time for power series multiplication from the total running time. Since $k = \mathbb{F}_3$, the multivariate power series are truncated in partial degree 3 in all variables, and

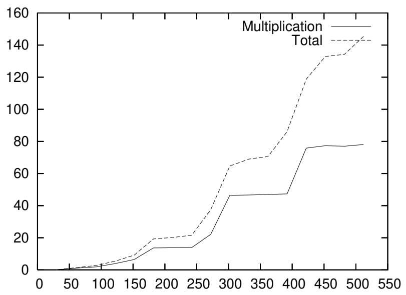

Figure 1: Composed sum over $\mathbb{F}_3$.

the number of variables grows with $\deg f$. We compared our multiplication algorithm to the naive one for this specific task: ours becomes more efficient for 7 variables or more.

4.2 Gradient of ideals

Let again $M = (X_1^{d_1}, \dots, X_n^{d_n})$, and T the terms not in M. Following [21], the *gradient* ∇M of M is defined as the 0-dimensional monomial ideal generated by the monomials not in T^+, where T^+ is the set of terms $\mathsf{T} \cup \cup_{i \le n} X_i \mathsf{T}$. Lecerf's deflation algorithm [21], which yields a quadratic Newton iteration for systems with multiple roots, relies on computations modulo an ideal of the form ∇M. We obtain the following estimate for this task.

PROPOSITION 2. *For any $\varepsilon > 0$, the total complexity of the product modulo the gradient of $M = (X_1^{d_1}, \dots, X_n^{d_n})$ is in $O(n\,(\deg_M)^{1+\varepsilon})$.*

PROOF. The set T^+ is the disjoint union $\mathsf{T} \cup \cup_{i \le n} \mathsf{T}_i$, where T_i is the set of terms $X_1^{t_1} \cdots X_n^{t_n}$, with $t_i = d_i$ and $0 \le t_j < d_j$ for $j \ne i$. Let then A and B in $\mathrm{Span}(\mathsf{T}^+)$, with

$$A = A_\mathsf{T} + \sum_{i \le n} A_i, \quad B = B_\mathsf{T} + \sum_{i \le n} B_i,$$

such that A_T and B_T are in $\mathrm{Span}(\mathsf{T})$, and A_i and B_i are in $\mathrm{Span}(\mathsf{T}_i)$ for all i. Computing $AB \bmod \nabla M$ is done by "expanding" the product.

We first compute $C = A_\mathsf{T} B_\mathsf{T} \bmod \nabla M$, which we write $C = C_\mathsf{T} + \sum_{i \le n} C_i$, with C_T in $\mathrm{Span}(\mathsf{T})$ and C_i in $\mathrm{Span}(\mathsf{T}_i)$. The component C_T is obtained by multiplication modulo M. To obtain C_i, we write $B_\mathsf{T} = b_i + X_i \mathfrak{b}_i$, with b_i in $k[X_1, \dots, X_{i-1}, X_{i+1}, \dots, X_n]$. Let $c_i = A_\mathsf{T} \mathfrak{b}_i \bmod M$; then the coefficient of a monomial t in C_i is obtained as the coefficient of t/X_i in c_i, so C_i is known as soon as c_i is.

Next, we compute $A_\mathsf{T} B_i$ and $A_i B_\mathsf{T} \bmod \nabla M$, for all i. This is done by writing $A_i = X^{d_i} \alpha_i$ and $B_i = X^{d_i} \beta_i$, so all coefficients of the products above are known as soon as $A_\mathsf{T} \beta_i \bmod M$ and $\alpha_i B_\mathsf{T} \bmod M$ are known. Finally, notice that for all i, j, $A_i B_j$ is 0 mod ∇M. Summarizing, $AB \bmod \nabla M$ can be computed for $3n + 1$ products modulo M and less than $3n \deg_M$ additions. Corollary 2 ends the proof. $\square$

Application: polynomial system solving. As mentioned above, computing modulo gradients of ideals can speed up the resolution of polynomial systems. Giving the main result of [21] would require to introduce extensive notation; we will simply repeat the main example given there. Let $F = F_1, F_2, F_3$ in $\mathbb{Q}[X_1, X_2, X_3]$, with

$$\begin{aligned} F_1 &= 2X_1 + 2X_1^2 + 2X_2 + 2X_2^2 + X_3^2 - 1, \\ F_2 &= (X_1 + X_2 - X_3 - 1)^3 - X_1^3 \\ F_3 &= (2X_1^3 + 5X_2^2 + 10X_3 + 5X_3^2 + 5)^3 - 1000X_1^5. \end{aligned}$$

Then $x^* = (0, 0, -1) \in \mathbb{Q}^3$ is a root of this system. Given a prime p, and the point $(0, 0, p-1) \in \mathbb{F}_p^3$, we are interested in computing approximations of x^* in the p-adic topology. However, x^* is a root of multiplicity 18 of F, so the Newton iterator fails to converge with quadratic speed to this root. Lecerf's deflation algorithm enables one to recover a quadratic convergence rate; it turns out that each step of the modified Newton iteration now requires to compute modulo the gradient of the ideal (X_1, X_2^3, X_3^4).

In general, one has to compute modulo the gradient of an ideal $M = (X_1^{d_1}, \dots, X_n^{d_n})$, where $\deg_M$ is upper bounded by the multiplicity of the root to approximate. Lecerf's algorithm uses a quadratic power series multiplication algorithm, so each lifting step has a complexity quadratic in the multiplicity; our algorithm reduces this to an almost linear complexity (Lecerf's result also counts the cost of handling exponents, which is not taken into account in our model). Note that this lifting algorithm is also a crucial part of a more general equidimensional decomposition algorithm [22].

5. CONCLUSION, OPEN PROBLEMS

Our approach requires to solve specific but well-known [29] evaluation / interpolation problems. Obtaining sharp complexity estimates for these tasks is of independent interest.

Before considering the general case, one may want to look at special cases first. We saw in Section 4 how to solve these problems in quasi-linear time for partial degree truncation, which corresponds to evaluation / interpolation on a rectangular grid. Another important special case is that of truncation in total degree, which corresponds to evaluation / interpolation on a simplex. For the moment, a fast algorithm (of linear complexity) for this purpose is missing.

To conclude, let us illustrate possible approaches on another particular case, illustrated on Figure 2. The monomial support T is on the leftmost picture; it consists in the set-theoretic difference of two cubes in 3-space. The goal is to evaluate or interpolate a polynomial P with this support on a set of points which form the same pattern. A natural

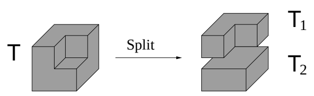

Figure 2: Splitting a monomial support.

idea is to apply a divide-and-conquer algorithm, by splitting T along the X_3-axis, into the two parts T_1 and T_2 displayed on the rightmost picture. The analogue of the classical fast evaluation algorithm is to reduce P modulo defining ideals of the two corresponding sets of points, and go recursively. As to interpolation, an interesting question would be whether evaluation and interpolation are equivalent, as is essentially the case when $n = 1$ [9]. Even for such special cases, precise answers to these questions still have to be written.

6. REFERENCES

[1] D. Bini. Relations between exact and approximate bilinear algorithms. Applications. *Calcolo*, 17(1):87–97, 1980.

[2] D. Bini, M. Capovani, F. Romani, and G. Lotti. $O(n^{2.7799})$ complexity for $n \times n$ approximate matrix multiplication. *Inf. Proc. Lett.*, 8(5):234–235, 1979.

[3] D. Bini, G. Lotti, and F. Romani. Approximate solutions for the bilinear form computational problem. *SIAM J. Comput.*, 9(4):692–697, 1980.

[4] C. Bischof, G. Corliss, and A. Griewank. Structured second- and higher-order derivatives through univariate Taylor series. *Opt. Meth. and Soft.*, 2:211–232, 1993.

[5] M. Bläser. The complexity of bivariate power series arithmetic. *TCS*, 295(1-3):65–83, 2003.

[6] L. I. Bluestein. A linear filtering approach to the computation of the discrete Fourier transform. *IEEE Trans. Electroacoustics*, AU-18:451–455, 1970.

[7] A. Bostan, P. Flajolet, B. Salvy, and É. Schost. Fast computation with two algebraic numbers, 2003.

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

[9] A. Bostan and É. Schost. On the complexities of multipoint evaluation and interpolation. *TCS*, 329(2):223–235, 2004.

[10] A. Bostan and É. Schost. Polynomial evaluation and interpolation on special sets of points, *J. Complexity*, to appear.

[11] P. Bürgisser, M. Clausen, and M. A. Shokrollahi. *Algebraic complexity theory*, volume 315 of *Grund. der Math. Wiss.* Springer, 1997.

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

[13] 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.

[14] I. Z. Emiris and V. Y. Pan. Applications of FFT. In *Algorithms and theory of computation handbook*, pages 17–1–17–30. CRC, Boca Raton, FL, 1999.

[15] I. Z. Emiris and V. Y. Pan. Symbolic and numeric methods for exploiting structure in constructing resultant matrices. *JSC*, 33(4):393–413, 2002.

[16] C. M. Fiduccia and Y. Zalcstein. Algebras having linear multiplicative complexities. *J. Assoc. Comput. Mach.*, 24(2):311–331, 1977.

[17] J. von zur Gathen and J. Gerhard. *Modern computer algebra*. Cambridge University Press, 1999.

[18] G. Hanrot and P. Zimmermann. A long note on Mulders' short product. *JSC*, 37(3):391–401, 2004.

[19] J. van der Hoeven. Relax, but don't be too lazy. *JSC*, 34(6):479–542, 2002.

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

[21] G. Lecerf. Quadratic Newton iteration for systems with multiplicity. *Found. of Comput. Math.*, 2(3):247–293, 2002.

[22] G. Lecerf. Computing the equidimensional decomposition of an algebraic closed set by means of lifting fibers. *J. Complexity*, 19(4):564–596, 2003.

[23] G. Lecerf and É. Schost. Fast multivariate power series multiplication in characteristic zero. *SADIO Electronic Journal*, 5(1), 2003.

[24] F. S. Macaulay. Some properties of enumeration in the theory of modular systems. *Proc. London Math. Soc.*, 26:531–555, 1927.

[25] F. Mora. De nugis Groebnerialium 2: Applying Macaulay's trick in order to easily write a Gröbner basis. *AAECC*, 13(6):437–446, 2003.

[26] T. Mulders. On short multiplications and divisions. *AAECC*, 11(1):69–88, 2000.

[27] V. Y. Pan. New techniques for the computation of linear recurrence coefficients. *Finite Fields and their Applications*, 6(1):93–118, 2000.

[28] L. R. Rabiner, R. W. Schafer, and C. M. Rader. The chirp z-transform algorithm and its application. *Bell System Tech. J.*, 48:1249–1292, 1969.

[29] T. Sauer. Lagrange interpolation on subgrids of tensor product grids. *Math. Comp.*, 73:181–190, 2004.

[30] A. Schönhage. Partial and total matrix multiplication. *SIAM J. Comput.*, 10(3):434–455, 1981.

[31] A. Schönhage. Fast parallel computation of characteristic polynomials by Leverrier's power sum method adapted to fields of finite characteristic. In *Automata, languages and programming*, volume 700 of *LNCS*, pages 410–417. Springer, 1993.

[32] A. Schönhage. Bivariate polynomial multiplication patterns. In *AAECC-11*, volume 948 of *LNCS*, pages 70–81. Springer, 1995.

[33] S. Winograd. Some bilinear forms whose multiplicative complexity depends on the field of constants. *Math. Systems Theory*, 10(2):169–180, 1976/77.

[34] R. Zippel. *Effective Polynomial Computation*. Kluwer Academic Publishers, 1993.