Fast Multivariate Power Series Multiplication in Characteristic Zero

Grégoire Lecerf
GAGE, École polytechnique (France)
June 11, 2001

Summary by Ludovic Meunier

Abstract
Let $S$ be a multivariate power series ring over a field of characteristic zero. The article [5] presents an asymptotically fast algorithm for multiplying two elements of $S$ truncated according to total degree. Up to logarithmic factors, the complexity of the algorithm is optimal in the sense that it is linear in the size of the output.

1. Introduction
Let $k$ be a field of characteristic zero. We write $S = k[[x_1, \ldots, x_n]]$ for the multivariate power series ring in the $n$ variables $x_1, \ldots, x_n$. Let $\mathcal{J}$ be any ideal of $S$. By computing at precision $\mathcal{J}$ in $S$, we understand computing modulo the ideal $\mathcal{J}$ in $S$. In other words, power series in $S$ are regarded as vectors in the $k$-algebra $S/\mathcal{J}$. We denote by $m$ the maximal ideal $(x_1, \ldots, x_n)$ in $S$ and by $d$ any positive integer. The paper [5] sets the problem of a fast algorithm for multiplying two power series in $S$ truncated in total degree $d$, that is computed at precision $m^{d+1}$.

The general question of a fast algorithm for multivariate multiplication in $S$ modulo any ideal remains an open problem and has received very little attention in the literature. Previous works (e.g., [2]) investigated computation modulo the ideal $(x_1^{d+1}, \ldots, x_n^{d+1})$, that is truncation according to partial degree with respect to each variable $x_i$. The method used is called Kronecker’s substitution and is briefly discussed in Section 3.

The need for multiplication routines modulo $m^{d+1}$ arises in various fields, such as polynomial system solving [7] and treatment of systems of partial differential equations.

The efficiency of the algorithm is measured with respect to the model of non-scalar complexity. By non-scalar complexity, we understand the number of primitive operations in the field $k$ needed to complete the algorithm, independently of the sizes of the numbers involved (see [3]). We now introduce some notation. We denote by $D = \deg(m^{d+1})$ the degree of the ideal $m^{d+1}$. $D$ is the number of monomials in $S$ which are not in $m^{d+1}$, that is the dimension of the $k$-algebra $S/m^{d+1}$. Simple combinatorial considerations give

$$D = \deg(m^{d+1}) = \binom{d+n}{n}.$$

We set $C := \deg(m^d)$ and denote by $\mathcal{M}_u(\delta)$ the complexity of the multiplication of two univariate polynomials of degree $\delta$ in $k[t]$.

The next section presents the algorithm; its complexity belongs to

$$O(D \log^3 D \log \log D).$$
Since $D$ is the size of the output, the algorithm is optimal, up to the logarithmic factors.

2. The Algorithm

2.1. Description. The first step of the algorithm consists in translating the multivariate problem into a univariate one. This is motivated by the fact that fast algorithms for univariate power series multiplication are known (e.g., [6]).

Let $t$ be a new variable. We consider the substitution

$$
\tilde{\mathcal{R}}_t : \quad S/m^{d+1} \quad \longrightarrow \quad k[x_1, \ldots, x_n][[t]]/(t^{d+1})
$$

$$
f(x_1, \ldots, x_n) \longmapsto f(x_1 t, \ldots, x_n t).
$$

If $f$ is an element of $S/m^{d+1}$, $\tilde{\mathcal{R}}_t(f)$ is a univariate power series in the single variable $t$ truncated at degree $d$. It can then be written $\tilde{\mathcal{R}}_t(f) = \tilde{f}_0 + \tilde{f}_1 t + \cdots + \tilde{f}_d t^d$, where each coefficient $\tilde{f}_i$ is a homogeneous multivariate polynomial in the variables $x_1, \ldots, x_n$ of total degree $i$. This remark on the degree suggests that:

1. the substitution $\tilde{\mathcal{R}}_t$ is optimal, in the sense that it provides us with a representation of $f$ that retains exactly the monomials that form a basis of $S/m^{d+1}$. In particular, the algorithm does not suffer from any overhead caused by unnecessary terms (see Section 3);

2. in view of the homogeneity of the $\tilde{f}_i$, keeping all of the variables $x_i$ is redundant. The substitution defined by

$$
\mathcal{R}_t : \quad S/m^{d+1} \quad \longrightarrow \quad k[x_2, \ldots, x_n][[t]]/(t^{d+1}) = (k[[t]]/(t^{d+1})) [x_2, \ldots, x_n]
$$

$$
f(x_1, \ldots, x_n) \longmapsto f(t, x_2 t, \ldots, x_n t)
$$

reduces the complexity in the step of evaluation-interpolation (see below): $n - 1$ variables, instead of $n$ variables, are actually needed.

The second step of the algorithm performs the multiplication. Let $f$ and $g$ be two power series in $S/m^{d+1}$ and $h$ be the product $fg$ in $S/m^{d+1}$. The equality $h = fg$ turns into

$$
(2) \quad \mathcal{R}_t(h) = \mathcal{R}_t(f) \mathcal{R}_t(g).
$$

Consequently, we concentrate on a fast way to compute $\mathcal{R}_t(h)$. We use an evaluation-interpolation scheme. We first consider the evaluation map at the point $P = (p_2, \ldots, p_n)$ in $k^{n-1}$ defined by

$$
\mathcal{E}_P : \quad (k[[t]]/(t^{d+1})) [x_2, \ldots, x_n] \quad \longrightarrow \quad k[[t]]/(t^{d+1})
$$

$$
f(x_2, \ldots, x_n) \longmapsto f(P).
$$

We then apply $\mathcal{E}_P$ to equation (2), which yields

$$
(3) \quad \mathcal{E}_P(\mathcal{R}_t(h)) = \mathcal{E}_P(\mathcal{R}_t(f)) \mathcal{E}_P(\mathcal{R}_t(g)) \mod t^{d+1}.
$$

Equation (3) holds for any point $P$ and computes the product $\mathcal{R}_t(h)$ at $P$ by using a univariate power series multiplication algorithm. Such an algorithm is described in [6].

The last step of the algorithm consists in reconstructing $h$ from a set of values of $\mathcal{R}_t(h)$. We regard $\mathcal{R}_t(h)$ as a multivariate polynomial in the variables $x_2, \ldots, x_n$. There exists an interpolation map

$$
\mathcal{I} : \quad (k[[t]]/(t^{d+1}))^C \quad \longrightarrow \quad (k[[t]]/(t^{d+1})) [x_2, \ldots, x_n]
$$

$$
(f(P_1), \ldots, f(P_C)) \longmapsto f(x_2, \ldots, x_n),
$$

which recovers $\mathcal{R}_t(h)$ from a set of $C$ pairwise distinct values $\{\mathcal{E}_{P_1}(\mathcal{R}_t(h)), \ldots, \mathcal{E}_{P_C}(\mathcal{R}_t(h))\}$. The evaluation points $P_i$, for $i$ in $1, \ldots, C$, are chosen to be powers of distinct prime numbers, namely $P_i = (p_{i,1}^{p_{i,2}}, \ldots, p_{i,n})$, where $p_{i,j}$ are distinct prime numbers. Note the key point is that the characteristic of the ground field $k$ is zero, so that all $\mathcal{E}_{P_i}(\mathcal{R}_t(h))$ have pairwise distinct values. An implementation
of both maps \( \mathcal{E}_P \) and \( \mathcal{I} \) is described by J. Canny, E. Kaltofen, and Y. Lakshman in [4]. Their method relies on fast univariate multipoint evaluation and interpolation (e.g., [1]).

Finally, we reconstruct \( h \) from \( \mathcal{R}_t(h) \). If \( \mathcal{R}_t(h) = h_0 + h_1 t + \cdots + h_d t^d \) is given, \( h \) is obtained by homogenizing each \( h_i \) in degree \( i \) with respect to the variable \( x_1 \) and then evaluating at \( t = 1 \).

We are now ready to unfold the algorithm.

**MultivariatePS_Mult** := proc((f,g)
(1) \( F \leftarrow \mathcal{R}_t(f); G \leftarrow \mathcal{R}_t(g); \) // new representation
(2) for \( i \) in \((P_1, \ldots, P_C)\) do // evaluation
    \( F_{P_i} \leftarrow \mathcal{E}_{P_i}(F); G_{P_i} \leftarrow \mathcal{E}_{P_i}(G); \)
(3) for \( i \) to \( C \) do // univariate multiplication
    \( H_{P_i} \leftarrow F_{P_i} G_{P_i}; \)
(4) \( \mathcal{R}_t(h) \leftarrow \mathcal{I}(H_{P_1}, \ldots, H_{P_C}); \) // interpolation
(5) \( h \leftarrow \text{homogenization in degree with respect to } x_1 \) // reconstruction
    in \( \mathcal{R}_t(h); \)
return \( h; \)

The next section derives the complexity result claimed by (1).

2.2. **Complexity.** Steps 1 and 5 can be performed in \( \mathcal{O}(C) \) operations. We examine the cost of Steps 2, 3, and 4 separately:

- Step 2 evaluates the \( d \) coefficients of \( F \) and \( G \) at \( C \) points. The \( C \) points \( P_i \) are chosen to be powers of the \( n - 1 \) distinct prime numbers \((p_2, \ldots, p_n)\), namely \( P_i = (p_2^{i-1}, \ldots, p_n^{i-1}) \). Each coefficient can be computed in \( \mathcal{O}(\mathcal{M}_u(C) \log C) \) operations, according to the algorithm for fast multipoint evaluation given in [4]. This yields an overall complexity of \( \mathcal{O}(d \mathcal{M}_u(C) \log C) \) for Step 2.

- Step 3 performs \( C \) univariate power series products. Each multiplication requires \( \mathcal{O}(\mathcal{M}_u(d)) \) operations. Complexity of Step 3 is then \( \mathcal{O}(C \mathcal{M}_u(d)) \).

- Step 4 interpolates the \( d \) coefficients of \( H \). Each interpolation requires \( \mathcal{O}(\mathcal{M}_u(C) \log C) \) operations, also using the algorithm presented in [4]. Step 4 then requires \( \mathcal{O}(d \mathcal{M}_u(C) \log C) \) operations.

The overall complexity of the algorithm is then derived by replacing \( \mathcal{M}_u(C) \) by its estimate \( \mathcal{O}(C \log C \log \log C) \) obtained in [6] and noting that \( C < D \log(D)/d \). This yields

\[ \mathcal{O}(D \log^3 D \log \log D). \]

2.3. **Generalization.** We mention that van der Hoeven generalized the algorithm to the case when

\[ \mathcal{J} = (x_1^{d_1} \cdots x_n^{d_n}, \text{ for } \alpha_1 d_1 + \cdots + \alpha_n d_n > d), \]

where the \( \alpha_i \) are positive integers, by using the substitution defined by

\[ \mathcal{V}_i : S/\mathcal{J} \quad \mapsto \quad k[x_2, \ldots, x_n][[t]]/(t^{d+1}) \]

\[ f(x_1, \ldots, x_n) \quad \mapsto \quad f(t^{\alpha_1}, x_2^{\alpha_2}, \ldots, x_n^{\alpha_n}) \]

instead of \( \mathcal{R}_1 \). The rest of the algorithm remains unaltered.
3. Appendix: Kronecker’s Substitution

Kronecker’s substitution is defined by the map

\[ \mathcal{K}_d : \frac{S}{J} \rightarrow k[[t]]/(t^{(2d+1)n}) \]

\[ f(x_1, \ldots, x_n) \mapsto f(t, t^{(2d+1)}, \ldots, t^{(2d+1)n-1}) \]

where \( J = (x_1^{d+1}, \ldots, x_n^{d+1}) \). This substitution truncates power series in partial degree \( d \) with respect to each variable \( x_i \). Let \( f \) be a power series in \( S/J \), one recovers the coefficient of \( x_1^{d+1} \ldots x_n^{d+1} \) in \( f \) by simply reading off the coefficient of \( t^{e_1+(2d+1)e_2+\ldots+(2d+1)^n-1} \) in \( \mathcal{K}_d(f) \). The cost of this algorithm is the cost of the multiplication of two univariate polynomials of degree \( (2d)^n \), that is \( \mathcal{O}(\mathcal{M}_n((2d)^n)) \). This is the lowest known complexity for multivariate power series multiplication modulo the ideal \( (x_1^{d+1}, \ldots, x_n^{d+1}) \). In particular, when addressed in this context, the algorithm presented above requires precision \( m^{nd+1} \) and yields a similar complexity.

Kronecker’s substitution may be used to compute modulo \( m^{d+1} \) as well. However, it results in a significant overhead of \( \mathcal{O}(2^d n!) \), for fixed \( n \) and \( d \gg n \), with respect to the size of the power series.

Bibliography


