Multiplication of Power Series

Éric Schost, LIX, École polytechnique

Setting

- When talking about multiplication of power series, one has to specify how to **truncate** them.
- The truncation pattern will be determined by the data of a **zero-dimensional monomial ideal**.
- **Example:** $M = \langle\, X_1^3,\ X_1^2X_2^2,\ X_2^3 \,\rangle$ corresponds to series with support in

$$\mathsf{T} = \{1,\, X_1,\, X_1^2,\, X_2,\, X_1X_2,\, X_1^2X_2,\, X_2^2,\, X_1X_2^2\}.$$

Here, the square of

$$1 + X_1 + X_1^2 + X_2 + X_1X_2 + X_1^2X_2 + X_2^2 + X_1X_2^2$$

is

$$1 + 2X_1 + 3X_1^2 + 2X_2 + 4X_1X_2 + 6X_1^2X_2 + 3X_2^2 + 6X_1X_2^2.$$

Applications

Many forms of Hensel lifting involve computing modulo

$$\langle X_1, \ldots, X_n \rangle^d = \langle \text{ all monomials of degree } d \, \rangle.$$

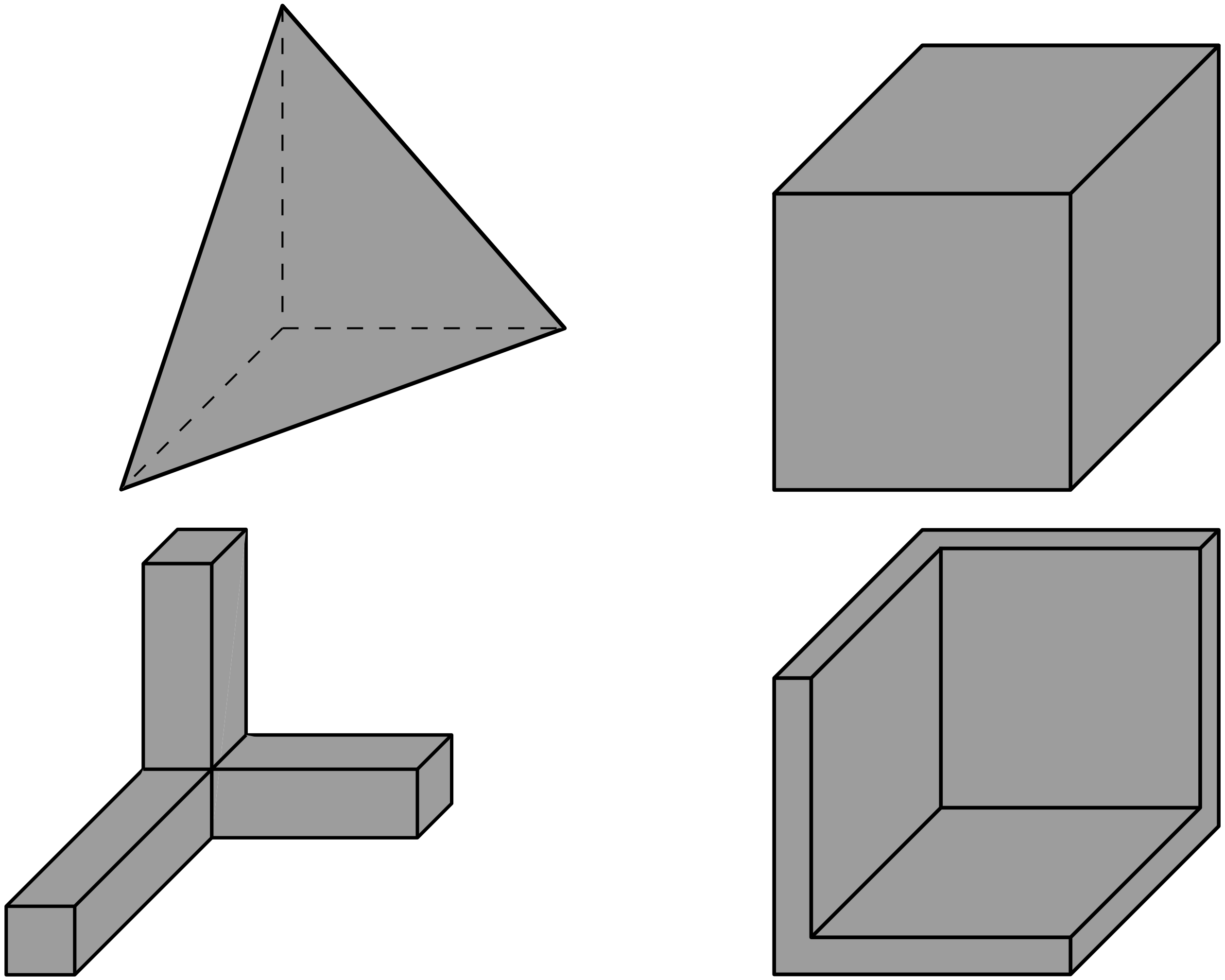
This is truncation in total degree.

Computations with algebraic numbers. Addition in characteristic p uses products truncated modulo $\langle X_1^p, \ldots, X_n^p \rangle$. This is truncation in partial degree.

Polynomial system solving. Lecerf's deflation algorithm for systems with multiplicities requires products modulo the "gradient" of the ideal

$$\langle \, X_1^{m_1}, \ldots, \, X_s^{m_s}, \, X_{s+1}, \, \ldots, \, X_n \, \rangle.$$

Pictorial representation



Some words on the complexity model

To measure the cost of multiplication algorithms, I will mention both bilinear and total complexity.

- Bilinear complexity only counts the “algebra operations”, that is, “essential multiplications”.

  If is often easier to estimate.

- Total complexity takes into account some linear operations as well.

Example: FFT in degree n.

evaluation at roots of 1 $\rightsquigarrow$ pairwise multiplications $\rightsquigarrow$ interpolation

Bilinear cost is n, total cost is $O(n \log(n))$.

Formally: computational model

Let D be an algebra with basis b, over a field k.

A **bilinear algorithm** is the data of

- $2s$ linear forms $f_1, \ldots, f_s$ and $h_1, \ldots, h_s$ over D,
- s elements $w_1, \ldots, w_s$ in D, such that the equality

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

  holds for all A, B in D.

Complexity measures:

- Bilinear complexity (rank) $= s$.
- Total complexity also counts the cost of computing $f_i(A)$, $h_i(B)$, and performing the recombination, in the basis b.

Overview

Let M be a zero-dimensional monomial ideal in $k[X_1, \ldots, X_n]$.

- The bilinear complexity of the product modulo M is

  $$O_{\log}(\mathrm{reg}_M \deg_M).$$

  reg_M : regularity of M, $\deg_M$: degree of M.

- Giving total complexity estimates requires to solve multivariate evaluation / interpolation problems.

- **Particular case:** truncation in partial degree. The total complexity is in

  $$O\left((\deg_M)^{1+\varepsilon}\right)$$

  for all ε.

Previous work

Few variables

1 variable.

- The bilinear complexity of the product modulo X_1^d (by FFT-like techniques) is the same as that of polynomials (Winograd, Fiduccia-Zalcstein), $\mathbf{2d-1}$.
- Improvements for some slower algorithms like Karatsuba or Toom-Cook (Mulders, Hanrot-Zimmermann).

2 variables.

- Truncation in total degree d (Schönhage, Bläser)

$$1.25\, d^2 \leq C_{\mathrm{Bilinear}} \leq 1.5\, d^2 \quad \deg_M \simeq 0.5\, d^2$$

- Truncation in partial degree d (Schönhage, Bläser)

$$2.33\, d^2 \leq C_{\mathrm{Bilinear}} \leq 3\, d^2 \quad \deg_M \simeq d^2$$

Several variables

Let T be the set of exponents in the monomial basis of $k[X_1, \ldots, X_n]/M$; then $|\mathsf{T}| = \deg_M$.

Naive product: the total complexity is

$$\sum_{t \in \mathsf{T}} (t_1 + 1) \cdots (t_n + 1).$$

Special cases

Truncation in total degree $\rightsquigarrow \frac{\deg_M^2}{n!}$.

Truncation in partial degree $\rightsquigarrow \frac{\deg_M^2}{2^n}$.

Using fast multivariate polynomial multiplication: for truncation in partial degree, $O_{\log}\left(2^n \deg_M\right)$.

Several variables, special case

Total degree. In total degree $d+1$, with n variables, the total complexity is

$$O_{\log}\left(\deg_M\right),$$

where $\deg_M = \binom{d+n}{n}$ and char $k=0$ (Lecerf-S.).

Previous result by Griewank, bilinear complexity only.

Generalization by van der Hoeven to weighted total degree truncation.

Improvements in the log factors by van der Hoeven, under some conditions on n, d, using his Truncated Fourier Transform.

Evaluation and interpolation in several variables

Monomial ideals and sets of points

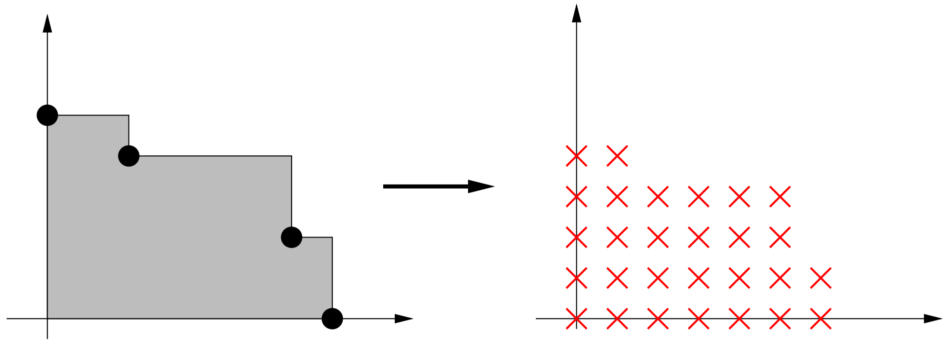
Setup. k is a field (infinite or large enough).

$M \subset k[X_1, \ldots, X_n]$ is the monomial ideal generated by some terms $g_1, \ldots, g_R$, where g_i does not divide g_j, $i \neq j$, and

$$g_i = X_1^{\delta_1^i} \cdots X_n^{\delta_n^i}.$$

The index set. $\mathsf{T} \subset \mathbb{N}^n$ is the set of exponents of the monomial basis of $k[X_1, \ldots, X_n]/M$.

Example: $M = \langle\, X_1^7,\ X_1^6 X_2^2,\ X_1^2 X_2^4,\ X_2^5 \,\rangle$



Some special sets of points

Values in k. Let d_i be minimal such that $X_i^{d_i}$ is in M.

For $i \le n$ and $j < d_i$, let $a_{i,j} \in k$ such that $a_{i,j} \neq a_{i,j'}$ for $j \neq j'$, and

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

The evaluation set. Define $\mathsf{A}_\mathsf{T} \subset \mathsf{A}_1 \times \cdots \times \mathsf{A}_n \subset k^n$ by

$$\mathsf{A}_\mathsf{T} = \{(a_{1,c_1}, \ldots, a_{n,c_n}) \mid (c_1, \ldots, c_n) \in \mathsf{T}\}.$$

Basic example: Take $a_{i,j} = j$. Then

$$\mathsf{A}_\mathsf{T} = \mathsf{T}.$$

An easy Gröbner basis

For $r \leq R$, recall that $g_r = X_1^{\delta_1^r} \cdots X_n^{\delta_n^r}$ and define

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

Example. Take $a_{i,j} = j$ for all i, j and $M = \langle\, X_1^{10},\, X_2^{10} \,\rangle$.

$$g_1 = X_1^{10} \mapsto G_1 = (X_1 - 0)(X_1 - 1) \cdots (X_1 - 9).$$

$$g_2 = X_2^{10} \mapsto G_2 = (X_2 - 0)(X_2 - 1) \cdots (X_2 - 9).$$

Prop. For any monomial order that refines degree,

- g_r is the leading term of G_r;
- $\langle G_1, \ldots, G_R \rangle$ is a Gröbner basis of $I(\mathsf{A}_\mathsf{T})$.

Macaulay, Hartshorne, Briançon-Galligo, Mora, Sauer ...

An evaluation / interpolation problem

Corollary. Let

$$\mathrm{Span}(\mathsf{T}) = \mathrm{Span}\{X_1^{t_1} \cdots X_n^{t_n} \mid (t_1, \ldots, t_n) \in \mathsf{T}\}.$$

It is the set of polynomials reduced with respect to M. Then map Eval

$$P \in \mathrm{Span}(\mathsf{T}) \mapsto [P(a) \mid a \in \mathsf{A}_\mathsf{T}]$$

is invertible. Let Interp be its inverse.

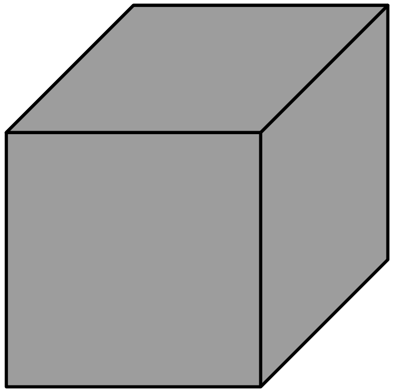
Question: How fast can we compute Eval and Interp?

In general, I don't know $\implies$ I write C_{Eval} and C_{Interp} for their complexity.

For some special cases, it becomes easier.

Classical example: $M = \langle X_1^{d_1}, \ldots, X_n^{d_n} \rangle$

Amounts to evaluation / interpolation on a rectangular grid.

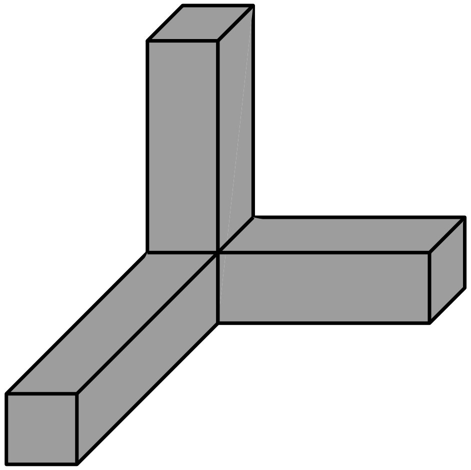


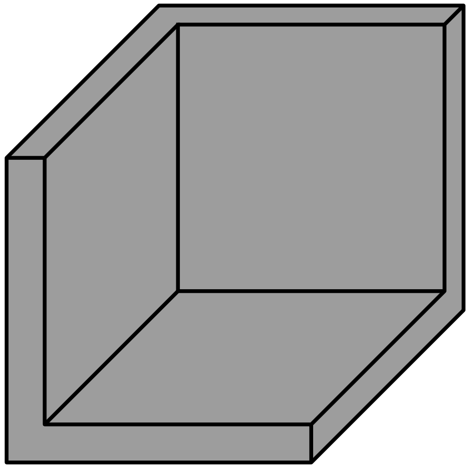
First evaluate X_n, then X_{n-1}, ...

- Choose primitive roots of 1 if available;
- else, choose points at a geometric progression;
- else, choose arbitrary points;

$\rightsquigarrow$ an algorithm in $O_{\log}(\deg_M) = O_{\log}(d_1 \cdots d_n)$.

Less classical examples





E.g., evaluation on the left-hand example, we would

- Split the “right-hand” arm from the rest of the body.
- Do some analogue of classical multipoint evaluation.
- Proceed recursively.

A general algorithm for these questions remains to be written.

Main results

Review of the setup

Let k be infinite or "large enough", and $n \geq 1$.

- Let $g_1, \ldots, g_R$ be terms in $k[X_1, \ldots, X_n]$, such that g_i does not divide g_j, $i \neq j$.

- Let $M = \langle g_1, \ldots, g_R \rangle$.

- Let T be the exponents of the terms not in M.

- Let $\deg_M = |\mathsf{T}|$ = degree of M.

- Let $\mathrm{reg}_M = 1 + \max\{|t| \text{ for } t \in \mathsf{T}\}$ = regularity of M.

- Let A_{T} be the set of points associated to T.

Complexity notation

- $\mathsf{M}_{\text{Bilinear}}(d)$ and $\mathsf{M}(d)$ are the bilinear and total cost of univariate polynomial multiplication in degree d.

  Using FFT,

  $$\mathsf{M}_{\text{Bilinear}}(d) = O(d\log(d)), \quad \mathsf{M}(d) = O(d\log(d)\log\log(d))$$

  Schönhage-Strassen, Cantor-Kaltofen.

Main results: general case

Theorem. The bilinear complexity of the multiplication in Q is upper bounded by

$$\mathsf{M}_{\mathrm{Bilinear}}(\mathrm{reg}_M)\,\deg_M\,.$$

Theorem. The total complexity is in

$$O\Big((C_{\mathsf{Eval}} + C_{\mathsf{Interp}})\mathrm{reg}_M + \mathsf{M}(\mathrm{reg}_M)\,\deg_M\Big).$$

What is it worth? An optimal result would be in $O(\deg_M)$.

Here, the best we could hope for would be in $O(\mathrm{reg}_M\,\deg_M)$.

This would require sharp results for C_{Eval} and C_{Interp}.

Main results: partial degree truncation

Theorem. Let $d_1, \ldots, d_n$ be in $\mathbb{N}_{>0}$. The total complexity of the product modulo $M = \langle X_1^{d_1}, \ldots, X_n^{d_n} \rangle$ is in

$$O_{\log}\left((d_1 + \cdots + d_n)\, d_1 \cdots d_n\right).$$

Corollary. For any $\varepsilon > 0$, the total complexity of the product modulo $M = \langle X_1^{d_1}, \ldots, X_n^{d_n} \rangle$ is in

$$O\left((d_1 \cdots d_n)^{1+\varepsilon}\right).$$

Idea of proof. Use Kronecker's substitution for large d_i's and the previous theorem for small ones.

Proof

Approximate algorithms

Let D be a k-algebra and ε a new indeterminate.

An **approximate (bilinear) algorithm** for the product in D is

- $2s$ linear forms $f_1, \ldots, f_s$ and $h_1, \ldots, h_s$ with coefficients in $k(\varepsilon)$
- s elements $w_1, \ldots, w_s$ in $D \otimes k(\varepsilon)$ (*i.e.* in D with coefficients in $k(\varepsilon)$), such that one has for all A, B in D

$$AB = \sum_{i=1}^{s} f_i(A) h_i(B) w_i \mod \varepsilon.$$

Origins: matrix multiplication.

- Bini *et al.*: approximate (floating-point) product.
- Bini: relation to exact computation.

Example: multiplication modulo X_1^2

Let $A = a_0 + a_1 X_1$ and $B = b_0 + b_1 X_1$. Then

$$AB \mod X_1^2 = \underbrace{(AB \mod (X_1^2 - \varepsilon^2))}_{C} \mod \varepsilon$$

1. Evaluation

$$A(\pm\varepsilon) = a_0 \pm a_1\varepsilon, \quad B(\pm\varepsilon) = b_0 \pm b_1\varepsilon$$

2. Pairwise products

$$C(\varepsilon) = A(\varepsilon)B(\varepsilon), \quad C(-\varepsilon) = A(-\varepsilon)B(-\varepsilon)$$

3. Interpolation

$$C = C(\varepsilon)\frac{X_1 + \varepsilon}{2\varepsilon} + C(-\varepsilon)\frac{-X_1 + \varepsilon}{2\varepsilon} = a_0b_0 + (a_0b_1 + a_1b_0)X_1 + \varepsilon^2 a_1 b_1.$$

Example: multiplication modulo X_1^d

The example generalizes to $k[X_1]/X_1^d$ using the deformation

$$k[X_1]/(X_1^d - \varepsilon^d) \equiv \prod_{i=0}^{d-1} k[X_1]/(X_1 - \omega^i \varepsilon),$$

where ω is a primitive dth root of 1.

We get the algorithm:

1. Evaluation at $\varepsilon, \omega\varepsilon, \ldots, \omega^{d-1}\varepsilon$.

2. Pairwise products.

3. Interpolation at $\varepsilon, \omega\varepsilon, \ldots, \omega^{d-1}\varepsilon$.

Complexity: $d \times \mathsf{M}(d)$.

A deformation for the general case

For $r \leq R$, recall that:

- $g_r = X_1^{\delta_1^r} \cdots X_n^{\delta_n^r}$
- $G_r = \prod_{i=1}^n \prod_{j=0}^{\delta_i^r - 1} (X_i - a_{i,j})$
- $\langle G_1, \ldots, G_R \rangle$ is the ideal of the set of point A_T.

The connexion between the two situations is done by introducing

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

$\Longrightarrow$ $\langle G_1^\varepsilon, \ldots, G_R^\varepsilon \rangle$ is the ideal of the set of point $\varepsilon \mathsf{A}_\mathsf{T}$.

Algorithm

Using

$$AB \mod \langle g_1, \ldots, g_R \rangle = (AB \mod \langle G_1^\varepsilon, \ldots, G_R^\varepsilon \rangle) \mod \varepsilon$$

and

$$k[X_1, \ldots, X_n]/\langle G_1^\varepsilon, \ldots, G_R^\varepsilon \rangle \equiv \prod_{a \in \mathsf{A}_\mathsf{T}} k[X_1, \ldots, X_n]/(X_1 - \varepsilon a_1, \ldots, X_n - \varepsilon a_n)$$

the product modulo $\langle g_1, \ldots, g_R \rangle$ is obtained by:

1. **Evaluation** at $[(\varepsilon a_1, \ldots, \varepsilon a_n) \mid a \in \mathsf{A}_\mathsf{T}]$.

2. **Pairwise products**.

3. **Interpolation** at $[(\varepsilon a_1, \ldots, \varepsilon a_n) \mid a \in \mathsf{A}_\mathsf{T}]$.

4. **Specializing** ε at 0.

Complexity estimates

Color code: red = Total complexity, blue = Bilinear complexity

1. **Evaluation:** polynomial in $k[\varepsilon]$ of degree $\leq \mathrm{reg}_M$ on $\mathsf{A_T}$.

   $\mathsf{Eval}(\mathsf{A_T})\mathrm{reg}_M$.

2. **Pairwise products:** $\deg_M$ products in $k[\varepsilon]$ in degree $\leq \mathrm{reg}_M$.

   $\mathsf{M}_{\mathrm{Bilinear}}(\mathrm{reg}_M)\deg_M$, $\mathsf{M}(\mathrm{reg}_M)\deg_M$.

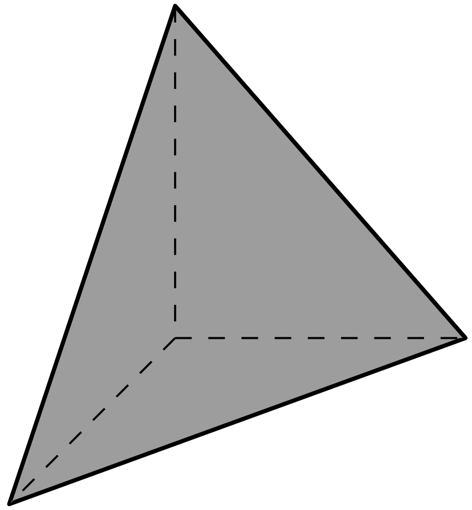
3. **Interpolation:** polynomial in $k[\varepsilon]$ of degree $\leq 2\mathrm{reg}_M$ on $\mathsf{A_T}$.

   $\mathsf{Interp}(\mathsf{A_T})\mathrm{reg}_M$.

4. **Specialization:** free.

Open questions

- Are there algorithms for evaluation / interpolation in $O(\deg_M)$ in the general case?
- In the following special case?



- Are evaluation and interpolation essentially equivalent as in the univariate case?
- Is it possible to explode the fat point into "less fat points"?

Application

Sum of two algebraic numbers

Example: determine that $\sqrt{2}+\sqrt{3}$ cancels X^4-10X^2+1.

General case: let

$$F=\prod_i (X-f_i), \quad G=\prod_j (X-g_j)$$

over a field k. Compute the polynomial

$$R=\prod_{i,j}(X-(f_i+g_j)).$$

R is the characteristic polynomial of $X+Y$ in $k[X,Y]/(F(X),G(Y))$ so it has coefficients in k.

Its degree is $\deg R=\deg(F)\deg(G)$.

Newton sums in characteristic 0

Consider the **Newton sums** of F, G and R

$$N_s(F) = \sum_i f_i^s, \quad N_s(G) = \sum_j g_j^s, \quad N_s(R) = \sum_{i,j} (f_i + g_j)^s.$$

Define the **exponential generating series**

$$N(F) = \sum_{s \geq 0} \frac{N_s(F)}{s!} X^s, \quad N(G) = \sum_{s \geq 0} \frac{N_s(G)}{s!} X^s, \quad N(R) = \sum_{s \geq 0} \frac{N_s(R)}{s!} X^s.$$

Then: $N(R) = N(F) N(G)$.

Algorithm (Dvornicich, Traverso).

1. Compute the Newton sums of F and G. $O_{\log}(\deg R)$

2. Perform the univariate power series product. $O_{\log}(\deg R)$

3. Recover R from its Newton sums. $O_{\log}(\deg R)$

Newton sums in small characteristic

Over $\mathbb{Z}/p\mathbb{Z}$, with $p < \deg R$, steps **2.** and **3.** are ill-defined, because of division by 0.

Workarounds.

2'. Multiplication of multivariate power series, modulo

$$M = \langle X_1^p, \ldots, X_n^p \rangle,$$

with $\deg_M = \deg R$.

3'. Use more Newton sums (Schönhage, Kaltofen-Pan, Pan).

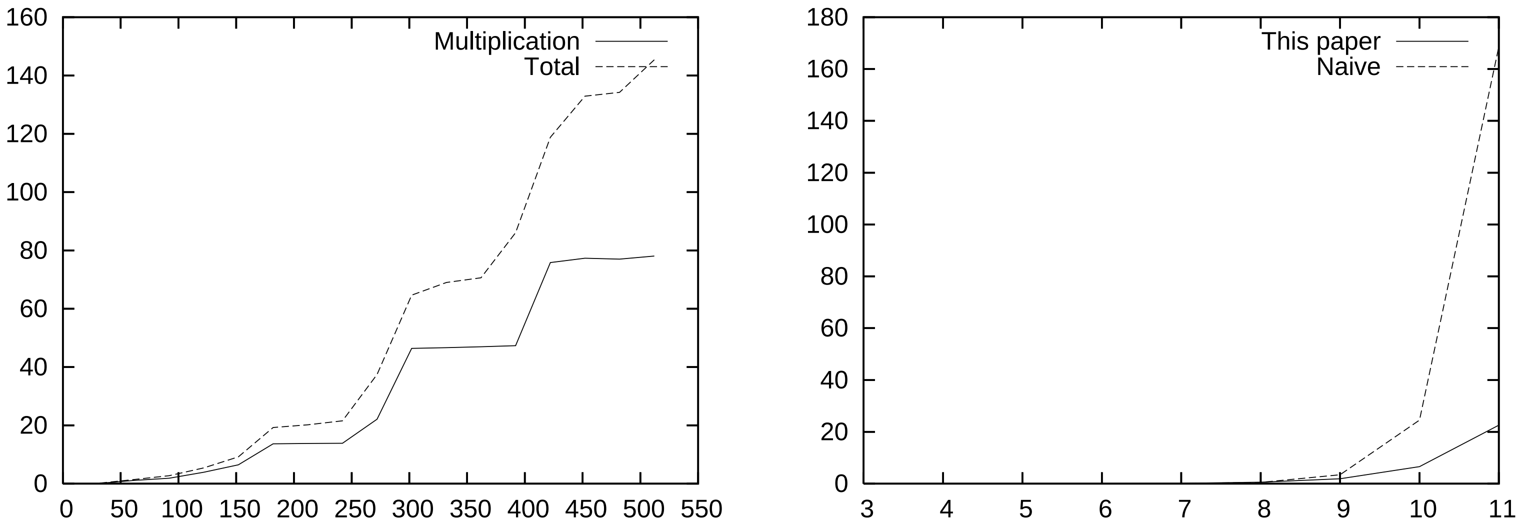
Th. In small characteristic (p fixed), the complexity of computing R is in $O_{\log}(\deg R)$ *bit* operations.

Tests in characteristic 3

When $p = 3$, in degree D, one computes series products modulo

$$\langle X_1^3, \ldots, X_n^3 \rangle,$$

where $n = \lceil \log_3(D) \rceil$.



All computations made using Shoup's NTL C++ library.