

# Deformation techniques for triangular arithmetic

A. Bostan, M. Chowdhury, J. van der Hoeven, É. Schost

# Triangular sets

Def.

- A **triangular set** is a family of polynomial  $\mathbf{T} = (T_1, T_2, \dots, T_n)$  in  $R[x_1, \dots, x_n]$ , where
  - $R$  a ring;
  - $T_i$  is in  $R[x_1, \dots, x_i]$ ;
  - $T_i$  is monic in  $X_i$ ;
  - $T_i$  is reduced modulo  $T_1, \dots, T_{i-1}$ .

Example

$$\begin{aligned}T_1(x_1) &= x_1^2 + 3x_1 \\T_2(x_1, x_2) &= x_2^2 + x_2x_1\end{aligned}$$

# Why?

## All-purpose polynomial system solving

- 1980's, 1990's: Wu, Kalkbrenner, Lazard, Wang, Aubry, Moreno Maza
- still a lot of hard questions, especially in positive dimension.

## The appropriate data-structure for a lot of more specialized questions

- often situations involving some form of symmetry;
- Galois theory, crypto, factoring in algebraic extensions, ...
- often in dimension zero.

# Goal of this work

## Our question.

- Given  $\mathbf{T}$  and  $A$  and  $B$  both reduced modulo  $\langle \mathbf{T} \rangle$ , how much does it cost to compute  $C = AB \bmod \langle \mathbf{T} \rangle$ ?

## Example (continued)

$$A = x_1x_2 + x_2 + x_1 + 1$$

$$B = x_1x_2 + x_2 + x_1 + 1$$

$$AB = x_1^2x_2^2 + 2x_2^2x_1 + 2x_2x_1^2 + 4x_1x_2 + x_2^2 + 2x_2 + x_1^2 + 2x_1 + 1$$

$$C = -6x_2x_1 + 2x_2 - x_1 + 1$$

# Why?

1. **General philosophy:** as with univariate polynomials, multiplication controls all other operations (especially when no splitting happens).

- inversion;
- resultant;
- Hensel lifting;
- change of order;
- ...

2. **Some concrete examples:**

- see later.

## Previous work

The complexity measure is

$$\delta = d_1 \cdots d_n, \quad \text{with} \quad d_i = \deg(T_i, x_i)$$

(this is essentially the input and output size).

**Theo.** [Li, Moreno Maza, Schost]

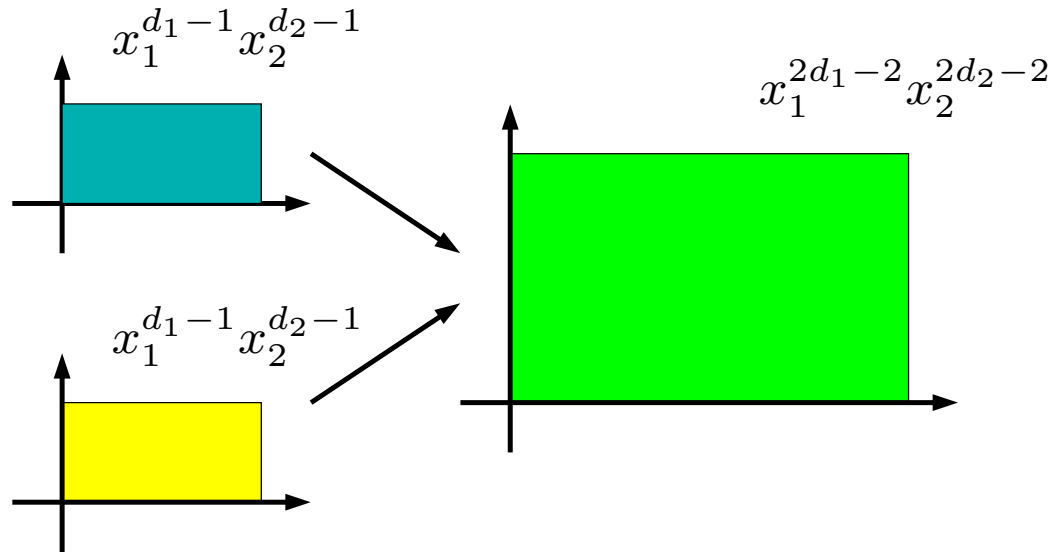
- The product  $AB \bmod \langle \mathbf{T} \rangle$  can be computed in time  $\tilde{O}(4^n \delta)$ .

The notation  $\tilde{O}$  hides logarithmic factors.

**Theo.** [Li, Moreno Maza, Schost]

- Let  $r = \max d_i$ .
- Suppose that for all  $i$ ,  $T_i$  is in  $R[x_i]$ , and that  $R$  has  $r$  interpolation points. Then the product  $AB \bmod \langle \mathbf{T} \rangle$  can be computed in time  $\tilde{O}(\delta r)$ .

# The problem with multivariate polynomials



In  $n$  variables,

- **polynomial** multiplication induces a  $2^n$  overhead;
- so **triangular** multiplication should avoid expansion.
- **Evaluation / interpolation** is the key.

# This work

We extend the second approach of [Li et al.](#) to a few more general situations.

- The **support** of the polynomials determines the complexity;
- for nice cases, we get results of the same form

$$O^{\sim}(\delta r) \quad \text{with} \quad r = \sum d_i.$$

## Remark

- For  $d_i = 2$  and  $\delta = 2^n$ , taking the best of both approaches ([general/specialized](#)) gives

$$\delta \left( e^{\sqrt{\log \delta}} \right)^{O(1)}.$$

# Examples

# Power series

The [monomial ideal](#)

$$\mathbf{T} = \begin{array}{|l} x_n^{d_n} \\ \vdots \\ x_2^{d_2} \\ x_1^{d_1} \end{array}$$

is triangular.

No quasi-linear algorithm was known until [Schost, 2005](#).

# Polynomial multiplication

## Reductions:

- to multiply **polynomials** of degree  $d$ , enough to multiply **series** mod  $x^d$ ;
- we can take  $d = 2^k$ .

## From uni- to multivariate (with $d = 8 = 2^3$ )

- write  $x = x_0$ , and introduce  $x_1, x_2$ ;
- use the equality between ideals

$$\left| \begin{array}{c} x_2 - x_0^4 \\ x_1 - x_0^2 \\ x_0^8 \end{array} \right| = \left| \begin{array}{c} x_0^2 - x_1 \\ x_1^2 - x_2 \\ x_2^2 \end{array} \right|$$

- change of basis is free via base-2 decomposition of indices.

# Exponential generating series multiplication

Let  $(a_i)_{i \leq d}$ ,  $(b_i)_{i \leq d}$  and  $(c_i)_{i \leq d}$  be sequences that satisfy

$$c_i = \sum_j \binom{i}{j} a_j b_{i-j} \quad i \leq d.$$

If we were over  $\mathbb{Q}$ , this would mean

$$\sum_{i \leq d} \frac{c_i}{i!} x^i = \sum_{i \leq d} \frac{a_i}{i!} x^i \sum_{i \leq d} \frac{b_i}{i!} x^i \pmod{x^d}.$$

Here, we suppose that

- all odd integers can be inverted in the base ring;  
examples:  $\mathbb{Q}$ , any  $\text{GF}(2^m)$ , any  $\mathbb{Z}/2^\ell\mathbb{Z}$
- and also that  $d = 2^k$ .

# Reduction to multivariate multiplication

For  $i \geq 0$ , let

$$i_{\star} = \frac{i!}{\text{largest power of 2 that divides } i!}$$

Prop.

- The  $c_i$  are given by

$$\sum_i \frac{c_i}{i_{\star}} x_0^{i_0} \cdots x_{k-1}^{i_{k-1}} = \sum_i \frac{a_i}{i_{\star}} x_0^{i_0} \cdots x_{k-1}^{i_{k-1}} \sum_i \frac{b_i}{i_{\star}} x_0^{i_0} \cdots x_{k-1}^{i_{k-1}} \pmod{\langle \mathbf{T} \rangle},$$

with

$$\mathbf{T} = \left| \begin{array}{l} x_0^2 - 2x_1 \\ \vdots \\ x_{k-2}^2 - 2x_{k-1} \\ x_{k-1}^2 \end{array} \right. \quad \text{and} \quad i = i_0 + 2i_1 + \cdots + 2^{k-1}i_{k-1}$$

# The canonical example

Addition of algebraic numbers:

$$f = \prod_i (x - f_i), \quad g = \prod_j (x - g_j), \quad h = \prod_{i,j} (x - (f_i + g_j)).$$

To compute  $h$ :

- compute the Newton sums of  $f$  and  $g$ ;
- deduce those of  $h$  by exponential generating series product;
- get  $h$  by exponentiation.

In **small characteristic**, say over  $\mathbb{F}_2$ :

- we compute the Newton sums modulo a higher power of 2;
- the exponential series product is done multivariate-ly.

# Artin-Schreier

To handle **degree- $p$  extensions**

- of the form  $\mathbb{F}[x]/\langle x^p - x - \alpha \rangle$ ,
- where  $\text{char}(\mathbb{F}) = p$ .

Algorithms such as **Couveignes'** generate **towers**

$$\mathbf{T} = \left| \begin{array}{l} x_n^p - x_n - \alpha_n(x_1, \dots, x_{n-1}) \\ \vdots \\ x_2^p - x_2 - \alpha_2(x_1) \\ x_1^p - x_1 - \alpha_1 \end{array} \right.$$

and we have to compute modulo such  $\mathbf{T}$ 's.

# Summary

## Multivariate power series

- $T_i = x_i^{d_i}$

## Polynomial multiplication

- $T_i = x_i^2 - x_{i-1}$

## Exponential generating series

- $T_i = x_i^2 - 2x_{i-1}$

## Artin-Schreier (over $\mathbb{F}_2$ )

- $T_i = x_i^2 + x_i + \alpha_i(x_1, \dots, x_{i-1})$

# The split case

# The nice case

When all polynomials **split into linear factors**, fast algorithms are available. Precisely, suppose that there exists:

- $a_1, \dots, a_{d_1}$  such that

$$T_1 = (x_1 - a_1) \cdots (x_1 - a_{d_1})$$

- for all  $i \leq d_1$ ,  $a_{i,1}, \dots, a_{i,d_2}$  such that

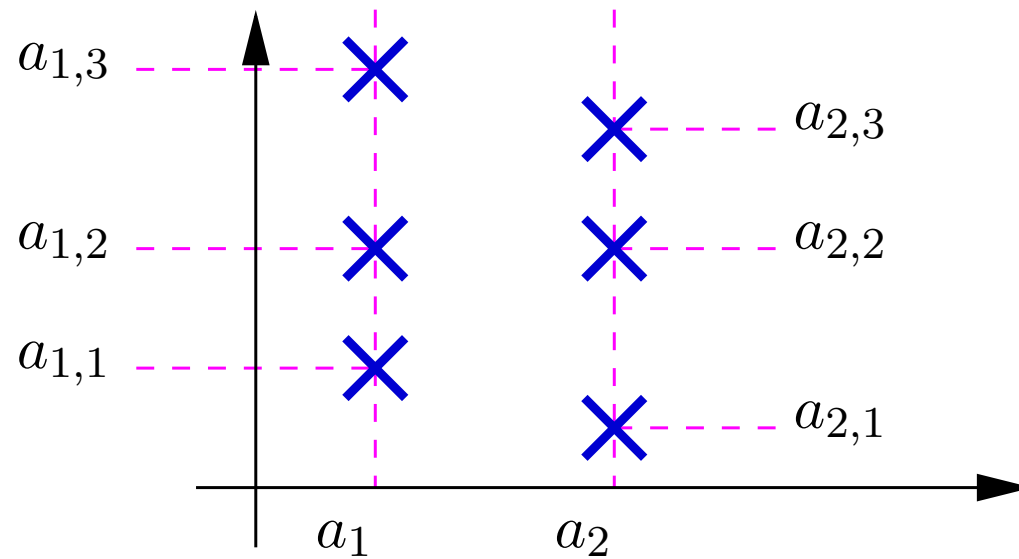
$$T_2(a_i, x_2) = (x_2 - a_{i,1}) \cdots (x_2 - a_{i,d_2})$$

- for all  $i \leq d_1, j \leq d_2$ ,  $a_{i,j,1}, \dots, a_{i,j,d_3}$  such that

$$T_3(a_i, a_j, x_3) = (x_3 - a_{i,j,1}) \cdots (x_3 - a_{i,j,d_3})$$

- ... up to  $n$ .

## What this means



The points  $P_{i,j,k,\dots} = (a_i, a_{i,j}, a_{i,j,k}, \dots)$  are the coordinates of the points

$$V = V(T_1, \dots, T_n).$$

# Modular multiplication

Remember that in one variable,

$$\text{FFT multiplication} = \text{multiplication mod } x^d - 1$$

Hence, to multiply  $A, B$  modulo  $\langle \mathbf{T} \rangle$ :

- evaluate  $A, B$  at  $V$ ;
- multiply the values;
- interpolate the result.

Extra condition

- the differences  $a_i - a_{i'}, \dots$  should all be units.

# Evaluation and interpolation

Similar to **multidimensional FFT**:

- Evaluate  $A$  at  $x_1 = a_1, \dots, a_{d_1}$ . We get  $A_1, \dots, A_{d_1}$ .  
Cost:  $M(d_1) \log(d_1) \times (d_2 \cdots d_n) = \delta \frac{M(d_1) \log(d_1)}{d_1}$ .
- For  $i \leq d_1$ , evaluate  $A_i$  at  $a_{i,1}, \dots, a_{i,d_2}$ . We get  $A_{i,1}, \dots, A_{i,d_2}$ .  
Cost:  $d_1 M(d_2) \log(d_2) \times (d_3 \cdots d_n) = \delta \frac{M(d_2) \log(d_2)}{d_2}$ .
- ... up to  $n$ .

$$\text{Total cost: } \delta \sum_i \frac{M(d_i) \log(d_i)}{d_i} \leq \delta \log^2(\delta) \log \log(\delta).$$

# The general case

# Deformation

Given  $\mathbf{T} = (T_1, \dots, T_n)$  in  $R[x_1, \dots, x_n]$ .

- Build  $\mathbf{U} = (U_1, \dots, U_n)$ , which **completely splits**, with pairwise distinct roots.
  - constraint:  $\deg(U_i, X_i) = \deg(T_i, X_i) = d_i$ ;
  - other than that, we are free to pick  $\mathbf{U}$  as we wish;
  - this requires that the base field has enough elements.
- Build  $\mathbf{V} = (V_1, \dots, V_n)$ , with

$$V_i = \eta T_i + (1 - \eta)U_i.$$

## Algorithm

- compute the product modulo  $\langle \mathbf{V} \rangle$ ;  
by **evaluation / interpolation**
- let  $\eta = 1$  in the result.

# The roots of $\mathbf{V}$

## Hensel's lemma

- Because  $\text{subs}(\eta = 0, \mathbf{V}) = \mathbf{U}$ , and because the roots of  $\mathbf{U}$  are simple in  $R^n$ ,  $\mathbf{V}$  admits pairwise distinct roots in  $R[[\eta]]^n$ .

## Cost to reach some precision $r$ .

- Lift the  $x_1$ -coordinates  
 $M(d_1) \log(d_1) M(r)$
  - Substitute them in  $V_2, \dots, V_n$   
 $M(d_1) \log(d_1) (d_2 + d_2 d_3 + \dots + d_2 \dots d_n) M(r)$
  - Lift the  $x_2$ -coordinates  
 $d_1 M(d_2) \log(d_2) M(r)$
  - Substitute them in  $V_3, \dots, V_n$   
 $d_1 M(d_2) \log(d_2) (d_2 d_3 + \dots + d_2 \dots d_n) M(r)$
  - ...
- }  $M(r) \delta \log^2(\delta) \log \log(\delta).$

# Overall complexity

## Precision

- Let  $r = \deg(AB \bmod \langle \mathbf{V} \rangle, \eta)$ .
- All computations done mod  $\eta^{r+1}$ .

No loss of precision in the algorithm

## Total cost

- Given  $\mathbf{T}$  and the roots of  $\mathbf{U}$ ,

$$M(r) \delta \sum_i \frac{M(d_i) \log(d_i)}{d_i} \quad i.e. \quad M(r) \delta \log^2(\delta) \log \log(\delta)$$

(constructing  $\mathbf{U}$  is negligible)

# Bounding the precision

# Basic remarks

## Remarks.

- The needed precision  $r$  is

$$\deg(x_1^{2d_1-2} \cdots x_n^{2d_n-2} \bmod \langle \mathbf{V} \rangle, \eta).$$

- In general,  $r = \delta$ , so this approach is useless.
- $r$  should be smaller for cases with structure.

## Example

- Suppose the  $T_i$  and  $U_i$  are univariate.
- Then  $V_i = x_i^{d_i} + v_{i,d_i-1}x_i^{d_i-1} + \cdots + v_{i,0}$ , linear in  $\eta$ .
- Each reduction **decrements** one partial degree and **increments** the degree in  $\eta$ .
- So  $r = \sum_i (d_i - 1)$ .

# Recurrence relations

Let  $\mathbf{e} = (e_1, \dots, e_n)$  and

$$W(\mathbf{e}) = \deg(x_1^{e_1} \cdots x_n^{e_n} \bmod \langle \mathbf{V} \rangle, \eta).$$

The **support** of  $\mathbf{V}$  determines **recurrence relations** on  $W$ . Write

$$V_n = x_n^{d_n} + \sum_{i \leq S} v_{\alpha_i} \mathbf{x}^{\alpha_i},$$

for some exponents  $\alpha_i$  in  $\mathbb{N}^n$  and coefficients  $v_{\alpha_i}$  linear in  $\eta$ .

Then,

- $W(\mathbf{e}) = W(e_1, \dots, e_{n-1})$  for  $e_n < d_n$ ;
- else,

$$W(\mathbf{e}) = 1 + \max_i W(\mathbf{e} + \beta_i), \quad \beta_i = \alpha_i - (0, \dots, 0, d_n).$$

## Simplest case

With  $T_i = x_i^2 + x_{i-1}$ , we take in characteristic  $\neq 2$

$$U_i = x_i^2 - 1 \implies V_i = x_i^2 + \eta x_{i-1} - (1 - \eta).$$

### Initial conditions

- $W(e_1) = \lfloor e_1/2 \rfloor \leq e_1/2$

### Recurrence

- for  $e_n = 0, 1$

$$W(e_1, \dots, e_n) = W(e_1, \dots, e_{n-1}).$$

- for  $e_n \geq 2$

$$W(e_1, \dots, e_n) \leq 1 + \max W(e_1, \dots, e_n - 2), W(e_1, \dots, e_{n-1} + 1, e_n - 2)$$

or

$$W(e_1, \dots, e_n) \leq 1 + W(e_1, \dots, e_{n-1} + 1, e_n - 2).$$

# Simplest case

Unrolling until  $e_n = 0, 1$

- $W(e_1, \dots, e_n) = \lfloor e_n/2 \rfloor + W(e_1, \dots, e_{n-1} + \lfloor e_n/2 \rfloor)$

Ansatz: assume

$$W(e_1, \dots, e_{n-1}) \leq w_1 e_1 + \dots + w_{n-1} e_{n-1}, \quad w_i \geq 0.$$

Then

$$W(e_1, \dots, e_n) \leq \frac{e_n}{2} + w_1 e_1 + \dots + w_{n-1} e_{n-1} + w_{n-1} \frac{e_n}{2}$$

so we also have sub-linearity in  $n$  variables with  $w_n = (w_{n-1} + 1)/2$ .

Finally,

$$r \leq \sum_{i \leq n} 2w_i \quad \text{with} \quad w_1 = \frac{1}{2}, \quad w_i = \frac{w_{i-1} + 1}{2},$$

so  $r \in O(n)$ .

# Simple case

With  $T_i = x_i^2 + x_{i-1}$ , we take in characteristic **2**

$$U_i = x_i^2 - x_i \implies V_i = x_i^2 + \eta x_{i-1} - (1 - \eta)x_i.$$

## Initial conditions

- $W(e_1) = e_1 - 1 \leq e_1$ .

## Recurrence

- for  $e_n = 0, 1$

$$W(e_1, \dots, e_n) = W(e_1, \dots, e_{n-1}).$$

- for  $e_n \geq 2$

$$W(e_1, \dots, e_n) \leq 1 + \max W(e_1, \dots, e_n - 1), W(e_1, \dots, e_{n-1} + 1, e_n - 2)$$

# Simple case

Unrolling until  $e_n = 0, 1$

- We can follow the directions

$$\beta_1 = (0, \dots, 0, -1), \quad \beta_2 = (0, \dots, 1, -2)$$

- We do  $a_1$  steps in the first direction,  $a_2$  steps in the second one.

When we stop, the  $n$ -coordinate is either 0 or 1:

$$e_n - a_1 - 2a_2 \in \{0, 1\}$$

So,

$$W(\mathbf{e}) \leq \max_{\substack{a_1, a_2 \in \mathbb{N} \\ e_n - a_1 - 2a_2 \in \{0, 1\}}} a_1 + a_2 + W(\mathbf{e} + a_1\beta_1 + a_2\beta_2)$$

# Simple case

Induction: assume

$$W(e_1, \dots, e_{n-1}) \leq w_1 e_1 + \dots + w_{n-1} e_{n-1}, \quad w_i \geq 0$$

Then,

$$\begin{aligned} W(\mathbf{e}) &= \max_{\substack{a_1, a_2 \in \mathbb{N} \\ e_n - a_1 - 2a_2 \in \{0, 1\}}} a_1 + a_2 + w_1 e_1 + \dots + w_{n-1} e_{n-1} + w_{n-1} a_2 \\ &\leq \max_{\substack{a_1, a_2 \in \mathbb{R}_{\geq 0} \\ e_n - a_1 - 2a_2 = 0}} a_1 + a_2 + w_1 e_1 + \dots + w_{n-1} e_{n-1} + w_{n-1} a_2 \end{aligned}$$

- The max is either at  $(a_1 = 0, a_2 = e_n/2)$  or  $(a_1 = e_n, a_2 = 0)$ .
- $w_n = \max 1, (w_{n-1} + 1)/2$
- as before,  $r = O(n)$ .

## In general

The  $w_i$ 's satisfy the recurrence

$$w_n = \max_i \frac{w_1\beta_{i,1} + \cdots + w_{n-1}\beta_{i,n-1} + 1}{-\beta_{i,n}}$$

Artin-Schreier, with  $p = 2$

$$V_n = x_n^2 + x_n + \eta x_1 \cdots x_{n-1} + \cdots$$

- $w_1 = 1$
- $w_n = (w_1 + \cdots + w_{n-1} + 1)/2$
- $r = O(\frac{3}{2}^n)$ .