Coppersmith's algorithm and polynomial equations

Éric Schost
University of Waterloo
eschost@uwaterloo.ca

Plan of the talk

1. **Wiedemann's algorithm**

2. **Blocking**

3. **Structured projections**

4. **Bonus: more examples**

1. Wiedemann's algorithm

Wiedemann's algorithm

$\boldsymbol{A}$ is a matrix in $\mathbb{K}^{\boldsymbol{D}\times\boldsymbol{D}}$.

- compute $2D$ terms $a_i = \boldsymbol{u}^T\boldsymbol{A}^i\boldsymbol{v}$, for random $\boldsymbol{u}, \boldsymbol{v}$ in $\mathbb{K}^{\boldsymbol{D}\times\mathbf{1}}$
- find the minimal polynomial of (a_i)
- (optional) use it to solve $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{y}$

Example

$$\boldsymbol{A} = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \quad \boldsymbol{u} = \begin{bmatrix} 1 & 2 \end{bmatrix} \quad \boldsymbol{v} = \begin{bmatrix} 1 \\ -2 \end{bmatrix}$$

sequence: $a_0 = -3, \quad a_1 = -13, \quad a_2 = -71, \quad a_3 = -381, \quad a_4 = -2047, \ldots$
recurrence: $a_{n+2} - 5a_{n+1} - 2a_n$
minimal polynomial: $X^2 - 5X - 2$.

Wiedemann. Solving sparse linear equations over finite fields (1986).

Some interesting matrices

Context

- $I = \langle f_1, \ldots, f_s \rangle$ ideal in $\mathbb{K}[X_1, \ldots, X_n]$
- I has dimension zero $V(I) = \{\boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_D\}$
- I separable (= radical over $\overline{\mathbb{K}}$) no multiplicities

Then:

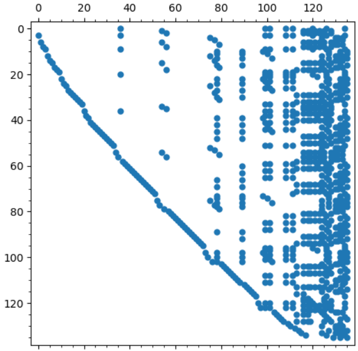
- $\mathbb{A} = \mathbb{K}[X_1, \ldots, X_n]/I$ has dimension D, basis $\mathscr{B} = (b_1, \ldots, b_D)$.
- any $a \in \mathbb{K}[X_1, \ldots, X_n]$ has a **multiplication matrix** in $\mathbb{A}$:

$$\boldsymbol{M}_a = \begin{bmatrix} & \vdots & \\ \cdots & \mathrm{coeff}(ab_j, b_i) & \cdots \\ & \vdots & \end{bmatrix} \simeq_{\mathrm{CRT}} \begin{bmatrix} a(\boldsymbol{\alpha}_1) & & \\ & \ddots & \\ & & a(\boldsymbol{\alpha}_D) \end{bmatrix}.$$

Large n

Solving polynomial equations:

- obtain $\mathbb{A}$ and $\mathscr{B}$ from a degree Gröbner basis computation
- some multiplication matrices look **sparse** (complicated structure)



Faugère, Mou. Sparse FGLM algorithms (2017).

Berthomieu, Neiger, Safey El Din. Faster change of order algorithm for Gröbner bases under shape and stability assumptions (2022).

Small n

Many algorithms (finite field isomorphism, irreducibility) … use $\boldsymbol{n = 1}$:

- frequent case: $\boldsymbol{I = \langle f(X) \rangle}$ in $\mathbb{K}[X]$
- use multiplication matrices that are **structured**, but **not necessarily** sparse.

Example

with $\boldsymbol{f = 7 + 49X + 100X^2 + 51X^3 + 8X^4 + X^5}$ in $\mathbb{F}_{101}[X]$
and $\boldsymbol{a = 73 + 97X + 25X^2 + 49X^3 + 84X^4}$

$$\boldsymbol{M}_X = \begin{bmatrix} 0 & 0 & 0 & 0 & 94 \\ 1 & 0 & 0 & 0 & 52 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 50 \\ 0 & 0 & 0 & 1 & 93 \end{bmatrix}$$

Small n

Many algorithms (finite field isomorphism, irreducibility) … use $\boldsymbol{n = 1}$:

- frequent case: $\boldsymbol{I} = \langle \boldsymbol{f(X)} \rangle$ in $\mathbb{K}[X]$
- use multiplication matrices that are **structured**, but **not necessarily** sparse.

Example

with $\boldsymbol{f = 7 + 49X + 100X^2 + 51X^3 + 8X^4 + X^5}$ in $\mathbb{F}_{101}[X]$
and $\boldsymbol{a = 73 + 97X + 25X^2 + 49X^3 + 84X^4}$

$$\boldsymbol{M}_a = \begin{bmatrix} 73 & 18 & 18 & 76 & 35 \\ 97 & 98 & 43 & 45 & 18 \\ 25 & 80 & 81 & 61 & 40 \\ 49 & 84 & 38 & 72 & 13 \\ 84 & 84 & 18 & 96 & 11 \end{bmatrix} = \begin{bmatrix} \boldsymbol{a} & \boldsymbol{M}_X \boldsymbol{a} & \boldsymbol{M}_X^2 \boldsymbol{a} & \boldsymbol{M}_X^3 \boldsymbol{a} & \boldsymbol{M}_X^4 \boldsymbol{a} \end{bmatrix}$$

Thiong Ly. Note for computing the minimum polynomial of elements in large finite fields (1988).

Back to Wiedemann

Consider the Wiedemann sequence $\boldsymbol{u}^T \boldsymbol{M}_a^i \boldsymbol{v}$, where

- $\boldsymbol{M}_a$ is the multiplication matrix by $\mathbf{a} \in \mathbb{A}$
- $\boldsymbol{v}$ is the coefficient vector of $\mathbf{g} \in \mathbb{A}$
- $\boldsymbol{u}$ is the coefficient vector of a linear form $\boldsymbol{\ell} : \mathbb{A} \to \mathbb{K}$

Then,

$$\boldsymbol{u}^T \boldsymbol{M}_a^i \boldsymbol{v} = \ell(a^i g).$$

Chinese Remainder Theorem: there are constants $\ell_1, \ldots, \ell_D$ such that

$$\ell = \ell_1 \mathrm{Ev}_{\boldsymbol{\alpha}_1} + \cdots + \ell_D \mathrm{Ev}_{\boldsymbol{\alpha}_D},$$

so

$$\boldsymbol{u}^T \boldsymbol{M}_a^i \boldsymbol{v} = \ell_1 \, a(\boldsymbol{\alpha}_1)^i g(\boldsymbol{\alpha}_1) + \cdots + \ell_D \, a(\boldsymbol{\alpha}_D)^i g(\boldsymbol{\alpha}_D).$$

Looking at the generating series

$$S_{\ell,g} := \sum_{i\geq 0} \frac{\boldsymbol{u}^T\,\boldsymbol{M}_a^i\,\boldsymbol{v}}{X^{i+1}} = \frac{\ell_1 g(\boldsymbol{\alpha}_1)}{X - a(\boldsymbol{\alpha}_1)} + \cdots + \frac{\ell_D g(\boldsymbol{\alpha}_D)}{X - a(\boldsymbol{\alpha}_D)}$$

$$= \frac{N_{\ell,g}(X)}{\mathrm{LCM}\big(X - a(\boldsymbol{\alpha}_1), \ldots, X - a(\boldsymbol{\alpha}_D)\big)}$$

Looking at the generating series

$$S_{\ell,g} := \sum_{i\geq 0} \frac{\boldsymbol{u}^T \boldsymbol{M}_a^i \boldsymbol{v}}{X^{i+1}} = \frac{\ell_1 g(\boldsymbol{\alpha}_1)}{X - a(\boldsymbol{\alpha}_1)} + \cdots + \frac{\ell_D g(\boldsymbol{\alpha}_D)}{X - a(\boldsymbol{\alpha}_D)}$$
$$= \frac{N_{\ell,g}(X)}{\mathrm{LCM}\big(X - a(\boldsymbol{\alpha}_1), \ldots, X - a(\boldsymbol{\alpha}_D)\big)}$$

1. for **generic ℓ**, the denominator of $S_{\ell,1}$ is the minimal polynomial of a

2. if also **the $a(\alpha_i)$'s are all distinct**, true for generic a

- the residue of $S_{\ell,1}$ at $a(\boldsymbol{\alpha}_i)$ is ℓ_i
- the residue of $S_{\ell,g}$ at $a(\boldsymbol{\alpha}_i)$ is $\ell_i g(\boldsymbol{\alpha}_i)$
- so the **numerators** $N_{\ell,1}$ and $N_{\ell,g}$ will give h such that $g = h(a)$

Looking at the generating series

$$S_{\ell,g} := \sum_{i \geq 0} \frac{\boldsymbol{u}^T \boldsymbol{M}_a^i \boldsymbol{v}}{X^{i+1}} = \frac{\ell_1 g(\boldsymbol{\alpha}_1)}{X - a(\boldsymbol{\alpha}_1)} + \cdots + \frac{\ell_D g(\boldsymbol{\alpha}_D)}{X - a(\boldsymbol{\alpha}_D)}$$
$$= \frac{N_{\ell,g}(X)}{\mathrm{LCM}\big(X - a(\boldsymbol{\alpha}_1), \ldots, X - a(\boldsymbol{\alpha}_D)\big)}$$

Shoup. Fast construction of irreducible polynomials over finite fields (1994).

Rouillier. Solving zero-dimensional systems through the Rational Univariate Representation (1999).

Bostan, Salvy, S. Fast algorithms for zero-dimensional polynomial systems using duality (2003).

Example: primitive element for $\mathbb{Q}(\sqrt{2}, \sqrt{3})$

Consider $\boldsymbol{I = \langle X_1^2 - 2, X_2^2 - 3 \rangle}$ in $\mathbb{Q}[X_1, X_2]$, so that

$$\mathbb{A} = \mathbb{Q}[X_1, X_2]/I = \mathrm{Span}(1, X_1, X_2, X_1 X_2)$$

Choose

- $\boldsymbol{a = X_1 + X_2}$
- $\boldsymbol{\ell(f_0 + f_1 X_1 + f_2 X_2 + f_3 X_1 X_2) = f_0}$
- $\boldsymbol{g = X_1}$.

Example: primitive element for $\mathbb{Q}(\sqrt{2}, \sqrt{3})$

Consider $\boldsymbol{I = \langle X_1^2 - 2, X_2^2 - 3\rangle}$ in $\mathbb{Q}[X_1, X_2]$, so that

$$\mathbb{A} = \mathbb{Q}[X_1, X_2]/I = \mathrm{Span}(1, X_1, X_2, X_1X_2)$$

Choose

- $\boldsymbol{a = X_1 + X_2}$
- $\boldsymbol{\ell(f_0 + f_1X_1 + f_2X_2 + f_3X_1X_2) = f_0}$
- $\boldsymbol{g = X_1}$.

We get

$$\begin{aligned} \boldsymbol{S_{\ell,1}} &= \textstyle\sum_{i\geq 0} \frac{\ell(a^i)}{X^{i+1}} &&= \frac{1}{X} + \frac{5}{X^3} + \frac{49}{X^5} + \cdots &&= \boldsymbol{\frac{-5X+X^3}{1-10X^2+X^4}} \\ \boldsymbol{S_{\ell,X_1}} &= \textstyle\sum_{i\geq 0} \frac{\ell(a^iX_1)}{X^{i+1}} &&= \frac{2}{X^2} + \frac{22}{X^4} + \cdots &&= \boldsymbol{\frac{2+2X^2}{1-10X^2+X^4}}. \end{aligned}$$

Example: primitive element for $\mathbb{Q}(\sqrt{2}, \sqrt{3})$

Consider $\boldsymbol{I = \langle X_1^2 - 2, X_2^2 - 3\rangle}$ in $\mathbb{Q}[X_1, X_2]$, so that

$$\mathbb{A} = \mathbb{Q}[X_1, X_2]/I = \mathrm{Span}(1, X_1, X_2, X_1X_2)$$

Choose

- $\boldsymbol{a = X_1 + X_2}$
- $\boldsymbol{\ell(f_0 + f_1X_1 + f_2X_2 + f_3X_1X_2) = f_0}$
- $\boldsymbol{g = X_1}$.

We get

$$\begin{aligned} \boldsymbol{S_{\ell,1}} &= \textstyle\sum_{i\geq 0} \frac{\ell(a^i)}{X^{i+1}} &&= \tfrac{1}{X} + \tfrac{5}{X^3} + \tfrac{49}{X^5} + \cdots &&= \boldsymbol{\tfrac{-5X+X^3}{1-10X^2+X^4}} \\ \boldsymbol{S_{\ell,X_1}} &= \textstyle\sum_{i\geq 0} \frac{\ell(a^iX_1)}{X^{i+1}} &&= \tfrac{2}{X^2} + \tfrac{22}{X^4} + \cdots &&= \boldsymbol{\tfrac{2+2X^2}{1-10X^2+X^4}}. \end{aligned}$$

Set $h = (2+2X^2)/(-5X+X^3) \bmod (1-10X^2+X^4) = \frac{1}{2}X - \frac{9}{2}X^3$;

then

$$h(\sqrt{2}+\sqrt{3}) = \sqrt{2}$$

Complexity issues

Bottleneck: computing $\boldsymbol{u}^T \, \boldsymbol{M}_a^i \, \boldsymbol{v} = \ell(a^i g)$, $i = 0, \ldots, 2D$

- if $\boldsymbol{M}_a$ sparse ($O(D)$ entries) $\boldsymbol{O(D^2)}$
  (conjecturally **not** the case in general when solving polynomial systems)
- if $n = 1$, use **modular composition** techniques $\boldsymbol{O(D^{(\omega+1)/2})}$
  (ω is the matrix multiplication exponent)

Brent, Kung. Fast algorithms for manipulating formal power series (1978).

Shoup. Fast construction of irreducible polynomials over finite fields (1994).

Complexity issues

Bottleneck: computing $\boldsymbol{u}^T\, \boldsymbol{M}_a^i\, \boldsymbol{v} = \ell(a^i g)$, $i = 0, \ldots, 2D$

- if $\boldsymbol{M}_a$ sparse ($O(D)$ entries) $\boldsymbol{O(D^2)}$
  (conjecturally **not** the case in general when solving polynomial systems)
- if $n = 1$, use **modular composition** techniques $\boldsymbol{O(D^{(\omega+1)/2})}$
  (ω is the matrix multiplication exponent)

> **Brent, Kung. Fast algorithms for manipulating formal power series (1978).**

> **Shoup. Fast construction of irreducible polynomials over finite fields (1994).**

Power series manipulations: quasi-linear time $\boldsymbol{\tilde{O}(D)}$

- rational reconstruction
- modular inverse

2. Blocking

Blocking

Replace the scalar sequence $\boldsymbol{u}^T \boldsymbol{M}_a^i \boldsymbol{v}$ by the sequence of $\boldsymbol{m} \times \boldsymbol{m}$ matrices

$$\boldsymbol{U}^T \boldsymbol{M}_a^i \boldsymbol{V}, \quad \boldsymbol{U}, \boldsymbol{V} \in \mathbb{K}^{D \times m}.$$

What changes?

- should need fewer terms in the sequence (about $2D/m$)
- but computing each term is more expensive
- and we need a replacement for Berlekamp-Massey.

Coppersmith. Solving homogeneous linear equations over $\mathrm{GF}(2)$ via block Wiedemann algorithm (1994).

Matrix generating series

Now, we are looking for a matrix fraction decomposition

$$\sum_{i \geq 0} \frac{\boldsymbol{U}^T \boldsymbol{M}_a^i \boldsymbol{V}}{X^{i+1}} = \boldsymbol{T}^{-1}(X)\boldsymbol{N}(X),$$

with $\boldsymbol{N}$ and $\boldsymbol{T}$ in $\mathbb{K}[X]^{m \times m}$ ($\boldsymbol{T}$ satisfies a minimality property)

Proposition.

For generic choices of $\boldsymbol{U}$ and $\boldsymbol{V}$:

- $\boldsymbol{N}$ and $\boldsymbol{T}$ have degree at most D/m
- $2D/m$ terms in the sequence are enough to recover them
- the m largest invariant factors of $\boldsymbol{T}$ and $X\boldsymbol{I} - \boldsymbol{M}_a$ are the same.

Matrix generating series

Kailath. Linear systems (1980).

Kaltofen. Analysis of Coppersmith's block Wiedemann algorithm for the parallel solution of sparse linear systems (1994).

Villard. A study of Coppersmith's block Wiedemann algorithm using matrix polynomials (1997).

Kaltofen, Villard. On the complexity of computing determinants (2005).

Complexity issues

Matrix sequence: still $O(D)$ matrix vector products

- M_a sparse $O(D^2)$ but easy to parallelize
- $n = 1$ next part of the talk

Dense matrix operations $\tilde{O}(m^{\omega-1}D)$

- reconstruct N, T
- find the determinant of T, solving a linear system

Giorgi, Jeannerod, Villard. On the complexity of polynomial matrix computations (2003).

Storjohann. High-order lifting and integrality certification (2003).

Finding the minimal / characteristic polynomial

Suppose, as before:

- $\mathbb{A} = \mathbb{K}[X_1, \ldots, X_n]/I$, with $V(I) = \{\boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_D\}$
- $\boldsymbol{M}_a$ is the multiplication matrix by $a \in A$
- the $a(\boldsymbol{\alpha}_i)$'s are all distinct

For generic $\boldsymbol{U}, \boldsymbol{V}$, $P = \det(\boldsymbol{T}(X))$ is the minimal / characteristic polynomial of a.

Steel. Direct solution of the (11,9,8)-MinRank problem by the block Wiedemann algorithm in Magma with a Tesla GPU (2015).

Using the numerators

Recall: we also want numerators for $\ell(a^i)$ and $\ell(a^i X_1), \ldots, \ell(a^i X_n)$.

Observation.

$$\frac{N}{P} = \sum_{i \geq 0} \frac{\boldsymbol{u}^{\boldsymbol{T}} \boldsymbol{M}_a^i \boldsymbol{w}}{X^{i+1}}$$

Using the numerators

Recall: we also want numerators for $\ell(a^i)$ and $\ell(a^i X_1), \ldots, \ell(a^i X_n)$.

Observation.

$$N = P \sum_{i \geq 0} \frac{\boldsymbol{u}^{\boldsymbol{T}} \boldsymbol{M}_a^i \boldsymbol{w}}{X^{i+1}}$$

Using the numerators

Recall: we also want numerators for $\ell(a^i)$ and $\ell(a^i X_1), \ldots, \ell(a^i X_n)$.

Observation.

$$N = P \sum_{i \geq 0} \frac{\boldsymbol{u}^{\boldsymbol{T}} \boldsymbol{M}_a^i \boldsymbol{w}}{X^{i+1}} = [P\ 0 \cdots 0] \sum_{i \geq 0} \frac{\boldsymbol{U}^T \boldsymbol{M}_a^i \boldsymbol{w}}{X^{i+1}}$$

Using the numerators

Recall: we also want numerators for $\ell(a^i)$ and $\ell(a^i X_1), \ldots, \ell(a^i X_n)$.

Observation.

$$N = P \sum_{i \geq 0} \frac{\boldsymbol{u}^{\boldsymbol{T}} \, \boldsymbol{M}_a^i \, \boldsymbol{w}}{X^{i+1}} = [P \; 0 \cdots 0] \sum_{i \geq 0} \frac{\boldsymbol{U}^T \, \boldsymbol{M}_a^i \, \boldsymbol{w}}{X^{i+1}}$$

$$= \underbrace{\left([P \; 0 \cdots 0] \boldsymbol{T}(X)^{-1}\right)}_{\text{degree at most } D} \underbrace{\left(\boldsymbol{T}(X) \sum_{i \geq 0} \frac{\boldsymbol{U}^T \, \boldsymbol{M}_a^i \, \boldsymbol{w}}{X^{i+1}} \right)}_{\text{degree at most } D/m}$$

Hyun, Neiger, S, Rahkooy. Block-Krylov techniques in the context of sparse-FGLM algorithms (2017).

3. Structured projections for small n

A special case

Take $\boldsymbol{I} = \langle \boldsymbol{f(X)} \rangle$ in $\mathbb{K}[X]$, and a of degree less than $D = \deg(f)$.

Difficult to compute $\boldsymbol{U}^T \boldsymbol{M}_a^i \boldsymbol{V},\ i = 0, \ldots, 2D/m$ fast in general, so we set

$$\boldsymbol{Z} = \begin{bmatrix} 1 & & & \\ & 1 & & \\ & & \ddots & \\ & & & 1 \\ 0 & & & 0 \\ \vdots & & & \vdots \\ 0 & & & 0 \end{bmatrix}$$

and take

$$\boldsymbol{U} = \boldsymbol{V} = \boldsymbol{Z}.$$

Structured projections

Kaltofen. On computing determinants of matrices without divisions (1992).

Shoup. Fast construction of irreducible polynomials over finite fields(1994).

Kaltofen, Villard. On the complexity of computing determinants (2005).

Eberly, Giesbrecht, Giorgi, Storjohann, Villard. Solving sparse rational linear systems (2006), Faster inversion and other black box matrix computations using efficient block projections (2007).

Villard. On computing the resultant of generic bivariate polynomials (2018).

A faster projection

Proposition.

We can compute $Z^T M_a^i Z$, $i < 2D/m$, in time $O(mD + m(D/m)^{(\omega+1)/2})$.

Proof: a baby steps / giant steps algorithm for structured matrices.

Remark: these are $2mD$ numbers, naive algorithm $O(D^2)$

Kaltofen. On computing determinants of matrices without divisions (1992).

Kaltofen, Villard. On the complexity of computing determinants (2005).

Neiger, Salvy, S, Villard. Faster modular composition (2023).

A faster projection

Proposition.

We can compute $\boldsymbol{Z}^T \boldsymbol{M}_a^i \boldsymbol{Z}$, $i < 2D/m$, in time $\boldsymbol{O(mD + m(D/m)^{(\omega+1)/2})}$.

Corollary.

For $\boldsymbol{m = D^{1/3}}$ and for **generic a**, we can compute

- matrix numerator $\boldsymbol{N}(X)$, denominator $\boldsymbol{T}(X)$
- $\det(\boldsymbol{T}) =$ minimal polynomial of a mod f.

in time $\boldsymbol{O(D^{(\omega+2)/3})}$

- Shoup: $\boldsymbol{O(D^{(\omega+1)/2})}$ $\qquad \omega \le 2.37 \implies 1.69$
- Villard: $\boldsymbol{O(D^{2-1/\omega})}$ $\qquad \omega \le 2.37 \implies 1.58$
- our algorithm: $\boldsymbol{O(D^{(\omega+2)/3})}$ $\qquad \omega \le 2.37 \implies 1.46$

Modular composition

Definition.

Given h, a, f of degrees D, compute $h(a) \bmod f$.

Brent, Kung. Fast algorithms for manipulating formal power series (1978) $O(D^{(\omega+1)/2})$

Kedlaya, Umans. Fast polynomial factorization and modular composition (2011)
$(D \log(|\mathbb{K}|))^{1+o(1)}$ bit operations, $\mathbb{K}$ finite

Modular composition

Proposition.

Fix f and h with $\deg(h) < D$.
For **generic a**, we can compute $\boldsymbol{h(a) \bmod f}$ in time $\boldsymbol{O(D^{(\omega+2)/3})}$.

Proof: Reduce $[h\ 0 \cdots\ 0]^T$ by denominator $\boldsymbol{T}$ and do a bivariate modular composition.

Nüsken, Ziegler. Fast multipoint evaluation of bivariate polynomials (2004).

Theorem.

Las Vegas algorithm with same runtime ($\mathbb{K}$ large enough)

4. Bonus: more examples

Bivariate resultant

Similar approach: for $\boldsymbol{S}(X)$ Sylvester matrix of $F(X,Y), G(X,Y)$

- compute **structured projections** $\boldsymbol{Z}^T\boldsymbol{S}(X)^{-1}\boldsymbol{Z} \bmod X^k$
- reconstruct a matrix denominator
- compute its determinant

Bivariate resultant

Similar approach: for $\boldsymbol{S}(X)$ Sylvester matrix of $F(X,Y), G(X,Y)$

- compute **structured projections** $\boldsymbol{Z}^T \boldsymbol{S}(X)^{-1} \boldsymbol{Z} \bmod X^k$
- reconstruct a matrix denominator
- compute its determinant

Remark:

$$\sum_{i \geq 0} \frac{\boldsymbol{Z}^T \boldsymbol{M}^i \boldsymbol{Z}}{X^{i+1}} = \boldsymbol{Z}^T (X\boldsymbol{I} - \boldsymbol{M})^{-1} \boldsymbol{Z}$$

Bivariate resultant

Similar approach: for $\boldsymbol{S}(X)$ Sylvester matrix of $F(X,Y), G(X,Y)$

- compute **structured projections** $\boldsymbol{Z}^T \boldsymbol{S}(X)^{-1} \boldsymbol{Z} \bmod X^k$
- reconstruct a matrix denominator
- compute its determinant

For **generic** inputs of degree d_X, d_Y

- first subcubic algorithm $\boldsymbol{\tilde{O}(d_X d_Y^{2-1/\omega})}$ $\quad 2 - 1/\omega \simeq 1.58$

Villard. On computing the resultant of generic bivariate polynomials (2018).

- improved $\boldsymbol{\tilde{O}(d_X d_Y^{(\omega+2)/3})}$ if $d_X \leq d_Y^{1/3}$ $\quad (\omega+2)/3 \simeq 1.46$

Pernet, Signargout, Villard. High-order lifting for polynomial Sylvester matrices (2023).

Randomization still open

Speculation

Key ingredient in the latest algorithms: speeding up projections using

- baby steps / giant steps
- structured matrices algorithms

Other algorithms use block-Wiedemann techniques for "special" matrices $\boldsymbol{M}$...

- polynomial factorization (for $\boldsymbol{M}$ = matrix of the Frobenius)

> **Kaltofen, Lobo. Factoring high-degree polynomials by the black-box Berlekamp algorithm (1994).**

- characteristic polynomial in Drinfeld modules (for $c_0\boldsymbol{I} + c_1\boldsymbol{M} + c_2\boldsymbol{M}^2$)

> **Musleh, S. Computing the characteristic polynomial of a finite rank two Drinfeld module (2019).**