

Vincent Neiger  LIP6, Sorbonne Université, France
Bruno Salvy Inria, ENS Lyon, France
Éric Schost U. Waterloo, Canada
Gilles Villard CNRS, ENS Lyon, France

faster modular composition, using two relation matrices

Seminar Pascaline
LIP, Lyon, France – 1 October 2026

[J.ACM'24: [doi:10.1145/3638349](https://doi.org/10.1145/3638349)] [ISSAC'26: [doi:10.1145/3815436.3815451](https://doi.org/10.1145/3815436.3815451)]

outline

- ▶ context and result
- ▶ motivations and approach
- ▶ overview of the algorithms
- ▶ a soft epilogue

outline

▶ context and result

- ▶ univariate polynomial computations
- ▶ modular composition and minpoly
- ▶ improved algebraic complexity bound

▶ motivations and approach

▶ overview of the algorithms

▶ a soft epilogue

modular composition

composition:

(coefficients in a field)

$$[in] \quad a(x) = a_0 + a_1x + a_2x^2 + \cdots + a_{n-1}x^{n-1}$$

$$[in] \quad p(y) = p_0 + p_1y + p_2y^2 + \cdots + p_{n-1}y^{n-1}$$

$$[out] \quad p(a(x)) = p_0 + p_1a(x) + p_2a(x)^2 + \cdots + p_{n-1}a(x)^{n-1}$$

size $n^2 \rightsquigarrow$ optimistic target complexity = $O(n^2)$

modular composition

composition:

(coefficients in a field)

$$[in] \quad a(x) = a_0 + a_1x + a_2x^2 + \cdots + a_{n-1}x^{n-1}$$

$$[in] \quad p(y) = p_0 + p_1y + p_2y^2 + \cdots + p_{n-1}y^{n-1}$$

$$[out] \quad p(a(x)) = p_0 + p_1a(x) + p_2a(x)^2 + \cdots + p_{n-1}a(x)^{n-1}$$

size $n^2 \rightsquigarrow$ optimistic target complexity = $O(n^2)$

modular composition:

$$[in] \quad f(x) = f_0 + f_1x + f_2x^2 + \cdots + f_{n-1}x^{n-1} + x^n$$

$$[out] \quad p(a) \bmod f = p_0\alpha_0 + p_1\alpha_1 + p_2\alpha_2 + \cdots + p_{n-1}\alpha_{n-1}$$

where $\alpha_k = a^k \bmod f \in \mathbb{K}[x]_{<n}$

size $n \rightsquigarrow$ optimistic target complexity = $O(n)$

modular composition

composition:

(coefficients in a field)

$$[in] \quad a(x) = a_0 + a_1x + a_2x^2 + \cdots + a_{n-1}x^{n-1}$$

$$[in] \quad p(y) = p_0 + p_1y + p_2y^2 + \cdots + p_{n-1}y^{n-1}$$

$$[out] \quad p(a(x)) = p_0 + p_1a(x) + p_2a(x)^2 + \cdots + p_{n-1}a(x)^{n-1}$$

size $n^2 \rightsquigarrow$ optimistic target complexity = $O(n^2)$

modular composition:

$$[in] \quad f(x) = f_0 + f_1x + f_2x^2 + \cdots + f_{n-1}x^{n-1} + x^n$$

$$[out] \quad p(a) \bmod f = p_0\alpha_0 + p_1\alpha_1 + p_2\alpha_2 + \cdots + p_{n-1}\alpha_{n-1}$$

where $\alpha_k = a^k \bmod f \in \mathbb{K}[x]_{<n}$

size $n \rightsquigarrow$ optimistic target complexity = $O(n)$

a fundamental problem:

[Kaltofen-Shoup '98+'99] [Abelard-Couvreur-Lecerf '20+'22]

[Shoup '99] [Giesbrecht-Jamshidpey-Schost '21+'26]

- ▶ polynomial factorization
- ▶ Riemann-Roch space computations
- ▶ minimal polynomial modulo f
- ▶ computing normal bases

“fast”: measuring efficiency

efficient algorithms for polynomials, matrices, power series, ...
with coefficients in some base field $\mathbb{K}$

- ▶ low **complexity bound**
- ▶ low **execution time**

..., low memory usage, low power consumption, ...

- ▶ prime field $\mathbb{K} = \mathbb{Z}/p\mathbb{Z}$
- ▶ rational numbers $\mathbb{K} = \mathbb{Q}$



- ▶ finite extensions $\mathbb{K}[x]/\langle f(x) \rangle$
- ▶ rational fractions $\mathbb{K}(x, y, z)$

“fast”: measuring efficiency

efficient algorithms for polynomials, matrices, power series, ...
with coefficients in some base field $\mathbb{K}$

- ▶ low complexity bound
- ▶ low execution time

..., low memory usage, low power consumption, ...

- ▶ prime field $\mathbb{K} = \mathbb{Z}/p\mathbb{Z}$
- ▶ rational numbers $\mathbb{K} = \mathbb{Q}$



- ▶ finite extensions $\mathbb{K}[x]/\langle f(x) \rangle$
- ▶ rational fractions $\mathbb{K}(x, y, z)$

algebraic complexity bounds

↪ count basic operations in $\mathbb{K}$

- ▶ operations $+, -, \times, \div, =$
- ▶ notation $O(\cdot)$ and $\tilde{O}(\cdot)$
- ▶ this talk: no parallelism, single thread

related?

practical performance

↪ measure software runtime

for the few runtimes in this talk:

- ▶ my laptop with FLINT/NTL/PML
- ▶ prime field $\mathbb{K}$ of size $\approx 2^{60} \approx 10^{18}$

fundamental computations on polynomials

most problems: quasi-linear via reductions to multiplication

$$M(n) \in O(n \log(n) \log \log(n)) \subset \tilde{O}(n)$$



fundamental computations on polynomials

most problems: quasi-linear via reductions to multiplication

$$M(n) \in O(n \log(n) \log \log(n)) \subset \tilde{O}(n)$$

$$O(M(n))$$

- addition $a + b$, multiplication $a * b$
- **division** with remainder $a = qb + r$
- power series **inverse** $a^{-1} \bmod x^n$

$$O(M(n) \log(n))$$

- **eval./interp.** $a \leftrightarrow a(x_1), \dots, a(x_n)$
- extended **GCD** $au + bv = \gcd(a, b)$
- rational **reconstruction** $a = \frac{p}{q} \bmod M$



fundamental computations on polynomials

most problems: quasi-linear via reductions to multiplication

$$M(n) \in O(n \log(n) \log \log(n)) \subset \tilde{O}(n)$$

$O(M(n))$

- ▶ addition $a + b$, multiplication $a * b$
- ▶ **division** with remainder $a = qb + r$
- ▶ power series **inverse** $a^{-1} \bmod x^n$

$O(M(n) \log(n))$

- ▶ **eval./interp.** $a \leftrightarrow a(x_1), \dots, a(x_n)$
- ▶ extended **GCD** $au + bv = \gcd(a, b)$
- ▶ rational **reconstruction** $a = \frac{p}{q} \bmod M$

recent breakthrough: in $O(M(n) \log(n))$,

- ▶ **series composition** $p(a) \bmod x^n$
- ▶ **series power projections** $\ell(a^k \bmod x^n)$

[Kinoshita-Li 2024]
via [Bostan-Mori 2021]



fundamental computations on polynomials

most problems: quasi-linear via reductions to multiplication

$$M(n) \in O(n \log(n) \log \log(n)) \subset \tilde{O}(n)$$

$O(M(n))$

- ▶ addition $a + b$, multiplication $a * b$
- ▶ **division** with remainder $a = qb + r$
- ▶ power series **inverse** $a^{-1} \bmod x^n$

$O(M(n) \log(n))$

- ▶ **eval./interp.** $a \leftrightarrow a(x_1), \dots, a(x_n)$
- ▶ extended **GCD** $au + bv = \gcd(a, b)$
- ▶ rational **reconstruction** $a = \frac{p}{q} \bmod M$

recent breakthrough: in $O(M(n) \log(n))$,

- ▶ **series composition** $p(a) \bmod x^n$
- ▶ **series power projections** $\ell(a^k \bmod x^n)$

[Kinoshita-Li 2024]
via [Bostan-Mori 2021]

most problems, except. . .

Open problem 2.4

[book: Algebraic Complexity Theory]

Can Brent&Kung's algorithm for the **general composition problem** be improved substantially?

Modern Computer Algebra

Research problem 12.19

[book: Modern Computer Algebra]

Can you improve the cost for **modular composition of polynomials** to, say, $\tilde{O}(n^{1.5})$ or better?

target problems and complexity result

[in] polynomials $a, f \in \mathbb{K}[x]$ with $n = \deg(f) > \deg(a)$

modular composition

[in] polynomial $p \in \mathbb{K}[y]$
[out] $p(a) \bmod f$

$\xleftrightarrow{T}$

power projections

[in] linear form $\ell : \mathbb{K}[x]/\langle f \rangle \rightarrow \mathbb{K}$
[out] $[\ell(a^k \bmod f)]_{k=0, \dots, n-1}$

minimal polynomial

[out] small-degree $p \in \mathbb{K}[y]$ s.t. $p(a) = 0 \bmod f$

target problems and complexity result

[in] polynomials $a, f \in \mathbb{K}[x]$ with $n = \deg(f) > \deg(a)$

modular composition

[in] polynomial $p \in \mathbb{K}[y]$
[out] $p(a) \bmod f$

power projections

[in] linear form $\ell : \mathbb{K}[x]/\langle f \rangle \rightarrow \mathbb{K}$
[out] $[\ell(a^k \bmod f)]_{k=0, \dots, n-1}$

$\xleftrightarrow{T}$

minimal polynomial

[out] small-degree $p \in \mathbb{K}[y]$ s.t. $p(a) = 0 \bmod f$

exponent	$\omega = 3$	$\omega = 2.81..$	$\omega = 2.37..$	$\omega = 2.0..$
$(\omega + 1)/2$	2	1.9	1.69	1.5

previous work (deterministic)

- naive: $O(nM(n)) \subseteq \tilde{O}(n^2)$
- baby/giant steps: $O(n^{(\omega+1)/2})$

[Brent-Kung 1978] [Shoup 1994]

target problems and complexity result

[in] polynomials $a, f \in \mathbb{K}[x]$ with $n = \deg(f) > \deg(a)$

modular composition

[in] polynomial $p \in \mathbb{K}[y]$
[out] $p(a) \bmod f$

$\xleftrightarrow{T}$

power projections

[in] linear form $\ell : \mathbb{K}[x]/\langle f \rangle \rightarrow \mathbb{K}$
[out] $[\ell(a^k \bmod f)]_{k=0, \dots, n-1}$

minimal polynomial

[out] small-degree $p \in \mathbb{K}[y]$ s.t. $p(a) = 0 \bmod f$

exponent	$\omega = 3$	$\omega = 2.81..$	$\omega = 2.37..$	$\omega = 2.0..$
$(\omega + 1)/2$	2	1.9	1.69	1.5
$(\omega + 2)/3$	1.67	1.6	1.46	1.33

previous work (deterministic)

- naive: $O(nM(n)) \subseteq \tilde{O}(n^2)$
- baby/giant steps: $O(n^{(\omega+1)/2})$

[Brent-Kung 1978] [Shoup 1994]

previous work

- bit complexity: $O((n \log q)^{1+\varepsilon})$
- algebraic, Las Vegas: $\tilde{O}(n^{(\omega+2)/3})$

[Kedlaya-Umans '11] [N.-S.-S.-V.'24]

target problems and complexity result

[in] polynomials $a, f \in \mathbb{K}[x]$ with $n = \deg(f) > \deg(a)$

modular composition

[in] polynomial $p \in \mathbb{K}[y]$
 [out] $p(a) \bmod f$

power projections

[in] linear form $\ell : \mathbb{K}[x]/\langle f \rangle \rightarrow \mathbb{K}$
 [out] $[\ell(a^k \bmod f)]_{k=0, \dots, n-1}$

minimal polynomial

[out] small-degree $p \in \mathbb{K}[y]$ s.t. $p(a) = 0 \bmod f$

result: complexity $\tilde{O}(n^{(\omega+3)/4})$, for generic $a(x)$

exponent	$\omega = 3$	$\omega = 2.81..$	$\omega = 2.37..$	$\omega = 2.0..$
$(\omega + 1)/2$	2	1.9	1.69	1.5
$(\omega + 2)/3$	1.67	1.6	1.46	1.33
$(\omega + 3)/4$	1.5	1.45	1.34	1.25

previous work (deterministic)

- naive: $O(nM(n)) \subseteq \tilde{O}(n^2)$
- baby/giant steps: $O(n^{(\omega+1)/2})$

[Brent-Kung 1978] [Shoup 1994]

previous work

- bit complexity: $O((n \log q)^{1+\varepsilon})$
- algebraic, Las Vegas: $\tilde{O}(n^{(\omega+2)/3})$

[Kedlaya-Umans '11] [N.-S.-S.-V.'24]

outline

▶ context and result

- ▶ univariate polynomial computations
- ▶ modular composition and minpoly
- ▶ improved algebraic complexity bound

▶ motivations and approach

▶ overview of the algorithms

▶ a soft epilogue

outline

▶ context and result

- ▶ univariate polynomial computations
- ▶ modular composition and minpoly
- ▶ improved algebraic complexity bound

▶ motivations and approach

- ▶ complexity and software motivations
- ▶ baby steps – giant steps
- ▶ computing fewer powers $a^k \bmod f$

▶ overview of the algorithms

▶ a soft epilogue

- ▶ breakthrough: bit cost $O((n \log(q))^{1+\varepsilon})$ for any $\varepsilon > 0$, over $\mathbb{F}_q$
[Kedlaya-Umans '08+'11] [Bhargava-Ghosh-Kumar-Mohapatra '23]
- ▶ improved algorithm and analysis: $n^{5/(\log_2 n)^{1/2}} \tilde{O}(n \log r)$ over $\mathbb{Z}/r\mathbb{Z}$
[van der Hoeven-Lecerf, J.Complexity'20 + preprint'26]
more precisely, $O(n \cdot n^{5/(\log_2 n)^{1/2}} \cdot (\log n)^{13.5} \cdot \log r \cdot (\log \log r)^3)$

almost linear bit complexity

complexity and software motivations

- breakthrough: bit cost $O((n \log(q))^{1+\varepsilon})$ for any $\varepsilon > 0$, over $\mathbb{F}_q$
[Kedlaya-Umans '08+'11] [Bhargava-Ghosh-Kumar-Mohapatra '23]
 - improved algorithm and analysis: $n^{5/(\log_2 n)^{1/2}} \tilde{O}(n \log r)$ over $\mathbb{Z}/r\mathbb{Z}$
[van der Hoeven-Lecerf, J.Complexity'20 + preprint'26]
- more precisely, $O(n \cdot n^{5/(\log_2 n)^{1/2}} \cdot (\log n)^{13.5} \cdot \log r \cdot (\log \log r)^3)$

almost linear bit complexity

complexity and software motivations

breaking news!

Subject: A new algebraic algorithm for modular composition

Date: August 2026

Dear Vincent,

This is Baitian Li. It has been a long time since we last spoke about the implementation of the power series composition algorithm in the FLINT library; I hope you are doing well.

I am writing to share a new algorithm for modular composition. [...]

[...]

- ▶ breakthrough: bit cost $O((n \log(q))^{1+\varepsilon})$ for any $\varepsilon > 0$, over $\mathbb{F}_q$
[Kedlaya-Umans '08+'11] [Bhargava-Ghosh-Kumar-Mohapatra '23]
 - ▶ improved algorithm and analysis: $n^{5/(\log_2 n)^{1/2}} \tilde{O}(n \log r)$ over $\mathbb{Z}/r\mathbb{Z}$
[van der Hoeven-Lecerf, J.Complexity'20 + preprint'26]
- more precisely, $O(n \cdot n^{5/(\log_2 n)^{1/2}} \cdot (\log n)^{13.5} \cdot \log r \cdot (\log \log r)^3)$

almost linear bit complexity

complexity and software motivations

breaking news!

Subject: A new algebraic algorithm for modular composition

Date: August 2026

I am writing to share a new algorithm for modular composition. [...]

[...]

Following this idea seems to lead to an almost-linear-time algebraic algorithm for modular composition.

- ▶ breakthrough: bit cost $O((n \log(q))^{1+\varepsilon})$ for any $\varepsilon > 0$, over $\mathbb{F}_q$
[Kedlaya-Umans '08+'11] [Bhargava-Ghosh-Kumar-Mohapatra '23]
 - ▶ improved algorithm and analysis: $n^{5/(\log_2 n)^{1/2}} \tilde{O}(n \log r)$ over $\mathbb{Z}/r\mathbb{Z}$
[van der Hoeven-Lecerf, J.Complexity'20 + preprint'26]
- more precisely, $O(n \cdot n^{5/(\log_2 n)^{1/2}} \cdot (\log n)^{13.5} \cdot \log r \cdot (\log \log r)^3)$

almost linear bit complexity

complexity and software motivations

breaking news!

Subject: A new algebraic algorithm for modular composition

Date: August 2026

I am writing to share a new algorithm for modular composition. [...]

[...]

Following this idea seems to lead to an almost-linear-time algebraic algorithm for modular composition.

almost-linear-time algebraic algorithm

talk ends here! time for lunch!

- breakthrough: bit cost $O((n \log(q))^{1+\varepsilon})$ for any $\varepsilon > 0$, over $\mathbb{F}_q$
[Kedlaya-Umans '08+'11] [Bhargava-Ghosh-Kumar-Mohapatra '23]
 - improved algorithm and analysis: $n^{5/(\log_2 n)^{1/2}} \tilde{O}(n \log r)$ over $\mathbb{Z}/r\mathbb{Z}$
[van der Hoeven-Lecerf, J.Complexity'20 + preprint'26]
- more precisely, $O(n \cdot n^{5/(\log_2 n)^{1/2}} \cdot (\log n)^{13.5} \cdot \log r \cdot (\log \log r)^3)$

almost linear bit complexity

complexity and software motivations

practical considerations

n	$n^{5/(\log_2 n)^{1/2}}$	$\longleftarrow = 32\sqrt{\log_2 n}$
10^4	$3.1 \cdot 10^5$	
10^5	$1.4 \cdot 10^6$	
10^6	$5.2 \cdot 10^6$	
10^7	$1.8 \cdot 10^7$	
10^8	$5.7 \cdot 10^7$	
10^9	$1.7 \cdot 10^8$	
10^{10}	$4.7 \cdot 10^8$	
10^{11}	$1.3 \cdot 10^9$	
10^{12}	$3.2 \cdot 10^9$	
10^{13}	$7.8 \cdot 10^9$	

$$n^{5/(\log_2 n)^{1/2}} \cdot (\log n)^{13.5} > n \text{ until } n \approx 10^{49}$$

*"it remains a **major challenge** to make [Kedlaya and Umans'] algorithm **efficient for practical input sizes**"*

[van der Hoeven-Lecerf '20]

- breakthrough: bit cost $O((n \log(q))^{1+\varepsilon})$ for any $\varepsilon > 0$, over $\mathbb{F}_q$
[Kedlaya-Umans '08+'11] [Bhargava-Ghosh-Kumar-Mohapatra '23]
 - improved algorithm and analysis: $n^{5/(\log_2 n)^{1/2}} \tilde{O}(n \log r)$ over $\mathbb{Z}/r\mathbb{Z}$
[van der Hoeven-Lecerf, J.Complexity'20 + preprint'26]
- more precisely, $O(n \cdot n^{5/(\log_2 n)^{1/2}} \cdot (\log n)^{13.5} \cdot \log r \cdot (\log \log r)^3)$

almost linear bit complexity

complexity and software motivations

quasi-linear algebraic algorithm for series

- algorithm is valid over any field $\mathbb{K}$ (in fact, commutative ring)
- complexity bound is quasi-linear with small extra factors

n	[Brent-Kung 1978]		[Kinoshita-Li 2024]		speedup	
$\mathbb{K}$	$\mathbb{Z}/r\mathbb{Z}$	$\mathbb{Z}$	$\mathbb{Z}/r\mathbb{Z}$	$\mathbb{Z}$	$\mathbb{Z}/r\mathbb{Z}$	$\mathbb{Z}$
200	0.000285	0.00243	0.000347	0.00284	0.821	0.856
400	0.000869	0.0153	0.000839	0.0144	1.036	1.062
800	0.00254	0.108	0.00193	0.0683	1.316	1.581
1600	0.00763	0.682	0.00434	0.339	1.758	2.012
3200	0.0238	4.651	0.00974	1.716	2.444	2.710
6400	0.0776	33.263	0.0218	8.462	3.560	3.931
12800	0.257	266.59	0.0487	43.331	5.277	6.152
204800	37.76		1.556		24.267	

baby steps – giant steps

[in] $f(x)$ of degree n , $a(x)$ of degree $< n$, $p(y)$ of degree $< n$
[out] $p(a(x)) \bmod f(x) = p_0\alpha_0(x) + p_1\alpha_1(x) + \cdots + p_{n-1}\alpha_{n-1}(x)$
where $\alpha_k = a^k \bmod f \in \mathbb{K}[x]_{<n}$

naive cost: $\tilde{O}(n^2)$ ▶ dominant step: compute n powers of $a \bmod f$
▶ how to require fewer powers?

$$\sum_{k < n} p_k \alpha_k = \begin{bmatrix} p_0 & p_1 & \cdots & p_{n-1} \end{bmatrix} \begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \vdots \\ \alpha_{n-1} \end{bmatrix}$$

baby steps – giant steps

[in] $f(x)$ of degree n , $a(x)$ of degree $< n$, $p(y)$ of degree $< n$
[out] $p(a(x)) \bmod f(x) = p_0\alpha_0(x) + p_1\alpha_1(x) + \cdots + p_{n-1}\alpha_{n-1}(x)$
where $\alpha_k = a^k \bmod f \in \mathbb{K}[x]_{<n}$

naive cost: $\tilde{O}(n^2)$ ▶ dominant step: compute n powers of $a \bmod f$
▶ how to require fewer powers?

$$\sum_{k < n} p_k \alpha_k = \begin{bmatrix} p_0 & p_1 & \cdots & p_{n-1} \end{bmatrix} \begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \vdots \\ \alpha_{n-1} \end{bmatrix}$$

split exponents as $k = ir + j$ with $r = \sqrt{n}$ and $i, j < r$
 $\rightsquigarrow \alpha_k = a^{ir} \cdot a^j \bmod f = \alpha_{ir} \alpha_j \bmod f$

$$\sum_{i,j < r} p_{ir+j} \alpha_{ir} \alpha_j = \begin{bmatrix} \alpha_0 & \alpha_r & \cdots & \alpha_{(r-1)r} \end{bmatrix} \begin{bmatrix} p_0 & p_1 & \cdots & p_{r-1} \\ p_r & p_{r+1} & \cdots & p_{2r-1} \\ \vdots & \vdots & \ddots & \vdots \\ p_{n-r} & p_{n-r+1} & \cdots & p_{n-1} \end{bmatrix} \begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \vdots \\ \alpha_{r-1} \end{bmatrix}$$

baby steps – giant steps

[in] $f(x)$ of degree n , $a(x)$ of degree $< n$, $p(y)$ of degree $< n$
[out] $p(a) \bmod f = \sum_{k < n} p_k \alpha_k$ where $\alpha_k = a^k \bmod f \in \mathbb{K}[x]_{< n}$

$2r$ powers of $a \bmod f$ by **splitting** exponents
 $k = ir + j$ with $r = \sqrt{n}$ and $i, j < r$

[Paterson-Stockmeyer 1971]
[Brent-Kung 1978]

$$p(a) \bmod f = \begin{bmatrix} \alpha_0 & \alpha_r & \cdots & \alpha_{(r-1)r} \end{bmatrix} \begin{bmatrix} p_0 & p_1 & \cdots & p_{r-1} \\ p_r & p_{r+1} & \cdots & p_{2r-1} \\ \vdots & \vdots & \ddots & \vdots \\ p_{n-r} & p_{n-r+1} & \cdots & p_{n-1} \end{bmatrix} \begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \vdots \\ \alpha_{r-1} \end{bmatrix}$$



baby steps – giant steps

[in] $f(x)$ of degree n , $a(x)$ of degree $< n$, $p(y)$ of degree $< n$
[out] $p(a) \bmod f = \sum_{k < n} p_k \alpha_k$ where $\alpha_k = a^k \bmod f \in \mathbb{K}[x]_{<n}$

$2r$ powers of $a \bmod f$ by **splitting** exponents
 $k = ir + j$ with $r = \sqrt{n}$ and $i, j < r$

[Paterson-Stockmeyer 1971]
[Brent-Kung 1978]

$$p(a) \bmod f = \begin{bmatrix} \alpha_0 & \alpha_r & \cdots & \alpha_{(r-1)r} \end{bmatrix} \begin{bmatrix} p_0 & p_1 & \cdots & p_{r-1} \\ p_r & p_{r+1} & \cdots & p_{2r-1} \\ \vdots & \vdots & \ddots & \vdots \\ p_{n-r} & p_{n-r+1} & \cdots & p_{n-1} \end{bmatrix} \begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \vdots \\ \alpha_{r-1} \end{bmatrix}$$

vector in $\mathbb{K}[x]_{<n}^{1 \times \sqrt{n}}$  square matrix in $\mathbb{K}^{\sqrt{n} \times \sqrt{n}}$ vector in $\mathbb{K}[x]_{<n}^{\sqrt{n} \times 1}$



cost bound: $\tilde{O}(n^{3/2})$ for $2\sqrt{n}$ powers of $a \bmod f$
 $+ O(n^{(\omega+1)/2})$ for $\mathbb{K}$ -matrix multiplication $\sqrt{n} \times \sqrt{n} \times n$

runtimes: $\blacktriangleright$ much faster than naive approach
 $\blacktriangleright \tilde{O}(n^{3/2})$ regime lasts until largish n

baby steps – giant steps

[in] $f(x)$ of degree n , $a(x)$ of degree $< n$, $p(y)$ of degree $< n$
[out] $p(a) \bmod f = \sum_{k < n} p_k \alpha_k$ where $\alpha_k = a^k \bmod f \in \mathbb{K}[x]_{<n}$

$2r$ powers of $a \bmod f$ by **splitting** exponents
 $k = ir + j$ with $r = \sqrt{n}$ and $i, j < r$

[Paterson-Stockmeyer 1971]
[Brent-Kung 1978]

$$p(a) \bmod f = \begin{bmatrix} \alpha_0 & \alpha_r & \cdots & \alpha_{(r-1)r} \end{bmatrix} \begin{bmatrix} p_0 & p_1 & \cdots & p_{r-1} \\ p_r & p_{r+1} & \cdots & p_{2r-1} \\ \vdots & \vdots & \ddots & \vdots \\ p_{n-r} & p_{n-r+1} & \cdots & p_{n-1} \end{bmatrix} \begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \vdots \\ \alpha_{r-1} \end{bmatrix}$$

vector in $\mathbb{K}[x]_{<n}^{1 \times \sqrt{n}}$ square matrix in $\mathbb{K}^{\sqrt{n} \times \sqrt{n}}$ vector in $\mathbb{K}[x]_{<n}^{\sqrt{n} \times 1}$

cost bound: $\tilde{O}(n^{3/2})$ for $2\sqrt{n}$ powers of $a \bmod f$
 $+ O(n^{(\omega+1)/2})$ for $\mathbb{K}$ -matrix multiplication $\sqrt{n} \times \sqrt{n} \times n$

runtimes: $\blacktriangleright$ much faster than naive approach
 $\blacktriangleright \tilde{O}(n^{3/2})$ regime lasts until largish n

how to require even fewer powers?

fewer powers, episode 1: bivariate input $p(y) \rightarrow P(x, y)$

[in] $f(x)$ and $a(x)$ as before,

[in] $P(x, y) = \sum_{k < s^2} P_k(x) y^k$ of degrees $< (s, s^2)$ for $s = n^{1/3}$

[out] $P(x, a) \bmod f = \sum_{k < s^2} P_k(x) \alpha_k$ where $\alpha_k = a^k \bmod f$

2s powers of $a \bmod f$ by splitting exponents

$k = is + j$ with $s = n^{1/3}$ and $i, j < s$

[Paterson-Stockmeyer 1971]

[Brent-Kung 1978]

[Nüsken-Ziegler 2004]

$$P(x, a) \bmod f = \begin{bmatrix} \alpha_0 & \alpha_s & \cdots & \alpha_{(s-1)s} \end{bmatrix} \begin{bmatrix} P_0 & P_1 & \cdots & P_{s-1} \\ P_s & P_{s+1} & \cdots & P_{2s-1} \\ \vdots & \vdots & \ddots & \vdots \\ P_{s^2-s} & P_{s^2-s+1} & \cdots & P_{s^2-1} \end{bmatrix} \begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \vdots \\ \alpha_{s-1} \end{bmatrix}$$



fewer powers, episode 1: bivariate input $p(y) \rightarrow P(x, y)$

[in] $f(x)$ and $a(x)$ as before,

[in] $P(x, y) = \sum_{k < s^2} P_k(x) y^k$ of degrees $< (s, s^2)$ for $s = n^{1/3}$

[out] $P(x, a) \bmod f = \sum_{k < s^2} P_k(x) \alpha_k$ where $\alpha_k = a^k \bmod f$

2s powers of $a \bmod f$ by splitting exponents

$k = is + j$ with $s = n^{1/3}$ and $i, j < s$

[Paterson-Stockmeyer 1971]

[Brent-Kung 1978]

[Nüsken-Ziegler 2004]

$$P(x, a) \bmod f = \begin{bmatrix} \alpha_0 & \alpha_s & \cdots & \alpha_{(s-1)s} \end{bmatrix} \begin{bmatrix} P_0 & P_1 & \cdots & P_{s-1} \\ P_s & P_{s+1} & \cdots & P_{2s-1} \\ \vdots & \vdots & \ddots & \vdots \\ P_{s^2-s} & P_{s^2-s+1} & \cdots & P_{s^2-1} \end{bmatrix} \begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \vdots \\ \alpha_{s-1} \end{bmatrix}$$

vector in $\mathbb{K}[x]_{<n}^{1 \times s}$



square matrix in $\mathbb{K}[x]_{<s}^{s \times s}$

vector in $\mathbb{K}[x]_{<n}^{s \times 1}$



cost bound: $\tilde{O}(n^{4/3})$ for 2s powers of $a \bmod f$
 $+ \tilde{O}(n^{(\omega+2)/3})$ for matrix multiplication in $\mathbb{K}[x]_{<s^2}^{s \times s}$

fewer powers, episode 1: bivariate input $p(y) \rightarrow P(x, y)$

[in] $f(x)$ and $a(x)$ as before,

[in] $P(x, y) = \sum_{k < s^2} P_k(x) y^k$ of degrees $< (s, s^2)$ for $s = n^{1/3}$

[out] $P(x, a) \bmod f = \sum_{k < s^2} P_k(x) \alpha_k$ where $\alpha_k = a^k \bmod f$

2s powers of $a \bmod f$ by splitting exponents

$k = is + j$ with $s = n^{1/3}$ and $i, j < s$

[Paterson-Stockmeyer 1971]

[Brent-Kung 1978]

[Nüsken-Ziegler 2004]

$$P(x, a) \bmod f = \begin{bmatrix} \alpha_0 & \alpha_s & \cdots & \alpha_{(s-1)s} \end{bmatrix} \begin{bmatrix} P_0 & P_1 & \cdots & P_{s-1} \\ P_s & P_{s+1} & \cdots & P_{2s-1} \\ \vdots & \vdots & \ddots & \vdots \\ P_{s^2-s} & P_{s^2-s+1} & \cdots & P_{s^2-1} \end{bmatrix} \begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \vdots \\ \alpha_{s-1} \end{bmatrix}$$

vector in $\mathbb{K}[x]_{<n}^{1 \times s}$



square matrix in $\mathbb{K}[x]_{<s}^{s \times s}$

vector in $\mathbb{K}[x]_{<n}^{s \times 1}$



cost bound: $\tilde{O}(n^{4/3})$ for 2s powers of $a \bmod f$

+ $\tilde{O}(n^{(\omega+2)/3})$ for matrix multiplication in $\mathbb{K}[x]_{<s}^{s \times s}$

could we transform $p(y) \rightarrow P(x, y)$?

s.t. $p(a) = P(x, a) \bmod f$

how to require even fewer powers?

fewer powers, new episode: bivariate powers $a(x)^k \rightarrow A_k(x, y)$

[in] $f(x)$ and $a(x)$ as before, back to $p(y)$ univariate of degree $< n$

write $p(y) = \sum_{k < s^2} P_k(y) y^{ks}$ with $\deg(P_k) < s$, for $s = n^{1/3}$

[out] $p(a) \bmod f = \sum_{k < s^2} P_k(a) \alpha_{ks}$ where $\alpha_{ks} = a^{ks} \bmod f$

2s powers of $a \bmod f$ by splitting exponents

$ks = is^2 + js$ with $s = n^{1/3}$ and $i, j < s$

$$p(a) \bmod f = \begin{bmatrix} \alpha_0 & \alpha_{s^2} & \cdots & \alpha_{(s-1)s^2} \end{bmatrix} \begin{bmatrix} P_0 & P_1 & \cdots & P_{s-1} \\ P_s & P_{s+1} & \cdots & P_{2s-1} \\ \vdots & \vdots & \ddots & \vdots \\ P_{s^2-s} & P_{s^2-s+1} & \cdots & P_{s^2-1} \end{bmatrix} \begin{bmatrix} \alpha_0 \\ \alpha_s \\ \vdots \\ \alpha_{(s-1)s} \end{bmatrix}$$

vector in $\mathbb{K}[x]_{<n}^{1 \times s}$



square matrix in $\mathbb{K}[y]_{<s}^{s \times s}$

vector in $\mathbb{K}[x]_{<n}^{s \times 1}$



fewer powers, new episode: bivariate powers $a(x)^k \rightarrow A_k(x, y)$

[in] $f(x)$ and $a(x)$ as before, back to $p(y)$ univariate of degree $< n$

write $p(y) = \sum_{k < s^2} P_k(y) y^{ks}$ with $\deg(P_k) < s$, for $s = n^{1/3}$

[out] $p(a) \bmod f = \sum_{k < s^2} P_k(a) \alpha_{ks}$ where $\alpha_{ks} = a^{ks} \bmod f$

2s powers of $a \bmod f$ by splitting exponents

$ks = is^2 + js$ with $s = n^{1/3}$ and $i, j < s$

$A_{ks}(x, y) \in \mathbb{K}[x, y]_{<(s^2, s)}$

$A_{ks}(x, a) = a^{ks} \bmod f$

$$p(a) \bmod f = \begin{bmatrix} A_0 & A_{s^2} & \cdots & A_{(s-1)s^2} \end{bmatrix} \begin{bmatrix} P_0 & P_1 & \cdots & P_{s-1} \\ P_s & P_{s+1} & \cdots & P_{2s-1} \\ \vdots & \vdots & \ddots & \vdots \\ P_{s^2-s} & P_{s^2-s+1} & \cdots & P_{s^2-1} \end{bmatrix} \begin{bmatrix} A_0 \\ A_s \\ \vdots \\ A_{(s-1)s} \end{bmatrix}$$

vector in $\mathbb{K}[x, y]_{<(s^2, s)}^{1 \times s}$



square matrix in $\mathbb{K}[y]_{<s}^{s \times s}$

vector in $\mathbb{K}[x, y]_{<(s^2, s)}^{s \times 1}$



fewer powers, new episode: bivariate powers $a(x)^k \rightarrow A_k(x, y)$

[in] $f(x)$ and $a(x)$ as before, back to $p(y)$ univariate of degree $< n$

write $p(y) = \sum_{k < s^2} P_k(y) y^{ks}$ with $\deg(P_k) < s$, for $s = n^{1/3}$

[out] $p(a) \bmod f = \sum_{k < s^2} P_k(a) \alpha_{ks}$ where $\alpha_{ks} = a^{ks} \bmod f$

2s powers of $a \bmod f$ by splitting exponents

$ks = is^2 + js$ with $s = n^{1/3}$ and $i, j < s$

$$A_{ks}(x, y) \in \mathbb{K}[x, y]_{<(s^2, s)}$$

$$A_{ks}(x, a) = a^{ks} \bmod f$$

$$p(a) \bmod f = \begin{bmatrix} A_0 & A_{s^2} & \cdots & A_{(s-1)s^2} \end{bmatrix} \begin{bmatrix} P_0 & P_1 & \cdots & P_{s-1} \\ P_s & P_{s+1} & \cdots & P_{2s-1} \\ \vdots & \vdots & \ddots & \vdots \\ P_{s^2-s} & P_{s^2-s+1} & \cdots & P_{s^2-1} \end{bmatrix} \begin{bmatrix} A_0 \\ A_s \\ \vdots \\ A_{(s-1)s} \end{bmatrix}$$

vector in $\mathbb{K}[x, y]_{<(s^2, s)}^{1 \times s}$



square matrix in $\mathbb{K}[y]_{<s}^{s \times s}$

vector in $\mathbb{K}[x, y]_{<(s^2, s)}^{s \times 1}$



cost bound: $\tilde{O}(n^{4/3})$ for 2s powers of $a \bmod f$

+ $\tilde{O}(n^{(\omega+2)/3})$ for matrix multiplication in $\mathbb{K}[x, y]_{<(s, s)}^{s \times s}$

how to transform $a(x)^k \rightarrow A_k(x, y)$?

s.t. $a^k = A_k(x, a) \bmod f$

how to require even fewer powers?

outline

▶ context and result

- ▶ univariate polynomial computations
- ▶ modular composition and minpoly
- ▶ improved algebraic complexity bound

▶ motivations and approach

- ▶ complexity and software motivations
- ▶ baby steps – giant steps
- ▶ computing fewer powers $a^k \bmod f$

▶ overview of the algorithms

▶ a soft epilogue

outline

▶ context and result

- ▶ univariate polynomial computations
- ▶ modular composition and minpoly
- ▶ improved algebraic complexity bound

▶ motivations and approach

- ▶ complexity and software motivations
- ▶ baby steps – giant steps
- ▶ computing fewer powers $a^k \bmod f$

▶ overview of the algorithms

- ▶ reductions modulo a bivariate ideal
- ▶ exploiting small $\mathbb{K}[x]$ - and $\mathbb{K}[y]$ -relations
- ▶ ? genericity $\longrightarrow$ randomization ?

▶ a soft epilogue

balancing degrees: from univariate to bivariate

choose parameters $m \ll d$ with $md = \Theta(n)$

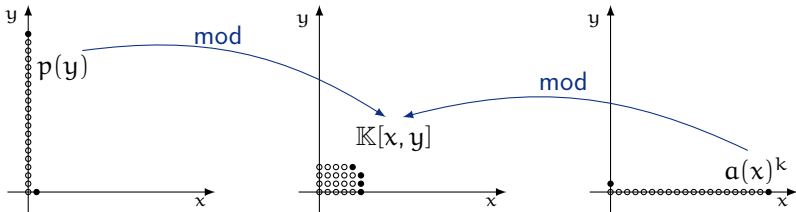
transform $p(y)$ into $P(x, y)$ such that $\begin{cases} \deg_{x,y}(P) < (m, d) \\ p(a) = P(x, a) \bmod f \end{cases}$

transform $\alpha_k(x)$ into $A_k(x, y)$ such that $\begin{cases} \deg_{x,y}(A_k) < (d, m) \\ a^k = A_k(x, a) \bmod f \end{cases}$

computations modulo $\mathcal{I} = \langle f(x), y - a(x) \rangle$

$$p(y) = P(x, y) \bmod \mathcal{I}$$

$$y^k = a(x)^k = A_k(x, y) \bmod \mathcal{I}$$



balancing degrees: from univariate to bivariate

choose parameters $m \ll d$ with $md = \Theta(n)$

transform $p(y)$ into $P(x, y)$ such that
$$\begin{cases} \deg_{x,y}(P) < (m, d) \\ p(a) = P(x, a) \bmod f \end{cases}$$

transform $\alpha_k(x)$ into $A_k(x, y)$ such that
$$\begin{cases} \deg_{x,y}(A_k) < (d, m) \\ a^k = A_k(x, a) \bmod f \end{cases}$$

computations modulo $\mathcal{I} = \langle f(x), y - a(x) \rangle$

$$p(y) = P(x, y) \bmod \mathcal{I}$$

$$y^k = a(x)^k = A_k(x, y) \bmod \mathcal{I}$$

$\mathbb{K}[y]$ -relation matrix: in $\mathbb{K}[y]^{m \times m}$

- ▶ “good” **basis** of $\mathcal{I} \cap \mathbb{K}[x, y]_{< (m, \cdot)}$
- ▶ **generically** small degrees $\sim n/m$
- ▶ fast computation: **truncated powers** and matrix Padé reconstruction

[Villard '18] [N.-S.-S.-V. '24]

$\mathbb{K}[x]$ -relation matrix: in $\mathbb{K}[x]^{m \times m}$

- ▶ “good” **basis** of $\mathcal{I} \cap \mathbb{K}[x, y]_{< (\cdot, m)}$
- ▶ **generically** small degrees $\sim n/m$
- ▶ fast computation: **shifted Popov basis of relations** $\sum_{k < m} q_k a^k = 0 \bmod f$

[Storjohann '06] [Zhou-Labahn '12] [N. '16]

balancing degrees: from univariate to bivariate

choose parameters $m \ll d$ with $md = \Theta(n)$

transform $p(y)$ into $P(x, y)$ such that
$$\begin{cases} \deg_{x,y}(P) < (m, d) \\ p(a) = P(x, a) \bmod f \end{cases}$$

transform $\alpha_k(x)$ into $A_k(x, y)$ such that
$$\begin{cases} \deg_{x,y}(A_k) < (d, m) \\ a^k = A_k(x, a) \bmod f \end{cases}$$

computations modulo $\mathcal{I} = \langle f(x), y - a(x) \rangle$

$$p(y) = P(x, y) \bmod \mathcal{I}$$

$$y^k = a(x)^k = A_k(x, y) \bmod \mathcal{I}$$

$\mathbb{K}[y]$ -relation matrix: in $\mathbb{K}[y]^{m \times m}$
[Villard '18] [N.-S.-S.-V. '24]

$\mathbb{K}[x]$ -relation matrix: in $\mathbb{K}[x]^{m \times m}$
[Storjohann '06] [Zhou-Labahn '12] [N. '16]

cost bound: $\tilde{O}(n^{(\omega+3)/4})$, taking $m = n^{1/4}$
 $\tilde{O}(n^{5/4})$ for powers
+ $\tilde{O}(m^\omega \frac{n}{m})$ for polynomial matrix operations

$\mathbb{K}[x]$ -relation matrix: fast algebraic approximants

ideal $\mathcal{I} = \langle f(x), y - a(x) \rangle$:

set of all $G(x, y)$ such that $G(x, a(x)) = 0 \bmod f(x)$

$\rightsquigarrow$ seek $G = \sum_{k < m} p_k(x) y^k \in \mathcal{I}$ of small x -degree

find a minimal basis of

$$\left\{ [p_0(x) \ p_1(x) \ \cdots \ p_{m-1}(x)] \mid \sum_{k < m} p_k(x) a^k(x) = 0 \bmod f \right\}$$

approximation problem: Hermite-Padé / vector M-Padé

deterministic, no genericity, cost $\tilde{O}(m^{\omega-1}n)$ for any $m < n$

[Beckermann-Labahn '97, Storjohann '06, Zhou-Labahn '12, Neiger '16]

algorithms designed for $(a_0, a_1, \dots, a_{m-1})$ instead of $(1, a, a^2, \dots, a^{m-1}) \dots$

[reminder] minimal polynomial mod $f(x)$

ideal $\mathcal{J} = \langle f(x), y - a(x) \rangle$:

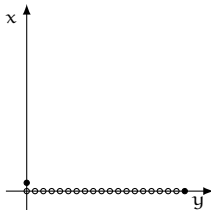
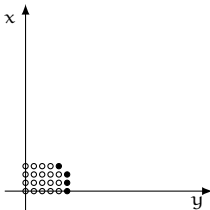
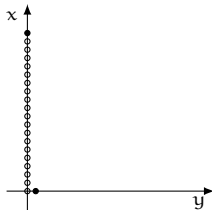
set of all $G(x, y)$ such that $G(x, a(x)) = 0 \bmod f(x)$

minimal polynomial = $g(y)$ of smallest degree in $\mathcal{J}$

example: $g(y) = (y - 1)^{16}$ is the minpoly
of $a(x) = x^2 + 1$ modulo $f(y) = x^{32}$

relation to **bivariate resultant**, and specific **ideal bases**

$$\mathcal{J} = \langle f(x), y - a(x) \rangle = \langle g(y), x - b(y) \rangle$$



using power projections

[Shoup 1994, 1999]

0. choose **random** vector $[\ell_1 \ \cdots \ \ell_n] \in \mathbb{K}^n$
 $\rightarrow$ defines a linear form $\ell : \mathbb{K}[x]/\langle f \rangle \rightarrow \mathbb{K}$
1. compute **linear recurrent sequence**
 $\ell(1), \ell(a \bmod f), \dots, \ell(a^{2n-1} \bmod f)$
2. compute **minimal recurrence relation** $g(y)$
via Berlekamp-Massey / Padé approximation

$$\begin{aligned} & \text{minpoly } g(y) \\ & \quad \downarrow \\ & g(a) = 0 \bmod f \\ & \quad \downarrow \\ & g(y) = \text{relation for } (a^k \bmod f)_k \\ & \quad \downarrow \\ & g(y) = \text{relation for } (\ell(a^k \bmod f))_k \end{aligned}$$

using power projections

[Shoup 1994, 1999]

0. choose **random** vector $[\ell_1 \ \cdots \ \ell_n] \in \mathbb{K}^n$

→ defines a linear form $\ell : \mathbb{K}[x] / \langle f \rangle \rightarrow \mathbb{K}$

1. compute **linear recurrent sequence**

$$\ell(1), \ell(a \bmod f), \dots, \ell(a^{2^n-1} \bmod f)$$

2. compute **minimal recurrence relation** $g(y)$

via Berlekamp-Massey / Padé approximation

$$\text{minpoly } g(y)$$

$$\Downarrow$$

$$g(a) = 0 \bmod f$$

$$\Downarrow$$

$$g(y) = \text{relation for } (a^k \bmod f)_k$$

$$\Downarrow$$

$$g(y) = \text{relation for } (\ell(a^k \bmod f))_k$$

→ related to algorithm of [Wiedemann 1986]:

$$\ell(a^k \bmod f) = [\ell_1 \ \cdots \ \ell_n] A^k \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

where $A \in \mathbb{K}^{n \times n}$ is the “multiplication matrix” of $a(x)$ modulo $f(x)$

for **generic** $a(x)$ and $f(0) \neq 0$, choose $\ell = [1 \ 0 \ \cdots \ 0]$

then $\ell(a^k \bmod f) = \text{constant coeff of } a^k \bmod f$

$\mathbb{K}[y]$ -relation matrix: blocking & baby-step giant-step

block Wiedemann approach

[Coppersmith 1994]

iterating projection by $1 \times n$ vector on powers $A^0, A^1, \dots, A^{2n-1}$

$\Rightarrow$ iterating projection by $m \times n$ matrix on powers $A^0, A^1, \dots, A^{2d-1}$

choose $m \ll n$ and take $d = n/m$

$\mathbb{K}[y]$ -relation matrix: blocking & baby-step giant-step

block Wiedemann approach [Coppersmith 1994]

iterating projection by $1 \times n$ vector on powers $A^0, A^1, \dots, A^{2n-1}$
 $\Rightarrow$ iterating projection by $m \times n$ matrix on powers $A^0, A^1, \dots, A^{2d-1}$

choose $m \ll n$ and take $d = n/m$

1. compute linear recurrent matrix sequence:

$$I_m, \begin{bmatrix} I_m & 0 \end{bmatrix} A \begin{bmatrix} I_m \\ 0 \end{bmatrix}, \dots, \begin{bmatrix} I_m & 0 \end{bmatrix} A^{2d-1} \begin{bmatrix} I_m \\ 0 \end{bmatrix}$$

2. compute minimal matrix recurrence relation $R_y \in \mathbb{K}[y]^{m \times m}$
via matrix-Berlekamp-Massey / matrix-Padé, complexity $\tilde{O}(m^\omega d)$

$\mathbb{K}[y]$ -relation matrix: blocking & baby-step giant-step

block Wiedemann approach

[Coppersmith 1994]

iterating projection by $1 \times n$ vector on powers $A^0, A^1, \dots, A^{2n-1}$

$\Rightarrow$ iterating projection by $m \times n$ matrix on powers $A^0, A^1, \dots, A^{2d-1}$

choose $m \ll n$ and take $d = n/m$

1. compute linear recurrent matrix sequence:

$$I_m, \begin{bmatrix} I_m & 0 \end{bmatrix} A \begin{bmatrix} I_m \\ 0 \end{bmatrix}, \dots, \begin{bmatrix} I_m & 0 \end{bmatrix} A^{2d-1} \begin{bmatrix} I_m \\ 0 \end{bmatrix}$$

2. compute minimal matrix recurrence relation $R_y \in \mathbb{K}[y]^{m \times m}$
via matrix-Berlekamp-Massey / matrix-Padé, complexity $\tilde{O}(m^\omega d)$

step 1: computing coefficient i of $x^j a^k \bmod f$, for $i, j < m$, $k < 2d$
 $\rightarrow$ baby-step giant-step in $\tilde{O}(md^{(\omega+1)/2})$

- ▶ $g(y) = \det(R_y(y))$ is the minimal polynomial of α modulo f
- ▶ $R_y(y)$ is a good basis of $\mathcal{I} = \langle f(x), y - \alpha(x) \rangle$

good: $\deg(R_y) \leq d$, Popov form, predictable degrees, ...

balancing degrees: from univariate to bivariate

choose parameters $m \ll d$ with $md = \Theta(n)$

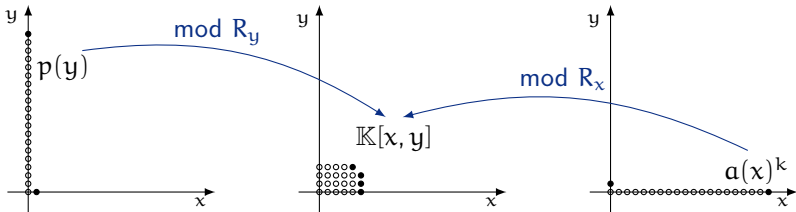
transform $p(y)$ into $P(x, y)$ such that $\begin{cases} \deg_{x,y}(P) < (m, d) \\ p(a) = P(x, a) \bmod f \end{cases}$

transform $\alpha_k(x)$ into $A_k(x, y)$ such that $\begin{cases} \deg_{x,y}(A_k) < (d, m) \\ a^k = A_k(x, a) \bmod f \end{cases}$

computations modulo $\mathcal{I} = \langle f(x), y - a(x) \rangle$

$$p(y) = P(x, y) \bmod \mathcal{I}$$

$$y^k = a(x)^k = A_k(x, y) \bmod \mathcal{I}$$



genericity and randomization

non-generic $\alpha(x) \longrightarrow$

- $\mathcal{J} \cap \mathbb{K}[x, y]_{<(m, \cdot)}$ might not generate $\mathcal{J}$ (not an issue)
- finding a basis of $\mathcal{J} \cap \mathbb{K}[x, y]_{<(m, \cdot)}$ seems more difficult

randomization by change of basis

take a random $\gamma \in \mathbb{K}[x]/\langle f(x) \rangle$

w.h.p. γ has minimal polynomial $g(y)$ of degree n

$\Rightarrow 1, \gamma, \gamma^2, \dots, \gamma^{n-1}$ is a basis of $\mathbb{K}[x]/\langle f(x) \rangle$

$\Rightarrow$ isomorphism $\begin{array}{ccc} \mathbb{K}[x]/\langle f(x) \rangle & \rightarrow & \mathbb{K}[y]/\langle g(y) \rangle \\ \alpha(x) & \mapsto & b(y) \text{ such that } b(\gamma) = \alpha \bmod f \end{array}$

tentative algorithm:

1. compute $b(y)$ and $g(y)$
2. compute $\beta(y) = h(b(y)) \bmod g(y)$
3. return $\beta(\gamma(x)) \bmod f(x)$

$\rightsquigarrow$ Las Vegas composition when working with $\mathbb{K}[y]$ -relations only

$\rightsquigarrow$ this has exponent $\frac{\omega+2}{3}$; improving to $\frac{\omega+3}{4}$ seems within reach

bi-level structures: an open problem

structured relations

$$p_1 a_0 + p_2 a_1 + \cdots + p_m a_{m-1} = 0 \bmod f$$

↓
structured a_i 's

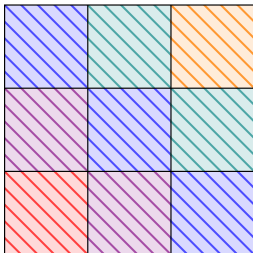
$$p_1 1 + p_2 a + \cdots + p_m a^{m-1} = 0 \bmod f$$

$$p_1 f + p_2 a' + \cdots + p_m a^{(m-1)} = 0 \bmod f$$

$\rightsquigarrow$ P-recursive recurrences, algebraic/D-finite series, modular composition, bivariate interpolation, ...

- ▶ here, input/output size is $O(n)$
- ▶ most algorithms ignore this structure
- ▶ some recent progress, but restricted

how to leverage this structure?



BTTB: block Toeplitz with Toeplitz blocks

$$\begin{bmatrix} 1 & 1 & \cdots & 1 \\ \alpha_1 & \alpha_2 & \cdots & \alpha_8 \\ \alpha_1^2 & \alpha_2^2 & \cdots & \alpha_8^2 \\ \alpha_1^3 & \alpha_2^3 & \cdots & \alpha_8^3 \\ \alpha_1^4 & \alpha_2^4 & \cdots & \alpha_8^4 \\ \hline \beta_1 & \beta_2 & \cdots & \beta_8 \\ \alpha_1 \beta_1 & \alpha_2 \beta_2 & \cdots & \alpha_8 \beta_8 \\ \alpha_1^2 \beta_1 & \alpha_2^2 \beta_2 & \cdots & \alpha_8^2 \beta_8 \\ \hline \beta_1^2 & \beta_2^2 & \cdots & \beta_8^2 \end{bmatrix}$$

bi-level Vandermonde

outline

▶ context and result

- ▶ univariate polynomial computations
- ▶ modular composition and minpoly
- ▶ improved algebraic complexity bound

▶ motivations and approach

- ▶ complexity and software motivations
- ▶ baby steps – giant steps
- ▶ computing fewer powers $a^k \bmod f$

▶ overview of the algorithms

- ▶ reductions modulo a bivariate ideal
- ▶ exploiting small $\mathbb{K}[x]$ - and $\mathbb{K}[y]$ -relations
- ▶ ? genericity $\longrightarrow$ randomization ?

▶ a soft epilogue

outline

▶ context and result

- ▶ univariate polynomial computations
- ▶ modular composition and minpoly
- ▶ improved algebraic complexity bound

▶ motivations and approach

- ▶ complexity and software motivations
- ▶ baby steps – giant steps
- ▶ computing fewer powers $a^k \bmod f$

▶ overview of the algorithms

- ▶ reductions modulo a bivariate ideal
- ▶ exploiting small $\mathbb{K}[x]$ - and $\mathbb{K}[y]$ -relations
- ▶ ? genericity $\longrightarrow$ randomization ?

▶ a soft epilogue

- ▶ libraries NTL/FLINT and PML
- ▶ recent and current developments
- ▶ wishlist and aims

open-source mathematics software system



Python/Cython

The Hermite form is also normalized, i.e., the pivot polynomials are monic.

goals: **complete, robust, available**

and, ultimately, **performance**

via interfaces to FLINT, LinBox, NTL, ...

OUTPUT:

```
sage: M.<M> = GF(7)[[]]
sage: A = matrix(M, 2, 3, [x, 1, 2*x, x, 1+x, 2])
sage: A.hermite_form()
[[ [ x 1 2*x]
[ 0 x 5*x + 2]
sage: A.hermite_form(transformation=True)
[[ [ x 1 2*x] [1 0]
[ 0 x 5*x + 2] [6 1]
]
sage: A = matrix(M, 2, 3, [x, 1, 2*x, 2*x, 2, 4*x])
sage: A.hermite_form(transformation=True, include_zero_rows=False)
([ [ x 1 2*x], [0 4]
[ 0 0 0], [5 1]
]
sage: H, U = A.hermite_form(transformation=True, include_zero_rows=True); H, U
[[ [ x 1 2*x] [0 4]
[ 0 0 0], [5 1]
]
sage: U * A == H
True
sage: H, U = A.hermite_form(transformation=True, include_zero_rows=False)
sage: U * A
[[ x 1 2*x]
sage: U * A == H
True
```

See also: `is_hermite()`.

`is_hermite(row_wise=True, lower_echelon=False, include_zero_vectors=True)`

Return a boolean indicating whether this matrix is in Hermite form.

high-performance matrix/polynomail libraries

FLINT – LinBox – NTL C/C++

goal: **fully optimized core operations**

careful implementations of the best algorithms

software development for polynomial matrices

```
187 j = std::distance(rem_order.begin(), std::max_element(rem_order.b
);
188
189 long deg = order[rem_index[j]] - rem_order[j];
190
191 // record the coefficients of degree deg of the column j of residual
192 // also keep track of which of these are nonzero,
193 // and among the nonzero ones, which is the first with smallest shift
194 Vec<zz_p> const_residual;
195 const_residual.SetLength(rdim);
196 VecLong indices_nonzero;
197 long piv = -1;
198 for (long i = 0; i < rdim; ++i)
199 {
200     const_residual[i] = coeff(residual[i][j], deg);
201     if (const_residual[i] != 0)
202     {
203         indices_nonzero.push_back(i);
204         if (piv < 0 || shift[i] < shift[piv])
205             piv = i;
206     }
207 }
208
209 // if indices_nonzero is empty, const_residual is already zero, there
210 if (not indices_nonzero.empty())
211 {
212     // update all rows of appbas and residual in indices_nonzero exce
src/mat_lzz_pX_approxinant.cpp
```

open-source mathematics software system



Python/Cython

The Hermite form is also normalized, i.e., the pivot polynomials are monic.

goals: **complete, robust, available**

and, ultimately, **performance**

via interfaces to FLINT, LinBox, NTL, ...

OUTPUT:

high-performance matrix/polynomials libraries

FLINT – LinBox – NTL C/C++

goal: **fully optimized core operations**

careful implementations of the best algorithms

software development for polynomial matrices

Polynomial Matrix Library C/C++

585 files, 101k lines of code, including 28k lines of comments

<https://github.com/vneiger/pml>

[Hyun-Neiger-Schost '19]

► NTL-based version: stable, but no new features

► current development in synergy with FLINT

► **welcome** comments, suggestions, contributions

"hey, this doesn't work!"

"hi, any plans for implementing this?"

"how to compute bivariate resultants with PML?"

wide range of algorithms

efficiency = state of the art

kernel, high-order lifting,
system solving, reduced form...

polynomial matrix multiplication

most fundamental operation $\Rightarrow$ must be thoroughly optimized

Schönhage's rule 2: Do not waste a factor of two!

various algorithms + use of thresholds:

- ▶ small size or small degree: **Karatsuba**, **Strassen**, **Waksman**, ...
- ▶ medium degrees: **Vandermonde** evaluation-interpolation
- ▶ medium size and large degrees: **FFT** evaluation-interpolation (multimodular computations with several FFT primes)
- ▶ large size and large degrees: **special** evaluation-interpolation (using points in geometric progression)

m	d	FLINT	FFT	geometric	Waksman	Vandermonde	Vandermonde2
2	20000	1.05e-02	4.80e-03	3.06e-02	7.76e-03	∞	∞
4	20000	6.82e-02	2.43e-02	1.19e-01	4.84e-02	∞	∞
8	20000	5.80e-01	1.06e-01	4.82e-01	3.39e-01	∞	∞
16	20000	4.54e+00	5.88e-01	1.98e+00	2.52e+00	∞	∞
32	20000	3.72e+01	2.63e+00	8.28e+00	1.89e+01	∞	∞
32	1000	1.49e+00	1.14e-01	3.58e-01	7.76e-01	∞	∞
64	1000	1.20e+01	6.22e-01	1.53e+00	6.02e+00	∞	∞
128	1000	4.04e+01	3.82e+00	6.49e+00	4.76e+01	∞	∞
128	10	6.50e-02	4.98e-02	3.16e-02	2.88e-01	3.68e-02	2.81e-02
256	10	3.32e-01	2.59e-01	1.71e-01	2.42e+00	1.85e-01	1.61e-01
512	10	1.73e+00	1.55e+00	1.14e+00	2.02e+01	1.21e+00	1.08e+00
1024	10	8.94e+00	9.92e+00	7.03e+01	1.69e+02	7.42e+00	7.22e+00

fraction reconstruction and basis reduction

[Giorgi-Jeannerod-Villard 2003, Algo. PM-Basis]

reconstruct a matrix of degree $< d$ as a fraction with degrees $\leq d/2$

- i.e. Padé approximation for $m \times m$ matrices: $F = P^{-1}Q \bmod x^d$
- **interpolation** variant also implemented, modulo $\prod_{1 \leq i \leq d} (x - \alpha_i)$
- interpolation **usually faster** and often serves the **same purpose**

→ block Wiedemann, matrix Berlekamp-Massey, modular composition, bivariate resultants, ...

m	n	d	approx	interp	d	approx	interp	int-geom
8	4	32	2.22e-4	1.65e-04	32768	2.45	10.8	1.26
32	16	32	4.77e-3	3.86e-03	4096	4.20	5.42	2.50
128	64	32	2.20e-1	1.22e-01	1024	22.3	20.9	14.4
256	128	32	1.60e+0	8.16e-01	256	25.3	16.8	15.4

fresh from this week

m	n	d	approx	interp	d	approx	interp	int-geom
8	4	32	2.26e-4	1.65e-04	32768	1.12	10.3	1.22
32	16	32	4.88e-3	3.86e-03	4096	2.73	5.37	2.52
128	64	32	1.74e-1	1.14e-01	1024	17.9	19.9	13.5
256	128	32	1.08e+0	6.92e-01	256	18.8	14.9	12.7

fraction reconstruction and basis reduction

[Giorgi-Jeannerod-Villard 2003, Algo. PM-Basis]

reconstruct a matrix of degree $< d$ as a fraction with degrees $\leq d/2$

- ▶ i.e. Padé approximation for $m \times m$ matrices: $F = P^{-1}Q \bmod x^d$
- ▶ **interpolation** variant also implemented, modulo $\prod_{1 \leq i \leq d} (x - \alpha_i)$
- ▶ interpolation **usually faster** and often serves the **same purpose**

~> block Wiedemann, matrix Berlekamp-Massey, modular composition, bivariate resultants, ...

m	n	d	approx	interp	d	approx	interp	int-geom
8	4	32	2.22e-4	1.65e-04	32768	2.45	10.8	1.26
32	16	32	4.77e-3	3.86e-03	4096	4.20	5.42	2.50
128	64	32	2.20e-1	1.22e-01	1024	22.3	20.9	14.4
256	128	32	1.60e+0	8.16e-01	256	25.3	16.8	15.4

basis reduction: find a minimal form R of a matrix $A \in \mathbb{K}[x]^{m \times m}$

expansion of A^{-1} satisfies, ultimately, a **matrix recurrence of order $\leq d$**

$$A^{-1} = F_0 + F_1x + \dots + F_{md}x^{md} + \dots + F_{md+2d-1}x^{md+2d-1} \bmod x^{md+2d}$$

find F by **high-order lifting** [Storjohann 2003] and reconstruct $F = R^{-1}Q \bmod x^{2d}$

~> core ingredient in transformation to Popov form [Gupta-Sarkar-Storjohann-Valeriate '11+'12]

m	d	Newton	high-order	reconstruct	total
4	24574	1.251	1.688	8.772	10.02
16	1534	4.457	3.044	8.506	11.55
64	94	30.62	5.509	5.833	11.34
128	94	371.1	29.17	37.23	66.41

linear system solving over $\mathbb{F}_p[x]$

- ▶ Dixon's solver is often the most efficient [Dixon 1982]
- ▶ solver by minimal kernel basis is not far behind, and more general
- ▶ solver by high-order lifting seems slower

$$\begin{aligned} &\tilde{O}(m^3 d) \\ &\tilde{O}(m^\omega d) \\ &\tilde{O}(m^\omega d) \end{aligned}$$

m	d	Dixon	kernel	high-order lifting
16	1024	0.695	1.96	2.39
32	1024	2.88	8.06	13.8
128	512	37.2	84.2	266

linear system solving over $\mathbb{F}_p[x]$

- Dixon's solver is often the most efficient [Dixon 1982]
- solver by minimal kernel basis is not far behind, and more general
- solver by high-order lifting seems slower

$$\begin{aligned} &\tilde{O}(m^3 d) \\ &\tilde{O}(m^\omega d) \\ &\tilde{O}(m^\omega d) \end{aligned}$$

m	d	Dixon	kernel	high-order lifting
16	1024	0.695	1.96	2.39
32	1024	2.88	8.06	13.8
128	512	37.2	84.2	266

determinant

- expansion by minors for small dimensions ($m \leq 4$ or 5)
- evaluation/interpolation at md points in geometric progression
- solving a linear system with random right-hand side [Pan, 1988]
- triangularization via minimal kernel basis [Labahn-Neiger-Zhou, 2017]

$$\begin{aligned} &\tilde{O}(d) \\ &\tilde{O}(m^{\omega+1} d) \\ &\tilde{O}(m^\omega d) \\ &\tilde{O}(m^\omega d) \end{aligned}$$

m	d	minors	evaluation	linsolve	triangular
4	65536	0.673	1.90	5.78	0.686
16	4096	∞	3.75	3.52	6.12
32	4096	∞	26.5	15.3	32.4
64	2048	∞	109	35.9	71.0
128	512	∞	∞	40.7	71.8

various problems and reductions

reductions of most problems to polynomial matrix multiplication

matrix $m \times m$ of degree d $\rightarrow \tilde{O}(m^\omega d)$
of “average” degree $\frac{D}{m}$ $\rightarrow \tilde{O}(m^\omega \frac{D}{m})$

classical matrix operations

- ▶ multiplication
- ▶ kernel, system solving
- ▶ rank, determinant
- ▶ inversion $\tilde{O}(m^3 d)$

univariate specific operations

- ▶ truncated inverse, QuoRem
- ▶ vector M-Padé
- ▶ simultaneous M-Padé
- ▶ general univariate relations

colors indicate status:

available, close to satisfactory

available, incomplete / not optimized

not available yet, perspectives

transformation to normal forms

- ▶ triangularization: Hermite form
- ▶ row reduction: Popov form
- ▶ diagonalization: Smith form

various problems and reductions

reductions of most problems to polynomial matrix multiplication

matrix $m \times m$ of degree d $\rightarrow \tilde{O}(m^\omega d)$
of “average” degree $\frac{D}{m}$ $\rightarrow \tilde{O}(m^\omega \frac{D}{m})$

classical matrix operations

- ▶ multiplication
- ▶ kernel, system solving
- ▶ rank, determinant
- ▶ inversion $\tilde{O}(m^3 d)$

univariate specific operations

- ▶ truncated inverse, QuoRem
- ▶ vector M-Padé
- ▶ simultaneous M-Padé
- ▶ general univariate relations

transformation to normal forms

- ▶ triangularization: Hermite form
- ▶ row reduction: Popov form
- ▶ diagonalization: Smith form

colors indicate status:

available, close to satisfactory

available, incomplete / not optimized

not available yet, perspectives

conclusion and perspectives

faster algorithms & complexity results

- ▶ $P(x, y)$ in $\mathbb{K}[x, y]$ evaluated at points $(\alpha_i, \beta_i)_{1 \leq i \leq n}$
- ▶ $P(x, y)$ in $\mathbb{K}[x, y]$ composed with $y = a \bmod f$
- ▶ new techniques for computations modulo ideals $\langle f(x), y - a(x) \rangle$
baby steps–giant steps, bivariate matrices, univariate algebraic relations
- ▶ related: truncated powers, power projections, inverse composition, ...
- ▶ equivalences: bivariate composition – characteristic polynomial

Schönhage's rule 8: the development of fast algorithms is slow!

perspectives & open questions

- ▶ careful implementations of the new algorithms
(the previous $(\omega + 2)/3$ algorithm already brings a speedup for large degrees)
- ▶ further consequences on related problems
(Guruswami-Sudan decoding, bivariate resultants, algebraic approximants, ...)
- ▶ design a Las Vegas randomization approach to avoid genericity
- ▶ very open (July 26) — closed (August 26, pending peer reviewing):
any idea towards quasi-linear, or maybe $(\omega + 4)/5$?

more complete timings of series composition

n	[Brent-Kung 1978]		[Kinoshita-Li 2024]		speedup	
n	$\mathbb{Z}/r\mathbb{Z}$	$\mathbb{Z}$	$\mathbb{Z}/r\mathbb{Z}$	$\mathbb{Z}$	$\mathbb{Z}/r\mathbb{Z}$	$\mathbb{Z}$
100	0.0000779	0.000298	0.000162	0.000443	0.481	0.673
200	0.000285	0.00243	0.000347	0.00284	0.821	0.856
400	0.000869	0.0153	0.000839	0.0144	1.036	1.062
800	0.00254	0.108	0.00193	0.0683	1.316	1.581
1600	0.00763	0.682	0.00434	0.339	1.758	2.012
3200	0.0238	4.651	0.00974	1.716	2.444	2.710
6400	0.0776	33.263	0.0218	8.462	3.560	3.931
12800	0.257	266.59	0.0487	43.331	5.277	6.152
25600	0.882		0.115		7.670	
51200	3.036		0.266		11.414	
102400	10.82		0.646		16.749	
204800	37.76		1.556		24.267	
409600	140.573		3.736		37.627	
819200	497.694		10.274		48.442	

some timings of modular composition on NTL

complexity: $\tilde{O}(n^{3/2})$ for $O(\sqrt{n})$ multiplications by a and a modulo g
+ $O(n^{(\omega+1)/2})$ for matrix multiplication

in practice: ▶ **much faster** than naive approach
▶ $\tilde{O}(n^{3/2})$ **regime lasts** until largish n

```
// Horner evaluation h(a), modulo g :
zz_pX b;
b = coeff(h, n-1);
for (long k = n-2; k >= 0; --k)
{
    b = (a * b) % g;
    b = b + coeff(h, k);
}
```

n	Horner	Horner with precomputations	NTL built-in Brent-Kung
100	0.00229	0.00227	0.000441
200	0.0162	0.00691	0.00110
400	0.117	0.0278	0.00312
800	0.637	0.116	0.00944
1600	2.52	0.515	0.0281
3200	10.4	2.23	0.0884
6400	45.8	9.61	0.273

field $\mathbb{K} = \mathbb{F}_p$, prime p with 60 bits
NTL 11.4.3 on Intel Core i7-7600U @ 2.80GHz

software improvements

efficient implementation for the **minimal polynomial**
for large degrees, **outperforms the state of the art**

algorithm of [Neiger-Salvy-Schost-Villard '24]

field $\mathbb{K} = \mathbb{F}_p$, prime p with 60 bits

Intel Core i7-7600U @ 2.80GHz

random input polynomials $\Rightarrow$ “generic”

n	general prime		FFT prime	
	NTL	new	NTL	new
5k	0.349	0.496	0.130	0.208
20k	3.13	3.19	1.21	1.39
80k	31.5	23.6	13.9	10.7
320k	311	178	158	91.0

uses many types of computations on matrices over $\mathbb{K}[x]$

$\leadsto$ relies on the Polynomial Matrix Library

- multiplication for various parameters
- matrix-Padé approximation
- matrix division with remainder
- determinant
- system solving
- kernel

<https://github.com/vneiger/pml>