Vincent Neiger
LIP6, Sorbonne Université, France

designing and exploiting fast algorithms for univariate polynomial matrices

Journées Nationales de Calcul Formel
Centre International de Rencontre Mathématiques
Marseille Luminy, France, 4 March 2024

outline

- approximate/interpolate
- characteristic polynomial
- modular composition
- change of order

outline

- **approximate/interpolate**
  - introduction, links with structured matrices
  - vector interpolation & matrix normal forms
  - iterative & divide and conquer algorithms
- **characteristic polynomial**
- **modular composition**
- **change of order**

Padé approximation:

given power series $f(x)$ at precision d,
given degree constraints $d_1, d_2 > 0$,
$\rightarrow$ compute polynomials $(p(x), q(x))$ of degrees $< (d_1, d_2)$
and such that $f = \frac{p}{q} \bmod x^d$

strong links with linearly recurrent sequences

Padé approximation:

given power series $f(x)$ at precision d,
given degree constraints $d_1, d_2 > 0$,
$\rightarrow$ compute polynomials $(p(x), q(x))$ of degrees $< (d_1, d_2)$
and such that $f = \frac{p}{q} \bmod x^d$

strong links with linearly recurrent sequences

Cauchy interpolation:

given $M(x) = (x - \alpha_1) \cdots (x - \alpha_d) \in \mathbb{K}[x]$,
for pairwise distinct $\alpha_1, \ldots, \alpha_d \in \mathbb{K}$,
given degree constraints $d_1, d_2 > 0$,
$\rightarrow$ compute polynomials $(p(x), q(x))$ of degrees $< (d_1, d_2)$
and such that $f = \frac{p}{q} \bmod M(x)$

Padé approximation:

given power series $f(x)$ at precision d,
given degree constraints $d_1, d_2 > 0$,
$\rightarrow$ compute polynomials $(p(x), q(x))$ of degrees $< (d_1, d_2)$
and such that $f = \frac{p}{q} \bmod x^d$

strong links with linearly recurrent sequences

Cauchy interpolation:

given $M(x) = (x - \alpha_1) \cdots (x - \alpha_d) \in \mathbb{K}[x]$,
for pairwise distinct $\alpha_1, \ldots, \alpha_d \in \mathbb{K}$,
given degree constraints $d_1, d_2 > 0$,
$\rightarrow$ compute polynomials $(p(x), q(x))$ of degrees $< (d_1, d_2)$
and such that $f = \frac{p}{q} \bmod M(x)$

- degree constraints specified by the context
- usual choices have $d_1 + d_2 \approx d$ and existence of a solution

$\mathbb{K} = \mathbb{F}_7$
$f = 2x^7 + 2x^6 + 5x^4 + 2x^2 + 4$
$d = 8$, $d_1 = 3$, $d_2 = 6$
$\rightarrow$ look for (p, q) of degree $< (3, 6)$ such that $f = \frac{p}{q} \bmod x^8$

$$\begin{bmatrix} q & p \end{bmatrix} \begin{bmatrix} f \\ -1 \end{bmatrix} = 0 \bmod x^8$$

$\mathbb{K} = \mathbb{F}_7$
$f = 2x^7 + 2x^6 + 5x^4 + 2x^2 + 4$
$d = 8,\ d_1 = 3,\ d_2 = 6$
$\rightarrow$ look for (p, q) of degree $< (3, 6)$ such that $f = \frac{p}{q} \bmod x^8$

$$\begin{bmatrix} q & p \end{bmatrix} \begin{bmatrix} f \\ -1 \end{bmatrix} = 0 \bmod x^8$$

$$[q_0\ \ q_1\ \ q_2\ \ q_3\ \ q_4\ \ 1 \mid p_0\ \ p_1\ \ p_2] \left[\begin{array}{cccccccc} 4 & 0 & 2 & 0 & 5 & 0 & 2 & 2 \\ & 4 & 0 & 2 & 0 & 5 & 0 & 2 \\ & & 4 & 0 & 2 & 0 & 5 & 0 \\ & & & 4 & 0 & 2 & 0 & 5 \\ & & & & 4 & 0 & 2 & 0 \\ & & & & & 4 & 0 & 2 \\ \hdashline 6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ & 6 & 0 & 0 & 0 & 0 & 0 & 0 \\ & & 6 & 0 & 0 & 0 & 0 & 0 \end{array}\right] = 0$$

approximation and structured linear system

$\mathbb{K} = \mathbb{F}_7$
$f = 2x^7 + 2x^6 + 5x^4 + 2x^2 + 4$
$d = 8,\ d_1 = 3,\ d_2 = 6$
$\rightarrow$ look for (p, q) of degree $< (3, 6)$ such that $f = \frac{p}{q} \bmod x^8$

$$\begin{bmatrix} q & p \end{bmatrix} \begin{bmatrix} f \\ -1 \end{bmatrix} = 0 \bmod x^8$$

$$[q_0\ \ q_1\ \ q_2\ \ q_3\ \ q_4\ \ 1 \mid p_0\ \ p_1\ \ p_2] \left[\begin{array}{cccccccc} 4 & 0 & 2 & 0 & 5 & 0 & 2 & 2 \\ & 4 & 0 & 2 & 0 & 5 & 0 & 2 \\ & & 4 & 0 & 2 & 0 & 5 & 0 \\ & & & 4 & 0 & 2 & 0 & 5 \\ & & & & 4 & 0 & 2 & 0 \\ & & & & & 4 & 0 & 2 \\ \hdashline 6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ & 6 & 0 & 0 & 0 & 0 & 0 & 0 \\ & & 6 & 0 & 0 & 0 & 0 & 0 \end{array}\right] = 0$$

Sur la généralisation des fractions continues algébriques;

PAR M. H. PADÉ,

Docteur ès Sciences mathématiques,
Professeur au lycée de Lille.

[1894, Journal de mathématiques pures et appliquées]

INTRODUCTION.

M. Hermite s'est, dans un travail récemment paru [1], occupé de la généralisation des fractions continues algébriques. La question est de déterminer les polynomes $X_1, X_2, \ldots, X_n$, de degrés $\mu_1, \mu_2, \ldots, \mu_n$, qui satisfont à l'équation

$$S_1 X_1 + S_2 X_2 + \ldots + S_n X_n = S\, x^{\mu_1+\mu_2+\ldots+\mu_n+n-1},$$

$S_1, S_2, \ldots, S_n$ étant des séries entières données, et S une série également entière. Ou plutôt, il s'agit d'obtenir un algorithme qui permette le calcul de proche en proche de ces systèmes de n polynomes, et qui soit analogue à l'algorithme par lequel le numérateur et le dénominateur d'une réduite d'une fraction continue se déduisent des numérateurs et dénominateurs des réduites précédentes. D'élégantes considé-

Hermite-Padé approximation

[Hermite 1893, Padé 1894]

input:

- polynomials $f_1, \ldots, f_m \in \mathbb{K}[x]$
- precision $d \in \mathbb{Z}_{>0}$
- degree bounds $d_1, \ldots, d_m \in \mathbb{Z}_{>0}$

output:

polynomials $p_1, \ldots, p_m \in \mathbb{K}[x]$ such that

- $p_1 f_1 + \cdots + p_m f_m = 0 \bmod x^d$
- $\deg(p_i) < d_i$ for all i

(Padé approximation: particular case $m = 2$ and $f_2 = -1$)

M-Padé approximation / vector rational interpolation
[Cauchy 1821, Mahler 1968]

input:

- polynomials $f_1, \ldots, f_m \in \mathbb{K}[x]$
- pairwise distinct points $\alpha_1, \ldots, \alpha_d \in \mathbb{K}$
- degree bounds $d_1, \ldots, d_m \in \mathbb{Z}_{>0}$

output:

polynomials $p_1, \ldots, p_m \in \mathbb{K}[x]$ such that

- $p_1(\alpha_i)f_1(\alpha_i) + \cdots + p_m(\alpha_i)f_m(\alpha_i) = 0$ for all $1 \leqslant i \leqslant d$
- $\deg(p_i) < d_i$ for all i

(rational interpolation: particular case $m = 2$ and $f_2 = -1$)

approximation and interpolation

the vector case

this talk: modular equation and fast algebraic algorithms
[van Barel-Bultheel 1992; Beckermann-Labahn 1994, 1997, 2000; Giorgi-Jeannerod-Villard 2003; Storjohann 2006; Zhou-Labahn 2012; Jeannerod-Neiger-Schost-Villard 2017, 2020]

input:

- polynomials $f_1, \ldots, f_m \in \mathbb{K}[x]$
- field elements $\alpha_1, \ldots, \alpha_d \in \mathbb{K}$ $\rightsquigarrow$ not necessarily distinct
- degree bounds $d_1, \ldots, d_m \in \mathbb{Z}_{>0}$ $\rightsquigarrow$ general "shift" $\mathbf{s} \in \mathbb{Z}^m$

output:

polynomials $p_1, \ldots, p_m \in \mathbb{K}[x]$ such that

- $p_1 f_1 + \cdots + p_m f_m = 0 \bmod \prod_{1 \leqslant i \leqslant d}(x - \alpha_i)$
- $\deg(p_i) < d_i$ for all i $\rightsquigarrow$ minimal $\mathbf{s}$-row degree

(Hermite-Padé: $\alpha_1 = \cdots = \alpha_d = 0$; interpolation: pairwise distinct points)

(bivariate) interpolation and structured linear system

application to bivariate interpolation:
given pairwise distinct points $\{(\alpha_i, \beta_i), 1 \leqslant i \leqslant 8\}$
$= \{(24, 80), (31, 73), (15, 73), (32, 35), (83, 66), (27, 46), (20, 91), (59, 64)\}$,
compute a bivariate polynomial $Q(x, y) \in \mathbb{K}[x, y]$
such that $Q(\alpha_i, \beta_i) = 0$ for $1 \leqslant i \leqslant 8$

$$\left.\begin{array}{l} M(x) = (x - 24) \cdots (x - 59) \\ L(x) = \text{Lagrange interpolant} \end{array}\right\} \longrightarrow \text{solutions} = \text{ideal } \langle M(x), y - L(x) \rangle$$

solutions of smaller x-degree: $Q(x, y) = Q_0(x) + Q_1(x)y + Q_2(x)y^2$

$$Q(x, L(x)) = \begin{bmatrix} Q_0 & Q_1 & Q_2 \end{bmatrix} \begin{bmatrix} 1 \\ L \\ L^2 \end{bmatrix} = 0 \bmod M(x)$$

- instance of univariate rational vector interpolation
- with a structured input equation (powers of $L \bmod M$)

(bivariate) interpolation and structured linear system

application to bivariate interpolation:
given pairwise distinct points $\{(\alpha_i, \beta_i), 1 \leqslant i \leqslant 8\}$
$= \{(24,80),(31,73),(15,73),(32,35),(83,66),(27,46),(20,91),(59,64)\}$,
compute a bivariate polynomial $Q(x,y) \in \mathbb{K}[x,y]$
such that $Q(\alpha_i, \beta_i) = 0$ for $1 \leqslant i \leqslant 8$

add degree constraints: seek $Q(x,y)$ of the form
$q_{00} + q_{01}x + q_{02}x^2 + q_{03}x^3 + q_{04}x^4 + (q_{10} + q_{11}x + q_{12}x^2)y + q_{20}y^2$:

$$
\left[\begin{array}{ccccc:ccc:c} q_{00} & q_{01} & q_{02} & q_{03} & q_{04} & q_{10} & q_{11} & q_{12} & q_{20} \end{array}\right]
\left[\begin{array}{cccc}
1 & 1 & \cdots & 1 \\
\alpha_1 & \alpha_2 & \cdots & \alpha_8 \\
\alpha_1^2 & \alpha_2^2 & \cdots & \alpha_8^2 \\
\alpha_1^3 & \alpha_2^3 & \cdots & \alpha_8^3 \\
\alpha_1^4 & \alpha_2^4 & \cdots & \alpha_8^4 \\
\hdashline
\beta_1 & \beta_2 & \cdots & \beta_8 \\
\alpha_1\beta_1 & \alpha_2\beta_2 & \cdots & \alpha_8\beta_8 \\
\alpha_1^2\beta_1 & \alpha_2^2\beta_2 & \cdots & \alpha_8^2\beta_8 \\
\hdashline
\beta_1^2 & \beta_2^2 & \cdots & \beta_8^2
\end{array}\right] = 0
$$

- $\mathbb{K}$-linear system
- two levels of structure

$$Q(x,y) = (2x^4 + 56x^3 + 42x^2 + 48x + 15) + (72x^2 + 12x + 30)y + y^2$$

why polynomial matrices here?

why polynomial matrices here?

omitting degree constraints, the set of solutions is
$\mathcal{S} = \{(p_1, \ldots, p_m) \in \mathbb{K}[x]^m \mid p_1 f_1 + \cdots + p_m f_m = 0 \bmod M\}$

recall $M(x) = \prod_{1 \leqslant i \leqslant d}(x - \alpha_i)$

why polynomial matrices here?

omitting degree constraints, the set of solutions is
$\mathcal{S} = \{(p_1, \ldots, p_m) \in \mathbb{K}[x]^m \mid p_1 f_1 + \cdots + p_m f_m = 0 \bmod M\}$

recall $M(x) = \prod_{1 \leqslant i \leqslant d}(x - \alpha_i)$

$\mathcal{S}$ is a "free $\mathbb{K}[x]$-module of rank m": admits a basis consisting of m elements

why polynomial matrices here?

omitting degree constraints, the set of solutions is
$\mathcal{S} = \{(p_1, \ldots, p_m) \in \mathbb{K}[x]^m \mid p_1 f_1 + \cdots + p_m f_m = 0 \bmod M\}$

recall $M(x) = \prod_{1 \leqslant i \leqslant d}(x - \alpha_i)$

$\mathcal{S}$ is a "free $\mathbb{K}[x]$-module of rank m": admits a basis consisting of m elements

basis of solutions:

- square nonsingular matrix $\mathbf{P}$ in $\mathbb{K}[x]^{m \times m}$
- each row of $\mathbf{P}$ is a solution $[p_{i,1} \ \cdots \ p_{i,m}]$
- any solution is a $\mathbb{K}[x]$-combination $\mathbf{uP}$, $\mathbf{u} \in \mathbb{K}[x]^{1 \times m}$

i.e. $\mathcal{S}$ is the $\mathbb{K}[x]$-row space of $\mathbf{P}$

polynomial matrices enter the arena

why polynomial matrices here?

omitting degree constraints, the set of solutions is
$\mathcal{S} = \{(p_1, \ldots, p_m) \in \mathbb{K}[x]^m \mid p_1 f_1 + \cdots + p_m f_m = 0 \bmod M\}$

recall $M(x) = \prod_{1 \leqslant i \leqslant d}(x - \alpha_i)$

$\mathcal{S}$ is a "free $\mathbb{K}[x]$-module of rank m": admits a basis consisting of m elements

basis of solutions:
- square nonsingular matrix $\mathbf{P}$ in $\mathbb{K}[x]^{m \times m}$
- each row of $\mathbf{P}$ is a solution $[p_{i,1} \;\cdots\; p_{i,m}]$
- any solution is a $\mathbb{K}[x]$-combination $\mathbf{uP}, \mathbf{u} \in \mathbb{K}[x]^{1 \times m}$

i.e. $\mathcal{S}$ is the $\mathbb{K}[x]$-row space of $\mathbf{P}$

computing a basis of $\mathcal{S}$ with "minimal degrees"
- has many more applications than a single small-degree solution
- is in most cases the fastest known strategy anyway(!)

$\rightsquigarrow$ degree minimality ensured via shifted reduced forms

polynomial matrices: reminder

$$\mathbf{A} = \begin{bmatrix} 3x+4 & x^3+4x+1 & 4x^2+3 \\ 5 & 5x^2+3x+1 & 5x+3 \\ 3x^3+x^2+5x+3 & 6x+5 & 2x+1 \end{bmatrix} \in \mathbb{K}[x]^{3\times 3}$$

3×3 matrix of degree 3
with entries in $\mathbb{K}[x] = \mathbb{F}_7[x]$

operations on $\mathbb{K}[x]^{m\times m}_{<d}$

- combination of matrix and polynomial computations
- addition in $O(m^2 d)$, naive multiplication in $O(m^3 d^2)$

[Cantor-Kaltofen'91]

multiplication in $O(m^\omega d \log(d) \; + \; m^2 d \log(d) \log\log(d))$

$\in O(m^\omega M(d)) \subset O\tilde{}(m^\omega d)$

applying univariate polynomial techniques directly:

- Newton truncated inversion, matrix-QuoRem $O\tilde{}(m^\omega d)$ 🥳
- inversion & determinant by evaluation-interpolation $O\tilde{}(m^{\omega+1} d)$ 😐
- vector rational approximation & interpolation ??? 🤔

applying matrix techniques directly: echelonization is exponential time 🌋

polynomial matrices: main computational problems

reductions to `PolMatMul` **via vector interpolation**

matrix $m \times m$ of degree d $\quad\rightarrow\quad O\tilde{~}(m^\omega d)$

of "average" degree $\frac{D}{m}$ $\quad\rightarrow\quad O\tilde{~}(m^\omega \frac{D}{m})$

classical matrix operations

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

univariate specific operations

- truncated inverse, QuoRem
- Hermite-Padé approximation
- vector rational interpolation
- syzygies / modular equations

transformation to **normal forms**

- echelonization: Hermite form
- row reduction: Popov form
- diagonalization: Smith form

polynomial matrices: main computational problems

reductions to `PolMatMul` **via vector interpolation**

matrix $m \times m$ of degree d $\rightarrow$ $O\tilde{~}(m^\omega d)$

of "average" degree $\frac{D}{m}$ $\rightarrow$ $O\tilde{~}(m^\omega \frac{D}{m})$

classical matrix operations

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

univariate specific operations

- truncated inverse, QuoRem
- Hermite-Padé approximation
- vector rational interpolation
- syzygies / modular equations

transformation to **normal forms**

- echelonization: Hermite form
- row reduction: Popov form
- diagonalization: Smith form

polynomial matrices: main computational problems

reductions to `PolMatMul` **via vector interpolation**

matrix $m \times m$ of degree d $\rightarrow$ $O\tilde{~}(m^\omega d)$

of "average" degree $\frac{D}{m}$ $\rightarrow$ $O\tilde{~}(m^\omega \frac{D}{m})$

classical matrix operations

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

univariate specific operations

- truncated inverse, QuoRem
- Hermite-Padé approximation
- vector rational interpolation
- syzygies / modular equations

transformation to **normal forms**

- echelonization: Hermite form
- row reduction: Popov form
- diagonalization: Smith form

polynomial matrices: main computational problems

reductions to PolMatMul **via vector interpolation**

matrix $m \times m$ of degree d $\rightarrow$ $O\tilde{~}(m^\omega d)$

of "average" degree $\frac{D}{m}$ $\rightarrow$ $O\tilde{~}(m^\omega \frac{D}{m})$

classical matrix operations

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

univariate specific operations

- truncated inverse, QuoRem
- Hermite-Padé approximation
- vector rational interpolation
- syzygies / modular equations

transformation to **normal forms**

- echelonization: Hermite form
- row reduction: Popov form
- diagonalization: Smith form

matrix normal forms

working over $\mathbb{K} = \mathbb{Z}/7\mathbb{Z}$

$$\mathbf{A} = \begin{bmatrix} 3x+4 & x^3+4x+1 & 4x^2+3 \\ 5 & 5x^2+3x+1 & 5x+3 \\ 3x^3+x^2+5x+3 & 6x+5 & 2x+1 \end{bmatrix}$$

using elementary row operations, transform $\mathbf{A}$ into...

Hermite form $$\mathbf{H} = \begin{bmatrix} x^6+6x^4+x^3+x+4 & 0 & 0 \\ 5x^5+5x^4+6x^3+2x^2+6x+3 & x & 0 \\ 3x^4+5x^3+4x^2+6x+1 & 5 & 1 \end{bmatrix}$$

Popov form $$\mathbf{P} = \begin{bmatrix} x^3+5x^2+4x+1 & 2x+4 & 3x+5 \\ 1 & x^2+2x+3 & x+2 \\ 3x+2 & 4x & x^2 \end{bmatrix}$$

Hermite and Popov forms

nonsingular $\mathbf{A} \in \mathbb{K}[x]^{m \times m}$

elementary row transformations

Hermite form [Hermite, 1851]
- triangular
- column normalized

$$\begin{bmatrix} \mathbf{16} & & & \\ 15 & \mathbf{0} & & \\ 15 & & \mathbf{0} & \\ 15 & & & \mathbf{0} \end{bmatrix} \quad \begin{bmatrix} \mathbf{4} & & & \\ 3 & \mathbf{7} & & \\ 1 & 5 & \mathbf{3} & \\ 3 & 6 & 1 & \mathbf{2} \end{bmatrix}$$

Hermite and Popov forms

nonsingular $\mathbf{A} \in \mathbb{K}[x]^{m \times m}$

elementary row transformations

Hermite form [Hermite, 1851]
- triangular
- column normalized

Popov form [Popov, 1972]
- minimal row degrees
- column normalized

$$\begin{bmatrix} \mathbf{16} & & & \\ 15 & \mathbf{0} & & \\ 15 & & \mathbf{0} & \\ 15 & & & \mathbf{0} \end{bmatrix} \quad \begin{bmatrix} \mathbf{4} & & & \\ 3 & \mathbf{7} & & \\ 1 & 5 & \mathbf{3} & \\ 3 & 6 & 1 & \mathbf{2} \end{bmatrix}$$

$$\begin{bmatrix} \mathbf{4} & 3 & 3 & 3 \\ 3 & \mathbf{4} & 3 & 3 \\ 3 & 3 & \mathbf{4} & 3 \\ 3 & 3 & 3 & \mathbf{4} \end{bmatrix} \quad \begin{bmatrix} \mathbf{7} & 0 & 1 & 5 \\ 0 & \mathbf{1} & & 0 \\ & & \mathbf{2} & \\ 6 & 0 & 1 & \mathbf{6} \end{bmatrix}$$

Hermite and Popov forms

nonsingular $\mathbf{A} \in \mathbb{K}[x]^{m \times m}$

elementary row transformations

Hermite form [Hermite, 1851]
- triangular
- column normalized

Popov form [Popov, 1972]
- minimal row degrees
- column normalized

$$\begin{bmatrix} \mathbf{16} & & & \\ 15 & \mathbf{0} & & \\ 15 & & \mathbf{0} & \\ 15 & & & \mathbf{0} \end{bmatrix} \quad \begin{bmatrix} \mathbf{4} & & & \\ 3 & \mathbf{7} & & \\ 1 & 5 & \mathbf{3} & \\ 3 & 6 & 1 & \mathbf{2} \end{bmatrix} \qquad \begin{bmatrix} \mathbf{4} & 3 & 3 & 3 \\ 3 & \mathbf{4} & 3 & 3 \\ 3 & 3 & \mathbf{4} & 3 \\ 3 & 3 & 3 & \mathbf{4} \end{bmatrix} \quad \begin{bmatrix} \mathbf{7} & 0 & 1 & 5 \\ 0 & \mathbf{1} & & 0 \\ & & \mathbf{2} & \\ 6 & 0 & 1 & \mathbf{6} \end{bmatrix}$$

$\preccurlyeq_{\mathrm{pot}}$ reduced Gröbner basis $\preccurlyeq_{\mathrm{top}}$

$\mathbb{K}[x]$-module $\mathcal{S} \subset \mathbb{K}[x]^{1 \times m}$ of rank m

Hermite and Popov forms

nonsingular $\mathbf{A} \in \mathbb{K}[x]^{m \times m}$

elementary row transformations

Hermite form [Hermite, 1851]
- triangular
- column normalized

Popov form [Popov, 1972]
- minimal row degrees
- column normalized

$$\begin{bmatrix} \mathbf{16} & & & \\ 15 & \mathbf{0} & & \\ 15 & & \mathbf{0} & \\ 15 & & & \mathbf{0} \end{bmatrix} \quad \begin{bmatrix} \mathbf{4} & & & \\ 3 & \mathbf{7} & & \\ 1 & 5 & \mathbf{3} & \\ 3 & 6 & 1 & \mathbf{2} \end{bmatrix} \qquad \begin{bmatrix} \mathbf{4} & 3 & 3 & 3 \\ 3 & \mathbf{4} & 3 & 3 \\ 3 & 3 & \mathbf{4} & 3 \\ 3 & 3 & 3 & \mathbf{4} \end{bmatrix} \quad \begin{bmatrix} \mathbf{7} & 0 & 1 & 5 \\ 0 & \mathbf{1} & & 0 \\ & & \mathbf{2} & \\ 6 & 0 & 1 & \mathbf{6} \end{bmatrix}$$

invariant: $D = \deg(\det(\mathbf{A})) = 4+7+3+2 = 7+1+2+6$

- average column degree is $\frac{D}{m}$
- size of object is $mD + m^2 = m^2(\frac{D}{m} + 1)$

target cost: $O\tilde{~}(m^{\omega} \frac{D}{m})$

Hermite and Popov forms

nonsingular $\mathbf{A} \in \mathbb{K}[x]^{m \times m}$

$$\begin{array}{c} \swarrow \quad \text{elementary row transformations} \quad \searrow \end{array}$$

Hermite form [Hermite, 1851]
- triangular
- column normalized

Popov form [Popov, 1972]
- minimal row degrees
- column normalized

$$\begin{bmatrix} \mathbf{16} & & & \\ 15 & \mathbf{0} & & \\ 15 & & \mathbf{0} & \\ 15 & & & \mathbf{0} \end{bmatrix} \quad \begin{bmatrix} \mathbf{4} & & & \\ 3 & \mathbf{7} & & \\ 1 & 5 & \mathbf{3} & \\ 3 & 6 & 1 & \mathbf{2} \end{bmatrix} \qquad \begin{bmatrix} \mathbf{4} & 3 & 3 & 3 \\ 3 & \mathbf{4} & 3 & 3 \\ 3 & 3 & \mathbf{4} & 3 \\ 3 & 3 & 3 & \mathbf{4} \end{bmatrix} \quad \begin{bmatrix} \mathbf{7} & 0 & 1 & 5 \\ 0 & \mathbf{1} & & 0 \\ & & \mathbf{2} & \\ 6 & 0 & 1 & \mathbf{6} \end{bmatrix}$$

[Beckermann-Labahn-Villard, 1999; Mulders-Storjohann, 2003]

shifted reduced form:
arbitrary degree constraints + **no** column normalization

$\approx$ minimal, non-reduced, $\prec$-Gröbner basis

shifted forms

shift: integer tuple $\mathbf{s} = (s_1, \ldots, s_m)$ acting as column weights
$\rightarrow$ connects Popov and Hermite forms

$\mathbf{s} = (0,0,0,0)$ Popov	$\begin{bmatrix} \mathbf{4} & 3 & 3 & 3 \\ 3 & \mathbf{4} & 3 & 3 \\ 3 & 3 & \mathbf{4} & 3 \\ 3 & 3 & 3 & \mathbf{4} \end{bmatrix}$	$\begin{bmatrix} \mathbf{7} & 0 & 1 & 5 \\ 0 & \mathbf{1} & & 0 \\ & & \mathbf{2} & \\ 6 & 0 & 1 & \mathbf{6} \end{bmatrix}$
$\mathbf{s} = (0,2,4,6)$ $\mathbf{s}$-Popov	$\begin{bmatrix} \mathbf{7} & 4 & 2 & 0 \\ 6 & \mathbf{5} & 2 & 0 \\ 6 & 4 & \mathbf{3} & 0 \\ 6 & 4 & 2 & \mathbf{1} \end{bmatrix}$	$\begin{bmatrix} \mathbf{8} & 5 & 1 & \\ 7 & \mathbf{6} & 1 & \\ & & \mathbf{2} & \\ 0 & 1 & & \mathbf{0} \end{bmatrix}$
$\mathbf{s} = (0, D, 2D, 3D)$ Hermite	$\begin{bmatrix} \mathbf{16} & & & \\ 15 & \mathbf{0} & & \\ 15 & & \mathbf{0} & \\ 15 & & & \mathbf{0} \end{bmatrix}$	$\begin{bmatrix} \mathbf{4} & & & \\ 3 & \mathbf{7} & & \\ 1 & 5 & \mathbf{3} & \\ 3 & 6 & 1 & \mathbf{2} \end{bmatrix}$

- normal form, average column degree D/m
- shifts arise naturally in algorithms (approximants, kernel, …)
- they allow one to specify non-uniform degree constraints

from normal forms to relations

$$\begin{cases} p_1 f_{11} + \cdots + p_m f_{1m} & = & 0 \bmod g_1 \\ \vdots & \vdots & \vdots \\ p_1 f_{n1} + \cdots + p_m f_{nm} & = & 0 \bmod g_n \end{cases}$$

reconstruction as relations

high-order lifting [Storjohann, 2003]

[Giorgi-Jeannerod-Villard 2003] [Neiger 2016] [Neiger-Vu 2017]

normal form computation

Popov form

shifted Popov form

Hermite form

iterative & divide and conquer algorithms

iterative algorithm [van Barel-Bultheel / Beckermann-Labahn / Kötter-Vardy]

input: vector $\mathbf{F} = \begin{bmatrix} f_1 \\ \vdots \\ f_m \end{bmatrix}$, points $\alpha_1, \ldots, \alpha_d \in \mathbb{K}$, shift $\boldsymbol{s} = (s_1, \ldots, s_m) \in \mathbb{Z}^m$

1. $\mathbf{P} = \begin{bmatrix} -\mathbf{p}_1- \\ \vdots \\ -\mathbf{p}_m- \end{bmatrix}$ = identity matrix in $\mathbb{K}[x]^{m \times m}$

2. for i from 1 to d:

- **a.** choose pivot π with smallest s_π such that $f_\pi(\alpha_i) \neq 0$
  update pivot shift $s_\pi = s_\pi + 1$
- **b.** constant elimination: for $j \neq \pi$ do $\mathbf{p}_j \leftarrow \mathbf{p}_j - \frac{f_j(\alpha_i)}{f_\pi(\alpha_i)} \mathbf{p}_\pi$
  polynomial elimination: $\mathbf{p}_\pi \leftarrow (x - \alpha_i)\mathbf{p}_\pi$
- **c.** compute residual equation: for $j \neq \pi$ do $f_j \leftarrow f_j - \frac{f_j(\alpha_i)}{f_\pi(\alpha_i)} f_\pi$
  $f_\pi \leftarrow (x - \alpha_i) f_\pi$

after i iterations: $\mathbf{P}$ is an $\boldsymbol{s}$-reduced basis of solutions for $(\alpha_1, \ldots, \alpha_i)$

iterative algorithm [van Barel-Bultheel / Beckermann-Labahn / Kötter-Vardy]

parameters: $d = 8$ $m = 4$ $\mathbf{s} = (0, 2, 4, 6)$, base field $\mathbb{F}_{97}$

input: $(24, 31, 15, 32, 83, 27, 20, 59)$ and $\mathbf{F} = [1 \ \ L \ \ L^2 \ \ L^3]^\mathsf{T}$

iteration: $i = 1$ **point:** 24, 31, 15, 32, 83, 27, 20, 59

shift $[0 \quad 2 \quad 4 \quad 6]$

basis $\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

values $\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 80 & 73 & 73 & 35 & 66 & 46 & 91 & 64 \\ 95 & 91 & 91 & 61 & 88 & 79 & 36 & 22 \\ 34 & 47 & 47 & 1 & 85 & 45 & 75 & 50 \end{bmatrix}$

iterative & divide and conquer algorithms

iterative algorithm [van Barel-Bultheel / Beckermann-Labahn / Kötter-Vardy]

parameters: $d = 8$ $m = 4$ $\mathbf{s} = (0, 2, 4, 6)$, base field $\mathbb{F}_{97}$

input: $(24, 31, 15, 32, 83, 27, 20, 59)$ and $\mathbf{F} = [1 \;\; L \;\; L^2 \;\; L^3]^T$

iteration: $i = 1$ **point:** 24, 31, 15, 32, 83, 27, 20, 59

shift $[0 \;\; 2 \;\; 4 \;\; 6]$

basis

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

values

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 80 & 73 & 73 & 35 & 66 & 46 & 91 & 64 \\ 95 & 91 & 91 & 61 & 88 & 79 & 36 & 22 \\ 34 & 47 & 47 & 1 & 85 & 45 & 75 & 50 \end{bmatrix}$$

iterative algorithm [van Barel-Bultheel / Beckermann-Labahn / Kötter-Vardy]

parameters: $d = 8$ $m = 4$ $\mathbf{s} = (0, 2, 4, 6)$, base field $\mathbb{F}_{97}$

input: $(24, 31, 15, 32, 83, 27, 20, 59)$ and $\mathbf{F} = [1 \;\; L \;\; L^2 \;\; L^3]^T$

iteration: $i = 1$ **point:** 24, 31, 15, 32, 83, 27, 20, 59

shift $[0 \quad 2 \quad 4 \quad 6]$

basis $\begin{bmatrix} 1 & 0 & 0 & 0 \\ 17 & 1 & 0 & 0 \\ 2 & 0 & 1 & 0 \\ 63 & 0 & 0 & 1 \end{bmatrix}$

values $\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 90 & 90 & 52 & 83 & 63 & 11 & 81 \\ 0 & 93 & 93 & 63 & 90 & 81 & 38 & 24 \\ 0 & 13 & 13 & 64 & 51 & 11 & 41 & 16 \end{bmatrix}$

iterative algorithm [van Barel-Bultheel / Beckermann-Labahn / Kötter-Vardy]

parameters: $d = 8$ $m = 4$ $\mathbf{s} = (0, 2, 4, 6)$, base field $\mathbb{F}_{97}$

input: $(24, 31, 15, 32, 83, 27, 20, 59)$ and $\mathbf{F} = [1 \;\; L \;\; L^2 \;\; L^3]^T$

iteration: $i = 1$ **point:** 24, 31, 15, 32, 83, 27, 20, 59

shift $[1 \quad 2 \quad 4 \quad 6]$

basis $\begin{bmatrix} x+73 & 0 & 0 & 0 \\ 17 & 1 & 0 & 0 \\ 2 & 0 & 1 & 0 \\ 63 & 0 & 0 & 1 \end{bmatrix}$

values $\begin{bmatrix} 0 & 7 & 88 & 8 & 59 & 3 & 93 & 35 \\ 0 & 90 & 90 & 52 & 83 & 63 & 11 & 81 \\ 0 & 93 & 93 & 63 & 90 & 81 & 38 & 24 \\ 0 & 13 & 13 & 64 & 51 & 11 & 41 & 16 \end{bmatrix}$

iterative algorithm [van Barel-Bultheel / Beckermann-Labahn / Kötter-Vardy]

parameters: $d = 8$ $m = 4$ $\mathbf{s} = (0, 2, 4, 6)$, base field $\mathbb{F}_{97}$

input: $(24, 31, 15, 32, 83, 27, 20, 59)$ and $\mathbf{F} = [1 \;\; L \;\; L^2 \;\; L^3]^T$

iteration: $i = 2$ **point:** 24, 31, 15, 32, 83, 27, 20, 59

shift $[1 \quad 2 \quad 4 \quad 6]$

basis
$$\begin{bmatrix} x + 73 & 0 & 0 & 0 \\ 17 & 1 & 0 & 0 \\ 2 & 0 & 1 & 0 \\ 63 & 0 & 0 & 1 \end{bmatrix}$$

values
$$\begin{bmatrix} 0 & 7 & 88 & 8 & 59 & 3 & 93 & 35 \\ 0 & 90 & 90 & 52 & 83 & 63 & 11 & 81 \\ 0 & 93 & 93 & 63 & 90 & 81 & 38 & 24 \\ 0 & 13 & 13 & 64 & 51 & 11 & 41 & 16 \end{bmatrix}$$

iterative algorithm [van Barel-Bultheel / Beckermann-Labahn / Kötter-Vardy]

parameters: $d = 8$ $m = 4$ $\mathbf{s} = (0, 2, 4, 6)$, base field $\mathbb{F}_{97}$

input: $(24, 31, 15, 32, 83, 27, 20, 59)$ and $\mathbf{F} = [1 \;\; L \;\; L^2 \;\; L^3]^T$

iteration: $i = 2$ **point:** 24, 31, 15, 32, 83, 27, 20, 59

shift $[1 \;\; 2 \;\; 4 \;\; 6]$

basis
$$\begin{bmatrix} x + 73 & 0 & 0 & 0 \\ x + 90 & 1 & 0 & 0 \\ 56x + 16 & 0 & 1 & 0 \\ 12x + 66 & 0 & 0 & 1 \end{bmatrix}$$

values
$$\begin{bmatrix} 0 & 7 & 88 & 8 & 59 & 3 & 93 & 35 \\ 0 & 0 & 81 & 60 & 45 & 66 & 7 & 19 \\ 0 & 0 & 74 & 26 & 96 & 55 & 8 & 44 \\ 0 & 0 & 2 & 63 & 80 & 47 & 90 & 48 \end{bmatrix}$$

iterative algorithm [van Barel-Bultheel / Beckermann-Labahn / Kötter-Vardy]

parameters: $d = 8$ $m = 4$ $\mathbf{s} = (0, 2, 4, 6)$, base field $\mathbb{F}_{97}$

input: $(24, 31, 15, 32, 83, 27, 20, 59)$ and $\mathbf{F} = [1 \;\; L \;\; L^2 \;\; L^3]^T$

iteration: $i = 2$ **point:** 24, 31, 15, 32, 83, 27, 20, 59

shift $[2 \quad 2 \quad 4 \quad 6]$

basis

$$\begin{bmatrix} x^2 + 42x + 65 & 0 & 0 & 0 \\ x + 90 & 1 & 0 & 0 \\ 56x + 16 & 0 & 1 & 0 \\ 12x + 66 & 0 & 0 & 1 \end{bmatrix}$$

values

$$\begin{bmatrix} 0 & 0 & 47 & 8 & 61 & 85 & 44 & 10 \\ 0 & 0 & 81 & 60 & 45 & 66 & 7 & 19 \\ 0 & 0 & 74 & 26 & 96 & 55 & 8 & 44 \\ 0 & 0 & 2 & 63 & 80 & 47 & 90 & 48 \end{bmatrix}$$

iterative & divide and conquer algorithms

iterative algorithm [van Barel-Bultheel / Beckermann-Labahn / Kötter-Vardy]

parameters: $d = 8$ $m = 4$ $\mathbf{s} = (0, 2, 4, 6)$, base field $\mathbb{F}_{97}$

input: $(24, 31, 15, 32, 83, 27, 20, 59)$ and $\mathbf{F} = [1 \;\; L \;\; L^2 \;\; L^3]^T$

iteration: $i = 3$ **point:** 24, 31, 15, 32, 83, 27, 20, 59

shift $[2 \;\; 2 \;\; 4 \;\; 6]$

basis
$$\begin{bmatrix} x^2 + 42x + 65 & 0 & 0 & 0 \\ x + 90 & 1 & 0 & 0 \\ 56x + 16 & 0 & 1 & 0 \\ 12x + 66 & 0 & 0 & 1 \end{bmatrix}$$

values
$$\begin{bmatrix} 0 & 0 & 47 & 8 & 61 & 85 & 44 & 10 \\ 0 & 0 & 81 & 60 & 45 & 66 & 7 & 19 \\ 0 & 0 & 74 & 26 & 96 & 55 & 8 & 44 \\ 0 & 0 & 2 & 63 & 80 & 47 & 90 & 48 \end{bmatrix}$$

iterative algorithm [van Barel-Bultheel / Beckermann-Labahn / Kötter-Vardy]

parameters: $d = 8$ $m = 4$ $\mathbf{s} = (0, 2, 4, 6)$, base field $\mathbb{F}_{97}$

input: $(24, 31, 15, 32, 83, 27, 20, 59)$ and $\mathbf{F} = [1 \ \ L \ \ L^2 \ \ L^3]^T$

iteration: $i = 3$ **point:** 24, 31, 15, 32, 83, 27, 20, 59

$$\begin{array}{ll}
\textbf{shift} & \qquad [3 \quad 2 \quad 4 \quad 6] \\
\textbf{basis} & \begin{bmatrix} x^3 + 27x^2 + 17x + 92 & 0 & 0 & 0 \\ 54x^2 + 38x + 11 & 1 & 0 & 0 \\ 17x^2 + 91x + 54 & 0 & 1 & 0 \\ 66x^2 + 68x + 88 & 0 & 0 & 1 \end{bmatrix}
\end{array}$$

$$\textbf{values} \quad \begin{bmatrix} 0 & 0 & 0 & 39 & 74 & 50 & 26 & 52 \\ 0 & 0 & 0 & 7 & 41 & 0 & 55 & 74 \\ 0 & 0 & 0 & 65 & 66 & 45 & 77 & 20 \\ 0 & 0 & 0 & 9 & 32 & 31 & 84 & 29 \end{bmatrix}$$

iterative & divide and conquer algorithms

iterative algorithm [van Barel-Bultheel / Beckermann-Labahn / Kötter-Vardy]

parameters: $d = 8$ $m = 4$ $\mathbf{s} = (0, 2, 4, 6)$, base field $\mathbb{F}_{97}$

input: $(24, 31, 15, 32, 83, 27, 20, 59)$ and $\mathbf{F} = [1 \;\; L \;\; L^2 \;\; L^3]^T$

iteration: $i = 4$ **point:** 24, 31, 15, 32, 83, 27, 20, 59

shift $[3 \quad 2 \quad 4 \quad 6]$

basis

$$\begin{bmatrix} x^3 + 27x^2 + 17x + 92 & 0 & 0 & 0 \\ 54x^2 + 38x + 11 & 1 & 0 & 0 \\ 17x^2 + 91x + 54 & 0 & 1 & 0 \\ 66x^2 + 68x + 88 & 0 & 0 & 1 \end{bmatrix}$$

values

$$\begin{bmatrix} 0 & 0 & 0 & 39 & 74 & 50 & 26 & 52 \\ 0 & 0 & 0 & 7 & 41 & 0 & 55 & 74 \\ 0 & 0 & 0 & 65 & 66 & 45 & 77 & 20 \\ 0 & 0 & 0 & 9 & 32 & 31 & 84 & 29 \end{bmatrix}$$

iterative & divide and conquer algorithms

iterative algorithm [van Barel-Bultheel / Beckermann-Labahn / Kötter-Vardy]

parameters: $d = 8$ $\quad m = 4$ $\quad \mathbf{s} = (0, 2, 4, 6)$, $\quad$ base field $\mathbb{F}_{97}$

input: $(24, 31, 15, 32, 83, 27, 20, 59)$ and $\mathbf{F} = [1 \;\; L \;\; L^2 \;\; L^3]^T$

iteration: $i = 4$ $\qquad$ **point:** 24, 31, 15, 32, 83, 27, 20, 59

shift $\quad [3 \quad 3 \quad 4 \quad 6]$

basis

$$\begin{bmatrix} x^3 + 31x^2 + 27x + 3 & 36 & 0 & 0 \\ 54x^3 + 56x^2 + 56x + 36 & x + 65 & 0 & 0 \\ 56x^2 + 43x + 35 & 60 & 1 & 0 \\ 52x^2 + 33x + 60 & 68 & 0 & 1 \end{bmatrix}$$

values

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 95 & 50 & 66 & 0 \\ 0 & 0 & 0 & 0 & 54 & 0 & 19 & 58 \\ 0 & 0 & 0 & 0 & 4 & 45 & 79 & 95 \\ 0 & 0 & 0 & 0 & 7 & 31 & 41 & 17 \end{bmatrix}$$

iterative & divide and conquer algorithms

iterative algorithm [van Barel-Bultheel / Beckermann-Labahn / Kötter-Vardy]

parameters: $d = 8$ $m = 4$ $\mathbf{s} = (0, 2, 4, 6)$, base field $\mathbb{F}_{97}$

input: $(24, 31, 15, 32, 83, 27, 20, 59)$ and $\mathbf{F} = [1 \;\; L \;\; L^2 \;\; L^3]^T$

iteration: $i = 5$ **point:** 24, 31, 15, 32, 83, 27, 20, 59

shift $[4 \quad 3 \quad 4 \quad 6]$

basis
$$\begin{bmatrix} x^4 + 45x^3 + 73x^2 + 90x + 42 & 36x + 19 & 0 & 0 \\ 81x^3 + 20x^2 + 9x + 20 & x + 67 & 0 & 0 \\ 2x^3 + 21x^2 + 41 & 35 & 1 & 0 \\ 52x^3 + 15x^2 + 79x + 22 & 0 & 0 & 1 \end{bmatrix}$$

values
$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 13 & 13 & 0 \\ 0 & 0 & 0 & 0 & 0 & 89 & 55 & 58 \\ 0 & 0 & 0 & 0 & 0 & 48 & 17 & 95 \\ 0 & 0 & 0 & 0 & 0 & 12 & 78 & 17 \end{bmatrix}$$

iterative & divide and conquer algorithms

iterative algorithm [van Barel-Bultheel / Beckermann-Labahn / Kötter-Vardy]

parameters: $d = 8$ $m = 4$ $\mathbf{s} = (0, 2, 4, 6)$, base field $\mathbb{F}_{97}$

input: $(24, 31, 15, 32, 83, 27, 20, 59)$ and $\mathbf{F} = [1 \;\; L \;\; L^2 \;\; L^3]^\mathsf{T}$

iteration: $i = 6$ **point:** $24, 31, 15, 32, 83, 27, 20, 59$

shift $[4 \quad 4 \quad 4 \quad 6]$

basis
$$\begin{bmatrix} x^4 + 19x^3 + 57x^2 + 44x + 26 & 74x + 43 & 0 & 0 \\ 81x^4 + 64x^3 + 51x^2 + 68x + 42 & x^2 + 40x + 34 & 0 & 0 \\ 3x^3 + 44x^2 + 54x + 64 & 6x + 49 & 1 & 0 \\ 28x^3 + 45x^2 + 44x + 52 & 50x + 52 & 0 & 1 \end{bmatrix}$$

values
$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 66 & 70 \\ 0 & 0 & 0 & 0 & 0 & 0 & 3 & 13 \\ 0 & 0 & 0 & 0 & 0 & 0 & 56 & 55 \\ 0 & 0 & 0 & 0 & 0 & 0 & 15 & 7 \end{bmatrix}$$

iterative & divide and conquer algorithms

iterative algorithm [van Barel-Bultheel / Beckermann-Labahn / Kötter-Vardy]

parameters: $d = 8$ $m = 4$ $\mathbf{s} = (0, 2, 4, 6)$, base field $\mathbb{F}_{97}$

input: $(24, 31, 15, 32, 83, 27, 20, 59)$ and $\mathbf{F} = [1 \;\; L \;\; L^2 \;\; L^3]^T$

iteration: $i = 7$ **point:** 24, 31, 15, 32, 83, 27, 20, 59

shift $[5 \quad 4 \quad 4 \quad 6]$

basis
$$\begin{bmatrix} x^5 + 96x^4 + 65x^3 + 68x^2 + 19x + 62 & 74x^2 + 18x + 13 & 0 & 0 \\ 6x^4 + 94x^3 + 44x^2 + 66x + 32 & x^2 + 19x + 10 & 0 & 0 \\ 55x^4 + 78x^3 + 75x^2 + 49x + 39 & 2x + 86 & 1 & 0 \\ 13x^4 + 81x^3 + 10x^2 + 34x + 2 & 42x + 29 & 0 & 1 \end{bmatrix}$$

values
$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 14 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 25 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 44 \end{bmatrix}$$

iterative & divide and conquer algorithms

iterative algorithm [van Barel-Bultheel / Beckermann-Labahn / Kötter-Vardy]

parameters: $d = 8$ $m = 4$ $\mathbf{s} = (0, 2, 4, 6)$, base field $\mathbb{F}_{97}$

input: $(24, 31, 15, 32, 83, 27, 20, 59)$ and $\mathbf{F} = [1 \;\; L \;\; L^2 \;\; L^3]^T$

iteration: $i = 8$ **point:** 24, 31, 15, 32, 83, 27, 20, 59

shift $[5 \quad 5 \quad 4 \quad 6]$

basis
$$\begin{bmatrix} x^5 + 12x^4 + 10x^3 + 34x^2 + 65x + 2 & 60x^2 + 43x + 67 & 0 & 0 \\ 6x^5 + 31x^4 + 27x^3 + 89x^2 + 18x + 52 & x^3 + 57x^2 + 53x + 89 & 0 & 0 \\ 2x^4 + 56x^3 + 42x^2 + 48x + 15 & 72x^2 + 12x + 30 & 1 & 0 \\ 40x^4 + 19x^3 + 14x^2 + 40x + 49 & 53x^2 + 79x + 74 & 0 & 1 \end{bmatrix}$$

values
$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

iterative algorithm [van Barel-Bultheel / Beckermann-Labahn / Kötter-Vardy]

parameters: $d = 8$ $m = 4$ $\mathbf{s} = (0, 2, 4, 6)$, base field $\mathbb{F}_{97}$

input: $(24, 31, 15, 32, 83, 27, 20, 59)$ and $\mathbf{F} = [1 \;\; L \;\; L^2 \;\; L^3]^T$

iteration: $i = 8$ **point:** 24, 31, 15, 32, 83, 27, 20, 59

shift $[5 \quad 5 \quad 4 \quad 6]$

basis

$$\begin{bmatrix} x^5 + 12x^4 + 10x^3 + 34x^2 + 65x + 2 & 60x^2 + 43x + 67 & 0 & 0 \\ 6x^5 + 31x^4 + 27x^3 + 89x^2 + 18x + 52 & x^3 + 57x^2 + 53x + 89 & 0 & 0 \\ 2x^4 + 56x^3 + 42x^2 + 48x + 15 & 72x^2 + 12x + 30 & 1 & 0 \\ 40x^4 + 19x^3 + 14x^2 + 40x + 49 & 53x^2 + 79x + 74 & 0 & 1 \end{bmatrix}$$

$$Q(x, y) = (2x^4 + 56x^3 + 42x^2 + 48x + 15) + (72x^2 + 12x + 30)y + y^2$$

values

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

iterative & divide and conquer algorithms

iterative algorithm: complexity aspects

at step i, $\mathbf{P}$ and $\mathbf{F}$ are left multiplied by $\mathbf{E}_i = \begin{bmatrix} \mathbf{I}_{\pi-1} & \boldsymbol{\lambda_1} & \mathbf{0} \\ \mathbf{0} & x-\alpha & \mathbf{0} \\ \mathbf{0} & \boldsymbol{\lambda_2} & \mathbf{I}_{m-\pi} \end{bmatrix}$
where $\boldsymbol{\lambda_1} \in \mathbb{K}^{(\pi-1)\times 1}$ and $\boldsymbol{\lambda_2} \in \mathbb{K}^{(m-\pi)\times 1}$ are constant

at step i, $\mathbf{P}$ and $\mathbf{F}$ are left multiplied by $\mathbf{E}_i = \begin{bmatrix} \mathbf{I}_{\pi-1} & \boldsymbol{\lambda_1} & \mathbf{0} \\ \mathbf{0} & x-\alpha & \mathbf{0} \\ \mathbf{0} & \boldsymbol{\lambda_2} & \mathbf{I}_{m-\pi} \end{bmatrix}$
where $\boldsymbol{\lambda_1} \in \mathbb{K}^{(\pi-1)\times 1}$ and $\boldsymbol{\lambda_2} \in \mathbb{K}^{(m-\pi)\times 1}$ are constant

complexity $O(m^2d^2)$:

- iteration with d steps
- each step: evaluation of $\mathbf{F}$ + multiplications $\mathbf{E}_i\mathbf{F}$ and $\mathbf{E}_i\mathbf{P}$
- at any stage $\mathbf{P}$ has degree $\leqslant d$ and dimensions $m \times m$
- at any stage $\mathbf{F}$ has degree $< 2d$ and dimensions $m \times 1$

one gets $O(md^2)$ with either:

. normalizing at each step + finer analysis

. "balanced" input shift + finer analysis

iterative & divide and conquer algorithms

iterative algorithm: complexity aspects

at step i, $\mathbf{P}$ and $\mathbf{F}$ are left multiplied by $\mathbf{E}_i = \begin{bmatrix} \mathbf{I}_{\pi-1} & \boldsymbol{\lambda}_1 & \mathbf{0} \\ \mathbf{0} & x-\alpha & \mathbf{0} \\ \mathbf{0} & \boldsymbol{\lambda}_2 & \mathbf{I}_{m-\pi} \end{bmatrix}$
where $\boldsymbol{\lambda}_1 \in \mathbb{K}^{(\pi-1)\times 1}$ and $\boldsymbol{\lambda}_2 \in \mathbb{K}^{(m-\pi)\times 1}$ are constant

complexity $O(m^2 d^2)$**:**

- iteration with d steps
- each step: evaluation of $\mathbf{F}$ + multiplications $\mathbf{E}_i\mathbf{F}$ and $\mathbf{E}_i\mathbf{P}$
- at any stage $\mathbf{P}$ has degree $\leqslant d$ and dimensions $m \times m$
- at any stage $\mathbf{F}$ has degree $< 2d$ and dimensions $m \times 1$

one gets $O(md^2)$ with either:
. normalizing at each step + finer analysis
. "balanced" input shift + finer analysis

correctness:

- the main task is to prove the base case ($d = 1$, single point)
- then, correctness follows from the "basis multiplication theorem"

iterative & divide and conquer algorithms

general multiplication-based approach for relations

algorithms based on polynomial matrix multiplication

[Beckermann-Labahn '94+'97] [Giorgi-Jeannerod-Villard 2003]

- compute a first basis $\mathbf{P}_1$ for a subproblem
- update the input instance to get the second subproblem
- compute a second basis $\mathbf{P}_2$ for this second subproblem
- the output basis of solutions is $\mathbf{P}_2\mathbf{P}_1$

we want $\mathbf{P}_2\mathbf{P}_1$ shifted reduced

$\mathbf{P}_2\mathbf{P}_1$ reduced not implied by "$\mathbf{P}_1$ reduced and $\mathbf{P}_2$ reduced"

iterative & divide and conquer algorithms

general multiplication-based approach for relations

algorithms based on polynomial matrix multiplication
[Beckermann-Labahn '94+'97] [Giorgi-Jeannerod-Villard 2003]

- compute a first basis $\mathbf{P}_1$ for a subproblem
- update the input instance to get the second subproblem
- compute a second basis $\mathbf{P}_2$ for this second subproblem
- the output basis of solutions is $\mathbf{P}_2\mathbf{P}_1$

we want $\mathbf{P}_2\mathbf{P}_1$ shifted reduced

$\mathbf{P}_2\mathbf{P}_1$ reduced not implied by "$\mathbf{P}_1$ reduced and $\mathbf{P}_2$ reduced"

theorem:

($\mathbf{P}_1$ is s-reduced and $\mathbf{P}_2$ is $\mathbf{t}$-reduced") $\Rightarrow$ $\mathbf{P}_2\mathbf{P}_1$ is s-reduced

where $\mathbf{t}$ is a shift trivially computed from s and $\mathbf{P}_1$ $\quad (\mathbf{t} = \mathrm{rdeg}_{s}(\mathbf{P}_1))$

bonus: detailed statement and proof

let $\mathcal{M} \subseteq \mathcal{M}_1$ be two $\mathbb{K}[x]$-submodules of $\mathbb{K}[x]^m$ of rank m,
let $\mathbf{P}_1 \in \mathbb{K}[x]^{m\times m}$ be a basis of $\mathcal{M}_1$,
let $s \in \mathbb{Z}^m$ and $t = \mathrm{rdeg}_s(\mathbf{P}_1)$,

- the rank of the module $\mathcal{M}_2 = \{\boldsymbol{\lambda} \in \mathbb{K}[x]^{1\times m} \mid \boldsymbol{\lambda}\mathbf{P}_1 \in \mathcal{M}\}$ is m and for any basis $\mathbf{P}_2 \in \mathbb{K}[x]^{m\times m}$ of $\mathcal{M}_2$, the product $\mathbf{P}_2\mathbf{P}_1$ is a basis of $\mathcal{M}$
- if $\mathbf{P}_1$ is s-reduced and $\mathbf{P}_2$ is t-reduced, then $\mathbf{P}_2\mathbf{P}_1$ is s-reduced

bonus: detailed statement and proof

let $\mathcal{M} \subseteq \mathcal{M}_1$ be two $\mathbb{K}[x]$-submodules of $\mathbb{K}[x]^m$ of rank m,
let $\mathbf{P}_1 \in \mathbb{K}[x]^{m\times m}$ be a basis of $\mathcal{M}_1$,
let $s \in \mathbb{Z}^m$ and $t = \mathrm{rdeg}_s(\mathbf{P}_1)$,

- the rank of the module $\mathcal{M}_2 = \{\boldsymbol{\lambda} \in \mathbb{K}[x]^{1\times m} \mid \boldsymbol{\lambda}\mathbf{P}_1 \in \mathcal{M}\}$ is m and for any basis $\mathbf{P}_2 \in \mathbb{K}[x]^{m\times m}$ of $\mathcal{M}_2$, the product $\mathbf{P}_2\mathbf{P}_1$ is a basis of $\mathcal{M}$
- if $\mathbf{P}_1$ is s-reduced and $\mathbf{P}_2$ is t-reduced, then $\mathbf{P}_2\mathbf{P}_1$ is s-reduced

Let $\mathbf{A} \in \mathbb{K}[x]^{m\times m}$ denote the adjugate of $\mathbf{P}_1$. Then, we have $\mathbf{A}\mathbf{P}_1 = \det(\mathbf{P}_1)\mathbf{I}_m$. Thus, $\mathbf{p}\mathbf{A}\mathbf{P}_1 = \det(\mathbf{P}_1)\mathbf{p} \in \mathcal{M}$ for all $\mathbf{p} \in \mathcal{M}$, and therefore $\mathcal{M}\mathbf{A} \subseteq \mathcal{M}_2$. Now, the nonsingularity of $\mathbf{A}$ ensures that $\mathcal{M}\mathbf{A}$ has rank m; this implies that $\mathcal{M}_2$ has rank m as well (see e.g. [Dummit-Foote 2004, Sec. 12.1, Thm. 4]). The matrix $\mathbf{P}_2\mathbf{P}_1$ is nonsingular since $\det(\mathbf{P}_2\mathbf{P}_1) \neq 0$. Now let $\mathbf{p} \in \mathcal{M}$; we want to prove that $\mathbf{p}$ is a $\mathbb{K}[x]$-linear combination of the rows of $\mathbf{P}_2\mathbf{P}_1$. First, $\mathbf{p} \in \mathcal{M}_1$, so there exists $\boldsymbol{\lambda} \in \mathbb{K}[x]^{1\times m}$ such that $\mathbf{p} = \boldsymbol{\lambda}\mathbf{P}_1$. But then $\boldsymbol{\lambda} \in \mathcal{M}_2$, and thus there exists $\boldsymbol{\mu} \in \mathbb{K}[x]^{1\times m}$ such that $\boldsymbol{\lambda} = \boldsymbol{\mu}\mathbf{P}_2$. This yields the combination $\mathbf{p} = \boldsymbol{\mu}\mathbf{P}_2\mathbf{P}_1$.

bonus: detailed statement and proof

let $\mathcal{M} \subseteq \mathcal{M}_1$ be two $\mathbb{K}[x]$-submodules of $\mathbb{K}[x]^m$ of rank m,
let $\mathbf{P}_1 \in \mathbb{K}[x]^{m \times m}$ be a basis of $\mathcal{M}_1$,
let $\mathbf{s} \in \mathbb{Z}^m$ and $\mathbf{t} = \mathrm{rdeg}_{\mathbf{s}}(\mathbf{P}_1)$,

- the rank of the module $\mathcal{M}_2 = \{\boldsymbol{\lambda} \in \mathbb{K}[x]^{1 \times m} \mid \boldsymbol{\lambda}\mathbf{P}_1 \in \mathcal{M}\}$ is m and for any basis $\mathbf{P}_2 \in \mathbb{K}[x]^{m \times m}$ of $\mathcal{M}_2$, the product $\mathbf{P}_2\mathbf{P}_1$ is a basis of $\mathcal{M}$
- if $\mathbf{P}_1$ is $\mathbf{s}$-reduced and $\mathbf{P}_2$ is $\mathbf{t}$-reduced, then $\mathbf{P}_2\mathbf{P}_1$ is $\mathbf{s}$-reduced

Let $\mathbf{d} = \mathrm{rdeg}_{\mathbf{t}}(\mathbf{P}_2)$; we have $\mathbf{d} = \mathrm{rdeg}_{\mathbf{s}}(\mathbf{P}_2\mathbf{P}_1)$ by the predictable degree property. Using $\mathbf{X}^{-\mathbf{d}}\mathbf{P}_2\mathbf{P}_1\mathbf{X}^{\mathbf{s}} = \mathbf{X}^{-\mathbf{d}}\mathbf{P}_2\mathbf{X}^{\mathbf{t}}\mathbf{X}^{-\mathbf{t}}\mathbf{P}_1\mathbf{X}^{\mathbf{s}}$, we obtain that $\mathrm{lm}_{\mathbf{s}}(\mathbf{P}_2\mathbf{P}_1) = \mathrm{lm}_{\mathbf{t}}(\mathbf{P}_2)\mathrm{lm}_{\mathbf{s}}(\mathbf{P}_1)$. By assumption, $\mathrm{lm}_{\mathbf{t}}(\mathbf{P}_2)$ and $\mathrm{lm}_{\mathbf{s}}(\mathbf{P}_1)$ are invertible, and therefore $\mathrm{lm}_{\mathbf{s}}(\mathbf{P}_2\mathbf{P}_1)$ is invertible as well; thus $\mathbf{P}_2\mathbf{P}_1$ is $\mathbf{s}$-reduced.

iterative & divide and conquer algorithms

divide and conquer algorithm [Beckermann-Labahn '94+'97]

input: $\mathbf{F}, (\alpha_1, \ldots, \alpha_d), \mathbf{s}$
output: $\mathbf{P}$

- if $d \leqslant \texttt{threshold}$: call iterative algorithm
- else:

  a. $M_1 \leftarrow (x - \alpha_1) \cdots (x - \alpha_{\lfloor d/2 \rfloor})$; $M_2 \leftarrow (x - \alpha_{\lfloor d/2 \rfloor + 1}) \cdots (x - \alpha_d)$

  b. $\mathbf{P}_1 \leftarrow$ recursive call on $\mathbf{F}$ rem $M_1, (\alpha_1, \ldots, \alpha_{\lfloor d/2 \rfloor}), \mathbf{s}$

  c. updated shift: $\mathbf{t} \leftarrow \mathrm{rdeg}_{\mathbf{s}}(\mathbf{P}_1)$

  d. residual equation: $\mathbf{G} \leftarrow \frac{1}{M_1}\mathbf{P}_1\mathbf{F}$

  e. $\mathbf{P}_2 \leftarrow$ recursive call on $\mathbf{G}$ rem $M_2, (\alpha_{\lfloor d/2 \rfloor + 1}, \ldots, \alpha_d), \mathbf{t}$

  f. return the product $\mathbf{P}_2\mathbf{P}_1$

iterative & divide and conquer algorithms

divide and conquer algorithm [Beckermann-Labahn '94+'97]

input: $\mathbf{F}, (\alpha_1, \ldots, \alpha_d), \mathbf{s}$
output: $\mathbf{P}$

- if $d \leqslant \texttt{threshold}$: call iterative algorithm
- else:

  a. $M_1 \leftarrow (x - \alpha_1) \cdots (x - \alpha_{\lfloor d/2 \rfloor})$; $M_2 \leftarrow (x - \alpha_{\lfloor d/2 \rfloor + 1}) \cdots (x - \alpha_d)$

  b. $\mathbf{P}_1 \leftarrow$ recursive call on $\mathbf{F}$ rem $M_1, (\alpha_1, \ldots, \alpha_{\lfloor d/2 \rfloor}), \mathbf{s}$

  c. updated shift: $\mathbf{t} \leftarrow \mathrm{rdeg}_{\mathbf{s}}(\mathbf{P}_1)$

  d. residual equation: $\mathbf{G} \leftarrow \frac{1}{M_1} \mathbf{P}_1 \mathbf{F}$

  e. $\mathbf{P}_2 \leftarrow$ recursive call on $\mathbf{G}$ rem $M_2, (\alpha_{\lfloor d/2 \rfloor + 1}, \ldots, \alpha_d), \mathbf{t}$

  f. return the product $\mathbf{P}_2 \mathbf{P}_1$

correctness:

- correctness of base case
- then, direct consequence of the "basis multiplication theorem"
- residual: $\{\mathbf{p} \mid \mathbf{p}\mathbf{P}_1\mathbf{F} = 0 \bmod M\} = \{\mathbf{p} \mid \mathbf{p}(\frac{1}{M_1}\mathbf{P}_1\mathbf{F}) = 0 \bmod M_2\}$

iterative & divide and conquer algorithms

divide and conquer algorithm [Beckermann-Labahn '94+'97]

input: $\mathbf{F}, (\alpha_1, \ldots, \alpha_d), \mathbf{s}$
output: $\mathbf{P}$

- if $d \leqslant \texttt{threshold}$: call iterative algorithm
- else:

  a. $M_1 \leftarrow (x - \alpha_1) \cdots (x - \alpha_{\lfloor d/2 \rfloor})$; $M_2 \leftarrow (x - \alpha_{\lfloor d/2 \rfloor + 1}) \cdots (x - \alpha_d)$

  b. $\mathbf{P}_1 \leftarrow$ recursive call on $\mathbf{F}$ rem $M_1, (\alpha_1, \ldots, \alpha_{\lfloor d/2 \rfloor}), \mathbf{s}$

  c. updated shift: $\mathbf{t} \leftarrow \mathrm{rdeg}_{\mathbf{s}}(\mathbf{P}_1)$

  d. residual equation: $\mathbf{G} \leftarrow \frac{1}{M_1}\mathbf{P}_1\mathbf{F}$

  e. $\mathbf{P}_2 \leftarrow$ recursive call on $\mathbf{G}$ rem $M_2, (\alpha_{\lfloor d/2 \rfloor + 1}, \ldots, \alpha_d), \mathbf{t}$

  f. return the product $\mathbf{P}_2\mathbf{P}_1$

complexity $O(m^\omega M(d) \log(d))$**:**

- if $\omega = 2$, quasi-linear in worst-case output size (yet: s-Popov basis is smaller)
- most expensive step in the recursion is the product $\mathbf{P}_2\mathbf{P}_1$
- equation $\mathcal{C}(m, d) = \mathcal{C}(m, \lfloor d/2 \rfloor) + \mathcal{C}(m, \lceil d/2 \rceil) + O(m^\omega M(d))$

divide and conquer: complexity aspects

input: $\deg(\mathbf{F}) < d$ output: $\deg(\mathbf{P}) \leqslant d$

complexity of each step:

- residual $\mathbf{F} \leftarrow \frac{1}{M_1}\mathbf{P}_1\mathbf{F}$ $O(m^2 M(d))$
- $\mathbf{F}$ rem M_1 and $\mathbf{F}$ rem M_2 $O(m M(d))$
- product $\mathbf{P}_2\mathbf{P}_1$ $O(m^\omega M(d))$
- two recursive calls $2\mathcal{C}(m, \lfloor d/2 \rceil)$

iterative & divide and conquer algorithms

divide and conquer: complexity aspects

input: $\deg(\mathbf{F}) < d$ output: $\deg(\mathbf{P}) \leqslant d$

complexity of each step:

- residual $\mathbf{F} \leftarrow \frac{1}{M_1}\mathbf{P}_1\mathbf{F}$ — $O(m^2 M(d))$
- $\mathbf{F}$ rem M_1 and $\mathbf{F}$ rem M_2 — $O(m M(d))$
- product $\mathbf{P}_2\mathbf{P}_1$ — $O(m^\omega M(d))$
- two recursive calls — $2\mathcal{C}(m, \lfloor d/2 \rceil)$

$$\begin{cases} \mathcal{C}(m,d) = \mathcal{C}(m, \lfloor d/2 \rfloor) + \mathcal{C}(m, \lceil d/2 \rceil) + O(m^\omega M(d)) \\ d \text{ base cases, each one costs } O(m) \end{cases}$$

$$\Rightarrow \quad O(m^\omega M(d)\log(d))$$

unrolling: $m^\omega \left(M(d) + 2M(\frac{d}{2}) + 4M(\frac{d}{4}) + \cdots + \frac{d}{2}M(2)\right) + dm$

divide and conquer: complexity aspects

input: $\deg(\mathbf{F}) < d$ output: $\deg(\mathbf{P}) \leqslant d$

complexity of each step:

- residual $\mathbf{F} \leftarrow \frac{1}{M_1}\mathbf{P}_1\mathbf{F}$ $O(m^2 M(d))$
- $\mathbf{F}$ rem M_1 and $\mathbf{F}$ rem M_2 $O(mM(d))$
- product $\mathbf{P}_2\mathbf{P}_1$ $O(m^\omega M(d))$
- two recursive calls $2\mathcal{C}(m, \lfloor d/2 \rceil)$

output: $\deg(\mathbf{P}) \approx \lceil \frac{d}{m} \rceil$

$s = 0$ and generic $\mathbf{F}$:

$O(m^\omega M(\lceil \frac{d}{m} \rceil))$

unchanged

$O(m^\omega M(\lceil \frac{d}{m} \rceil))$

unchanged

- partial linearization

$$\begin{cases} \mathcal{C}(m, d) = \mathcal{C}(m, \lfloor d/2 \rfloor) + \mathcal{C}(m, \lceil d/2 \rceil) + O(m^\omega M(d)) \\ d \text{ base cases, each one costs } O(m) \end{cases}$$

$$\Rightarrow \quad O(m^\omega M(d) \log(d))$$

divide and conquer: complexity aspects

input: $\deg(\mathbf{F}) < d$ output: $\deg(\mathbf{P}) \leqslant d$

complexity of each step:

- residual $\mathbf{F} \leftarrow \frac{1}{M_1}\mathbf{P}_1\mathbf{F}$ — $O(m^2 \mathsf{M}(d))$
- $\mathbf{F}$ rem M_1 and $\mathbf{F}$ rem M_2 — $O(m\mathsf{M}(d))$
- product $\mathbf{P}_2\mathbf{P}_1$ — $O(m^\omega \mathsf{M}(d))$
- two recursive calls — $2\mathcal{C}(m, \lfloor d/2 \rceil)$

output: $\deg(\mathbf{P}) \approx \lceil \frac{d}{m} \rceil$

$s = 0$ and generic $\mathbf{F}$:

$O(m^\omega \mathsf{M}(\lceil \frac{d}{m} \rceil))$

unchanged

$O(m^\omega \mathsf{M}(\lceil \frac{d}{m} \rceil))$

unchanged

- partial linearization
- base case for $d \approx m$, costs $O(m^\omega)$

$$\begin{cases} \mathcal{C}(m, d) = \mathcal{C}(m, \lfloor d/2 \rfloor) + \mathcal{C}(m, \lceil d/2 \rceil) + O(m^\omega \mathsf{M}(d)) \\ d \text{ base cases, each one costs } O(m) \end{cases}$$

$$\Rightarrow \quad O(m^\omega \mathsf{M}(d)\log(d)) \qquad O(m^\omega \mathsf{M}(\lceil \tfrac{d}{m} \rceil)\log(\lceil \tfrac{d}{m} \rceil))$$

iterative & divide and conquer algorithms

divide and conquer: complexity aspects

input: $\deg(\mathbf{F}) < d$ output: $\deg(\mathbf{P}) \leqslant d$

complexity of each step:

- residual $\mathbf{F} \leftarrow \frac{1}{M_1}\mathbf{P}_1\mathbf{F}$ $O(m^2 M(d))$
- $\mathbf{F}$ rem M_1 and $\mathbf{F}$ rem M_2 $O(mM(d))$
- product $\mathbf{P}_2\mathbf{P}_1$ $O(m^\omega M(d))$
- two recursive calls $2\mathcal{C}(m, \lfloor d/2 \rceil)$

output: $\deg(\mathbf{P}) \approx \lceil \frac{d}{m} \rceil$

$s = 0$ and generic $\mathbf{F}$:

$O(m^\omega M(\lceil \frac{d}{m} \rceil))$

unchanged

$O(m^\omega M(\lceil \frac{d}{m} \rceil))$

unchanged

- partial linearization
- base case for $d \approx m$, costs $O(m^\omega)$

$$\begin{cases} \mathcal{C}(m, d) = \mathcal{C}(m, \lfloor d/2 \rfloor) + \mathcal{C}(m, \lceil d/2 \rceil) + O(m^\omega M(d)) \\ d \text{ base cases, each one costs } O(m) \end{cases}$$

$$\Rightarrow \quad O(m^\omega M(d) \log(d)) \qquad O(m^\omega M(\lceil \tfrac{d}{m} \rceil) \log(\lceil \tfrac{d}{m} \rceil))$$

m	n	d	PM-Basis	PM-Basis with linearization
4	1	65536	1.6693	**1.26891**
16	1	16384	1.8535	**0.89652**
64	1	2048	2.2865	**0.14362**
256	1	1024	36.620	**0.20660**

overview of the state of the art:

- recursive algorithm: from [Beckermann-Labahn 1994] (for Hermite-Padé) it also works for $\mathbf{F} \in \mathbb{K}[x]^{m\times n}$ with $n > 1$

- [Giorgi-Jeannerod-Villard 2003] achieved $O(m^\omega M(d)\log(d))$ for $\mathbf{F}$ mod x^d, with $n \geqslant 1$ and $n \in O(m)$

- for $s = \mathbf{0}$ and generic $\mathbf{F}$: $O\tilde{~}(m^\omega \lceil \frac{nd}{m} \rceil)$ [Lecerf, ca 2001, unpublished]

overview of the state of the art:

- recursive algorithm: from [Beckermann-Labahn 1994] (for Hermite-Padé)
  it also works for $\mathbf{F} \in \mathbb{K}[x]^{m \times n}$ with $n > 1$
- [Giorgi-Jeannerod-Villard 2003] achieved $O(m^\omega \mathsf{M}(d) \log(d))$
for $\mathbf{F}$ mod x^d, with $n \geqslant 1$ and $n \in O(m)$
- for $\mathbf{s} = \mathbf{0}$ and generic $\mathbf{F}$: $\tilde{O}(m^\omega \lceil \frac{nd}{m} \rceil)$ [Lecerf, ca 2001, unpublished]
- more recently: $\tilde{O}(m^{\omega-1} nd)$ for $\mathbf{F}$ mod x^d
[Storjohann 2006] [Zhou-Labahn 2012] [Jeannerod-Neiger-Villard 2020]
$\rightsquigarrow$ any $\mathbf{s}$, no genericity assumption, returns the canonical $\mathbf{s}$-Popov basis

overview of the state of the art:

- recursive algorithm: from [Beckermann-Labahn 1994] (for Hermite-Padé)
  it also works for $\mathbf{F} \in \mathbb{K}[x]^{m\times n}$ with $n > 1$

- [Giorgi-Jeannerod-Villard 2003] achieved $O(m^\omega \mathsf{M}(d)\log(d))$
for $\mathbf{F}$ mod x^d, with $n \geqslant 1$ and $n \in O(m)$

- for $s = \mathbf{0}$ and generic $\mathbf{F}$: $O\tilde{~}(m^\omega\lceil\frac{nd}{m}\rceil)$ [Lecerf, ca 2001, unpublished]

- more recently: $O\tilde{~}(m^{\omega-1}nd)$ for $\mathbf{F}$ mod x^d
[Storjohann 2006] [Zhou-Labahn 2012] [Jeannerod-Neiger-Villard 2020]
$\rightsquigarrow$ any s, no genericity assumption, returns the canonical s-Popov basis

- $\mathbf{F}$ mod M and general modular matrix equations in similar complexity
[Beckermann-Labahn 1997] [Jeannerod-Neiger-Schost-Villard 2017]
[Neiger-Vu 2017] [Rosenkilde-Storjohann 2021]
$\rightsquigarrow$ any s, no genericity assumption, returns the canonical s-Popov basis

polynomial matrices: two open questions

deterministic Smith form

$$\begin{bmatrix} & & \\ & \mathbf{A} & \\ & & \end{bmatrix} \longrightarrow \begin{bmatrix} s_1 & & & \\ & s_2 & & \\ & & \ddots & \\ & & & s_m \end{bmatrix}$$

s_{i+1} divides s_i

- complexity $O\tilde{~}(m^\omega \frac{D}{m})$ [Storjohann'03]
- Las Vegas randomized algorithm
- requires large field $\mathbb{K}$

deterministic algo in $O\tilde{~}(m^\omega \frac{D}{m})$?

polynomial matrices: two open questions

deterministic Smith form

$$\begin{bmatrix} & & \\ & \mathbf{A} & \\ & & \end{bmatrix} \longrightarrow \begin{bmatrix} s_1 & & & \\ & s_2 & & \\ & & \ddots & \\ & & & s_m \end{bmatrix}$$

s_{i+1} divides s_i

- complexity $O\tilde{~}(m^\omega \frac{D}{m})$ [Storjohann'03]
- Las Vegas randomized algorithm
- requires large field $\mathbb{K}$

deterministic algo in $O\tilde{~}(m^\omega \frac{D}{m})$?

algebraic interpolants

$\rightsquigarrow$ recurrence guessing, modular composition, bivariate interpo

$$\begin{array}{c} p_1 f_1 + p_2 f_2 + \cdots + p_m f_m = 0 \bmod M \\ \Big\downarrow \text{structured } f_i\text{'s} \\ p_1 1 + p_2 L + \cdots + p_m L^{m-1} = 0 \bmod M \end{array}$$

- most algorithms ignore the structure
- recent progress [Villard 2018]
- restrictive: genericity, specific m & d

how to leverage this structure?

outline

- **approximate/interpolate**
  - introduction, links with structured matrices
  - vector interpolation & matrix normal forms
  - iterative & divide and conquer algorithms
- **characteristic polynomial**
- **modular composition**
- **change of order**

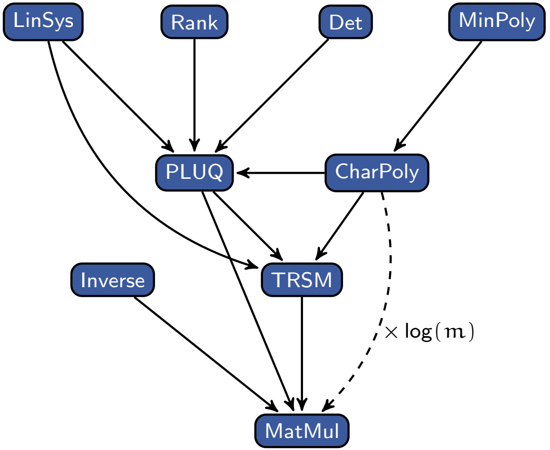
outline

- **approximate/interpolate**
  - introduction, links with structured matrices
  - vector interpolation & matrix normal forms
  - iterative & divide and conquer algorithms
- **characteristic polynomial**
  - previous work and log factors to remove
  - result: "asymptotically optimal" algorithm
  - new triangularization-based approach
- **modular composition**
- **change of order**

characteristic polynomial of a matrix

given $\mathbf{M} \in \mathbb{K}^{m \times m}$, compute $\det(x\mathbf{I}_m - \mathbf{M}) \in \mathbb{K}[x]$

$\mathbb{K}$-linear algebra: reductions of most problems to matrix multiplication

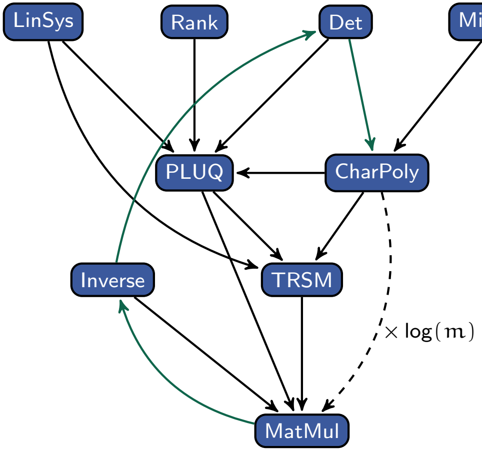


$$\left.\begin{array}{c}\text{LinSys}\\\text{Det}\\\text{Rank}\\\text{PLUQ}\\\text{TRSM}\\\text{Inverse}\end{array}\right\} = O(\text{MatMul})$$

characteristic polynomial of a matrix

given $\mathbf{M} \in \mathbb{K}^{m \times m}$, compute $\det(x\mathbf{I}_m - \mathbf{M}) \in \mathbb{K}[x]$

$\mathbb{K}$-linear algebra: reductions of most problems to matrix multiplication



$$\left.\begin{array}{c}\text{LinSys}\\ \text{Det}\\ \text{Rank}\\ \text{PLUQ}\\ \text{TRSM}\\ \text{Inverse}\end{array}\right\} = O(\text{MatMul})$$

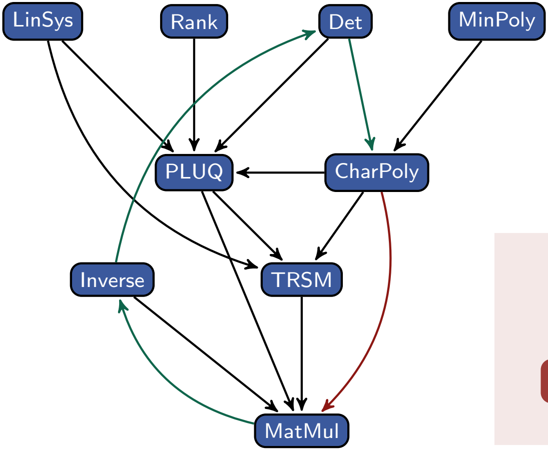
$\text{MatMul} = O(\text{CharPoly})$

[Baur-Strassen 1983]

characteristic polynomial of a matrix

given $\mathbf{M} \in \mathbb{K}^{m \times m}$, compute $\det(x\mathbf{I}_m - \mathbf{M}) \in \mathbb{K}[x]$

$\mathbb{K}$-linear algebra: reductions of most problems to matrix multiplication



$$\left.\begin{array}{c}\text{LinSys}\\ \text{Det}\\ \text{Rank}\\ \text{PLUQ}\\ \text{TRSM}\\ \text{Inverse}\end{array}\right\} = O(\text{MatMul})$$

$\text{MatMul} = O(\text{CharPoly})$

[Baur-Strassen 1983]

$\text{CharPoly} = O(\text{MatMul})$?

charpoly via $\mathbb{K}$-linear algebra

charpoly via $\mathbb{K}$-linear algebra

traces of powers $O(m^4)$ or $O(m^{\omega+1})$

- [LeVerrier 1840] [Faddeev'49, Souriau'48, ...]
- used by [Csanky'75] to prove CharPoly $\in \mathcal{NC}^2$

charpoly via $\mathbb{K}$-linear algebra

traces of powers $O(m^4)$ or $O(m^{\omega+1})$

- [LeVerrier 1840] [Faddeev'49, Souriau'48, ...]
- used by [Csanky'75] to prove CharPoly $\in \mathcal{NC}^2$

determinant expansion $O(m^4)$

- [Samuelson'42, Berkowitz'84]
- suited to division free algorithms
  [Abdlejaoued-Malaschonok'01, Kaltofen-Villard'05]

charpoly via $\mathbb{K}$-linear algebra

traces of powers $O(m^4)$ or $O(m^{\omega+1})$

- [LeVerrier 1840] [Faddeev'49, Souriau'48, ...]
- used by [Csanky'75] to prove CharPoly $\in \mathcal{NC}^2$

determinant expansion $O(m^4)$

- [Samuelson'42, Berkowitz'84]
- suited to division free algorithms
  [Abdlejaoued-Malaschonok'01, Kaltofen-Villard'05]

Krylov methods [Danilevskij'37, Keller-Gehrig'85, P.-Storjohann'07]

- deterministic $O(m^3)$ or $O(m^\omega \log(m))$
- generic $O(m^\omega)$
- Las Vegas randomized, requires large field $O(m^\omega)$

i.e. $\mathrm{card}(\mathbb{K}) \geqslant 2m^2$

charpoly via polynomial matrices

determinant of matrix $\mathbf{A} \in \mathbb{K}[x]^{m \times m}$

charpoly via polynomial matrices

determinant of matrix $\mathbf{A} \in \mathbb{K}[x]^{m \times m}$

evaluation-interpolation [folklore] $O(m^{\omega+1})$

at $\sim md$ points, requires large field

costs: for $\mathbf{A}$ of degree $d = 1$

charpoly via polynomial matrices

determinant of matrix $\mathbf{A} \in \mathbb{K}[x]^{m \times m}$

evaluation-interpolation [folklore] $O(m^{\omega+1})$

at $\sim md$ points, requires large field

costs: for $\mathbf{A}$ of degree $d = 1$

diagonalization [Storjohann 2003] $O(m^{\omega} \log(m)^2)$

Smith form: Las Vegas randomized, requires large field

charpoly via polynomial matrices

determinant of matrix $\mathbf{A} \in \mathbb{K}[x]^{m \times m}$

evaluation-interpolation [folklore] $O(m^{\omega+1})$

at $\sim md$ points, requires large field

costs: for $\mathbf{A}$ of degree $d = 1$

diagonalization [Storjohann 2003] $O(m^{\omega} \log(m)^2)$

Smith form: Las Vegas randomized, requires large field

partial triangularization

- iterative [Mulders-Storjohann 2003] $O(m^3)$
  via weak Popov form computations
- divide and conquer, generic [Giorgi-Jeannerod-Villard 2003] $O(m^{\omega})$
  diagonal of Hermite form must be $1, \ldots, 1, \det(\mathbf{A})$
- divide and conquer [N.-Labahn-Zhou 2017] $\tilde{O}(m^{\omega})$
  logarithmic factors in m and d

- divide and conquer with half-dimension blocks $\rightarrow$ no $\log(m)$
- iterative approaches in m steps $\rightarrow$ sometimes no $\log(m)$ [Pernet-Storjohann'07]
- multi-vector Krylov iterates: $\mathrm{CRP}(V \;\; \mathbf{M}V \;\; \cdots \;\; \mathbf{M}^m V) \rightarrow \log(m)$

in $\mathbb{K}$-linear algebra

sources of log factors

for polynomial matrices

- divide and conquer with half-dimension blocks $\rightarrow$ no $\log(m)$
- iterative approaches in m steps $\rightarrow$ sometimes no $\log(m)$ [Pernet-Storjohann'07]
- multi-vector Krylov iterates: $\mathrm{CRP}(V \;\; \mathbf{M}V \;\; \cdots \;\; \mathbf{M}^m V) \rightarrow \log(m)$

in $\mathbb{K}$-linear algebra

sources of log factors

for polynomial matrices

- divide and conquer with half-dimension blocks $\rightarrow$ no $\log(m)$

  provided degrees are controlled, e.g. kernel basis [Zhou-Labahn-Storjohann'12]

- divide and conquer on degree $\rightarrow$ $\log(d)$ but no $\log(m)$

  e.g. $\mathbb{K}[x]$-MatMul and approximant basis [Giorgi-Jeannerod-Villard'03]

- multi-vector Krylov iterates e.g. [Jeannerod-N.-Schost-Villard'17]

  since base cases of recursions on degree $=$ matrices over $\mathbb{K}$

  typically adds $O(m^\omega d \log(m))$ to the cost, non-negligible when $d = O(1)$

- looking for a matrix with unpredictable, unbalanced degrees

  $\log(m)$ steps in dimension $m \times m$, to uncover the degree profile [Zhou-Labahn'13]

  reminiscent of obstacles in the derandomization of [Pernet-Storjohann'07]

[Vincent Neiger & Clément Pernet, 2021]

deterministic algorithm with complexity $O(m^\omega)$

- polynomial matrices
- partial triangularization
- ternary divide and conquer
- exploiting degree knowledge

characteristic polynomial in the time of matrix multiplication

[Vincent Neiger & Clément Pernet, 2021]
deterministic algorithm with complexity $O(m^\omega)$

- polynomial matrices
- partial triangularization
- ternary divide and conquer
- exploiting degree knowledge

characteristic polynomial in the time of matrix multiplication

framework for complexity — clarification is needed!

For any MatMul exponent ω feasible (as of today),
there is a MatMul algorithm in $O(m^{\omega-\varepsilon})$ for some $\varepsilon > 0$
$\Rightarrow$ the CharPoly algorithm of [Keller-Gehrig'85] is

- deterministic
- in $O(m^{\omega-\varepsilon}\log(m)) \subset O(m^\omega)$

not entirely satisfactory. . .

[Vincent Neiger & Clément Pernet, 2021]

deterministic algorithm with complexity $O(m^\omega)$

- polynomial matrices
- partial triangularization
- ternary divide and conquer
- exploiting degree knowledge

characteristic polynomial in the time of matrix multiplication

framework for complexity — classical requirements

matrix multiplication in $\mathbb{K}^{m\times m}$

- choose a MatMul algorithm in $O(m^\omega)$
- use this one for all MatMul instances

our requirement: $2 < \omega \leqslant 3$

we gladly accept $\omega = 2.1$, please provide the algorithm

requirement: matrices in $\mathbb{K}[x]^{m\times m}_{\leqslant d}$ multiplied in $O(m^\omega \mathsf{M}(d))$

polynomial multiplication in $\mathbb{K}[x]$

- choose a PolMul algorithm in $O(\mathsf{M}(d))$
- use this one for all PolMul instances

our requirement: $\mathsf{M}(d)$ is superlinear and submultiplicative and reasonably good

$2\mathsf{M}(d) \leqslant \mathsf{M}(2d)$ $\quad$ $\mathsf{M}(d_1 d_2) \leqslant \mathsf{M}(d_1)\mathsf{M}(d_2)$

$\mathsf{M}(d) \in O(d^{\omega-1-\varepsilon})$ for some $\varepsilon > 0$

ternary recursion & complexity analysis

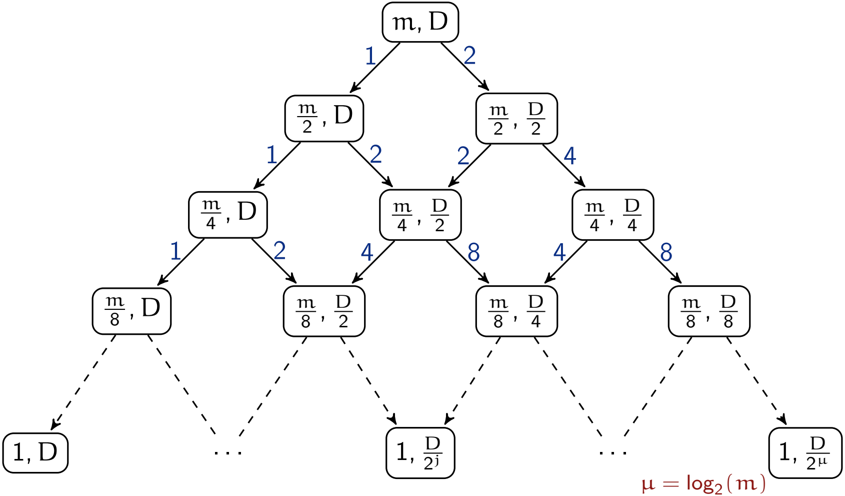
determinant of $\mathbf{A} \in \mathbb{K}[x]^{m \times m}$ of average row degree $\frac{D}{m} = \frac{\text{degdet}}{m}$

$$\mathcal{C}(m, D) \leqslant 2\mathcal{C}(\tfrac{m}{2}, \tfrac{D}{2}) + \mathcal{C}(\tfrac{m}{2}, D) + O(m^{\omega} \mathsf{M}(\tfrac{D}{m}) \log(\tfrac{D}{m}))$$

ternary recursion & complexity analysis

determinant of $\mathbf{A} \in \mathbb{K}[x]^{m \times m}$ of average row degree $\frac{D}{m} = \frac{\mathsf{degdet}}{m}$

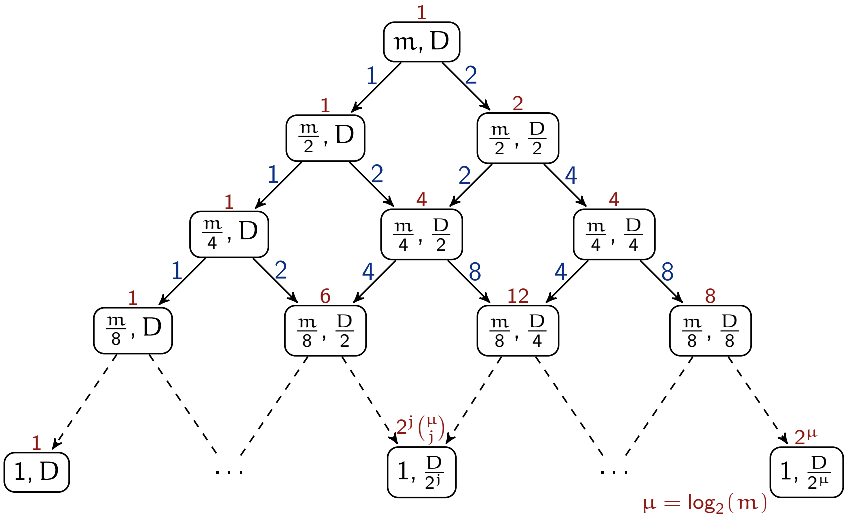
$$\mathcal{C}(m, D) \leqslant 2\mathcal{C}(\tfrac{m}{2}, \tfrac{D}{2}) + \mathcal{C}(\tfrac{m}{2}, D) + O(m^{\omega} \mathsf{M}(\tfrac{D}{m}) \log(\tfrac{D}{m}))$$



ternary recursion & complexity analysis

determinant of $\mathbf{A} \in \mathbb{K}[x]^{m \times m}$ of average row degree $\frac{D}{m} = \frac{\mathsf{degdet}}{m}$

$$\mathcal{C}(m, D) \leqslant 2\mathcal{C}(\tfrac{m}{2}, \tfrac{D}{2}) + \mathcal{C}(\tfrac{m}{2}, D) + O(m^\omega \mathsf{M}(\tfrac{D}{m}) \log(\tfrac{D}{m}))$$

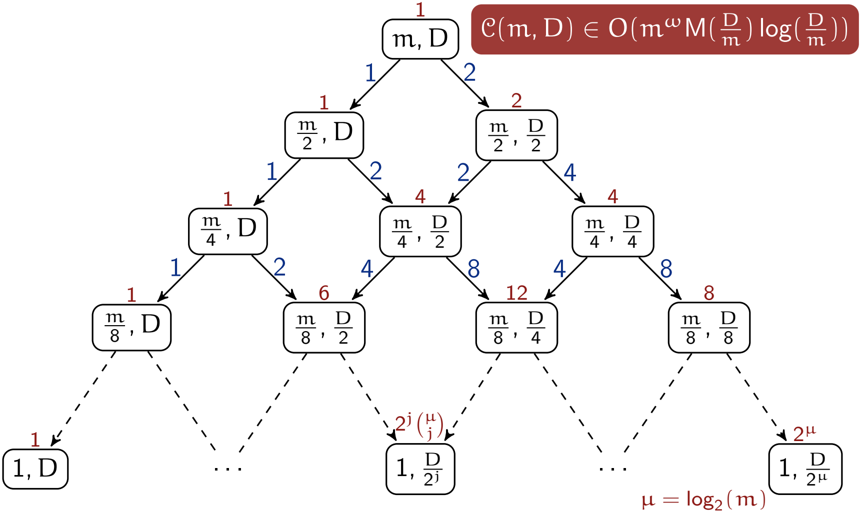


ternary recursion & complexity analysis

determinant of $\mathbf{A} \in \mathbb{K}[x]^{m \times m}$ of average row degree $\frac{D}{m} = \frac{\text{degdet}}{m}$

$$\mathcal{C}(m, D) \leqslant 2\mathcal{C}(\tfrac{m}{2}, \tfrac{D}{2}) + \mathcal{C}(\tfrac{m}{2}, D) + O(m^\omega \mathsf{M}(\tfrac{D}{m}) \log(\tfrac{D}{m}))$$

$$\mathcal{C}(m, D) \in O(m^\omega \mathsf{M}(\tfrac{D}{m}) \log(\tfrac{D}{m}))$$



partial block triangularization

[Mulders-Storjohann 2003, Giorgi-Jeannerod-Villard 2003, Zhou 2012, N.-Labahn-Zhou 2017]

triangularization of $m \times m$ matrix $\mathbf{A}$ using $\frac{m}{2} \times \frac{m}{2}$ blocks

not computed

$$\begin{bmatrix} * & * \\ \mathbf{K}_1 & \mathbf{K}_2 \end{bmatrix} \begin{bmatrix} \mathbf{A}_1 & \mathbf{A}_2 \\ \mathbf{A}_3 & \mathbf{A}_4 \end{bmatrix} = \begin{bmatrix} \mathbf{R} & * \\ \mathbf{0} & \mathbf{B} \end{bmatrix}$$

kernel basis of $\left[\begin{smallmatrix} \mathbf{A}_1 \\ \mathbf{A}_3 \end{smallmatrix}\right]$

$\mathbf{K}_1\mathbf{A}_2 + \mathbf{K}_2\mathbf{A}_4$

row basis of $\left[\begin{smallmatrix} \mathbf{A}_1 \\ \mathbf{A}_3 \end{smallmatrix}\right]$

property: $\det(\mathbf{A}) = \det(\mathbf{R})\det(\mathbf{B})$

partial block triangularization

[Mulders-Storjohann 2003, Giorgi-Jeannerod-Villard 2003, Zhou 2012, N.-Labahn-Zhou 2017]

triangularization of $m \times m$ matrix $\mathbf{A}$ using $\frac{m}{2} \times \frac{m}{2}$ blocks

not computed

$$\begin{bmatrix} * & * \\ \mathbf{K}_1 & \mathbf{K}_2 \end{bmatrix} \begin{bmatrix} \mathbf{A}_1 & \mathbf{A}_2 \\ \mathbf{A}_3 & \mathbf{A}_4 \end{bmatrix} = \begin{bmatrix} \mathbf{R} & * \\ \mathbf{0} & \mathbf{B} \end{bmatrix}$$

kernel basis of $\left[\begin{smallmatrix} \mathbf{A}_1 \\ \mathbf{A}_3 \end{smallmatrix}\right]$ $\quad$ $\mathbf{K}_1\mathbf{A}_2 + \mathbf{K}_2\mathbf{A}_4$ $\quad$ row basis of $\left[\begin{smallmatrix} \mathbf{A}_1 \\ \mathbf{A}_3 \end{smallmatrix}\right]$

property: $\det(\mathbf{A}) = \det(\mathbf{R})\det(\mathbf{B})$

generic input $\Rightarrow$ $\det(\mathbf{A})$ without $\log(m)$ [Giorgi-Jeannerod-Villard'03]

$\mathbf{A}_1$ and $\mathbf{A}_3$ are coprime $\Rightarrow \mathbf{R} = \mathbf{I}_{m/2} \Rightarrow \det(\mathbf{A}) = \det(\mathbf{B})$

- compute kernel $[\mathbf{K}_1 \;\; \mathbf{K}_2]$; deduce $\mathbf{B}$ by MatMul $\quad O(m^\omega M(d)\log(d))$
- recursively, compute $\det(\mathbf{B})$, return it

$\mathbf{A}$ and $[\mathbf{K}_1 \;\; \mathbf{K}_2]$ have degree $d \Rightarrow \mathbf{B}$ has degree $2d$: controlled total degree

complexity $\mathcal{C}(m,d) = \mathcal{C}(\frac{m}{2}, 2d) + O(m^\omega M(d)\log(d))$

partial block triangularization

[Mulders-Storjohann 2003, Giorgi-Jeannerod-Villard 2003, Zhou 2012, N.-Labahn-Zhou 2017]

triangularization of $m \times m$ matrix $\mathbf{A}$ using $\frac{m}{2} \times \frac{m}{2}$ blocks

not computed

$$\begin{bmatrix} * & * \\ \mathbf{K}_1 & \mathbf{K}_2 \end{bmatrix} \begin{bmatrix} \mathbf{A}_1 & \mathbf{A}_2 \\ \mathbf{A}_3 & \mathbf{A}_4 \end{bmatrix} = \begin{bmatrix} \mathbf{R} & * \\ \mathbf{0} & \mathbf{B} \end{bmatrix}$$

kernel basis of $\left[\begin{smallmatrix} \mathbf{A}_1 \\ \mathbf{A}_3 \end{smallmatrix}\right]$ $\quad$ $\mathbf{K}_1\mathbf{A}_2 + \mathbf{K}_2\mathbf{A}_4$ $\quad$ row basis of $\left[\begin{smallmatrix} \mathbf{A}_1 \\ \mathbf{A}_3 \end{smallmatrix}\right]$

property: $\det(\mathbf{A}) = \det(\mathbf{R})\det(\mathbf{B})$

general input $\Rightarrow$ $\det(\mathbf{A})$ with $\log(m)$ [Labahn-N.-Zhou'17]

matrix degree not controlled: degree of $\mathbf{B}$ up to $D = |\mathrm{rdeg}(\mathbf{A})| \leqslant md$
but controlled average row degree: at most $\frac{D}{m}$

- compute kernel $[\mathbf{K}_1 \;\; \mathbf{K}_2]$; deduce $\mathbf{B}$ by MatMul $\quad O\tilde{}(m^\omega \frac{D}{m})$
- compute row basis $\mathbf{R}$ $\quad O\tilde{}(m^\omega \frac{D}{m})$ with $\log(m)$
- recursively, compute $\det(\mathbf{R})$ and $\det(\mathbf{B})$, return $\det(\mathbf{R})\det(\mathbf{B})$

partial block triangularization

[Mulders-Storjohann 2003, Giorgi-Jeannerod-Villard 2003, Zhou 2012, N.-Labahn-Zhou 2017]

triangularization of $m \times m$ matrix $\mathbf{A}$ using $\frac{m}{2} \times \frac{m}{2}$ blocks

not computed

$$\begin{bmatrix} * & * \\ \mathbf{K}_1 & \mathbf{K}_2 \end{bmatrix} \begin{bmatrix} \mathbf{A}_1 & \mathbf{A}_2 \\ \mathbf{A}_3 & \mathbf{A}_4 \end{bmatrix} = \begin{bmatrix} \mathbf{R} & * \\ \mathbf{0} & \mathbf{B} \end{bmatrix}$$

kernel basis of $\left[\begin{smallmatrix} \mathbf{A}_1 \\ \mathbf{A}_3 \end{smallmatrix}\right]$

$\mathbf{K}_1\mathbf{A}_2 + \mathbf{K}_2\mathbf{A}_4$

row basis of $\left[\begin{smallmatrix} \mathbf{A}_1 \\ \mathbf{A}_3 \end{smallmatrix}\right]$

property: $\det(\mathbf{A}) = \det(\mathbf{R})\det(\mathbf{B})$

be lazy: if hard to compute, don't compute [N.-Pernet'21]

obstacle = removing log factors in row basis computation

$\Rightarrow$ solution: remove row basis computation

$$\begin{bmatrix} \mathbf{I}_{m/2} & \mathbf{0} \\ \mathbf{K}_1 & \mathbf{K}_2 \end{bmatrix} \begin{bmatrix} \mathbf{A}_1 & \mathbf{A}_2 \\ \mathbf{A}_3 & \mathbf{A}_4 \end{bmatrix} = \begin{bmatrix} \mathbf{A}_1 & \mathbf{A}_2 \\ \mathbf{0} & \mathbf{B} \end{bmatrix}$$

property: $\det(\mathbf{A}) = \det(\mathbf{A}_1)\det(\mathbf{B}) / \det(\mathbf{K}_2)$

further obstacles (consequences of laziness)

$$\begin{bmatrix} \mathbf{I}_{m/2} & \mathbf{0} \\ \mathbf{K}_1 & \mathbf{K}_2 \end{bmatrix} \begin{bmatrix} \mathbf{A}_1 & \mathbf{A}_2 \\ \mathbf{A}_3 & \mathbf{A}_4 \end{bmatrix} = \begin{bmatrix} \mathbf{A}_1 & \mathbf{A}_2 \\ \mathbf{0} & \mathbf{B} \end{bmatrix}$$

property: $\det(\mathbf{A}) = \det(\mathbf{A}_1)\det(\mathbf{B}) / \det(\mathbf{K}_2)$

further obstacles (consequences of laziness)

$$\begin{bmatrix} \mathbf{I}_{m/2} & \mathbf{0} \\ \mathbf{K}_1 & \mathbf{K}_2 \end{bmatrix} \begin{bmatrix} \mathbf{A}_1 & \mathbf{A}_2 \\ \mathbf{A}_3 & \mathbf{A}_4 \end{bmatrix} = \begin{bmatrix} \mathbf{A}_1 & \mathbf{A}_2 \\ \mathbf{0} & \mathbf{B} \end{bmatrix}$$

property: $\det(\mathbf{A}) = \det(\mathbf{A}_1)\det(\mathbf{B}) / \det(\mathbf{K}_2)$

- 👍 no $\log(m)$ in the computation of $\mathbf{A}_1$, $\mathbf{B}$, $\mathbf{K}_2$
- 👎 requires nonsingular $\mathbf{A}_1$, otherwise $\det(\mathbf{K}_2) = 0$
- 👎 3 recursive calls in matrix size $m/2$ is 👍, but requires $\sum \mathrm{rdeg}(\mathbf{A}_1) \leqslant D/2$ otherwise degree control is too weak. (this implies $\sum \mathrm{rdeg}(\mathbf{K}_2) \leqslant D/2$)

further obstacles (consequences of laziness)

$$\begin{bmatrix} \mathbf{I}_{m/2} & \mathbf{0} \\ \mathbf{K}_1 & \mathbf{K}_2 \end{bmatrix} \begin{bmatrix} \mathbf{A}_1 & \mathbf{A}_2 \\ \mathbf{A}_3 & \mathbf{A}_4 \end{bmatrix} = \begin{bmatrix} \mathbf{A}_1 & \mathbf{A}_2 \\ \mathbf{0} & \mathbf{B} \end{bmatrix}$$

property: $\det(\mathbf{A}) = \det(\mathbf{A}_1)\det(\mathbf{B})/\det(\mathbf{K}_2)$

- 👍 no $\log(m)$ in the computation of $\mathbf{A}_1$, $\mathbf{B}$, $\mathbf{K}_2$
- 👎 requires nonsingular $\mathbf{A}_1$, otherwise $\det(\mathbf{K}_2) = 0$
- 👎 3 recursive calls in matrix size $m/2$ is 👍, but requires $\sum \mathrm{rdeg}(\mathbf{A}_1) \leqslant D/2$ otherwise degree control is too weak. (this implies $\sum \mathrm{rdeg}(\mathbf{K}_2) \leqslant D/2$)

solution: require $\mathbf{A}$ in weak Popov form

(the characteristic matrix $\mathbf{A} = x\mathbf{I}_m - \mathbf{M}$ is in Popov form)

- 👍 implies $\mathbf{A}_1$ nonsingular and $\sum \mathrm{rdeg}(\mathbf{A}_1) \leqslant D/2$ up to easy transformations
- 👍 both $\mathbf{A}_1$ and $\mathbf{B}$ are also in weak Popov form $\Rightarrow$ suitable for recursive calls
- 👎 $\mathbf{K}_2$ is in "shifted reduced" form... find weak Popov $\mathbf{P}$ with same determinant

further obstacles (consequences of laziness)

$$\begin{bmatrix} \mathbf{I}_{m/2} & \mathbf{0} \\ \mathbf{K}_1 & \mathbf{K}_2 \end{bmatrix} \begin{bmatrix} \mathbf{A}_1 & \mathbf{A}_2 \\ \mathbf{A}_3 & \mathbf{A}_4 \end{bmatrix} = \begin{bmatrix} \mathbf{A}_1 & \mathbf{A}_2 \\ \mathbf{0} & \mathbf{B} \end{bmatrix}$$

property: $\det(\mathbf{A}) = \det(\mathbf{A}_1)\det(\mathbf{B})/\det(\mathbf{K}_2)$

- no $\log(m)$ in the computation of $\mathbf{A}_1$, $\mathbf{B}$, $\mathbf{K}_2$
- requires nonsingular $\mathbf{A}_1$, otherwise $\det(\mathbf{K}_2) = 0$
- 3 recursive calls in matrix size $m/2$ is 👍, but requires $\sum \mathrm{rdeg}(\mathbf{A}_1) \leqslant D/2$ otherwise degree control is too weak. (this implies $\sum \mathrm{rdeg}(\mathbf{K}_2) \leqslant D/2$)

solution: require A in weak Popov form

(the characteristic matrix $\mathbf{A} = x\mathbf{I}_m - \mathbf{M}$ is in Popov form)

- implies $\mathbf{A}_1$ nonsingular and $\sum \mathrm{rdeg}(\mathbf{A}_1) \leqslant D/2$ up to easy transformations
- both $\mathbf{A}_1$ and $\mathbf{B}$ are also in weak Popov form $\Rightarrow$ suitable for recursive calls
- $\mathbf{K}_2$ is in "shifted reduced" form... find weak Popov $\mathbf{P}$ with same determinant

solution: exploit degree knowledge to accelerate transformations

s-reduced $\Rightarrow$ s-weak Popov $\Rightarrow$ s-Popov

outline

- **approximate/interpolate**
  - introduction, links with structured matrices
  - vector interpolation & matrix normal forms
  - iterative & divide and conquer algorithms
- **characteristic polynomial**
  - previous work and log factors to remove
  - result: "asymptotically optimal" algorithm
  - new triangularization-based approach
- **modular composition**
- **change of order**

outline

- **approximate/interpolate**
  - introduction, links with structured matrices
  - vector interpolation & matrix normal forms
  - iterative & divide and conquer algorithms
- **characteristic polynomial**
  - previous work and log factors to remove
  - result: “asymptotically optimal” algorithm
  - new triangularization-based approach
- **modular composition**
  - problem and context
  - acceleration via polynomial matrices
  - overview of the main new ingredients
- **change of order**

univariate polynomials: open problems

polynomials in $\mathbb{K}[x]_{\leqslant n}$: almost all basic operations are quasi-linear

i.e. complexity $O\tilde{}(n)$

- addition $f + g$, multiplication $f * g$
- division with remainder $f = qg + r$
- extended GCD $fu + gv = \gcd(f, g)$
- truncated inverse $f^{-1} \bmod x^n$
- multipoint eval. $f \mapsto f(x_1), \ldots, f(x_n)$
- interpolation $f(x_1), \ldots, f(x_n) \mapsto f$

[von zur Gathen, Gerhard – Modern Computer Algebra]

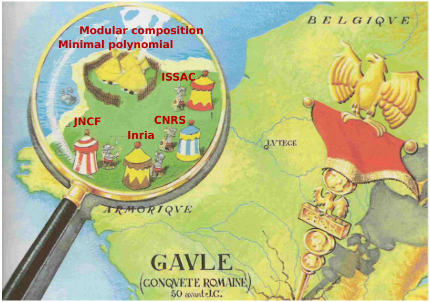
except. . .

univariate polynomials: open problems

minimal polynomial
given g, a, compute f such that $f(a) = 0 \bmod g$

modular composition
given g, a, h, compute $h(a) \bmod g$

related problems: power projections & inverse composition



The year is 2021 A.D.

Basic Polynomial Algebra is entirely occupied by Computer Algebraists.

Well not entirely!

One small village of indomitable open problems still holds out against the invaders. And life is not easy for the scientists who garrison the fortified camps of ISSAC, JNCF, Inria, CNRS...

complexity improvements

[Neiger-Salvy-Schost-Villard J.ACM 2024]

for generic input || using randomization

$$\left.\begin{array}{l}\textbf{minimal polynomial}\\ \textbf{modular composition}\end{array}\right\}\ \textbf{in}\ O\tilde{~}(n^{(\omega+2)/3})$$

exponent $(\omega+2)/3$: 1.67 for $\omega = 3$, 1.6 for $\omega = 2.8$, 1.46 for $\omega = 2.38$

previous work (composition)
- naive: $O\tilde{~}(n^2)$
- [Brent-Kung 1978]: $O(n^{(\omega+1)/2})$

previous work (minpoly)
- naive: $O\tilde{~}(n^\omega)$ or $O\tilde{~}(n^2)$
- [Shoup 1994]: $O(n^{(\omega+1)/2})$

exponent $(\omega+1)/2$: 2 for $\omega = 3$, 1.9 for $\omega = 2.8$, 1.69 for $\omega = 2.38$

breakthough [Kedlaya-Umans 2011]:
composition in $O\tilde{~}(n \log(q))$ bit operations, over $\mathbb{K} = \mathbb{F}_q$

quasi-linear bit complexity, yet currently impractical [van der Hoeven-Lecerf 2020]

software improvements

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

implementation for modular composition is in progress

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

random input polynomials $\Rightarrow$ "generic"

	general prime		FFT prime	
n	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

relies on PML for polynomial matrix operations:

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

input: $g(x)$ of degree n, $a(x)$ of degree $< n$, $h(y)$ of degree $< n$
output: $h(a(x)) \bmod g(x)$

$$h(a) \bmod g \; = \; h_0 + h_1(a \bmod g) + h_2(a^2 \bmod g) + \cdots + h_{n-1}(a^{n-1} \bmod g)$$

complexity: $\tilde{O}(n^2)$ for $O(n)$ multiplications by a modulo g
in practice: constant-factor speedup via precomputations on a and g

naive via Horner evaluation

classical composition algorithms

baby-step giant-step algorithm

input: $g(x)$ of degree n, $a(x)$ of degree $< n$, $h(y)$ of degree $< n$
output: $h(a(x)) \bmod g(x)$

$$h(a) \bmod g \;=\; h_0 + h_1(a \bmod g) + h_2(a^2 \bmod g) + \cdots + h_{n-1}(a^{n-1} \bmod g)$$

complexity: $\tilde{O}(n^2)$ for $O(n)$ multiplications by a modulo g
in practice: constant-factor speedup via precomputations on a and g

naive via Horner evaluation

classical composition algorithms

baby-step giant-step algorithm

[Paterson-Stockmeyer 1971, Brent-Kung 1978]

rely on matrix multiplication using "slices" of length $\nu = \sqrt{n}$
$h(y) = S_0(y) + y^{\nu}S_1(y) + y^{2\nu}S_2(y) + \cdots + y^{(\nu-1)\nu}S_{\nu-1}(y)$

define $\alpha = a^{\nu} \bmod g$

$$h(a) = S_0(a) + \alpha S_1(a) + \alpha^2 S_2(a) + \cdots + \alpha^{\nu-1}S_{\nu-1}(a) \mod g$$

complexity: $\tilde{O}(n^{3/2})$ for $O(\sqrt{n})$ multiplications by a and α 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

input: $g(x)$ of degree n, $a(x)$ of degree $< n$, $h(y)$ of degree $< n$
output: $h(a(x)) \bmod g(x)$

$$h(a) \bmod g \;=\; h_0 + h_1(a \bmod g) + h_2(a^2 \bmod g) + \cdots + h_{n-1}(a^{n-1} \bmod g)$$

complexity: $O\tilde{}(n^2)$ for $O(n)$ multiplications by a modulo g
in practice: constant-factor speedup via precomputations on a and g

naive via Horner evaluation

classical composition algorithms

baby-step giant-step algorithm

```
// 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

input: $g(x)$ of degree n, $a(x)$ of degree $< n$, $h(y)$ of degree $< n$
output: $h(a(x)) \bmod g(x)$

$$h(a) \bmod g \;=\; h_0 + h_1(a \bmod g) + h_2(a^2 \bmod g) + \cdots + h_{n-1}(a^{n-1} \bmod g)$$

complexity: $\tilde{O}(n^2)$ for $O(n)$ multiplications by a modulo g
in practice: constant-factor speedup via precomputations on a and g

naive via Horner evaluation

classical composition algorithms

baby-step giant-step algorithm

$$h(a) = S_0(a) + \alpha S_1(a) + \alpha^2 S_2(a) + \cdots + \alpha^{\nu-1} S_{\nu-1}(a)$$

recall: $\alpha = a^\nu \bmod g$

$$= \begin{bmatrix} 1 & \alpha & \cdots & \alpha^{\nu-1} \end{bmatrix} \begin{bmatrix} S_0(a) \\ S_1(a) \\ \vdots \\ S_{\nu-1}(a) \end{bmatrix}$$

$$= \begin{bmatrix} 1 & \alpha & \cdots & \alpha^{\nu-1} \end{bmatrix} \begin{bmatrix} S_{0,0} & S_{0,1} & \cdots & S_{0,\nu-1} \\ S_{1,0} & S_{1,1} & \cdots & S_{1,\nu-1} \\ \vdots & \vdots & & \vdots \\ S_{\nu-1,0} & S_{\nu-1,1} & \cdots & S_{\nu-1,\nu-1} \end{bmatrix} \begin{bmatrix} 1 \\ a \\ \vdots \\ a^{\nu-1} \end{bmatrix}$$

input: $g(x)$ of degree n, $a(x)$ of degree $< n$, $h(y)$ of degree $< n$
output: $h(a(x)) \bmod g(x)$

$$h(a) \bmod g \;=\; h_0 + h_1(a \bmod g) + h_2(a^2 \bmod g) + \cdots + h_{n-1}(a^{n-1} \bmod g)$$

complexity: $O\tilde{}(n^2)$ for $O(n)$ multiplications by a modulo g
in practice: constant-factor speedup via precomputations on a and g

naive via Horner evaluation

classical composition algorithms

baby-step giant-step algorithm

$$h(a) = S_0(a) + \alpha S_1(a) + \alpha^2 S_2(a) + \cdots + \alpha^{\nu-1} S_{\nu-1}(a)$$

recall: $\alpha = a^\nu \bmod g$

$$= \begin{bmatrix} 1 & \alpha & \cdots & \alpha^{\nu-1} \end{bmatrix} \begin{bmatrix} S_0(a) \\ S_1(a) \\ \vdots \\ S_{\nu-1}(a) \end{bmatrix}$$

length ν vectors over $\mathbb{K}[x]_{<n}$

$\nu \times \nu$ matrix over $\mathbb{K}$

$$= \begin{bmatrix} 1 & \alpha & \cdots & \alpha^{\nu-1} \end{bmatrix} \begin{bmatrix} S_{0,0} & S_{0,1} & \cdots & S_{0,\nu-1} \\ S_{1,0} & S_{1,1} & \cdots & S_{1,\nu-1} \\ \vdots & \vdots & & \vdots \\ S_{\nu-1,0} & S_{\nu-1,1} & \cdots & S_{\nu-1,\nu-1} \end{bmatrix} \begin{bmatrix} 1 \\ a \\ \vdots \\ a^{\nu-1} \end{bmatrix}$$

input: $g(x)$ of degree n, $a(x)$ of degree $< n$, $h(y)$ of degree $< n$
output: $h(a(x)) \bmod g(x)$

$$h(a) \bmod g = h_0 + h_1(a \bmod g) + h_2(a^2 \bmod g) + \cdots + h_{n-1}(a^{n-1} \bmod g)$$

complexity: $\tilde{O}(n^2)$ for $O(n)$ multiplications by a modulo g
in practice: constant-factor speedup via precomputations on a and g

naive via Horner evaluation

classical composition algorithms

baby-step giant-step algorithm

$$h(a) = S_0(a) + \alpha S_1(a) + \alpha^2 S_2(a) + \cdots + \alpha^{\nu-1} S_{\nu-1}(a)$$

recall: $\alpha = a^\nu \bmod g$

$$= \begin{bmatrix} 1 & \alpha & \cdots & \alpha^{\nu-1} \end{bmatrix} \begin{bmatrix} S_0(a) \\ S_1(a) \\ \vdots \\ S_{\nu-1}(a) \end{bmatrix}$$

length ν vectors over $\mathbb{K}[x]_{<n}$

$\nu \times \nu$ matrix over $\mathbb{K}$

$$\begin{bmatrix} S_{0,0} & S_{0,1} & \cdots & S_{0,\nu-1} \\ S_{1,0} & S_{1,1} & \cdots & S_{1,\nu-1} \\ \vdots & \vdots & & \vdots \\ S_{\nu-1,0} & S_{\nu-1,1} & \cdots & S_{\nu-1,\nu-1} \end{bmatrix} \begin{bmatrix} 1 \\ a \\ \vdots \\ a^{\nu-1} \end{bmatrix}$$

= matrix multiplication $(n \times \sqrt{n}) * (\sqrt{n} \times \sqrt{n}) * (\sqrt{n} \times n)$

Shoup's minpoly algorithm

[Shoup 1994, 1999]

0. choose random vector $[\ell_1 \ \cdots \ \ell_n] \in \mathbb{K}^n$
$\rightarrow$ defines a linear form $\ell : \mathbb{K}[x]/\langle g \rangle \rightarrow \mathbb{K}$

1. compute linear recurrent sequence
$\ell(1), \ell(a \bmod g), \ldots, \ell(a^{2n-1} \bmod g)$

2. compute minimal recurrence relation $f(y)$
via Berlekamp-Massey / Padé approximation

$$\begin{gathered}\text{minpoly } f(y)\\ \Downarrow\\ f(a) = 0 \bmod g\\ \Downarrow\\ f(y) = \text{relation for } (a^k \bmod g)_k\\ \Downarrow\\ f(y) = \text{relation for } (\ell(a^k \bmod g))_k\end{gathered}$$

Shoup's minpoly algorithm

[Shoup 1994, 1999]

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

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

1. compute linear recurrent sequence $\ell(1), \ell(a \bmod g), \ldots, \ell(a^{2n-1} \bmod g)$

2. compute minimal recurrence relation $f(y)$ via Berlekamp-Massey / Padé approximation

$$\begin{array}{c} \text{minpoly } f(y) \\ \Downarrow \\ f(a) = 0 \bmod g \\ \Downarrow \\ f(y) = \text{relation for } (a^k \bmod g)_k \\ \Downarrow \\ f(y) = \text{relation for } (\ell(a^k \bmod g))_k \end{array}$$

$\rightarrow$ related to algorithm of [Wiedemann 1986]:

$$\ell(a^k \bmod g) = \begin{bmatrix} \ell_1 & \cdots & \ell_n \end{bmatrix} \ \mathbf{A}^k \ \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

where $\mathbf{A} \in \mathbb{K}^{n \times n}$ is the "multiplication matrix" of $a(x)$ modulo $g(x)$

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

then $\ell(a^k \bmod g) =$ constant coeff of $a^k \bmod g$

new minpoly algorithm: blocking & baby-step giant-step

block Wiedemann approach [Coppersmith 1994]

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

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

new minpoly algorithm: blocking & baby-step giant-step

block Wiedemann approach [Coppersmith 1994]

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

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

1. compute linear recurrent matrix sequence:

$$\mathbf{I}_m, \quad \begin{bmatrix} \mathbf{I}_m & \mathbf{0} \end{bmatrix} \mathbf{A} \begin{bmatrix} \mathbf{I}_m \\ \mathbf{0} \end{bmatrix}, \quad \ldots, \quad \begin{bmatrix} \mathbf{I}_m & \mathbf{0} \end{bmatrix} \mathbf{A}^{2d-1} \begin{bmatrix} \mathbf{I}_m \\ \mathbf{0} \end{bmatrix}$$

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

new minpoly algorithm: blocking & baby-step giant-step

block Wiedemann approach [Coppersmith 1994]

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

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

1. compute linear recurrent matrix sequence:

$$\mathbf{I}_m, \quad \begin{bmatrix}\mathbf{I}_m & \mathbf{0}\end{bmatrix} \mathbf{A} \begin{bmatrix}\mathbf{I}_m \\ \mathbf{0}\end{bmatrix}, \quad \ldots, \quad \begin{bmatrix}\mathbf{I}_m & \mathbf{0}\end{bmatrix} \mathbf{A}^{2d-1} \begin{bmatrix}\mathbf{I}_m \\ \mathbf{0}\end{bmatrix}$$

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

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

- $f(y) = \det(\mathbf{P}(y))$ is the minimal polynomial of a modulo g
- $\mathbf{P}(y)$ is useful for modular composition

modular composition, first step

summary of the minpoly algorithm:

- specialization of first step of bivariate resultant [Villard 2018]
- accelerated by baby-step giant-step $\rightarrow \tilde{O}(md^{(\omega+1)/2} + m^{\omega}d)$
- genericity or randomization required for efficiency

computes an $m \times m$ polynomial matrix $\mathbf{P}(y)$ of degree $\leqslant d$
whose columns are minimal polynomial vectors of $a \bmod g$

change of representation

univariate vector	$\longleftrightarrow$	bivariate polynomial
$\begin{bmatrix} F_0(y) \\ F_1(y) \\ \vdots \\ F_{m-1}(y) \end{bmatrix}$	$\longleftrightarrow$	$F(x,y) = \sum_{i<m} F_i(y)x^i$

columns of $\mathbf{P}(y) \quad \Rightarrow \quad F(x,a) = 0 \bmod g$

Popov basis of submodule of canceling vectors in $\mathbb{K}[y]^m$ $\longleftrightarrow$ Gröbner basis of ideal $\langle g(x), y - a(x)\rangle$ in $\mathbb{K}[x,y]$

modular composition, second step

composition $h(y) \to b(x)$ $= h(a) \bmod g$

$$= h(a) + F(x, a) \bmod g$$

$$= H(x, a) \bmod g$$

$H(x, y) = h(y) + F(x, y)$ for any $F(x, y)$ generated by $\mathbf{P}(y)$

find $H(x, y)$ such that $\begin{cases} \deg_x(H) < m, \quad \deg_y(H) < d \\ h(a) = H(x, a) \bmod g \end{cases}$

modular composition, second step

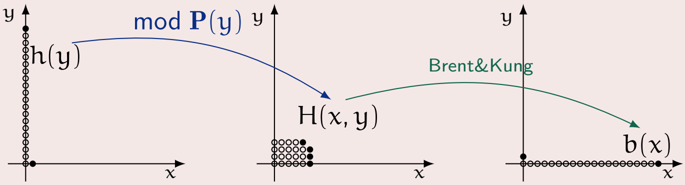
composition $h(y) \to b(x)$ $= h(a) \bmod g$
$= h(a) + F(x, a) \bmod g$
$= H(x, a) \bmod g$

$H(x,y) = h(y) + F(x,y)$ for any $F(x,y)$ generated by $\mathbf{P}(y)$

find $H(x,y)$ such that $\begin{cases} \deg_x(H) < m, \quad \deg_y(H) < d \\ h(a) = H(x,a) \bmod g \end{cases}$

computing $H(x,a) \bmod g$ costs $\tilde{O}(md^{(\omega+1)/2})$

extending Brent&Kung's approach [Nüsken-Ziegler'04]



modular composition, second step

composition $h(y) \to b(x)$ $= h(a) \bmod g$
$= h(a) + F(x,a) \bmod g$
$= H(x,a) \bmod g$

$H(x,y) = h(y) + F(x,y)$ for any $F(x,y)$ generated by $\mathbf{P}(y)$

find $H(x,y)$ such that $\begin{cases} \deg_x(H) < m, \quad \deg_y(H) < d \\ h(a) = H(x,a) \bmod g \end{cases}$

computing $H(x,a) \bmod g$ costs $\tilde{O}(md^{(\omega+1)/2})$

extending Brent&Kung's approach [Nüsken-Ziegler'04]

finding $H(x,y)$: matrix division with remainder

$$\begin{bmatrix} h(y) \\ 0 \\ \vdots \\ 0 \end{bmatrix} = \mathbf{P}(y)\mathbf{Q}(y) + \begin{bmatrix} H_0(y) \\ H_1(y) \\ \vdots \\ H_{m-1}(y) \end{bmatrix} \quad \text{degree} < d$$

complexity $\tilde{O}(m^\omega d)$

complexity minimized for $m = n^{1/3}, d = n^{2/3}$
$\tilde{O}(n^{(\omega+2)/3})$

outline

- **approximate/interpolate**
  - introduction, links with structured matrices
  - vector interpolation & matrix normal forms
  - iterative & divide and conquer algorithms
- **characteristic polynomial**
  - previous work and log factors to remove
  - result: "asymptotically optimal" algorithm
  - new triangularization-based approach
- **modular composition**
  - problem and context
  - acceleration via polynomial matrices
  - overview of the main new ingredients
- **change of order**

outline

approximate/interpolate

- introduction, links with structured matrices
- vector interpolation & matrix normal forms
- iterative & divide and conquer algorithms

characteristic polynomial

- previous work and log factors to remove
- result: "asymptotically optimal" algorithm
- new triangularization-based approach

modular composition

- problem and context
- acceleration via polynomial matrices
- overview of the main new ingredients

change of order

- problem and result
- assumptions and existing algorithms
- paradigm shift: ~~sparse~~ $\to$ structured

problem: change of monomial order

Input:

- two monomial orders $\preccurlyeq_1$ and $\preccurlyeq_2$ on $\mathbb{K}[x_1, \ldots, x_n]$
- a reduced $\preccurlyeq_1$-Gröbner basis $\mathcal{G}$

Assumption:

- the ideal $\mathcal{I} = \langle \mathcal{G} \rangle$ is zero-dimensional

Output:

- the reduced $\preccurlyeq_2$-Gröbner basis of $\mathcal{I}$

problem: change of monomial order

Input:

- two monomial orders $\preccurlyeq_1$ and $\preccurlyeq_2$ on $\mathbb{K}[x_1, \ldots, x_n]$
- a reduced $\preccurlyeq_1$-Gröbner basis $\mathcal{G}$

Assumption:

- the ideal $\mathcal{I} = \langle \mathcal{G} \rangle$ is zero-dimensional

Output:

- the reduced $\preccurlyeq_2$-Gröbner basis of $\mathcal{I}$

example: solving multivariate polynomial systems

$$\begin{cases} f_1(x_1, \ldots, x_n) = 0 \\ \vdots \\ f_m(x_1, \ldots, x_n) = 0 \end{cases} \quad \text{with finitely many solutions over } \overline{\mathbb{K}}$$

$\mathcal{G}_{\mathrm{drl}}$ the reduced $\preccurlyeq_{\mathrm{drl}}$-GB of $\langle f_1, \ldots, f_m \rangle$ [Faugère's F4/F5, 2002]

$$\Big\downarrow \text{ change of order } \preccurlyeq_{\mathrm{drl}} \rightarrow \preccurlyeq_{\mathrm{lex}}$$

$\mathcal{G}_{\mathrm{lex}}$ the reduced $\preccurlyeq_{\mathrm{lex}}$-GB of $\langle f_1, \ldots, f_m \rangle$

problem: change of monomial order

Input:

- two monomial orders $\preccurlyeq_1$ and $\preccurlyeq_2$ on $\mathbb{K}[x_1, \dots, x_n]$
- a reduced $\preccurlyeq_1$-Gröbner basis $\mathcal{G}$

Assumption:

- the ideal $\mathcal{I} = \langle \mathcal{G} \rangle$ is zero-dimensional

Output:

- the reduced $\preccurlyeq_2$-Gröbner basis of $\mathcal{I}$

example: multivariate interpolation with degree constraints

$\mathcal{I}$ = vanishing ideal of known points $\alpha_1, \dots, \alpha_D \in \mathbb{K}^n$

$\preccurlyeq$ = monomial order defined from the degree constraints

$\mathcal{G}_{\text{lex}}$ the reduced $\preccurlyeq_{\text{lex}}$-GB of $\mathcal{I}$

[Möller-Bucherberger 1982, Cerlienco-Mureddu 1995, Ceria-Mora 2019]

$$\big\downarrow \ \text{change of order } \preccurlyeq_{\text{lex}} \rightarrow \preccurlyeq$$

$\mathcal{G}$ the reduced $\preccurlyeq$-GB of $\mathcal{I}$

[Jérémy Berthomieu & Vincent Neiger & Mohab Safey El Din, ISSAC 2022]

deterministic change of order with complexity $\tilde{O}(t^{\omega-1}D)$

change of order: better complexity & faster implementation

for $\preccurlyeq_2 = \preccurlyeq_{\mathsf{lex}}$, under classical assumptions (stability + shape position)

ISSAC
International Symposium on
Symbolic and Algebraic Computation

July 4-7, 2022
Lille, France

[Jérémy Berthomieu & Vincent Neiger & Mohab Safey El Din, ISSAC 2022]
deterministic change of order with complexity $O\tilde{~}(t^{\omega-1}D)$

change of order: better complexity & faster implementation

for $\preccurlyeq_2 = \preccurlyeq_{\text{lex}}$, under classical assumptions (stability + shape position)

description of complexity

- ω = complexity exponent of matrix multiplication
  $O\tilde{~}(\cdot)$ hides a few logarithmic terms in $\frac{D}{t}$
- D = degree of the ideal $\mathcal{I} = \langle \mathcal{G} \rangle$
  = vector space dimension of $\mathbb{K}[x]/\mathcal{I}$
- t = number of polynomials in $\mathcal{G}$ with leading term divisible by x_n
  (in particular, $t \leqslant D$)

[Jérémy Berthomieu & Vincent Neiger & Mohab Safey El Din, ISSAC 2022]

deterministic change of order with complexity $\tilde{O}(t^{\omega-1}D)$

change of order: better complexity & faster implementation

for $\preccurlyeq_2 = \preccurlyeq_{\mathrm{lex}}$, under classical assumptions (stability + shape position)

summary of previous results

general algorithms (deterministic, $\preccurlyeq_1 \to \preccurlyeq_2$):

- no assumption: $O(nD^3)$ [Faugere-Gianni-Lazard-Mora 1993]
- with stability: $O(nD^\omega \log(D))$ [Neiger-Schost 2020]

specific algorithms (randomized, $\preccurlyeq_{\mathrm{drl}} \to \preccurlyeq_{\mathrm{lex}}$, with stability+shape):

- dense linear algebra: $O(D^\omega \log(D))$ [Faugère-Gaudry-Huot-Renault 2014]
- sparse linear algebra: $O(tD^2)$ [Faugère-Mou 2011+2017]

[Jérémy Berthomieu & Vincent Neiger & Mohab Safey El Din, ISSAC 2022]
deterministic change of order with complexity $O\tilde{}(t^{\omega-1}D)$

change of order: better complexity & faster implementation

for $\preccurlyeq_2 = \preccurlyeq_{\mathrm{lex}}$, under classical assumptions (stability + shape position)

ingredients of new algorithm

- paradigm shift concerning the core computational object:

$M \in \mathbb{K}^{D \times D}$ with t dense rows $\xrightarrow{\text{compress}}$ $\mathbf{P} \in \mathbb{K}[x_n]^{t \times t}$ of degree $\frac{D}{t}$

multiplication by x_n in $\mathbb{K}[x]/\mathcal{J}$ $\longrightarrow$ $\mathbb{K}[x_n]$-module, generates $\mathcal{J}$

- preserving essential consequence of stability: $\mathbf{P}$ obtained for free from $\mathcal{G}$

- new result: Hermite normal form of $\mathbf{P}$ yields $\mathcal{G}_{\mathrm{lex}}$

terms of modules, means that we are working column-wise, see the class docum

```
sage: M.degree_matrix(shifts=[-1,2], row_wise=False)
[ 0 -2 -1]
[ 5 -2 -2]
```

hermite_form(*include_zero_rows=True, transformation=False*)

Return the Hermite form of this matrix.

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

INPUT:

- `include_zero_rows` – boolean (default: `True`); if `False`, the zero rows in the output deleted
- `transformation` – boolean (default: `False`); if `True`, return the transformation mat

software performance

EXAMPLES:

```
sage: M.<x> = 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])
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.

```
    // order that remains to be dealt with
    VecLong rem_order(order);

    // indices of columns/orders that remain to be dealt with
    VecLong rem_index(cdim);
    std::iota(rem_index.begin(), rem_index.end(), 0);

    // all along the algorithm, shift = shifted row degrees of approximant ba
    // (initially, input shift = shifted row degree of the identity matrix)

    while (not rem_order.empty())
    {
        /** Invariant:
         *  - appbas is a shift-ordered weak Popov approximant basis for
         *  (pmat,reached_order) where doneorder is the tuple such that
         *  -->reached_order[j] + rem_order[j] == order[j] for j appearing in
         *  -->reached_order[j] == order[j] for j not appearing in rem_index

        long j=0; // value if columnwise (order_wise==False)
        if (order_wise)
            j = std::distance(rem_order.begin(), std::max_element(rem_order.be
);

        long deg = order[rem_index[j]] - rem_order[j];

        // record the coefficients of degree deg of the column j of residual
        // also keep track of which of these are nonzero,
        // and among the nonzero ones, which is the first with smallest shift
        Vec<zz_p> const_residual;
        const_residual.SetLength(rdim);
        VecLong indices_nonzero;
        long piv = -1;
        for (long i = 0; i < rdim; ++i)
        {
            const_residual[i] = coeff(residual[i][j],deg);
            if (const_residual[i] != 0)
            {
                indices_nonzero.push_back(i);
                if (piv<0 || shift[i] < shift[piv])
                    piv = i;
            }
        }

        // if indices_nonzero is empty, const_residual is already zero, there
        if (not indices_nonzero.empty())
        {
            // update all rows of appbas and residual in indices_nonzero exce
```

src/mat_lzz_pX_approximant.cpp

open-source C/C++ software libraries

multivariate polynomial systems

msolve https://msolve.lip6.fr/

univariate polynomial matrices

PML https://github.com/vneiger/pml

software performance

open-source C/C++ software libraries

multivariate polynomial systems

msolve https://msolve.lip6.fr/

univariate polynomial matrices

PML https://github.com/vneiger/pml

compared algorithms:

- sparse FGLM [Faugère-Mou 2011,2017]
- block-Wiedemann variant [folklore]
- new Hermite normal form-based algorithm (without SIMD vectorization for the moment)

software performance

open-source C/C++ software libraries

multivariate polynomial systems

msolve https://msolve.lip6.fr/

univariate polynomial matrices

PML https://github.com/vneiger/pml

compared algorithms:

- sparse FGLM [Faugère-Mou 2011,2017]
- block-Wiedemann variant [folklore]
- new Hermite normal form-based algorithm (without SIMD vectorization for the moment)

software performance

random square system, n variables, degree d
over $\mathbb{K} = \mathbb{Z}/p\mathbb{Z}$ with 30-bit modulus p

open-source C/C++ software libraries

multivariate polynomial systems

msolve `https://msolve.lip6.fr/`

univariate polynomial matrices

PML `https://github.com/vneiger/pml`

compared algorithms:

- sparse FGLM [Faugère-Mou 2011,2017]
- block-Wiedemann variant [folklore]
- new Hermite normal form-based algorithm (without SIMD vectorization for the moment)

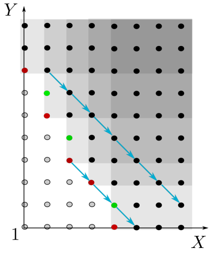
software performance

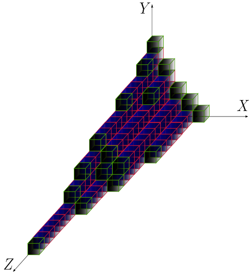
random square system, n variables, degree d over $\mathbb{K} = \mathbb{Z}/p\mathbb{Z}$ with 30-bit modulus p

n	d	D	t	spFGLM	block	HNF
12	2	4096	924	6.5	5.5	**5.3**
14	2	16384	3432	1011	358	**240**
16	2	65536	12870	58744	22059	**11359**
8	3	6561	1107	23.6	18.7	**15.1**
9	3	19683	3139	1302	525	**314**
10	3	59049	8953	34844	13315	**6709**
6	4	4096	580	4	3.5	**3.5**
7	4	16384	2128	575	225	**157**
8	4	65536	8092	36454	13609	**7231**

stability and multiplication matrix

x_n-**stability:** for any monomial $\mu \in \mathrm{lt}_{\preccurlyeq}(\mathcal{J})$ such that x_n divides μ, $\frac{x_i}{x_n}\mu \in \mathrm{lt}_{\preccurlyeq}(\mathcal{J})$ for all $i \in \{1,\dots,n-1\}$





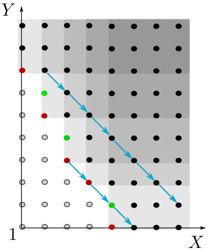
- related to classical notions of stability and of Borel-fixedness [Herzog-Hibi 2011, Galligo 1974, Bayer-Stillman 1987]
- easily verified: considering $\mu = \mathrm{lt}_{\preccurlyeq}(g)$ for $g \in \mathcal{G}$ is sufficient

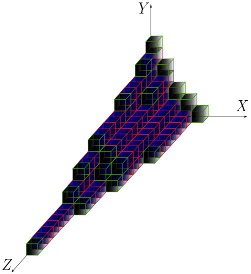
$\preccurlyeq$-monomial basis $\mathcal{B} = \{\varepsilon_1,\dots,\varepsilon_D\}$
$= $ monomials not in $\mathrm{lt}_{\preccurlyeq}(\mathcal{J})$
$= $ vector space basis of $\mathbb{K}[x]/\mathcal{J}$

- x_n-stability $\Leftrightarrow$ multiplying element $\varepsilon \in \mathcal{B}$ by x_n gives either $x_n\varepsilon \in \mathcal{B}$ or $x_n\varepsilon = \mathrm{lt}_{\preccurlyeq}(g)$ for some $g \in \mathcal{G}$
- in $\mathbb{K}[x]/\mathcal{J}$, the representation of $\mathrm{lt}_{\preccurlyeq}(g)$ on $\mathcal{B}$ is $\mathrm{lt}_{\preccurlyeq}(g) - g$

stability and multiplication matrix

x_n-stability: for any monomial $\mu \in \mathrm{lt}_{\preccurlyeq}(\mathcal{J})$ such that x_n divides μ, $\frac{x_i}{x_n}\mu \in \mathrm{lt}_{\preccurlyeq}(\mathcal{J})$ for all $i \in \{1, \ldots, n-1\}$





- related to classical notions of stability and of Borel-fixedness [Herzog-Hibi 2011, Galligo 1974, Bayer-Stillman 1987]
- easily verified: considering $\mu = \mathrm{lt}_{\preccurlyeq}(g)$ for $g \in \mathcal{G}$ is sufficient

$\preccurlyeq$-monomial basis $\mathcal{B} = \{\varepsilon_1, \ldots, \varepsilon_D\}$
= monomials not in $\mathrm{lt}_{\preccurlyeq}(\mathcal{J})$
= vector space basis of $\mathbb{K}[x]/\mathcal{J}$

- x_n-stability $\Leftrightarrow$ multiplying element $\varepsilon \in \mathcal{B}$ by x_n gives either $x_n\varepsilon \in \mathcal{B}$ or $x_n\varepsilon = \mathrm{lt}_{\preccurlyeq}(g)$ for some $g \in \mathcal{G}$
- in $\mathbb{K}[x]/\mathcal{J}$, the representation of $\mathrm{lt}_{\preccurlyeq}(g)$ on $\mathcal{B}$ is $\mathrm{lt}_{\preccurlyeq}(g) - g$

multiplication matrix $\mathbf{M}_n \in \mathbb{K}^{D\times D}$ of x_n in $\mathbb{K}[x]/\mathcal{J}$
- row i = representation of $x_n\varepsilon_i$ on $\mathcal{B}$
- deduced directly from $\hat{\mathcal{G}} = \{g \in \mathcal{G} \mid x_n \text{ divides } \mathrm{lt}_{\preccurlyeq}(g)\}$
- has $t = \#\hat{\mathcal{G}}$ dense rows and $D - t$ identity rows

shape position and lexicographic ideals

[Becker-Mora-Marinari-Traverso 1994]

shape position: $\mathcal{G}_{\text{lex}} = \{x_1 - g_1(x_n), \ldots, x_{n-1} - g_{n-1}(x_n), h(x_n)\}$
with $g_1, \ldots, g_{n-1}, h$ univariate in $\mathbb{K}[x_n]$
and $\deg(g_i) < \deg(h) = D$

x_n = smallest variable

g_i = parametrizations

shape position and lexicographic ideals

[Becker-Mora-Marinari-Traverso 1994]

shape position: $\mathcal{G}_{\text{lex}} = \{x_1 - g_1(x_n), \ldots, x_{n-1} - g_{n-1}(x_n), h(x_n)\}$
with $g_1, \ldots, g_{n-1}, h$ univariate in $\mathbb{K}[x_n]$
and $\deg(g_i) < \deg(h) = D$

x_n = smallest variable

g_i = parametrizations

for **polynomial system solving**:

- solutions = $(g_1(\alpha), \ldots, g_{n-1}(\alpha), \alpha)$ for all roots α of $h(x_n)$
- ensured by generic change of coordinates, if ideal is radical

shape position and lexicographic ideals

[Becker-Mora-Marinari-Traverso 1994]

shape position: $\mathcal{G}_{\text{lex}} = \{x_1 - g_1(x_n), \ldots, x_{n-1} - g_{n-1}(x_n), h(x_n)\}$
with $g_1, \ldots, g_{n-1}, h$ univariate in $\mathbb{K}[x_n]$
and $\deg(g_i) < \deg(h) = D$

x_n = smallest variable

g_i = parametrizations

for **polynomial system solving**:
- solutions = $(g_1(\alpha), \ldots, g_{n-1}(\alpha), \alpha)$ for all roots α of $h(x_n)$
- ensured by generic change of coordinates, if ideal is radical

computation from the multiplication matrix $\mathbf{M}_n$

$h \in \mathcal{I} \quad \Rightarrow \quad h(x_n)$ is zero in $\mathbb{K}[x]/\mathcal{I}$
- h gives a $\mathbb{K}$-linear combination between $\varepsilon_1, \varepsilon_1\mathbf{M}_n, \ldots, \varepsilon_1\mathbf{M}_n^D$
- the matrix $\begin{bmatrix} \varepsilon_1 \\ \varepsilon_1\mathbf{M}_n \\ \vdots \\ \varepsilon_1\mathbf{M}_n^{D-1} \end{bmatrix} \in \mathbb{K}^{D \times D}$ is invertible (taking $\varepsilon_1 = 1$)

$\Rightarrow$ $h(x_n)$ is the minpoly/charpoly of $\mathbf{M}_n$

previously: dense or sparse linear algebra

using dense linear algebra

$$\begin{bmatrix} -\mathrm{coeffs}(h) & 1 & & \\ -\mathrm{coeffs}(g_1) & & 1 & \\ \vdots & & & \ddots \\ -\mathrm{coeffs}(g_{n-1}) & & & & 1 \end{bmatrix} \begin{bmatrix} \varepsilon_1 \\ \varepsilon_1 \mathbf{M}_n \\ \vdots \\ \varepsilon_1 \mathbf{M}_n^{D-1} \\ \varepsilon_1 \mathbf{M}_n^{D} \\ \varepsilon_{x_1} \\ \vdots \\ \varepsilon_{x_{n-1}} \end{bmatrix} = 0$$

- compute Krylov iterates
  in $O(D^\omega \log(D))$ [Keller-Gehrig 1985]
  in $O(D^\omega)$ [Neiger-Pernet-Villard 2024]
- nullspace in $O(D^\omega)$ [Ibarra-Hui-Moran 1982]

deterministic algo in $O(D^\omega)$

previously: dense or sparse linear algebra

using dense linear algebra

$$\begin{bmatrix} -\mathrm{coeffs}(h) & 1 & & \\ -\mathrm{coeffs}(g_1) & & 1 & \\ \vdots & & & \ddots \\ -\mathrm{coeffs}(g_{n-1}) & & & & 1 \end{bmatrix} \begin{bmatrix} \varepsilon_1 \\ \varepsilon_1 \mathbf{M}_n \\ \vdots \\ \varepsilon_1 \mathbf{M}_n^{D-1} \\ \varepsilon_1 \mathbf{M}_n^{D} \\ \varepsilon_{x_1} \\ \vdots \\ \varepsilon_{x_{n-1}} \end{bmatrix} = 0$$

- compute Krylov iterates
  in $O(D^\omega \log(D))$ [Keller-Gehrig 1985]
  in $O(D^\omega)$ [Neiger-Pernet-Villard 2024]
- nullspace in $O(D^\omega)$ [Ibarra-Hui-Moran 1982]

deterministic algo in $O(D^\omega)$

using sparse linear algebra [Wiedemann 1986]

for random column vector $r \in \mathbb{K}^{D\times 1}$,
scalar sequence $(\varepsilon_1 \mathbf{M}_n^k r)_{0\leqslant k<2D}$
$\rightsquigarrow$ its minimal generator is $h(x_n)$

- compute recurrent sequence in $O(tD^2)$
  via matrix-vector products
- find generator h in $\tilde{O}(D)$ [GCD/Padé]
- find $g_1, \dots, g_{n-1}$ in $\tilde{O}(nD)$ via $n-1$
  Hankel systems [Faugère-Mou 2011, 2017]

randomized algo in $O(tD^2)$

paradigm shift: ~~sparse~~ $\rightarrow$ structured

multiplication by x_n in $\mathbb{K}[x_n]/\langle h(x_n)\rangle$

$$\updownarrow$$

companion matrix $\mathbf{M} = \begin{bmatrix} & 1 & & \\ & & \ddots & \\ & & & 1 \\ -h_0 & -h_1 & \cdots & -h_{D-1} \end{bmatrix}$

with charpoly$(\mathbf{M}) = h$

paradigm shift: ~~sparse~~ $\to$ structured

ideal $\mathcal{J} \subset \mathbb{F}_{29}[x_1, x_2, x_3]$ generated by the $\preccurlyeq_{\mathrm{drl}}$-GB

$$\begin{cases} \mathbf{x_3^4} + 3x_3^3 + 15x_1x_3 + 23x_2x_3 + 3x_3^2 + 26x_2 + 22x_3, \\ \mathbf{x_2x_3^2} + 5x_1x_3 + 28x_2x_3 + 3x_3^2 + 19x_1 + 15x_2 + 17, \\ \mathbf{x_1x_3^2} + 18x_3^3 + 24x_1x_3 + 27x_2x_3 + 19x_3^2 + 2x_1 + 9x_3 + 3, \\ x_2^2 + 12x_1x_3 + 26x_2x_3 + 5x_3^2 + 9x_1 + 6x_2 + 8x_3 + 6, \\ x_1x_2 + 6x_1x_3 + x_2x_3 + 17x_3^2 + 28x_1 + 12x_2 + 8x_3 + 11, \\ x_1^2 + x_1x_3 + 10x_2x_3 + 2x_3^2 + 3x_1 + 16x_2 + 21 \end{cases}$$

- $t = 3$ polynomials with $\preccurlyeq_{\mathrm{drl}}$-leading term divisible by x_3

  the first 3, with leading terms $\mathbf{x_3^4}, \mathbf{x_2x_3^2}, \mathbf{x_1x_3^2}$

- x_3-stability holds

  easily verified: for $\mu \in \{x_2x_3^2, x_1x_3^2, x_3^4\}$, $\frac{x_1}{x_3}\mu$ and $\frac{x_2}{x_3}\mu$ are in $\mathrm{lt}_{\preccurlyeq_{\mathrm{drl}}}(\mathcal{J})$

- zero-dimensional with $D = 8$

  $\preccurlyeq_{\mathrm{drl}}$-monomial basis $\mathcal{B} = (1, x_3, x_3^2, x_3^3, x_2, x_2x_3, x_1, x_1x_3)$

paradigm shift: ~~sparse~~ $\to$ structured

ideal $\mathcal{I} \subset \mathbb{F}_{29}[x_1, x_2, x_3]$ generated by the $\preccurlyeq_{\mathsf{drl}}$-GB

$$\begin{cases} \boldsymbol{x_3^4} + 3x_3^3 + 15x_1x_3 + 23x_2x_3 + 3x_3^2 + 26x_2 + 22x_3, \\ \boldsymbol{x_2x_3^2} + 5x_1x_3 + 28x_2x_3 + 3x_3^2 + 19x_1 + 15x_2 + 17, \\ \boldsymbol{x_1x_3^2} + 18x_3^3 + 24x_1x_3 + 27x_2x_3 + 19x_3^2 + 2x_1 + 9x_3 + 3, \\ \dots \end{cases}$$

- $t = 3$, $D = 8$
- x_3-stable
- monomial basis $\mathcal{B}$ =

$(1, x_3, x_3^2, x_3^3, x_2, x_2x_3, x_1, x_1x_3)$

paradigm shift: ~~sparse~~ $\to$ structured

ideal $\mathcal{J} \subset \mathbb{F}_{29}[x_1, x_2, x_3]$ generated by the $\preccurlyeq_{\mathrm{drl}}$-GB

$$\begin{cases} \mathbf{x_3^4} + 3x_3^3 + 15x_1x_3 + 23x_2x_3 + 3x_3^2 + 26x_2 + 22x_3, \\ \mathbf{x_2x_3^2} + 5x_1x_3 + 28x_2x_3 + 3x_3^2 + 19x_1 + 15x_2 + 17, \\ \mathbf{x_1x_3^2} + 18x_3^3 + 24x_1x_3 + 27x_2x_3 + 19x_3^2 + 2x_1 + 9x_3 + 3, \\ \dots \end{cases}$$

- $t = 3$, $D = 8$
- x_3-stable
- monomial basis $\mathcal{B} = (1, x_3, x_3^2, x_3^3, x_2, x_2x_3, x_1, x_1x_3)$

multiplication by x_3 in $\mathbb{K}[x_1, x_2, x_3]/\mathcal{J} \longleftrightarrow \mathbb{K}[x_3]$-module structure

$$\mathbf{M} = \left[\begin{array}{cccc|cc|cc} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & -22 & -3 & -3 & -26 & -23 & 0 & -15 \\ \hline 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ -17 & 0 & -3 & 0 & -15 & -28 & -19 & -5 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ -3 & -9 & -19 & -18 & 0 & -27 & -2 & -24 \end{array}\right] \in \mathbb{K}^{D \times D}$$

paradigm shift: ~~sparse~~ $\to$ structured

ideal $\mathcal{J} \subset \mathbb{F}_{29}[x_1, x_2, x_3]$ generated by the $\preccurlyeq_{\mathrm{drl}}$-GB

$$\left\{\begin{array}{l} \boldsymbol{x_3^4} + 3x_3^3 + 15x_1x_3 + 23x_2x_3 + 3x_3^2 + 26x_2 + 22x_3, \\ \boldsymbol{x_2x_3^2} + 5x_1x_3 + 28x_2x_3 + 3x_3^2 + 19x_1 + 15x_2 + 17, \\ \boldsymbol{x_1x_3^2} + 18x_3^3 + 24x_1x_3 + 27x_2x_3 + 19x_3^2 + 2x_1 + 9x_3 + 3, \\ \ldots \end{array}\right.$$

- $t = 3$, $D = 8$
- x_3-stable
- monomial basis $\mathcal{B} =$ $(1, x_3, x_3^2, x_3^3, x_2, x_2x_3, x_1, x_1x_3)$

$\preccurlyeq$-Gröbner basis + x_3-stability $\Rightarrow$ basis of $\mathbb{K}[x_3]$-submodule of $\mathcal{J}$

$$\mathbf{M} = \left[\begin{array}{cccc|cc|cc} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & -22 & -3 & -3 & -26 & -23 & 0 & -15 \\ \hline 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ -17 & 0 & -3 & 0 & -15 & -28 & -19 & -5 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ -3 & -9 & -19 & -18 & 0 & -27 & -2 & -24 \end{array}\right] \in \mathbb{K}^{D \times D}$$

basis of $\mathbb{K}[x_3]$-module $\mathcal{J} \cap (\mathbb{K}[x_3] + x_2\mathbb{K}[x_3] + x_1\mathbb{K}[x_3])$

$$\mathbf{P} = \begin{bmatrix} x_3^4 + 3x_3^3 + 3x_3^2 + 22x_3 & 23x_3 + 26 & 15x_3 \\ 3x_3^2 + 17 & x_3^2 + 28x_3 + 15 & 5x_3 + 19 \\ 18x_3^3 + 19x_3^2 + 9x_3 + 3 & 27x_3 & x_3^2 + 24x_3 + 2 \end{bmatrix} \in \mathbb{K}[x_3]^{t \times t}$$

paradigm shift: ~~sparse~~ $\rightarrow$ structured

ideal $\mathcal{I} \subset \mathbb{F}_{29}[x_1, x_2, x_3]$ generated by the $\preccurlyeq_{\mathrm{drl}}$-GB

$$\left\{\begin{array}{l} \mathbf{x_3^4} + 3x_3^3 + 15x_1x_3 + 23x_2x_3 + 3x_3^2 + 26x_2 + 22x_3, \\ \mathbf{x_2x_3^2} + 5x_1x_3 + 28x_2x_3 + 3x_3^2 + 19x_1 + 15x_2 + 17, \\ \mathbf{x_1x_3^2} + 18x_3^3 + 24x_1x_3 + 27x_2x_3 + 19x_3^2 + 2x_1 + 9x_3 + 3, \\ \ldots \end{array}\right.$$

- $t = 3,\ D = 8$
- x_3-stable
- monomial basis $\mathcal{B} = (1, x_3, x_3^2, x_3^3, x_2, x_2x_3, x_1, x_1x_3)$

$\preccurlyeq$-Gröbner basis + x_3-stability $\Rightarrow$ basis of $\mathbb{K}[x_3]$-submodule of $\mathcal{I}$

$$\mathbf{M} = \left[\begin{array}{cccc|cc|cc} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & -22 & -3 & -3 & -26 & -23 & 0 & -15 \\ \hline 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ -17 & 0 & -3 & 0 & -15 & -28 & -19 & -5 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ -3 & -9 & -19 & -18 & 0 & -27 & -2 & -24 \end{array}\right] \in \mathbb{K}^{D \times D}$$

- $\det(\mathbf{P}) = \mathrm{charpoly}(\mathbf{M})$
- $\mathrm{Smith}(\mathbf{P}) \simeq \mathrm{Frob}(\mathbf{M})$
- [Storjohann 2000] [Pernet-Storjohann 2007]
- column degrees $(4, 2, 2)$

basis of $\mathbb{K}[x_3]$-module $\mathcal{I} \cap (\mathbb{K}[x_3] + x_2\mathbb{K}[x_3] + x_1\mathbb{K}[x_3])$

$$\mathbf{P} = \begin{bmatrix} x_3^4 + 3x_3^3 + 3x_3^2 + 22x_3 & 23x_3 + 26 & 15x_3 \\ 3x_3^2 + 17 & x_3^2 + 28x_3 + 15 & 5x_3 + 19 \\ 18x_3^3 + 19x_3^2 + 9x_3 + 3 & 27x_3 & x_3^2 + 24x_3 + 2 \end{bmatrix} \in \mathbb{K}[x_3]^{t \times t}$$

from Hermite normal form to lex basis

$$\hat{\mathcal{G}} = \begin{bmatrix} f_1 \\ f_2 \\ f_3 \end{bmatrix} \simeq \begin{array}{c} \begin{array}{ccc} \mu_1 = 1 & \mu_2 = x_2 & \mu_3 = x_1 \end{array} \\ \begin{bmatrix} x_3^4 + 3x_3^3 + 3x_3^2 + 22x_3 & 23x_3 + 26 & 15x_3 \\ 3x_3^2 + 17 & x_3^2 + 28x_3 + 15 & 5x_3 + 19 \\ 18x_3^3 + 19x_3^2 + 9x_3 + 3 & 27x_3 & x_3^2 + 24x_3 + 2 \end{bmatrix} \end{array} \in \mathbb{K}[x_3]^{t \times t}$$

from Hermite normal form to lex basis

$$\hat{\mathcal{G}} = \begin{bmatrix} f_1 \\ f_2 \\ f_3 \end{bmatrix} \simeq \begin{array}{c} \begin{array}{ccc} \mu_1 = 1 & \mu_2 = x_2 & \mu_3 = x_1 \end{array} \\ \begin{bmatrix} x_3^4 + 3x_3^3 + 3x_3^2 + 22x_3 & 23x_3 + 26 & 15x_3 \\ 3x_3^2 + 17 & x_3^2 + 28x_3 + 15 & 5x_3 + 19 \\ 18x_3^3 + 19x_3^2 + 9x_3 + 3 & 27x_3 & x_3^2 + 24x_3 + 2 \end{bmatrix} \end{array} \in \mathbb{K}[x_3]^{t \times t}$$

$$\Big\downarrow$$

Hermite normal form

complexity $O\tilde{~}(t^{\omega-1}D)$

[Giorgi-Jeannerod-Villard 2003]
[Gupta-Storjohann 2011]
[Labahn-Neiger-Zhou 2017]

$$\begin{bmatrix} x_3^8 + 26x_3^7 + 8x_3^6 + 17x_3^5 + 19x_3^4 + x_3^3 + 28x_3^2 + 20x_3 + 18 & 0 & 0 \\ 28x_3^7 + 23x_3^6 + 17x_3^5 + 25x_3^4 + 24x_3^3 + 17x_3^2 + 14x_3 + 4 & 1 & 0 \\ 6x_3^7 + 13x_3^6 + 22x_3^5 + 12x_3^4 + 28x_3^3 + 24x_3^2 + 26x_3 + 14 & 0 & 1 \end{bmatrix}$$

from Hermite normal form to lex basis

$$\begin{array}{cccc} & \mu_1 = 1 & \mu_2 = x_2 & \mu_3 = x_1 \\ \hat{\mathcal{G}} = \begin{bmatrix} f_1 \\ f_2 \\ f_3 \end{bmatrix} \simeq & \multicolumn{3}{c}{\begin{bmatrix} x_3^4 + 3x_3^3 + 3x_3^2 + 22x_3 & 23x_3 + 26 & 15x_3 \\ 3x_3^2 + 17 & x_3^2 + 28x_3 + 15 & 5x_3 + 19 \\ 18x_3^3 + 19x_3^2 + 9x_3 + 3 & 27x_3 & x_3^2 + 24x_3 + 2 \end{bmatrix} \in \mathbb{K}[x_3]^{t\times t}} \end{array}$$

Hermite normal form

complexity $O\tilde{~}(t^{\omega-1}D)$

[Giorgi-Jeannerod-Villard 2003]
[Gupta-Storjohann 2011]
[Labahn-Neiger-Zhou 2017]

$$\begin{bmatrix} x_3^8 + 26x_3^7 + 8x_3^6 + 17x_3^5 + 19x_3^4 + x_3^3 + 28x_3^2 + 20x_3 + 18 & 0 & 0 \\ 28x_3^7 + 23x_3^6 + 17x_3^5 + 25x_3^4 + 24x_3^3 + 17x_3^2 + 14x_3 + 4 & 1 & 0 \\ 6x_3^7 + 13x_3^6 + 22x_3^5 + 12x_3^4 + 28x_3^3 + 24x_3^2 + 26x_3 + 14 & 0 & 1 \end{bmatrix}$$

row $i \simeq$ polynomial $p_{i1}(x_3) + p_{i2}(x_3)x_2 + p_{i3}(x_3)x_1$

$$x_3^8 + 26x_3^7 + 8x_3^6 + 17x_3^5 + 19x_3^4 + x_3^3 + 28x_3^2 + 20x_3 + 18,$$
$$x_2 + 28x_3^7 + 23x_3^6 + 17x_3^5 + 25x_3^4 + 24x_3^3 + 17x_3^2 + 14x_3 + 4,$$
$$x_1 + 6x_3^7 + 13x_3^6 + 22x_3^5 + 12x_3^4 + 28x_3^3 + 24x_3^2 + 26x_3 + 14$$

from Hermite normal form to lex basis

$$\hat{\mathcal{G}} = \begin{bmatrix} f_1 \\ f_2 \\ f_3 \end{bmatrix} \simeq \begin{array}{c} \begin{array}{ccc} \mu_1 = 1 & \qquad \mu_2 = x_2 & \qquad \mu_3 = x_1 \end{array} \\ \begin{bmatrix} x_3^4 + 3x_3^3 + 3x_3^2 + 22x_3 & 23x_3 + 26 & 15x_3 \\ 3x_3^2 + 17 & x_3^2 + 28x_3 + 15 & 5x_3 + 19 \\ 18x_3^3 + 19x_3^2 + 9x_3 + 3 & 27x_3 & x_3^2 + 24x_3 + 2 \end{bmatrix} \end{array} \in \mathbb{K}[x_3]^{t \times t}$$

$$\Big\downarrow$$

Hermite normal form

complexity $O\tilde{~}(t^{\omega-1}D)$

[Giorgi-Jeannerod-Villard 2003]
[Gupta-Storjohann 2011]
[Labahn-Neiger-Zhou 2017]

$$\begin{bmatrix} x_3^8 + 26x_3^7 + 8x_3^6 + 17x_3^5 + 19x_3^4 + x_3^3 + 28x_3^2 + 20x_3 + 18 & 0 & 0 \\ 28x_3^7 + 23x_3^6 + 17x_3^5 + 25x_3^4 + 24x_3^3 + 17x_3^2 + 14x_3 + 4 & 1 & 0 \\ 6x_3^7 + 13x_3^6 + 22x_3^5 + 12x_3^4 + 28x_3^3 + 24x_3^2 + 26x_3 + 14 & 0 & 1 \end{bmatrix}$$

$$\Big\downarrow \ \text{row } i \simeq \text{polynomial } p_{i1}(x_3) + p_{i2}(x_3)x_2 + p_{i3}(x_3)x_1$$

$$\begin{aligned} & x_3^8 + 26x_3^7 + 8x_3^6 + 17x_3^5 + 19x_3^4 + x_3^3 + 28x_3^2 + 20x_3 + 18, \\ & x_2 + 28x_3^7 + 23x_3^6 + 17x_3^5 + 25x_3^4 + 24x_3^3 + 17x_3^2 + 14x_3 + 4, \\ & x_1 + 6x_3^7 + 13x_3^6 + 22x_3^5 + 12x_3^4 + 28x_3^3 + 24x_3^2 + 26x_3 + 14 \end{aligned}$$

= lex basis

- improved complexity bound and faster software implementation
- based on the identification and exploitation of an algebraic structure
  $\rightsquigarrow$ $\mathbb{K}[x_n]$-modules and univariate polynomial matrix computations
- relating the Hermite normal form of a $\mathbb{K}[x_n]$-submodule of $\mathcal{I}$ and the lexicographic Gröbner basis of the ideal $\mathcal{I}$

summary

change of order: conclusion

perspectives

- software: add SIMD vectorization + integrate into msolve
  https://github.com/algebraic-solving/msolve
  https://msolve.lip6.fr/
- handle case with $\mathcal{I}$ non-radical but $\sqrt{\mathcal{I}}$ in shape position?
- relax assumptions about stability and shape position?

summary

approximate/interpolate

- introduction, links with structured matrices
- vector interpolation & matrix normal forms
- iterative & divide and conquer algorithms

characteristic polynomial

- previous work and log factors to remove
- result: “asymptotically optimal” algorithm
- new triangularization-based approach

modular composition

- problem and context
- acceleration via polynomial matrices
- overview of the main new ingredients

change of order

- problem and result
- assumptions and existing algorithms
- paradigm shift: ~~sparse~~ $\rightarrow$ structured