Jérémy Berthomieu

Vincent Neiger

Mohab Safey El Din

LIP6, Sorbonne University, France

Faster change of order algorithm for Gröbner bases under shape and stability assumptions

ACA 2022
Gebze Technical University, Turkey
15th August 2022

outline

- problem and result
- previous work
- new ingredients
- conclusion | perspectives

outline

- **problem and result**
- **previous work**
- **new ingredients**
- **conclusion | perspectives**

- problem: change of monomial order
- faster algorithm via univariate matrices
- improved software performances

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

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

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

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

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

problem: change of monomial order

Input:

- two monomial orders $\preccurlyeq_1$ and $\preccurlyeq_2$ on $\mathbb{K}[x_1, \ldots, x_n]$
- 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: multivariate interpolation with degree constraints

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

$\preccurlyeq$ = 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]

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

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

description of complexity

- ω = complexity exponent of matrix multiplication
  $\tilde{O}(\cdot)$ 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 $\preccurlyeq_2 = \preccurlyeq_{\text{lex}}$, 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, $\preccurlyeq_{\text{drl}} \to \preccurlyeq_{\text{lex}}$, 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 $\tilde{O}(t^{\omega-1}D)$

change of order: better complexity & faster implementation

for $\preccurlyeq_2 = \preccurlyeq_{\text{lex}}$, 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 $\xrightarrow{\text{compress}}$ $P \in \mathbb{K}[x_n]^{t\times t}$ of degree $\frac{D}{t}$

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

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

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

```
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.

```
164     // order that remains to be dealt with
165     VecLong rem_order(order);
166
167     // indices of columns/orders that remain to be dealt with
168     VecLong rem_index(cdim);
169     std::iota(rem_index.begin(), rem_index.end(), 0);
170
171     // all along the algorithm, shift = shifted row degrees of approximant ba
172     // (initially, input shift = shifted row degree of the identity matrix)
173
174     while (not rem_order.empty())
175     {
176         /** Invariant:
177          *  - appbas is a shift-ordered weak Popov approximant basis for
178          *  (pmat,reached_order) where doneorder is the tuple such that
179          *  -->reached_order[j] + rem_order[j] == order[j] for j appearing in
180          *  -->reached_order[j] == order[j] for j not appearing in rem_index
185         long j=0; // value if columnwise (order_wise==False)
186         if (order_wise)
187             j = std::distance(rem_order.begin(), std::max_element(rem_order.b
    );
188
189         long deg = order[rem_index[j]] - rem_order[j];
190
191         // record the coefficients of degree deg of the column j of residual
192         // also keep track of which of these are nonzero,
193         // and among the nonzero ones, which is the first with smallest shift
194         Vec<zz_p> const_residual;
195         const_residual.SetLength(rdim);
196         VecLong indices_nonzero;
197         long piv = -1;
198         for (long i = 0; i < rdim; ++i)
199         {
200             const_residual[i] = coeff(residual[i][j],deg);
201             if (const_residual[i] != 0)
202             {
203                 indices_nonzero.push_back(i);
204                 if (piv<0 || shift[i] < shift[piv])
205                     piv = i;
206             }
207         }
208
209         // if indices_nonzero is empty, const_residual is already zero, there
210         if (not indices_nonzero.empty())
211         {
212             // update all rows of appbas and residual in indices_nonzero exce
```

src/mat_lzz_pX_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

```
164     // order that remains to be dealt with
165     VecLong rem_order(order);
166
167     // indices of columns/orders that remain to be dealt with
168     VecLong rem_index(cdim);
169     std::iota(rem_index.begin(), rem_index.end(), 0);
170
171     // all along the algorithm, shift = shifted row degrees of approximant ba
172     // (initially, input shift = shifted row degree of the identity matrix)
173
174     while (not rem_order.empty())
175     {
176         /** Invariant:
177          *  - appbas is a shift-ordered weak Popov approximant basis for
178          *  (pmat,reached_order) where doneorder is the tuple such that
179          *  -->reached_order[j] + rem_order[j] == order[j] for j appearing in
180          *  -->reached_order[j] == order[j] for j not appearing in rem_index
```

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.

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

src/mat_lzz_pX_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`

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**

outline

- **problem and result**
- **previous work**
- **new ingredients**
- **conclusion | perspectives**

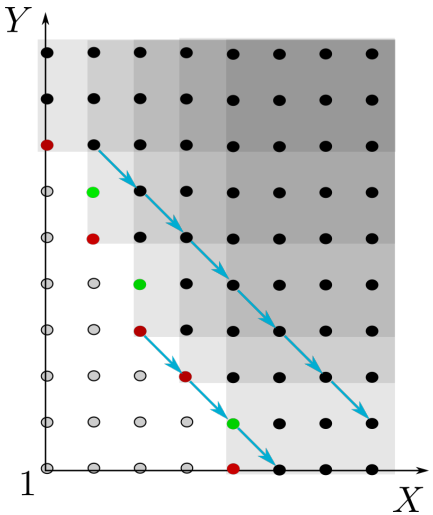
- problem: change of monomial order
- faster algorithm via univariate matrices
- improved software performances

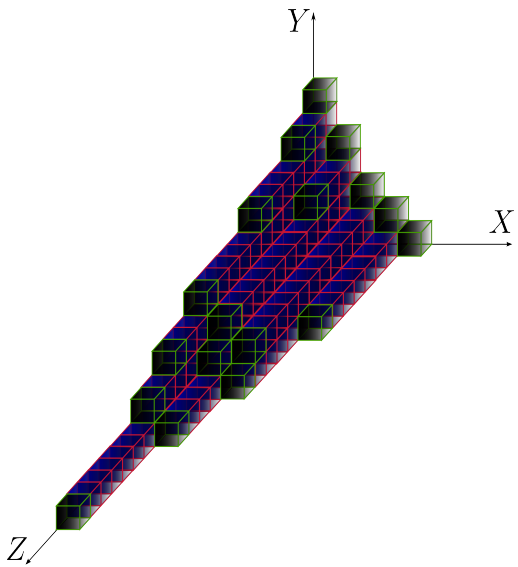
outline

- **problem and result**
  - problem: change of monomial order
  - faster algorithm via univariate matrices
  - improved software performances
- **previous work**
  - stability and multiplication matrix
  - shape position and lexicographic basis
  - previously: dense or sparse linear algebra
- **new ingredients**
- **conclusion | perspectives**

stability and multiplication matrix

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





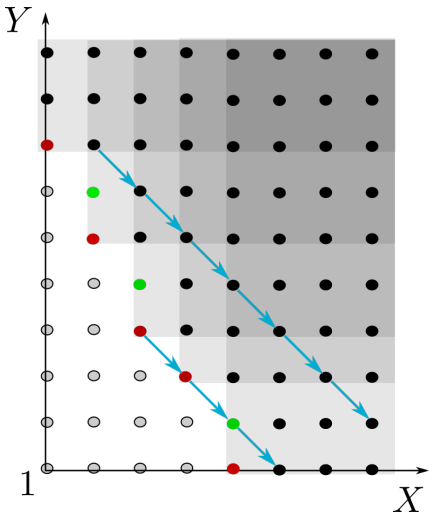
- related to classical notions of stability and of Borel-fixedness [Herzog-Hibi 2011, Galligo 1974, Bayer-Stillman 1987]
- easily verified: considering $\mu = \mathrm{lt}_\preccurlyeq(g)$ for $g \in \mathcal{G}$ is sufficient

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

- 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 $\mathbb{K}[x]/\mathcal{I}$, the representation of $\mathrm{lt}_\preccurlyeq(g)$ on $\mathcal{B}$ is $\mathrm{lt}_\preccurlyeq(g) - g$

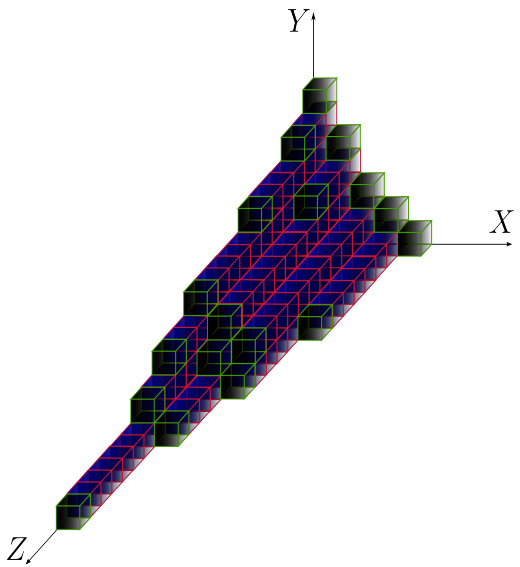
stability and multiplication matrix

x_n-stability: for any monomial $\mu \in \mathrm{lt}_\preccurlyeq(\mathcal{I})$ such that x_n divides μ, $\frac{x_i}{x_n}\mu \in \mathrm{lt}_\preccurlyeq(\mathcal{I})$ 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 $\mathrm{lt}_\preccurlyeq(\mathcal{I})$
$=$ vector space basis of $\mathbb{K}[x]/\mathcal{I}$



- 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 $\mathbb{K}[x]/\mathcal{I}$, the representation of $\mathrm{lt}_\preccurlyeq(g)$ on $\mathcal{B}$ is $\mathrm{lt}_\preccurlyeq(g) - g$

multiplication matrix $M_n \in \mathbb{K}^{D\times D}$ of x_n in $\mathbb{K}[x]/\mathcal{I}$
- 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 basis

[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 basis

[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 basis

[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 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 $\varepsilon_1, \varepsilon_1 M_n, \ldots, \varepsilon_1 M_n^D$
- the matrix $\begin{bmatrix} \varepsilon_1 \\ \varepsilon_1 M_n \\ \vdots \\ \varepsilon_1 M_n^{D-1} \end{bmatrix} \in \mathbb{K}^{D \times D}$ is invertible (taking $\varepsilon_1 = 1$)

$\Rightarrow$ $h(x_n)$ is the minpoly/charpoly of M_n

previously: dense or sparse linear algebra

using dense linear algebra

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

- compute matrix in $O(D^\omega \log(D))$

via matrix-matrix products [Keller-Gehrig 1985]

- nullspace computation in $O(D^\omega)$

[Ibarra-Hui-Moran 1982]

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

previously: dense or sparse linear algebra

using dense linear algebra

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

- compute matrix in $O(D^\omega \log(D))$

via matrix-matrix products [Keller-Gehrig 1985]

- nullspace computation in $O(D^\omega)$

[Ibarra-Hui-Moran 1982]

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

using sparse linear algebra [Wiedemann 1986]

for random column vector $r \in \mathbb{K}^{D\times 1}$,
scalar sequence $(\varepsilon_1 M_n^k r)_{0\leqslant k<2D}$
$\rightsquigarrow$ 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 $g_1, \ldots, g_{n-1}$ in $\tilde{O}(nD)$ via $n-1$

Hankel systems [Faugère-Mou 2011, 2017]

randomized algo in $O(tD^2)$

outline

- **problem and result**
  - problem: change of monomial order
  - faster algorithm via univariate matrices
  - improved software performances
- **previous work**
  - stability and multiplication matrix
  - shape position and lexicographic basis
  - previously: dense or sparse linear algebra
- **new ingredients**
- **conclusion | perspectives**

outline

problem and result

- problem: change of monomial order
- faster algorithm via univariate matrices
- improved software performances

previous work

- stability and multiplication matrix
- shape position and lexicographic basis
- previously: dense or sparse linear algebra

new ingredients

- paradigm shift: ~~sparse~~ $\rightarrow$ structured
- a univariate module with known basis
- from Hermite normal form to lex basis

conclusion | perspectives

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

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

$$\updownarrow$$

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

with charpoly$(M) = h$

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

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

- zero-dimensional with $D = 8$

  $\preccurlyeq_{\text{drl}}$-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~~ $\rightarrow$ structured

ideal $\mathcal{I} \subset \mathbb{F}_{29}[x_1, x_2, x_3]$ generated by the $\preccurlyeq_{\mathsf{drl}}$-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 $\mathcal{B} = (1, x_3, x_3^2, x_3^3, x_2, x_2x_3, x_1, x_1x_3)$

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

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

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

- $t = 3$, $D = 8$
- 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 $\mathbb{K}[x_1, x_2, x_3]/\mathcal{I} \longleftrightarrow \mathbb{K}[x_3]$-module structure

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

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

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

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

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

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

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

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

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

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

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

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

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

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

- $\det(P) = \text{charpoly}(M)$
- $\text{Smith}(P) \simeq \text{Frob}(M)$
- [Storjohann 2000]

[Pernet-Storjohann 2007]

- column degrees $(4, 2, 2)$

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

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

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

from Hermite normal form to lex basis

$$\begin{matrix} & & \mu_1 = 1 & & \mu_2 = x_2 & & \mu_3 = x_1 \\ & & \vdots & & \vdots & & \vdots \end{matrix}$$

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

from Hermite normal form to lex basis

$$\begin{matrix} & \mu_1 = 1 & & \mu_2 = x_2 & & \mu_3 = x_1 \\ & \vdots & & \vdots & & \vdots \end{matrix}$$

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

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

$$\begin{matrix} & \mu_1 = 1 & & \mu_2 = x_2 & & \mu_3 = x_1 \\ & \vdots & & \vdots & & \vdots \end{matrix}$$

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

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

$$\mu_1 = 1 \qquad \mu_2 = x_2 \qquad \mu_3 = x_1$$

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

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 $p_{i1}(x_3) + p_{i2}(x_3)x_2 + p_{i3}(x_3)x_1$

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

= lex basis

outline

problem and result

- problem: change of monomial order
- faster algorithm via univariate matrices
- improved software performances

previous work

- stability and multiplication matrix
- shape position and lexicographic basis
- previously: dense or sparse linear algebra

new ingredients

- paradigm shift: ~~sparse~~ $\rightarrow$ structured
- a univariate module with known basis
- from Hermite normal form to lex basis

conclusion | perspectives

- 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

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?