msolve: A Library for Solving Multivariate Polynomial Systems

MAX Seminar, 2021

Jérémy Berthomieu[1] Christian Eder [2] Vincent Neiger[1] Mohab Safey El Din[1]

[1]Sorbonne Université, Paris, France

[2]TU Kaiserslautern

Polynomial system solving

Let $\mathbb{K}$ be a field and $f_1, \ldots, f_s$ be polynomials in $\mathbb{K}[x_1, \ldots, x_n]$

"Solve" $f_1 = \cdots = f_s = 0 \rightsquigarrow$ solution set in $\mathbb{K}'^n$ where $\mathbb{K} \subset \mathbb{K}'$

Polynomial system solving

Let $\mathbb{K}$ be a field and $f_1, \ldots, f_s$ be polynomials in $\mathbb{K}[x_1, \ldots, x_n]$

"Solve" $f_1 = \cdots = f_s = 0 \rightsquigarrow$ solution set in $\mathbb{K}'^n$ where $\mathbb{K} \subset \mathbb{K}'$

- $\mathbb{K} = \mathbb{K}'$ is a prime field
- $\mathbb{K} = \mathbb{Q}$ and $\mathbb{K}' = \mathbb{R}$ or $\mathbb{K}' = \mathbb{C}$

Polynomial system solving

Let $\mathbb{K}$ be a field and $f_1, \ldots, f_s$ be polynomials in $\mathbb{K}[x_1, \ldots, x_n]$

"Solve" $f_1 = \cdots = f_s = 0 \rightsquigarrow$ solution set in $\mathbb{K}'^n$ where $\mathbb{K} \subset \mathbb{K}'$

- $\mathbb{K} = \mathbb{K}'$ is a prime field
- $\mathbb{K} = \mathbb{Q}$ and $\mathbb{K}' = \mathbb{R}$ or $\mathbb{K}' = \mathbb{C}$

- $\mathcal{NP}$-hardness of multivariate solving
- Bézout bound
- Zero vs positive dimensional systems

Polynomial system solving

Let $\mathbb{K}$ be a field and $f_1, \ldots, f_s$ be polynomials in $\mathbb{K}[x_1, \ldots, x_n]$

"Solve" $f_1 = \cdots = f_s = 0 \rightsquigarrow$ solution set in $\mathbb{K}'^n$ where $\mathbb{K} \subset \mathbb{K}'$

- $\mathbb{K} = \mathbb{K}'$ is a prime field
- $\mathbb{K} = \mathbb{Q}$ and $\mathbb{K}' = \mathbb{R}$ or $\mathbb{K}' = \mathbb{C}$

- $\mathcal{NP}$-hardness of multivariate solving
- Bézout bound
- Zero vs positive dimensional systems

Algorithmic backgrounds:

- Gröbner bases, resultants, Regular chains, geometric resolution
- Cylindrical Algebraic Decomposition, Critical Point Methods

Polynomial system solving

Let $\mathbb{K}$ be a field and $f_1, \ldots, f_s$ be polynomials in $\mathbb{K}[x_1, \ldots, x_n]$

"Solve" $f_1 = \cdots = f_s = 0 \rightsquigarrow$ solution set in $\mathbb{K}'^n$ where $\mathbb{K} \subset \mathbb{K}'$

- $\mathbb{K} = \mathbb{K}'$ is a prime field
- $\mathbb{K} = \mathbb{Q}$ and $\mathbb{K}' = \mathbb{R}$ or $\mathbb{K}' = \mathbb{C}$

- $\mathcal{NP}$-hardness of multivariate solving
- Bézout bound
- Zero vs positive dimensional systems

Algorithmic backgrounds:

- **Gröbner bases**, resultants, Regular chains, geometric resolution
- Cylindrical Algebraic Decomposition, Critical Point Methods

msolve's focus and features

Algorithmic background. **Gröbner bases**

msolve's focus and features

Algorithmic background. **Gröbner bases**

Restrictions

- $\mathbb{K} = \frac{\mathbb{Z}}{p\mathbb{Z}}$ for $p < 2^{31}$ or $\mathbb{K} = \mathbb{Q}$
- Computes Gröbner bases for $\mathbb{K} = \frac{\mathbb{Z}}{p\mathbb{Z}}$
- Focus on zero-dimensional systems (losing multiplicities)

$$w(t) = 0, \quad x_i = \frac{v_i(t)}{w'(t)}, \quad 1 \leq i \leq n$$

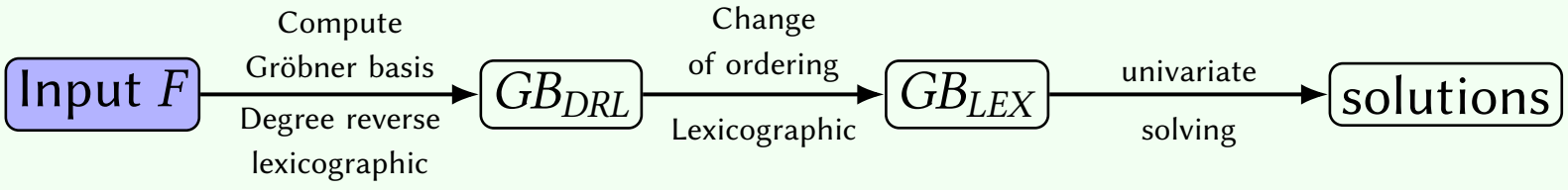
msolve's focus and features

Algorithmic background. **Gröbner bases**

Restrictions

- $\mathbb{K} = \frac{\mathbb{Z}}{p\mathbb{Z}}$ for $p < 2^{31}$ or $\mathbb{K} = \mathbb{Q}$
- Computes Gröbner bases for $\mathbb{K} = \frac{\mathbb{Z}}{p\mathbb{Z}}$
- Focus on zero-dimensional systems (losing multiplicities)

$$w(t) = 0, \quad x_i = \frac{v_i(t)}{w'(t)}, \quad 1 \leq i \leq n$$



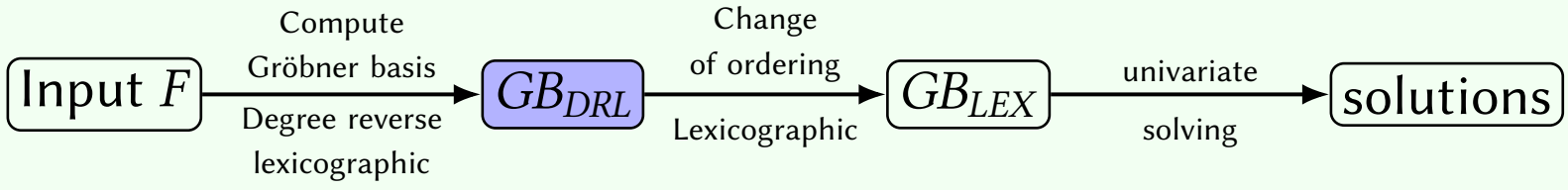
msolve's focus and features

Algorithmic background. **Gröbner bases**

Restrictions

- $\mathbb{K} = \frac{\mathbb{Z}}{p\mathbb{Z}}$ for $p < 2^{31}$ or $\mathbb{K} = \mathbb{Q}$
- Computes Gröbner bases for $\mathbb{K} = \frac{\mathbb{Z}}{p\mathbb{Z}}$
- Focus on zero-dimensional systems (losing multiplicities)

$$w(t) = 0, \quad x_i = \frac{v_i(t)}{w'(t)}, \quad 1 \leq i \leq n$$



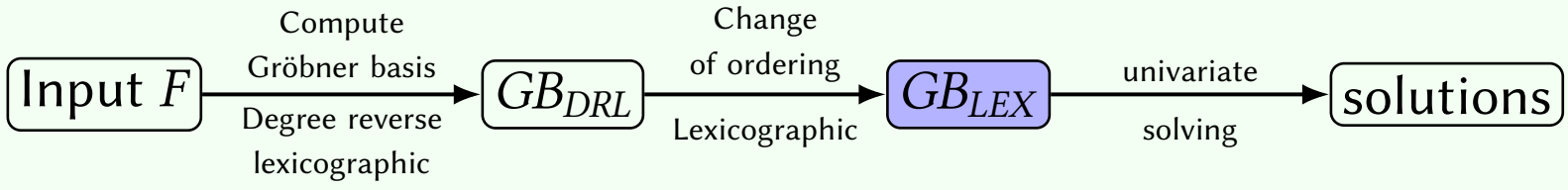
msolve's focus and features

Algorithmic background. **Gröbner bases**

Restrictions

- $\mathbb{K} = \frac{\mathbb{Z}}{p\mathbb{Z}}$ for $p < 2^{31}$ or $\mathbb{K} = \mathbb{Q}$
- Computes Gröbner bases for $\mathbb{K} = \frac{\mathbb{Z}}{p\mathbb{Z}}$
- Focus on zero-dimensional systems (losing multiplicities)

$$w(t) = 0, \quad x_i = \frac{v_i(t)}{w'(t)}, \quad 1 \leq i \leq n$$



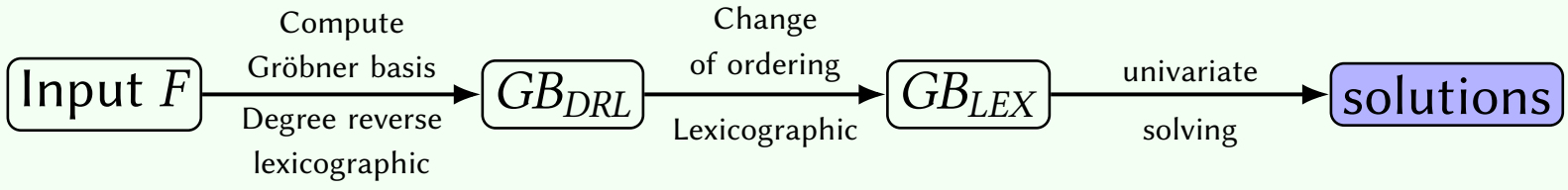
msolve's focus and features

Algorithmic background. **Gröbner bases**

Restrictions

- $\mathbb{K} = \frac{\mathbb{Z}}{p\mathbb{Z}}$ for $p < 2^{31}$ or $\mathbb{K} = \mathbb{Q}$
- Computes Gröbner bases for $\mathbb{K} = \frac{\mathbb{Z}}{p\mathbb{Z}}$
- Focus on zero-dimensional systems (losing multiplicities)

$$w(t) = 0, \quad x_i = \frac{v_i(t)}{w'(t)}, \quad 1 \leq i \leq n$$



msolve's goals

- Provide **up-to-date implementations** of Gröbner bases algorithms

> This includes Faugère's F4 and F5 algorithms, change of ordering algorithms, multi-modular strategies, Hensel liftings, etc.

msolve's goals

- Provide **up-to-date implementations** of Gröbner bases algorithms

  This includes Faugère's F4 and F5 algorithms, change of ordering algorithms, multi-modular strategies, Hensel liftings, etc.

- A tool for **fundamental research**

  $\rightsquigarrow$ open source library ; counters for arithmetic operations

msolve's goals

- Provide **up-to-date implementations** of Gröbner bases algorithms

  This includes Faugère's F4 and F5 algorithms, change of ordering algorithms, multi-modular strategies, Hensel liftings, etc.

- A tool for **fundamental research**

  $\rightsquigarrow$ open source library ; counters for arithmetic operations

- A tool for **solving** multivariate systems

  $\rightsquigarrow$ performance issues, applications, etc.

msolve's goals

- Provide **up-to-date implementations** of Gröbner bases algorithms

  This includes Faugère's F4 and F5 algorithms, change of ordering algorithms, multi-modular strategies, Hensel liftings, etc.

- A tool for **fundamental research**

  $\rightsquigarrow$ open source library ; counters for arithmetic operations

- A tool for **solving** multivariate systems

  $\rightsquigarrow$ performance issues, applications, etc.

- Clearer view of current challenges e.g. current ratio of time between GB_{DRL} vs GB_{LEX} ?

msolve's goals

- Provide **up-to-date implementations** of Gröbner bases algorithms

  This includes Faugère's F4 and F5 algorithms, change of ordering algorithms, multi-modular strategies, Hensel liftings, etc.

- A tool for **fundamental research**

  $\rightsquigarrow$ open source library ; counters for arithmetic operations

- A tool for **solving** multivariate systems

  $\rightsquigarrow$ performance issues, applications, etc.

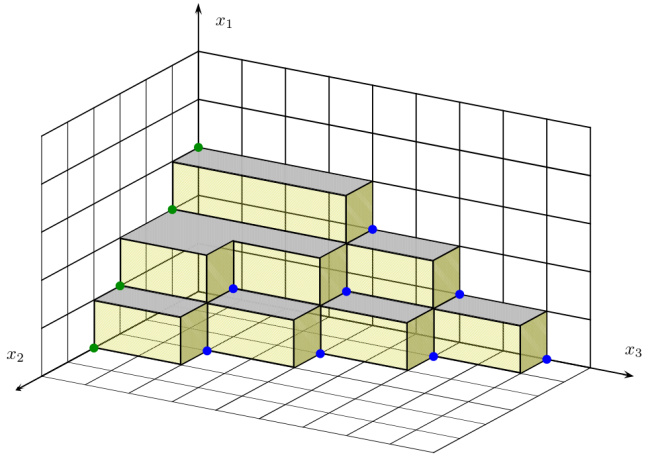
- Clearer view of current challenges e.g. current ratio of time between GB_{DRL} vs GB_{LEX} ?

- **New algorithms**

Gröbner bases

$I = \langle f_1, \ldots, f_s \rangle \subset \mathbb{K}[x_1, \ldots, x_n]$, $\succ$ a monomial order, leading monomials

A finite set $G \subset I$ is a Gröbner basis of $(I, \succ)$ iff $\langle \mathrm{LM}_\succ(G) \rangle = \langle \mathrm{LM}_\succ(I) \rangle$

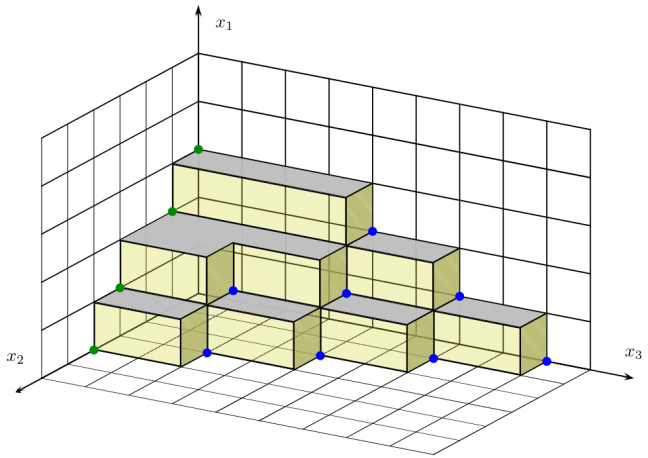


- Combinatorial structure of polynomial ideals
- Basis $\mathscr{B}$ of quotient ring $\frac{\mathbb{K}[x_1,\ldots,x_n]}{I}$
- Generic staircase **Moreno-Socías**
- $\mathscr{B}$ is **finite** iff $\dim(I) \leq 0$

Gröbner bases

$I = \langle f_1, \ldots, f_s \rangle \subset \mathbb{K}[x_1, \ldots, x_n]$, $\quad \succ$ a monomial order, leading monomials

A finite set $G \subset I$ is a Gröbner basis of $(I, \succ)$ iff $\langle \mathrm{LM}_\succ(G) \rangle = \langle \mathrm{LM}_\succ(I) \rangle$



- Combinatorial structure of polynomial ideals
- Basis $\mathscr{B}$ of quotient ring $\frac{\mathbb{K}[x_1,\ldots,x_n]}{I}$
- Generic staircase **Moreno-Socías**
- $\mathscr{B}$ is **finite** iff $\dim(I) \leq 0$

$$S_\succ(g,f) = \frac{\mathrm{lcm}(\mathrm{LM}_\succ(f),\mathrm{LM}_\succ(g))}{\mathrm{LM}_\succ(g)} g - \frac{\mathrm{lcm}(\mathrm{LM}_\succ(f),\mathrm{LM}_\succ(g))}{\mathrm{LM}_\succ(f)} f$$

Buchberger's criterion

Let $G = \{g_1, \ldots, g_k\}$; G is a GB iff

$$\mathrm{NormalForm}_\succ(S_\succ(g_i, g_j), G) = 0 \text{ for all } (i,j).$$

Faugère's F4 algorithm in msolve

Input. $F = (f_1, \ldots, f_s) \subset \mathbb{K}[x_1, \ldots, x_n], \succ$

Output. Gröbner basis G for $F, \succ$

- $G \leftarrow \{f_1, \ldots, f_s\}$
- $P \leftarrow \{\text{pairs}(f_i, f_j) \mid 1 \leq i < j \leq m\}$
- While $(P \neq \emptyset)$
  - **choose** $S \subset P, P \leftarrow P \setminus S$
  - $M \leftarrow$ **symbolic preprocess** (S, G)
  - $L \leftarrow$ **linear algebra** (M)
  - For $f \in L$ with $\text{LM}_\succ(f) \notin \text{LM}_\succ(G)$
    - $P \leftarrow P \cup \{(f, g) \mid g \in G\}$
    - $G \leftarrow G \cup \{f\}$
- **return** G

Faugère's F4 algorithm in msolve

$\succ$ = *DRL* (grevlex)

Input. $F = (f_1, \ldots, f_s) \subset \mathbb{K}[x_1, \ldots, x_n], \succ$

Output. Gröbner basis G for $F, \succ$

- $G \leftarrow \{f_1, \ldots, f_s\}$
- $P \leftarrow \{\text{pairs}(f_i, f_j) \mid 1 \leq i < j \leq m\}$
- While $(P \neq \emptyset)$
  - **choose** $S \subset P, P \leftarrow P \setminus S$
  - $M \leftarrow$ **symbolic preprocess** (S, G)
  - $L \leftarrow$ **linear algebra** (M)
  - For $f \in L$ with $\text{LM}_\succ(f) \notin \text{LM}_\succ(G)$
    - $P \leftarrow P \cup \{(f, g) \mid g \in G\}$
    - $G \leftarrow G \cup \{f\}$
- **return** G

Faugère's F4 algorithm in msolve

$\succ$ = *DRL* (grevlex)

Input. $F = (f_1, \ldots, f_s) \subset \mathbb{K}[x_1, \ldots, x_n], \succ$

Output. Gröbner basis G for $F, \succ$

- $G \leftarrow \{f_1, \ldots, f_s\}$
- $P \leftarrow \{\text{pairs}(f_i, f_j) \mid 1 \leq i < j \leq m\}$
- While $(P \neq \emptyset)$
  - **choose** $S \subset P, P \leftarrow P \setminus S$
  - $M \leftarrow$ **symbolic preprocess** (S, G)
  - $L \leftarrow$ **linear algebra** (M)
  - For $f \in L$ with $\text{LM}_\succ(f) \notin \text{LM}_\succ(G)$
    - $P \leftarrow P \cup \{(f, g) \mid g \in G\}$
    - $G \leftarrow G \cup \{f\}$
- **return** G

Time complexity: $O(N \dim^\omega)$

$N =$ number of matrices

$\dim =$ max. size of matrices

$\omega =$ matrix mult. exponent

Generic case. $N = n$, $\dim = \binom{n+\mathbb{D}}{n}$

with $\mathbb{D} = \sum_i (D_i - 1)$

Faugère's F4 algorithm in msolve

$\succ$ = *DRL* (grevlex)

Input. $F = (f_1, \ldots, f_s) \subset \mathbb{K}[x_1, \ldots, x_n], \succ$

Output. Gröbner basis G for $F, \succ$

- $G \leftarrow \{f_1, \ldots, f_s\}$
- $P \leftarrow \{\text{pairs}(f_i, f_j) \mid 1 \leq i < j \leq m\}$
- While $(P \neq \emptyset)$
  - **choose** $S \subset P, P \leftarrow P \setminus S$
  - $M \leftarrow$ **symbolic preprocess** (S, G)
  - $L \leftarrow$ **linear algebra** (M)
  - For $f \in L$ with $\text{LM}_\succ(f) \notin \text{LM}_\succ(G)$
    - $P \leftarrow P \cup \{(f, g) \mid g \in G\}$
    - $G \leftarrow G \cup \{f\}$
- **return** G

Time complexity: $O(N \dim^\omega)$

N = number of matrices

$\dim$ = max. size of matrices

ω = matrix mult. exponent

Generic case. $N = n$, $\dim = \binom{n+\mathbb{D}}{n}$ with $\mathbb{D} = \sum_i (D_i - 1)$

Selection strategy: degree wise

Faugère's F4 algorithm in msolve

$\succ$ = *DRL* (grevlex)

Input. $F = (f_1, \ldots, f_s) \subset \mathbb{K}[x_1, \ldots, x_n], \succ$

Output. Gröbner basis G for $F, \succ$

- $G \leftarrow \{f_1, \ldots, f_s\}$
- $P \leftarrow \{\text{pairs}(f_i, f_j) \mid 1 \leq i < j \leq m\}$
- While $(P \neq \emptyset)$
  - **choose** $S \subset P, P \leftarrow P \setminus S$
  - $M \leftarrow$ **symbolic preprocess** (S, G)
  - $L \leftarrow$ **linear algebra** (M)
  - For $f \in L$ with $\text{LM}_\succ(f) \notin \text{LM}_\succ(G)$
    - $P \leftarrow P \cup \{(f, g) \mid g \in G\}$
    - $G \leftarrow G \cup \{f\}$
- **return** G

Time complexity: $O\left(N \dim^\omega\right)$

N = number of matrices

$\dim$ = max. size of matrices

ω = matrix mult. exponent

Generic case. $N = n$, $\dim = \binom{n+\mathbb{D}}{n}$ with $\mathbb{D} = \sum_i (D_i - 1)$

Selection strategy: degree wise

Bottlenecks.

✓ Memory usage

Two hash tables (basis + pre-process)

Faugère's F4 algorithm in msolve

$\succ$ = *DRL* (grevlex)

Input. $F = (f_1, \ldots, f_s) \subset \mathbb{K}[x_1, \ldots, x_n], \succ$

Output. Gröbner basis G for $F, \succ$

- $G \leftarrow \{f_1, \ldots, f_s\}$
- $P \leftarrow \{\text{pairs}(f_i, f_j) \mid 1 \leq i < j \leq m\}$
- While $(P \neq \emptyset)$
  - **choose** $S \subset P, P \leftarrow P \setminus S$
  - $M \leftarrow$ **symbolic preprocess** (S, G)
  - $L \leftarrow$ **linear algebra** (M)
  - For $f \in L$ with $\text{LM}_\succ(f) \notin \text{LM}_\succ(G)$
    - $P \leftarrow P \cup \{(f, g) \mid g \in G\}$
    - $G \leftarrow G \cup \{f\}$
- **return** G

Time complexity: $O\left(N \dim^\omega\right)$

N = number of matrices

$\dim$ = max. size of matrices

ω = matrix mult. exponent

Generic case. $N = n$, $\dim = \binom{n+\mathbb{D}}{n}$ with $\mathbb{D} = \sum_i (D_i - 1)$

Selection strategy: degree wise

Bottlenecks.

✓ Memory usage

Two hash tables (basis + pre-process)

✓ Fast divisibility check

Use of short divisor mask test

Faugère's F4 algorithm in msolve

$\succ$ = *DRL* (grevlex)

Input. $F = (f_1, \ldots, f_s) \subset \mathbb{K}[x_1, \ldots, x_n], \succ$

Output. Gröbner basis G for $F, \succ$

- $G \leftarrow \{f_1, \ldots, f_s\}$
- $P \leftarrow \{\text{pairs}(f_i, f_j) \mid 1 \leq i < j \leq m\}$
- While $(P \neq \emptyset)$
  - **choose** $S \subset P, P \leftarrow P \setminus S$
  - $M \leftarrow$ **symbolic preprocess** (S, G)
  - $L \leftarrow$ **linear algebra** (M)
  - For $f \in L$ with $\text{LM}_\succ(f) \notin \text{LM}_\succ(G)$
    - $P \leftarrow P \cup \{(f, g) \mid g \in G\}$
    - $G \leftarrow G \cup \{f\}$
- **return** G

Time complexity: $O(N \dim^\omega)$

N = number of matrices

$\dim$ = max. size of matrices

ω = matrix mult. exponent

Generic case. $N = n$, $\dim = \binom{n+\mathbb{D}}{n}$ with $\mathbb{D} = \sum_i (D_i - 1)$

Selection strategy: degree wise

Bottlenecks.

✓ Memory usage

Two hash tables (basis + pre-process)

✓ Fast divisibility check

Use of short divisor mask test

✓ Linear algebra

sparse / dense + sparse (+ probabilistic variants)

7−, 15− and 31− bit AVX2 implementations

Faugère's F4 algorithm in msolve

$\succ$ = *DRL* (grevlex)

Input. $F = (f_1, \ldots, f_s) \subset \mathbb{K}[x_1, \ldots, x_n], \succ$

Output. Gröbner basis G for $F, \succ$

- $G \leftarrow \{f_1, \ldots, f_s\}$
- $P \leftarrow \{\text{pairs}(f_i, f_j) \mid 1 \leq i < j \leq m\}$
- While $(P \neq \emptyset)$
  - **choose** $S \subset P, P \leftarrow P \setminus S$
  - $M \leftarrow$ **symbolic preprocess** (S, G)
  - $L \leftarrow$ **linear algebra** (M)
  - For $f \in L$ with $\text{LM}_\succ(f) \notin \text{LM}_\succ(G)$
    - $P \leftarrow P \cup \{(f, g) \mid g \in G\}$
    - $G \leftarrow G \cup \{f\}$
- **return** G

Time complexity: $O(N \dim^\omega)$

N = number of matrices

$\dim$ = max. size of matrices

ω = matrix mult. exponent

Generic case. $N = n$, $\dim = \binom{n+\mathbb{D}}{n}$ with $\mathbb{D} = \sum_i (D_i - 1)$

Selection strategy: degree wise

Bottlenecks.

✓ Memory usage

Two hash tables (basis + pre-process)

✓ Fast divisibility check

Use of short divisor mask test

✓ Linear algebra

sparse / dense + sparse (+ probabilistic variants)

7−, 15− and 31− bit AVX2 implementations

Faugère, Steele

Monagan/Pearce

msolve linear algebra

Katsura-n
regular systems
2^n solutions

Eco-n
irregular systems
2^{n-1} solutions

Examples	-l 1	-l 2	prob -l 1	prob -l 2
Katsura-10	1.23	0.6	0.4	0.27
Katsura-11	6.17	3.8	1.6	1.1
Katsura-12	44.6	29.6	7	5.5
Katsura-13	385.7	289	43.5	37
Katsura-14	3 545.5	2 594	303	231
Katsura-15	29 988	22 704	2123	1 694
Katsura-16	259 240	202 927	16 275	12 474
Katsura-17	2 323 299	1 789 828	124 310	101 869

Timings (seconds) for 31-bit prime

msolve linear algebra

Katsura-n
regular systems
2^n solutions

Eco-n
irregular systems
2^{n-1} solutions

Examples	-l 1	-l 2	prob -l 1	prob -l 2
Katsura-10	1.23	0.6	0.4	0.27
Katsura-11	6.17	3.8	1.6	1.1
Katsura-12	44.6	29.6	7	5.5
Katsura-13	385.7	289	43.5	37
Katsura-14	3 545.5	2 594	303	231
Katsura-15	29 988	22 704	2123	1 694
Katsura-16	259 240	202 927	16 275	12 474
Katsura-17	2 323 299	1 789 828	124 310	101 869
Eco-10	0.26	0.32	0.1	0.1
Eco-11	1.45	1.37	0.4	0.6
Eco-12	9.95	7.2	2.3	1.7
Eco-13	65	46	10.7	9.2
Eco-14	528	370	59	47.6
Eco-15	3 842	2 800	373	294
Eco-16	60 994	50 395	4 974	4 582
Eco-17	543 286	482 634	52 690	48 080

Timings (seconds) for 31-bit prime

msolve linear algebra

Examples	-l 1	-l 2	prob -l 1	prob -l 2
Katsura-10	1.23	0.6	0.4	0.27
Katsura-11	6.17	3.8	1.6	1.1
Katsura-12	44.6	29.6	7	5.5
Katsura-13	385.7	289	43.5	37
Katsura-14	3 545.5	2 594	303	231
Katsura-15	29 988	22 704	2123	1 694
Katsura-16	259 240	202 927	16 275	12 474
Katsura-17	2 323 299	1 789 828	124 310	101 869
Eco-10	0.26	0.32	0.1	0.1
Eco-11	1.45	1.37	0.4	0.6
Eco-12	9.95	7.2	2.3	1.7
Eco-13	65	46	10.7	9.2
Eco-14	528	370	59	47.6
Eco-15	3 842	2 800	373	294
				4 582
				48 080

Katsura-n
regular systems
2^n solutions

Eco-n
irregular systems
2^{n-1} solutions

Sparse linear algebra seems competitive and
the probabilistic one looks promising

me

msolve linear algebra

Examples	-l 1	-l 2	prob -l 1	prob -l 2
Katsura-10	1.23	0.6	0.4	0.27
Katsura-11	6.17	3.8	1.6	1.1
Katsura-12	44.6	29.6	7	5.5
Katsura-13	385.7	289	43.5	37
Katsura-14	3 545.5	2 594	303	231
Katsura-15	29 988	22 704	2123	1 694
Katsura-16	259 240	202 927	16 275	12 474
Katsura-17	2 323 299	1 789 828	124 310	101 869
Eco-10	0.26	0.32	0.1	0.1
Eco-11	1.45	1.37	0.4	0.6
				1.7
				9.2
Eco-14	328	370	39	47.6
Eco-15	3 842	2 800	373	294
				4 582
				48 080

Katsura-n
regular systems
2^n solutions

2^{n-1} solutions

👍 Linear algebra never took less than 90% of the runtime

Sparse linear algebra seems competitive and
the probabilistic one looks promising

me

F4 tracer

Gröbner trace algorithm (Traverso, ISSAC'88)

Learn from a first modular computation reductions to zero

F4 tracer

Gröbner trace algorithm (Traverso, ISSAC'88)

Learn from a first modular computation reductions to zero

Examples	`msolve` F4 learn	`msolve` F4 tracer	speed-up (learn/tracer)	`msolve` prob	speed-up (prob / tracer)
Katsura-9	0.17	0.03	5.67	0.06	2
Katsura-10	0.81	0.09	9	0.24	2.67
Katsura-11	6.26	0.45	13.9	1.34	2.98
Katsura-12	56.1	3.10	18.1	8.61	2.78
Katsura-13	425	19	22.4	53	2.79
Katsura-14	3336	128	26.1	318	2.5
Katsura-15	27960	1000	27.96	2209	2.2
Katsura-16	259240	7518	34.5	12474	1.66

F4 tracer

Gröbner trace algorithm (Traverso, ISSAC'88)

Learn from a first modular computation reductions to zero

Examples	`msolve` F4 learn	`msolve` F4 tracer	speed-up (learn/tracer)	`msolve` prob	speed-up (prob / tracer)
Katsura-9	0.17	0.03	5.67	0.06	2
Katsura-10	0.81	0.09	9	0.24	2.67
Katsura-11	6.26	0.45	13.9	1.34	2.98
Katsura-12	56.1	3.10	18.1	8.61	2.78
Katsura-13	425	19	22.4	53	2.79
Katsura-14	3336	128	26.1	318	2.5
Katsura-15	27960	1000	27.96	2209	2.2
Katsura-16	259240	7518	34.5	12474	1.66
Eco-10	0.28	0.05	5.6	0.1	2
Eco-11	1.21	0.17	7.11	0.39	2.29
Eco-12	11.6	1.1	10.54	2.25	2.05
Eco-13	67.3	6.6	10.2	11.7	1.77
Eco-14	516	34.8	14.8	67	1.92
Eco-15	3476	153	22.7	466.15	3

F4 tracer

Gröbner trace algorithm (Traverso, ISSAC'88)

Learn from a first modular computation reductions to zero

Examples	msolve F4 learn	msolve F4 tracer	speed-up (learn/tracer)	msolve prob	speed-up (prob / tracer)
Katsura-9	0.17	0.03	5.67	0.06	2
Katsura-10	0.81	0.09	9	0.24	2.67
Katsura-11	6.26	0.45	13.9	1.34	2.98
Katsura-12	56.1	3.10	18.1	8.61	2.78
Katsura-13	425	19	22.4	53	2.79
Katsura-14	3336	128	26.1	318	2.5
Katsura-15	27960	1000	27.96	2209	2.2
Katsura-16	259240	7518	34.5	12474	1.66
Eco-10	0.28	0.05	5.6	0.1	2
Eco-11	1.21	0.17	7.11	0.39	2.29
Eco-12	11.6	1.1	10.54	2.25	2.05
Eco-13	67.3	6.6	10.2	11.7	1.77
Eco-14	516	34.8	14.8	67	1.92
Eco-15	3476	153	22.7	466.15	3
Henrion-6	0.22	0.07	3.14	0.11	1.57
Henrion-7	27.5	6.5	4.23	9.55	1.47

F4 tracer

Gröbner trace algorithm (Traverso, ISSAC'88)

Learn from a first modular computation reductions to zero

Examples	msolve F4 learn	msolve F4 tracer	speed-up (learn/tracer)	msolve prob	speed-up (prob / tracer)
Katsura-9	0.17	0.03	5.67	0.06	2
Katsura-10	0.81	0.09	9	0.24	2.67
Katsura-11	6.26	0.45	13.9	1.34	2.98
Katsura-12	56.1	3.10	18.1	8.61	2.78
Katsura-13	425	19	22.4	53	2.79
Katsura-14	3336	128	26.1	318	2.5
Katsura-15	27960	1000	27.96	2209	2.2
Katsura-16	259240	7518	34.5	12474	1.66
Eco-10	0.28	0.05	5.6	0.1	2
Eco-11	1.21	0.17	7.11	0.39	2.29
Eco-12	11.6	1.1	10.54	2.25	2.05
Eco-13	67.3	6.6	10.2	11.7	1.77
Eco-14	516	34.8	14.8	67	1.92
Eco-15	3476	153	22.7	466.15	3
Henrion-6	0.22	0.07	3.14	0.11	1.57
Henrion-7	27.5	6.5	4.23	9.55	1.47
CP(3,6,2)	0.6	0.12	5	0.22	1.83
CP(3,7,2)	8.18	1.23	6.65	1.97	1.6
CP(3,8,2)	111.5	12.6	8.85	18.5	1.47

F4 tracer

Gröbner trace algorithm (Traverso, ISSAC'88)

Learn from a first modular computation reductions to zero

Examples	msolve F4 learn	msolve F4 tracer	speed-up (learn/tracer)	msolve prob	speed-up (prob / tracer)
Katsura-9	0.17	0.03	5.67	0.06	2
Katsura-10	0.81	0.09	9	0.24	2.67
Katsura-11	6.26	0.45	13.9	1.34	2.98
Katsura-12	56.1	3.10	18.1	8.61	2.78
Katsura-13	425	19	22.4	53	2.79
Katsura-14	3336	128	26.1	318	2.5
Katsura-15	27960	1000	27.96	2209	2.2
Katsura-16	259240	7518	34.5	12474	1.66
					2
					2.29
					2.05
					1.77
					1.92
Eco-15	3476	153	22.7	466.15	3
Henrion-6	0.22	0.07	3.14	0.11	1.57
Henrion-7	27.5	6.5	4.23	9.55	1.47
CP(3,6,2)	0.6	0.12	5	0.22	1.83
CP(3,7,2)	8.18	1.23	6.65	1.97	1.6
CP(3,8,2)	111.5	12.6	8.85	18.5	1.47

👍 The trace algorithm (adapted to F4) looks more promising than the probabilistic linear algebra

Modular F4 timings

Magma 2.23-8 timings on Katsura-11

(Xeon(R) E7-4850 v3 @ 2.20GHz)

- 16 to 23-bit primes: 1.9 secs
- 24-bit primes: 48.9 secs
- 31-bit primes: 244 secs

Maple 2020 timings on Katsura-11

(Xeon(R) W-10885M @ 2.40GHz)

- 16 to 31-bit primes: 0.66 secs

Modular F4 timings

Examples	`msolve` F4 (learn)	`msolve` F4 (tracer)	`maple`

Magma 2.23-8 timings on Katsura-11
(Xeon(R) E7-4850 v3 @ 2.20GHz)

- 16 to 23-bit primes: 1.9 secs
- 24-bit primes: 48.9 secs
- 31-bit primes: 244 secs

Maple 2020 timings on Katsura-11
(Xeon(R) W-10885M @ 2.40GHz)

- 16 to 31-bit primes: 0.66 secs

Timings (seconds) for 31-bit primes

Modular F4 timings

Magma 2.23-8 timings on Katsura-11
(Xeon(R) E7-4850 v3 @ 2.20GHz)

- 16 to 23-bit primes: 1.9 secs
- 24-bit primes: 48.9 secs
- 31-bit primes: 244 secs

Maple 2020 timings on Katsura-11
(Xeon(R) W-10885M @ 2.40GHz)

- 16 to 31-bit primes: 0.66 secs

Examples	`msolve` F4 (learn)	`msolve` F4 (tracer)	`maple`
Katsura-9	0.17	0.03	0.1
Katsura-10	0.81	0.09	0.36
Katsura-11	6.26	0.45	1.82
Katsura-12	56.1	3.1	8.5
Katsura-13	425	18.9	60.9
Katsura-14	3336	128	393

Timings (seconds) for 31-bit primes

Modular F4 timings

Magma 2.23-8 timings on Katsura-11
(Xeon(R) E7-4850 v3 @ 2.20GHz)

- 16 to 23-bit primes: 1.9 secs
- 24-bit primes: 48.9 secs
- 31-bit primes: 244 secs

Maple 2020 timings on Katsura-11
(Xeon(R) W-10885M @ 2.40GHz)

- 16 to 31-bit primes: 0.66 secs

Examples	`msolve` F4 (learn)	`msolve` F4 (tracer)	`maple`
Katsura-9	0.17	0.03	0.1
Katsura-10	0.81	0.09	0.36
Katsura-11	6.26	0.45	1.82
Katsura-12	56.1	3.1	8.5
Katsura-13	425	18.9	60.9
Katsura-14	3336	128	393
Eco-10	0.28	0.05	0.14
Eco-11	1.21	0.17	0.56
Eco-12	11.6	1.07	2.97
Eco-13	67.3	6.61	15.1
Eco-14	516	34.8	104.8

Timings (seconds) for 31-bit primes

Modular F4 timings

Magma 2.23-8 timings on Katsura-11
(Xeon(R) E7-4850 v3 @ 2.20GHz)

- 16 to 23-bit primes: 1.9 secs
- 24-bit primes: 48.9 secs
- 31-bit primes: 244 secs

Maple 2020 timings on Katsura-11
(Xeon(R) W-10885M @ 2.40GHz)

- 16 to 31-bit primes: 0.66 secs

Examples	`msolve` F4 (learn)	`msolve` F4 (tracer)	`maple`
Katsura-9	0.17	0.03	0.1
Katsura-10	0.81	0.09	0.36
Katsura-11	6.26	0.45	1.82
Katsura-12	56.1	3.1	8.5
Katsura-13	425	18.9	60.9
Katsura-14	3336	128	393
Eco-10	0.28	0.05	0.14
Eco-11	1.21	0.17	0.56
Eco-12	11.6	1.07	2.97
Eco-13	67.3	6.61	15.1
Eco-14	516	34.8	104.8
Henrion-6	0.22	0.07	0.17
Henrion-7	27.5	6.51	12.8

Timings (seconds) for 31-bit primes

Modular F4 timings

Magma 2.23-8 timings on Katsura-11
(Xeon(R) E7-4850 v3 @ 2.20GHz)

- 16 to 23-bit primes: 1.9 secs
- 24-bit primes: 48.9 secs
- 31-bit primes: 244 secs

Maple 2020 timings on Katsura-11
(Xeon(R) W-10885M @ 2.40GHz)

- 16 to 31-bit primes: 0.66 secs

Examples	`msolve` F4 (learn)	`msolve` F4 (tracer)	`maple`
Katsura-9	0.17	0.03	0.1
Katsura-10	0.81	0.09	0.36
Katsura-11	6.26	0.45	1.82
Katsura-12	56.1	3.1	8.5
Katsura-13	425	18.9	60.9
Katsura-14	3336	128	393
Eco-10	0.28	0.05	0.14
Eco-11	1.21	0.17	0.56
Eco-12	11.6	1.07	2.97
Eco-13	67.3	6.61	15.1
Eco-14	516	34.8	104.8
Henrion-6	0.22	0.07	0.17
Henrion-7	27.5	6.51	12.8
Noon-7	5.3	0.93	1.97
Noon-8	153	17.5	32.4
Phuoc-1	4.65	3.42	4.6

Timings (seconds) for 31-bit primes

Modular F4 timings

Magma 2.23-8 timings on Katsura-11
(Xeon(R) E7-4850 v3 @ 2.20GHz)

- 16 to 23-bit primes: 1.9 secs
- 24-bit primes: 48.9 secs
- 31-bit primes: 244 secs

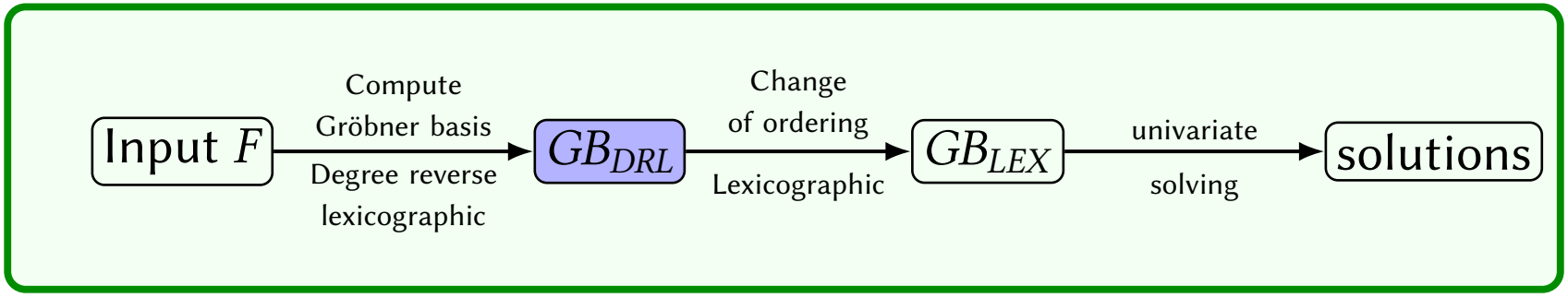
Maple 2020 timings on Katsura-11
(Xeon(R) W-10885M @ 2.40GHz)

- 16 to 31-bit primes: 0.66 secs

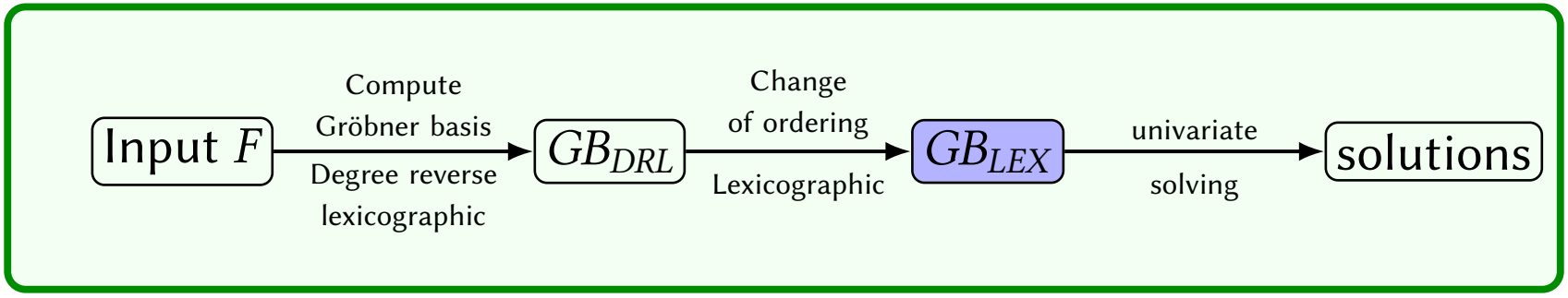
Examples	`msolve` F4 (learn)	`msolve` F4 (tracer)	`maple`
Katsura-9	0.17	0.03	0.1
Katsura-10	0.81	0.09	0.36
Katsura-11	6.26	0.45	1.82
Katsura-12	56.1	3.1	8.5
Katsura-13	425	18.9	60.9
Katsura-14	3336	128	393
Eco-10	0.28	0.05	0.14
Eco-11	1.21	0.17	0.56
Eco-12	11.6	1.07	2.97
Eco-13	67.3	6.61	15.1
Eco-14	516	34.8	104.8
Henrion-6	0.22	0.07	0.17
Henrion-7	27.5	6.51	12.8
Noon-7	5.3	0.93	1.97
Noon-8	153	17.5	32.4
Phuoc-1	4.65	3.42	4.6
CP(3,6,2)	0.59	0.12	0.31
CP(3,7,2)	8.18	1.23	2.78
CP(3,8,2)	111.5	12.2	24.6

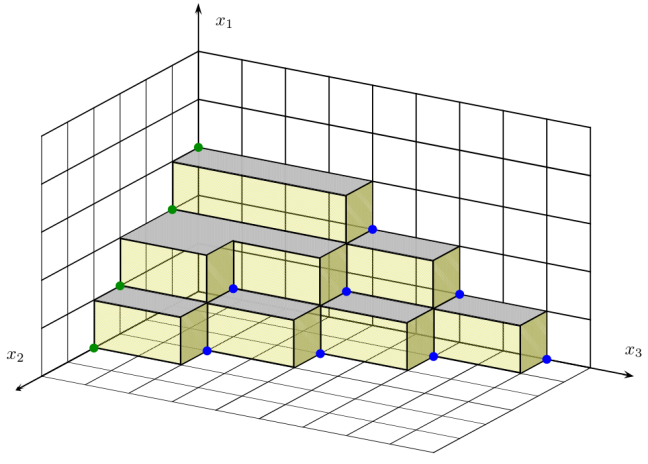
Timings (seconds) for 31-bit primes

Change of monomial orders



Change of monomial orders



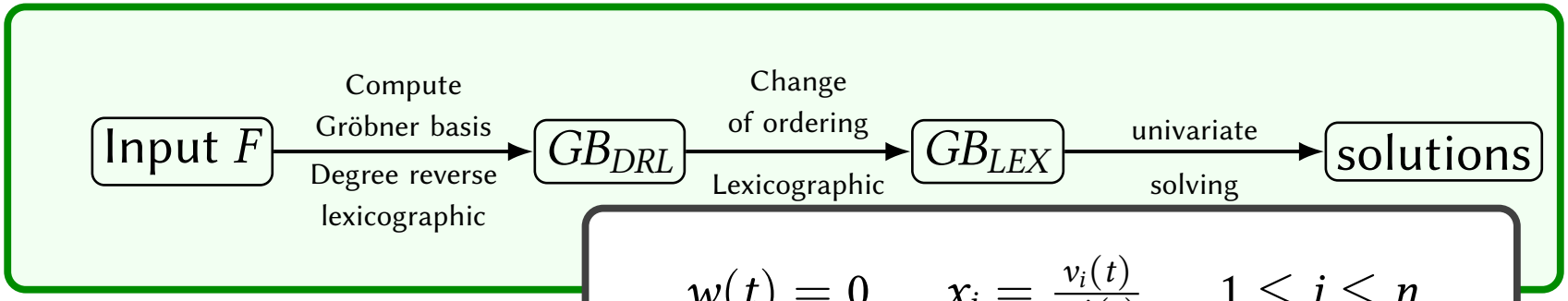


- Basis $\mathscr{B}$ of $\mathbb{A} = \frac{\mathbb{K}[x_1,\ldots,x_n]}{\langle F \rangle}$ is finite. FGLM algo.
- Shape position for radical ideals is generic

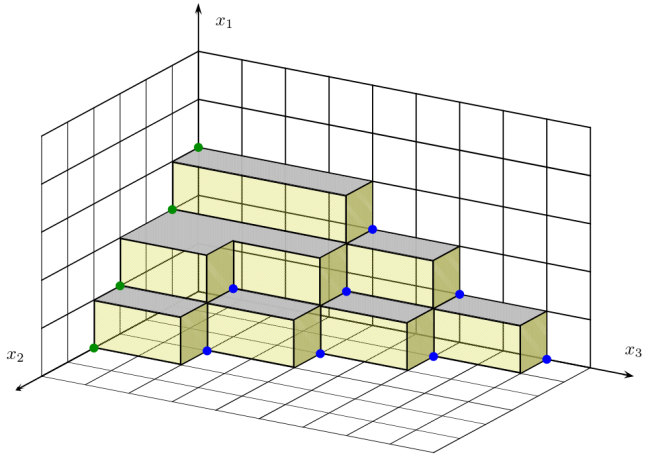
For $x_1 \succ_{LEX} \cdots \succ_{LEX} x_n$, GB_{LEX} looks like

$$w(x_n),\ x_i + v_i(x_n), \quad n-1 \geq i \geq 1$$

Change of monomial orders



$$w(t) = 0, \quad x_i = \frac{v_i(t)}{w'(t)}, \quad 1 \leq i \leq n$$

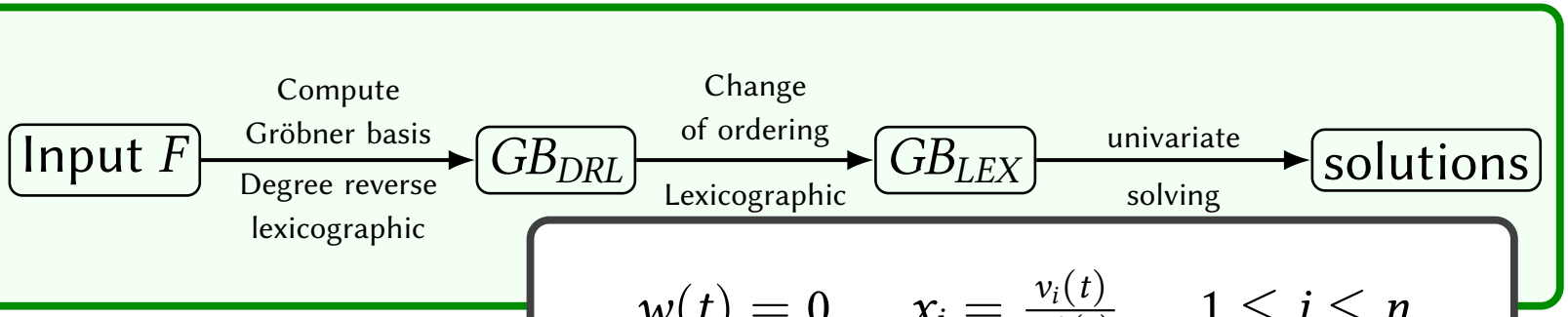


- Basis $\mathscr{B}$ of $\mathbb{A} = \frac{\mathbb{K}[x_1,\ldots,x_n]}{\langle F \rangle}$ is finite. FGLM algo.
- Shape position for radical ideals is generic

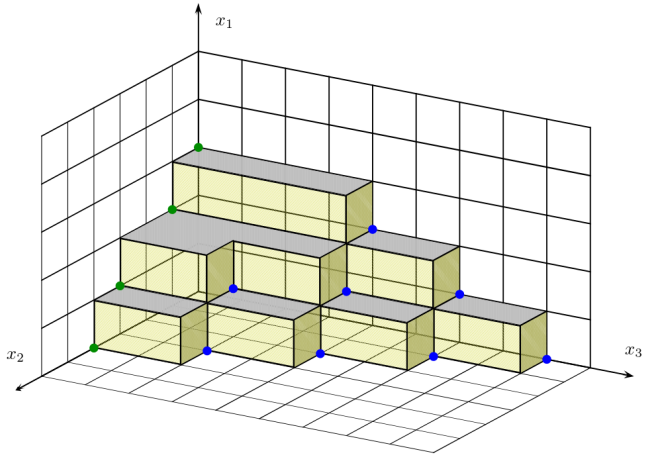
For $x_1 \succ_{LEX} \cdots \succ_{LEX} x_n$, GB_{LEX} looks like

$$w(x_n), \; x_i + v_i(x_n), \quad n-1 \geq i \geq 1$$

Change of monomial orders



$$w(t) = 0, \quad x_i = \frac{v_i(t)}{w'(t)}, \quad 1 \leq i \leq n$$



- Basis $\mathscr{B}$ of $\mathbb{A} = \frac{\mathbb{K}[x_1,\ldots,x_n]}{\langle F \rangle}$ is finite. FGLM algo.
- Shape position for radical ideals is generic

For $x_1 \succ_{LEX} \cdots \succ_{LEX} x_n$, GB_{LEX} looks like

$$w(x_n), \; x_i + v_i(x_n), \quad n-1 \geq i \geq 1$$

The generic staircase of grevlex Gröbner bases **(Moreno-Socías)**

For any $m \in \mathscr{B}$, either $mx_n \in \mathscr{B}$ or $mx_n \in \mathrm{LM}_{\succ_{DRL}}(GB_{DRL})$

Faugère/Mou's FGLM in msolve

Restrictions

msolve's change of order implementation requires genericity

Genericity checked and achieved by adding $t+\sum_{i=1}^{n} r_i x_i$ (r_i random) to the input system

Faugère/Mou's FGLM in msolve

Restrictions

msolve's change of order implementation requires genericity

Genericity checked and achieved by adding $t+\sum_{i=1}^{n} r_i x_i$ (r_i random) to the input system

For any $m \in \mathscr{B}$, either $mx_n \in \mathscr{B}$ or $mx_n \in \mathrm{LM}_{\succ_{DRL}}(GB_{DRL})$

Faugère/Mou's FGLM in msolve

Restrictions

msolve's change of order implementation requires genericity
Genericity checked and achieved by adding $t+\sum_{i=1}^{n} r_i x_i$ (r_i random) to the input system

For any $m \in \mathscr{B}$, either $mx_n \in \mathscr{B}$ or $mx_n \in \mathrm{LM}_{\succ_{DRL}}(GB_{DRL})$

$$f \in \mathbb{A} \to x_n f \in \mathbb{A}$$

Faugère/Mou's FGLM in msolve

Restrictions

msolve's change of order implementation requires genericity
Genericity checked and achieved by adding $t+\sum_{i=1}^{n} r_i x_i$ (r_i random) to the input system

For any $m \in \mathscr{B}$, either $mx_n \in \mathscr{B}$ or $mx_n \in \mathrm{LM}_{\succ_{DRL}}(GB_{DRL})$

$f \in \mathbb{A} \to x_n f \in \mathbb{A}$

- $\delta - m$ "trivial" rows

Faugère/Mou's FGLM in msolve

Restrictions

msolve's change of order implementation requires genericity
Genericity checked and achieved by adding $t+\sum_{i=1}^{n} r_i x_i$ (r_i random) to the input system

For any $m \in \mathscr{B}$, either $mx_n \in \mathscr{B}$ or $mx_n \in \mathrm{LM}_{\succ_{DRL}}(GB_{DRL})$

$f \in \mathbb{A} \to x_n f \in \mathbb{A}$

- $\delta - m$ "trivial" rows
- m "dense" rows

Faugère/Mou's FGLM in msolve

Restrictions

msolve's change of order implementation requires genericity

Genericity checked and achieved by adding $t+\sum_{i=1}^{n} r_i x_i$ (r_i random) to the input system

For any $m \in \mathscr{B}$, either $mx_n \in \mathscr{B}$ or $mx_n \in \mathrm{LM}_{\succ_{DRL}}(GB_{DRL})$

$$f \in \mathbb{A} \to x_n f \in \mathbb{A}$$

- $\delta - m$ "trivial" rows
- m "dense" rows

Complexity

$$O(m\delta^2 + n\delta \log^2 \delta)$$

Wiedemann + BMS algos

Generic asymptotics

$m \simeq \sqrt{\frac{6}{n\pi}} D^{n-1}$ with $\delta = D^n$

Faugère/Mou's FGLM in msolve

Restrictions

msolve's change of order implementation requires genericity

Genericity checked and achieved by adding $t+\sum_{i=1}^{n} r_i x_i$ (r_i random) to the input system

For any $m \in \mathscr{B}$, either $mx_n \in \mathscr{B}$ or $mx_n \in \mathrm{LM}_{\succ_{DRL}}(GB_{DRL})$

$$f \in \mathbb{A} \to x_n f \in \mathbb{A}$$

- $\delta - m$ "trivial" rows
- m "dense" rows

Complexity

$O(m\delta^2 + n\delta \log^2 \delta)$

Wiedemann + BMS algos

Generic asymptotics

$m \simeq \sqrt{\frac{6}{n\pi}} D^{n-1}$ with $\delta = D^n$

- ✓ dedicated encoding of matrices
- ✓ trivial / dense rows
- ✓ AVX2 matrix vector product
- ✓ BMS handled by FLINT

Faugère/Mou's FGLM in msolve

Restrictions

msolve's change of order implementation requires genericity

Genericity checked and achieved by adding $t+\sum_{i=1}^{n} r_i x_i$ (r_i random) to the input system

For any $m \in \mathscr{B}$, either $mx_n \in \mathscr{B}$ or $mx_n \in \mathrm{LM}_{\succ_{DRL}}(GB_{DRL})$

$$f \in \mathbb{A} \to x_n f \in \mathbb{A}$$

- $\delta - m$ "trivial" rows
- m "dense" rows

Complexity

$O(m\delta^2 + n\delta \log^2 \delta)$

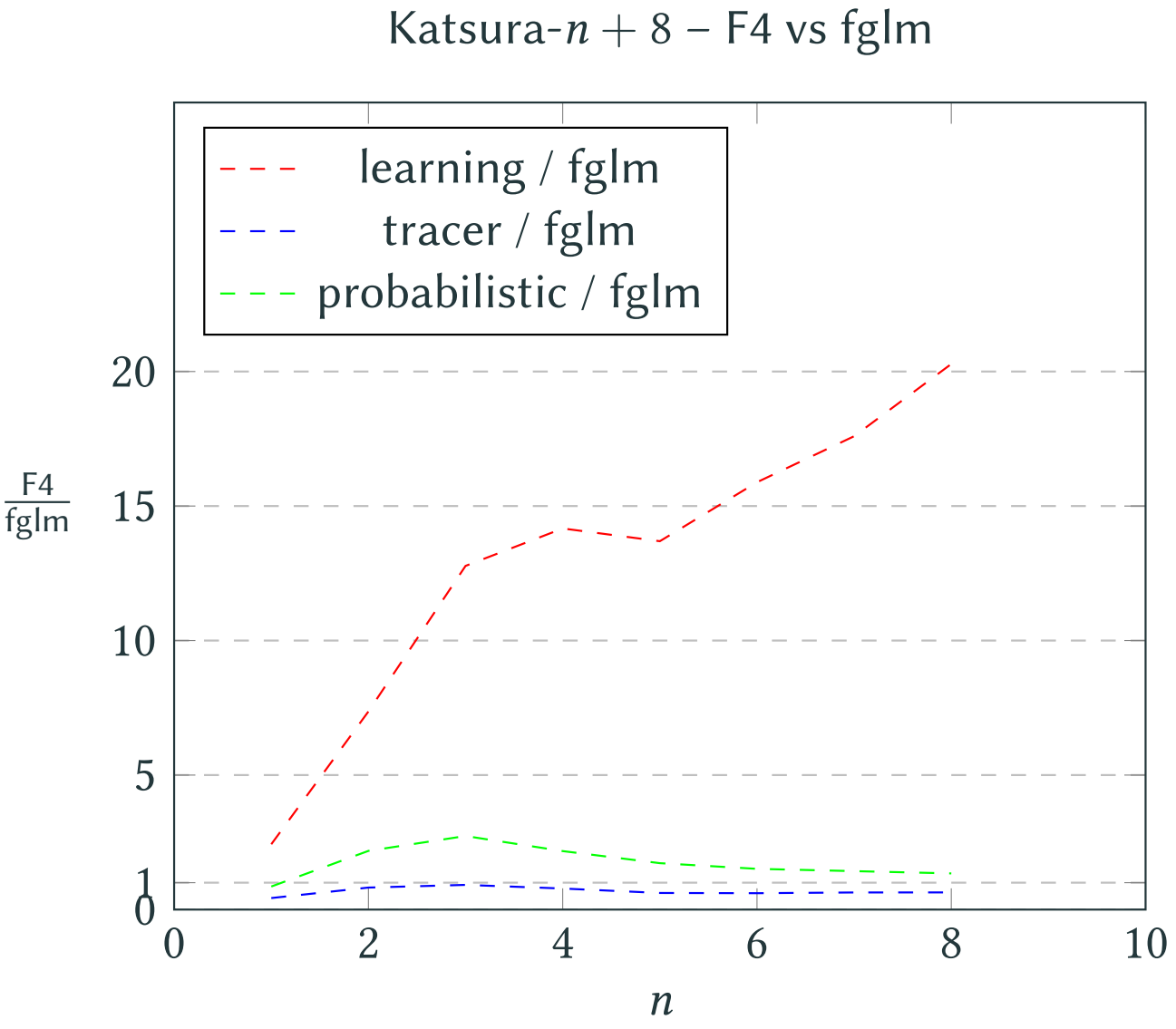
Wiedemann + BMS algos

Generic asymptotics

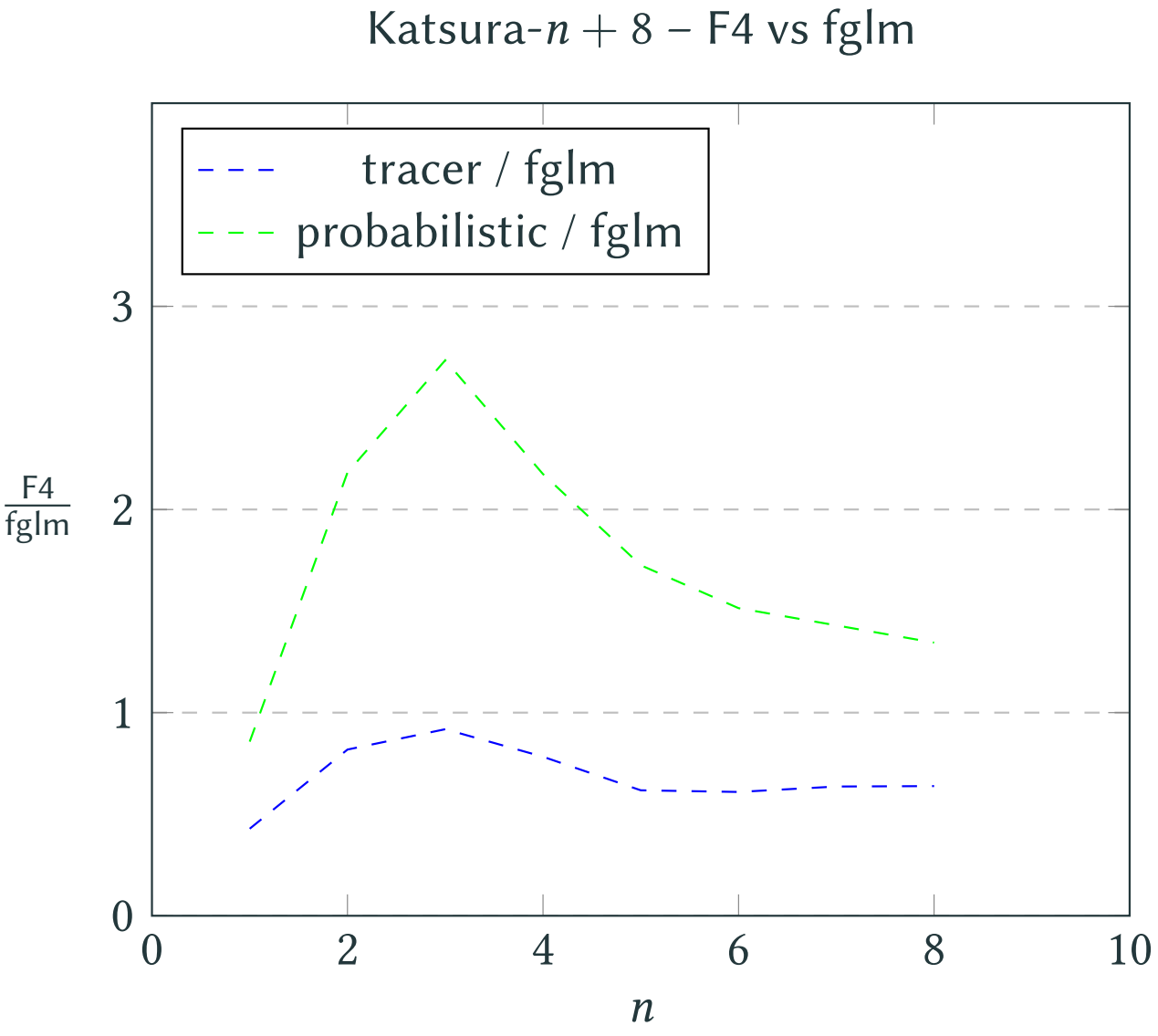
$m \simeq \sqrt{\frac{6}{n\pi}} D^{n-1}$ with $\delta = D^n$

- ✓ dedicated encoding of matrices
- ✓ trivial / dense rows
- ✓ AVX2 matrix vector product
- ✓ BMS handled by FLINT
- ✓ Non-radical case $\rightsquigarrow$ new saturation

msolve F4 / msolve Sparse FGLM



msolve F4 / msolve Sparse FGLM



FGLM comparisons

Examples	`msolve` FGLM	`maple` FGLM	ratio
Katsura-10	0.11	0.15	1.36
Katsura-11	0.49	0.74	1.51
Katsura-12	3.96	5.4	1.36
Katsura-13	30.6	35.7	1.16
Katsura-14	210	271	1.29
Eco-11	0.07	0.12	1.71
Eco-12	0.34	0.85	2.5
Eco-13	2.12	6.7	3.16
Eco-14	25.9	69.1	2.67
Henrion-6	0.11	0.11	1
Henrion-7	20.46	27.1	1.32
Noon-7	1.95	3.13	1.6
Noon-8	72.3	76.2	1.05

Solving

Examples	DEG	msolve(trace)	msolve(prob)	speed-up	maple	speed-up	magma	speed-up
Katsura-9	256	4.89	7.49	1.53	104	21.27	2522	515
Katsura-10	512	43.7	70.5	1.61	1 278	29.24	82 540	1 888
Katsura-11	1024	424	814	1.92	7 812	18.4	-	
Katsura-12	2048	6 262	11 215	1.79	120 804	19.29	-	
Katsura-13	4096	89 390	148 372	1.66	-		-	
Katsura-14	8192	1 308 602	2 000 170	1.53	-		-	
Eco-10	256	12.5	21.2	1.69	26.3	2.1	6520	521.6
Eco-11	512	90.3	161	1.78	312	3.45	214 770	2378
Eco-12	1024	877	1 619	1.84	4 287	4.88	-	
Eco-13	2048	12 137	19 553	1.61	66 115	5.44	-	
Eco-14	4096	167 798	254 389	1.51	-		-	
Henrion-5	100	0.71	0.83	1.17	2.7	3.8	93	130.98
Henrion-6	720	138	157	1.13	1 470	10.65	-	
Henrion-7	5040	117 803	127 456	1.08	-		-	
CP(3,5,2)	288	18.1	19.2	1.06	249	13.75	-	
CP(3,6,2)	720	390	450	1.15	23 440	60	-	
CP(3,7,2)	1728	9 643	11 511	1.19	-		-	
CP(3,8,2)	4032	269 766	323 838	1.2	-		-	

Solving

Examples	DEG	msolve(trace)	msolve(prob)	speed-up	maple	speed-up	magma	speed-up
Katsura-9	256	4.89	7.49	1.53	104	21.27	2522	515
Katsura-10	512	43.7	70.5	1.61	1 278	29.24	82 540	1 888
Katsura-11	1024	424	814	1.92	7 812	18.4	-	
Katsura-12	2048	6 262	11 215	1.79	120 804	19.29	-	
Katsura-13	4096	89 390	148 372	1.66	-		-	
Katsura-14	8192	1 308 602	2 000 170	1.53	-		-	
Eco-10	256	12.5	21.2	1.69	26.3	2.1	6520	521.6
Eco-11	512	90.3	161	1.78	312	3.45	214 770	2378
Eco-12	1024	877	1 619	1.84	4 287	4.88	-	
Eco-13	2048	12 137	19 553	1.61	66 115	5.44	-	
Eco-14	4096	167 798	254 389	1.51	-		-	
Henrion-5	100	0.71	0.83	1.17	2.7	3.8	93	130.98
Henrion-6	720	138	157	1.13	1 470	10.65	-	
Henrion-7	5040	117 803	127 456	1.08	-		-	
CP(3,5,2)	288	18.1	19.2	1.06	249	13.75	-	
CP(3,6,2)	720	390	450	1.15	23 440	60	-	
CP(3,7,2)	1728	9 643	11 511	1.19	-		-	
CP(3,8,2)	4032	269 766	323 838	1.2	-		-	
Noon-7	2173	4039	5 045	1.25	432	0.1	-	
Noon-8	6545	598 647	640 177	1.07	5997	0.01	-	

Solving

Examples	DEG	msolve(trace)	msolve(prob)	speed-up	maple	speed-up	magma	speed-up
Katsura-9	256	4.89	7.49	1.53	104	21.27	2522	515
Katsura-10	512	43.7	70.5	1.61	1 278	29.24	82 540	1 888
Katsura-11	1024	424	814	1.92	7 812	18.4	-	
Katsura-12	2048	6 262	11 215	1.79	120 804	19.29	-	
Katsura-13	4096	89 390	148 372	1.66	-		-	
Katsura-14	8192	1 308 602	2 000 170	1.53	-		-	
Eco-10	256	12.5	21.2	1.69	26.3	2.1	6520	521.6
							70	2378
							-	
							-	
							-	
							93	130.98
							-	
Henrion-7	5040	117 803	127 456	1.08	-		-	
CP(3,5,2)	288	18.1	19.2	1.06	249	13.75	-	
CP(3,6,2)	720	390	450	1.15	23 440	60	-	
CP(3,7,2)	1728	9 643	11 511	1.19	-		-	
CP(3,8,2)	4032	269 766	323 838	1.2	-		-	
Noon-7	2173	4039	5 045	1.25	432	0.1	-	
Noon-8	6545	598 647	640 177	1.07	5997	0.01	-	

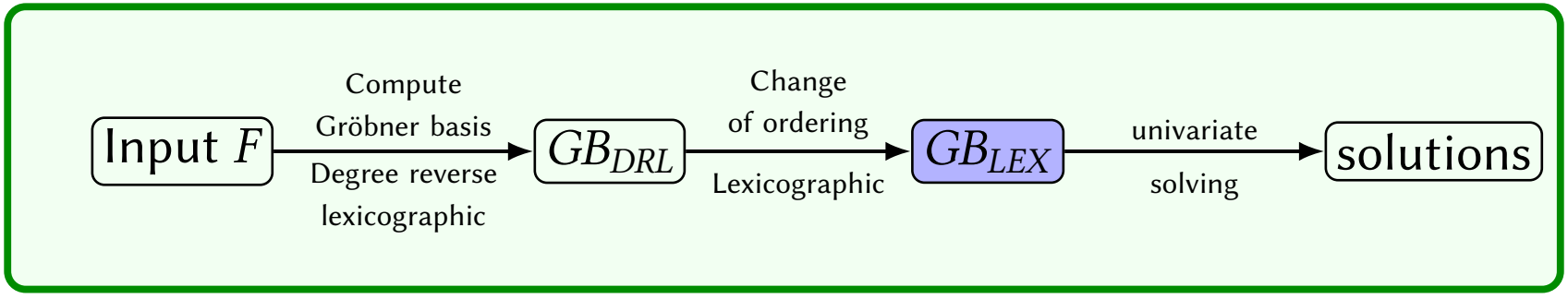
On Noon examples we suffer from the bit size of our output parametrizations (which could be split in many small components)

Memory usage

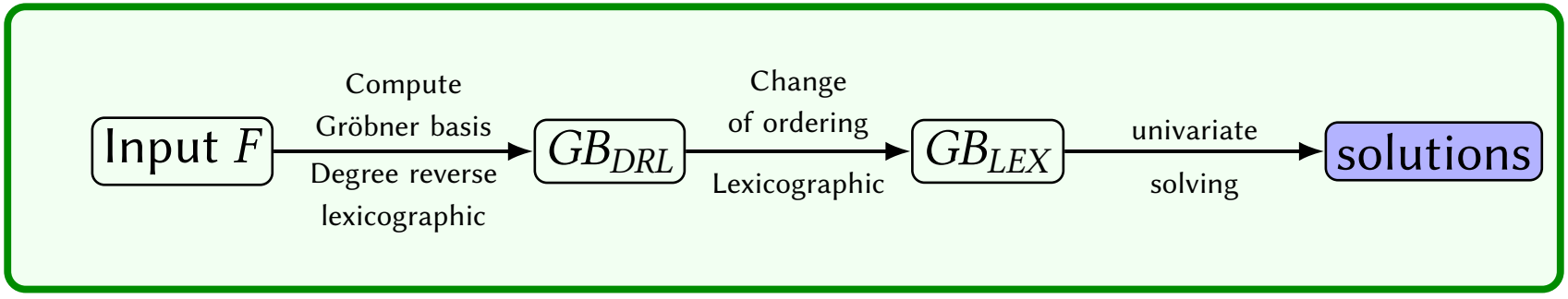
Examples	msolve	maple	magma
Katsura-9	15	271	71
Katsura-10	25	276	223
Katsura-11	66	2,279	–
Katsura-12	229	1,123	–
Katsura-13	1,037	–	–
Eco-10	27	210	213
Eco-11	82	428	354
Eco-12	117	1,027	–
Eco-13	318	8,654	–
Eco-14	15,748	–	–
Henrion-5	11	26	23
Henrion-6	47	171	–
Henrion-7	3,428	–	–
Noon-7	209	419	–
Noon-8	881	1,227	–
Phuoc-1	176	–	–
CP(3, 5, 2)	17	525	–
CP(3, 6, 2)	55	7,885	–
CP(3, 7, 2)	312	–	–
CP(4, 4, 3)	24	635	–
CP(4, 5, 3)	2,065	–	–

Maximal memory usage given in MB

Univariate real root isolation



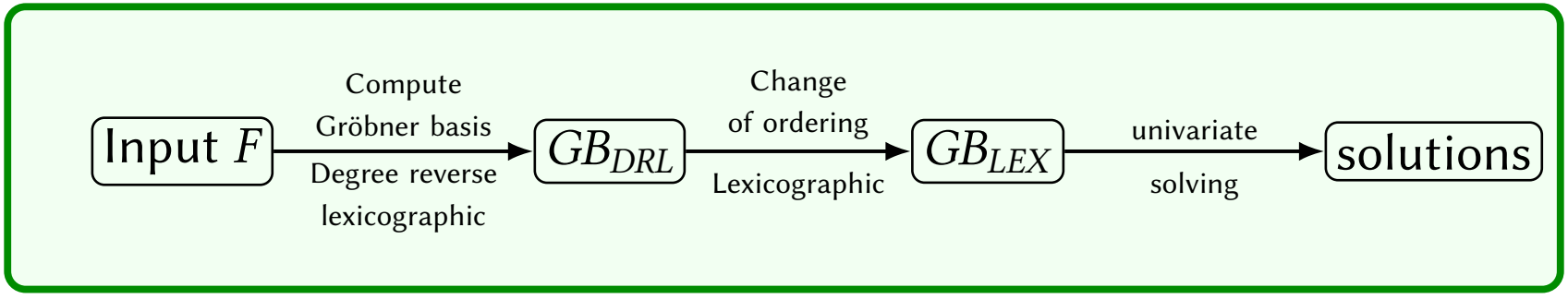
Univariate real root isolation



Subdivision algorithms

Descartes' rule of signs

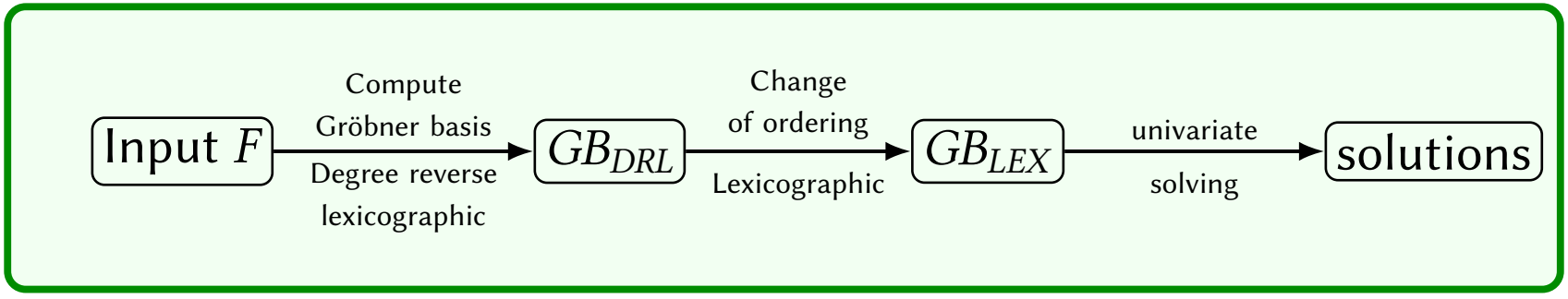
Univariate real root isolation



Subdivision algorithms **Descartes' rule of signs**

- Reduce the search to roots in $(0, 1)$
- Decide if there is 0 or 1 root
- If no decision can be made
  - Recursive call over $\left(0, \frac{1}{2}\right)$
  - Recursive call over $\left(\frac{1}{2}, 1\right)$

Univariate real root isolation



Subdivision algorithms

- Reduce the search to roots in $(0, 1)$
- Decide if there is 0 or 1 root
- If no decision can be made
  - Recursive call over $\left(0, \frac{1}{2}\right)$
  - Recursive call over $\left(\frac{1}{2}, 1\right)$

Descartes' rule of signs

Complexity:
♯nodes × Cost of
Taylor-shifts

✓ Fast Taylor-shift
Gathen/Gerhard (ISSAC'97)

Comparisons

Examples	♯ sols	msolve	maple		SLV		tdescartes	
		time	time	ratio	time	ratio	time	ratio
Katsura-10	120	3.1	4.8	1.5	3.8	1.2	20	6.5
Katsura-11	216	27	60	2.2	50.5	1.9	156	5.8
Katsura-12	326	207	656	3.2	555	2.7	2, 206	10.6
Katsura-13	582	2 220	16 852	7.6	13 651	6.1	22 945	10.3
Katsura-14	900	20 149	250 094	12.4	252 183	12.5	384 566	19.1
Katsura-15	1, 606	197 048	3 588 835	18.2	3 540 480	18.0	5 178 180	26.3
Katsura-16	2, 543	1 849 986	–	–	–	–	–	–
Katsura-17	4, 428	16 128 000	–	–	–	–	–	–

Real root isolation timings given in seconds

Warning: uses maple-v16

Saturation of polynomial ideals and Gröbner bases

joint with **J. Berthomieu** and **C. Eder**

Motivations and state-of-the-art

Ideal theoretic operations and geometry

Ubiquitous for **geometric computing**.

Motivations and state-of-the-art

Ideal theoretic operations and geometry

Ubiquitous for **geometric computing**.

Key operation: computing colon/saturated ideals.

Let $I \subset \mathbb{K}[x_1, \ldots, x_n]$ and $\varphi \in \mathbb{K}[x_1, \ldots, x_n]$

$$I : \langle \varphi \rangle = \{g \mid g\varphi \in I\} \quad \text{and} \quad I : \langle \varphi \rangle^{\infty} = \{g \mid \exists k \in \mathbb{N}\ g\varphi^k \in I\}$$

Motivations and state-of-the-art

Ideal theoretic operations and geometry

Ubiquitous for **geometric computing**.

Key operation: computing colon/saturated ideals.

Let $I \subset \mathbb{K}[x_1, \ldots, x_n]$ and $\varphi \in \mathbb{K}[x_1, \ldots, x_n]$

$$I : \langle \varphi \rangle = \{g \mid g\varphi \in I\} \quad \text{and} \quad I : \langle \varphi \rangle^\infty = \{g \mid \exists k \in \mathbb{N}\ g\varphi^k \in I\}$$

- There exists $N \in \mathbb{N}$ s.t.
  $I : \langle \varphi \rangle^\infty = I : \langle \varphi^N \rangle = I : \langle \varphi^{N+1} \rangle = I : \langle \varphi^{N+2} \rangle = \cdots$
- $\text{Variety}(I : \langle \varphi \rangle^\infty) = \text{Closure}(\text{Variety}(I) - \text{Variety}(\varphi))$.
- Applications in **algorithms of real algebraic geometry**

Motivations and state-of-the-art

Ideal theoretic operations and geometry

Ubiquitous for **geometric computing**.

Key operation: computing colon/saturated ideals.

Let $I \subset \mathbb{K}[x_1, \ldots, x_n]$ and $\varphi \in \mathbb{K}[x_1, \ldots, x_n]$

$$I : \langle \varphi \rangle = \{g \mid g\varphi \in I\} \quad \text{and} \quad I : \langle \varphi \rangle^\infty = \{g \mid \exists k \in \mathbb{N}\ g\varphi^k \in I\}$$

- There exists $N \in \mathbb{N}$ s.t.
  $I : \langle \varphi \rangle^\infty = I : \langle \varphi^N \rangle = I : \langle \varphi^{N+1} \rangle = I : \langle \varphi^{N+2} \rangle = \cdots$
- $\text{Variety}(I : \langle \varphi \rangle^\infty) = \text{Closure}(\text{Variety}(I) - \text{Variety}(\varphi))$.
- Applications in **algorithms of real algebraic geometry**

State-of-the-art

Rabinowitsch (Rainich)'s trick $I + \langle t\varphi - 1 \rangle \cap \mathbb{K}[x_1, \ldots, x_n] = I : \langle \varphi \rangle^\infty$.

Bayer (1982) – Homogeneization + graded monomial orderings

Geometric resolution approaches (Giusti/Lecerf/Salvy et al.)

Faugère's F4 algorithm

Input. $F = (f_1, \ldots, f_s)\mathbb{K}[x_1, \ldots, x_n], \succ$

Output. Gröbner basis G for $\langle F \rangle, \succ$

- $G \leftarrow \{f_1, \ldots, f_s\}$
- $P \leftarrow \{\text{pairs}(f_i, f_j) \mid 1 \leq i < j \leq m\}$
- While $(P \neq \emptyset)$
  - Choose subset $S \subset P$, $P \leftarrow P \setminus S$
  - $L \leftarrow$ symbolic preprocess(S, G)
  - $L \leftarrow$ linear algebra(L)
  - For $f \in L$ with $\text{LM}_{\succ}(f) \notin L(G)$
    - $P \leftarrow P \cup \{(f, g) \mid g \in G\}$
    - $G \leftarrow G \cup \{f\}$
- **return** G

Faugère's F4 algorithm

Input. $F = (f_1, \ldots, f_s)\mathbb{K}[x_1, \ldots, x_n], \succ$

Output. Gröbner basis G for $\langle F \rangle, \succ$

- $G \leftarrow \{f_1, \ldots, f_s\}$
- $P \leftarrow \{\text{pairs}(f_i, f_j) \mid 1 \leq i < j \leq m\}$
- While $(P \neq \emptyset)$
  - Choose subset $S \subset P$, $P \leftarrow P \setminus S$
  - $L \leftarrow$ symbolic preprocess(S, G)
  - $L \leftarrow$ linear algebra(L)
  - For $f \in L$ with $\text{LM}_{\succ}(f) \notin L(G)$
    - $P \leftarrow P \cup \{(f, g) \mid g \in G\}$
    - $G \leftarrow G \cup \{f\}$
- **return** G

F4 (degree selection strategy) $\rightarrow$
basis of
$$\{g = \sum_i q_i g_i \mid g_i \in G, \deg(q_i g_i) \leq D\}$$

Simple idea. Guess polynomials in $I : \langle \varphi \rangle$ **"on the fly"** through linear algebra **"à la F4"**.

$\rightsquigarrow$ **degree-wise guessing**

New saturation procedure

Reduction to kernel computations

$$f = \sum_{\alpha \in \Sigma} c_\alpha m_\alpha \in I : \langle \varphi \rangle \iff \sum_{\alpha \in \Sigma} c_\alpha \mathsf{NormalForm}_{\succ}(m_\alpha . \varphi, G) = 0$$

New saturation procedure

Reduction to kernel computations

$$f = \sum_{\alpha \in \Sigma} c_\alpha m_\alpha \in I : \langle \varphi \rangle \iff \sum_{\alpha \in \Sigma} c_\alpha \,\mathsf{NormalForm}_\succ(m_\alpha.\varphi, G) = 0$$

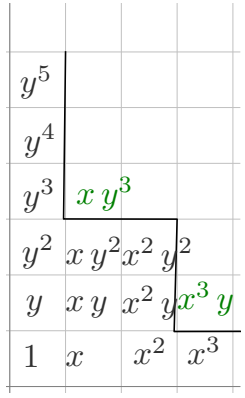
Basic lemma Berthomieu/Eder/S.

Let $S = \{m \mid m \notin \mathsf{LM}_\succ(G)\}$

f is in the reduced GB of $\langle I \rangle : \langle \varphi \rangle$

and $\mathsf{LM}_\succ(f) \notin \mathsf{LM}_\succ(G)$

Then $f \notin I$ and $\mathrm{support}(f) \subset S$.



New saturation procedure

Reduction to kernel computations

$$f = \sum_{\alpha \in \Sigma} c_\alpha m_\alpha \in I : \langle \varphi \rangle \iff \sum_{\alpha \in \Sigma} c_\alpha \,\mathrm{NormalForm}_\succ(m_\alpha.\varphi, G) = 0$$

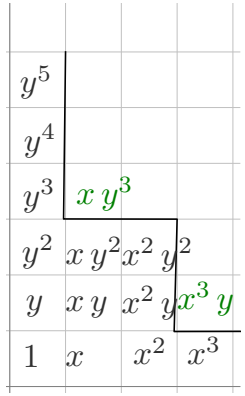
Basic lemma Berthomieu/Eder/S.

Let $S = \{m \mid m \notin \mathrm{LM}_\succ(G)\}$

f is in the reduced GB of $\langle I \rangle : \langle \varphi \rangle$

and $\mathrm{LM}_\succ(f) \notin \mathrm{LM}_\succ(G)$

Then $f \notin I$ and $\mathrm{support}(f) \subset S$.



Truncated F4sat algorithm

- $\mathrm{NormalForm}_\succ(m\varphi, G)$ for $m \in S$ "à la F4" degree-wise.
- Compute vanishing linear combinations of them.
- Update the basis + ...

New saturation procedure

Reduction to kernel computations

$$f = \sum_{\alpha \in \Sigma} c_\alpha m_\alpha \in I : \langle \varphi \rangle \iff \sum_{\alpha \in \Sigma} c_\alpha \,\mathrm{NormalForm}_\succ(m_\alpha.\varphi, G) = 0$$

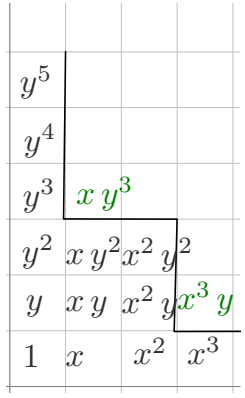
Basic lemma Berthomieu/Eder/S.

Let $S = \{m \mid m \notin \mathrm{LM}_\succ(G)\}$

f is in the reduced GB of $\langle I \rangle : \langle \varphi \rangle$

and $\mathrm{LM}_\succ(f) \notin \mathrm{LM}_\succ(G)$

Then $f \notin I$ and $\mathrm{support}(f) \subset S$.



Truncated F4sat algorithm

- $\mathrm{NormalForm}_\succ(m\varphi, G)$ for $m \in S$ "à la F4" degree-wise.
- Compute vanishing linear combinations of them.
- Update the basis + ...Termination criterion.

The F4SAT algorithm (Berthomieu/Eder/S.)

Input. $(f_1, \ldots, f_s, \varphi) \subset \mathbb{K}[x_1, \ldots, x_n], \succ$
Output. Gröbner basis G for $\langle f_1, \ldots, f_s \rangle : \langle \varphi \rangle^\infty$ w.r.t. $\succ$

- $G \leftarrow \{f_1, \ldots, f_s\}$
- $P \leftarrow \{\text{pairs}(f_i, f_j) \mid 1 \leq i < j \leq m\}$
- While $(P \neq \emptyset)$
  - Choose subset $S \subset P$, $P \leftarrow P \setminus S$
  - $L \leftarrow$ symbolic preprocess(S, G)
  - $L \leftarrow$ linear algebra(L)
  - For $f \in L$ with $\text{LM}_\succ(f) \notin L(G)$
    - $P \leftarrow P \cup \{(f, g) \mid g \in G\}$
    - $G \leftarrow G \cup \{f\}$
- **return** G

The F4SAT algorithm (Berthomieu/Eder/S.)

Input. $(f_1, \ldots, f_s, \varphi) \subset \mathbb{K}[x_1, \ldots, x_n], \succ$
Output. Gröbner basis G for $\langle f_1, \ldots, f_s \rangle : \langle \varphi \rangle^\infty$ w.r.t. $\succ$

- $G \leftarrow \{f_1, \ldots, f_s\}$
- $P \leftarrow \{\text{pairs}(f_i, f_j) \mid 1 \leq i < j \leq m\}$
- While $(P \neq \emptyset)$
  - Choose subset $S \subset P$, $P \leftarrow P \setminus S$
  - $L \leftarrow$ symbolic preprocess(S, G)
  - $L \leftarrow$ linear algebra(L)
  - For $f \in L$ with $\text{LM}_\succ(f) \notin L(G)$
    - $P \leftarrow P \cup \{(f, g) \mid g \in G\}$
    - $G \leftarrow G \cup \{f\}$
- **return** G

- for $m \notin \text{LM}_\succ(G), \deg(m) \leq \deg(G)$, compute NormalForm$(m\varphi, G)$
- Macaulay matrix M of these forms
- Basis of the kernel
- Update the basis G and pairs set P

The F4SAT algorithm (Berthomieu/Eder/S.)

Input. $(f_1, \ldots, f_s, \varphi) \subset \mathbb{K}[x_1, \ldots, x_n], \succ$
Output. Gröbner basis G for $\langle f_1, \ldots, f_s \rangle : \langle \varphi \rangle^\infty$ w.r.t. $\succ$

- $G \leftarrow \{f_1, \ldots, f_s\}$
- $P \leftarrow \{\text{pairs}(f_i, f_j) \mid 1 \leq i < j \leq m\}$
- While $(P \neq \emptyset)$
  - Choose subset $S \subset P$, $P \leftarrow P \setminus S$
  - $G \leftarrow G \cup \{f\}$
- **return** G

- for $m \notin \mathrm{LM}_\succ(G), \deg(m) \leq \deg(G)$, compute NormalForm$(m\varphi, G)$
- Macaulay matrix M of these forms
- Basis of the kernel
- Update the basis G and pairs set P

Termination criterion based on properties raised by Bayer (1982)
Works only with total degree orderings

Practical results

Computing limits of critical points in singular hypersurfaces

Examples	F4SAT (learn 1)	F4SAT (learn 2)	F4SAT (apply)	msolve (prob.)	msolve (learn)	msolve (apply)	Maple (prob.)	Singular	Macaulay2
SOS-d3-n6-p2	1.31	0.,41	0.31	0.77	2.40	0.40	1.12	52.2	248
SOS-d3-n6-p3	43.7	5.55	1.84	25.2	142	16.6	35.4	2,902	38,636
SOS-d3-n6-p4	533	53.1	19.7	171	882	126	223	39,501	–
SOS-d3-n6-p5	1,863	184	104	276	1,145	183	394	42,854	–
SOS-d4-n6-p2	972	107	77	253	1,176	191	394	28,043	–
SOS-d4-n6-p3	31,101	1,316	596	7,444	43,803	6,336	8,817	–	
SOS-d2-n7-p6	5.13	1.82	0.77	3.01	15.3	1.84	4.95	443	3,903
SOS-d3-n7-p2	13.4	3.61	2.23	9.59	54.1	5.29	12.5	872	8,117
SOS-d3-n7-p3	1,263	164	32.4	533	3,647	406	984	–	–
SOS-d3-n7-p4	22,296	2,235	469	6,605	47,286	5,348	10,001	–	–
SOS-d3-n7-p5	126,006	137,724	2,881	29,740	204,718	22,925	33,635	–	–
SOS-d2-n8-p5	11.7	8.37	1.79	15.1	99.9	7.92	20.4	3,972	31,282
SOS-d2-n8-p6	95.7	63.7	10.5	54.3	387	33.8	63.1	15,950	–
SOS-d2-n8-p7	265	79.6	22.2	81.9	556	47.2	122	15,125	–
SOS-d3-n8-p2	228	276	18.1	98.3	787	71.7	135	15,252	–
SOS-d3-n8-p3	25,593	3,716	471	11,050	107,744	8,984	13,705	–	

Timings in seconds, *DRL* GB, positive-to-positive-dimensional case

Towards faster change of ordering algorithms

joint with **J. Berthomieu** and **V. Neiger**

Back to FGLM

- Fast implementation available in msolve
- BUT FGLM takes now more time than F4-tracer

system	nvars	D	t	F4	F4tracer	fglm	ratio (fglm/tracer)
random-2-10	10	1024	252	1.72	0.23	0.18	0.78
random-2-11	11	2048	462	11.55	1.13	1.17	1.03
random-2-12	12	4096	924	115.89	8.3	6.53	0.78
random-2-13	13	8192	1716	970	62	103	1.67
random-2-14	14	16384	3432	7921	460	1011	2.2
random-2-15	15	32768	6435	61381	3193	7844	2.45
random-2-16	16	65536	12870	482515	24523	58744	2.4

msolve timings in secs for square systems of random quadrics

Back to FGLM

- Fast implementation available in msolve
- BUT FGLM takes now more time than F4-tracer

system	nvars	D	t	F4	F4tracer	fglm	ratio (fglm/tracer)
random-3-7	7	2187	393	3.59	0.72	0.98	1.36
random-3-8	8	6561	1107	122.55	12.77	23.62	1.85
random-3-9	9	19683	3139	3552.7	361	1302	3.61
random-3-10	10	59049	8953	95052	8664	34844	4.01

msolve timings in secs for square systems of random cubics

Back to FGLM

- 👍 Fast implementation available in msolve
- 👎 BUT FGLM takes now more time than F4-tracer

system	nvars	D	t	F4	F4tracer	fglm	ratio (fglm/tracer)
random-3-7	7	2187	393	3.59	0.72	0.98	1.36
random-3-8	8	6561	1107	122.55	12.77	23.62	1.85
random-3-9	9	19683	3139	3552.7	361	1302	3.61
random-3-10	10	59049	8953	95052	8664	34844	4.01

msolve timings in secs for square systems of random cubics

- 👎 overall behaviour of FGLM is not satisfactory

$\rightsquigarrow$ new approach needed.

A closer look at the complexity

For any $m \in \mathscr{B}$, either $mx_n \in \mathscr{B}$ or $mx_n \in \mathsf{LM}_{\succ_{DRL}}(GB_{DRL})$

$$f \in \mathbb{A} \to x_n f \in \mathbb{A}$$

- $\delta - m$ "trivial" rows
- m "dense" rows

Complexity

$$O(m\delta^2 + n\delta \log^2 \delta)$$

Wiedemann + BM algos

Generic asymptotics $m \simeq \sqrt{\frac{6}{n\pi}} D^{n-1}$ with $\delta = D^n$

A closer look at the complexity

For any $m \in \mathscr{B}$, either $mx_n \in \mathscr{B}$ or $mx_n \in \mathsf{LM}_{\succ_{DRL}}(GB_{DRL})$

$$f \in \mathbb{A} \to x_n f \in \mathbb{A}$$

- $\delta - m$ "trivial" rows
- m "dense" rows

Complexity

$$O(m\delta^2 + n\delta \log^2 \delta)$$

Wiedemann + BM algos

Generic asymptotics $m \simeq \sqrt{\frac{6}{n\pi}} D^{n-1}$ with $\delta = D^n$

Goal. $O(m^2\delta)$ would bring a speed-up $D\sqrt{\frac{n\pi}{6}}$

See also **Berthomieu/Bostan/Ferguson/S.** for other asymptotics

The structure of matrix multiplications

$$M_3 = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ -45 & -27 & -26 & 28 & -16 & 46 & 12 & 17 \\ 24 & -3 & 36 & -31 & 1 & -14 & -22 & -30 \\ 30 & -27 & -2 & 44 & 11 & -45 & 10 & -10 \end{pmatrix}.$$

$$S = \left[1, x_3, x_2, x_1, x_3^2, x_2x_3, x_1x_3, x_3^3\right] - \mod 97$$

The structure of matrix multiplications

$$M_3 = \left(\begin{array}{cccc|cc|cc} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 30 & 70 & 11 & 87 & 95 & 52 & 44 & 10 \\ \hline 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 52 & 70 & 81 & 17 & 71 & 46 & 28 & 12 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 24 & 94 & 1 & 67 & 36 & 83 & 66 & 75 \end{array}\right).$$

$$S = \left[1, x_3, x_3^2, x_3^3, x_2, x_2x_3, x_1, x_1x_3\right]$$

The structure of matrix multiplications

$$M_3 = \left(\begin{array}{cccc|cc|cc} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 30 & 70 & 11 & 87 & 95 & 52 & 44 & 10 \\ \hline 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 52 & 70 & 81 & 17 & 71 & 46 & 28 & 12 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 24 & 94 & 1 & 67 & 36 & 83 & 66 & 75 \end{array}\right).$$

$$S = \left[1, x_3, x_3^2, x_3^3, x_2, x_2x_3, x_1, x_1x_3\right]$$

$$M_3' = \begin{pmatrix} 30 + 70x_3 + 11x_3^2 + 87x_3^3 + 96x_3^4 & 95 + 52x_3 & 44 + 10x_3 \\ 52 + 70x_3 + 81x_3^2 + 17x_3^3 & 71 + 46x_3 + 96x_3^2 & 28 + 12x_3 \\ 24 + 94x_3 + x_3^2 + 67x_3^3 & 36 + 83x_3 & 66 + 75x_3 + 96x_3^2 \end{pmatrix}.$$

The structure of matrix multiplications

$$M_3 = \left(\begin{array}{cccc|cc|cc} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 30 & 70 & 11 & 87 & 95 & 52 & 44 & 10 \\ \hline 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{array}\right).$$

$\det(M_3')$ is the characteristic polynomial of M_3

Storjohann's PhD

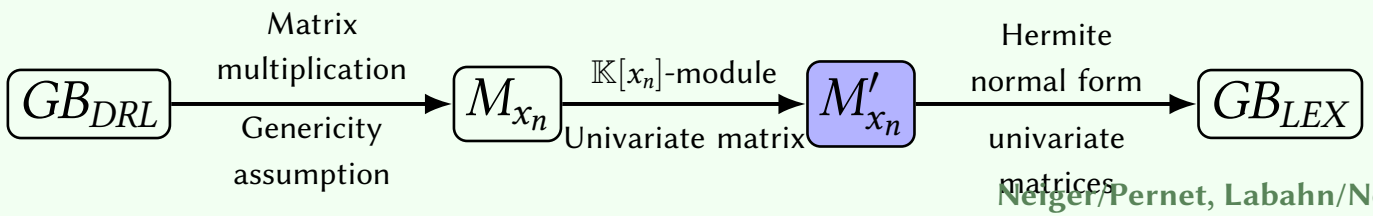
$$S = \left[1, x_3, x_3^2, x_3^3, x_2, x_2x_3, x_1, x_1x_3\right]$$

$$M_3' = \begin{pmatrix} 30 + 70x_3 + 11x_3^2 + 87x_3^3 + 96x_3^4 & 95 + 52x_3 & 44 + 10x_3 \\ 52 + 70x_3 + 81x_3^2 + 17x_3^3 & 71 + 46x_3 + 96x_3^2 & 28 + 12x_3 \\ 24 + 94x_3 + x_3^2 + 67x_3^3 & 36 + 83x_3 & 66 + 75x_3 + 96x_3^2 \end{pmatrix}.$$

BNS algorithm

Berthomieu/Neiger/S.

Under genericity assumptions on the staircase
can be retrieved by changing the last variable



Neiger/Pernet, Labahn/Neiger/Zhou

Complexity

$\tilde{O}(t^{\omega-1}\delta)$ arithmetic operations.

Timings

Example	δ	t	New	current msolve
random-2-14	$16,384$	$3,432$	250	$1,011$
random-3-9	$19,683$	$3,139$	318	$1,302$
random-4-7	$16,384$	$2,128$	320	574
random-5-6	$15,625$	$1,751$	280	422

Timings in seconds – 31 bit prime

Conclusions and perspectives

- ✓ Up-to-date efficient implementation of core algorithms for Gröbner bases and polynomial system solving
- ✓ Can solve polynomial systems that are out of reach of leading computer algebra systems
- ✗ Maple could be improved by revisiting the whole solving process and implementing/using Faugère/Mou's FGLM algorithm
- ✗ Magma could be improved by using rational parametrizations, and implementing Faugère/Mou's FGLM algorithm

Conclusions and perspectives

- ✓ Up-to-date efficient implementation of core algorithms for Gröbner bases and polynomial system solving
- ✓ Can solve polynomial systems that are out of reach of leading computer algebra systems
- ✗ Maple could be improved by revisiting the whole solving process and implementing/using Faugère/Mou's FGLM algorithm
- ✗ Magma could be improved by using rational parametrizations, and implementing Faugère/Mou's FGLM algorithm
- ✗ Make msolve easier to use $\rightsquigarrow$ integration in computer algebra systems

Perspectives

- ✗ Some missing functionalities: elimination orders, regularisation techniques, Hilbert series, splitting techniques, GBLA, etc.
- ✗ **Algorithmic impact: new algorithms for polynomial ideals**
- ✗ F5 algorithms + Equidimensional decompositions (with Eder/Lairez/Mohr)
- ✗ Massively parallel computations $\rightsquigarrow$ multi-threading + GPU
- ✗ Next step: polynomial systems with $\simeq$ 100 000 solutions.

Perspectives

- ✗ Some missing functionalities: elimination orders, regularisation techniques, Hilbert series, splitting techniques, GBLA, etc.
- ✗ **Algorithmic impact: new algorithms for polynomial ideals**
- ✗ F5 algorithms + Equidimensional decompositions (with Eder/Lairez/Mohr)
- ✗ Massively parallel computations $\rightsquigarrow$ multi-threading + GPU
- ✗ Next step: polynomial systems with $\simeq$ 100 000 solutions.

Next release: january 2022 `https://msolve.lip6.fr`