Fast Gröbner basis computation and polynomial reduction for generic bivariate ideals

Joris van der Hoeven, Robin Larrieu

Laboratoire d'Informatique de l'Ecole Polytechnique (LIX)

JNCF 2019 – Luminy
05th February 2019

Introduction

Let $\langle A, B\rangle$ be the ideal generated by A and B ($A, B \in \mathbb{K}[X, Y]$).

- Given $P \in \mathbb{K}[X, Y]$, check if $P \in \langle A, B\rangle$.
  (ideal membership test)
- Compute a normal form of $\bar{P} \in \mathbb{K}[X, Y]/\langle A, B\rangle$.
  (computation in the quotient algebra)

Introduction

Let $\langle A, B\rangle$ be the ideal generated by A and B $(A, B \in \mathbb{K}[X, Y])$.

- Given $P \in \mathbb{K}[X, Y]$, check if $P \in \langle A, B\rangle$.
  (ideal membership test)
- Compute a normal form of $\bar{P} \in \mathbb{K}[X, Y]/\langle A, B\rangle$.
  (computation in the quotient algebra)

Classical solution using *Gröbner bases*.

Introduction

- Fast Gröbner basis algorithms rely on linear algebra (ex: F4, F5. . .)
- Can we do it with polynomial arithmetic?

- Fast Gröbner basis algorithms rely on linear algebra (ex: F4, F5...)
- Can we do it with polynomial arithmetic?
  - Given a Gröbner basis G, can we reduce P modulo G faster?

- Fast Gröbner basis algorithms rely on linear algebra (ex: F4, F5...)
- Can we do it with polynomial arithmetic?
  - Given a Gröbner basis G, can we reduce P modulo G faster?
  - Are these ideas useful to compute G faster?

Introduction

- Fast Gröbner basis algorithms rely on linear algebra (ex: F4, F5...)
- Can we do it with polynomial arithmetic?
  - Given a Gröbner basis G, can we reduce P modulo G faster?
  - Are these ideas useful to compute G faster?

Setting and notations

- $I = \langle A, B \rangle$ with generic $A, B \in \mathbb{K}[X, Y]$ given in total degree.
- Use the degree lexicographic order to compute G.
- $\deg A = n$ and $\deg B = m$ with $n \leqslant m$ (in this talk $n = m$)
- We want to reduce P with $\deg P = d$

Introduction

- Fast Gröbner basis algorithms rely on linear algebra (ex: F4, F5...)
- Can we do it with polynomial arithmetic?
  - Given a Gröbner basis G, can we reduce P modulo G faster?
  - Are these ideas useful to compute G faster?

Setting and notations

- $I = \langle A, B \rangle$ with generic $A, B \in \mathbb{K}[X, Y]$ given in total degree.
- Use the degree lexicographic order to compute G.
- $\deg A = n$ and $\deg B = m$ with $n \leqslant m$ (in this talk $n = m$)
- We want to reduce P with $\deg P = d$

Main result

In this specific setting, a quasi-optimal algorithm exists !

Outline

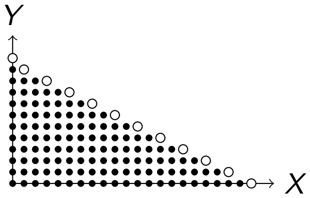
1. Presentation of the problem
   - Polynomial reduction: complexity
   - Gröbner bases: concise representation
2. Faster computation
   - Polynomial reduction
   - Gröbner basis

Outline

1. Presentation of the problem
   - Polynomial reduction: complexity
   - Gröbner bases: concise representation

2. Faster computation

Polynomial reduction: complexity



- A, B: $\Theta(n^2)$ coefficients
- $\mathbb{K}[X, Y]/I$: dimension $\Theta(n^2)$
- G: $\Theta(n^3)$ coefficients ($\Theta(n^2)$ for each G_i)

Reduction using G needs at least $\Theta(n^3)$ $\Longrightarrow$ reduction with less information?

Related results

Theorem (van der Hoeven – ACA 2015)

The extended reduction of P modulo G can be computed in quasi-linear time with respect to the size of the equation

$$P = \sum_i Q_i G_i + R$$

Related results

Theorem (van der Hoeven – ACA 2015)

The extended reduction of P modulo G can be computed in quasi-linear time with respect to the size of the equation

$$P = \sum_i Q_i G_i + R$$

- But this equation has size $\Theta(n^3)$ and we would like to achieve $\tilde{O}(n^2)$ complexity...

Related results

Theorem (van der Hoeven – ACA 2015)

The extended reduction of P modulo G can be computed in quasi-linear time with respect to the size of the equation

$$P = \sum_i Q_i G_i + R$$

- But this equation has size $\Theta(n^3)$ and we would like to achieve $\tilde{O}(n^2)$ complexity...
- $\Longrightarrow$ Somehow reduce the size of the equation.

Related results

Theorem (van der Hoeven, L. – ISSAC 2018)

A special class of bases called *vanilla Gröbner bases* admit a terse representation in $\tilde{O}(n^2)$ space. Assuming this representation has been precomputed, reduction can be done in time $\tilde{O}(n^2)$.

Related results

Theorem (van der Hoeven, L. – ISSAC 2018)

A special class of bases called *vanilla Gröbner bases* admit a terse representation in $\tilde{O}(n^2)$ space. Assuming this representation has been precomputed, reduction can be done in time $\tilde{O}(n^2)$.

- Problem: in this setting, G is *not* vanilla.
  (vanilla Gröbner bases rely on different assumptions)

Related results

Theorem (van der Hoeven, L. – ISSAC 2018)

A special class of bases called *vanilla Gröbner bases* admit a terse representation in $\tilde{O}(n^2)$ space. Assuming this representation has been precomputed, reduction can be done in time $\tilde{O}(n^2)$.

- Problem: in this setting, G is *not* vanilla.
  (vanilla Gröbner bases rely on different assumptions)
- But ... similar ideas still apply.
  (We use essentially the same tricks, although the algorithm is very different).

Gröbner bases: concise representation – 1

The Gröbner basis is generated by A and B $\implies$ there are relations between the G_i (redundant information)

Gröbner bases: concise representation – 1

The Gröbner basis is generated by A and $B \implies$ there are relations between the G_i (redundant information)

- Reduced Gröbner basis:
  $G_{i+2}^{\text{red}} = Spol(G_i^{\text{red}}, G_{i+1}^{\text{red}})$ rem $G_0^{\text{red}}, \ldots, G_{i+1}^{\text{red}}$

Gröbner bases: concise representation – 1

The Gröbner basis is generated by A and B $\Longrightarrow$ there are relations between the G_i (redundant information)

- Reduced Gröbner basis: $G_{i+2}^{\text{red}} = Spol(G_i^{\text{red}}, G_{i+1}^{\text{red}})$ rem $G_0^{\text{red}}, \ldots, G_{i+1}^{\text{red}}$
- Remark: $G_{i+2} = Spol(G_i, G_{i+1})$ rem G_i, G_{i+1} also gives a Gröbner basis.

Gröbner bases: concise representation – 1

The Gröbner basis is generated by A and B $\Longrightarrow$ there are relations between the G_i (redundant information)

- Reduced Gröbner basis:
  $G_{i+2}^{\text{red}} = Spol(G_i^{\text{red}}, G_{i+1}^{\text{red}})$ rem $G_0^{\text{red}}, \ldots, G_{i+1}^{\text{red}}$
- Remark: $G_{i+2} = Spol(G_i, G_{i+1})$ rem G_i, G_{i+1} also gives a Gröbner basis.

$$\begin{pmatrix} G_{i+1} \\ G_{i+2} \end{pmatrix} = M_i \begin{pmatrix} G_i \\ G_{i+1} \end{pmatrix}$$

Gröbner bases: concise representation – 1

The Gröbner basis is generated by A and B $\implies$ there are relations between the G_i (redundant information)

- Reduced Gröbner basis: $G_{i+2}^{\text{red}} = Spol(G_i^{\text{red}}, G_{i+1}^{\text{red}})$ rem $G_0^{\text{red}}, \ldots, G_{i+1}^{\text{red}}$
- Remark: $G_{i+2} = Spol(G_i, G_{i+1})$ rem G_i, G_{i+1} also gives a Gröbner basis.

$$\begin{pmatrix} G_{i+k} \\ G_{i+k+1} \end{pmatrix} = M_{i,k} \begin{pmatrix} G_i \\ G_{i+1} \end{pmatrix}$$

Gröbner bases: concise representation – 1

The Gröbner basis is generated by A and B $\implies$ there are relations between the G_i (redundant information)

- Reduced Gröbner basis: $G_{i+2}^{\text{red}} = Spol(G_i^{\text{red}}, G_{i+1}^{\text{red}})$ rem $G_0^{\text{red}}, \ldots, G_{i+1}^{\text{red}}$
- Remark: $G_{i+2} = Spol(G_i, G_{i+1})$ rem G_i, G_{i+1} also gives a Gröbner basis.

$$\begin{pmatrix} G_{i+k} \\ G_{i+k+1} \end{pmatrix} = M_{i,k} \begin{pmatrix} G_i \\ G_{i+1} \end{pmatrix}$$

$G_0 \cong A$, $G_1 \cong B$ and well-chosen $M_{i,k}$ hold all information about G.

Gröbner bases: concise representation – 1

The Gröbner basis is generated by A and $B \implies$ there are relations between the G_i (redundant information)

- Reduced Gröbner basis:
  $G_{i+2}^{\text{red}} = Spol(G_i^{\text{red}}, G_{i+1}^{\text{red}})$ rem $G_0^{\text{red}}, \ldots, G_{i+1}^{\text{red}}$
- Remark: $G_{i+2} = Spol(G_i, G_{i+1})$ rem G_i, G_{i+1} also gives a Gröbner basis.

$$\begin{pmatrix} G_{i+k} \\ G_{i+k+1} \end{pmatrix} = M_{i,k} \begin{pmatrix} G_i \\ G_{i+1} \end{pmatrix}$$

$G_0 \cong A$, $G_1 \cong B$ and well-chosen $M_{i,k}$ hold all information about G. Also, little information is required to compute the $M_{i,k}$.
$\cong$ (univariate) GCD computation on the main diagonals of A, B.

Gröbner bases: concise representation – 2

The coefficients of each G_i are needed to compute the reduction, but there are too many.

- Keep only enough coefficients to evaluate Q_i
- Then, rewrite $G_i = f(G_k, G_{k+1})$ to evaluate the remainder.

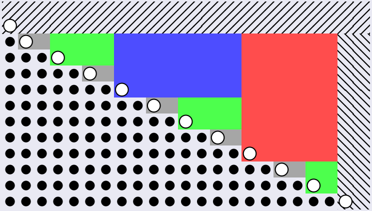
Gröbner bases: concise representation – 2

The coefficients of each G_i are needed to compute the reduction, but there are too many.

- Keep only enough coefficients to evaluate Q_i
- Then, rewrite $G_i = f(G_k, G_{k+1})$ to evaluate the remainder.

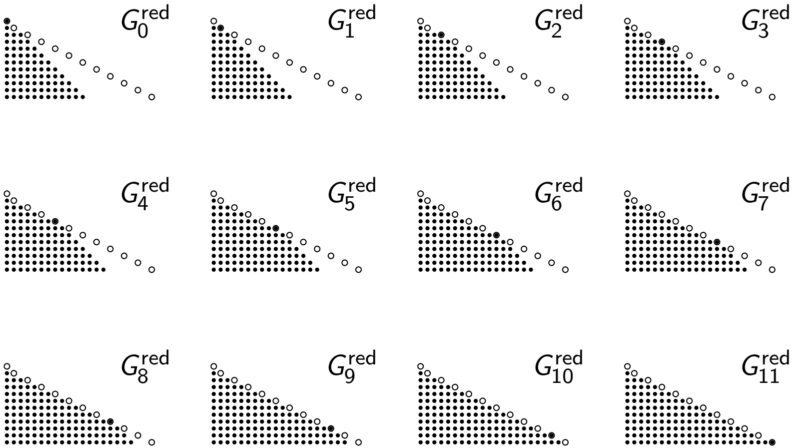
$\Longrightarrow$ Control the degree of the quotients.

Dichotomic selection strategy

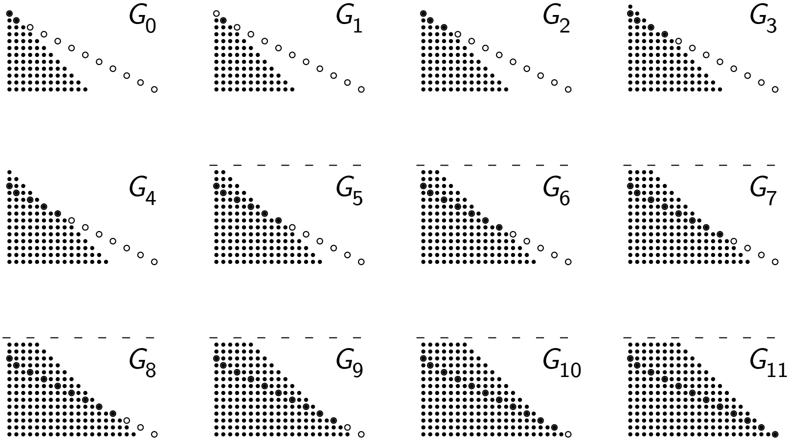


- $n/2$ quotients of degree 1
- $n/4$ quotients of degree 4
- $n/8$ quotients of degree 10
- ...
- $n/2^i$ quotients of degree $3 \times 2^{i-1} - 2$

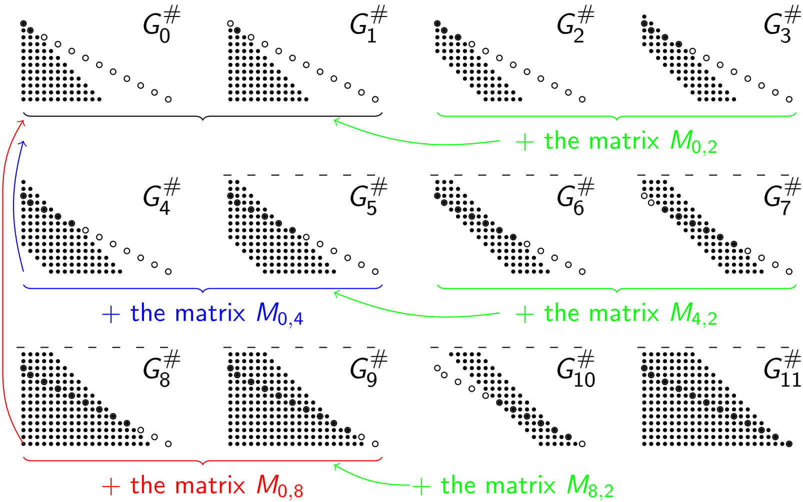
Gröbner bases: concise representation – Example



Gröbner bases: concise representation – Example



Gröbner bases: concise representation – Example



Outline

Polynomial reduction – Overview

Reminder

Equation $P = \sum_i Q_i G_i + R$ is too large: $\Theta(n^3)$ instead of $\tilde{O}(n^2)$

Adapt the algorithm to take advantage of the concise representation:

Polynomial reduction – Overview

Reminder

Equation $P = \sum_i Q_i G_i + R$ is too large: $\Theta(n^3)$ instead of $\tilde{O}(n^2)$

Adapt the algorithm to take advantage of the concise representation:

- Use the truncated elements $G_i^{\#}$ instead of G_i to reduce the size of the equation.

Polynomial reduction – Overview

Reminder

Equation $P = \sum_i Q_i G_i + R$ is too large: $\Theta(n^3)$ instead of $\tilde{O}(n^2)$

Adapt the algorithm to take advantage of the concise representation:

- Use the truncated elements $G_i^{\#}$ instead of G_i to reduce the size of the equation.
- The precision of $G_i^{\#}$ is chosen (by definition) sufficient to compute Q_i.

Polynomial reduction – Overview

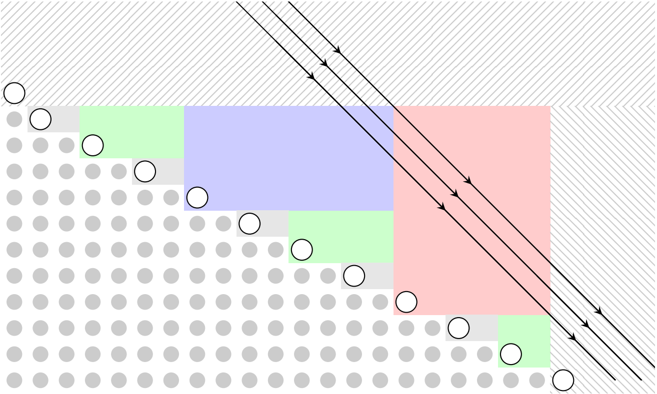
Reminder

Equation $P = \sum_i Q_i G_i + R$ is too large: $\Theta(n^3)$ instead of $\tilde{O}(n^2)$

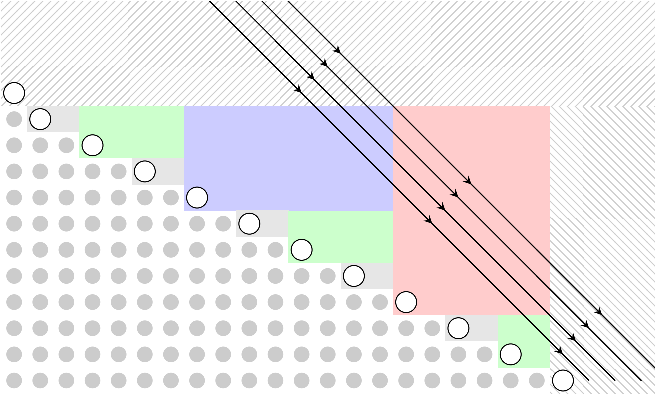
Adapt the algorithm to take advantage of the concise representation:

- Use the truncated elements $G_i^{\#}$ instead of G_i to reduce the size of the equation.
- The precision of $G_i^{\#}$ is chosen (by definition) sufficient to compute Q_i.
- Once Q_i is known, replace $Q_i G_i$ by $S_k G_k + S_{k+1} G_{k+1}$ to increase precision.

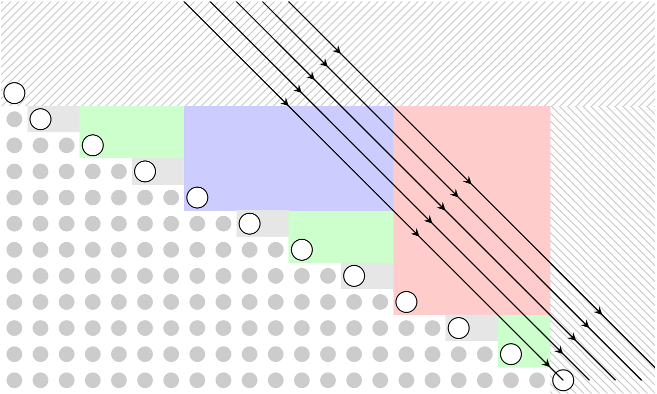
Polynomial reduction – Example



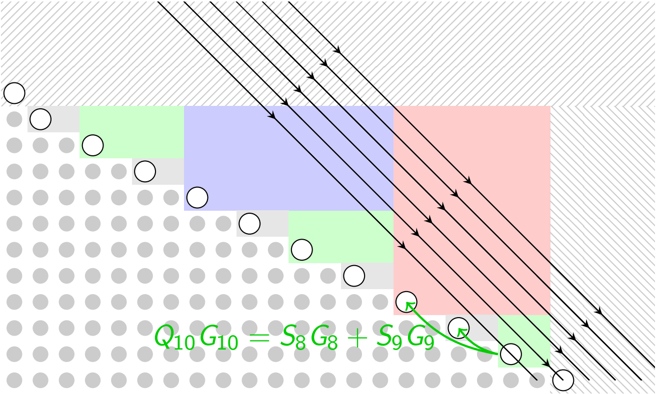
Polynomial reduction – Example



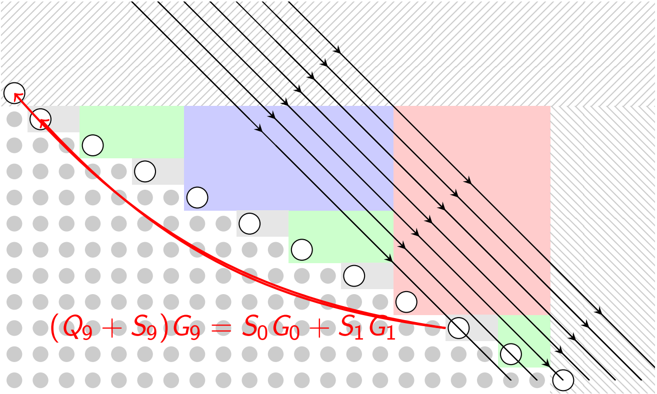
Polynomial reduction – Example



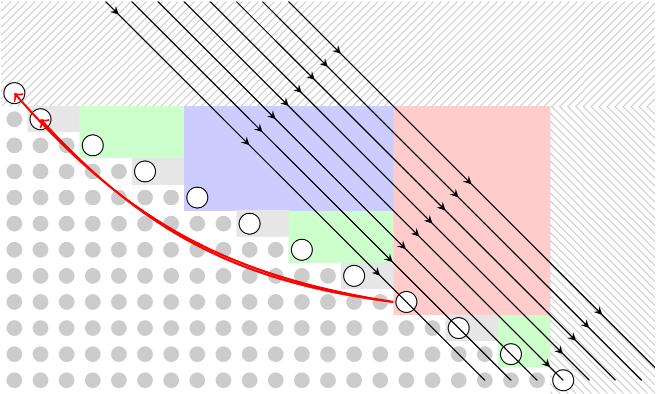
Polynomial reduction – Example



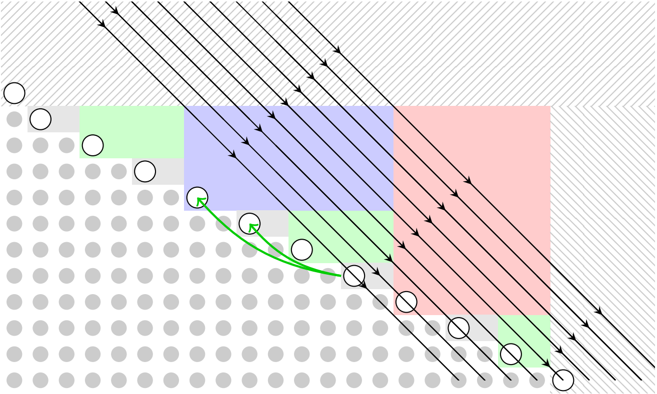
Polynomial reduction – Example



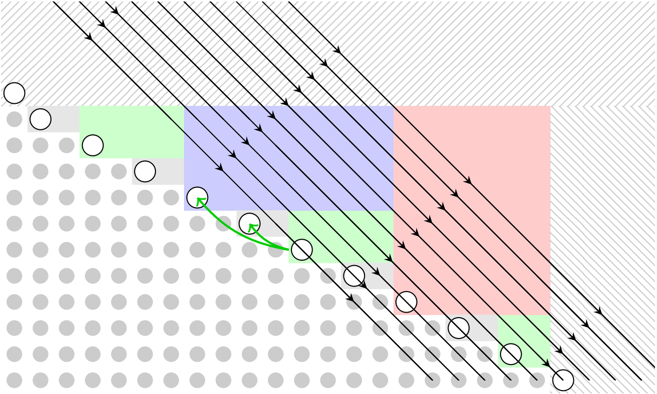
Polynomial reduction – Example



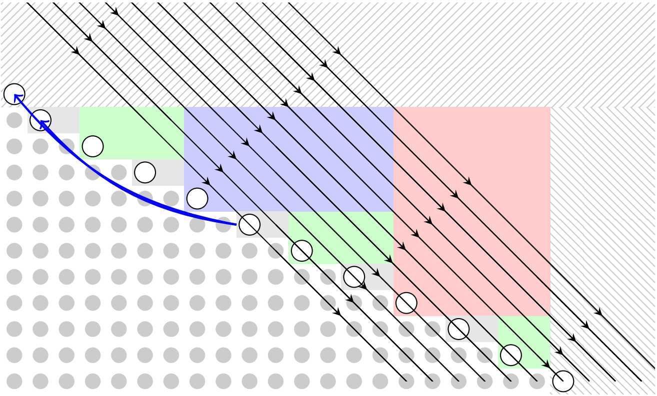
Polynomial reduction – Example



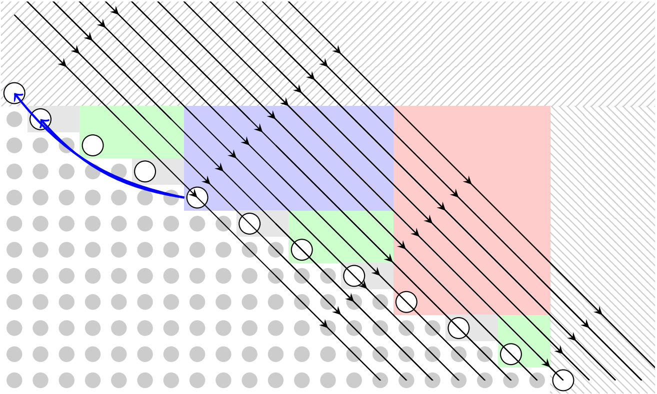
Polynomial reduction – Example



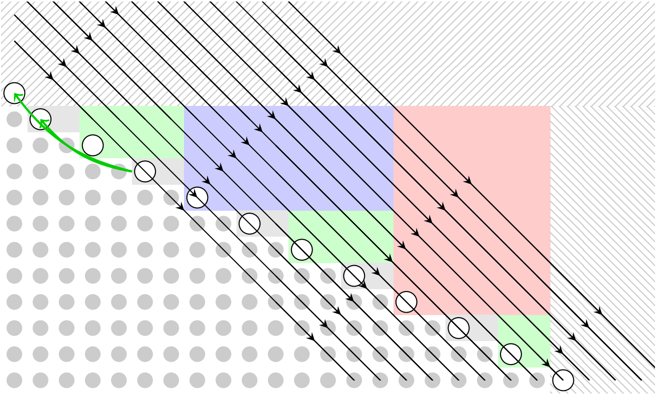
Polynomial reduction – Example



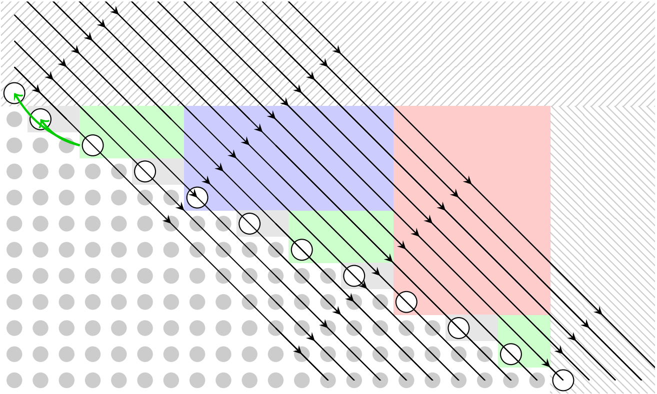
Polynomial reduction – Example



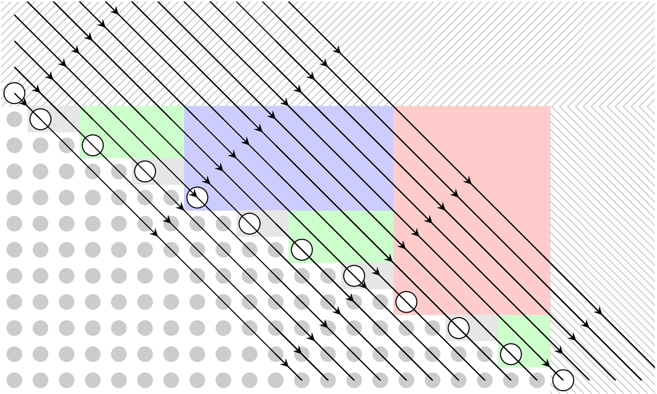
Polynomial reduction – Example



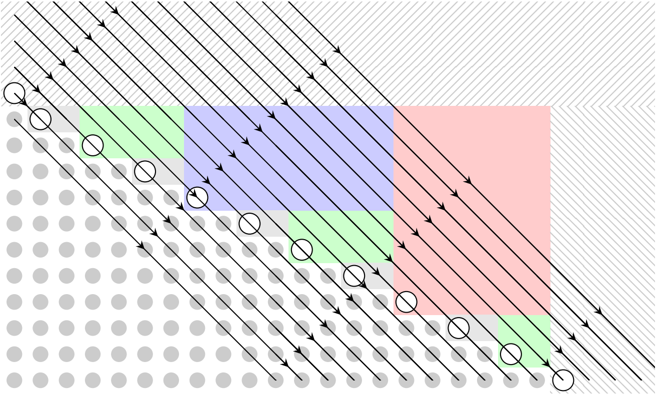
Polynomial reduction – Example



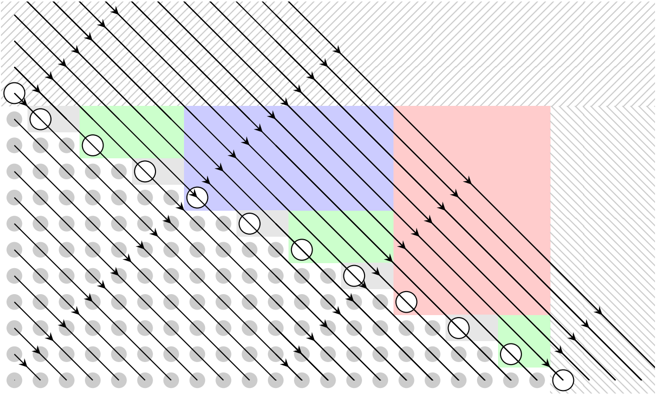
Polynomial reduction – Example



Polynomial reduction – Example



Polynomial reduction – Example



Gröbner basis

- Compute the concise representation: $\tilde{O}(n^2)$.
- Let $t_i := X^{\max(0,2i-1)} Y^{n-i} = \mathrm{lt}(G_i)$.
- Reduce t_i modulo G and let R_i be the remainder: $\tilde{O}(n^2)$ for each element $\implies$ $\tilde{O}(n^3)$.
- Set $G_i^{\mathrm{red}} := t_i - R_i$.

Gröbner basis

- Compute the concise representation: $\tilde{O}(n^2)$.
- Let $t_i := X^{\max(0,2i-1)} Y^{n-i} = \operatorname{lt}(G_i)$.
- Reduce t_i modulo G and let R_i be the remainder: $\tilde{O}(n^2)$ for each element $\Longrightarrow$ $\tilde{O}(n^3)$.
- Set $G_i^{\text{red}} := t_i - R_i$.

$\Rightarrow$ This is quasi-optimal since G has size $\Theta(n^3)$.

Conclusion

Main result

In a generic bivariate setting, there are quasi-optimal algorithms for polynomial reduction (in terms of the size of A, B, P) and to compute the reduced Gröbner basis (in terms of the output size)

In other words

- Structure of $\mathbb{K}[X, Y]/\langle A, B\rangle$ with quasi-optimal complexity.
- Quasi-optimal ideal membership test $P \in^? \langle A, B\rangle$.
- Quasi-optimal multiplication in $\mathbb{K}[X, Y]/\langle A, B\rangle$.

Conclusion

Main result

In a generic bivariate setting, there are quasi-optimal algorithms for polynomial reduction (in terms of the size of A, B, P) and to compute the reduced Gröbner basis (in terms of the output size)

In other words

- Structure of $\mathbb{K}[X, Y]/\langle A, B\rangle$ with quasi-optimal complexity.
- Quasi-optimal ideal membership test $P \in^? \langle A, B\rangle$.
- Quasi-optimal multiplication in $\mathbb{K}[X, Y]/\langle A, B\rangle$.

Generalization:

- Slightly degenerate cases ?
- More than 2 variables ?

Conclusion

Main result

In a generic bivariate setting, there are quasi-optimal algorithms for polynomial reduction (in terms of the size of A, B, P) and to compute the reduced Gröbner basis (in terms of the output size)

In other words

- Structure of $\mathbb{K}[X, Y]/\langle A, B\rangle$ with quasi-optimal complexity.
- Quasi-optimal ideal membership test $P \in^? \langle A, B\rangle$.
- Quasi-optimal multiplication in $\mathbb{K}[X, Y]/\langle A, B\rangle$.

Generalization:

- Slightly degenerate cases ? $\rightarrow$ seems feasible.
- More than 2 variables ? $\rightarrow$ much more difficult.

Conclusion

Proof-of-concept implementation (in Sage) at
https://www.lix.polytechnique.fr/~larrieu/

- Mainly intended as correctness proof.
- Missing (fast) implementation of some primitives $\Longrightarrow$ reduction is not competitive in practice.
- Computing the concise representation is faster than Sage's builtin Gröbner basis library for $n \geqslant 160$.

Conclusion

Proof-of-concept implementation (in Sage) at
https://www.lix.polytechnique.fr/~larrieu/

- Mainly intended as correctness proof.
- Missing (fast) implementation of some primitives $\Longrightarrow$ reduction is not competitive in practice.
- Computing the concise representation is faster than Sage's builtin Gröbner basis library for $n \geqslant 160$.

Thank you for your attention