Signature-based Gröbner basis computations for approximate input

Vasily Galkin
galkin-vv@yandex.ru

Moscow State University
Faculty of Mechanics and Mathematics
Department of Higher Algebra

Main goals

Signature-based Gröbner basis computation algorithms: application to solving polynomial systems with inexact input data in $\mathbb{C}$.

1. Definitions
2. Problems of application
3. Known methods
4. Inexact zero
   - Approximate arithmetic instead of floats
   - Non-invertible approximate elements classification
   - Examples
5. New ideas
   - Strict modular method for approximate Gröbner basis
   - Strict modular method for approximate linear system
   - No-restart TSV method for signature-based algorithms
   - "Strict modular" and "no-restart TSV" methods: system solving

Signature-based algorithms

- Compute Gröbner basis G for ideal generated by given polynomials $I = (f_1, \dots, f_n)$ in polynomial ring P
- Are one of the most effective algorithms for Gröbner basis computation
- Track *signature* of every polynomial p, corresponding to "highest" element of input-representation

$$p = \sum_{i=1}^{n} q_i f_i$$

- Using signature-based rules maintain an ordered queue of polynomials to process

Polynomial systems with inexact input data

- Arises from different areas (statistics, video analytics, etc.)
- Formulated as a set of polynomial equations $f_i = 0$ with *approximate number* coefficients

Definitions

Approximate (complex) number is a pair (c, ε), $c \in \mathbb{C}$, $\varepsilon \in \mathbb{R}$
Specialization of complex number a is $\widehat{a} \in \mathbb{C}$, $|\widehat{a} - c| < \varepsilon$

Requirements for approximate Gröbner basis used to solve polynomial system: a set G of polynomials with approximate coefficients satisfying following:

> *The approximate Gröbner basis* $GB(f_i)$ *should have specialization* $\widehat{GB(f_i)}$ *equal to exact Gröbner basis* $GB\left(\widehat{f_i}\right)$ *for any specialization* $\widehat{f_i}$ *of input data* f_i.

The root of problems: test for equality to zero

- Algorithm branching points: is $a = 0$ in $f = ax^2 + by^2$?

formal	determine $\mathrm{HM}(f)$: is it x^2 or y^2 ?
practical	can we reduce polynomial $x^3 + xy$ by f ?

- Branching points in signature-based algorithms are unavoidable:
  - need test coefficients of non-reducible highest monomials (HC)
  - specific polynomial selection is required for correctness
- Approximated numbers having zero specialization
  - may be input data coefficients
  - may appear as a result of inexact computations
  - don't allow comparison with zero with boolean result

Gröbner basis algorithms dealing with approximate data in $\mathbb{R}$

- Comprehensive Gröbner basis computation
  - treat coefficients as symbols and use symbolic computation
  - slow arithmetic with huge symbolic expressions
- Exact Gröbner basis computation in $\mathbb{Q}$
  - selects some $\mathbb{Q}$-specialization of approximate number
  - slow arithmetic with huge denominators
  - solves task only for single specialization
- Approximate Gröbner basis computation using numeric or modular methods for zero-comparison
  - result correctness is not guaranteed even for single specialization
- Approximate Gröbner basis computation changing monomial order to skip HC zero-comparison (TSV, etc.)
  - requires restart from scratch of signature-based algorithms

Approximate arithmetic instead of floats

Addition $(c_1, \varepsilon_1) + (c_2, \varepsilon_2) = (c_1 + c_2, \varepsilon_1 + \varepsilon_2)$

Multiplication $(c_1, \varepsilon_1) \times (c_2, \varepsilon_2) = (c_1 c_2, \varepsilon_1 |c_2| + \varepsilon_2 |c_1| + \varepsilon_1 \varepsilon_2)$

Subtraction based on addition and multiplication by -1:

$$(c_1, \varepsilon_1) - (c_2, \varepsilon_2) = (c_1 - c_2, \varepsilon_1 + \varepsilon_2)$$

Inversion defined only for numbers not having zero specialization, or equivalently satisfying $|c| - \varepsilon > 0$:

$$\frac{1}{(c, \varepsilon)} = \left(\frac{1}{c}, \frac{\varepsilon}{|c| \left(|c| - \varepsilon \right)} \right)$$

- Track all possible specializations
- Slower than floating point arithmetic only by a constant factor

Non-invertible approximate elements classification

Definitions

- *symbolic zero* – corresponding computations in $\mathbb{C}$ for any input data specialization give exact zero.
- *input-inspired zero* – $\mathbb{C}$-computations for some but not all input data specializations give exact zero
- *computation-introduced zero* – $\mathbb{C}$-computations for input data specializations never give exact zero

Example

$$
\begin{aligned}
f_1 &= y^2 z + a \\
f_2 &= y^2 z^2 + xz + 1 \\
f_3 &= y^3 z + xy + 1 \\
f_4 &= f_2 - z f_1 = xz - az + 1 \\
f_5 &= f_3 - y f_1 = xy - ay + 1 \\
f_6 &= z f_5 - y f_4 = (a - a)\, yz + z - y
\end{aligned}
$$

$HC(f_6)$ is symbolic zero.

Example

$$
\begin{aligned}
f_1 &= y^2 z + z^2 + az \\
f_2 &= xyz \\
f_3 &= xy^2 + bx + 1 \\
f_4 &= -(y f_2 - x f_1) = xz^2 + axz \\
f_5 &= (z f_3 - x f_1) + f_4 = ((b - a) + a) xz + z
\end{aligned}
$$

$HC(f_5)$ may be input-inspired or computation-introduced zero.
Concrete zero type depends on a and b values.

Strict modular method for approximate Gröbner basis

Main idea: compute in finite field and treat all zeros as symbolic.

- Initially selected prime module
  - known methods: prime is assumed to be "big enough"
  - strict method: initially selected prime used only to compute a hypothesis of the linear system form; rechecked later
- Result approximate coefficients come from linear system solution
  - avoid computation-introduced zeros arose from $a - a$ expressions
  - detect all symbolic zeros using modular computation (like known method)
  - allow to estimate correctness probability based on system properties and prime number
  - separately select prime module for every system to ensure required correctness probability

Strict modular method for approximate linear system

Zero-testings are needed during Gaussian elimination. Modular computations allows to detect all symbolic zeros but may detect superfluous ones leading to error. To solve a system with error probability $\leqslant 2\alpha$ we should randomly take a prime number from a set of at least N_p prime numbers greater than P_0.

- $P_0 = \sqrt{\frac{R2^R V}{\alpha}}$, $N_p = \frac{R2^R \log_{P_0} 2Z_0}{\alpha}$

R number of rows in a system

V number of approximate coefficients in a system

Z_0 maximal denominator of exact $\mathbb{Q}$ coefficient

Fact

Required number of bits is linear in the number of system rows. Method fails on input-inspired zeros.

No-restart TSV method for signature-based algorithms

TSV main idea: add monomial y and input polynomial $x_1^{i_1} \cdots x_m^{i_m} - y$ to avoid zero testing of $x_1^{i_1} \cdots x_m^{i_m}$ coefficient.
Result is GB of extended ideal $I^e = (I, x_1^{i_1} \cdots x_m^{i_m} - y', \ldots)$

- new polynomial
  - original TSV method: added with small signature and restart algorithm from scratch
  - no-restart TSV method: added with "just-before-current" signature and continue processing
- real-weighted order
  - real and monomial parameter for every input polynomial
  - added polynomial signatures can be configured by parameters
  - allows addition of polynomial with signature just before given
  - applies only for signature order independent algorithms

“Strict modular” and “no-restart TSV” methods: system solving

1. Compute a Gröbner basis of extended ideal I^e with a signature-based algorithm using real-weighted order
    1. Compute approximate result by strict modular method
    2. If it fails on non-invertible element, proceed by no-restart TSV
2. Using $\mathrm{GB}\,(I^e)$ and without using $\mathrm{GB}\,(I)$ determine one of:
    1. P/I is vector space; find it's basis B, normal form operator ϕ
    2. I is not zero-dimensional; then the algorithm is inapplicable
3. Solve the system by “action matrix method” reformulated in terms of B and ϕ instead of $\mathrm{GB}\,(I)$.

- Problems of known methods are avoided
    - arithmetic operations has predictable complexion
    - no restarts from scratch
    - all input data specializations are considered
    - the probability of error strictly estimated
- Open questions in complexity estimation are introduced
    - huge prime modules may appear
    - influence of real-weighted order on speed is unknown