Sparse $\sqrt{\text{FGLM}}$ using the block Wiedemann algorithm

Seung Gyu Hyun, Vincent Neiger, Hamid Rahkooy, Éric Schost

University of Waterloo, DTU Compute

Introduction

- Compute the Gröbner basis for degrevlex ordering first (fast) and convert to lex ordering (better structure)

Main Problem

Input:

- Zero-dimensional ideal $I \subset \mathbb{K}[x_1, \ldots, x_n]$ by means of a monomial basis $\mathbb{B} \subset Q$, $Q = \mathbb{K}[x_1, \ldots, x_n]/I$
- Multiplication matrices $T_1, \ldots, T_n \in \mathbb{K}^{D \times D}$ of $x_1, \ldots, x_n$, with $D = dim_{\mathbb{K}}(Q)$

Output:

- Lex Gröbner basis of $\sqrt{I}$

Assumptions

- Base field is larger than D
- x_n *separates* the points of $V(I)$
- Ensured by a generic change of coordinates
- Under assumption, $\sqrt{I}$ is in shape position
- $I \subset \mathbb{K}[x_1, \ldots, x_n]$ is in *shape position* if its Gröbner basis has the form $(x_1 - R_1(x_n), \ldots, x_{n-1} - R_{n-1}(x_n), R(x_n))$

Sparse FGLM

- Sparse FGLM algorithm of [1] computes lex basis of an ideal I when I is in shape position
- Exploits the sparsity of T_i's
- Difficult to parallelize

Differences

- If I is not in shape position, Sparse FGLM uses Berlekamp-Massey-Sakata to compute the lex basis
- We compute lex basis of $\sqrt{I}$ (weaker), which is in shape position by assumption

Block Sparse $\sqrt{\text{FGLM}}$

- Inspired by Coppersmith's block Wiedemann Algorithm [2]
- **Key idea:** Compute sequences of small matrices that require less terms than linear sequences

Algorithm

1. Choose $U, V \in \mathbb{K}^{m \times D}$, where m is the number of threads supported
2. Compute $d = \frac{D}{M}$ and $s = (UT_n^iV^t)_{0 \le i < 2d}$
3. Compute $S = MatrixBerlekampMassey(s)$, and $N = S \sum s^{(i)}/x^{i+1}$
4. Find the Smith form of S, ASB with invariant factors $I_1, \ldots I_D$. Set $a = \left[\frac{I_D b_1}{I_1} \frac{I_D b_2}{I_2}, \cdots, \frac{I_D b_{D-1}}{I_{D-1}}, b_D\right] A$, where b_i is the i^{th} entry of the last row of B
5. Set N^* to be the $(m, 1)$-th entry of aN and R_n as the square-free part of I_D
6. For $j = 1 \ldots n-1$:

   6a. Compute $s_j = (UT_n^iT_jV^t)_{0 \le i < d}$ and $N_j = S \sum s_j^{(i)}/x^{i+1}$

   6b. Set $N_j^* = \tilde{u}N_j$ and R_j as N_j^*/N^* mod R_n written in x_n
7. Return $(x_1 - R_1, \ldots, x_{n-1} - R_{n-1}, R_n)$

- Analysis for Coppersmith's algorithm done by [Villard '97], [3], other useful properties [Kaltofen '95],[Kaltofen, Yuhasz '06]; Correctness of the algorithm also follows from [5]

Example

Input: $I = \langle f_1^2, f_2^2, f_3 \rangle \subset GF(9001)[x_1, x_2, x_3]$ and $D = 32$, with

$$f_1 = -4022x_1^2 - 2799x_1x_2 + \ldots, \quad f_2 = -4319x_1^2 - 711x_1x_2 + \ldots, \quad f_3 = 4199x_1^2 + 2325x_1x_2 + \ldots$$

Step 1: Choose $m = 2$ and $U, V \in GF(9001)^{32 \times 2}$

$$U = \begin{bmatrix} 1898 & 6830 & 3494 & 169 & 7991 & 3352 \ldots \\ 3161 & 8858 & 8467 & 5882 & 8037 & 3726 \ldots \end{bmatrix} \quad V = \begin{bmatrix} 7595 & 8416 & 2285 & 8351 & 550 & 7012 \ldots \\ 823 & 5686 & 6539 & 7884 & 7105 & 3427 \ldots \end{bmatrix}^t$$

Step 2 & 3: $d = 16$,

$$s = \left(\begin{bmatrix} 31 & 6977 \\ 1178 & 1695 \end{bmatrix}, \begin{bmatrix} 8212 & 1663 \\ 4811 & 4837 \end{bmatrix} \ldots \right) S = \begin{bmatrix} x^{16} + \ldots & 423x^{15} + \ldots \\ 6426x^{15} + \ldots & x^{16} + \ldots \end{bmatrix} N = \begin{bmatrix} 6191/x^{16} + \ldots & 8101/x^{16} + \ldots \\ 7116/x^{16} + \ldots & 2129/x^{16} + \ldots \end{bmatrix}$$

Step 4 & 5: $I_1 = x^8 + 6990x^7 + 191x^6 + \ldots$ and $I_2 = x^{24} + 2968x^{23} + 8589x^{22} + \ldots$, with

$$a = \left[2575x^7 + 4809x^6 + \ldots \quad x^8 + 3391x^7 + \ldots\right], \; N^* = 1178x^{23} + 8727x^{22} + \ldots, \; R_3 = x^8 + 6990x^7 + \ldots$$

Step 6: For $j = 1$, $N_1^* = 1178x^{23} + 8727x^{22} + \ldots$ and $R_1 = 7964x^7 + 4071x^6 + \ldots$

For $j = 2$, $N_2^* = 6587x^{23} + 3987x^{22} + \ldots$ and $R_2 = 1443x^7 + 7818x^6 + \ldots$

Parallel Computations

- Bottleneck is computing the sequence $(UT_n^i)_{0 \le i < 2d}$
- Can parallelize by computing the sequences $(U_1T_n^i), \ldots, (U_mT_n^i)$ separately, where U_i is the i^{th} row of U
- When $m = 1$, same computation as Sparse FGLM

Conclusion

References

[1] J.-C. Faugère and C. Mou. Sparse FGLM algorithms. *Journal of Symbolic Computation*, 80(3):538–569, 2017.

[2] Don Coppersmith. Solving linear equations over GF(2): block Lanczos algorithm. *Linear Algebra and its Applications*, 192:33–60, 1993.

[3] E. Kaltofen and G. Villard. On the complexity of computing determinants. *Comput. Complexity*, 13(3-4):91–130, 2004.

[4] J.-C. Faugère, P. Gianni, D. Lazard, and T. Mora. Efficient computation of zero-dimensional Gröbner bases by change of ordering. *Journal of Symbolic Computation*, 16(4):329–344, 1993.

[5] A. Bostan, B. Salvy, and É. Schost. Fast algorithms for zero-dimensional polynomial systems using duality. *Applicable Algebra in Engineering, Communication and Computing*, 14:239–272, 2003.

[6] V. Neiger. *Bases of relations in one or several variables: fast algorithms and applications.* PhD thesis, École Normale Supérieure de Lyon, November 2016.

[7] J.-C. Faugère, P. Gaudry, L. Huot, and G. Renault. Polynomial Systems Solving by Fast Linear Algebra. https://hal.archives-ouvertes.fr/hal-00816724, 2013.