Sparse FGLM using the block Wiedemann algorithm

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

University of Waterloo, DTU Compute

Introduction

- Gröbner basis of an ideal is essential in solving systems of polynomials
- Orderings such as degree reverse lexicographical ordering *(degrevlex)* make computing the Gröbner basis faster
- Orderings such as lexicographical order *(lex)* make finding solution coordinates easier
- Compute first for degrevlex ordering and convert to lex ordering
- Sparse FGLM algorithm of [1] computes lex basis of a radical ideal I when I is in shape position (although they also provide an algorithm for non-radical ideals)
- $I \subset \mathbb{K}[x_1, \ldots, x_n]$ is in *shape position* if its Gröbner basis has the form $(x_1 - R_1(x_n), \ldots, x_{n-1} - R_{n-1}(x_n), R(x_n))$
- Difficult to parallelize

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 $\sqrt{I}$

Block Sparse FGLM

- Inspired by Coppersmith's block Wiedemann Algorithm [3]
- **Key idea:** Compute sequences of small matrices that require less terms than linear sequences

Algorithm

1. Choose $U, V \in \mathbb{K}^{D\times m}$, where m is the number of threads supported
2. Compute $s = (UT_n^iV)_{0\le i<\frac{2D}{M}}$, $S = MatrixBerlekampMassey(s)$, and $N = S\sum s^{(i)}/x^{i+1}$
3. Find the Smith form of S, $D = ASB$ with invariant factors $I_1, \ldots I_D$. Set $\tilde{u} = \left[\frac{I_D b_1}{I_1} \frac{I_D b_2}{I_2}, \cdots, \frac{I_D b_{D-1}}{I_{D-1}}, b_D\right] A$, where b_i is the i^{th} entry of the last row of B
4. Set n^* to be the $(m, 1)$-th entry of $\tilde{u}N$ and R_n as I_D written in x_n
5. For $j = 1 \ldots n-1$:
   - 5a. Compute $s_j = (UT_n^iT_jV)_{0\le i<\frac{D}{M}}$ and $N_j = S\sum s_j^{(i)}/x^{i+1}$
   - 5b. Set $n_j = \tilde{u}N_j$ and R_j as n_j/n^* mod R_n written in x_n
6. Return $(x_1 - R_1, \ldots, x_{n-1} - R_{n-1}, R_n)$

- Analysis for Coppersmith's algorithm has been provided by [Kaltofen '95], [Villard '97], [Kaltofen, Villard '04], [Kaltofen, Yuhasz '06]

Example

We will demonstrate our new algorithm by running it on $I = \langle f_1, f_2, f_3\rangle \subset \mathbb{K}[x_1, x_2, x_3]$,

$$f_1 = 3426x_1^2 - 4443x_1x_2 + 2004x_2^2 + 2335x_1x_3 - 74x_2x_3 + 4215x_3^2 - 1405x_1 + 4108x_2 - 1838x_3 - 2741$$
$$f_2 = -4303x_1^2 - 1401x_1x_2 - 2604x_2^2 + 2745x_1x_3 + 3440x_2x_3 + 3331x_3^2 + 2112x_1 - 271x_2 - 2272x_3 - 3090$$
$$f_3 = -4160x_1^2 + 1056x_1x_2 - 252x_2^2 - 2842x_1x_3 - 3643x_2x_3 + 3024x_3^2 + 3353x_1 + 3908x_2 - 426x_3 + 4197$$

We choose $m = 2$ and two random matrices (with coefficients in [0,…999] for this specific example)

$$U = \begin{bmatrix} 568 & 651 & 852 & 933 & 279 & 835 & 446 & 135 \\ 485 & 707 & 441 & 238 & 678 & 552 & 95 & 900 \end{bmatrix} \quad V = \begin{bmatrix} 338 & 147 & 24 & 526 & 549 & 806 & 741 & 966 \\ 243 & 637 & 563 & 545 & 580 & 432 & 544 & 165 \end{bmatrix}^t$$

Computing the minimal generating polynomial of $s = (UT_3^iV)_{0\le i<8}$,

$$S = \begin{bmatrix} x^4 + 5848x^3 + 1193x^2 + 5800x + 4050 & 5414x^3 + 6409x^2 + 223x + 783 \\ 4469x^3 + 7812x^2 + 80x + 554 & x^4 + 3102x^3 + 4076x^2 + 1871x + 3985 \end{bmatrix}$$

After steps 4 and 5,

$$n = 320x_3^7 + 3312x_3^6 + 38x_3^5 + 763x_3^4 + 5895x_3^3 + 6024x_3^2 + 8927x_3 + 1804$$
$$R_3 = x_3^8 + 8950x_3^7 + 8272x_3^6 + 2637x_3^5 + 3062x_3^4 + 7018x_3^3 + 6992x_3^2 + 8980x_3 + 7724$$

For $j = 1$,

$$n_1 = 7981x_3^7 + 7201x_3^6 + 1005x_3^5 + 1044x_3^4 + 3205x_3^3 + 6671x_3^2 + 7423x_3 + 832$$
$$R_1 = 4200x_3^7 + 5082x_3^6 + 6769x_3^5 + 5671x_3^4 + 4288x_3^3 + 8885x_3^2 + 7423x_3 + 4930$$

For $j = 2$,

$$n_2 = 4648x_3^7 + 377x_3^6 + 5551x_3^5 + 1738x_3^4 + 2653x_3^3 + 7064x_3^2 + 3229x_3 + 8719$$
$$R_2 = 1168x_3^7 + 5878x_3^6 + 2896x_3^5 + 4452x_3^4 + 3821x_3^3 + 7586x_3^2 + 7952x_3 + 28$$

Conclusion

References

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

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

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

[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.