

# 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  $I \subset \mathbb{K}[x_1, \dots, x_n]$  by means of a monomial basis  $\mathbb{B} \subset Q$ ,  $Q = \mathbb{K}[x_1, \dots, x_n]/I$
- Multiplication matrices  $T_1, \dots, T_n \in \mathbb{K}^{D \times D}$  of  $x_1, \dots, x_n$ , 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
- $I \subset \mathbb{K}[x_1, \dots, x_n]$  is in *shape position* if its Gröbner basis has the form  $(x_1 - R_1(x_n), \dots, x_{n-1} - R_{n-1}(x_n), R(x_n))$

## 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 $\sqrt{\text{FGLM}}$

**Key idea:** use sequences of small matrices, requires less terms than scalar sequences (cf. Coppersmith's block Wiedemann Algorithm)

## Algorithm

Choose  $U, V \in \mathbb{K}^{m \times D}$ , where  $m$  is the number of threads supported  
 $s = (UT_n^i V^t)_{0 \leq i < 2d}$ , with  $d = \frac{D}{m}$   
 $S = \text{MatrixBerlekampMassey}(s)$ ,  $N = S \sum_{i \geq 0} \frac{s_i}{x^{i+1}}$   
 $A, S, B = \text{SmithForm}(S)$ , with invariant factors  $I_1, \dots, I_d$   
 $a = \begin{bmatrix} I_d b_1 & I_d b_2 & \dots & I_d b_{d-1} & b_d \end{bmatrix} A$ , where  $b_i$  is the  $i^{\text{th}}$  entry of the last row of  $B$   
 $N^* = (m, 1)$ -th entry of  $aN$   
 $R_n = \text{SquareFreePart}(I_d)$   
For  $j = 1 \dots n - 1$ :  
 $s_j = (UT_n^i T_j V^t)_{0 \leq i < d}$  and  $N_j = S \sum_{i \geq 0} \frac{s_{ji}}{x^{i+1}}$   
 $R_j = aN_j / N^* \bmod R_n$

**Correctness:** follows from the analysis of Coppersmith's algorithm by [Villard '97] and [3], and generating series properties in [5]. See also [Kaltofen '95], [Kaltofen, Yuhasz '06].

## Example

**Input:**  $I = \langle f_1^2, f_2^2, f_3 \rangle \subset \mathbb{F}_{9001}[x_1, x_2, x_3]$  of degree  $D = 32$ , with

$$f_1 = 4979x_1^2 + 6202x_1x_2 + \dots, \quad f_2 = 4682x_1^2 + 8290x_1x_2 + \dots, \quad f_3 = 4199x_1^2 + 2325x_1x_2 + \dots$$

**Step 1** with  $m = 2$

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

**Step 2 & 3** with  $d = 16$

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

**Step 4:**  $a = [2575\mathbf{x}^7 + \dots \quad \mathbf{x}^8 + \dots]$

**Step 5:**  $[N^*] = [2575\mathbf{x}^7 + \dots \quad \mathbf{x}^8 + \dots] \begin{bmatrix} 6191\mathbf{x}^{15} + \dots \\ 7116\mathbf{x}^{15} + \dots \end{bmatrix} = [1178\mathbf{x}^{23} + 8727x^{22} + \dots]$

**Step 6** for  $j = 1$

$$s_1 = \left( \begin{bmatrix} 3029 & 8903 \\ 1538 & 5610 \end{bmatrix}, \begin{bmatrix} 1914 & 3734 \\ 5221 & 5431 \end{bmatrix}, \dots \right) \implies N_1 = \begin{bmatrix} 1374\mathbf{x}^{15} + \dots & 3271\mathbf{x}^{15} + \dots \\ 4027\mathbf{x}^{15} + \dots & 1575\mathbf{x}^{15} + \dots \end{bmatrix}$$

**Step 7:**  $[N_1^*] = [2575\mathbf{x}^7 + \dots \quad \mathbf{x}^8 + \dots] \begin{bmatrix} 1374\mathbf{x}^{15} + \dots \\ 4027\mathbf{x}^{15} + \dots \end{bmatrix} = [1538\mathbf{x}^{23} + 6498x^{22} + \dots]$

## Parallel Computations

- Bottleneck is computing the sequence  $(UT_n^i)_{0 \leq i < 2d}$
- Can parallelize by computing the sequences  $(U_1 T_n^i), \dots, (U_m T_n^i)$  separately, where  $U_i$  is the  $i^{\text{th}}$  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.