Sparse FGLM using the block Wiedemann algorithm

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

Overview Computing the Gröbner basis of an ideal with respect to a term ordering is an essential step in solving systems of polynomials (in what follows, we restrict our attention to systems with finitely many solutions.) Certain term orderings, such as the degree reverse lexicographical ordering (degrevlex), make the computation of the Gröbner basis faster, while other orderings, such as the lexicographical ordering (lex), make it easier to find the coordinates of the solutions. Thus, one typically first compute a Gröbner basis using a degrevlex ordering, then converts it to either a lex Gröbner basis, or a related representation, such as Rouillier's Rational Univariate Representation [8].

Consider a zero-dimensional ideal $I \subset \mathbb{K}[x_1, \dots, x_n]$, given by means of a monomial basis $\mathscr{B}$ of $Q = \mathbb{K}[x_1, \dots, x_n]/I$, together with the multiplication matrices $T_1, \dots, T_n \in \mathbb{K}^{D \times D}$ of respectively $x_1, \dots, x_n$ in Q in $\mathscr{B}$, with $D = \dim_{\mathbb{K}}(Q)$. In all that follows, we assume that x_n *separates* the points of $V(I)$. However, we do not assume that I is radical, or that it is in *shape position* (that is, that its lex basis for the order $x_1 > \cdots > x_n$ has the form $(x_1 - G_1(x_n), \dots, x_{n-1} - G_{n-1}(x_n), G_n(x_n)))$; note that our assumption can be ensured by a generic change of coordinates, while the shape position one may not hold for any choice of coordinates. Under our assumption, the *radical* of I is in shape position, with Gröbner basis $R = (x_1 - R_1(x_n), \dots, x_{n-1} - R_{n-1}(x_n), R_n(x_n))$.

The FGLM algorithm [4] computes the Gröbner basis of I for the lex order $x_1 > \cdots > x_n$ with a runtime cubic in the dimension D. Although recent work has reduced the runtime exponent from 3 to the exponent of matrix multiplication ω [3, 7], linear algebra techniques based on duality often turn out to be more efficient. The basic idea behind these techniques is to compute values of one (or possibly several) linear forms $\ell : Q \to \mathbb{K}$ at well-chosen elements of Q, and to deduce our output by solving some well-structured linear system (typically, Hankel).

Several forms of these ideas exist. The Rational Univariate Representation algorithm takes for ℓ the trace $\mathrm{tr} : Q \to \mathbb{K}$ and computes values of the form $\mathrm{tr}(x_n^i)$ and $\mathrm{tr}(x_j x_n^i)$; from there, the output of the algorithm is equivalent to the data of R, together with the multiplicities of all points. Values such as $\mathrm{tr}(x_n^i)$ can be computed as $vT_n^i e_n$, where v is the row vector of the traces of the monomial basis $\mathscr{B}$, and e_n is the column vector of coordinates of x_n on $\mathscr{B}$. Computing the values of the trace on $\mathscr{B}$ is however costly, so several algorithms use random linear forms instead. In [1], the authors show how we can recover the Gröbner basis of R of $\sqrt{I}$ (together with the nil-indices of all points) from the values $\ell(x_n^i)$ and $\ell(x_j x_n^i)$, for a random ℓ, whereas Faugère and Mou show in [5] how to compute the basis G of I itself using similar sets of values, when I is in shape position (in general, the algorithm falls back on Berlekamp-Massey-Sakata techniques). Quite importantly, they also demonstrate that exploiting the sparsity of matrices $T_1, \dots, T_n$ is critical for the efficiency of these algorithms.

The computation of sequences such as $\ell(x_n^i)$ is difficult to parallelize. In this work, in the continutation of [1], we use techniques inspired by Coppersmith's block Wiedemann algorithm [2] to compute the Gröbner basis R of $\sqrt{I}$.

Computing R_n. We choose an integer m, random matrices $u \in \mathbb{K}^{m \times D}$, $v \in \mathbb{K}^{D \times m}$, and we consider the matrix sequence $s = (uT_n^i v)_{i \geq 0}$. For generic matrices u, v, Theorem 2.12 in [6] shows that if $S \in \mathbb{K}[x]^{m \times m}$ is a minimal generating polynomial of $(s_i)_{i \geq 0}$, its determinant is a multiple of the minimal polynomial of x_n in Q, and divides its characteristic polynomial. This polynomial must then factor as $\prod_{\alpha=(\alpha_1,\ldots,\alpha_n)\in V(I)} (x - \alpha_n)^{e_\alpha}$, for certain positive integers e; in particular, its squarefree part (written in the variable x_n) is the last polynomial R_n in R.

Using u-resultant techniques. We propose to recover the whole Gröbner basis R of $\sqrt{I}$ by computing a truncation of a polynomial akin to the u-resultant; we show here this idea for the computation of R_{n-1}.

Let λ be a new variable and consider the matrix $T_\lambda = T_n + \lambda T_{n-1}$; this is the multiplication matrix by $x_n + \lambda x_{n-1}$ in $Q \otimes_{\mathbb{K}} \mathbb{K}(\lambda)$. Applying the above algorithm to the sequence $s_\lambda = (uT_\lambda^i v)_{i \geq 0}$ gives us the polynomial $M_\lambda = \prod_{\alpha=(\alpha_1,\ldots,\alpha_n)\in V(I)} (x - (\alpha_n + \lambda \alpha_{n-1}))$; it is well-known that the polynomial R_{n-1} can be deduced from the coefficients of λ^0 and λ^1 in M_λ in quasi-linear time.

Hence, it is enough to compute $M_\lambda \bmod \lambda^2$. For this, we compute the sequence $s_\lambda \bmod \lambda^2 = (s_i + is_i^* \lambda)_{i \geq 0}$, with $s_i^* = uT_n^{i-1} T_{n-1} v$, and we apply e.g. the matrix Berlekamp Massey algorithm to this sequence, over the coefficient ring $\mathbb{K}[\lambda]/\lambda^2$; in generic coordinates, we expect that the calculation carries over as if we were over a field.

The case of radical ideals. It is of course desirable to avoid computing minimal matrix polynomials over non-reduced rings, if possible. For radical ideals, we propose an alternative solution that avoids this issue (extending this idea to arbitrary I is work in progress).

Suppose that we have computed the minimal generating polynomial S, together with its determinant (in the radical case, it coincides with $R_n(x_n)$); we can then compute the numerator matrix N of the generating series $G = \sum_{i \geq 0} s_i / x^{i+1}$ by a single product of polynomial matrices $N = SG$. We repeat this operation with the generating series $\sum_{i \geq 0} s_i^* / x^{i+1}$, for s_i^* as in the previous paragraph, to obtain a numerator matrix N^*. The Smith form of S, $D = ASB$, only has $R_n(x)$ as a non-trivial invariant factor. Then, let n and n^* be the entries of indices $(1, m)$ in respectively AN and AN^*; we can verify that $R_{n-1} = n^*/n \bmod R_n$.

As an example, we will run the algorithm on $I = \langle f_1, f_2, f_3 \rangle \subset \mathbb{F}(9001)[x_1, x_2, x_1]$,

$$\begin{aligned}
f_1 &= 3536x_3^2 + 6536x_3x_2 + 3900x_2^2 + 3722x_3x_1 + 580x_2x_1 + 7635x_1^2 + 4203x_3 + 1386x_2 + 2491x_1 + 250 \\
f_2 &= 3987x_3^2 + 3953x_3x_2 + 6122x_2^2 + 5115x_3x_1 + 7660x_2x_1 + 8669x_1^2 + 4098x_3 + 7705x_2 + 2449x_1 + 1134 \\
f_3 &= 8589x_3^2 + 1291x_3x_2 + 5321x_2^2 + 765x_3x_1 + 6052x_2x_1 + 8178x_1^2 + 2764x_3 + 957x_2 + 7079x_1 + 517.
\end{aligned}$$

We choose $m = 2$ and two random matrices

$$u = \begin{bmatrix} 291 & 22 & 337 & 924 & 414 & 666 & 574 & 707 \\ 57 & 801 & 513 & 135 & 447 & 107 & 942 & 320 \end{bmatrix}$$

$$v = \begin{bmatrix} 553 & 81 & 56 & 261 & 109 & 890 & 477 & 53 \\ 725 & 642 & 905 & 612 & 952 & 158 & 235 & 783 \end{bmatrix}^t$$

The monomial basis of Q is $\mathscr{B} = [x_1^3, x_1^2, x_2x_1, x_3x_1, x_1, x_2, x_3, 1]$, with $D = 8$. Computing the minimal generating polynomial of $s_i = uT_3^i v$, we get

$$S = \begin{bmatrix} 8226 + 5622x + 7693x^2 + 5033x^3 + x^4 & 2919 + 8706x + 3736x^2 + 4829x^3 \\ 7928 + 3675x + 7471x^2 + 3510x^3 & 3113 + 5615x + 2702x^2 + 3353x^3 + x^4 \end{bmatrix},$$

with determinant

$$R_3 = x^8 + 8386x^7 + 8262x^6 + 7301x^5 + 318x^4 + 4870x^3 + 715x^2 + 8568x + 8433.$$

We compute the numerator matrix N of the generating series $\sum_{i\geq 0} s_i/x^{i+1}$ and multiply it by the change of basis of A obtained from the Smith form computation of S; the $(1,2)$-entry in the result is

$$n = 5527x^7 + 2064x^6 + 4391x^5 + 1308x^4 + 6797x^3 + 1328x^2 + 6317x + 8700.$$

Computing the numerator matrix N^* of the generating series $\sum_{i\geq 0} s'_i/x^{i+1}$, with $s'_i = uT_3^{i-1}T_2v$, and doing as above, we obtain

$$n^* = 8649x^7 + 3840x^6 + 4938x^5 + 1734x^4 + 1525x^3 + 2362x^2 + 1780x + 2397.$$

The polynomial $R_2(x_3)$ is then given by $n^*/n \bmod R_3$, written in the variable x_3, and is equal to

$$R_2 = 2287x_3^7 + 8269x_3^6 + 8475x_3^5 + 6889x_3^4 + 4785x_3^3 + 3960x_3^2 + 3902x_3 + 4153.$$

Doing the same with the sequence $uT_3^{i-1}T_1v$ gives us a polynomial $R_1(x_3)$, which finally leads us to the Gröbner basis $(x_1 - R_1(x_3), x_2 - R_2(x_3), R_3(x_3))$ of I.

References

[1] 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.

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

[3] 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.

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

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

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

[8] F. Rouillier. Solving zero-dimensional systems through the rational univariate representation. *Applicable Algebra in Engineering, Communication and Computing*, 9(5):433–461, 1999.