On Implementing Signature-based Gröbner Basis Algorithms Using Linear Algebraic Routines from M4RI

Yao Sun, Dongdai Lin
SKLOIS, Institute of Information Engineering, CAS, (China)

Dingkang Wang
KLMM, Academy of Mathematics and Systems Science, CAS, (China)

dwang@mmrc.iss.ac.cn

Abstract

Gröbner bases, proposed by Buchberger in 1965 [5], have been proven to be very useful in many aspects of algebra. Faugère introduced the concept of signatures for polynomials and presented the famous F5 algorithm [9]. Since then, signature-based algorithms have been widely investigated, and several variants of F5 have been presented, including F5C [7], extended F5 [13], F5 with revised criterion [4], and RB [8]. Gao et al. proposed another signature based algorithm G2V [11] in a different way from F5, and GVW[12] is an extended version of G2V. The authors also studied generalized criteria and signature-based algorithms in solvable polynomial algebra in [15, 16].

For implementations of signature-based algorithms, Roune and Stillman efficiently implemented GVW and AP without using linear algebra [14]. Faugère mentioned a matrix F5 in [10]. An F5 algorithm in F4 style was described in more detail by Albrecht and Perry [1].

In signature-based algorithms, each polynomial is assigned a signature, and polynomials can only be reduced by polynomials with smaller signatures. When implementing signature-based algorithms using linear algebra, such as [1], rows of the constructed matrices are also assigned with signatures. These matrices can only be eliminated from one side due to the constraints of signatures, i.e. rows can only be reduced by rows with smaller signatures. However, most public libraries on linear algebra do not provide routines for such one-side elimination, such that most public libraries cannot be applied to the implementations of signature-based algorithms directly.

In this talk, we present a method of rotating rows during the elimination, to ensure rows with larger signatures can always be reduced by rows with smaller signatures. Our method only needs to revise the swapping procedure during the elimination, and can be easily applied to most public libraries on linear algebra. We have applied our method to the M4RI package [3], and implemented the GVW algorithm by using the modified routines over the finite field $GF(2)$. Due to the efficient routines modified from M4RI, our implemented GVW algorithm is more efficient than some of Gröbner basis implementations on public softwares.

Our method of rotating rows can be illustrated by the following example. Let A be a matrix with entries in $\mathbb{F}_2$. Assume A has the following form:

$$\begin{matrix} S_1 \\ S_2 \\ S_3 \\ S_4 \\ S_5 \\ S_6 \end{matrix} \quad \begin{pmatrix} 0 & 1 & * & * & * & * \\ 0 & 0 & 0 & 1 & * & * \\ 0 & 0 & 1 & * & * & * \\ 0 & 1 & * & * & * & * \\ 1 & * & * & * & * & * \\ 1 & * & * & * & * & * \end{pmatrix},$$

where "$*$" may be 1 or 0, S_i is the signature of each row, and we assume $S_1 \prec_s S_2 \prec_s \cdots \prec_s S_6$.

To reduce A to row-echelon form, we first find the pivot entry in the first column. *We must search the pivot entry from top to bottom (i.e. from lower signatures to higher signatures).* Then we find the entry at row 5 and col 1 is a pivot. If we use general methods of elimination, we need to swap row 1 and row 5 directly, and clear entries at column 1 by the row with signature S_5. Next, when doing elimination in the second column, the row with signature S_4 is selected as pivot row, and needs to eliminate other rows. However, this will leads to errors in signature-based algorithms, because the row with signature S_1 has a smaller signature than

S_4 and cannot be eliminated by the row with signature S_4. So we cannot swap row 1 and row 5 directly.

To make further eliminations correct, we swap row 1 and row 5 in a special manner. First, we pick up the row 5 with signature S_5. Second, we move rows 4, 3, 2, and 1 to rows 5, 4, 3, and 2 respectively. At last, we put the row with signature S_5 at row 1. After this swap, matrix A becomes the following form.

$$\begin{matrix} S_5 \\ S_1 \\ S_2 \\ S_3 \\ S_4 \\ S_6 \end{matrix} \quad \begin{pmatrix} 1 & * & * & * & * & * \\ 0 & 1 & * & * & * & * \\ 0 & 0 & 0 & 1 & * & * \\ 0 & 0 & 1 & * & * & * \\ 0 & 1 & * & * & * & * \\ 1 & * & * & * & * & * \end{pmatrix}.$$

Next, we use the row with S_5 to clear all entries at column 1 below this row, and then column 1 is done. For column 2, we find pivots from rows with $S_1, ..., S_4$ and S_6, and repeat the above processes. Elimination terminates when the matrix becomes an upper triangular form.

This swap makes eliminations correct in signature-based algorithm for the following reasons. On one hand, since pivot rows (e.g. row of S_5) are finding from low signatures to high signatures, rows with smaller signature (e.g. rows of $S_1, \ldots, S_4$) cannot be reduced by pivot rows (e.g. row of S_5). On the other hand, after swaps, rows below pivot rows (e.g. rows of $S_1, \ldots, S_4$ and S_6) are still in an increasing order on signatures.

Using this special swap, the echelon form of A is in an upper triangular form, such that divide-and-conquer methods of PLE decomposition [2] can be used, and hence, the eliminations can be speeded up significantly.

In our implementation, we modify many subroutines of *mzd_ple*() in M4RI library to use this swap. The new function with the special swap is called *gvw_ple*(). We compare the efficiency of *mzd_ple*() and *gvw_ple*() in Table 1. Examples with density $\approx$ 50% are generated randomly by routines from M4RI. Since the densities of matrices in Gröbner basis computations are usually very small, we also generate some randomized matrices with density $\approx$ 3%. The first column in Table 1 is the size of matrices, and the timings in this table are given by seconds. All matrices are generated over $GF(2)$.

Tests	density ≈ 50%		density ≈ 3%	
	mzd_ple()	gvw_ple()	mzd_ple()	gvw_ple()
$10,000 \times 10,000$	0.378	0.382	0.345	0.354
$10,000 \times 30,000$	1.342	1.301	1.268	1.262
$30,000 \times 10,000$	1.432	1.443	1.403	1.418
$30,000 \times 30,000$	7.661	7.655	7.604	7.577
$30,000 \times 60,000$	18.684	18.671	18.651	18.634
$60,000 \times 30,000$	19.396	19.296	19.282	19.298
$60,000 \times 60,000$	58.373	58.636	54.509	54.263
$60,000 \times 100,000$	123.321	123.298	119.479	122.523
$100,000 \times 60,000$	119.991	118.388	108.565	108.501
$100,000 \times 100,000$	266.817	267.191	237.401	237.560
$150,000 \times 150,000$	817.682	817.750	700.032	700.781

Table 1: mzd_ple() vs gvw_ple()

From the above table, we can see the function *mzd_ple*() and *gvw_ple*() almost have the same efficiency.

In Table 2, we compare our implementation of GVW with some intrinsic implementations on public softwares, including Gröbner basis functions on Maple (version 17, setting "method = fgb"), Singular (version 3-1-6), and Magma (version 2.12-16)[1], and the computing times in seconds are listed. In the column of Exam., $n \times n$ means that the input polynomial system has n polynomials with n variables. These square polynomial systems were generated by Courtois in [6]. The Computer we used is MacBook Pro with 2.6 GHz Intel Core i7, 16 GB memory.

[1] Magma 2.12-16 is an old version.

Exam.	Maple	Singular	Magma	GVW
16×16	4.088	5.210	0.484	0.560
17×17	9.891	12.886	0.874	0.893
18×18	22.340	31.590	1.513	1.556
19×19	48.314	84.771	2.792	2.742
20×20	107.064	265.325	5.226	4.676
21×21	218.479	724.886	10.468	14.991
22×22	839.067	$> 1h$	37.144	28.947

Table 2: Maple, Singular and Magma vs M-GVW

From the above table, we can see that, due to the efficiency of routines from M4RI, our implementation of M-GVW is more efficient than some of functions from existing public softwares. However, since the matrices in large polynomial systems become quite sparse, our implementation may not perform very good for large systems at present.

Keywords
Gröbner basis, linear algebra, implementation.

References

[1] M. Albrecht and J. Perry. F4/5. Preprint, arXiv:1006.4933v2 [math.AC], 2010.

[2] M. Albrecht and C. Pernet. Efficient decomposition of dense matrices over GF(2). Arxiv.org: 1006.1744, 2011.

[3] M. Albrecht and G. Bard. The M4RI library – Version 20130416. 2013. `http://m4ri.sagemath.org` .

[4] A. Arri and J. Perry. The F5 criterion revised. J. Symb. Comput. vol 46, 1017-1029, 2011.

[5] B. Buchberger. Ein Algorithmus zum auffinden der Basiselemente des Restklassenringes nach einem nulldimensionalen Polynomideal. PhD thesis, 1965.

[6] N. Courtois. Benchmarking algebraic, logical and constraint solvers and study of selected hard problems, 2013. `http://www.cryptosystem.net/aes/hardproblems.html`.

[7] C. Eder and J. Perry. F5C: a variant of Faugère's F5 algorithm with reduced Gröbner bases. J. Symb. Comput., vol. 45(12), 1442-1458, 2010.

[8] C. Eder and B.H. Roune. Signature rewriting in Gröbner basis computation. In proc. ISSAC'13, ACM Press, New York, USA, 331-338, 2013.

[9] J.-C. Faugère. A new efficient algorithm for computing Gröbner bases without reduction to zero (F_5). In proc. ISSAC'02, ACM Press, New York, USA, 75-82, 2002.

[10] J.-C. Faugère and S. Rahmany. Solving systems of polynomial equations with symmetries using SAGBI-Gröbner bases. In proc. ISSAC '09, ACM Press, New York, USA, 151-158, 2009.

[11] S.H. Gao, Y.H. Guan, and F. Volny. A new incremental algorithm for computing Gröbner bases. In proc. ISSAC'10, ACM Press, New York, USA, 13-19, 2010.

[12] S.H. Gao, F. Volny, and M.S. Wang. A new algorithm for computing Gröbner bases. Cryptology ePrint Archive, Report 2010/641, 2010. Latest version is given in July, 2013, and downloaded from `http://www.math.clemson.edu/~sgao/pub.html`.

[13] A. Hashemi and G. Ars. Extended F5 criteria. J. Symb. Comput., vol. 45(12), 1330-1340, 2010.

[14] B.H. Roune and M. Stillman. Practical Gröbner basis computation. In proc. ISSAC'12, ACM Press, 2012.

[15] Y. Sun and D.K. Wang. A generalized criterion for signature related Gröbner basis algorithms. In Proc. ISSAC'11, ACM Press, 337-344, 2011.

[16] Y. Sun, D.K. Wang, D.X. Ma, and Y. Zhang. A signature-based algorithm for computing Gröbner bases in solvable polynomial algebras. In Proc. ISSAC'12, ACM Press, 351-358, 2012.