数理解析研究所講究録
第 646 巻 1988 年 7-18

A Modular Gröbner Basis Method for Algebraic Equations

Tateaki Sasaki† & Taku Takeshima‡

†The Institute of Physical and Chemical Research
2-1, Hirosawa, Wako-shi, Saitama 351-01, Japan
‡Fujitsu Limited
17-25, Shinkamata 1-chome, Ota-ku, Tokyo 144, Japan

Abstract

A modular Gröbner basis method for solving systems of algebraic equations is described. Given equations with integer coefficients, the method calculates Gröbner basis over $\mathbf{Z}/(p_i), i = 1, \ldots, k$, where $p_1, \ldots, p_k$ are distinct primes, then it converts them to a Gröbner basis over $\mathbf{Q}$. By this method, we can perform the reduction of equations efficiently by avoiding intermediate coefficient growth. Furthermore, a device to avoid unnecessary S-polynomial construction is presented.

§1. Introduction

After Buchberger's invention of Gröbner basis construction algorithm [1], a drastic progress has been accomplished for solving systems of algebraic equations [2∼5]. Compared with the conventional resultant method, Gröbner basis methods suppress the intermediate expression growth strongly. However, computation of Gröbner basis is still time-consuming because of the following two points: one is the intermediate coefficient growth and the other is the construction of S-polynomials which are reduced to zero immediately. When calculating Gröbner basis actually, we readily find very large-sized numbers in the basis polynomials and even larger numbers in the intermediate polynomials. Furthermore, we can find that considerable part of computation time is spent for the construction of S-polynomials which are reduced to zero immediately. In fact, when solving a large-sized system of equations, the computation time is mostly spent for the coefficient calculation and the construction of zero-reduced S-polynomials.

When the coefficient domain is **Z** (integer ring) or **Q** (rational number field), the coefficient growth problem can be solved by utilizing modular arithmetic. In the case of Gröbner basis construction, Trinks [6] presented a p-adic construction method in the context of solving algebraic equations, and Winkler [7] investigated the utilization of Hensel construction. In this paper, we investigate a simpler modular method, *i.e.*, the utilization of Chinese remainder algorithm.

In §**2**, we describe a modular Gröbner basis algorithm grossly. The algorithm is elaborated in §**3**, and several examples with timing data are presented in §**4**.

§2. Gross description of algorithm

We first define several notations. In the following, we consider that the polynomials are in $K[x_1, \ldots, x_n]$, with K a number field.

Term order $\triangleright$. Let $T_i = c_i x_1^{e_{i1}} \cdots x_n^{e_{in}}$ be a monomial in $K[x_1, \ldots, x_n]$, where $c_i \in K$. By using the n-tuple $(e_{i1}, \ldots, e_{in})$, we can order the monomials in $K[x_1, \ldots, x_n]$ uniquely, and we denote the order by $\triangleright$. The order $\triangleright$ is such that if $T_1/c_1 \neq T_2/c_2$ then either $T_1 \triangleright T_2$ or $T_2 \triangleright T_1$ and if $T_1 \triangleright T_2$ and $T_2 \triangleright T_3$ then $T_1 \triangleright T_2$, for any T_1, T_2 and T_3.

Head term and head coefficient (abbreviated to *ht* and *hc*, respectively.) Let P be a polynomial. The highest order monomial, with respect to $\triangleright$, of P is called the head term of P and written as $ht(P)$. Let $ht(P) = cx_1^{e_1} \cdots x_n^{e_n}$, with $c \in K$, then c is called the head coefficient of P and written as $hc(P)$.

S-polynomial (abbreviated to *Spol.*) Given polynomials P_1 and P_2, S-polynomial of P_1 and P_2 is defined by

$$Spol(P_1, P_2) = \frac{lcm}{ht(P_1)} P_1 - \frac{lcm}{ht(P_2)} P_2, \tag{1}$$

where $lcm = LCM(ht(P_1), ht(P_2))$.

We also use the terms ***M-reduction*** and ***Gröbner basis***. For these terms and the Gröbner basis construction algorithm, see [1].

Consider solving a system of algebraic equations

$$\{F_1(x_1, \ldots, x_n) = 0, \ldots, F_r(x_1, \ldots, x_n) = 0\}, \tag{2}$$

where $F_i \in \mathbf{Z}[x_1, \ldots, x_n], i = 1, \ldots, r$. Assuming that the ideal $(F_1, \ldots, F_r)$ is zero-dimensional, *i.e.*, the system (2) has finite solutions, we want to reduce (2) to the following form

$$\{G_1(x_1) = 0, G_2(x_1, x_2) = 0, \ldots, G_n(x_1, \ldots, x_n) = 0\}, \tag{3}$$

where $G_i \in \mathbf{Q}[x_1, \ldots, x_i], i = 1, \ldots, n$, and all the roots of (2) are given by the roots of (3). Note that, when (2) has no multiple roots, then (3) often has the following form.

$$\{G_1(x_1) = 0, x_2 - \widetilde{G}_2(x_1) = 0, \ldots, x_n - \widetilde{G}_n(x_1) = 0\}. \tag{3'}$$

As is well-known, the system (3) is the reduced Gröbner basis of the ideal $(F_1, \ldots, F_r)$ with the lexicographic term-order

$$x_n \triangleright \ldots \triangleright x_2 \triangleright x_1. \tag{4}$$

Modular construction of Gröbner basis is quite simple in principle. Below, we give the algorithm in a gross form.

Modular Gröbner basis construction [in a gross form].

Input: a set of polynomials $\{F_1, \ldots, F_r\} \in \mathbf{Z}[x_1, \ldots, x_n]$;

a set of distinct primes $\{p_1, p_2, \ldots\}$;

Output: a Gröbner basis $\Gamma = \{G_1, \ldots, G_s\}$ of $(F_1, \ldots, F_r)$;

Step 1 [Gröbner basis in $\mathbf{Z}/(p_i)[x_1, \ldots, x_n], i = 1, \ldots, k$]: For sufficiently many primes $p_1, \ldots, p_k$, calculate a reduced Gröbner basis

$\Gamma^{(i)} = \{G_1^{(i)}, \ldots, G_s^{(i)}\}$ in $\mathbf{Z}/(p_i)[x_1, \ldots, x_n]$, $i = 1, \ldots, k$;

Step 2 [Gröbner basis in $\mathbf{Z}/(p_1 \cdots p_k)[x_1, \ldots, x_n]$]: By applying the Chinese remainder algorithm, construct a Gröbner basis $\Gamma^{(0)} = \{G_1^{(0)}, \ldots, G_s^{(0)}\}$ such that

$\Gamma^{(0)} \equiv \Gamma^{(i)}$ $(mod\ p_i)$, $i = 1, \ldots, k$;

Step 3 [Gröbner basis in $\mathbf{Q}/(p_1 \ldots p_k)[x_1, \ldots, x_n]$]: Convert the integer coefficients in $\Gamma^{(0)}$ in such a way that an integer c is converted to a rational a/b satisfying

$c \equiv a/b$ $(mod\ p_1 \cdots p_k)$ and $|a|, |b| < \sqrt{p_1 \cdots p_k/2}$;

Step 4 : Check that the basis constructed in **Step 3** is actually the reduced Gröbner basis in $\mathbf{Q}[x_1, \ldots, x_n]$. □

It is noted that every head coefficient of S-polynomial constructed through this algorithm should be normalized to 1 in order to recover true Gröbner basis over $q[x_1, \ldots, x_n]$.

§3. Detailed description of the algorithm

Now, we elaborate the algorithm presented grossly in §2.

3.1. The number of primes

We denote the Gröbner basis constructed in the above **Step 3**, *i.e.*, a Gröbner basis in $\mathbf{Q}/(p_1 \cdots p_k)[x_1, \ldots, x_n]$, by $\Gamma_{(p_1 \cdots p_k)}$. We note that we have no method of knowing the value of k, the number of necessary primes, in advance of the computation, and we must estimate k by checking the Gröbner basis constructed. The estimation is done as follows: if $\Gamma_{(p_1 \cdots p_k)} = \Gamma_{(p_1 \cdots p_k p_{k+1})}$ then k is a desired value. (This does not mean that k is a sufficient value.) Therefore, we calculate $\Gamma_{(p_1 \cdots p_i)}$ after the construction of each $\Gamma^{(i)}$.

3.2. Integer interpolation

There are two algorithms for performing the above **Step 2**: the Newton interpolation formula and the Lagrange interpolation formula. Since we calculate interpolations after the construction of each basis $\Gamma^{(i)}$, the Newton algorithm is apparently suited for our computation. The algorithm, in the form of being used iteratively, is as follows.

Newton interpolation algorithm [to be used iteratively.]

```
Input:  a set of distinct primes $(p_1, \ldots, p_{i-1}, p_i)$;
        a set of interpolation coefficients $(v_1, \ldots, v_{i-1})$;
        an $(i-1)$st interpolation $u$ and the $i$-th residue $u_i$;
Output: the $i$-th interpolation $u$ s.t. $u \equiv u_j \ (mod\ p_j), j = 1, \ldots, i$,
        and the set of interpolation coefficients $(v_1, \ldots, v_{i-1}, v_i)$;
  (I1) if $i = 1$ then $u := v_1 := u_1$; return $u$ and $(v_1)$;
  (I2) Calculate integers $w_j, j = 1, \ldots, i-1$, s.t. $w_j p_j \equiv 1 \ (mod\ p_i)$;
  (I3) Calculate $v_i$ as $v_i = (\cdots((u_i - v_1)w_1 - v_2)w_2 - \cdots - v_{i-1})w_{i-1} \ (mod\ p_i)$;
  (I4) return $u := u + v_i(p_1 \cdots p_{i-1})$ and $(v_1, \ldots, v_{i-1}, v_i)$.   □
```

3.3. Conversion to rationals

The conversion from integer to rational number modulo $p_1 \cdots p_k$ is based on the following well-known theorem:

Theorem [well-known]. *Let $m, A, B \in \mathbf{N}$ satisfying $2AB < m$. For any $u \in \mathbf{Z}$, there exists at most one rational number a/b in the range $-A \le a \le A$ and $1 \le b \le B$, satisfying*

$$bu \equiv a \ (mod\ m), GCD(a, b) = 1. \tag{5}$$

We use the theorem by setting $m = p_1 \cdots p_k$ and $A = B = [\sqrt{m/2}]$.

Given positive integers $m (= p_1 \cdots p_k)$ and u, the following algorithm calculate a rational number a/b s.t. $a/b \equiv u \ (mod\ m)$.

Algorithm CONV:INT2RAT.

```
Input: $u$, modulus $m$, and $sqm = \sqrt{m/2}$, where $0 < u < m$;
Output: integers $a$ and $b$ s.t. $a/b \equiv u \ (mod\ m)$, $-sqm < a < sqm, 0 < b < sqm$;
```

```
(R1) $\vec{a} := (1, 0, m); \vec{b} := (0, 1, u);$
(R2) **while** $b_3 \geq sqm$ **do**
         $\{ q := a_3/b_3; \vec{r} := \vec{a} - q\vec{b}; \vec{a} := \vec{b}; \vec{b} := \vec{r} \};$
     **if** $b_3 = 0$ **then return** NIL;
(R3) **if** $|b_3| < sqm$ **then return** $(a := sign(b_2)b_3, b := |b_2|)$
         **else** $\vec{b} := \vec{b} + \vec{a};$
     **if** $b_3 < sqm$ **then goto** (R3)
         **else return** NIL.
```
□

In the above algorithm, the following relations hold always.

$$a_1 m + a_2 u = a_3, b_1 m + b_2 u = b_3.$$

In the **while** loop of (R2), the values of a_3 and b_3 decrease steadily and the values of $|a_2|$ and $|b_2|$ increase steadily. After the **while** loop. we have $b_3 < \sqrt{m/2}$ and if $|b_2| < \sqrt{m/2}$ then we find the desired rational. if $|b_2| \geq \sqrt{m/2}$ then the value of b_3 has been decreased too much. So, we increase the value of b_3 in (R3).

Note 1. *If NIL is returned by CONV:INT2RAT then it means that there is no rational a/b satisfying (5) for modulus m. Such a case may happen when $GCD(m, u) \neq 1$. For a given rational a/b, however, if $ub \equiv a \pmod{m}$ then we can recover a/b from u so far as m is sufficiently large.*

Note 2. *The iteration search in (R3) step can be performed efficiently if we utilize a binary splitting of the quotient q.*

3.4. Unlucky primes

If a rational a/b is such that p_i divides b then there is no image of a/b in $\mathbf{Z}/(p_1 \cdots p_i \cdots p_k)$. Therefore, if a numeric coefficient a/b in a Gröbner basis over $\mathbf{Q}$ is such that p_i divides b then the prime p_i is not adequate as a modulus. We call such a prime *unlucky*. Since initial polynomials are in $\mathbf{Z}[x_1, \ldots, x_n]$, the denominators of the coefficients in the Gröbner basis are only factors of products of head coefficients of S-polynomials constructed (*cf.* Eq.(1). The M-reduction does not introduce new denominator factors.) Hence, if the prime p_i does not divide the head coefficient of *every* S-polynomial constructed, then it is an adequate modulus. We use word-size primes as $p_1, p_2, \ldots$, hence we encounter unlucky primes very rarely.

3.5. History of basis construction process

In the actual computation of a Gröbner basis over $\mathbf{Z}/(p)$, we cannot see whether the head term of an S-polynomial vanishes, so long as we compute the basis independently from other basis over $\mathbf{Z}/(p'), p' \neq p$. Therefore, we preserve the history of basis construction process. The history is the following list:

$$\begin{aligned}\text{HIST:=}((\#11\ \#12\ &ht(Spol(F_{\#11}, F_{\#12}))),\\ (\#21\ \#22\ &ht(Spol(F_{\#21}, F_{\#22}))),\\ \cdots\cdots\cdots\cdots\cdots &\quad).\end{aligned} \tag{6}$$

Here, $Spol(F_{\#11}, F_{\#12})$ is a non-zero S-polynomial constructed first, $Spol(F_{\#21}, F_{\#22})$ is a non-zero S-polynomial constructed second, and so on.

The HIST is used as follows. We initialize HIST by the Gröbner basis construction for the first prime p_1. Next, consider the basis construction for the i-th prime p_i, and suppose we calculate $Sp^{(i,j)} = Spol(F_{\#j1}, F_{\#j2})$, which is a non-zero S-polynomial constructed j-th. Let $Sp^{(*,j)} = Spol(F_{\#j1}, F_{\#j2})$ saved in HIST. If $ht(Sp^{(i,j)}) \lhd ht(Sp^{(*,j)})$ then p_i is an unlucky prime hence we discard the p_i. If $ht(Sp^{(i,j)}) \rhd ht(Sp^{(*,j)})$ then all the primes $p_1, \ldots, p_{i-1}$ are unlucky and we initialize HIST by the basis construction for the prime p_i.

3.6. Avoiding zero-reduced S-polynomial construction

As we have mentioned in §1, in the construction of Gröbner basis of many elements, most computation time is spent to construct S-polynomials which are reduced to zero immediately. Using the HIST defined above, we can avoid such zero-reduced S-polynomial construction for many primes, reducing the computation time largely.

The method is as follows. We construct HIST by the basis construction process for the first several primes, say p_1, p_2 and p_3. The HIST thus constructed will be almost valid and, for the rest primes ($p_4, p_5, \ldots$, in this case), we construct only the S-polynomials registered in HIST sequentially.

3.7. Correctness check in $\mathbf{Q}[x_1, \ldots, x_n]$

The correctness check of the basis constructed (*i.e.*, **Step 4** of the gross algorithm in §2) is performed by showing that $Spol(G_i, G_j)$ is M-reduced to 0 by $\Gamma_{(p_1 \cdots p_k)}$ for any pair (G_i, G_j) in $\Gamma_{(p_1 \cdots p_k)}$, and that each F_i in (2) is M-reduced to 0 by $\Gamma_{(p_1 \cdots p_k)}$. This check is not always done quickly because we must handle the large-sized coefficients exactly. Fortunately, the Gröbner basis we are calculating is of the form (3') or similar to (3'), and the basis elements are quite suited for performing the check quickly. For example, if $GCD(ht(G_i), ht(G_j)) = 1$ (or a number) then we may skip the check for $Spol(G_i, G_j)$.

3.8. On the term order $\triangleright$

So far, the term-order $\triangleright$ is assumed to be the lexicographic order. If, however, the reduced Gröbner basis is of the form (3'), we can choose another term-order to perform the basis construction efficiently (see [8].) The order is as follows:

$$\begin{cases} \text{total-degree order for } \; x_2, \ldots, x_n, \\ \qquad\qquad\qquad\qquad x_n, \ldots, x_2 \triangleright x_1 \; . \end{cases}$$

In the actual implementation of the modular Gröbner basis method, we had better test this order first. If we find that the basis is not of the form (3') then we apply the lexicographic order.

3.9. Algorithm

Summarizing the above discussions, we have the following algorithm.

Modular Gröbner basis construction [in a elaborated form].

Input: a set of polynomials $\mathcal{F} = \{F_1, \ldots, F_r\} \in \mathbf{Z}[x_1, \ldots, x_n]$;

a set of distinct primes $\mathcal{P}$;

Output: a Gröbner basis $\Gamma = \{G_1, \ldots, G_s\}$ of $(F_1, \ldots, F_r)$;

```
(T1) take out a prime from $\mathcal{P}$ and set it to $p_1$;
(T2) calculate a reduced Gröbner basis over $\mathbf{Z}/(p_1)$ and set it to $\Gamma^{(1)}$ and $\Gamma^{(0)}$,
     and make a HIST newly;
(T3) convert coefficients of $\Gamma^{(0)}$ to rationals by algorithm CONV:INT2RAT
     and set the result to $\Gamma_{(p_1)}$;
(T4) $i$:=2;
(T5) take out a prime from $\mathcal{P}$ and set it to $p_i$;
(T6) calculate a reduced Gröbner basis over $\mathbf{Z}/(p_i)$ and set it to $\Gamma^{(i)}$
     comparing calculated S-polynomials with HIST,
     when comparing S-polynomials with HIST,
         if $ht(Sp^{(i,j)}) \lhd (Sp^{(*,j)})$
           then { quit calculation of $\Gamma^{(i)}$; goto (T5)};
         if $ht(Sp^{(i,j)}) \rhd (Sp^{(*,j)})$
           then {continue calculation of $\Gamma^{(i)}$; renew HIST henceforce;
                 $\Gamma^{(1)}$:=$\Gamma^{(i)}$; $p_1$:=$p_i$; $i$:=2; goto (T5)};
(T7) construct new $\Gamma^{(0)}$ from $\Gamma^{(i)}$ and old $\Gamma^{(0)}$
     by Newton interpolation algorithm;
(T8) convert coefficients of $\Gamma^{(0)}$ to rationals by algorithm CONV:INT2RAT
     and set the result to $\Gamma_{(p_1 \cdots p_i)}$;
(T9) if $\Gamma_{(p_1 \cdots p_i)} \neq \Gamma_{(p_1 \cdots p_{i-1})}$ then { $i$:=$i+1$; goto (T5)};
     if $\Gamma_{(p_1 \cdots p_i)}$ is really a Gröbner basis over $\mathbf{Q}[x_1, \ldots, x_n]$
       then return( $\Gamma_{(p_1 \cdots p_i)}$ )
       else { $i$:=$i+1$; goto (T5)};
```

§4. Empirical study

We have inplemented the above-mentioned algorithm on the algebra system GAL and studied the effectiveness of the algorithm by the following three examples.

Example 1. (Klein's equation)

$$P_1 = (x_1^6 + x_2^6) + 522(x_1^5 x_2 - x_1 x_2^5) - 10005(x_1^4 x_2^2 + x_1^2 x2^4) - u_1 = 0,$$
$$P_2 = -(x_1^4 + x_2^4) + 228(x_1^3 x_2 - x_1 x_2^3) - 494 x_1^2 x_2^2 - u_2 = 0,$$
$$P_3 = x_1 x_2 (x_1^2 + 11 x_1 x_2 - x_2^2)^5 - u_3 = 0.$$

We calculate the reduced Gröbner basis of (P_1, P_2, P_3) with the ordering $x_1, x_2 \triangleright u_1, u_2, u_3$, where total-degree order is assumed for $\{x_1, x_2\}$ and $\{u_1, u_2, u_3\}$, respectively.

Example 2. (Katsura's equation #3)

$$P_1 = 2(x_4^2 + x_3^2 + x_2^2) + x_1^2 - x_1 = 0,$$
$$P_2 = 2(x_4 x_3 + x_3 x_2 + x_2 x_1) - x_2 = 0,$$
$$P_3 = 2(x_4 x_2 + x_3 x_1) + x_2^2 - x_3 = 0,$$
$$P_4 = 2(x_4 + x_3 + x_2) + x_1 - 1 = 0.$$

We calculate the reduced Gröbner basis of (P_1, P_2, P_3, P_4) with the ordering $x_4, x_3, x_2 \triangleright x_1$, where the total-degree order is assumed for $\{x_2, x_3, x_4\}$. The result is of form $(3')$.

Example 3. (Katsura's equation #4)

$$P_1 = 2(x_5^2 + x_4^2 + x_3^2 + x_2^2) + x_1^2 - x_1 = 0,$$
$$P_2 = 2(x_5 x_4 + x_4 x_3 + x_3 x_2 + x_2 x_1) - x_2 = 0,$$
$$P_3 = 2(x_5 x_3 + x_4 x_2 + x_3 x_1) + x_2^2 - x_3 = 0,$$
$$P_4 = 2(x_5 x_2 + x_4 x_1 + x_3 x_2) - x_4 = 0,$$
$$P_5 = 2(x_5 + x_4 + x_3 + x_2) + x_1 - 1 = 0.$$

We calculate the reduced Gröbner basis of $(P_1, \ldots, P_5)$ similarly as Example 2. The result is of form $(3')$.

Table 1 shows a comparison of three algorithms: algorithm C is the conventional one based on the rational arithmetic; M1 and M2 are modular algorithms, where M1 does not utilize the HIST while M2 does. We used primes of order 10^6, and we have encountered no unlucky primes in our test.

The number for k in Table 1 shows the size of coefficients (rationals in this case) of the

Table 1. Comparison of modular algorithm (M1 & M2) and conventional algorithm(C). Timing data are in mili-seconds on a FACOM-780 computer. (k is the number of primes used in each computation.)

No.	*Algorithm C* (conventional)	*Algorithm M1* (HIST=nil)	*Algorithm M2* (use HIST)
Example 1	221	591 (k=5)	389 (k=5)
Example 2	347	385 (k=5)	386 (k=5)
Example 3	>600,000	28,593 (k=16)	28,709 (k=16)

basis polynomials obtained. Example 1 is a "small-sized" problem for which the intermediate coefficient growth is quite weak, and the modular method is not effective for this case. Example 2 causes a "weak" intermediate coefficient growth, hence the modular method is not bad compared with the conventional method although the Gröbner basis is calculated for five distinct primes. Example 3 causes a "strong" intermediate coefficient growth, and we find the modular method is actually quite effective for such problems. Furthermore, comparison of algorithms M1 and M2 indicates that the most time-consuming part of our algorithm is not the calculation of many zero-reduced polynomials over $\mathbf{Z}/(p)$ but the arithmetic of long numbers. Hence, any improvement for reducing the size of long numbers handled is obviously desirable. Our current program is not tuned up yet and we promise particular improvement shall speed up performance by about three times faster than current timing data show.

references

[1] Buchberger, B., An Algorithm for Finding a Basis for the Residue Class Ring of a Zero-dimensional Polynomial Ideal (German.) Ph.D. Thesis, Math. Inst., Univ. of Insbruck, Austria (1965).

[2] Buchberger, B., Ein algorithmisches Kriterium für die Lösbarkeit eines algebraischen Gleichungssystems, Aequationes Mathematicae **4** (1970), pp.374-383.

[3] Trinks, W. L., Über Buchbergers Verfahren, Systeme algebraischer Gleichungen zu lösen, J. Number Theory **10** (1978), pp.475-488.

[4] Böge, W., Gebauer, R., Kredel, H., Some Examples for Solving Systems of Algebraic Equations by Calculating Gröbner Bases, J. Symb. Comp. **2** (1986), pp.83-98.

[5] Kobayashi, H., Moritsugu, S., Hogan, R. W., On Solving Systems of Algebraic Equations, preprint (Nov. 1987), submitted to JSC.

[6] Trinks, W., On Improving Approximate Results of Buchberger's Algorithm by Newton's Method, SIGSAM Bulletin **18** (1984), pp.7-11.

[7] Winkler, F., p-adic Methods for the Computation of Gröbner Bases, paper presented at EUROCAL'87, June 1987.

[8] Sasaki, T., Some Algebraic Algorithms based on Head Term Elimination over Polynomial Rings, papaer presented at EUROCAL'87, June 1987.